aws

Panduan Pengembang V2

Amazon Lex



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

Amazon Lex: Panduan Pengembang V2

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

Merek dagang dan tampilan dagang Amazon tidak boleh digunakan sehubungan dengan produk atau layanan apa pun yang bukan milik Amazon, dengan cara apa pun yang dapat menyebabkan kebingungan di antara pelanggan, atau dengan cara apa pun yang merendahkan atau mendiskreditkan Amazon. Semua merek dagang lain yang tidak dimiliki oleh Amazon merupakan hak milik masing-masing pemiliknya, yang mungkin atau mungkin tidak terafiliasi, terkait dengan, atau disponsori oleh Amazon.

Table of Contents

Apa itu Amazon Lex V2? 1
Kasus penggunaan umum untuk Amazon Lex V2 3
Membayar Amazon Lex V2 4
Apakah Anda First-time Pengguna Amazon Lex V2? 4
Bekerja dengan AWS SDK 4
Menggunakan Amazon Lex V2 dengan AWS SDK 5
Fitur terbaru 8
AMAZON.BedrockAgentIntent untuk terhubung dengan Agen Batuan Dasar dan Pangkalan Pengetahuan 8
NLU yang dibantu untuk meningkatkan klasifikasi niat dan resolusi slot 8
Dukungan kosakata khusus untuk 17 bahasa tambahan 8
QinConnect maksud bawaan untuk integrasi Amazon Connect 8
AMAZON.Currency dan slot AMAZON.Confirmation bawaan sekarang mendukung semua lokal 9
Niat QnA yang ditingkatkan dengan Basis Pengetahuan Batuan Dasar dan Pagar Pembatas 9
Support untuk model Anthropic Claude 3 9
Ketahanan global untuk replikasi bot di seluruh wilayah 9
Dukungan regional untuk AWS GovCloud (US-West) 10
Fitur AI generatif untuk Amazon Lex V2 10
AMAZON.Confirmation slot bawaan untuk disambiguasi yang Yes/No/Maybe/Don tidak diketahui. 10
Mengukur kinerja bisnis dengan Analytics 11
Mengevaluasi kinerja bot dengan meja kerja Uji 11
Templat bot spesifik vertikal 11
Jaringan bot 11
Visual conversation builder 12
Jenis slot komposit 12
Pencabangan kondisional 12
Perancang chatbot otomatis 12
Petunjuk runtime 13
Kosakata kustom 13
Jenis slot tata bahasa 13
Konsep inti Amazon Lex V2 14
Jalur Pembelajaran Mulai Cepat 14

Proses Pengembangan Terperinci 14
Konsep Inti dan Terminologi 15
Fitur Amazon Lex V2 Tingkat Lanjut 17
Ketersediaan Wilayah 17
Bahasa yang didukung 17
Bahasa dan lokal yang didukung 18
Bahasa dan lokal yang didukung oleh fitur Amazon Lex V2 21
Wilayah 24
Memulai 25
Ikhtisar 25
Daftar Periksa Prasyarat 25
Pengantar Amazon Lex V2 25
Langkah 1: Menyiapkan akun 26
Mendaftar ke AWS 26
Mmebuat pengguna IAM 27
Memberikan akses programatis 28
Langkah berikutnya 28
Langkah 2: Memulai (konsol) 28
Ikhtisar Konsol 28
Pilih Jalur Pembelajaran Anda 29
Latihan Dasar 29
Mulai Cepat: Buat chatbot dalam 5 menit 30
Latihan 1: Buat chatbot dari template 32
Latihan 2: Tinjau alur percakapan 35
Latihan 3: Bangun chatbot layanan pelanggan tingkat lanjut 46
Praktik Terbaik untuk Memulai 68
Contoh bot 73
Asisten agen call center 73
Langkah 1: Buat Indeks Amazon Kendra 74
Langkah 2: Buat Bot Amazon Lex V2 75
Langkah 3: Tambahkan Kustom dan Built-in Maksud 76
Langkah 4: Siapkan Amazon Cognito 77
Langkah 5: Menyebarkan Bot Anda sebagai Aplikasi Web 79
Langkah 6: Gunakan Bot 79
Bekerja dengan bot Amazon Lex V2 83
Perubahan alur percakapan di Amazon Lex V2 84

Berbagai cara untuk membuat bot dengan Amazon Lex V2 85
Menggunakan konsol 86
Menggunakan templat bot 87
Membuat bot Amazon Lex V2 menggunakan Desainer Chatbot Otomatis 90
Menambahkan bahasa baru ke bot Amazon Lex V2 99
Menambahkan maksud 99
Mengkonfigurasi prompt dalam urutan tertentu 102
Sampel ucapan 102
Struktur niat 104
Membuat jalur percakapan 127
Menggunakan pembangun percakapan Visual 145
Maksud bawaan 155
Menambahkan jenis slot 187
Built-in jenis slot 188
Jenis slot khusus 204
Jenis slot tata bahasa 207
Jenis slot komposit 356
Menguji bot 31
Optimalkan bot Lex V2 dengan AI generatif 368
Pembuat bot deskriptif dari deskripsi Anda 370
Contoh untuk pembuat bot deskriptif 374
Izin yang diperlukan untuk NLD 377
Gunakan generasi ucapan untuk menghasilkan contoh ucapan untuk pengenalan maksud 378
Menggunakan resolusi slot berbantuan untuk nilai slot 379
Contoh resolusi slot berbantuan 381
Aktifkan dalam konfigurasi AI generatif 384
Aktifkan resolusi slot berbantuan untuk slot Anda 385
Izin untuk resolusi slot berbantuan 387
Menggunakan BedrockAgentIntent untuk menggunakan Agen Batuan Dasar di Amazon Lex V2 388
Aktifkan dalam konfigurasi AI generatif 389
Aktifkan Intent Agen Batuan Dasar untuk bot Anda 390
Izin untuk Maksud Agen Batuan Dasar 390
Permintaan sampel 391
Meningkatkan klasifikasi maksud dan resolusi slot 393
Menonaktifkan NLU yang Dibantu 395

Selesaikan masukan pengguna yang ambigu dengan Disambiguasi Intent 395
Menonaktifkan Disambiguasi Intent 397
Optimalkan Bot menggunakan Bot Analyzer yang didukung AI 398
Amazon.qn AIntent 403
Izin 405
Membuat jaringan bot untuk bot Lex V2 Anda 406
Buat jaringan bot untuk bot Lex V2 Anda 407
Kelola jaringan bot Anda untuk bot Lex V2 Anda 408
Versi jaringan bot Anda untuk Lex V2 409
Alias untuk jaringan bot Anda untuk Lex V2 409
Integrasi saluran untuk jaringan bot Lex V2 Anda 409
Menyebarkan bot dari Lex V2 untuk lingkungan produksi Anda 411
Pembuatan versi dan alias dengan bot Lex V2 Anda 411
Versi 411
Alias untuk bot Lex V2 Anda 413
Integrasi dengan aplikasi Java 414
Gunakan Ketahanan Global untuk menyebarkan bot ke Wilayah lain 418
Izin 420
Menyebarkan Ketahanan Global dengan bot Lex V2 Anda 421
Integrasi dengan platform perpesanan 424
Integrasi dengan Facebook 425
Integrasi dengan Slack 428
Integrasi dengan Twilio SMS 432
Integrasi dengan pusat kontak 434
Amazon Chime SDK 435
Connect Pelanggan 436
Awan Genesys 437
Memahami percakapan bot Amazon Lex V2 438
Konteks percakapan dengan bot Lex V2 Anda 439
Mengatur konteks maksud untuk bot Lex V2 Anda 440
Menggunakan nilai slot default dalam maksud untuk bot Lex V2 Anda 442
Mengatur atribut sesi untuk bot Lex V2 Anda 443
Mengatur atribut permintaan untuk bot Lex V2 Anda 445
Mengatur batas waktu sesi 448
Berbagi informasi antar maksud dengan bot Lex V2 Anda 448
Mengatur atribut kompleks di bot Lex V2 Anda 449

Memahami sesi bot 450
Memulai sesi baru 452
Beralih maksud 452
Melanjutkan niat sebelumnya 453
Memvalidasi nilai slot 454
Mengintegrasikan fungsi AWS Lambda 455
AWS Lambdaformat acara masukan untuk Lex V2 455
AWS Lambdaformat respons untuk Lex V2 463
Bidang yang diperlukan dalam respons 466
Struktur umum 469
Niat 469
Slot 470
Status sesi 473
Membuat dan melampirkan fungsi Lambda ke alias bot 477
Lampirkan AWS Lambda fungsi ke bot Amazon Lex V2 menggunakan konsol 480
Lampirkan AWS Lambda fungsi ke bot Amazon Lex V2 menggunakan operasi API 482
Mendebug fungsi Lambda 488
Menyesuaikan interaksi bot dengan pengguna di Lex V2 489
Menganalisis sentimen pengguna dalam percakapan 489
Menggunakan skor kepercayaan diri untuk meningkatkan akurasi percakapan 490
Menggunakan skor keyakinan niat untuk meningkatkan pemilihan maksud dengan Lex V2 . 491
Menggunakan skor kepercayaan transkripsi suara untuk meningkatkan percakapan dengan bot Lex V2 Anda 494
Menyesuaikan transkripsi ucapan untuk digunakan dengan bot Lex V2 Anda 504
Mengkonfigurasi preferensi model pengenalan suara 504
Meningkatkan pengenalan suara dengan kosakata khusus 511
Mengkonfigurasi sensitivitas deteksi aktivitas suara 519
Menggunakan diarisasi speaker untuk fokus pada pembicara utama 522
Meningkatkan pengenalan nilai slot dengan petunjuk runtime dalam percakapan 528
Menangkap nilai slot dengan gaya ejaan selama percakapan 531
Menggunakan pengisi audio untuk meningkatkan respons bot 538
Jenis pengisi audio yang tersedia 539
Parameter waktu 539
Mengkonfigurasi pengisi audio 540
Pengisi audio dengan pesan sementara agen AI 541
Pengisi audio dengan dialog dan kait kode pemenuhan 541

Praktik terbaik untuk pengisi audio 542
Memantau kinerja bot di Lex V2 543
Mengukur kinerja bisnis dengan Analytics 543
Definisi kunci 544
Hasil penyaringan 546
Gambaran umum 547
Dasbor percakapan 551
Dasbor kinerja 556
Menggunakan API untuk analitik 560
Mengelola izin akses untuk analitik 567
Mengaktifkan log percakapan untuk bot Lex V2 Anda 567
Mencatat percakapan dengan log percakapan di Lex V2 568
Mengaburkan nilai slot dalam log percakapan dari Lex V2 585
Pengambilan log percakapan selektif di Lex V2 586
Kesalahan logging dengan log kesalahan di Lex V2 594
Kebijakan IAM untuk Log Kesalahan 595
Mengaktifkan Log Kesalahan di Lex V2 595
Menonaktifkan Log Kesalahan di Lex V2 596
Memantau metrik operasional di Lex V2 596
Mengukur metrik operasional dengan Lex CloudWatch V2 596
Melihat acara dengan CloudTrail 605
Mengevaluasi kinerja bot Lex V2 dengan Test Workbench 608
Menghasilkan set tes untuk Test Workbench 609
Kelola set tes 622
Jalankan tes 647
Cakupan set uji di Meja Kerja Uji 649
Lihat hasil tes 650
Rincian hasil tes di Test Workbench 651
Streaming percakapan ke bot Lex V2 Anda 658
Memulai aliran percakapan ke bot Amazon Lex V2 659
Urutan waktu acara untuk percakapan audio saat menggunakan bot Amazon Lex V2 663
Memulai percakapan streaming 665
Pengkodean aliran acara 681
Mengaktifkan bot Anda untuk terganggu 683
Menunggu pengguna untuk memberikan informasi tambahan 684
Mengkonfigurasi pembaruan kemajuan pemenuhan untuk bot Lex V2 Anda 686

Pembaruan pemenuhan 686
Respon pasca-pemenuhan 688
Timeout untuk masukan pengguna 690
Cara kerja perilaku interupsi di bot Lex V2 691
Mengatur batas waktu untuk input suara 691
Batas waktu untuk masukan teks 693
Tetapkan konfigurasi untuk input DTMF 693
Mengimpor dan mengekspor bot di Lex V2 696
Mengekspor bot dari Lex V2 696
Izin IAM diperlukan untuk mengekspor bot di Lex V2 697
Mengekspor bot Lex V2 (konsol) 698
Mengimpor bot di Lex V2 700
Izin IAM diperlukan untuk mengimpor 701
Mengimpor bot Lex V2 (konsol) 702
Menggunakan kata sandi saat mengimpor atau mengekspor 703
Format JSON untuk mengimpor dan mengekspor bot di Lex V2 704
Struktur file manifes 705
Struktur file bot 705
Struktur file lokal bot 706
Struktur file maksud 706
Struktur file slot 708
Struktur file jenis slot 711
Struktur file kosakata khusus 714
Menandai sumber daya di Lex V2 716
Menandai sumber daya Anda dengan konsol atau API 716
Batasan tag saat menggunakan Lex V2 717
Sumber daya penandaan (konsol) 717
Keamanan 719
Perlindungan data 720
Enkripsi saat diam 720
Enkripsi saat bergerak 721
Manajemen identitas dan akses 722
Audiens 722
Mengautentikasi dengan identitas 723
Mengelola akses menggunakan kebijakan 724
Bagaimana Amazon Lex V2 bekerja dengan IAM 726

Identity-based contoh kebijakan 735
Resource-based contoh kebijakan 746
Kebijakan terkelola AWS 753
Menggunakan peran tertaut layanan 771
Menggunakan peran layanan 776
Migrasi ke peran layanan 778
Pemecahan masalah 780
Pencatatan dan Pemantauan 782
Validasi kepatuhan 783
Ketahanan 784
Keamanan infrastruktur 784
Titik akhir VPC (AWS PrivateLink) 785
Pertimbangan untuk titik akhir VPC Amazon Lex V2 785
Membuat titik akhir VPC antarmuka untuk Amazon Lex V2 785
Membuat kebijakan titik akhir VPC untuk Amazon Lex V2 786
Contoh kode 787
Skenario 787
Membangun chatbot Amazon Lex V2 787
Panduan dan praktik terbaik 789
Kuota 792
Build-time kuota 792
Kuota runtime 794
Panduan migrasi 799
Ikhtisar Amazon Lex V2 799
Beberapa bahasa dalam bot 799
Arsitektur informasi yang disederhanakan 799
Peningkatan produktivitas pembangun 800
AWS CloudFormation sumber daya 802
Amazon Lex V2 dan CloudFormation template 802
Pelajari lebih lanjut tentang CloudFormation 803
Riwayat dokumen 804
Referensi API 823
AWS Glosarium 824
..... dcccxxv

Apa itu Amazon Lex V2?

Amazon Lex V2 adalah layanan AWS untuk membangun antarmuka percakapan untuk aplikasi yang menggunakan suara dan teks. Amazon Lex V2 menyediakan fungsionalitas mendalam dan fleksibilitas pemahaman bahasa alami (NLU) dan pengenalan suara otomatis (ASR) sehingga Anda dapat membangun pengalaman pengguna yang sangat menarik dengan interaksi percakapan yang hidup, dan membuat kategori produk baru.

Amazon Lex V2 memungkinkan pengembang untuk membangun chatbots percakapan dengan cepat. Dengan Amazon Lex V2, tidak diperlukan keahlian pembelajaran mendalam — untuk membuat chatbot, Anda menentukan alur percakapan dasar di konsol Amazon Lex V2. Amazon Lex V2 mengelola dialog dan secara dinamis menyesuaikan respons dalam percakapan. Dengan menggunakan konsol, Anda dapat membuat, menguji, dan mempublikasikan chatbot teks atau suara Anda. Anda kemudian dapat menambahkan antarmuka percakapan ke chatbots di perangkat seluler, aplikasi web, dan platform obrolan (misalnya, Facebook Messenger).

Amazon Lex V2 sekarang menyertakan AI-powered fitur-fitur canggih yang membuat pembuatan bot menjadi lebih kuat dan mudah diakses. NLU yang dibantu menggunakan Large Language Models (LLM) untuk meningkatkan klasifikasi maksud dan resolusi slot sambil tetap berada dalam maksud dan slot yang dikonfigurasi bot Anda. Ini berarti pemahaman yang lebih baik tentang permintaan pengguna dengan lebih sedikit data pelatihan yang diperlukan. Amazon Lex V2 juga mendukung kosakata khusus dalam 17 bahasa tambahan, memungkinkan penyebaran global dengan akurasi pengenalan suara yang lebih baik di berbagai pasar.

Dengan Multi-Region Replikasi (MRR), Anda sekarang dapat menerapkan bot Anda di beberapa wilayah AWS untuk meningkatkan ketersediaan dan pemulihan bencana, memastikan antarmuka percakapan Anda tetap dapat diakses oleh pengguna di seluruh dunia.

Amazon Lex V2 menyediakan integrasi dengan AWS Lambda, dan Anda dapat berintegrasi dengan banyak layanan lain di platform AWS, termasuk Connect Customer, Amazon Comprehend, dan Amazon Kendra. Integrasi dengan Lambda menyediakan akses bot ke konektor perusahaan tanpa server yang dibuat sebelumnya untuk menautkan ke data dalam aplikasi SaaS seperti Salesforce.

Untuk bot yang dibuat setelah 17 Agustus 2022, Anda dapat menggunakan percabangan bersyarat untuk mengontrol alur percakapan dengan bot Anda. Dengan percabangan bersyarat Anda dapat membuat percakapan yang kompleks tanpa perlu menulis kode Lambda.

Amazon Lex V2 memberikan manfaat berikut:

- Kesederhanaan - Amazon Lex V2 memandu Anda menggunakan konsol untuk membuat bot Anda sendiri dalam hitungan menit. Anda memberikan beberapa contoh frasa, dan Amazon Lex V2 membangun model bahasa alami lengkap di mana bot dapat berinteraksi menggunakan suara dan teks untuk mengajukan pertanyaan, mendapatkan jawaban, dan menyelesaikan tugas-tugas canggih.

- Teknologi deep learning yang demokratisasi — Amazon Lex V2 menyediakan teknologi ASR dan NLU untuk menciptakan sistem Speech Language Understanding (SLU). Melalui SLU, Amazon Lex V2 mengambil ucapan bahasa alami dan input teks, memahami maksud di balik input, dan memenuhi maksud pengguna dengan menjalankan fungsi bisnis yang sesuai.

  Pengenalan ucapan dan pemahaman bahasa alami adalah beberapa masalah yang paling menantang untuk dipecahkan dalam ilmu komputer, membutuhkan algoritma pembelajaran mendalam yang canggih untuk dilatih pada sejumlah besar data dan infrastruktur. Amazon Lex V2 menempatkan teknologi pembelajaran mendalam dalam jangkauan semua pengembang. Bot Amazon Lex V2 mengonversi ucapan masuk menjadi teks dan memahami maksud pengguna untuk menghasilkan respons cerdas sehingga Anda dapat fokus membangun bot dengan nilai tambah bagi pelanggan Anda dan menentukan kategori produk yang sepenuhnya baru yang dimungkinkan melalui antarmuka percakapan.

- Penerapan dan penskalaan yang mulus — Dengan Amazon Lex V2, Anda dapat membuat, menguji, dan menerapkan bot langsung dari konsol Amazon Lex V2. Amazon Lex V2 memungkinkan Anda mempublikasikan bot suara atau teks untuk digunakan di perangkat seluler, aplikasi web, dan layanan obrolan (misalnya, Facebook Messenger). Timbangan Amazon Lex V2 secara otomatis. Anda tidak perlu khawatir tentang penyediaan perangkat keras dan mengelola infrastruktur untuk memberi daya pada pengalaman bot Anda.

- Built-in integrasi dengan platform AWS — Amazon Lex V2 beroperasi secara native dengan layanan AWS lainnya, seperti AWS Lambda dan Amazon CloudWatch. Anda dapat memanfaatkan kekuatan platform AWS untuk keamanan, pemantauan, otentikasi pengguna, logika bisnis, penyimpanan, dan pengembangan aplikasi seluler.

- Cost-effectiveness— Dengan Amazon Lex V2, tidak ada biaya di muka atau biaya minimum. Anda hanya dikenakan biaya untuk permintaan teks atau ucapan yang dibuat. Harga pay-as-you-go dan biaya rendah per permintaan menjadikan layanan ini cara yang hemat biaya untuk membangun antarmuka percakapan. Dengan tingkat gratis Amazon Lex V2, Anda dapat dengan mudah mencoba Amazon Lex V2 tanpa investasi awal.

Kasus penggunaan umum untuk Amazon Lex V2

Amazon Lex V2 memungkinkan Anda membangun antarmuka percakapan yang canggih untuk berbagai skenario bisnis. Berikut adalah beberapa kasus penggunaan populer:

- Customer Support — Buat chatbot cerdas yang dapat menangani pertanyaan pelanggan umum, memecahkan masalah, dan meningkatkan masalah kompleks ke agen manusia. Integrasikan dengan CRM dan sistem basis pengetahuan yang ada.
- E-commerce dan Ritel — Bangun chatbot asisten belanja yang membantu pelanggan menemukan produk, memeriksa status pesanan, memproses pengembalian, dan memberikan rekomendasi yang dipersonalisasi berdasarkan riwayat pembelian.
- Pemesanan Janji Temu — Kembangkan chatbots penjadwalan untuk perawatan kesehatan, layanan profesional, atau perhotelan yang dapat memeriksa ketersediaan, memesan janji temu, mengirim pengingat, dan menangani pembatalan.
- IT Helpdesk — Buat chatbot dukungan internal yang dapat mengatur ulang kata sandi, memberikan panduan instalasi perangkat lunak, melacak tiket TI, dan menghubungkan karyawan dengan sumber daya teknis yang tepat.
- Layanan Keuangan — Bangun chatbot perbankan yang dapat memeriksa saldo akun, mentransfer dana, memberikan riwayat transaksi, dan menawarkan saran keuangan sambil mempertahankan standar keamanan yang ketat.
- Layanan SDM dan Karyawan — Kembangkan chatbot asisten SDM yang dapat menjawab pertanyaan kebijakan, membantu pendaftaran manfaat, memproses permintaan waktu istirahat, dan memberikan dukungan orientasi untuk karyawan baru.

Amazon Lex V2 terintegrasi secara mulus dengan platform komunikasi populer termasuk Slack, Microsoft Teams, WhatsApp Facebook Messenger, dan aplikasi web khusus, sehingga mudah untuk menyebarkan chatbots Anda di mana pengguna Anda sudah berkomunikasi.

Membayar Amazon Lex V2

Amazon Lex V2 menagih Anda hanya untuk permintaan teks atau ucapan yang Anda buat. Model ini memberi Anda layanan biaya variabel yang dapat tumbuh bersama bisnis Anda sambil memberi Anda keuntungan biaya infrastruktur AWS. Untuk informasi selengkapnya, lihat Harga Amazon Lex V2.

Saat Anda mendaftar AWS, akun AWS Anda secara otomatis mendaftar untuk semua layanan di AWS, termasuk Amazon Lex V2. Nmaun, Anda hanya dikenai biaya untuk layanan yang digunakan. Jika Anda adalah pelanggan Amazon Lex V2 baru, Anda dapat memulai dengan Amazon Lex V2 secara gratis. Untuk informasi selengkapnya, lihat tingkat gratis AWS .

Untuk melihat tagihan Anda, buka Dasbor Manajemen Penagihan dan Biaya di konsol AWS Manajemen Penagihan dan Biaya. Untuk mempelajari selengkapnya tentang Akun AWS penagihan, lihat Panduan AWS Billing Pengguna. Jika Anda memiliki pertanyaan tentang penagihan AWS dan Akun AWS, hubungi AWS Support.

Apakah Anda First-time Pengguna Amazon Lex V2?

Jika Anda adalah pengguna pertama kali Amazon Lex V2, kami sarankan Anda membaca bagian berikut secara berurutan:

1. Konsep inti Amazon Lex V2— Bagian ini memperkenalkan Amazon Lex V2 dan fitur yang Anda gunakan untuk membuat chatbot.
2. Memulai dengan Amazon Lex V2— Di bagian ini, Anda mengatur akun Anda dan menguji Amazon Lex V2.
3. Referensi API - Bagian ini berisi rincian tentang operasi API.

Menggunakan Amazon Lex V2 dengan AWS SDK

Kit pengembangan perangkat lunak AWS (SDK) tersedia untuk banyak bahasa pemrograman populer. Setiap SDK menyediakan API, contoh kode, dan dokumentasi yang memudahkan developer untuk membangun aplikasi dalam bahasa pilihan mereka.

Dokumentasi SDK	Contoh kode
AWS SDK untuk C++	AWS SDK untuk C++ contoh kode

Dokumentasi SDK	Contoh kode
AWS CLI	Contoh kode AWS CLI
AWS SDK untuk Go	AWS SDK untuk Go contoh kode
AWS SDK untuk Java	AWS SDK untuk Java contoh kode
AWS SDK untuk JavaScript	AWS SDK untuk JavaScript contoh kode
AWS SDK untuk Kotlin	AWS SDK untuk Kotlin contoh kode
AWS SDK untuk .NET	AWS SDK untuk .NET contoh kode
AWS SDK untuk PHP	AWS SDK untuk PHP contoh kode
Alat AWS untuk PowerShell	Alat AWS untuk PowerShell contoh kode
AWS SDK untuk Python v4	AWS SDK untuk contoh kode v4 Python
AWS SDK untuk Python (Boto3)	AWS SDK untuk Python (Boto3) contoh kode
AWS SDK untuk Ruby	AWS SDK untuk Ruby contoh kode
AWS SDK for Rust	AWS SDK for Rust contoh kode
AWS SDK for SAP ABAP	AWS SDK for SAP ABAP contoh kode
AWS SDK for Swift	AWS SDK for Swift contoh kode

Ketersediaan contoh

Tidak dapat menemukan apa yang Anda butuhkan? Minta contoh kode menggunakan tautan Berikan umpan balik di bagian bawah halaman ini.

Menggunakan Amazon Lex V2 dengan AWS SDK

Di luar antarmuka konsol, Amazon Lex V2 menyediakan akses terprogram yang komprehensif melalui AWS SDK. Ini memungkinkan Anda untuk mengintegrasikan kemampuan AI percakapan

langsung ke dalam aplikasi Anda, mengotomatiskan tugas manajemen bot, dan membangun solusi yang dapat diskalakan.

Saat Anda menggunakan AWS SDK dengan Amazon Lex V2, Anda dapat:

- Mengotomatiskan pembuatan dan pengelolaan bot — Membuat, memperbarui, dan menerapkan bot secara terprogram tanpa interaksi konsol manual
- Integrasikan dengan aplikasi yang ada — Tambahkan antarmuka percakapan ke aplikasi web, aplikasi seluler, dan sistem perusahaan
- Operasi bot skala - Kelola beberapa bot, maksud, dan jenis slot secara efisien melalui kode
- Menerapkan alur kerja khusus - Bangun alur percakapan canggih yang terintegrasi dengan logika bisnis Anda

Tabel di bawah ini menunjukkan AWS SDK yang mendukung operasi Amazon Lex V2. Pilih SDK yang sesuai dengan lingkungan pengembangan Anda dan ikuti tautan yang disediakan untuk memulai penginstalan dan implementasi.

Bahasa Pemrograman	AWS SDK	Memulai Sumber Daya
Java	AWS SDK for Java 2.x	Panduan Pengembang Referensi API
Python	AWS SDK UNTUK PYTHON (BOTO3)	Memulai Referensi API
JavaScript/ Node.js	AWS SDK untuk v3 JavaScript	Panduan Pengembang Referensi API
.NET/C#	AWS SDK for .NET	Panduan Pengembang Referensi API
Go	AWS SDK for Go v2	Panduan Pengembang Referensi API

Bahasa Pemrograman	AWS SDK	Memulai Sumber Daya
Ruby	AWS SDK for Ruby	Panduan Pengembang Referensi API
PHP	AWS SDK for PHP	Panduan Pengembang Referensi API

Untuk memulai dengan SDK apa pun:

1. Instal SDK untuk bahasa pemrograman pilihan Anda menggunakan petunjuk penginstalan di Panduan Pengembang
2. Konfigurasikan kredenal AWS dan setelan wilayah Anda
3. Siapkan izin IAM yang diperlukan untuk operasi Amazon Lex V2
4. Tinjau dokumentasi Referensi API untuk operasi spesifik yang Anda butuhkan
5. Mulailah dengan operasi dasar seperti membuat bot atau mencantumkan sumber daya yang ada

Untuk contoh terperinci dan panduan langkah demi langkah dalam membuat bot secara terprogram, lihat tautan dokumentasi SDK yang disediakan dalam tabel di atas.

Fitur terbaru

Topik ini memberikan informasi tentang fitur terbaru yang ditawarkan Amazon Lex V2:

AMAZON.BedrockAgentIntent untuk terhubung dengan Agen Batuan Dasar dan Pangkalan Pengetahuan

Amazon Lex V2 sekarang menyediakan maksud AMAZON.BedrockAgentIntent bawaan untuk menghubungkan bot Anda secara mulus dengan Agen Bedrock Amazon dan Pangkalan Pengetahuan untuk meningkatkan kemampuan AI percakapan.

- Dokumentasi

NLU yang dibantu untuk meningkatkan klasifikasi niat dan resolusi slot

Amazon Lex V2 sekarang menawarkan NLU Berbantuan untuk meningkatkan klasifikasi maksud dan resolusi slot menggunakan Large Language Models (LLM) sambil tetap berada dalam maksud dan slot yang dikonfigurasi bot Anda.

- Dokumentasi

Dukungan kosakata khusus untuk 17 bahasa tambahan

Amazon Lex V2 sekarang mendukung kosakata khusus dalam 17 bahasa tambahan untuk meningkatkan akurasi pengenalan suara.

- Dokumentasi

QinConnect maksud bawaan untuk integrasi Amazon Connect

Amazon Lex V2 sekarang menyediakan maksud QinConnect bawaan untuk menghubungkan bot Anda dengan Amazon Connect secara mulus untuk pengalaman pusat kontak yang ditingkatkan.

- Dokumentasi

AMAZON.Currency dan slot AMAZON.Confirmation bawaan sekarang mendukung semua lokal

Amazon Lex V2 sekarang mendukung AMAZON.Currency dan AMAZON.Confirmation built-in jenis slot di semua lokal, berkembang melampaui lokal Inggris dan Spanyol yang didukung sebelumnya.

- Dokumentasi

Niat QnA yang ditingkatkan dengan Basis Pengetahuan Batuan Dasar dan Pagar Pembatas

Maksud bawaan QnA untuk AI generatif sekarang mendukung Basis Pengetahuan Batuan Dasar dan Pagar Pembatas untuk respons yang lebih aman dan akurat.

- Dokumentasi

Support untuk model Anthropic Claude 3

Maksud bawaan QnA untuk AI generatif sekarang mendukung model Anthropic Claude 3 Haiku dan Anthropic Claude 3 Sonnet untuk meningkatkan kemampuan percakapan.

- Dokumentasi

Ketahanan global untuk replikasi bot di seluruh wilayah

Amazon Lex V2 sekarang menawarkan ketahanan Global untuk mereplikasi bot Anda di wilayah AWS kedua untuk meningkatkan ketersediaan dan pemulihan bencana.

- Dokumentasi

Dukungan regional untuk AWS GovCloud (US-West)

Amazon Lex V2 sekarang tersedia di AWS GovCloud (US-West).

- Titik akhir dan kuota Amazon Lex

Fitur AI generatif untuk Amazon Lex V2

Amazon Lex V2 sekarang memungkinkan Anda memanfaatkan kemampuan AI generatif Amazon Bedrock untuk bot Anda.

- Pembangun bot deskriptif
  - Apa postingan baru
  - Dokumentasi
- Resolusi slot yang dibantu
  - Apa postingan baru
  - Dokumentasi
- Generasi ucapan
  - Apa postingan baru
  - Dokumentasi
- `AMAZON.QnAIntent`(FAQ Percakapan)
  - Apa postingan baru
  - Dokumentasi
- Posting Blog AWS Machine Learning

AMAZON.Confirmation slot bawaan untuk disambiguasi yang Yes/No/Maybe/Don tidak diketahui.

Amazon Lex V2 sekarang menawarkan slot `AMAZON.Confimation` built-in untuk meningkatkan akurasi konfirmasi slot dan Yes/No/Maybe/Don tidak tahu tanggapan.

- Dokumentasi

Mengukur kinerja bisnis dengan Analytics

Amazon Lex V2 sekarang menawarkan pengguna kemampuan untuk melihat kinerja maksud dan slot di dasbor Analytics.

- Apa postingan baru
- Dokumentasi

Mengevaluasi kinerja bot dengan meja kerja Uji

Amazon Lex V2 sekarang menawarkan kepada pengguna kemampuan untuk membuat dan menjalankan set pengujian untuk mengukur kinerja bot dan meningkatkan metrik bot.

- Apa postingan baru
- Dokumentasi
- Posting Blog AWS Machine Learning

Templat bot spesifik vertikal

Amazon Lex V2 sekarang menawarkan templat bot pra-bangun kepada pengguna dengan alur percakapan siap pakai bersama dengan data pelatihan dan permintaan dialog, untuk modalitas suara dan obrolan.

- Apa postingan baru
- Dokumentasi

Jaringan bot

Amazon Lex V2 sekarang menawarkan pengguna kemampuan untuk menggabungkan beberapa bot ke dalam satu jaringan dan kemampuan untuk merutekan permintaan ke bot yang sesuai berdasarkan input pengguna.

- Apa postingan baru
- Dokumentasi

Visual conversation builder

Amazon Lex V2 sekarang menawarkan pembangun percakapan seret dan lepas untuk merancang dan memvisualisasikan jalur percakapan dengan mudah dengan menggunakan maksud dalam lingkungan visual yang kaya.

- Apa postingan baru
- Dokumentasi
- Posting Blog AWS Machine Learning

Jenis slot komposit

Amazon Lex V2 sekarang menawarkan pengguna kemampuan untuk menggabungkan beberapa slot ke dalam slot komposit menggunakan ekspresi logis.

- Apa postingan baru
- Dokumentasi

Pencabangan kondisional

Amazon Lex V2 sekarang menawarkan pengguna kemampuan untuk menulis kondisi untuk lebih mengontrol jalur yang diambil pelanggan melalui percakapan dengan bot Anda.

- Apa postingan baru
- Dokumentasi

Perancang chatbot otomatis

Amazon Lex V2 sekarang menawarkan kepada pengguna opsi untuk mendesain chatbot secara otomatis dari transkrip percakapan. Baca contoh untuk penggunaan.

- Apa postingan baru
- Dokumentasi
- Posting Blog AWS Machine Learning

- Halaman Desainer Chatbot Otomatis Amazon Lex

Petunjuk runtime

Amazon Lex V2 sekarang menawarkan kepada pengguna opsi untuk mengonfigurasi petunjuk runtime untuk meningkatkan pengenalan frasa guna meningkatkan elisitasi nilai slot.

- Apa postingan baru
- Dokumentasi

Kosakata kustom

Amazon Lex V2 sekarang menawarkan kepada pengguna opsi untuk membuat kosakata khusus, daftar frasa yang dapat menyertakan kata benda yang tepat atau kata-kata khusus domain, untuk dikenali Amazon Lex V2 dalam input audio.

- Apa postingan baru
- Dokumentasi
- Posting Blog AWS Machine Learning

Jenis slot tata bahasa

Amazon Lex V2 sekarang menawarkan pengguna kemampuan untuk menulis tata bahasa dalam format XMLmengikuti Speech Recognition Grammar Specification (SRGS) untuk mengumpulkan informasi dalam percakapan.

- Apa postingan baru
- Dokumentasi
- Posting Blog AWS Machine Learning

Konsep inti Amazon Lex V2

Amazon Lex V2 memungkinkan Anda membangun aplikasi obrolan (bot) untuk memperoleh informasi dari pengguna untuk menyelesaikan tugas. Misalnya, Anda dapat membuat chatbot untuk memberikan dukungan pelanggan, menjawab pertanyaan yang sering diajukan, atau membuat janji temu. Berikut adalah langkah-langkah khas untuk bekerja dengan Amazon Lex V2:

Jalur Pembelajaran Mulai Cepat

Baru mengenal Amazon Lex V2? Ikuti jalur pembelajaran progresif ini untuk memulai dengan cepat:

1. Mulai dengan Template (5 menit) — Pilih dari template chatbot yang sudah dibuat sebelumnya seperti Customer Support FAQ, Appointment Booking, atau Order Status. Template mencakup maksud, slot, dan contoh ucapan yang telah dikonfigurasi sebelumnya.
2. Kustomisasi Chatbot Anda (15 menit) - Ubah template agar sesuai dengan kasus penggunaan spesifik Anda. Tambahkan maksud Anda sendiri, perbarui contoh ucapan, dan konfigurasikan jenis slot untuk domain Anda.
3. Uji dan Perbaiki (10 menit) — Gunakan konsol uji bawaan untuk melakukan percakapan dengan chatbot Anda. Aktifkan NLU Berbantuan untuk pemahaman yang lebih baik dengan data pelatihan minimal.
4. Terapkan dan Integrasikan (20 menit) — Publikasikan chatbot Anda dan integrasikan dengan platform pilihan Anda (Slack, aplikasi web, atau aplikasi seluler).

Total waktu untuk chatbot kerja: ~ 50 menit

Untuk pemahaman yang lebih komprehensif, lanjutkan dengan proses pengembangan terperinci di bawah ini.

Proses Pengembangan Terperinci

Untuk bot yang lebih kompleks atau saat membangun dari awal, ikuti proses pengembangan komprehensif ini:

1. Buat bot dan tambahkan satu atau beberapa bahasa. Konfigurasikan bot sehingga memahami tujuan pengguna, terlibat dalam percakapan dengan pengguna untuk memperoleh informasi, dan memenuhi maksud pengguna.

2. Uji botnya. Anda dapat menggunakan klien jendela uji yang disediakan oleh konsol Amazon Lex V2.
3. Publikasikan versi dan buat alias.
4. Menyebarkan bot. Anda dapat menggunakan bot di aplikasi atau platform perpesanan Anda sendiri seperti Facebook Messenger atau Slack

Konsep Inti dan Terminologi

Sebelum memulai, biasakan diri Anda dengan konsep dan terminologi inti Amazon Lex V2 berikut:

- Bot — Bot melakukan tugas otomatis seperti memesan pizza, memesan hotel, memesan bunga, dan sebagainya. Bot Amazon Lex V2 didukung oleh kemampuan pengenalan suara otomatis (ASR) dan pemahaman bahasa alami (NLU).

  Bot Amazon Lex V2 dapat memahami masukan pengguna yang disediakan dengan teks atau ucapan dan bahasa alami percakapan.
- Bahasa — Bot Amazon Lex V2 dapat berkomunikasi dalam satu atau lebih bahasa. Setiap bahasa tidak tergantung pada yang lain, Anda dapat mengonfigurasi Amazon Lex V2 untuk berkomunikasi dengan pengguna menggunakan kata dan frasa asli. Untuk informasi selengkapnya, lihat Bahasa dan lokal yang didukung oleh Amazon Lex V2.
- Intent — Intent mewakili tindakan yang ingin dilakukan pengguna. Anda membuat bot untuk mendukung satu atau lebih maksud terkait. Misalnya, Anda dapat membuat maksud yang memesan pizza dan minuman. Untuk setiap maksud, Anda memberikan informasi yang diperlukan berikut:
  - Nama maksud — Nama deskriptif untuk maksud tersebut. Misalnya, **`OrderPizza`**.
  - Contoh ucapan — Bagaimana pengguna dapat menyampaikan maksud tersebut. Misalnya, pengguna mungkin mengatakan “Bisakah saya memesan pizza” atau “Saya ingin memesan pizza.”
  - Cara memenuhi maksud — Bagaimana Anda ingin memenuhi maksud setelah pengguna memberikan informasi yang diperlukan. Kami menyarankan Anda membuat fungsi Lambda untuk memenuhi maksud.

    Anda dapat mengonfigurasi intent secara opsional sehingga Amazon Lex V2 mengembalikan informasi kembali ke aplikasi klien untuk pemenuhan yang diperlukan.

Selain maksud khusus, Amazon Lex V2 menyediakan maksud bawaan untuk mengatur bot Anda dengan cepat. Untuk informasi selengkapnya, lihat Maksud bawaan.

Amazon Lex selalu menyertakan maksud fallback untuk setiap bot. Maksud fallback digunakan setiap kali Amazon Lex tidak dapat menyimpulkan maksud pengguna. Untuk informasi selengkapnya, lihat AMAZON.FallbackIntent.

- Slot — Niat dapat memerlukan nol atau lebih slot, atau parameter. Anda menambahkan slot sebagai bagian dari konfigurasi maksud. Saat runtime, Amazon Lex V2 meminta pengguna untuk nilai slot tertentu. Pengguna harus memberikan nilai untuk semua slot yang diperlukan sebelum Amazon Lex V2 dapat memenuhi maksud tersebut.

  Misalnya `OrderPizza` intent membutuhkan slot seperti ukuran, jenis kerak, dan jumlah pizza. Untuk setiap slot, Anda memberikan jenis slot dan satu atau lebih petunjuk yang dikirimkan Amazon Lex V2 ke klien untuk mendapatkan nilai dari pengguna. Seorang pengguna dapat membalas dengan nilai slot yang berisi kata-kata tambahan, seperti “pizza besar tolong” atau “mari kita tetap dengan kecil.” Amazon Lex V2 masih memahami nilai slotnya.
- Jenis slot — Setiap slot memiliki tipe. Anda dapat membuat jenis slot Anda sendiri, atau Anda dapat menggunakan jenis slot bawaan. Misalnya, Anda dapat membuat dan menggunakan jenis slot berikut untuk `OrderPizza` intent:
  - Ukuran - Dengan nilai enumerasi`Small`,, `Medium` dan. `Large`
  - Kerak — Dengan nilai enumerasi dan. `Thick` `Thin`

  Amazon Lex V2 juga menyediakan tipe slot bawaan. Misalnya, `AMAZON.Number` adalah jenis slot built-in yang dapat Anda gunakan untuk jumlah pizza yang dipesan. Untuk informasi selengkapnya, lihat Maksud bawaan.
- Versi — Versi adalah snapshot bernomor dari pekerjaan Anda yang dapat Anda publikasikan untuk digunakan di berbagai bagian alur kerja Anda, seperti pengembangan, penerapan beta, dan produksi. Setelah Anda membuat versi, Anda dapat menggunakan bot seperti yang ada saat versi dibuat. Setelah Anda membuat versi, itu tetap sama saat Anda terus bekerja pada aplikasi Anda.
- Alias — Alias adalah penunjuk ke versi bot tertentu. Dengan alias, Anda dapat memperbarui versi yang digunakan aplikasi klien Anda. Misalnya, Anda dapat mengarahkan alias ke versi 1 bot Anda. Ketika Anda siap untuk memperbarui bot, Anda menerbitkan versi 2 dan mengubah alias untuk menunjuk ke versi baru. Karena aplikasi Anda menggunakan alias alih-alih versi tertentu, semua klien Anda mendapatkan fungsionalitas baru tanpa perlu diperbarui.

Fitur Amazon Lex V2 Tingkat Lanjut

Selain konsep inti di atas, Amazon Lex V2 menyertakan fitur-fitur canggih yang meningkatkan kemampuan bot:

- Assisted NLU — Menggunakan Large Language Models (LLMs) untuk meningkatkan klasifikasi maksud dan resolusi slot. Fitur ini membantu bot Anda memahami permintaan pengguna dengan lebih akurat, bahkan ketika mereka menggunakan frasa yang berbeda dari contoh pelatihan Anda. NLU yang dibantu bekerja dalam maksud dan slot Anda yang dikonfigurasi, memberikan pemahaman yang lebih baik tanpa memerlukan data pelatihan yang ekstensif.
- Percakapan Multi-giliran — Amazon Lex V2 dapat mempertahankan konteks di beberapa putaran percakapan, memungkinkan back-and-forth interaksi alami. Pengguna dapat memberikan informasi secara bertahap, berubah pikiran, atau mengajukan pertanyaan klarifikasi tanpa kehilangan konteks percakapan.
- Context Switching — Bot tingkat lanjut dapat menangani perubahan topik dalam percakapan. Misalnya, pengguna mungkin mulai bertanya tentang informasi akun, kemudian beralih ke menempatkan pesanan, dan kembali ke topik asli. Amazon Lex V2 dapat mengelola sakelar konteks ini dengan anggun.
- Strategi Fallback — Saat Amazon Lex V2 tidak memahami permintaan pengguna, Anda dapat mengonfigurasi perilaku fallback yang canggih termasuk pertanyaan klarifikasi, permintaan saran, atau eskalasi ke agen manusia. Ini memastikan pengguna selalu memiliki jalur ke depan dalam percakapan.
- Conversation Flow Management — Gunakan percabangan bersyarat dan kontrol alur percakapan untuk membuat pola dialog yang kompleks tanpa menulis kode. Anda dapat merutekan percakapan berdasarkan respons pengguna, nilai slot, atau data eksternal.

Ketersediaan Wilayah

Untuk daftar Wilayah AWS tempat Amazon Lex V2 tersedia, lihat titik akhir dan kuota Amazon Lex V2 di Referensi Umum Amazon Web Services.

Bahasa dan lokal yang didukung oleh Amazon Lex V2

Amazon Lex V2 mendukung berbagai bahasa dan lokal. Topik ini mencantumkan bahasa yang didukung dan fitur yang mendukung bahasa-bahasa ini.

Bahasa dan lokal yang didukung

Amazon Lex V2 mendukung bahasa dan lokal berikut.

Kode	Bahasa dan lokal
AF_ZA*	Afrikaans (Afrika Selatan)
AR_AE	Teluk Arab (Uni Emirat Arab)
Ar_sa*	Arab (Arab Saudi)
BG_BG*	Bulgaria (Bulgaria)
C_es	Catalan (Spanyol)
cs_CZ*	Ceko (Republik Ceko)
cy_GB*	Welsh (Inggris Raya)
da_dk*	Denmark (Denmark)
de_AT	Jerman (Austria)
de_CH*	Jerman (Swiss)
de_DE	Jerman (Jerman)
en_AB*	Inggris (Skotlandia)
en_AU	Inggris (Australia)
en_GB	Inggris (Inggris)
en_IE*	Inggris (Irlandia)
en_in	Inggris (India)
en_NZ*	Inggris (Selandia Baru)
en_US	Inggris (AS)

Kode	Bahasa dan lokal
en_WL*	Inggris (Wales)
en_ZA	Bahasa Inggris (Afrika Selatan)
es_419	Spanyol (Amerika Latin)
es_es	Spanyol (Spanyol)
es_MX*	Spanyol (Meksiko)
es_US	Spanyol (AS)
et_et*	Estonia (Estonia)
FA_IR*	Farsi (Iran)
fi_fi	Finlandia (Finlandia)
fr_BE*	Prancis (Belgia)
fr_CA	Prancis (Kanada)
fr_FR	Prancis (Prancis)
Dia*	Ibrani (Israel)
Hi in	Hindi (India)
HR_HR*	Kroasia (Kroasia)
hu_HU*	Hungaria (Hongaria)
ID_ID*	Indonesia (Indonesia)
Adakan_adalah*	Islandia (Islandia)
IT_IT	Italia (Italia)
J_JP	Jepang (Japan)

Kode	Bahasa dan lokal
Km_KH*	Khmer (Kamboja)
K_KR	Korea (Korea)
Lt_LT*	Lithuania (Lithuania)
Lv_Lv*	Latvian (Latvia)
MS_saya*	Melayu (Malaysia)
nl_be*	Belanda (Belgia)
nl_NL	Belanda (Belanda)
Tidak_tidak	Norwegia (Norwegia)
pl_PL	Polandia (Polandia)
pt_BR	Bahasa Portugis (Brasil)
pt_PT	Portugis (Portugal)
RO_RO*	Rumania (Rumania)
ru_RU*	Rusia (Rusia)
sk_SK*	Slowakia (Slovakia)
S_si*	Slovenia (Slovenia)
Begitu*	Somalia (Somalia)
SR_RS*	Serbia (Serbia)
SU_ID*	Bahasa Sunda (Indonesia)
sv_SE	Swedia (Swedia)
T_th*	Thai (Thailand)

Kode	Bahasa dan lokal
TL_pH*	Tagalog/Filipino (Filipina)
tr_TR*	Turki (Turki)
Inggris/UA*	Ukraina (Ukraina)
vi_VN*	Vietnam (Vietnam)
zH_CN	Mandarin (RRC)
Zh_HK	Kanton (Hong Kong)
U_da*	Zulu (Afrika Selatan)

Lokal dengan dukungan fitur terbatas

Lokal yang ditandai dengan tanda bintang (*) memiliki dukungan fitur terbatas melalui AI generatif dan Pengenalan Ucapan Otomatis (ASR) atau Text-To-Speech (TTS) pihak ketiga. Lihat tabel berikut untuk daftar lengkap fitur yang didukung.

Dukungan fitur AI generatif

Dukungan fitur AI generatif termasuk NLU Ber bantuan dalam Mode Utama, Resolusi Slot Ber bantuan, dan Disambiguasi Intent.

Bahasa dan lokal yang didukung oleh fitur Amazon Lex V2

Tabel berikut mencantumkan fitur Amazon Lex V2 yang terbatas pada bahasa dan lokal tertentu. Semua fitur Amazon Lex V2 lainnya didukung dalam semua bahasa dan lokal.

Fitur	Bahasa dan lokal yang didukung
AMAZON.AlphaNumeric	Semua bahasa dan lokal kecuali bahasa Korea (ko_KR) dan lokal dukungan fitur terbatas
AMAZON.KendraSearchIntent	Inggris (US) (en_US)

Fitur	Bahasa dan lokal yang didukung
Meningkatkan pengenalan suara dengan kosakata khusus	Inggris (UK) (en_GB) Inggris (US) (en_US)
Desainer Chatbot Otomatis	Inggris (US) (en_US)
Ketersediaan wilayah	Bahasa dan lokal berikut tidak tersedia di Wilayah Asia Pasifik (Singapura) (ap-south east-1) dan Afrika (Cape Town) (ap-south-1): • Teluk Arab (Uni Emirat Arab) (ar_AE) • Katalan (Spanyol) (ca_ES) • Finlandia (Finlandia) (fi_FI) • Hindi (India) (Hi_in) • Belanda (Belanda) (nl_NL) • Norwegia (Norwegia) (no_NO) • Polandia (pl_PL) • Portugis (Brasil) (pt_BR) • Portugis (Portugal) (pt_PT) • Swedia (sv_SE) • Mandarin (RRC) (zh_CN) • Kanton (Hong Kong) (zh_HK)
Mengatur konteks maksud untuk bot Lex V2 Anda	Inggris (US) (en_US)
Jenis slot tata bahasa	Inggris (Australia) (en_AU) Inggris (UK) (en_GB) Inggris (US) (en_US)
Menggunakan beberapa nilai dalam slot	Inggris (US) (en_US)

Fitur	Bahasa dan lokal yang didukung
Meningkatkan pengenalan nilai slot dengan petunjuk runtime dalam percakapan	Inggris (UK) (en_GB) Inggris (US) (en_US)
Menangkap nilai slot dengan gaya ejaan selama percakapan	Inggris (Australia) (en_AU) Inggris (UK) (en_GB) Inggris (US) (en_US)
Menggunakan skor kepercayaan diri untuk meningkatkan akurasi percakapan	Inggris (UK) (en_GB) Inggris (US) (en_US)
Hanya dengan Third-party ASR (Deepgram)	Belanda (Belgia) (nl_BE) Prancis (Belgia) (fr_BE)

Fitur	Bahasa dan lokal yang didukung
Hanya dengan Third-party TTS (ElevenLabs)	Afrikaans (Afrika Selatan) (AF_ZA) Bulgaria (Bulgaria) (bg_BG) Kroasia (Kroasia) (hr_HR) Inggris (Skotlandia) (en_AB) Estonia (Estonia) (et_ET) Farsi (Iran) (fa_IR) Ibrani (Israel) (he_IL) Hungaria (Hongaria) (hu_HU) Bahasa Indonesia (Indonesia) (ID_ID) Latvia (Latvia) (lv_LV) Lituania (Lituania) (lt_LT) Melayu (Malaysia) (ms_MY) Serbia (Serbia) (sr_RS) Slowakia (Slovakia) (sk_SK) Slovenia (Slovenia) (sl_SI)

Wilayah

Untuk daftar Wilayah AWS tempat Amazon Lex V2 tersedia, lihat Wil ayah AWS dan titik akhir di Referensi Umum AWS.

Memulai dengan Amazon Lex V2

Ikhtisar

Jika Anda baru mengenal Amazon Lex V2, kami sarankan Anda membaca Konsep inti Amazon Lex V2 terlebih dahulu.

Daftar Periksa Prasyarat

Sebelum Anda mulai membangun bot Amazon Lex V2 pertama Anda, pastikan Anda memiliki yang berikut:

- Akun AWS — Anda memerlukan akun AWS yang aktif. Jika Anda tidak memilikinya, Anda dapat membuatnya secara gratis dan memanfaatkan AWS Free Tier, yang mencakup 10.000 permintaan teks dan 5.000 permintaan ucapan per bulan untuk Amazon Lex V2. Untuk petunjuk penyiapan terperinci, lihatMendaftar ke AWS.
- Pengguna IAM dengan Izin - Buat pengguna IAM dengan izin Amazon Lex V2 yang sesuai. Kami akan memandu Anda melalui pembuatan pengguna administrator untuk memulai. Untuk informasi lengkap tentang izin dan kebijakan IAM, lihat. Manajemen identitas dan akses untuk Amazon Lex V2
- Browser yang Didukung - Gunakan browser web modern (Chrome, Firefox, Safari, atau Edge) untuk mengakses konsol Amazon Lex V2.
- Komitmen Waktu - Biarkan 30-60 menit untuk menyelesaikan latihan memulai, tergantung pada tingkat pengalaman Anda.

Perkiraan Biaya: Latihan dalam panduan ini menggunakan sumber daya AWS Tingkat Gratis. Jika Anda berada dalam batas tingkat gratis, seharusnya tidak ada biaya. Jika tidak, biaya biasanya di bawah $1 untuk menyelesaikan semua latihan.

Pengantar Amazon Lex V2

Sebelum dapat menggunakan Amazon Lex V2, Anda harus membuat akun AWS dan mendapatkan ID akun AWS. Anda membuat akun pengguna dan peran IAM yang memberi Anda izin yang diperlukan untuk menggunakan Amazon Lex V2

Setelah membuat akun, Anda dapat mulai menggunakan Amazon Lex V2 dengan AWS Console atau dengan API.

Amazon Lex V2 menyediakan operasi API yang dapat Anda integrasikan dengan aplikasi yang ada. Untuk daftar operasi yang didukung, lihat Referensi API. Anda dapat menggunakan salah satu opsi berikut:

- AWS SDK — Saat menggunakan permintaan SDKs Anda ke Amazon Lex V2 secara otomatis ditandatangani dan diautentikasi menggunakan kredensi yang Anda berikan. Kami menyarankan Anda menggunakan SDK untuk membangun aplikasi Anda.
- AWS CLI — Anda dapat menggunakan AWS CLI untuk mengakses fitur Amazon Lex V2 apa pun tanpa harus menulis kode apa pun.

Langkah 1: Siapkan Akun AWS dan buat Pengguna administrator

Sebelum Anda menggunakan Amazon Lex V2 untuk pertama kalinya, selesaikan tugas-tugas berikut:

1. Mendaftar ke AWS
2. Mmebuat pengguna IAM

Mendaftar ke AWS

Jika Anda sudah memiliki akun AWS, lewati tugas ini.

Saat Anda mendaftar ke Amazon Web Services (AWS), akun AWS Anda secara otomatis mendaftar untuk semua layanan di AWS, termasuk Amazon Lex V2. Anda hanya membayar biaya layanan yang Anda gunakan.

Dengan Amazon Lex V2, Anda hanya membayar untuk sumber daya yang Anda gunakan. Jika Anda adalah pelanggan AWS baru, Anda dapat memulai dengan Amazon Lex V2 secara gratis. Untuk informasi selengkapnya, lihat Tingkatan Gratis AWS.

Jika Anda sudah memiliki akun AWS, lewati ke tugas berikutnya. Jika Anda belum memiliki akun AWS, gunakan prosedur berikut untuk membuatnya.

Untuk membuat akun AWS

- Untuk membuat akun AWS, lihat Bagaimana cara membuat dan mengaktifkan akun AWS baru? di Pusat Pengetahuan AWS.

Tuliskan ID akun AWS Anda karena Anda akan membutuhkannya untuk tugas berikutnya.

Mmebuat pengguna IAM

Layanan di AWS, seperti Amazon Lex V2, mengharuskan Anda memberikan kredensi saat Anda mengaksesnya sehingga layanan dapat menentukan apakah Anda memiliki izin untuk mengakses sumber daya yang dimiliki oleh layanan tersebut.

Buat akun pengguna IAM untuk mengakses akun Anda untuk Amazon Lex V2. Untuk panduan komprehensif tentang pembuatan pengguna IAM, lihat Memulai IAM di Panduan Pengguna IAM.

- Gunakan AWS Identity and Access Management (IAM) untuk membuat pengguna IAM
- Tambahkan pengguna ke grup IAM dengan izin administratif
- Berikan izin administratif kepada pengguna IAM yang Anda buat.

Anda kemudian dapat mengakses AWS menggunakan URL khusus dan kredensional pengguna IAM.

Latihan Memulai dalam panduan ini mengasumsikan Anda memiliki pengguna (`adminuser`) dengan hak istimewa administrator. Ikuti prosedur untuk membuat `adminuser` di akun Anda.

Untuk membuat pengguna administrator dan masuk ke konsol tersebut

1. Buat pengguna administrator yang dipanggil `adminuser` di akun AWS Anda. Untuk petunjuk, lihat Membuat pengguna IAM Pertama Anda dan Grup Administrator di Panduan Pengguna IAM.
2. Sebagai pengguna, Anda dapat masuk ke AWS Management Console menggunakan URL khusus. Untuk informasi selengkapnya, Cara Pengguna Masuk ke Akun Anda dalam Panduan Pengguna IAM.

Untuk informasi lebih lanjut tentang IAM, lihat Panduan Pengguna IAM.

Masalah Pengaturan Umum

Jika Anda mengalami masalah selama penyiapan akun, berikut adalah solusi umum:

- Kesalahan Izin Ditolak - Pastikan pengguna IAM Anda memiliki izin Amazon Lex V2 yang diperlukan. Untuk memulai, izin administrator direkomendasikan. Untuk informasi rinci tentang izin Amazon Lex V2, lihat bagian Keamanan dalam panduan ini.

- Masalah Akses Konsol — Pastikan Anda masuk dengan URL pengguna IAM yang benar untuk akun Anda, bukan konsol AWS root. Untuk informasi selengkapnya tentang praktik terbaik IAM, lihat bagian Keamanan dalam panduan ini.
- Ketersediaan Wilayah — Amazon Lex V2 tidak tersedia di semua wilayah AWS. Periksa titik akhir dan kuota Amazon Lex untuk mengonfirmasi ketersediaan di wilayah pilihan Anda. Untuk informasi tentang lokal yang didukung, lihatBahasa dan lokal yang didukung oleh Amazon Lex V2.
- Batas Tingkat Gratis - Pantau penggunaan Anda agar tetap dalam batas tingkat gratis: 10.000 permintaan teks dan 5.000 permintaan ucapan per bulan. Untuk informasi harga terperinci, lihat Harga Amazon Lex.
- Batas Kuota — Jika Anda menemukan batas layanan, tinjau kuota saat ini dan permintaan meningkat jika diperlukan. Untuk informasi selengkapnya, lihat Panduan dan praktik terbaik.

Untuk bantuan pemecahan masalah tambahan, lihat bagian Pemantauan dalam panduan ini untuk informasi tentang log kesalahan dan debugging.

Memberikan akses programatis

Untuk informasi tentang cara mendapatkan ID kunci akses dan kunci akses rahasia, lihat Memahami dan mendapatkan kredensi AWS Anda di AWS General Reference.

Pengaturan Lingkungan Pengembang

Untuk Pengembang: Jika Anda berencana untuk menggunakan AWS CLI atau SDKs, Anda juga ingin mengonfigurasi lingkungan pengembangan Anda. Instal AWS CLI dan konfigurasikan dengan kunci akses Anda. Untuk petunjuk terperinci, lihat Menginstal AWS CLI. Untuk contoh SDK, lihatContoh bot.

Langkah berikutnya

Langkah 2: Memulai (konsol)

Langkah 2: Memulai (konsol)

Ikhtisar Konsol

Cara termudah untuk mempelajari cara menggunakan Amazon Lex V2 adalah dengan menggunakan konsol. Pilih jalur pembelajaran yang paling sesuai dengan peran dan tingkat pengalaman Anda:

Pilih Jalur Pembelajaran Anda

Pilih jalur pembelajaran yang paling sesuai dengan peran dan tingkat pengalaman Anda:

- Jalur Tanpa Kode (Pengguna Bisnis) - Sempurna untuk analis bisnis, manajer produk, dan pengguna non-teknis yang ingin membangun bot hanya menggunakan antarmuka konsol.
  - Mulailah dengan: Mulai Cepat: Buat chatbot dalam 5 menit
  - Kemudian coba: Latihan 1: Buat chatbot dari template
  - Belajar: Praktik Terbaik untuk Memulai
- Jalur Pengembang - Ideal untuk pengembang perangkat lunak yang ingin mengintegrasikan Amazon Lex V2 dengan aplikasi, menggunakan APIs, dan menerapkan logika bisnis khusus.
  - Mulailah dengan: Latihan 1: Buat chatbot dari template
  - Lanjutan: Latihan 3: Bangun chatbot layanan pelanggan tingkat lanjut
  - Kemudian jelajahi: Mengintegrasikan AWS Lambda fungsi ke bot Amazon Lex V2 Anda
  - Ulasan: Contoh bot
  - Lanjutan: Integrasi API dan penggunaan SDK
- Enterprise Path — Dirancang untuk arsitek dan insinyur yang merencanakan penyebaran skala besar dengan persyaratan keamanan, kepatuhan, dan skalabilitas.
  - Mulailah dengan: Latihan 1: Buat chatbot dari template
  - Contoh perusahaan: Latihan 3: Bangun chatbot layanan pelanggan tingkat lanjut
  - Ulasan: Keamanan dan praktik terbaik IAM
  - Jelajahi: Gunakan Ketahanan Global untuk menyebarkan bot ke Wilayah lain
  - Rencana: Pemantauan, pencatatan, dan optimalisasi biaya

Latihan Dasar

Terlepas dari jalur yang Anda pilih, kami sarankan memulai dengan latihan dasar ini:

- Latihan 1 — Buat bot Amazon Lex V2 menggunakan templat, bot yang telah ditentukan sebelumnya yang menyediakan semua konfigurasi bot yang diperlukan. Anda hanya melakukan pekerjaan minimum untuk menguji end-to-end pengaturan.
- Latihan 2 - Tinjau struktur JSON yang dikirim antara aplikasi klien Anda dan bot Amazon Lex V2.

- Latihan 3 — Bangun bot obrolan layanan pelanggan tingkat lanjut yang menunjukkan kemampuan tingkat perusahaan termasuk manajemen pesanan, penjualan cerdas, pembuatan prospek, dan fitur bertenaga AI untuk pengalaman pelanggan yang dipersonalisasi.

Mulai Cepat: Buat chatbot dalam 5 menit

Panduan memulai cepat ini membantu Anda membuat chatbot Amazon Lex V2 yang berfungsi hanya dalam 5 menit menggunakan templat yang sudah dibuat sebelumnya. Anda akan memiliki chatbot fungsional yang dapat segera Anda uji dan sesuaikan dengan kebutuhan Anda.

Langkah 1: Pilih Template

Amazon Lex V2 menyediakan beberapa templat pra-bangun untuk kasus penggunaan umum:

- Customer Support FAQ - Menangani pertanyaan layanan pelanggan umum
- Pemesanan Janji Temu - Jadwalkan dan kelola janji temu
- Status Pesanan - Periksa informasi pesanan dan status pengiriman
- Helpdesk TI - Memberikan dukungan teknis dan pemecahan masalah

Untuk memilih template

1. Buka konsol Amazon Lex V2 di https://console.aws.amazon.com/lexv2/.
2. Pilih Buat bot.
3. Pilih Mulai dengan template.
4. Pilih template Customer Support FAQ untuk memulai cepat ini.
5. Masukkan nama bot, seperti**`MyFirstChatbot`**.
6. Pilih Buat.

Langkah 2: Kustomisasi Cepat (Opsional)

Template sudah dikonfigurasi sebelumnya, tetapi Anda dapat dengan cepat menyesuaikannya untuk kebutuhan spesifik Anda:

Untuk menyesuaikan chatbot Anda

1. Dalam ikhtisar bot, tinjau maksud yang telah dikonfigurasi sebelumnya.

2. Pilih maksud untuk memodifikasi, seperti GetAccountInfo.
3. Tambahkan contoh ucapan Anda sendiri yang cocok dengan cara pelanggan Anda membuat frasa permintaan.
4. Perbarui pesan respons agar sesuai dengan suara merek Anda.
5. Pilih Simpan maksud.

Langkah 3: Uji Bot Anda

Uji chatbot Anda segera menggunakan konsol uji bawaan:

Untuk menguji chatbot Anda

1. Pilih Build untuk mengkompilasi bot Anda.
2. Tunggu hingga build selesai (biasanya 1-2 menit).
3. Di konsol uji di sebelah kanan, ketik: **`I need help with my account`**
4. Tekan Enter dan amati respons bot.
5. Coba frasa lain untuk menguji pemahaman bot.

Langkah 4: Menyebarkan Bot Anda

Setelah Anda puas dengan tanggapan chatbot Anda, gunakan untuk digunakan:

Untuk menyebarkan chatbot Anda

1. Pilih Publikasikan dari menu tindakan bot.
2. Buat versi baru dengan memilih Buat versi.
3. Buat alias (seperti “Produksi”) yang menunjuk ke versi Anda.
4. Pilih metode integrasi Anda.

Langkah Berikutnya

Selamat! Anda sekarang memiliki chatbot Amazon Lex V2 yang berfungsi. Inilah yang dapat Anda lakukan selanjutnya:

- Aktifkan NLU Berbantuan - Tingkatkan pemahaman dengan pemrosesan bahasa alami bertenaga AI.

- Tambahkan Lebih Banyak Maksud — Perluas kemampuan chatbot Anda dengan kasus penggunaan tambahan.
- Integrasikan dengan Lambda - Tambahkan logika bisnis dan integrasi sistem eksternal. Lihat Mengintegrasikan AWS Lambda fungsi ke bot Amazon Lex V2 Anda.
- Mengatur Pemantauan - Lacak penggunaan dan kinerja. Lihat Mengukur metrik operasional dengan Amazon CloudWatch.
- Pelajari Fitur Lanjutan — Jelajahi manajemen alur percakapan, dialog multi-putaran, dan peralihan konteks.

Untuk panduan yang lebih rinci, lanjutkan dengan. Latihan 1: Buat chatbot dari template

Latihan 1: Buat chatbot dari template

Dalam latihan ini, Anda membuat chatbot Amazon Lex V2 pertama Anda dan mengujinya di konsol Amazon Lex V2. Untuk latihan ini, Anda menggunakan OrderFlowerstemplate, yang menunjukkan kasus penggunaan praktis di dunia nyata untuk e-commerce.

OrderFlowers Contoh Bot

Anda menggunakan OrderFlowerstemplate untuk membuat chatbot Amazon Lex V2 yang dapat menangani permintaan pemesanan bunga. Contoh ini menunjukkan bagaimana bisnis dapat mengotomatiskan pengambilan pesanan dengan chatbots cerdas. Untuk informasi lebih lanjut tentang struktur bot, lihatKonsep inti Amazon Lex V2.

- Maksud — Bot mencakup satu maksud utama:
  - `OrderFlowers`- Menangani permintaan pemesanan bunga dengan mengumpulkan jenis bunga, tanggal pengambilan, dan waktu penjemputan
- Jenis slot — Bot menggunakan tipe slot bawaan yang secara otomatis mengenali dan menangani format data umum:
  - AMAZON.Date- Mengenali tanggal seperti “besok”, “Jumat depan”, atau “15 Maret”
  - AMAZON.Time- Mengenali waktu seperti “2 PM”, “siang”, atau “seperempat tiga”
  - `FlowerTypes`(custom) - Varietas bunga tertentu seperti “mawar”, “tulip”, “lili”
- Slot — OrderFlowers Maksudnya memerlukan informasi berikut sebelum bot dapat memenuhi pesanan bunga:
  - `FlowerType`(tipe FlowerTypes kustom) - Jenis bunga yang dipesan

  - `PickupDate`(AMAZON.Datetipe) - Kapan harus mengambil bunga
  - `PickupTime`(AMAZON.Timetipe) - Jam berapa untuk mengambil bunga
- Sampel Ucapan — Contoh ucapan berikut menunjukkan cara alami pengguna dapat meminta pesanan bunga:
  - “Saya ingin mengambil bunga”
  - “Saya ingin memesan beberapa bunga”
  - “Bisakah saya mendapatkan bunga untuk diambil?”
  - “Aku harus membeli bunga”
- Prompts — Setelah bot mengidentifikasi intent, bot menggunakan prompt berikut untuk mengisi slot:
  - Prompt untuk `FlowerType` slot — “Jenis bunga apa yang ingin Anda pesan?”
  - Prompt untuk `PickupDate` slot — “Hari apa Anda ingin {FlowerType} diambil?”
  - Prompt untuk `PickupTime` slot — “Pada jam berapa Anda ingin {FlowerType} diambil?”
  - Pernyataan konfirmasi — “Oke, {FlowerType} Anda akan siap untuk diambil oleh {PickupTime} di {PickupDate}. Apakah ini terdengar baik-baik saja?”

Buat Bot Anda

Untuk membuat bot Amazon Lex V2 (Konsol)

1. Masuk ke AWS Management Console dan buka konsol Amazon Lex di https://console.aws.amazon.com/lex/.
2. Pilih Buat bot.
3. Untuk metode Creation, pilih Mulai dengan contoh.
4. Di bagian Contoh bot, pilih OrderFlowersdari daftar.
5. Di bagian konfigurasi Bot berikan bot nama dan deskripsi opsional. Nama harus unik di akun Anda.
6. Di bagian Izin, pilih Buat peran baru dengan izin Amazon Lex dasar. Ini akan membuat peran AWS Identity and Access Management (IAM) dengan izin yang dibutuhkan Amazon Lex V2 untuk menjalankan bot Anda.
7. Di bagian Children's Online Privacy Protection Act (COPPA), buatlah pilihan yang tepat.
8. Di bagian Session timeout dan Advanced settings, tinggalkan default.
9. Pilih Berikutnya. Amazon Lex V2 membuat bot Anda.

Tambahkan Bahasa ke Bot Anda

Setelah Anda membuat bot Anda, Anda harus menambahkan satu atau lebih bahasa yang didukung bot. Sebuah bahasa berisi maksud, jenis slot, dan slot yang digunakan bot untuk berkomunikasi dengan pengguna.

Untuk menambahkan bahasa ke bot

1. Di bagian Bahasa, pilih bahasa yang didukung, dan tambahkan deskripsi.
2. Biarkan bidang ambang batas skor keyakinan interaksi Suara dan klasifikasi Intent dengan defaultnya.
3. Pilih Selesai untuk menambahkan bahasa ke bot.

Uji Bot Anda

Setelah Anda memilih Selesai, konsol akan membuka editor maksud. Anda dapat menggunakan editor maksud untuk memeriksa maksud yang digunakan oleh bot. Setelah selesai memeriksa bot, Anda dapat mengujinya.

Untuk menguji OrderFlowers bot

1. Pilih Build di bagian atas halaman. Tunggu sampai bot dibangun.
2. Saat build selesai, pilih Test untuk membuka jendela pengujian.
3. Uji botnya. Mulailah percakapan dengan salah satu contoh ucapan, seperti “Saya ingin mengambil bunga.”

Memungkinkan NLU untuk Meningkatkan Pemahaman

Sekarang setelah Anda memiliki chatbot yang berfungsi, mari tingkatkan dengan Assisted NLU untuk meningkatkan pengenalan maksud dan resolusi slot. NLU yang dibantu menggunakan Large Language Models (LLMs) untuk lebih memahami permintaan pengguna, bahkan ketika mereka menggunakan frasa yang berbeda dari contoh pelatihan Anda.

Untuk mengaktifkan NLU yang Dibantu

1. Di konsol Amazon Lex V2, navigasikan ke pengaturan bot Anda.
2. Di panel navigasi kiri, pilih Pengaturan bot.

3. Di bawah Assisted NLU, pilih Aktifkan.
4. Pilih Simpan untuk menerapkan perubahan.
5. Bangun bot Anda lagi untuk menerapkan peningkatan NLU Berbantuan.

Uji Peningkatan: Coba variasi ini di konsol pengujian Anda untuk melihat bagaimana NLU Berbantuan menangani berbagai frasa:

- “Saya ingin membeli beberapa mawar” (harus memicu OrderFlowers niat dan menangkap FlowerType)
- “Bisakah saya mengirimkan bunga besok?” (harus memicu OrderFlowers niat dan menangkap PickupDate)
- “Saya butuh tulip untuk diambil pada jam 3 sore” (harus memicu OrderFlowers niat dan menangkap dan) FlowerType PickupTime

Perhatikan bagaimana chatbot dapat memahami variasi alami ini tanpa mengharuskan Anda untuk menambahkannya sebagai contoh ucapan eksplisit. Ini didukung oleh Assisted NLU, yang menggunakan AI untuk meningkatkan pemahaman bahasa alami.

Langkah selanjutnya

Sekarang setelah Anda membuat bot pertama menggunakan template, Anda dapat menggunakan konsol untuk membuat bot Anda sendiri. Untuk instruksi tentang membuat bot khusus, dan untuk informasi lebih lanjut tentang membuat bot, lihatBekerja dengan bot Amazon Lex V2.

Latihan 2: Tinjau alur percakapan

Dalam latihan ini Anda meninjau struktur JSON yang dikirim antara aplikasi klien Anda dan bot Amazon Lex V2 yang Anda buat. Latihan 1: Buat chatbot dari template Percakapan menggunakan RecognizeTextoperasi untuk menghasilkan struktur JSON. RecognizeUtteranceMengembalikan informasi yang sama seperti header HTTP dalam respon.

Struktur JSON dibagi oleh setiap putaran percakapan. Giliran adalah permintaan dari aplikasi klien dan respons dari bot.

Giliran 1

Selama giliran pertama percakapan, aplikasi klien memulai percakapan dengan bot Anda. Baik URI dan badan permintaan memberikan informasi tentang permintaan tersebut.

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

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

- URI mengidentifikasi bot yang berkomunikasi dengan aplikasi klien. Ini juga mencakup pengidentifikasi sesi yang dihasilkan oleh aplikasi klien yang mengidentifikasi percakapan tertentu antara pengguna dan bot.
- Isi permintaan berisi teks yang diketik pengguna ke aplikasi klien. Dalam hal ini, hanya teks yang dikirim, namun aplikasi Anda dapat mengirim informasi tambahan, seperti atribut permintaan atau status sesi. Untuk informasi lebih lanjut, lihat RecognizeTextoperasi.

Dari`text`, Amazon Lex V2 mendeteksi maksud pengguna, untuk memesan bunga. Amazon Lex V2 memilih salah satu slot maksud (`FlowerType`) dan salah satu petunjuk untuk slot, dan kemudian mengirimkan respons berikut ke aplikasi klien. Klien menampilkan respons kepada pengguna.

```
{
    "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"
    }
}
```

Giliran 2

Pada gilirannya 2, pengguna merespons prompt dari bot Amazon Lex V2 pada gilirannya 1 dengan nilai yang mengisi slot. `FlowerType`

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

Respons untuk giliran 2 menunjukkan `FlowerType` slot diisi dan memberikan prompt untuk memperoleh nilai slot berikutnya.

```
{
    "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"
    }
}
```

Giliran 3

Pada gilirannya 3, pengguna merespons prompt dari bot Amazon Lex V2 pada gilirannya 2 dengan nilai yang mengisi slot. `PickupDate`

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

Respons untuk giliran 3 baik `PickupDate` slot `FlowerType` dan diisi dan memberikan prompt untuk memperoleh nilai slot terakhir.

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

Giliran 4

Pada gilirannya 4, pengguna memberikan nilai slot akhir untuk maksud tersebut, waktu bunga diambil.

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

Sebagai tanggapan, Amazon Lex V2 mengirimkan prompt konfirmasi kepada pengguna untuk mengonfirmasi bahwa pesanan sudah benar. `dialogAction`Diatur ke `ConfirmIntent` dan `confirmationState` adalah`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"
    }
}
```

Giliran 5

Pada giliran terakhir, pengguna merespons dengan prompt konfirmasi.

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

Sebagai tanggapan, Amazon Lex V2 mengirimkan menunjukkan bahwa maksud telah dipenuhi dengan menyetel `confirmationState` ke `Confirmed` dan `dialogAction` to close. Semua nilai slot tersedia untuk aplikasi klien.

```
{
    "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"
    }
}
```

Latihan 3: Bangun chatbot layanan pelanggan tingkat lanjut

Dalam latihan lanjutan ini, Anda akan membangun chatbot layanan pelanggan yang canggih untuk perusahaan e-commerce. Bot ini menunjukkan kemampuan tingkat perusahaan termasuk manajemen pesanan, peningkatan cerdas, perolehan prospek, dan pengoptimalan pendapatan. Chatbot menggunakan fitur yang didukung AI untuk memberikan pengalaman pelanggan yang dipersonalisasi dan mendorong pertumbuhan bisnis.

SmartCommerce Ikhtisar Bot Layanan Pelanggan

Bot Layanan SmartCommerce Pelanggan dirancang untuk menangani interaksi pelanggan yang kompleks sambil memaksimalkan peluang pendapatan. Contoh ini menunjukkan bagaimana bisnis dapat mengotomatiskan layanan pelanggan sambil mendorong pertumbuhan penjualan melalui manajemen percakapan yang cerdas.

- Maksud Kustom - Bot mencakup beberapa maksud khusus untuk layanan pelanggan yang komprehensif:
  - `CheckOrderStatus`- Memverifikasi dan memberikan informasi status pesanan
  - `ProcessReturn`- Menangani permintaan pengembalian dan pertukaran
  - `UpsellProducts`- Merekomendasikan produk dan layanan tambahan
  - `CaptureLeadInfo`- Mengumpulkan informasi pelanggan untuk menghasilkan prospek
  - `ScheduleCallback`- Membukukan panggilan balik layanan pelanggan
- Intent Bawaan — Memanfaatkan maksud bawaan Amazon Lex V2 untuk interaksi umum:
  - AMAZON.HelpIntent- Memberikan bantuan dan bimbingan
  - AMAZON.CancelIntent- Menangani permintaan pembatalan
  - AMAZON.StopIntent- Mengakhiri percakapan dengan anggun
- Jenis Slot Kustom - Jenis slot khusus untuk data spesifik bisnis:
  - `ProductCategories`- Elektronik, Pakaian, Rumah, Buku, Olahraga
  - `ReturnReasons`- Cacat, Ukuran Salah, Berubah Pikiran, Tidak Seperti yang Dijelaskan
  - `ContactPreferences`- Email, SMS, Panggilan Telepon, Pemberitahuan Dalam Aplikasi
  - `CustomerTiers`- Perunggu, Perak, Emas, Platinum
- Jenis Slot Bawaan - Menggunakan slot internal Amazon Lex V2 untuk format data umum:
  - AMAZON.Date- Untuk tanggal pengiriman dan penjadwalan panggilan balik
  - AMAZON.Time- Untuk waktu callback dan jendela pengiriman
  - AMAZON.EmailAddress- Untuk pengumpulan email pelanggan
  - AMAZON.PhoneNumber- Untuk nomor telepon pelanggan
  - AMAZON.Number- Untuk nomor dan jumlah pesanan

Konfigurasi Maksud Terperinci

CheckOrderStatus Niat

Maksud ini menangani verifikasi pesanan dan pertanyaan status, memberikan informasi pesanan real-time kepada pelanggan sambil mengidentifikasi peluang peningkatan penjualan.

- Slot yang Diperlukan:
  - `OrderNumber`(AMAZON.Number) - Nomor pesanan pelanggan
  - `CustomerEmail`(AMAZON.EmailAddress) - Email untuk verifikasi

- Sampel Ucapan:
  - “Apa status pesanan saya {OrderNumber}”
  - “Saya perlu memeriksa nomor pesanan {OrderNumber}”
  - “Di mana paket saya”
  - “Lacak pesanan saya {OrderNumber}”
  - “Apakah pesanan saya sudah dikirim”
- Fitur bertenaga AI:
  - Menggunakan NLU Berbantuan untuk memahami variasi seperti “paket saya”, “barang saya”, “pengiriman saya”
  - Secara otomatis mengekstrak nomor urut dari pola bicara alami

UpsellProducts Niat

Maksud ini secara proaktif mengidentifikasi peluang pendapatan dengan merekomendasikan produk pelengkap dan layanan premium berdasarkan riwayat pelanggan dan konteks interaksi saat ini.

- Slot yang Diperlukan:
  - `ProductCategory`(ProductCategories) - Kategori minat
  - `CustomerTier`(CustomerTiers) - Tingkat loyalitas pelanggan
  - `Budget`(AMAZON.Number) - Kisaran anggaran pelanggan
- Sampel Ucapan:
  - “Tunjukkan produk terkait”
  - “Apa lagi yang akan terjadi dengan ini”
  - “Apakah Anda memiliki kesepakatan”
  - “Saya tertarik dengan {ProductCategory} item”
  - “Apa yang dijual hari ini”
- Fitur Optimasi Pendapatan:
  - Harga dinamis berdasarkan tingkat pelanggan
  - Rekomendasi yang dipersonalisasi menggunakan riwayat pembelian
  - Pembuatan penawaran waktu terbatas untuk mendorong pembelian langsung

CaptureLeadInfo Niat

Maksud ini secara sistematis mengumpulkan informasi pelanggan untuk menghasilkan prospek, menciptakan data berharga yang dapat dijual kepada agregator atau digunakan untuk kampanye pemasaran yang ditargetkan.

- Slot yang Diperlukan:
  - `CustomerFirstName`(AMAZON.FirstName) - Nama depan pelanggan untuk personalisasi
  - `CustomerLastName`(AMAZON.LastName) - Nama belakang pelanggan untuk pencarian riwayat penjualan
  - `CustomerEmail`(AMAZON.EmailAddress) - Alamat email
  - `CustomerPhone`(AMAZON.PhoneNumber) - Nomor telepon
  - `InterestCategory`(ProductCategories) - Kepentingan produk
  - `ContactPreference`(ContactPreferences) - Metode kontak yang disukai
- Sampel Ucapan:
  - “Saya ingin mendengar tentang produk baru”
  - “Daftarkan saya untuk pembaruan”
  - “Saya tertarik dengan {InterestCategory}”
  - “Beri saya informasi tentang penjualan”
  - “Saya ingin bergabung dengan milis Anda”
  - “Nama saya adalah {CustomerFirstName} {CustomerLastName}”
  - “Saya {CustomerFirstName} {CustomerLastName} dan saya ingin rekomendasi yang dipersonalisasi”
- Fitur Generasi Lead:
  - Pengumpulan data terstruktur untuk integrasi CRM
  - Lead scoring berdasarkan kualitas interaksi
  - Kualifikasi dan perutean prospek otomatis
  - Pemformatan data untuk penjualan agregator
- Integrasi Sejarah Penjualan:
  - Pencarian nama pelanggan di tabel riwayat pembelian yang disimpan S3
  - Analisis pola pembelian untuk rekomendasi yang dipersonalisasi
  - Identifikasi perilaku pembelian lintas kategori

- Analisis tren pembelian musiman untuk penawaran yang ditargetkan
- Perhitungan nilai seumur hidup pelanggan untuk harga berbasis tingkatan

Alur Percakapan Multi-Giliran

SmartCommerce Bot dirancang untuk menangani percakapan multi-putaran yang kompleks yang menjangkau banyak maksud dan mempertahankan konteks di seluruh interaksi. Ini menciptakan pengalaman alami seperti manusia sambil memaksimalkan peluang pendapatan.

Contoh Alur Percakapan

Putaran 1-3: Pemeriksaan Status Pesanan

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

Turn 4-6: Peluang Upselling

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

Giliran 7-11: Pembuatan Prospek dengan Pencarian Riwayat Penjualan

```
Customer: "Not right now, but I'd like to hear about future deals"
Bot: "Absolutely! I can set you up to receive personalized offers. What's your first
 name?"
Customer: "John"
Bot: "Thanks John! And your last name so I can look up your purchase history for better
 recommendations?"
Customer: "Smith"
Bot: "Perfect! Let me check your purchase history... I see you've bought electronics
 and home items before.
     Based on your past purchases, I can offer you exclusive deals on smart home
 accessories.
     How would you prefer to be contacted?"
Customer: "Email is fine"
Bot: "Great! I'll make sure john.doe@email.com gets personalized deals based on your
 purchase history.
     You're all set for our premium recommendations program!"
```

Fitur Manajemen Konteks

- Ketekunan Sesi: Mempertahankan informasi pelanggan di berbagai maksud
- Intent Chaining: Transisi mulus antara pemeriksaan order, upselling, dan lead capture
- Personalisasi Dinamis: Menyesuaikan penawaran berdasarkan tingkat pelanggan dan riwayat pembelian
- Pelacakan Pendapatan: Memantau tingkat keberhasilan upsell dan skor kualitas prospek

Fitur yang didukung AI

Chatbot ini memanfaatkan beberapa kemampuan AI untuk menyediakan layanan pelanggan yang cerdas dan dipersonalisasi sekaligus mendorong pertumbuhan bisnis.

Implementasi NLU yang Dibantu

NLU yang dibantu menggunakan Model Bahasa Besar untuk memahami maksud pelanggan bahkan ketika mereka menggunakan frasa non-standar atau menggabungkan beberapa permintaan dalam satu ucapan.

- Pemahaman Bahasa Alami:

  - “Barang saya belum tiba” → CheckOrderStatus niat
  - “Aku ingin mengembalikan sampah ini” → niat ProcessReturn
  - “Tunjukkan padaku apa lagi yang kamu punya” → UpsellProducts niat
- Pengakuan Multi-Niat:
  - “Periksa pesanan saya 12345 dan daftarkan saya untuk penawaran” → + CheckOrderStatus CaptureLeadInfo

Resolusi Slot Generatif

Menggunakan AI untuk mengekstrak nilai slot dari input bahasa alami yang kompleks tanpa memerlukan kecocokan yang tepat dengan data pelatihan.

- Ekstraksi Cerdas:
  - “Saya membeli beberapa elektronik minggu lalu, nomor pesanan adalah sesuatu seperti 12345" → OrderNumber: 12345,: Elektronik ProductCategory
  - “Hubungi saya di ponsel saya di 555-123-4567 atau email saya di john@company.com" → Telepon: 555-123-4567, Email: john@company.com

Integrasi Analisis Sentimen

Memantau sentimen pelanggan sepanjang percakapan untuk menyesuaikan pendekatan dan meningkat bila perlu.

- Perutean Berbasis Sentimen:
  - Sentimen positif → Pendekatan upselling agresif
  - Sentimen netral → Layanan standar dengan upselling yang lembut
  - Sentimen negatif → Fokus pada resolusi masalah, peningkatan minimal

Verifikasi Pesanan dan Sistem Konfirmasi

Bot mencakup sistem konfirmasi komprehensif yang memverifikasi identitas pelanggan, mengonfirmasi detail pesanan, dan mengirimkan konfirmasi melalui beberapa saluran.

Proses Verifikasi Multi-Faktor

- Verifikasi Utama:

    - Validasi nomor pesanan
    - Konfirmasi alamat email
    - 4 digit terakhir metode pembayaran (untuk operasi sensitif)
- Verifikasi Sekunder:
    - Konfirmasi alamat pengiriman
    - Validasi tanggal pembelian
    - Verifikasi detail produk

Sistem Konfirmasi Multi-Saluran

Bot secara otomatis mengirimkan konfirmasi melalui beberapa saluran berdasarkan preferensi pelanggan dan jenis tindakan.

- Konfirmasi Email:
    - Pembaruan status pesanan dengan informasi pelacakan
    - Otorisasi pengembalian dengan label pengiriman prabayar
    - Konfirmasi pembelian Upsell dengan detail pengiriman
    - Konfirmasi pengambilan prospek dengan penawaran selamat datang
- Konfirmasi SMS:
    - Pembaruan status pesanan segera
    - Pemberitahuan pengiriman dengan jendela waktu
    - Peringatan penjualan kilat untuk pelanggan yang tertarik
- Pemberitahuan Dalam Aplikasi:
    - Pembaruan pesanan waktu nyata
    - Rekomendasi produk yang dipersonalisasi
    - Pembaruan program loyalitas

Integrasi Point of Sale (POS)

Chatbot terintegrasi langsung dengan sistem POS untuk memproses transaksi, menerapkan diskon, dan memperbarui inventaris secara real-time.

Pemrosesan Transaksi Real-Time

- Alur Transaksi Upsell:

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

- Pelacakan Pendapatan:
  - Tingkatkan tingkat konversi berdasarkan tingkat pelanggan
  - Peningkatan nilai pesanan rata-rata per interaksi
  - Atribusi pendapatan untuk interaksi chatbot
  - Dampak nilai seumur hidup pelanggan

Mesin Harga Dinamis

- Harga Berbasis Tingkat:
  - Perunggu: Harga standar
  - Silver: 5% discount on upsells
  - Emas: 10-20% discount on upsells
  - Platinum: 25% discount +gratis ongkos kirim
- Harga Kontekstual:
  - Skenario penyelesaian masalah: Diskon tambahan untuk menjaga kepuasan
  - Pelanggan bernilai tinggi: Tingkatan harga eksklusif
  - Izin inventaris: Diskon agresif untuk barang yang bergerak lambat

Sistem Generasi Timbal Tingkat Lanjut

Chatbot mencakup sistem generasi prospek canggih yang menangkap, memenuhi syarat, dan memonetisasi data pelanggan melalui beberapa aliran pendapatan.

Strategi Pengambilan Timbal Multi-Touch

- Penangkapan Oportunistik:
  - Selama pemeriksaan status pesanan: “Apakah Anda ingin pembaruan pada produk serupa?”
  - Setelah penyelesaian masalah: “Bisakah kami memberi tahu Anda tentang peningkatan produk?”
  - Selama interaksi upsell: “Haruskah kami mengingatkan Anda tentang penawaran masa depan dalam kategori ini?”
- Tangkapan Nilai Tambah:
  - Akses harga anggota eksklusif
  - Akses awal ke peluncuran produk baru
  - Rekomendasi produk yang dipersonalisasi
  - Penawaran khusus ulang tahun dan ulang tahun

Kualifikasi Pimpinan Otomatis

- Kriteria Penilaian:
  - Nilai riwayat pembelian: 0-25 poin
  - Tingkat keterlibatan: 0-20 poin
  - Kelengkapan informasi kontak: 0-15 poin
  - Kategori produk luasnya bunga: 0-15 poin
  - Tanggapan terhadap upaya upsell: 0-25 poin
- Kategori Lead:
  - Hot Leads (80-100 poin): Kontak tim penjualan langsung
  - Prospek Hangat (60-79 poin): Urutan pengasuhan otomatis
  - Cold Leads (40-59 poin): Buletin bulanan
  - Prospek (0-39 poin): Email promosi triwulanan

Memimpin Monetisasi Data untuk Agregator

Format sistem dan paket memimpin data untuk dijual ke agregator pihak ketiga, menciptakan aliran pendapatan tambahan.

- Paket Data:

  - Premium Package: Info kontak lengkap, riwayat pembelian, preferensi, skor keterlibatan
  - Standard Package: Info kontak, preferensi dasar, minat kategori
  - Basic Package: Alamat email, kategori minat utama
- Fitur Kepatuhan:
  - Kepatuhan GDPR dengan pelacakan persetujuan eksplisit
  - Kepatuhan CCPA dengan mekanisme opt-out
  - Kepatuhan CAN-SPAM untuk pemasaran email
  - Kebijakan penyimpanan dan penghapusan data
- Optimalisasi Pendapatan:
  - Harga dinamis berdasarkan skor kualitas prospek
  - Kemitraan data eksklusif dengan agregator premium
  - Penawaran waktu nyata untuk prospek bernilai tinggi

Strategi Optimalisasi Pendapatan

Setiap interaksi dirancang untuk memaksimalkan pendapatan melalui peningkatan cerdas, penjualan silang, dan optimasi nilai seumur hidup pelanggan.

Strategi Peningkatan Cerdas

- Peningkatan Kontekstual:
  - Pemeriksaan status pesanan → “Pesanan Anda sudah siap! Tambahkan pengiriman dipercepat hanya dengan $5?”
  - Permintaan pengembalian → “Alih-alih kembali, apakah Anda ingin menukar model kami yang ditingkatkan?”
  - Pertanyaan produk → “Item ini berpasangan sempurna dengan [produk pelengkap] - bundel dan hemat 15%”
- Penjualan Berbasis Urgensi:
  - “Penawaran ini berakhir dalam 24 jam”
  - “Hanya tersisa 3 stok pada harga ini”
  - “Flash sale berakhir pada tengah malam”

Peluang Cross-Selling

- Penjualan Ekosistem Produk:
  - Elektronik → Aksesoris, jaminan, layanan instalasi
  - Pakaian → Item yang cocok, produk perawatan, layanan penataan
  - Perlengkapan rumah → Barang pelengkap, produk perawatan, konsultasi desain
- Layanan Upselling:
  - Jaminan yang diperpanjang dan rencana perlindungan
  - Tingkatan dukungan pelanggan premium
  - Layanan berlangganan dan pengiriman otomatis
  - Instalasi dan pengaturan profesional

Optimalisasi Nilai Seumur Hidup Pelanggan

- Integrasi Program Loyalitas:
  - Peningkatan tingkat otomatis berdasarkan interaksi chatbot
  - Poin bonus untuk terlibat dengan penawaran upsell
  - Hadiah dan diskon khusus chatbot eksklusif
- Strategi Retensi:
  - Resolusi masalah proaktif untuk mencegah churn
  - Penawaran win-back yang dipersonalisasi untuk pelanggan yang tidak aktif
  - Penawaran perayaan ulang tahun dan tonggak sejarah

Prosedur Implementasi

Ikuti step-by-step prosedur ini untuk membangun chatbot layanan pelanggan tingkat lanjut Anda dengan semua fitur pengoptimalan pendapatan.

Prasyarat dan Persyaratan Pengaturan

Sebelum membuat SmartCommerce bot, pastikan Anda memiliki penyiapan akun AWS yang diperlukan, izin, dan memahami pertimbangan layanan.

Akun AWS dan Persyaratan Akses

- Akun AWS: Anda memerlukan akun AWS aktif dengan penagihan diaktifkan. Jika Anda tidak memilikinya, daftar di aws.amazon.com.
- Izin IAM: Pengguna atau peran AWS Anda harus memiliki izin berikut:
  - `lex:*`- Izin Amazon Lex V2 penuh untuk pembuatan dan manajemen bot
  - `iam:CreateRole`- Untuk membuat peran layanan untuk bot
  - `iam:AttachRolePolicy`- Untuk melampirkan kebijakan ke peran layanan
  - `lambda:CreateFunction`- Untuk integrasi Lambda (opsional)
  - `logs:CreateLogGroup`- Untuk CloudWatch log Amazon Log
- Pemilihan Wilayah: Pilih wilayah AWS yang mendukung Amazon Lex V2 dan audiens target Anda. Daerah yang direkomendasikan meliputi:
  - US East (N. Virginia) - us-east-1
  - US West (Oregon) - us-west-2
  - Europe (Ireland) - eu-west-1
  - Asia Pacific (Sydney) - ap-southeast-2

Batas Layanan dan Kuota

Amazon Lex V2 memiliki beberapa batasan layanan yang dapat memengaruhi pengembangan dan penyebaran bot Anda:

- Batas Bot:
  - Maksimum 100 bot per akun per wilayah
  - Maksimum 100 maksud per bot
  - Maksimum 100 jenis slot per bot
  - Maksimum 200 slot per niat
- Batas Runtime:
  - Batas waktu sesi maksimum 15 menit
  - Maksimal 1.000 permintaan per detik (dapat ditingkatkan)
  - Maksimal 1.500 karakter per masukan teks
- Batas Data Pelatihan:
  - Maksimum 1.500 contoh ucapan per maksud

- Maksimum 10.000 nilai jenis slot
- Maksimum 140 karakter per contoh ucapan

> **Note**
>
> Jika Anda memerlukan batas yang lebih tinggi untuk penggunaan produksi, Anda dapat meminta peningkatan kuota melalui AWS Support Center.

Pertimbangan Biaya

Memahami harga Amazon Lex V2 membantu Anda merencanakan dan menganggarkan untuk penyebaran chatbot Anda:

- Harga Berbasis Permintaan:
  - Permintaan teks: $0,00075 per permintaan setelah 10.000 permintaan pertama per bulan
  - Permintaan ucapan: $0,004 per permintaan setelah 1.000 permintaan pertama per bulan
- Tingkat Gratis:
  - 10.000 permintaan teks per bulan untuk tahun pertama
  - 1.000 permintaan pidato per bulan untuk tahun pertama
- Biaya Tambahan:
  - Fungsi Lambda (jika digunakan): $0,20 per 1 juta permintaan+waktu komputasi
  - CloudWatch Log Amazon Log: $0,50 per GB tertelan
  - Biaya transfer data untuk integrasi eksternal
- Kiat Optimalisasi Biaya:
  - Gunakan atribut sesi untuk mengurangi panggilan API yang berlebihan
  - Menerapkan alur percakapan yang efisien untuk meminimalkan belokan
  - Pantau penggunaan melalui CloudWatch Log untuk mengidentifikasi peluang optimasi

> **Important**
>
> Untuk informasi harga saat ini, kunjungi halaman Harga Amazon Lex karena tarif dapat berubah.

Ikhtisar Integrasi Lambda

Sementara latihan ini berfokus pada konfigurasi bot Lex, fitur-fitur canggih yang dijelaskan (integrasi POS, perolehan prospek, pengoptimalan pendapatan) biasanya memerlukan fungsi Lambda untuk pemrosesan backend.

- Saat Lambda Dibutuhkan:
  - Verifikasi status pesanan terhadap database eksternal
  - Pemeriksaan inventaris waktu nyata untuk peningkatan penjualan
  - Penyimpanan dan pengambilan data pelanggan
  - Pencarian riwayat penjualan di tabel S3 menggunakan nama depan dan belakang pelanggan
  - Analisis pola pembelian untuk rekomendasi yang dipersonalisasi
  - Integrasi pemrosesan pembayaran
  - Pengiriman konfirmasi email dan SMS
- Integrasi Sejarah Penjualan S3:
  - Kueri tabel S3 menggunakan CustomerFirstName dan slot CustomerLastName
  - Analisis riwayat pembelian untuk mengidentifikasi pola dan preferensi pembelian
  - Hasilkan rekomendasi produk yang dipersonalisasi berdasarkan pembelian sebelumnya
  - Hitung nilai seumur hidup pelanggan untuk strategi penetapan harga dinamis
  - Identifikasi peluang cross-sell dari analisis korelasi pembelian
- Persyaratan Pengaturan Lambda Dasar:
  - Peran eksekusi Lambda dengan izin yang sesuai
  - Konfigurasi VPC jika mengakses sumber daya pribadi
  - Variabel lingkungan untuk konfigurasi
  - Penanganan kesalahan dan implementasi logging
- Poin Integrasi:
  - Pemenuhan niat - Proses maksud selesai
  - Validasi slot - Validasi input pengguna secara real-time
  - Manajemen dialog - Kontrol alur percakapan

> **Note**
>
> Untuk latihan ini, kita akan fokus pada konfigurasi bot Lex. Integrasi Lambda dapat ditambahkan nanti saat kebutuhan Anda berkembang. Bot akan bekerja dengan respons statis untuk tujuan pengujian dan demonstrasi.

Daftar Periksa Pra-Implementasi

Sebelum melanjutkan pembuatan bot, verifikasi bahwa Anda memiliki:

- ✓ Akun AWS aktif dengan penagihan diaktifkan
- ✓ Izin IAM yang sesuai untuk Lex dan layanan terkait
- ✓ Wilayah AWS target yang dipilih
- ✓ Batas dan kuota layanan yang ditinjau
- ✓ Implikasi biaya yang dipahami
- ✓ Pendekatan integrasi Lambda yang direncanakan (jika diperlukan)
- ✓ Akses ke AWS Management AWS Management Console

Buat SmartCommerce Bot

Untuk membuat bot layanan SmartCommerce pelanggan

1. Masuk ke AWS Management Console dan buka konsol Amazon Lex di https://console.aws.amazon.com/lex/.
2. Pilih Buat bot.
3. Untuk metode Creation, pilih Create a blank bot.
4. Di bagian konfigurasi Bot:
   - Nama bot: **`SmartCommerceBot`**
   - Keterangan: **`Advanced customer service chatbot with upselling, lead generation, and revenue optimization`**
5. Di bagian Izin, pilih Buat peran baru dengan izin Amazon Lex dasar.
6. Di bagian Children's Online Privacy Protection Act (COPPA), pilih No.

7. Di bagian batas waktu sesi, atur **`15 minutes`** untuk memungkinkan percakapan multi-putaran yang kompleks.
8. Pilih Berikutnya.
9. Tambahkan bahasa Inggris (AS) sebagai bahasa dan pilih Selesai.

Buat Jenis Slot Kustom

Untuk membuat jenis slot khusus

1. Di panel navigasi kiri, pilih jenis Slot.
2. Pilih Tambahkan jenis slot dan buat jenis slot berikut:
    a. ProductCategories:
        - Elektronika
        - Pakaian
        - Rumah
        - Buku
        - Olahraga
        - Keindahan
        - Otomotif
    b. ReturnReasons:
        - Cacat
        - Ukuran yang salah
        - Berubah Pikiran
        - Tidak seperti yang dijelaskan
        - Rusak dalam Pengiriman
        - Harga Lebih Baik Ditemukan
    c. ContactPreferences:
        - Email
        - SMS
        - Panggilan Telepon

- Pemberitahuan Dalam Aplikasi
- Mail

d. CustomerTiers:

- Perunggu
- Perak
- Emas
- Platinum

Buat Maksud Kustom

Untuk membuat CheckOrderStatus maksud

1. Di panel navigasi kiri, pilih Maksud.
2. Pilih Tambahkan maksud dan beri nama. **`CheckOrderStatus`**
3. Tambahkan contoh ucapan berikut:
   - “Apa status pesanan saya {OrderNumber}”
   - “Saya perlu memeriksa nomor pesanan {OrderNumber}”
   - “Di mana paket saya”
   - “Lacak pesanan saya {OrderNumber}”
   - “Apakah pesanan saya sudah dikirim”
   - “Kapan pesanan saya akan tiba”
   - “Saya ingin tahu tentang pengiriman saya”
4. Tambahkan slot berikut:
   - `OrderNumber`(AMAZON.Number) - Diperlukan

     Prompt: “Bisakah Anda memberikan nomor pesanan Anda?”
   - `CustomerEmail`(AMAZON.EmailAddress) - Diperlukan

     Prompt: “Untuk memverifikasi identitas Anda, berikan alamat email yang terkait dengan pesanan ini.”
5. Setel prompt konfirmasi: “Biarkan saya memeriksa pesanan # {OrderNumber} untuk {CustomerEmail}. Apakah ini benar?”

6. Atur pesan pemenuhan: “Saya menemukan pesanan Anda! Urutan # {OrderNumber} adalah {OrderStatus}. {UpsellMessage}”

Untuk membuat UpsellProducts maksud

1. Buat maksud baru bernama**`UpsellProducts`**.
2. Tambahkan contoh ucapan:
   - “Tunjukkan produk terkait”
   - “Apa lagi yang akan terjadi dengan ini”
   - “Apakah Anda memiliki kesepakatan”
   - “Saya tertarik dengan {ProductCategory} item”
   - “Apa yang dijual hari ini”
   - “Tunjukkan lebih banyak opsi”
3. Tambahkan slot:
   - `ProductCategory`(ProductCategories) - Diperlukan

     Prompt: “Jenis produk apa yang Anda minati?”
   - `Budget`(AMAZON.Number) - Opsional

     Prompt: “Berapa kisaran anggaran Anda untuk item tambahan?”

Untuk membuat CaptureLeadInfo maksud

1. Buat maksud baru bernama**`CaptureLeadInfo`**.
2. Tambahkan contoh ucapan:
   - “Saya ingin mendengar tentang produk baru”
   - “Daftarkan saya untuk pembaruan”
   - “Saya tertarik dengan {InterestCategory}”
   - “Beri saya informasi tentang penjualan”
   - “Saya ingin bergabung dengan milis Anda”
   - “Kirimi saya penawaran dan penawaran”
   - “Nama saya adalah {CustomerFirstName} {CustomerLastName}”

- “Saya {CustomerFirstName} {CustomerLastName} dan saya ingin rekomendasi yang dipersonalisasi”

3. Tambahkan slot:

   - `CustomerFirstName`(AMAZON.FirstName) - Diperlukan

     Prompt: “Apa nama depan Anda untuk layanan yang dipersonalisasi?”

   - `CustomerLastName`(AMAZON.LastName) - Diperlukan

     Prompt: “Dan nama belakang Anda sehingga saya dapat mencari riwayat pembelian Anda?”

   - `CustomerEmail`(AMAZON.EmailAddress) - Diperlukan
   - `CustomerPhone`(AMAZON.PhoneNumber) - Opsional
   - `InterestCategory`(ProductCategories) - Diperlukan
   - `ContactPreference`(ContactPreferences) - Diperlukan

Aktifkan Fitur Bertenaga AI

Untuk mengaktifkan NLU Berbantuan dan fitur AI lainnya

1. Di panel navigasi kiri, pilih Pengaturan bot.
2. Di bawah Assisted NLU, pilih Aktifkan.
3. Di bawah Generative AI, aktifkan:

   - Resolusi slot berbantuan
   - Bangunan bot deskriptif

4. Di bawah Analisis Sentimen, pilih Aktifkan.
5. Pilih Simpan.

Menguji dan Menyebarkan Bot

Untuk menguji SmartCommerce bot

1. Pilih Build untuk mengkompilasi bot Anda.
2. Setelah build selesai, pilih Test.
3. Uji alur percakapan berikut:

a. Alur Status Pesanan:

- Pengguna: “Periksa pesanan saya 12345"
- Bot: Meminta verifikasi email
- Pengguna: Menyediakan email
- Bot: Menyediakan status dan menawarkan upsell

b. Alur Upselling:

- Pengguna: “Tunjukkan penawaran”
- Bot: Meminta kategori produk
- Pengguna: “Elektronik”
- Bot: Mempersembahkan penawaran yang dipersonalisasi

c. Aliran Generasi Timbal:

- Pengguna: “Daftarkan saya untuk pembaruan”
- Bot: Mengumpulkan informasi kontak
- Pengguna: Memberikan detail
- Bot: Mengonfirmasi berlangganan dan menawarkan diskon langsung

4. Verifikasi bahwa fitur AI berfungsi:

- Uji variasi bahasa alami seperti “barang-barang saya belum tiba”
- Verifikasi analisis sentimen menyesuaikan respons bot
- Konfirmasikan resolusi slot berfungsi dengan input yang kompleks

5. Setelah pengujian selesai, pilih Publikasikan untuk menyebarkan bot Anda.

Analisis dan Optimasi Kinerja

Pantau dan optimalkan kinerja SmartCommerce bot Anda menggunakan analisis komprehensif dan pelacakan pendapatan.

Metrik Kinerja Utama

- Metrik Pendapatan:
  - Tingkat konversi Upsell berdasarkan tingkat pelanggan
  - Peningkatan nilai pesanan rata-rata per interaksi

    - Pendapatan per percakapan
    - Dampak nilai seumur hidup pelanggan
- Metrik Pembuatan Pimpinan:
    - Tingkat penangkapan timbal menurut jenis interaksi
    - Skor kualitas prospek dan tingkat konversi
    - Pendapatan monetisasi data dari penjualan agregator
    - Tingkat pertumbuhan langganan email
- Metrik Operasional:
    - Akurasi pengenalan maksud
    - Tingkat keberhasilan pengisian slot
    - Tingkat penyelesaian percakapan
    - Skor kepuasan pelanggan

Strategi Optimasi Berkelanjutan

- Pengujian A/B:
    - Uji pendekatan perpesanan upsell yang berbeda
    - Bandingkan teknik penangkapan timah agresif vs lembut
    - Optimalkan persentase discount untuk konversi maksimum
- Optimasi Machine Learning:
    - Menganalisis pola percakapan yang sukses
    - Identifikasi waktu optimal untuk penawaran upsell
    - Perbaiki algoritma klasifikasi tingkat pelanggan

Kesimpulan Latihan

Selamat! Anda telah berhasil membangun chatbot layanan pelanggan tingkat lanjut yang menunjukkan kemampuan tingkat perusahaan untuk pengoptimalan pendapatan dan keterlibatan pelanggan. SmartCommerce Bot ini menampilkan bagaimana bisnis dapat memanfaatkan fitur yang didukung AI Amazon Lex V2 untuk:

- Maksimalkan Pendapatan: Melalui strategi upselling, cross-selling, dan harga dinamis yang cerdas

- Hasilkan Prospek: Dengan secara sistematis menangkap dan memenuhi syarat informasi pelanggan untuk beberapa aliran pendapatan
- Meningkatkan Pengalaman Pelanggan: Menggunakan AI untuk memahami bahasa alami dan memberikan interaksi yang dipersonalisasi
- Optimalkan Operasi: Melalui layanan pelanggan otomatis yang menskalakan secara efisien sambil mempertahankan kualitas

Teknik yang ditunjukkan dalam latihan ini dapat diadaptasi dan diperluas untuk berbagai industri dan kasus penggunaan, memberikan dasar untuk membangun solusi AI percakapan canggih yang mendorong pertumbuhan bisnis.

Langkah Berikutnya

Untuk lebih meningkatkan SmartCommerce bot Anda, pertimbangkan untuk menerapkan:

- Integrasi Lanjutan:
  - Integrasi sistem CRM untuk profil pelanggan yang lengkap
  - Koneksi sistem manajemen inventaris untuk ketersediaan waktu nyata
  - Integrasi pemrosesan pembayaran untuk transaksi tanpa batas
- Penerapan Multi-Saluran:
  - Integrasi widget obrolan situs web
  - Koneksi platform media sosial
  - Implementasi antarmuka suara
- Analisis Tingkat Lanjut:
  - Pengembangan dasbor khusus untuk metrik bisnis
  - Analisis prediktif untuk perilaku pelanggan
  - Pelacakan ROI dan pemodelan atribusi

Praktik Terbaik untuk Memulai

Prinsip Desain Percakapan

Mengikuti prinsip-prinsip desain percakapan ini sejak awal akan membantu Anda membangun chatbot Amazon Lex V2 yang lebih efektif, dapat dipelihara, dan ramah pengguna yang menyediakan interaksi alami.

Prinsip Desain Inti

- Mulai dengan Sasaran Pengguna - Rancang bot Anda di sekitar apa yang ingin dicapai pengguna, bukan apa yang dapat dilakukan sistem Anda. Fokus pada perjalanan pengguna dan hasil yang diinginkan.
- Gunakan Bahasa Alami - Tulis petunjuk dan tanggapan secara percakapan. Hindari jargon teknis dan bicaralah seperti keinginan manusia yang membantu.
- Berikan Opsi yang Jelas - Saat pengguna macet, tawarkan contoh spesifik tentang apa yang dapat mereka katakan daripada teks bantuan umum.
- Keep It Simple - Mulailah dengan fungsionalitas dasar dan secara bertahap tambahkan kompleksitas. Pengguna harus dapat menyelesaikan tugas-tugas umum dengan cepat.
- Tangani Kesalahan dengan Anggun - Ketika bot tidak mengerti, berikan panduan yang bermanfaat daripada hanya mengatakan “Saya tidak mengerti.”
- Konfirmasikan Tindakan Penting - Selalu konfirmasi sebelum mengambil tindakan yang tidak dapat dengan mudah dibatalkan, seperti menempatkan pesanan atau menghapus informasi.
- Menyediakan Rute Escape - Selalu berikan pengguna cara untuk memulai dari awal, mendapatkan bantuan, atau terhubung dengan manusia saat dibutuhkan.

Praktik Terbaik Alur Percakapan

- Tetapkan Harapan yang Jelas - Biarkan pengguna tahu apa yang bot bisa dan tidak bisa lakukan di awal percakapan.
- Gunakan Pengungkapan Progresif - Mintalah informasi satu per satu daripada membuat pengguna kewalahan dengan banyak pertanyaan.
- Berikan Konteks - Ingatkan pengguna informasi apa yang telah Anda kumpulkan dan apa yang masih Anda butuhkan.
- Membuat Koreksi Mudah - Memungkinkan pengguna untuk mengoreksi informasi tanpa memulai dari awal sepenuhnya.

Kasus dan Contoh Penggunaan Dunia Nyata

Contoh praktis ini menunjukkan cara menerapkan prinsip desain percakapan ke skenario umum yang dihadapi pengguna Amazon Lex V2 baru.

Kasus Penggunaan 1: Pemesanan Janji Temu

Skenario: Kantor medis ingin mengotomatiskan penjadwalan janji temu.

Tantangan: Pengguna perlu memberikan beberapa informasi (jenis layanan, tanggal, waktu, info kontak) dan mungkin ingin mengubah detailnya.

Pendekatan Solusi:

- Mulai Luas: “Jenis janji apa yang ingin Anda jadwalkan?” (gigi, pemeriksaan mata, konsultasi)
- Persempit: “Kapan Anda lebih suka janji temu gigi Anda?” (Terima masukan fleksibel seperti “minggu depan” atau “Jumat sore”)
- Konfirmasi dan Tawarkan Perubahan: “Saya sudah menjadwalkan untuk pembersihan gigi pada hari Jumat, 15 Maret pukul 2 siang. Apakah ini berhasil untukmu?”
- Menangani Perubahan: Jika pengguna mengatakan “Bisakah kita melakukan 3 PM sebagai gantinya?” , perbarui waktu tanpa memulai ulang seluruh proses.

Teknik Kunci:

- Gunakan AMAZON.Date dan AMAZON.Time untuk date/time masukan yang fleksibel
- Buat jenis slot khusus untuk jenis janji temu
- Gunakan petunjuk konfirmasi sebelum menyelesaikan pemesanan

Kasus Penggunaan 2: Permintaan Status Pesanan

Skenario: Perusahaan e-commerce ingin pelanggan memeriksa status pesanan tanpa menelepon dukungan.

Tantangan: Pengguna mungkin tidak memiliki nomor pesanan mereka, atau mungkin bertanya dengan cara yang berbeda.

Pendekatan Solusi:

- Beberapa Poin Masuk: Terima “Di mana pesanan saya?” , “Lacak paket saya”, atau “Status pesanan”
- Identifikasi Fleksibel: “Saya dapat membantu Anda melacak pesanan Anda. Apakah Anda memiliki nomor pesanan Anda, atau apakah Anda lebih suka menggunakan alamat email Anda?”

- Panduan Bermanfaat: “Nomor pesanan Anda biasanya ada di email konfirmasi Anda dan dimulai dengan 'ORD-'”
- Hasil yang Jelas: “Pesanan Anda #ORD -12345 dikirim kemarin dan akan tiba besok jam 8 malam. Apakah Anda ingin melacak detailnya?”

Teknik Kunci:

- Gunakan jenis slot bawaan seperti AMAZON. AlphaNumeric untuk nomor pesanan
- Menyediakan beberapa cara untuk mengidentifikasi pesanan (email, telepon, nomor pesanan)
- Berikan informasi yang jelas dan dapat ditindaklanjuti dalam tanggapan

Kasus Penggunaan 3: FAQ dan Support

Skenario: Perusahaan perangkat lunak ingin menangani pertanyaan dukungan umum secara otomatis.

Tantangan: Pengguna mengajukan pertanyaan yang sama dengan berbagai cara, dan beberapa masalah memerlukan bantuan manusia.

Pendekatan Solusi:

- Pengenalan Maksud Luas: Latih maksud untuk mengenali variasi seperti “Saya tidak bisa masuk”, “Masalah masuk”, “Kata sandi tidak berfungsi”
- Pemecahan Masalah Terpandu: “Mari kita coba beberapa langkah cepat. Pertama, sudahkah Anda mencoba mengatur ulang kata sandi Anda?”
- Jalur Eskalasi: “Jika langkah-langkah ini tidak berhasil, saya dapat menghubungkan Anda dengan tim dukungan kami. Apakah Anda ingin saya melakukan itu?”
- Tindak lanjut: “Apakah ini menyelesaikan masalah Anda? Anda selalu dapat mengatakan 'bantuan' jika Anda membutuhkan bantuan dengan sesuatu yang lain.

Teknik Kunci:

- Gunakan banyak contoh ucapan untuk menangkap berbagai cara pengguna mengekspresikan masalah
- Memberikan step-by-step bimbingan daripada penjelasan panjang
- Selalu tawarkan jalan menuju dukungan manusia ketika bantuan otomatis tidak cukup

Menangani Skenario Kompleks

Bahkan bot sederhana menghadapi situasi yang kompleks. Berikut cara menanganinya dengan anggun:

- Saat Pengguna Berubah Pikiran:
  - Pengguna: “Sebenarnya, saya ingin mawar bukan tulip”
  - Bot: “Tidak masalah! Saya telah memperbarui pesanan Anda ke mawar. Mawar Anda akan siap untuk diambil pada hari Jumat pukul 2 siang. Apakah itu benar?”
- Ketika informasi tidak jelas:
  - Pengguna: “Saya butuh bunga untuk minggu depan”
  - Bot: “Saya akan dengan senang hati membantu! Hari mana minggu depan yang paling cocok untuk Anda? Anda bisa mengatakan sesuatu seperti 'Senin' atau 'Selasa depan'.
- Ketika Bot Tidak Mengerti:
  - Pengguna: “Saya ingin yang ungu dengan batang panjang”
  - Bot: “Saya ingin memastikan saya mendapatkan ini dengan benar. Apakah Anda mencari bunga ungu? Kami memiliki mawar ungu, tulip ungu, dan iris ungu yang tersedia.”
- Saat Pengguna Perlu Memulai Kembali:
  - Pengguna: “Batalkan ini, saya ingin memulai dari awal”
  - Bot: “Tentu saja! Aku telah menghapus pesanan Anda saat ini. Jenis bunga apa yang ingin kamu pesan hari ini?”

Contoh bot

Bagian ini memberikan contoh bot Amazon Lex yang dapat Anda gunakan sebagai titik awal untuk aplikasi Anda sendiri.

Topik

- Asisten agen call center

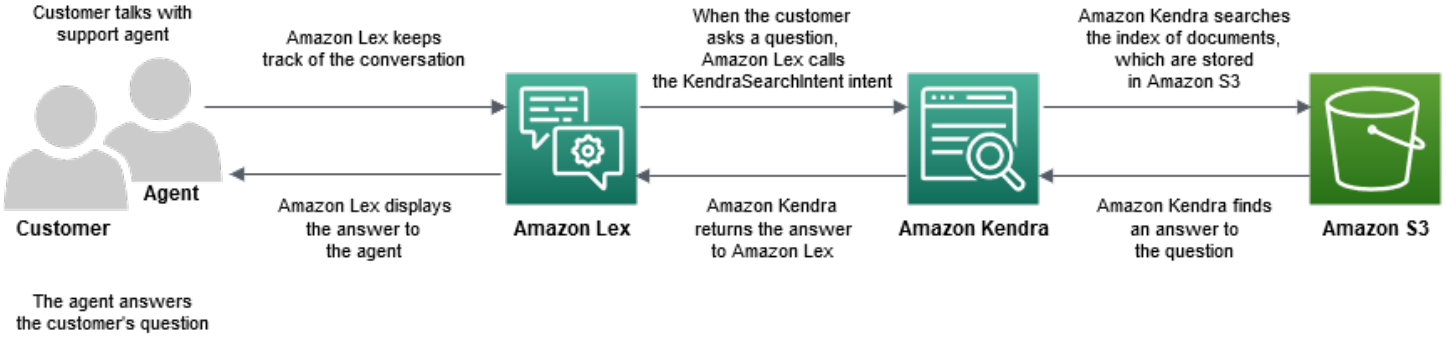
Asisten agen call center

Dalam tutorial ini, Anda menggunakan Amazon Lex V2 dengan Amazon Kendra untuk membangun bot bantuan agen yang membantu agen dukungan pelanggan dan mempublikasikannya sebagai aplikasi web. Amazon Kendra adalah layanan pencarian perusahaan yang menggunakan pembelajaran mesin untuk mencari dokumen untuk menemukan jawaban. Untuk informasi selengkapnya tentang Amazon Kendra, lihat Panduan Pengembang Amazon Kendra.

Bot Amazon Lex V2 banyak digunakan di call center sebagai titik kontak pertama bagi pelanggan. Bot seringkali mampu menyelesaikan pertanyaan pelanggan. Ketika bot tidak dapat menjawab pertanyaan, itu mentransfer percakapan ke karyawan dukungan pelanggan.

Dalam tutorial ini, kami membuat bot Amazon Lex V2 yang digunakan agen untuk menjawab pertanyaan pelanggan secara real time. Dengan membaca jawaban yang diberikan bot, agen terhindar dari mencari jawaban secara manual.

Bot dan aplikasi web yang Anda buat dalam tutorial ini membantu agen merespons pelanggan secara efisien dan akurat dengan menyediakan sumber daya yang tepat dengan cepat. Diagram berikut menunjukkan bagaimana aplikasi web bekerja.



Seperti yang ditunjukkan diagram, indeks dokumen Amazon Kendra disimpan dalam bucket Amazon Simple Storage Service (Amazon S3). Jika Anda belum memiliki bucket S3, Anda dapat mengaturnya saat membuat indeks Amazon Kendra. Selain Amazon S3, Anda akan menggunakan Amazon Cognito untuk tutorial ini. Amazon Cognito mengelola izin untuk menyebarkan bot sebagai aplikasi web.

Dalam tutorial ini, Anda membuat indeks Amazon Kendra yang memberikan jawaban atas pertanyaan pelanggan, membuat bot dan menambahkan maksud yang memungkinkannya menyarankan jawaban berdasarkan percakapan dengan pelanggan, mengatur Amazon Cognito untuk mengelola izin akses, dan menyebarkan bot sebagai aplikasi web.

Perkiraan waktu: 75 menit

Perkiraan biaya: $2,50 per jam untuk indeks Amazon Kendra dan $0,75 untuk 1000 permintaan Amazon Lex V2. Indeks Amazon Kendra Anda terus berjalan setelah Anda selesai dengan latihan ini. Pastikan untuk menghapusnya untuk menghindari biaya yang tidak perlu.

Catatan: Pastikan Anda memilih Wilayah AWS yang sama untuk semua layanan yang digunakan dalam tutorial ini.

Topik

- Langkah 1: Buat Indeks Amazon Kendra
- Langkah 2: Buat Bot Amazon Lex V2
- Langkah 3: Tambahkan Kustom dan Built-in Maksud
- Langkah 4: Siapkan Amazon Cognito
- Langkah 5: Menyebarkan Bot Anda sebagai Aplikasi Web
- Langkah 6: Gunakan Bot

Langkah 1: Buat Indeks Amazon Kendra

Mulailah dengan membuat indeks dokumen Amazon Kendra yang menjawab pertanyaan pelanggan. Indeks menyediakan API pencarian untuk kueri klien. Anda membuat indeks dari dokumen sumber. Amazon Kendra mengembalikan jawaban yang ditemukannya dalam dokumen yang diindeks ke bot, yang menampilkannya kepada agen.

Kualitas dan keakuratan tanggapan yang disarankan oleh Amazon Kendra bergantung pada dokumen yang Anda indeks. Dokumen harus menyertakan file yang sering diakses oleh agen dan

harus disimpan dalam ember S3. Anda dapat mengindeks data tidak terstruktur dan semi-terstruktur dalam.html, Microsoft Office (.doc, .ppt), PDF, dan format teks.

Untuk membuat indeks Amazon Kendra, lihat Memulai bucket (konsol) S3 di Panduan Pengembang Amazon Kendra.

Untuk menambahkan pertanyaan dan jawaban (FAQ) yang membantu menjawab pertanyaan pelanggan, lihat Menambahkan pertanyaan dan jawaban di Panduan Pengembang Amazon Kendra. Untuk tutorial ini, gunakan FAQ.csv file ML_ pada GitHub.

Langkah berikutnya

Langkah 2: Buat Bot Amazon Lex V2

Langkah 2: Buat Bot Amazon Lex V2

Amazon Lex V2 menyediakan antarmuka antara agen call center dan indeks Amazon Kendra. Ini melacak percakapan antara agen dan pelanggan dan memanggil `AMAZON.KendraSearchIntent` maksud berdasarkan pertanyaan yang diajukan pelanggan. Maksud adalah tindakan yang ingin dilakukan pengguna.

Amazon Kendra mencari dokumen yang diindeks dan mengembalikan jawaban ke Amazon Lex V2 yang ditampilkan di bot. Jawaban ini hanya terlihat oleh agen.

Untuk membuat bot asisten agen

1. Masuk ke AWS Management Console dan buka konsol Amazon Lex di https://console.aws.amazon.com/lex/.
2. Di panel navigasi, pilih Bots.
3. Pilih Buat.
4. Pilih Bot khusus dan konfigurasikan bot.
    a. Nama bot — Masukkan nama yang menunjukkan tujuan bot, seperti**`AgentAssistBot`**.
    b. Suara keluaran - Pilih Tidak Ada.
    c. Waktu tunggu sesi — Masuk**5**.
    d. COPPA — Pilih No.
5. Pilih Buat. Setelah membuat bot, Amazon Lex V2 menampilkan tab editor bot.

Langkah berikutnya

Langkah 3: Tambahkan Kustom dan Built-in Maksud

Langkah 3: Tambahkan Kustom dan Built-in Maksud

Intent mewakili tindakan yang agen call center ingin bot untuk melakukan. Dalam hal ini, agen ingin bot menyarankan tanggapan dan sumber daya yang bermanfaat berdasarkan percakapan agen dengan pelanggan.

Amazon Lex V2 memiliki dua jenis maksud: maksud khusus dan maksud bawaan. `AMAZON.KendraSearchIntent`adalah niat bawaan. Bot menggunakan `AMAZON.KendraSearchIntent` maksud untuk menanyakan indeks dan menampilkan respons yang disarankan oleh Amazon Kendra.

Bot dalam contoh ini tidak memerlukan maksud khusus. Namun, untuk membangun bot, Anda harus membuat setidaknya satu maksud khusus dengan setidaknya satu contoh ucapan. Maksud ini hanya diperlukan untuk membangun bot asisten agen Anda. Itu tidak melakukan fungsi lain. Ucapan untuk maksud tersebut tidak boleh menjawab pertanyaan apa pun yang mungkin diajukan pelanggan. Ini memastikan bahwa `AMAZON.KendraSearchIntent` dipanggil untuk menjawab pertanyaan pelanggan. Untuk informasi selengkapnya, lihat AMAZON.KendraSearchIntent.

Untuk membuat maksud kustom yang diperlukan

1. Pada halaman Memulai dengan bot Anda, pilih Buat maksud.
2. Untuk Tambah maksud, pilih Buat maksud.
3. Di kotak dialog Create intent, masukkan nama deskriptif untuk intent, seperti. **`RequiredIntent`**
4. Untuk Sampel ucapan, masukkan ucapan deskriptif, seperti. **`Required utterance`**
5. Pilih Simpan maksud.

Untuk menambahkan **`AMAZON.KendraSearchIntent`**pesan maksud dan respons

1. Di panel navigasi, pilih tanda plus (+) di sebelah Maksud.
2. Pilih Cari maksud yang ada.
3. Di kotak Maksud pencarian, masukkan**`AMAZON.KendraSearchIntent`**, lalu pilih dari daftar.
4. Berikan maksud nama deskriptif, seperti**`AgentAssistSearchIntent`**, lalu pilih Tambah.
5. Di editor maksud, pilih kueri Amazon Kendra untuk membuka opsi kueri.

6. Pilih indeks yang ingin dicari maksud,
7. Di bagian Respons, tambahkan tiga pesan berikut ke grup pesan.

```
I found an answer for the customer query: ((x-amz-lex:kendra-search-response-
question_answer-question-1)) and the answer is ((x-amz-lex:kendra-search-response-
question_answer-answer-1)).
I found an excerpt from a helpful document: ((x-amz-lex:kendra-search-response-
document-1)).
I think this answer will help the customer: ((x-amz-lex:kendra-search-response-
answer-1)).
```

8. Pilih Simpan maksud.
9. Pilih Build untuk membangun bot.

Langkah berikutnya

Langkah 4: Siapkan Amazon Cognito

Langkah 4: Siapkan Amazon Cognito

Untuk mengelola izin dan pengguna untuk aplikasi web, Anda perlu mengatur Amazon Cognito. Amazon Cognito memastikan bahwa aplikasi web aman dan memiliki kontrol akses. Amazon Cognito menggunakan kumpulan identitas untuk menyediakan kredensi AWS yang memberi pengguna Anda akses ke layanan AWS lainnya. Untuk tutorial ini, ia menyediakan akses ke Amazon Lex V2.

Saat membuat kumpulan identitas, Amazon Cognito memberi Anda peran AWS Identity and Access Management (IAM) untuk pengguna yang diautentikasi dan tidak diautentikasi. Anda mengubah peran IAM dengan menambahkan kebijakan yang memberikan akses ke Amazon Lex V2.

Untuk mengatur Amazon Cognito

1. Masuk ke AWS Management Console dan buka konsol Amazon Cognito di. https://console.aws.amazon.com/cognito/
2. Pilih Kelola Kolam Identitas.
3. Pilih Buat kolam identitas baru.
4. Konfigurasikan kumpulan identitas.
    a. Nama kolam identitas — Masukkan nama yang menunjukkan tujuan kumpulan, seperti**`BotPool`**.

b. Di bagian Identitas tidak diautentikasi, pilih Aktifkan akses ke identitas yang tidak diautentikasi.

5. Pilih Buat kolam.
6. Pada Identifikasi peran IAM yang akan digunakan dengan halaman kumpulan identitas baru Anda, pilih Lihat Detail.
7. Catat nama peran IAM. Anda akan memodifikasinya nanti.
8. Pilih Izinkan.
9. Pada halaman Memulai dengan Amazon Cognito, untuk Platform, pilih. JavaScript
10. Di bagian Dapatkan AWS Credentials, temukan dan rekam ID kumpulan Identity.
11. Untuk mengizinkan akses ke Amazon Lex V2, ubah peran IAM yang diautentikasi dan tidak diautentikasi.

    a. Masuk ke AWS Management Console dan buka konsol IAM di https://console.aws.amazon.com/iam/.

    b. Di panel navigasi, di bawah Manajemen Akses, pilih Peran.

    c. Di kotak pencarian, masukkan nama peran IAM yang diautentikasi dan pilih kotak centang di sebelahnya.

        i. Pilih Lampirkan kebijakan.

        ii. Di kotak pencarian, masukkan **`AmazonLexRunBotsOnly`** dan pilih kotak centang di sebelahnya.

        iii. Pilih Lampirkan kebijakan.

    d. Masukkan nama peran IAM yang tidak diautentikasi di kotak pencarian dan pilih kotak centang di sebelahnya.

        i. Pilih Lampirkan kebijakan.

        ii. Di kotak pencarian, masukkan **`AmazonLexRunBotsOnly`** dan pilih kotak centang di sebelahnya.

        iii. Pilih Lampirkan kebijakan.

Langkah berikutnya

Langkah 5: Menyebarkan Bot Anda sebagai Aplikasi Web

Langkah 5: Menyebarkan Bot Anda sebagai Aplikasi Web

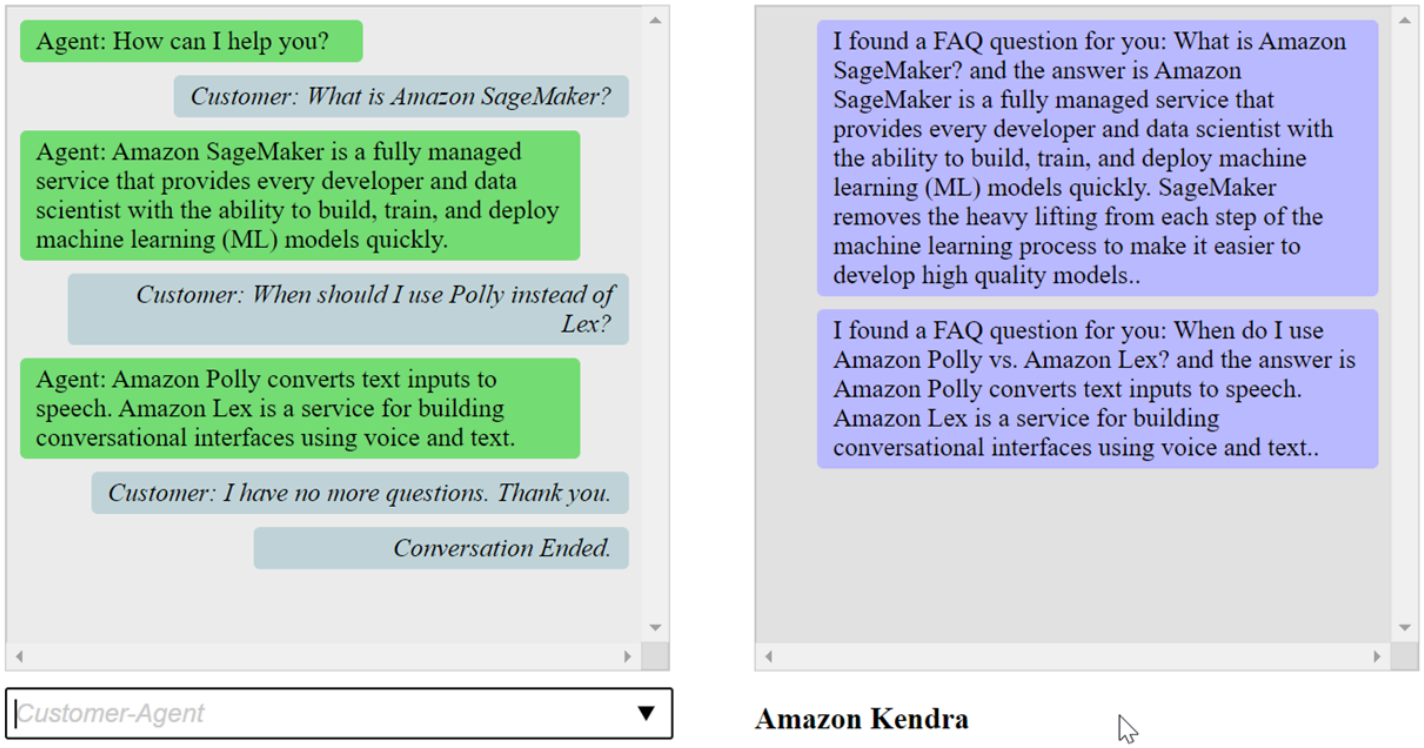
Untuk menyebarkan bot Anda sebagai aplikasi web

1. Unduh repositori di https://github.com/awsdocs/amazon-lex-developer-guide/blob/master/example_apps/agent_assistance_bot/ke komputer Anda.
2. Arahkan ke repositori yang diunduh dan buka file index.html di editor.
3. Buat perubahan berikut.
   a. Di `AWS.config.credentials` bagian ini, masukkan nama Wilayah dan ID kumpulan identitas Anda.
   b. Di `Amazon Lex V2 runtime parameters` bagian tersebut, masukkan nama bot.
   c. Simpan file tersebut.

Langkah 6: Gunakan Bot

Untuk tujuan demo, Anda memberikan masukan ke bot sebagai pelanggan dan sebagai agen. Untuk membedakan antara keduanya, pertanyaan yang diajukan oleh pelanggan dimulai dengan “Pelanggan:” dan jawaban yang diberikan oleh agen dimulai dengan “Agen:”. Anda dapat memilih dari menu input yang disarankan.

Jalankan aplikasi web Anda dengan membuka `index.html` untuk terlibat dalam percakapan dengan bot Anda seperti ini:.



`pushChat()`Fungsi dalam file index.html dijelaskan di bawah ini.

```
            var endConversationStatement = "Customer: I have no more questions. Thank
you."
            // If the agent has to send a message, start the message with 'Agent'
            var inputText = document.getElementById('input');
            if (inputText && inputText.value && inputText.value.trim().length > 0 &&
inputText.value[0]=='Agent') {
                showMessage(inputText.value, 'agentRequest','conversation');
                inputText.value = "";
            }
            // If the customer has to send a message, start the message with 'Customer'
            if(inputText && inputText.value && inputText.value.trim().length > 0 &&
inputText.value[0]=='Customer') {
                // disable input to show we're sending it
                var input = inputText.value.trim();
                inputText.value = '...';
                inputText.locked = true;
                customerInput = input.substring(2);
```

```
                // Send it to the Lex runtime
                var params = {
                    botAlias: '$LATEST',
                    botName: 'KendraTestBot',
                    inputText: customerInput,
                    userId: lexUserId,
                    sessionAttributes: sessionAttributes
                };

                showMessage(input, 'customerRequest', 'conversation');
                if(input== endConversationStatement){
                    showMessage('Conversation
Ended.','conversationEndRequest','conversation');
                }
                lexruntime.postText(params, function(err, data) {
                    if (err) {
                        console.log(err, err.stack);
                        showMessage('Error:  ' + err.message + ' (see console for
details)', 'lexError', 'conversation1')
                    }

                    if (data &&input!=endConversationStatement) {
                        // capture the sessionAttributes for the next cycle
                        sessionAttributes = data.sessionAttributes;

                            showMessage(data, 'lexResponse', 'conversation1');
                    }
                    // re-enable input
                    inputText.value = '';
                    inputText.locked = false;
                });
            }
            // we always cancel form submission
            return false;
```

Saat Anda memberikan masukan sebagai pelanggan, API runtime Amazon Lex V2 mengirimkannya ke Amazon Lex V2.

`showMessage(daText, senderRequest, displayWindow)`Fuction menampilkan percakapan antara agen dan pelanggan di jendela obrolan. Tanggapan yang disarankan oleh Amazon Kendra ditampilkan di jendela yang berdekatan. Percakapan berakhir ketika pelanggan mengatakan **“I have no more questions. Thank you.”**

Catatan: Harap hapus indeks Amazon Kendra Anda saat tidak digunakan.

Bekerja dengan bot Amazon Lex V2

Anda membuat bot Amazon Lex V2 untuk berinteraksi dengan pengguna Anda untuk memperoleh informasi guna menyelesaikan tugas. Misalnya, Anda dapat membuat bot yang mengumpulkan informasi yang diperlukan untuk memesan karangan bunga atau memesan kamar hotel.

Untuk membangun bot, Anda memerlukan informasi berikut:

1. Bahasa yang digunakan bot untuk berinteraksi dengan pelanggan. Anda dapat memilih satu atau lebih bahasa, setiap bahasa berisi maksud independen, slot, dan jenis slot.
2. Maksud, atau tujuan, yang bot membantu pengguna memenuhi. Bot dapat berisi satu atau lebih maksud, seperti memesan bunga, atau memesan hotel dan mobil sewaan. Anda perlu memutuskan pernyataan, atau ucapan, yang dibuat pengguna untuk memulai maksud.
3. Informasi, atau slot, yang perlu Anda kumpulkan dari pengguna untuk memenuhi niat. Misalnya, Anda mungkin perlu mendapatkan jenis bunga dari pengguna atau tanggal mulai reservasi hotel. Anda perlu menentukan satu atau lebih petunjuk yang digunakan Amazon Lex V2 untuk mendapatkan nilai slot dari pengguna.
4. Jenis slot yang Anda butuhkan dari pengguna. Anda mungkin perlu membuat jenis slot khusus, seperti daftar bunga yang dapat dipesan pengguna, atau Anda dapat menggunakan jenis slot bawaan, seperti menggunakan jenis `AMAZON.Date` slot untuk tanggal mulai reservasi.
5. Interaksi pengguna mengalir di dalam dan di antara maksud. Anda dapat mengonfigurasi alur percakapan untuk menentukan interaksi antara pengguna dan bot setelah intent dipanggil. Anda dapat membuat fungsi Lambda untuk memvalidasi dan memenuhi intent.

Topik

- Perubahan alur percakapan di Amazon Lex V2
- Berbagai cara untuk membuat bot dengan Amazon Lex V2
- Menambahkan bahasa baru ke bot Amazon Lex V2
- Menambahkan maksud
- Menambahkan jenis slot
- Menguji bot menggunakan konsol

> **Note**
>
> Pada 17 Agustus 2022, Amazon Lex V2 merilis perubahan pada cara percakapan dikelola dengan pengguna. Perubahan ini memberi Anda kontrol lebih besar atas jalur yang diambil pengguna melalui percakapan. Untuk informasi selengkapnya, lihat Perubahan alur percakapan di Amazon Lex V2. Bot yang dibuat sebelum 17 Agustus 2022 tidak mendukung pesan kait kode dialog, menyetel nilai, mengonfigurasi langkah selanjutnya, dan menambahkan kondisi.

Perubahan alur percakapan di Amazon Lex V2

Pada 17 Agustus 2022 Amazon Lex V2 merilis perubahan pada cara percakapan dikelola dengan pengguna. Perubahan ini memberi Anda kontrol lebih besar atas jalur yang diambil pengguna melalui percakapan.

Sebelum perubahan, Amazon Lex V2 mengelola percakapan dengan memunculkan slot berdasarkan prioritas mereka dalam niat. Anda dapat mengubah perilaku ini secara dinamis dan mengubah jalur percakapan berdasarkan input pengguna dengan menggunakan fungsi DialogAction Lambda. Ini dapat dilakukan dengan melacak keadaan percakapan saat ini dan secara terprogram memutuskan apa yang harus dilakukan selanjutnya berdasarkan status sesi.

Dengan perubahan ini, Anda dapat membuat jalur percakapan dan cabang bersyarat menggunakan konsol Amazon Lex V2 atau APIs tanpa menggunakan fungsi Lambda. Amazon Lex V2 melacak status percakapan dan mengontrol apa yang harus dilakukan selanjutnya berdasarkan kondisi yang ditentukan saat bot dibuat. Ini memungkinkan Anda untuk dengan mudah membuat percakapan yang kompleks saat merancang bot Anda.

Perubahan ini memberi Anda kendali penuh atas percakapan dengan pelanggan Anda. Namun, Anda tidak diharuskan untuk menentukan jalur. Jika Anda tidak menentukan jalur percakapan, Amazon Lex V2 akan membuat jalur default berdasarkan prioritas slot dalam maksud Anda. Anda dapat terus menggunakan fungsi Lambda untuk menentukan jalur percakapan secara dinamis. Dalam skenario seperti itu, percakapan dilanjutkan berdasarkan status sesi yang dikonfigurasi dalam fungsi Lambda.

Pembaruan ini memberikan yang berikut:

- Pengalaman konsol baru untuk membuat bot dengan alur percakapan yang kompleks.
- Pembaruan yang ada APIs untuk membuat bot untuk mendukung alur percakapan baru.

- Respons awal untuk mengirim pesan pada pemanggilan maksud.
- Tanggapan baru untuk elisitasi slot, pemanggilan Lambda sebagai pengait kode dialog dan konfirmasi.
- Kemampuan untuk menentukan langkah selanjutnya di setiap giliran percakapan.
- Evaluasi kondisi untuk merancang beberapa jalur percakapan.
- Pengaturan nilai slot dan atribut sesi kapan saja selama percakapan.

Perhatikan hal berikut untuk bot lama:

- Bot yang dibuat sebelum 17 Agustus 2022 terus menggunakan mekanisme lama untuk mengelola arus percakapan. Bot yang dibuat setelah tanggal tersebut menggunakan cara baru manajemen alur percakapan.
- Bot baru yang dibuat melalui impor setelah 17 Agustus 2022 menggunakan manajemen alur percakapan baru. Impor pada bot yang ada terus menggunakan cara lama manajemen percakapan.
- Untuk mengaktifkan manajemen alur percakapan baru untuk bot yang dibuat sebelum 17 Agustus 2022, ekspor bot, lalu impor bot menggunakan nama bot baru. Bot yang baru dibuat dari impor menggunakan manajemen alur percakapan baru.

Perhatikan hal berikut untuk bot baru yang dibuat setelah 17 Agustus 2022:

- Amazon Lex V2 mengikuti alur percakapan yang ditentukan persis seperti yang dirancang untuk memberikan pengalaman yang diinginkan. Anda harus mengonfigurasi semua cabang alur untuk menghindari jalur percakapan default selama runtime.
- Langkah-langkah percakapan mengikuti hook kode harus sepenuhnya dikonfigurasi, karena langkah-langkah yang tidak lengkap dapat menyebabkan kegagalan bot. Kami menyarankan Anda memvalidasi bot yang dibuat sebelum 17 Agustus 2022, karena untuk bot ini, tidak ada validasi otomatis langkah-langkah percakapan mengikuti kait kode.

Berbagai cara untuk membuat bot dengan Amazon Lex V2

Anda dapat membuat bot dengan Amazon Lex V2 dengan cara berikut:

1. Gunakan konsol Amazon Lex V2 untuk membuat bot menggunakan antarmuka situs web. Untuk informasi selengkapnya, lihat Membuat bot menggunakan konsol Amazon Lex V2.

2. Gunakan Descriptive Bot Builder untuk membuat bot menggunakan kemampuan AI generatif Amazon Bedrock. Untuk informasi selengkapnya, lihat Gunakan deskripsi untuk membangun bot di Lex V2 dengan pembuat bot deskriptif.
3. Gunakan templat bot untuk membuat bot yang telah dikonfigurasi sebelumnya yang cocok dengan kasus penggunaan bisnis umum. Untuk informasi selengkapnya, lihat Membuat bot Amazon Lex V2 menggunakan template.
4. Gunakan AWS SDK untuk membuat bot menggunakan operasi API.
5. Gunakan perancang Chatbot Otomatis untuk membuat bot menggunakan transkrip obrolan yang ada antara agen dan pelanggan. Untuk informasi selengkapnya, lihat Membuat bot Amazon Lex V2 menggunakan Desainer Chatbot Otomatis.
6. Impor definisi bot yang ada. Untuk informasi selengkapnya, lihat Mengimpor bot di Lex V2.
7. Gunakan CloudFormation untuk membuat bot. Lihat informasi yang lebih lengkap di Membuat sumber daya Amazon Lex V2 dengan AWS CloudFormation.

Topik

- Membuat bot menggunakan konsol Amazon Lex V2
- Membuat bot Amazon Lex V2 menggunakan template
- Membuat bot Amazon Lex V2 menggunakan Desainer Chatbot Otomatis

Membuat bot menggunakan konsol Amazon Lex V2

Anda membuat bot Amazon Lex V2 untuk berinteraksi dengan pengguna Anda untuk memperoleh informasi guna menyelesaikan tugas. Misalnya, Anda dapat membuat bot yang mengumpulkan informasi yang diperlukan untuk memesan bunga atau memesan kamar hotel. Untuk membuat bot menggunakan AWS Console, mulailah dengan menentukan nama, deskripsi, dan beberapa informasi dasar.

1. Masuk ke AWS Management Console dan buka konsol Amazon Lex di https://console.aws.amazon.com/lex/.
2. Pilih Buat bot.
3. Di bagian Metode pembuatan, pilih Tradisional dan kemudian pilih Buat bot kosong.
4. Di bagian konfigurasi Bot, beri bot nama dan deskripsi opsional.
5. Di bagian izin IAM, pilih peran AWS Identity and Access Management (IAM) yang memberikan izin Amazon Lex V2 untuk mengakses layanan AWS lainnya, seperti Amazon. CloudWatch

Anda dapat meminta Amazon Lex V2 membuat peran, atau Anda dapat memilih peran yang ada dengan CloudWatch izin.

6. Di bagian Children's Online Privacy Protection Act (COPPA), pilih respons yang sesuai.
7. Di bagian batas waktu sesi Idle, pilih durasi yang Amazon Lex V2 menyimpan sesi dengan pengguna terbuka. Amazon Lex V2 mempertahankan variabel sesi selama sesi sehingga bot Anda dapat melanjutkan percakapan dengan variabel yang sama.
8. Di bagian Pengaturan lanjutan (opsional), tambahkan tag yang membantu mengidentifikasi bot, mengontrol akses, dan memantau sumber daya.
9. Pilih Berikutnya untuk membuat bot dan pindah ke menambahkan bahasa.

Membuat bot Amazon Lex V2 menggunakan template

Amazon Lex V2 menawarkan solusi pra-bangun untuk menciptakan pengalaman dalam skala besar dan mendorong keterlibatan digital. Templat bot pra-bangun mengotomatiskan dan menstandarisasi pengalaman klien. Templat bot menyediakan alur percakapan yang siap digunakan bersama dengan data pelatihan dan permintaan dialog, untuk modalitas suara dan obrolan. Anda dapat mempercepat pengiriman solusi bot sambil mengoptimalkan sumber daya, sehingga Anda dapat fokus pada hubungan pelanggan.

Anda dapat membuat bot bawaan berdasarkan kasus penggunaan bisnis Anda. Anda dapat menggunakan CloudFormation konsol untuk memilih opsi bawaan untuk layanan terkait, seperti Amazon S3, Connect Customer, dan DynamoDB.

Saat ini, Amazon Lex V2 mendukung vertikal bisnis berikut:

- Jasa keuangan
- Pesanan eceran
- Asuransi mobil
- Telekomunikasi
- Layanan maskapai
- Template tambahan akan tersedia di masa depan.

Anda dapat membuat bot dengan template solusi bisnis yang disediakan, dan menyesuaikannya dengan kebutuhan bisnis Anda.

> **Note**
>
> Template membuat sumber daya di luar Amazon Lex V2 melalui CloudFormation tumpukan. Tumpukan mungkin perlu dimodifikasi di konsol lain seperti Lambda dan DynamoDB.

Prasyarat untuk membangun dan menerapkan template bot:

- Akun AWS
- Akses ke layanan AWS berikut:
  - Amazon Lex V2 untuk membuat bot
  - Lambda untuk fungsi login bisnis
  - DynamoDB untuk membuat tabel
  - Akses IAM untuk membuat kebijakan dan peran
  - AWS CloudFormation untuk menjalankan tumpukan
- Akses IAM dan kredensi kunci rahasia
- Connect Customer instance (opsional)

> **Note**
>
> Penggunaan layanan AWS yang berbeda menimbulkan biaya penggunaan masing-masing untuk setiap layanan.

Untuk membuat bot dari template Amazon Lex V2:

1. Masuk ke AWS Management Console dan buka konsol Amazon Lex di https://console.aws.amazon.com/lex/.
2. Pilih template Bot dari panel navigasi kiri.
3. Pilih vertikal bisnis mana yang ingin Anda gunakan untuk templat bot Anda. CATATAN: Ada 5 templat bot yang tersedia saat ini. Lebih banyak yang akan segera datang.
4. Pilih Buat untuk template yang ingin Anda gunakan. Tab terbuka di CloudFormation mana Anda dapat mengedit parameter untuk CloudFormation tumpukan. Semua opsi sudah selesai untuk template yang telah Anda pilih. Anda juga dapat mempelajari lebih lanjut tentang cara kerja templat bot dengan memilih Pelajari lebih lanjut.

5. Di CloudFormation konsol, CloudFormation buat konfigurasi default untuk setiap nilai untuk templat yang telah Anda pilih. Anda juga dapat memilih nama tumpukan Anda sendiri, CloudFormation parameter, tabel Amazon DynamoDB, dan parameter Connect Customer (opsional).
6. Di bagian bawah jendela, pilih Buat tumpukan.
7. CloudFormation memproses permintaan di latar belakang selama beberapa menit untuk mengonfigurasi bot baru Anda. CATATAN: Proses secara otomatis membuat sumber daya untuk tabel DynamoDB, alur kontak Connect Customer, dan instance Connect Customer. Anda dapat melacak kemajuan di CloudFormation konsol, lalu menavigasi kembali ke konsol Amazon Lex V2 setelah pembuatan CloudFormation tumpukan selesai.
8. Jika berhasil dibangun, sebuah pesan muncul dan Anda dapat memilih Pergi ke daftar bot untuk pergi ke halaman Bots, di mana Anda menemukan bot baru Anda yang siap untuk pengujian dan penggunaan Anda.

Mengkonfigurasi templat bot Anda

Fungsi Lambda - Templat bot secara otomatis membuat fungsi Lambda yang diperlukan untuk penerapan Anda. Jika beberapa bot adalah bagian dari solusi template, maka beberapa fungsi Lambda tercantum dalam CloudFormation parameter. Jika Anda memiliki fungsi Lambda yang ada untuk digunakan dengan bot Anda, Anda dapat memasukkan nama fungsi Lambda khusus Anda.

Amazon DynamoDB - Templat bot secara otomatis membuat tabel DynamoDB yang diperlukan untuk memuat data kebijakan sampel Anda. Anda juga dapat memasukkan nama tabel DynamoDB kustom Anda. Tabel DynamoDB kustom Anda harus diformat dengan cara yang sama seperti tabel default yang dibuat oleh penerapan templat bot.

Connect Customer - Anda dapat mengonfigurasi instans Connect Customer Anda untuk bekerja dengan template bot baru Anda dengan memasukkan ConnectInstance ARN dan unik. ContactFlowName Dengan menggunakan Connect Customer, Anda dapat menguji bot Anda menggunakan sistem IVR dari ujung ke ujung.

Memecahkan masalah template bot Anda

- Periksa apakah Anda memiliki izin yang tepat untuk membuat template yang Anda pilih. Pengguna memerlukan CloudFormation:CreateStack izin bersama dengan izin untuk sumber daya AWS yang tercantum dalam template. Daftar sumber daya yang memerlukan izin pengguna ada di bagian bawah halaman Buat template.

- Jika template bot Anda gagal dibuat, spanduk merah di dalam konsol Amazon Lex V2 menyediakan tautan ke CloudFormation tumpukan yang bertanggung jawab untuk membuat templat. Di dalam CloudFormation konsol, Anda dapat melihat tab peristiwa untuk melihat kesalahan spesifik yang menyebabkan template gagal. Setelah Anda meninjau CloudFormation kesalahan, lihat Pemecahan Masalah CloudFormation untuk informasi selengkapnya.
- Templat bot hanya berfungsi dengan data sampel. Anda harus mengisi tabel DynamoDB dengan data Anda untuk membuat template bekerja dengan data kustom Anda.

Membuat bot Amazon Lex V2 menggunakan Desainer Chatbot Otomatis

Perancang Chatbot Otomatis membantu Anda mendesain bot dari transkrip percakapan yang ada. Ini menganalisis transkrip dan menyarankan desain awal dengan maksud dan jenis slot. Anda dapat mengulangi desain bot, menambahkan prompt, membangun, menguji, dan menyebarkan bot.

Setelah Anda membuat bot baru atau menambahkan bahasa ke bot Anda menggunakan konsol atau API Amazon Lex V2, Anda dapat mengunggah transkrip percakapan antara dua pihak. Perancang chatbot otomatis menganalisis transkrip dan menentukan maksud dan jenis slot untuk bot. Ini juga memberi label percakapan yang memengaruhi pembuatan maksud atau jenis slot tertentu untuk ulasan Anda.

Anda menggunakan konsol Amazon Lex V2 atau API untuk menganalisis transkrip percakapan dan menyarankan maksud dan jenis slot untuk bot.

> **Note**
>
> Anda hanya dapat menggunakan transkrip dalam bahasa Inggris (AS).

Anda dapat meninjau maksud dan jenis slot yang disarankan setelah perancang chatbot menyelesaikan analisis. Setelah menambahkan maksud atau jenis slot yang disarankan, Anda dapat memodifikasinya atau menghapusnya dari desain bot menggunakan konsol atau API.

Perancang chatbot otomatis mendukung file transkrip percakapan menggunakan skema Contact Lens for Connect Customer. Jika Anda menggunakan aplikasi pusat kontak yang berbeda, Anda harus mengubah transkrip percakapan ke format yang digunakan oleh desainer chatbot. Untuk informasi, lihat Format transkrip masukan.

Untuk menggunakan desainer chatbot otomatis, Anda harus mengizinkan peran IAM yang menjalankan akses desainer. Untuk kebijakan IAM tertentu, lihatMemungkinkan pengguna untuk

menggunakan Desainer Chatbot Otomatis. Untuk mengaktifkan Amazon Lex V2 mengenkripsi data keluaran dengan AWS KMS kunci opsional, Anda perlu memperbarui kunci dengan kebijakan yang ditampilkan diMemungkinkan pengguna untuk menggunakan AWS KMS kunci untuk mengenkripsi dan mendekripsi file.

> **Note**
>
> Jika Anda menggunakan a KMS key, Anda harus memberikan KMS key kebijakan, terlepas dari IAM peran yang digunakan.

Topik

- Mengimpor transkrip percakapan
- Membuat maksud dan jenis slot
- Format transkrip masukan
- Format transkrip keluaran

Mengimpor transkrip percakapan

Mengimpor transkrip percakapan adalah proses tiga langkah:

1. Siapkan transkrip untuk diimpor dengan mengonversinya ke format yang benar. Jika Anda menggunakan Contact Lens untuk Connect Customer, transkrip sudah dalam format yang benar.
2. Unggah transkrip ke bucket Amazon S3. Jika Anda menggunakan Lensa Kontak, transkrip Anda sudah ada di bucket S3.
3. Analisis transkrip menggunakan konsol Amazon Lex V2 atau operasi API. Waktu yang diperlukan untuk menyelesaikan pelatihan tergantung pada volume transkrip dan kompleksitas percakapan. Biasanya, 500 baris transkrip dianalisis setiap menit.

Masing-masing langkah ini dijelaskan di bagian berikut.

Mengimpor transkrip dari Contact Lens untuk Connect Customer

Perancang chatbot otomatis Amazon Lex V2 kompatibel dengan file transkrip Lensa Kontak. Untuk menggunakan file transkrip Lensa Kontak, Anda harus mengaktifkan Lensa Kontak dan mencatat lokasi file outputnya.

Untuk mengekspor transkrip dari Lensa Kontak

1. Aktifkan Contact Lens di instans Connect Customer Anda. Untuk petunjuknya, lihat Aktifkan Lensa Kontak untuk Connect Customer di panduan administrator Connect Customer.
2. Perhatikan lokasi bucket S3 yang Connect Customer gunakan untuk instans Anda. Untuk melihat lokasi, buka halaman Penyimpanan data di konsol Connect Customer. Untuk petunjuknya, lihat Memperbarui setelan instans di panduan Connect Customer administrator.

Setelah Anda mengaktifkan Lensa Kontak dan mencatat lokasi file transkrip Anda, buka petunjuk Analisis transkrip Anda menggunakan konsol Amazon Lex V2 untuk mengimpor dan menganalisis transkrip Anda.

Siapkan transkrip

Siapkan transkrip Anda dengan membuat file transkrip.

- Buat satu file transkrip per percakapan yang mencantumkan interaksi antara para pihak. Setiap interaksi dalam percakapan dapat menjangkau beberapa baris. Anda dapat memberikan versi percakapan yang disunting dan tidak disunting.
- File harus dalam format JSON yang ditentukan dalamFormat transkrip masukan.
- Anda harus memberikan setidaknya 1.000 giliran percakapan. Untuk meningkatkan penemuan maksud dan jenis slot Anda, Anda harus memberikan sekitar 10.000 atau lebih giliran percakapan. Perancang chatbot otomatis hanya akan memproses 700.000 putaran pertama.
- Tidak ada batasan jumlah file transkrip yang dapat Anda unggah, juga tidak ada batasan ukuran file.

Jika Anda berencana untuk memfilter transkrip yang Anda impor berdasarkan tanggal, file harus dalam struktur direktori berikut:

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

File transkrip harus berisi tanggal dalam format “yyyy-mm-dd” di suatu tempat di nama file.

Untuk mengekspor transkrip dari aplikasi pusat kontak lainnya

1. Gunakan alat aplikasi pusat kontak Anda untuk mengekspor percakapan. Percakapan harus berisi setidaknya informasi yang ditentukan dalamFormat transkrip masukan.
2. Ubah transkrip yang dihasilkan oleh aplikasi pusat kontak Anda ke format yang dijelaskan dalamFormat transkrip masukan. Anda bertanggung jawab untuk melakukan transformasi.

Kami menyediakan tiga skrip untuk menyiapkan transkrip. File tersebut adalah:

- Skrip untuk menggabungkan transkrip Lensa Kontak dengan log percakapan Amazon Lex V2. Transkrip Lensa Kontak tidak menyertakan bagian dari percakapan Connect Customer yang berinteraksi dengan bot Amazon Lex V2. Skrip ini mengharuskan log percakapan diaktifkan untuk Amazon Lex V2, dan izin yang sesuai untuk menanyakan CloudWatch log percakapan Log dan bucket S3 Lensa Kontak.
- Skrip untuk mengubah analitik panggilan Amazon Transcribe ke format input Amazon Lex V2.
- Skrip untuk mengubah transkrip obrolan Connect Customer ke format input Amazon Lex V2.

Anda dapat mengunduh skrip dari GitHub repositori ini:. https://github.com/aws-samples/amazon-lex-bot-recommendation-integration

Unggah transkrip Anda ke bucket S3

Jika Anda menggunakan Contact Lens, file transkrip Anda sudah terkandung dalam bucket S3. Untuk lokasi dan nama file file transkrip Anda, lihat Contoh file keluaran Lensa Kontak di panduan administrator Connect Customer.

Jika Anda menggunakan aplikasi pusat kontak lain dan Anda belum menyiapkan bucket S3 untuk file transkrip Anda, ikuti prosedur ini. Jika tidak, jika Anda memiliki bucket S3 yang sudah ada, setelah masuk ke konsol Amazon S3, ikuti prosedur ini yang dimulai dengan langkah 5.

Untuk mengunggah file ke bucket S3

1. Masuk ke AWS Management Console dan buka konsol Amazon S3 di. https://console.aws.amazon.com/s3/
2. Pilih Buat bucket.
3. Beri nama bucket dan pilih Region. Wilayah harus sama dengan yang Anda gunakan untuk Amazon Lex V2. Tetapkan opsi lain sesuai kebutuhan untuk kasus penggunaan Anda.

4. Pilih Buat bucket.
5. Dari daftar bucket, pilih bucket yang sudah ada atau bucket yang baru saja Anda buat
6. Pilih Unggah.
7. Tambahkan file transkrip yang ingin Anda unggah.
8. Pilih Unggah.

Analisis transkrip Anda menggunakan konsol Amazon Lex V2

Anda hanya dapat menggunakan desain bot otomatis dalam bahasa kosong. Anda dapat menambahkan bahasa baru ke bot yang ada, atau membuat bot baru.

Untuk membuat bahasa baru di bot baru

1. Masuk ke AWS Management Console dan buka konsol Amazon Lex di https://console.aws.amazon.com/lex/.
2. Pilih Buat bot
3. Pilih Mulai dengan Desainer Chatbot Otomatis. Isi informasi untuk membuat bot baru Anda.
4. Pilih Berikutnya
5. Dalam Tambahkan bahasa ke bot isi informasi untuk bahasa tersebut.
6. Di lokasi file Transkrip di bagian S3, pilih bucket S3 yang berisi file transkrip Anda dan jalur lokal ke file jika perlu.
7. Anda dapat memilih yang berikut ini secara opsional:
   - AWS KMS Kunci untuk mengenkripsi data transkrip selama pemrosesan. Jika Anda tidak memilih kunci, AWS KMS kunci layanan digunakan.
   - Untuk memfilter transkrip ke rentang tanggal tertentu. Jika Anda memilih untuk memfilter transkrip, transkrip harus berada dalam struktur folder yang benar. Untuk informasi selengkapnya, lihat Siapkan transkrip.
8. Pilih Selesai.

Tunggu Amazon Lex V2 memproses transkrip. Anda melihat pesan penyelesaian saat analisis selesai.

Cara berhenti menganalisis transkrip Anda

Jika Anda perlu menghentikan analisis transkrip yang telah Anda unggah, Anda dapat menghentikan `BotRecommendation` pekerjaan yang sedang berjalan, yang `BotRecommendationStatus` berstatus sebagai pemrosesan. Anda dapat mengklik tombol Stop processing yang ada di banner setelah mengirimkan pekerjaan dari konsol atau dengan menggunakan CLI SDK untuk API. `StopBotRecommendation` Untuk informasi selengkapnya, lihat StopBotRecommendation

Setelah memanggil`StopBotRecommendation`, internal `BotRecommendationStatus` diatur ke `Stopping` dan Anda tidak dikenakan biaya. Untuk memastikan pekerjaan telah berhenti, Anda dapat memanggil `DescribeBotRecommendation` API dan memverifikasi bahwa `BotRecommendationStatus` itu`Stopped`. Ini biasanya memakan waktu 3-4 menit.

Anda tidak dikenakan biaya untuk pemrosesan setelah `StopBotRecommendation` API dipanggil.

Membuat maksud dan jenis slot

Setelah perancang chatbot membuat maksud dan jenis slot, Anda memilih maksud dan jenis slot untuk ditambahkan ke bot Anda. Anda dapat meninjau detail setiap maksud dan jenis slot untuk membantu Anda memutuskan rekomendasi mana yang paling relevan dengan kasus penggunaan Anda.

Anda dapat mengklik nama maksud yang disarankan untuk melihat contoh ucapan dan slot yang disarankan oleh perancang chatbot. Jika memilih Tampilkan transkrip terkait, Anda juga dapat menggulir percakapan yang Anda berikan. Transkrip ini memengaruhi rekomendasi perancang chatbot tentang maksud ini. Jika Anda mengklik contoh ucapan, Anda dapat meninjau percakapan utama dan pergantian dialog yang relevan, yang memengaruhi ucapan tertentu tersebut.

Anda dapat mengklik nama jenis slot tertentu untuk melihat nilai slot yang telah direkomendasikan. Jika Anda memilih Tampilkan transkrip terkait, Anda dapat meninjau percakapan yang memengaruhi jenis slot ini, dengan prompt agen yang memunculkan jenis slot yang disorot. Jika Anda mengklik nilai jenis slot tertentu, Anda dapat meninjau percakapan utama dan pergantian dialog yang relevan yang memengaruhi nilai ini.

Untuk meninjau dan menambahkan maksud dan jenis slot

1. Masuk ke AWS Management Console dan buka konsol Amazon Lex di https://console.aws.amazon.com/lex/.
2. Dari daftar bot, pilih bot yang ingin Anda gunakan.
3. Pilih Lihat bahasa.
4. Dari daftar bahasa, pilih bahasa yang akan digunakan.

5. Dalam struktur Percakapan, pilih Tinjau.
6. Dalam daftar maksud dan jenis slot, pilih yang akan ditambahkan ke bot. Anda dapat memilih jenis maksud atau slot untuk melihat detail dan transkrip terkait.

Maksud diurutkan berdasarkan keyakinan yang dimiliki Amazon Lex V2 bahwa maksud tersebut terkait dengan transkrip yang diproses.

Format transkrip masukan

Berikut ini adalah format file input untuk menghasilkan maksud dan jenis slot untuk bot Anda. File input harus berisi bidang-bidang ini. Bidang lain diabaikan.

Format input kompatibel dengan format output dari Contact Lens for Connect Customer. Jika Anda menggunakan Lensa Kontak, Anda tidak perlu memodifikasi file transkrip Anda. Untuk informasi selengkapnya, lihat Contoh file keluaran Lensa Kontak. Jika Anda menggunakan aplikasi pusat kontak lain, Anda harus mengubah file transkrip Anda ke format ini.

```
{
    "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"
        }
    ]
```

```
}
```

Bidang berikut harus ada dalam file input:

- Peserta mengidentifikasi peserta dalam percakapan dan peran yang mereka mainkan.
- Versi Versi format file input. Selalu “1.1.0".
- ContentMetadataMenunjukkan apakah Anda menghapus informasi sensitif dari transkrip. Setel `Output` bidang ke “Mentah” jika transkrip berisi informasi sensitif.
- CustomerMetadataPengidentifikasi unik untuk percakapan.
- Transkrip Teks percakapan antar pihak dalam percakapan. Setiap giliran percakapan diidentifikasi dengan pengenal unik.

Format transkrip keluaran

Format transkrip keluaran hampir sama dengan format transkrip input. Namun itu juga mencakup beberapa metadata pelanggan dan segmen daftar bidang yang memengaruhi saran maksud dan jenis slot. Anda dapat mengunduh transkrip keluaran dari halaman Tinjauan di konsol atau menggunakan Amazon Lex V2 API. Untuk informasi selengkapnya, lihat Format transkrip masukan.

```
{
    "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— Ada dua bidang yang ditambahkan ke `CustomerMetadata` bidang, nama file input yang berisi percakapan dan format input, yang selalu “Lex”.

- InfluencingSegments— Mengidentifikasi segmen percakapan yang memengaruhi saran dari maksud atau jenis slot. ID dari maksud atau jenis slot mengidentifikasi yang spesifik yang dipengaruhi oleh percakapan.

Menambahkan bahasa baru ke bot Amazon Lex V2

Anda menambahkan satu atau beberapa bahasa dan lokal ke bot Anda untuk memungkinkannya berkomunikasi dengan pengguna dalam bahasa mereka. Anda menentukan maksud, slot, dan jenis slot secara terpisah untuk setiap bahasa sehingga ucapan, petunjuk, dan nilai slot khusus untuk bahasa tersebut.

Bot Anda harus berisi setidaknya satu bahasa.

Untuk menambahkan bahasa ke bot Anda

1. Masuk ke AWS Management Console dan buka konsol Amazon Lex di https://console.aws.amazon.com/lex/.
2. Di bagian Bots, pilih bot yang ingin Anda tambahkan bahasa.
3. Di bagian Tambahkan bahasa, klik Lihat bahasa.
4. Di bagian Semua bahasa, klik Tambahkan bahasa.
5. Di bagian Tambahkan bahasa baru, pilih Tambahkan bahasa dari awal.
6. Di bagian Detail bahasa, pilih bahasa yang ingin Anda tambahkan.
7. Jika bot Anda mendukung interaksi suara, di bagian Suara, pilih suara Amazon Polly yang digunakan Amazon Lex V2 untuk berkomunikasi dengan pengguna. Jika bot Anda tidak mendukung suara, pilih Tidak Ada.
8. Di bagian ambang batas skor kepercayaan Klasifikasi, tetapkan nilai yang digunakan Amazon Lex V2 untuk menentukan apakah maksud sudah benar. Anda dapat menyesuaikan nilai ini setelah menguji bot Anda.
9. Pilih Tambahkan.

Menambahkan maksud

Maksud adalah tujuan yang ingin dicapai pengguna Anda, seperti memesan bunga atau memesan hotel. Bot Anda harus memiliki setidaknya satu niat.

Secara default, semua bot berisi satu maksud bawaan, maksud fallback. Maksud ini digunakan ketika Amazon Lex V2 tidak mengenali maksud lain apa pun. Misalnya, jika pengguna mengatakan “Saya ingin memesan bunga” ke maksud pemesanan hotel, maksud fallback dipicu.

Untuk menambahkan maksud

1. Masuk ke AWS Management Console dan buka konsol Amazon Lex di https://console.aws.amazon.com/lex/.
2. Dari daftar bot, pilih bot yang ingin Anda tambahkan intent, lalu dari Tambah bahasa pilih Lihat bahasa.
3. Pilih bahasa untuk menambahkan intent, lalu pilih Intent.
4. Pilih Tambah maksud, beri nama maksud Anda, lalu pilih Tambah.
5. Di editor maksud, tambahkan detail maksud Anda.

    - Alur percakapan — Gunakan diagram alur percakapan untuk melihat bagaimana dialog dengan bot Anda mungkin terlihat. Anda dapat memilih bagian percakapan yang berbeda untuk beralih ke bagian editor maksud tersebut.
    - Detail maksud — Berikan maksud nama dan deskripsi untuk membantu mengidentifikasi tujuan maksud. Anda juga dapat melihat pengenal unik yang ditetapkan Amazon Lex V2 ke intent.
    - Konteks - Mengatur konteks input dan output untuk maksud. Konteks adalah variabel status yang terkait dengan maksud. Konteks keluaran diatur ketika maksud terpenuhi. Maksud dengan konteks masukan hanya dapat dikenali jika konteksnya aktif. Maksud tanpa konteks input selalu dapat dikenali.
    - Contoh ucapan - Anda harus memberikan 10 frasa atau lebih yang Anda harapkan digunakan pengguna untuk memulai maksud. Amazon Lex V2 menggeneralisasi dari frasa ini untuk mengenali bahwa pengguna ingin memulai maksud.
    - Respons awal - Pesan awal yang dikirim ke pengguna setelah intent dipanggil. Anda dapat memberikan respons, menginisialisasi nilai, dan menentukan langkah selanjutnya yang dilakukan Amazon Lex V2 untuk merespons pengguna di awal intent.
    - Slot — Tentukan slot, atau parameter, yang diperlukan untuk memenuhi maksud. Setiap slot memiliki tipe yang mendefinisikan nilai yang dapat dimasukkan dalam slot. Anda dapat memilih dari jenis slot khusus Anda, atau Anda dapat memilih jenis slot bawaan.
    - Konfirmasi — Permintaan dan tanggapan ini digunakan untuk mengkonfirmasi atau menolak pemenuhan maksud. Permintaan konfirmasi meminta pengguna untuk meninjau nilai slot.

Misalnya, “Saya sudah memesan kamar hotel untuk hari Jumat. Apakah ini benar?” Respons deklinasi dikirim ke pengguna ketika mereka menolak konfirmasi. Anda dapat memberikan tanggapan, menetapkan nilai, dan menentukan langkah berikutnya yang diambil Amazon Lex V2 sesuai dengan respons konfirmasi atau deklinasi dari pengguna.

- Pemenuhan — Respon dikirim ke pengguna selama pemenuhan. Anda dapat mengatur pembaruan kemajuan pemenuhan di awal pemenuhan dan secara berkala saat pemenuhan sedang berlangsung. Misalnya, “Saya mengubah kata sandi Anda, ini mungkin memakan waktu beberapa menit” dan “Saya masih mengerjakan permintaan Anda.” Pembaruan pemenuhan hanya digunakan untuk percakapan streaming. Anda juga dapat mengatur pesan sukses pasca-pemenuhan, pesan kegagalan, dan pesan batas waktu. Anda dapat mengirim pesan pasca-pemenuhan untuk streaming dan percakapan reguler. Misalnya, jika pemenuhan berhasil, Anda dapat mengirim “Saya telah mengubah kata sandi Anda.” Jika pemenuhan tidak berhasil, Anda dapat mengirim tanggapan dengan informasi lebih lanjut, seperti “Saya tidak dapat mengubah kata sandi Anda, hubungi meja bantuan untuk bantuan.” Jika pemenuhan memakan waktu lebih lama dari periode batas waktu yang dikonfigurasi, Anda dapat mengirim pesan yang memberi tahu pengguna, seperti “Server kami sangat sibuk saat ini. Coba permintaanmu lagi nanti.” Anda dapat memberikan respons, menetapkan nilai, dan menentukan langkah selanjutnya yang dilakukan Amazon Lex V2 untuk merespons pengguna.
- Menutup tanggapan — Respons dikirim ke pengguna setelah maksud terpenuhi dan semua pesan lainnya diputar. Misalnya, terima kasih telah memesan kamar hotel. Atau dapat meminta pengguna untuk memulai niat yang berbeda, seperti, “Terima kasih telah memesan kamar, apakah Anda ingin memesan mobil sewaan?” Anda dapat memberikan tanggapan dan mengonfigurasi tindak lanjut tindakan berikutnya setelah memenuhi maksud dan merespons dengan respons penutupan.
- Kait kode - Tunjukkan apakah Anda menggunakan AWS Lambda fungsi untuk menginisialisasi maksud dan memvalidasi input pengguna. Anda menentukan fungsi Lambda dalam alias yang Anda gunakan untuk menjalankan bot.

6. Pilih Simpan maksud untuk menyimpan intent.

> **Note**
>
> Pada 17 Agustus 2022, Amazon Lex V2 merilis perubahan pada cara percakapan dikelola dengan pengguna. Perubahan ini memberi Anda kontrol lebih besar atas jalur yang diambil pengguna melalui percakapan. Untuk informasi selengkapnya, lihat Perubahan

alur percakapan di Amazon Lex V2. Bot yang dibuat sebelum 17 Agustus 2022 tidak mendukung pesan kait kode dialog, menyetel nilai, mengonfigurasi langkah selanjutnya, dan menambahkan kondisi.

Mengkonfigurasi prompt dalam urutan tertentu

Anda dapat mengonfigurasi bot untuk memutar pesan dalam urutan yang telah ditentukan dengan mencentang kotak untuk pesan Putar secara berurutan. Jika tidak, bot memainkan pesan dan variasi dalam urutan acak.

Permintaan yang dipesan memungkinkan pesan dan variasi grup pesan untuk diputar secara berurutan di antara percobaan ulang. Anda dapat menggunakan rephrasing alternatif pesan ketika respons yang tidak valid untuk prompt diberikan oleh pengguna, atau untuk konfirmasi maksud. Hingga dua variasi pesan asli dapat diatur di setiap slot. Anda dapat memilih apakah akan memutar pesan secara berurutan atau secara acak.

Prompt yang dipesan mendukung keempat jenis pesan: teks, respons muatan khusus, SSML, dan grup kartu. Tanggapan diurutkan dalam grup pesan yang sama. Grup pesan yang berbeda bersifat independen.

Topik

- Sampel ucapan
- Struktur niat
- Membuat jalur percakapan
- Menggunakan pembangun percakapan Visual
- Maksud bawaan

Sampel ucapan

Anda membuat contoh ucapan yang merupakan variasi frasa yang Anda harapkan digunakan pengguna untuk memulai maksud. Misalnya, untuk **`BookFlight`** maksud, Anda mungkin menyertakan ucapan seperti berikut ini:

1. Saya ingin memesan penerbangan
2. Bantu aku mendapatkan penerbangan.

3. Tiket pesawat, tolong!
4. penerbangan dari {*DepartureCity*} ke {*DestinationCity*}

Anda harus memberikan 10 atau lebih contoh ucapan. Berikan sampel yang mewakili berbagai struktur kalimat dan kata-kata yang dapat diucapkan pengguna. Pertimbangkan juga kalimat yang tidak lengkap, seperti pada contoh 3 dan 4 di atas. Anda juga dapat menggunakan slot yang telah Anda tetapkan untuk maksud dalam contoh ucapan dengan membungkus kurawal kurawal di sekitar nama slot, seperti pada {} pada contoh 4. *DepartureCity* Jika Anda menyertakan nama slot dalam contoh ucapan, Amazon Lex V2 mengisi slot maksud dengan nilai yang diberikan pengguna dalam ucapan tersebut.

Berbagai contoh ucapan membantu Amazon Lex V2 menggeneralisasi untuk secara efektif mengenali bahwa pengguna ingin memulai intent.

Anda dapat menambahkan contoh ucapan di editor maksud, pembuat percakapan visual, atau dengan operasi atau API CreateIntent. UpdateIntent Anda juga dapat menghasilkan contoh ucapan secara otomatis dengan memanfaatkan kemampuan AI generatif Amazon Bedrock. Untuk informasi selengkapnya, lihat Gunakan generasi ucapan untuk menghasilkan contoh ucapan untuk pengenalan maksud.

Menggunakan editor Intent atau pembuat percakapan Visual

1. Di editor Intent, navigasikan ke bagian Sample utterances. Di pembangun percakapan Visual, temukan bagian Sampel ucapan di blok Mulai.
2. Di kotak dengan teks transparan**`I want to book a flight`**, ketikkan contoh ucapan. Pilih Tambahkan ucapan untuk menambahkan ucapan.
3. Lihat contoh ucapan yang telah Anda tambahkan dalam mode Pratinjau atau teks Biasa. Dalam teks Biasa, setiap baris adalah ucapan yang terpisah. Dalam mode Pratinjau, arahkan kursor ke ucapan untuk mengungkapkan opsi berikut:
   - Pilih kotak teks untuk mengedit ucapan.
   - Pilih tombol x di sebelah kanan kotak teks untuk menghapus ucapan.
   - Seret tombol di sebelah kiri kotak teks untuk mengubah urutan ucapan sampel.
4. Gunakan bilah pencarian di bagian atas untuk mencari melalui contoh ucapan Anda dan menu tarik-turun di sebelahnya untuk mengurutkan berdasarkan urutan Anda menambahkan ucapan atau dalam urutan abjad.

Menggunakan operasi API

1. Buat maksud baru dengan CreateIntentoperasi atau perbarui yang sudah ada dengan UpdateIntentoperasi.
2. Permintaan API menyertakan `sampleUtterances` bidang, yang memetakan ke array SampleUtteranceobjek.
3. Untuk setiap contoh ucapan yang ingin Anda tambahkan, tambahkan `SampleUtterance` objek ke array. Tambahkan ucapan sampel sebagai nilai bidang. `utterance`
4. Untuk mengedit dan menghapus contoh ucapan, kirim permintaan. `UpdateIntent` Daftar ucapan yang Anda berikan di `sampleUtterances` bidang menggantikan ucapan yang ada.

> **Important**
>
> Bidang apa pun yang Anda biarkan kosong dalam `UpdateIntent` permintaan akan menyebabkan konfigurasi yang ada di intent dihapus. Gunakan DescribeIntentoperasi untuk mengembalikan konfigurasi bot dan menyalin konfigurasi apa pun yang tidak ingin Anda hapus ke dalam `UpdateIntent` permintaan.

Struktur niat

Maksud adalah tujuan yang ingin dicapai pengguna Anda, seperti memesan bunga atau memesan hotel. Bot Anda harus memiliki setidaknya satu niat. Maksud terdiri dari komponen-komponen berikut

- Respons awal - Pesan awal yang dikirim ke pengguna setelah intent dipanggil. Anda dapat mengatur respons, menginisialisasi nilai, dan menentukan langkah selanjutnya yang diambil bot Anda untuk merespons pengguna di awal intent.
- Slot — Parameter yang diperlukan untuk memenuhi maksud. Setiap slot memiliki tipe yang mendefinisikan nilai yang dapat dimasukkan dalam slot. Anda dapat memilih dari jenis slot khusus Anda atau Anda dapat memilih jenis slot bawaan.
- Konfirmasi — Setelah percakapan dengan pengguna selesai dan nilai slot untuk intent terisi, Anda dapat mengatur prompt konfirmasi untuk menanyakan pengguna apakah nilai slot sudah benar.
- Pemenuhan — Respons yang dikirim ke pengguna selama pemenuhan. Anda dapat mengatur pembaruan kemajuan pemenuhan di awal pemenuhan dan terus mengirim pembaruan berkala saat pemenuhan sedang berlangsung. Anda juga dapat mengatur pesan sukses pasca-pemenuhan, pesan kegagalan, dan pesan batas waktu.

- Respons penutupan — Respons penutupan yang dikirim ke pengguna setelah niat mereka terpenuhi. Anda dapat mengatur respons penutupan untuk mengakhiri percakapan, atau Anda dapat mengaturnya agar pengguna tahu bahwa mereka dapat melanjutkan dengan maksud lain.

Topik

- Respon awal
- Slot
- Konfirmasi
- Pemenuhan
- Tanggapan penutupan

Respon awal

Respons awal dikirim ke pengguna setelah Amazon Lex V2 menentukan maksud dan sebelum mulai memperoleh nilai slot. Anda dapat menggunakan tanggapan ini untuk memberi tahu pengguna tentang maksud yang diakui dan untuk mempersiapkan mereka untuk informasi yang Anda kumpulkan untuk memenuhi maksud tersebut.

Misalnya, jika tujuannya adalah menjadwalkan janji temu layanan untuk mobil, respons awal mungkin:

> Saya dapat membantu Anda menjadwalkan janji temu. Anda harus menyediakan merek, model, dan tahun mobil Anda.

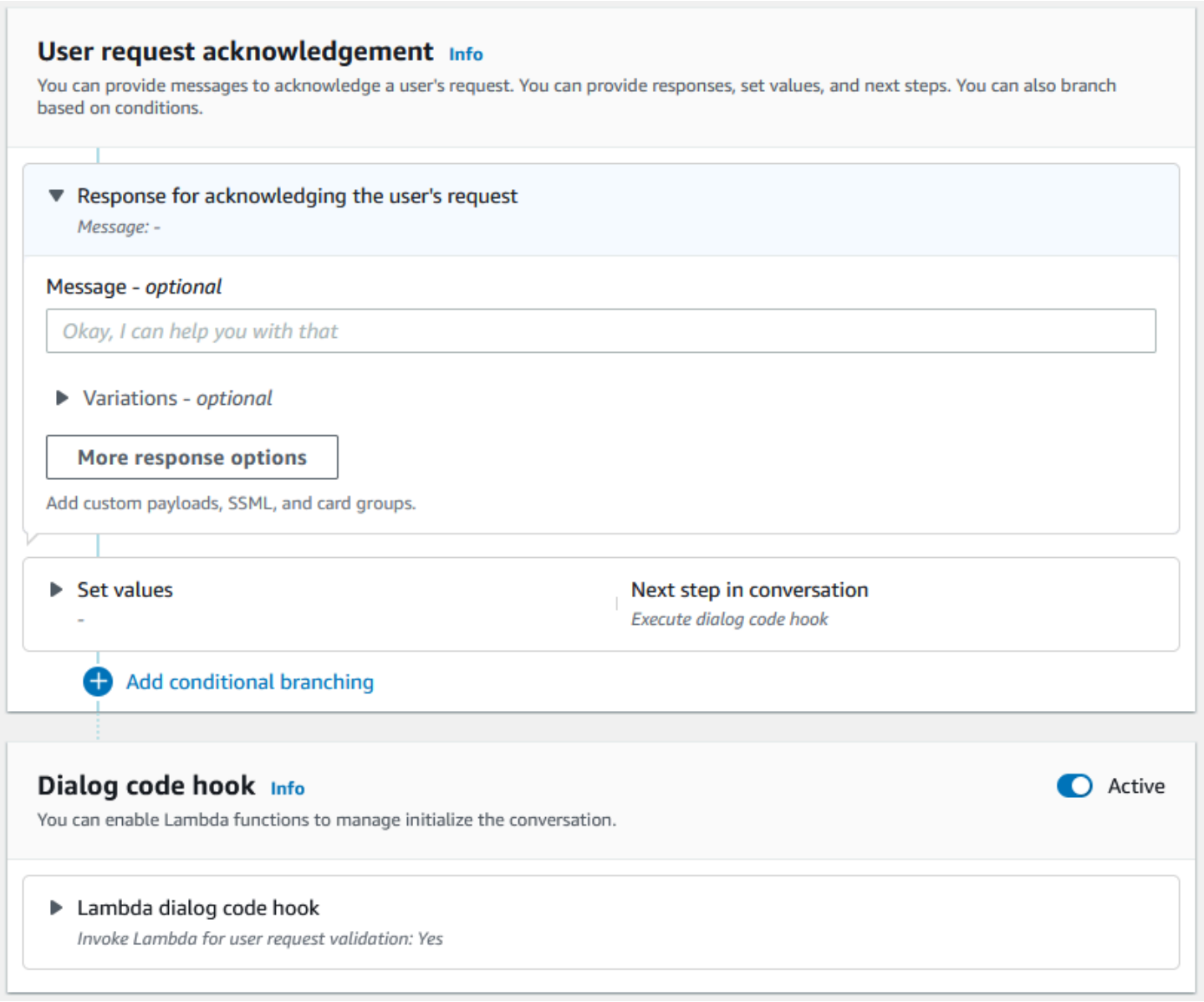
Pesan respons awal tidak diperlukan. Jika Anda tidak menyediakannya, Amazon Lex V2 terus mengikuti langkah berikutnya dari respons awal.

Anda dapat mengonfigurasi opsi berikut dalam respons awal:

- Konfigurasikan langkah berikutnya — Anda dapat memberikan langkah berikutnya dalam percakapan seperti melompat ke tindakan dialog tertentu, memunculkan slot tertentu, atau melompat ke maksud yang berbeda. Untuk informasi selengkapnya, lihat Konfigurasikan langkah selanjutnya dalam percakapan.
- Tetapkan nilai - Anda dapat mengatur nilai untuk slot dan atribut sesi. Untuk informasi selengkapnya, lihat Tetapkan nilai selama percakapan

- Tambahkan percabangan bersyarat — Anda dapat menerapkan kondisi setelah memainkan respons awal. Ketika suatu kondisi dievaluasi menjadi benar, tindakan yang Anda tentukan diambil. Untuk informasi selengkapnya, lihat Tambahkan kondisi ke percakapan cabang.
- Jalankan hook kode dialog - Anda dapat menentukan hook kode Lambda untuk menginisialisasi data dan menjalankan logika bisnis. Untuk informasi selengkapnya, lihat Memanggil hook kode dialog. Jika opsi untuk menjalankan fungsi Lambda diaktifkan untuk intent, hook kode dialog dijalankan secara default. Anda dapat menonaktifkan hook kode dialog dengan mengaktifkan tombol Aktif.

Dengan tidak adanya kondisi atau langkah eksplisit berikutnya, Amazon Lex V2 pindah ke slot berikutnya dalam urutan prioritas.



> **Note**
>
> Pada 17 Agustus 2022, Amazon Lex V2 merilis perubahan pada cara percakapan dikelola dengan pengguna. Perubahan ini memberi Anda kontrol lebih besar atas jalur yang diambil pengguna melalui percakapan. Untuk informasi selengkapnya, lihat Perubahan alur percakapan di Amazon Lex V2. Bot yang dibuat sebelum 17 Agustus 2022 tidak mendukung pesan kait kode dialog, menyetel nilai, mengonfigurasi langkah selanjutnya, dan menambahkan kondisi.

Slot

Slot adalah nilai yang disediakan oleh pengguna untuk memenuhi maksud. Ada dua jenis slot:

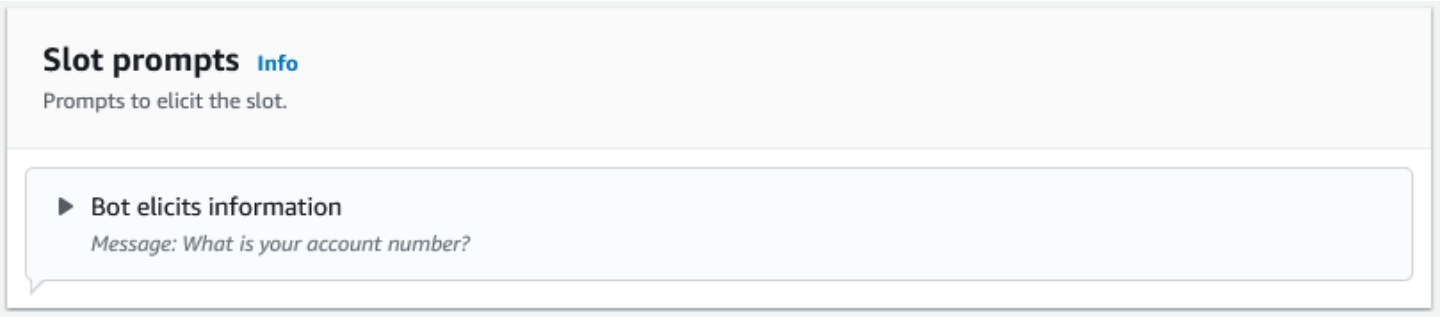
- Jenis slot bawaan — Anda dapat menggunakan tipe slot bawaan untuk menangkap nilai standar seperti nomor, nama, dan kota. Untuk daftar jenis slot bawaan yang didukung, lihatBuilt-in jenis slot.
- Jenis slot khusus - Anda dapat menggunakan jenis slot khusus untuk menangkap nilai khusus khusus untuk maksud. Misalnya, Anda dapat menggunakan jenis slot khusus untuk menangkap jenis akun sebagai “Memeriksa” atau “Tabungan”. Untuk informasi selengkapnya, lihat Jenis slot khusus.

Untuk menentukan slot dalam intent, Anda harus mengonfigurasi yang berikut ini:

- Info slot — Bidang ini berisi nama dan deskripsi opsional untuk slot. Misalnya, Anda dapat memberikan nama slot sebagai “AccountNumber” untuk menangkap nomor akun. Jika slot diperlukan sebagai bagian dari alur percakapan untuk memenuhi maksud, itu harus ditandai sebagai diperlukan.
- Jenis slot — Jenis slot mendefinisikan daftar nilai yang dapat diterima slot. Anda dapat membuat jenis slot khusus atau menggunakan jenis slot yang telah ditentukan sebelumnya.
- Slot prompt — Prompt slot adalah pertanyaan yang diajukan kepada pengguna untuk mengumpulkan informasi. Anda dapat mengonfigurasi jumlah percobaan ulang yang digunakan untuk mengumpulkan informasi dan variasi prompt yang digunakan untuk setiap percobaan ulang. Anda juga dapat mengaktifkan pemanggilan fungsi Lambda setelah setiap percobaan ulang untuk memproses input yang diambil dan mencoba menyelesaikan ke input yang valid.

- Tunggu dan Lanjutkan (opsional) — Dengan mengaktifkan perilaku ini, pengguna dapat mengucapkan frasa seperti “tahan sebentar” untuk membuat bot menunggu mereka menemukan informasi dan memberikannya. Ini diaktifkan hanya untuk streaming percakapan. Untuk informasi selengkapnya, lihat Mengaktifkan bot Amazon Lex V2 untuk menunggu pengguna memberikan informasi lebih lanjut selama jeda.
- Respons pengambilan slot — Anda dapat mengonfigurasi respons sukses dan respons kegagalan berdasarkan hasil menangkap nilai slot dari input pengguna.
- Percabangan bersyarat — Anda dapat menerapkan kondisi setelah memainkan respons awal. Ketika suatu kondisi dievaluasi menjadi benar, tindakan yang Anda tentukan diambil. Untuk informasi selengkapnya, lihat Tambahkan kondisi ke percakapan cabang.
- Kait kode dialog — Anda juga dapat menggunakan hook kode Lambda untuk memvalidasi nilai slot dan menjalankan logika bisnis. Untuk informasi selengkapnya, lihat Memanggil hook kode dialog.
- Jenis input pengguna - Anda dapat mengonfigurasi jenis input sehingga bot dapat menerima modalitas tertentu. Secara default, modalitas audio dan DTMF diterima. Anda dapat secara selektif mengaturnya ke audio saja atau DTMF saja.
- Batas waktu dan panjang input audio - Anda dapat mengonfigurasi batas waktu audio termasuk batas waktu suara dan batas waktu diam. Selain itu, Anda dapat mengatur panjang audio maksimal.
- Batas waktu input DTMF, karakter, dan panjang - Anda dapat mengatur batas waktu DTMF bersama dengan karakter penghapusan dan karakter akhir. Juga, Anda dapat mengatur panjang DTMF maks.
- Panjang teks - Anda dapat mengatur panjang maksimal untuk modalitas teks.

Setelah prompt slot dimainkan, pengguna memberikan nilai slot sebagai input. Jika Amazon Lex V2 tidak memahami nilai slot yang disediakan oleh pengguna, ia mencoba lagi memunculkan slot sampai memahami nilai atau sampai melebihi jumlah percobaan ulang maksimum yang Anda konfigurasikan untuk slot. Dengan menggunakan pengaturan coba lagi lanjutan, Anda dapat mengonfigurasi batas waktu, membatasi jenis input, dan mengaktifkan atau menonaktifkan interupsi untuk prompt awal dan percobaan ulang. Setelah setiap upaya menangkap input, Amazon Lex V2 dapat memanggil fungsi Lambda yang dikonfigurasi untuk bot dengan label pemanggilan yang disediakan untuk percobaan ulang. Anda dapat menggunakan fungsi Lambda, misalnya, untuk menerapkan logika bisnis Anda untuk mencoba menyelesaikannya ke nilai yang valid. Fungsi Lambda ini dapat diaktifkan dalam Opsi lanjutan untuk permintaan slot.



Anda dapat menentukan tanggapan yang harus dikirim bot ke pengguna setelah nilai slot dimasukkan atau jika jumlah percobaan ulang maksimum terlampaui. Misalnya, untuk bot untuk layanan penjadwalan untuk mobil, Anda dapat mengirim pesan ke pengguna ketika nomor identifikasi kendaraan (VIN) dimasukkan:

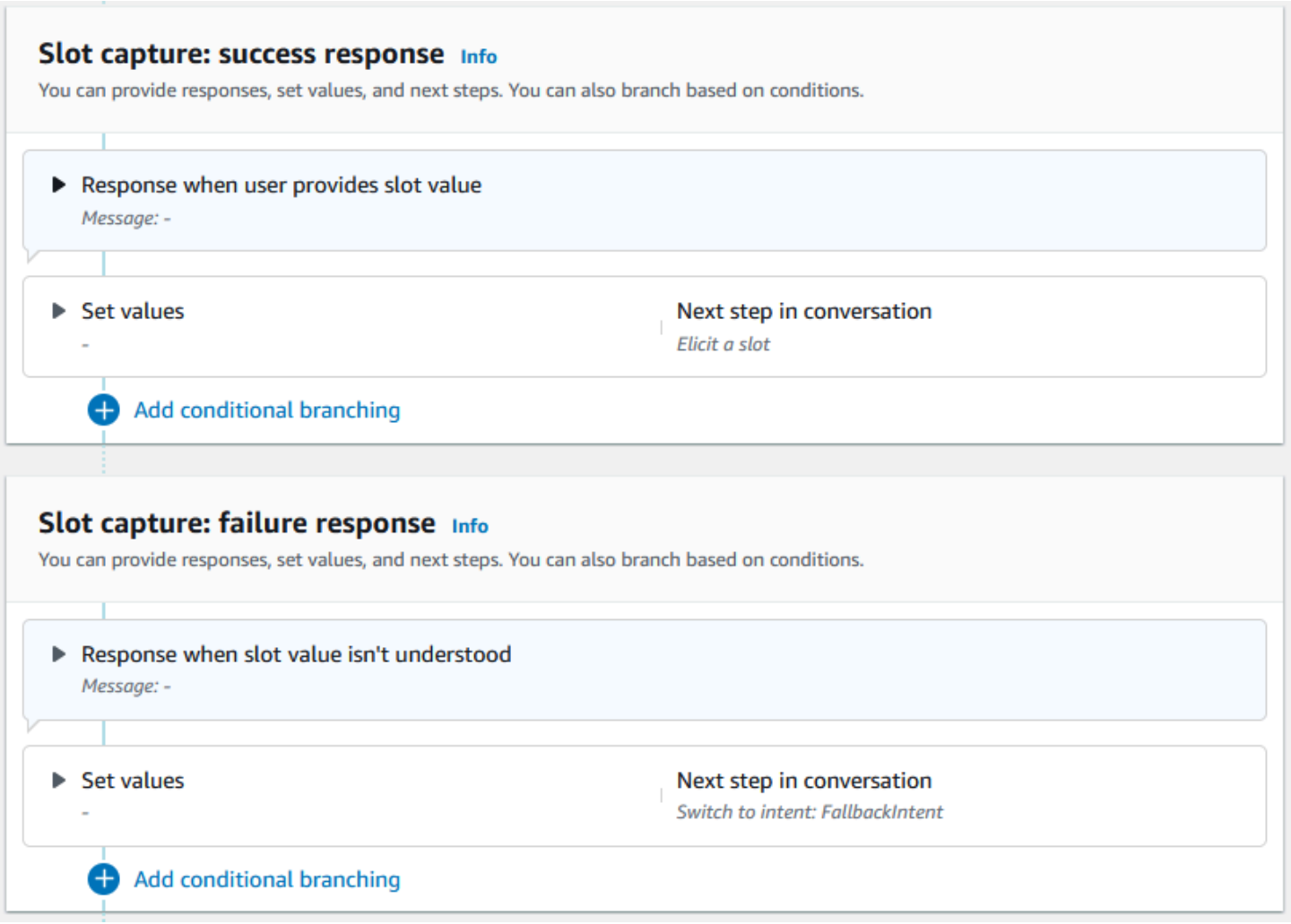
> Terima kasih telah memberikan nomor VIN mobil Anda. Sekarang saya akan melanjutkan untuk menjadwalkan janji temu.

Anda dapat membuat dua tanggapan:

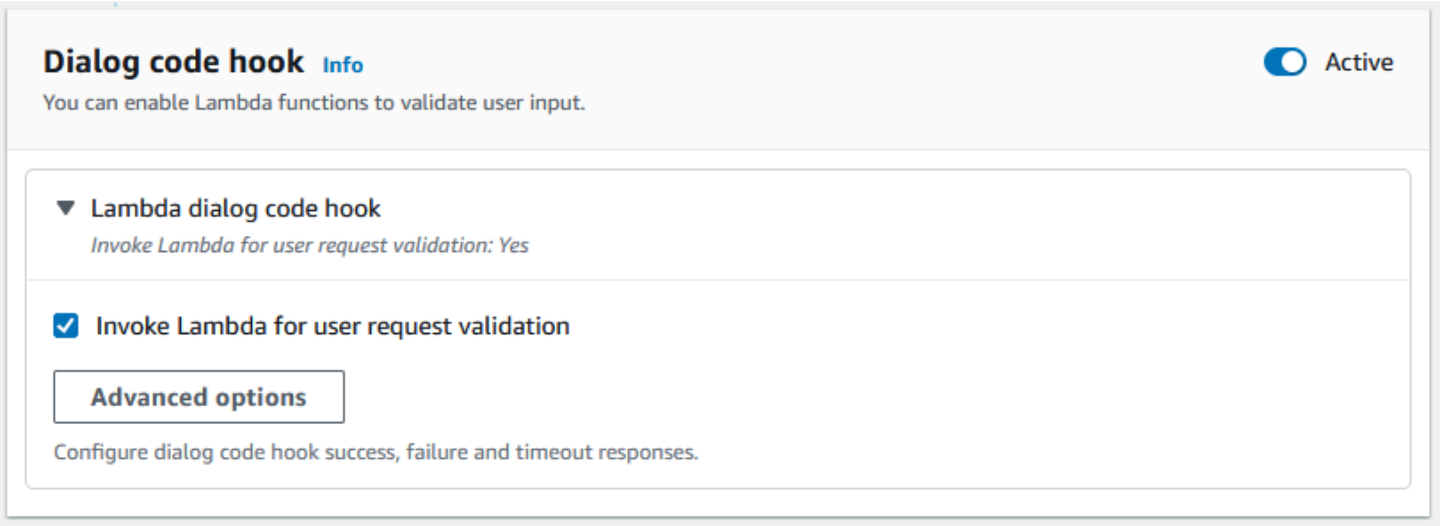
- Respon sukses — dikirim ketika Amazon Lex V2 memahami nilai slot.
- Respons kegagalan — dikirim ketika Amazon Lex V2 tidak dapat memahami nilai slot dari pengguna setelah jumlah maksimum percobaan ulang.

Anda dapat menetapkan nilai, mengonfigurasi langkah selanjutnya, dan menerapkan kondisi yang sesuai dengan setiap respons untuk merancang alur percakapan.

Dengan tidak adanya kondisi atau langkah eksplisit berikutnya, Amazon Lex V2 pindah ke slot berikutnya dalam urutan prioritas.



Anda dapat menggunakan fungsi Lambda untuk memvalidasi nilai slot yang telah dimasukkan pengguna dan menentukan tindakan selanjutnya. Misalnya, Anda dapat menggunakan fungsi validasi untuk memastikan bahwa nilai yang dimasukkan jatuh dalam rentang yang benar, atau yang diformat dengan benar. Untuk mengaktifkan fungsi Lambda, pilih kotak centang fungsi Invoke Lambda dan tombol Aktif di bagian kait kode Dialog. Anda dapat menentukan label pemanggilan untuk hook kode dialog. Label pemanggilan ini dapat digunakan dalam fungsi Lambda untuk menulis logika bisnis yang sesuai dengan elisitasi slot.



Slot yang tidak diperlukan untuk maksud tersebut bukan bagian dari alur percakapan utama. Namun, jika ucapan pengguna berisi nilai yang diidentifikasi bot Anda sesuai dengan slot opsional, itu dapat memunculkan slot dengan nilai itu. Misalnya, jika Anda mengonfigurasi bot intelijen bisnis untuk memiliki `City` slot opsional dan ucapan pengguna**`What is the sales for April in San Diego?`**, bot mengisi slot opsional dengan. **`San Diego`** Anda dapat mengonfigurasi logika bisnis untuk menggunakan nilai slot opsional, jika ada.

Slot yang tidak diperlukan untuk maksud tidak dapat diperoleh menggunakan langkah selanjutnya. Langkah-langkah ini hanya dapat diisi selama elisitasi maksud (seperti pada contoh sebelumnya) atau dapat diperoleh dengan menyetel status dialog dalam fungsi Lambda. Jika slot diperoleh menggunakan fungsi Lambda, Anda harus menggunakan fungsi Lambda untuk memutuskan langkah selanjutnya dalam percakapan setelah elisitasi slot selesai. Untuk mengaktifkan dukungan untuk langkah selanjutnya saat membangun bot, Anda harus menandai slot seperti yang diperlukan untuk maksud tersebut.

> **Note**
>
> Pada 17 Agustus 2022, Amazon Lex V2 merilis perubahan pada cara percakapan dikelola dengan pengguna. Perubahan ini memberi Anda kontrol lebih besar atas jalur yang diambil pengguna melalui percakapan. Untuk informasi selengkapnya, lihat Perubahan alur percakapan di Amazon Lex V2. Bot yang dibuat sebelum 17 Agustus 2022 tidak mendukung pesan kait kode dialog, menyetel nilai, mengonfigurasi langkah selanjutnya, dan menambahkan kondisi.

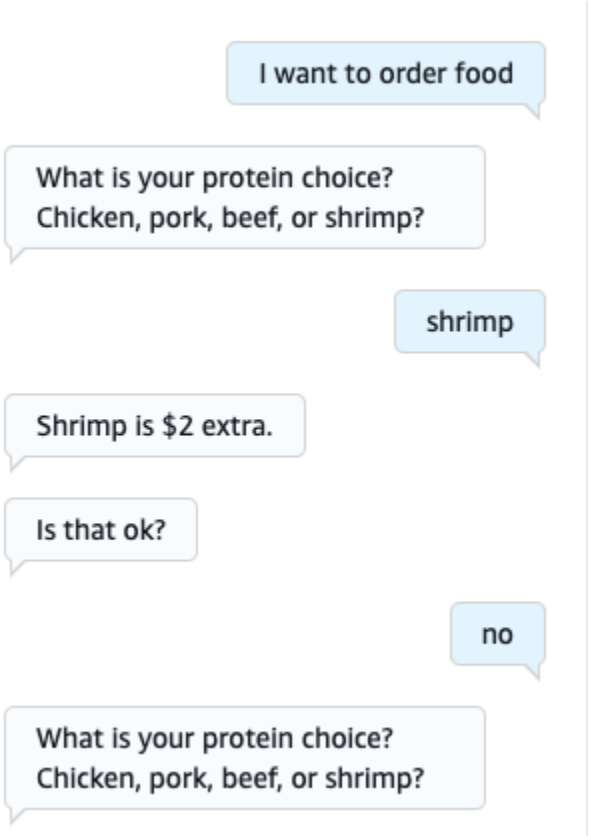
Topik berikut menjelaskan cara mengkonfigurasi bot untuk mendapatkan kembali nilai slot yang telah diisi dan cara membuat slot yang terdiri dari beberapa nilai:

Topik

- Memunculkan kembali slot
- Menggunakan beberapa nilai dalam slot

Memunculkan kembali slot

Anda dapat mengonfigurasi bot Anda untuk mendapatkan kembali slot yang telah diisi dengan mengatur nilai slot itu **`null`** dan mengatur langkah berikutnya dalam percakapan untuk kembali ke mendapatkan slot itu. Misalnya, Anda mungkin ingin mendapatkan kembali slot setelah pelanggan Anda menolak konfirmasi elicitation slot berdasarkan informasi tambahan, seperti dalam percakapan berikut:



Anda dapat mengonfigurasi loop dari respons konfirmasi kembali untuk mendapatkan kembali slot dengan editor maksud atau editor. Menggunakan pembangun percakapan Visual

> **Note**
>
> Anda dapat memutar kembali untuk mendapatkan kembali slot kapan saja dalam percakapan asalkan Anda menetapkan nilai slot itu sebelumnya. **`null`**

Mereproduksi contoh di atas dengan editor maksud

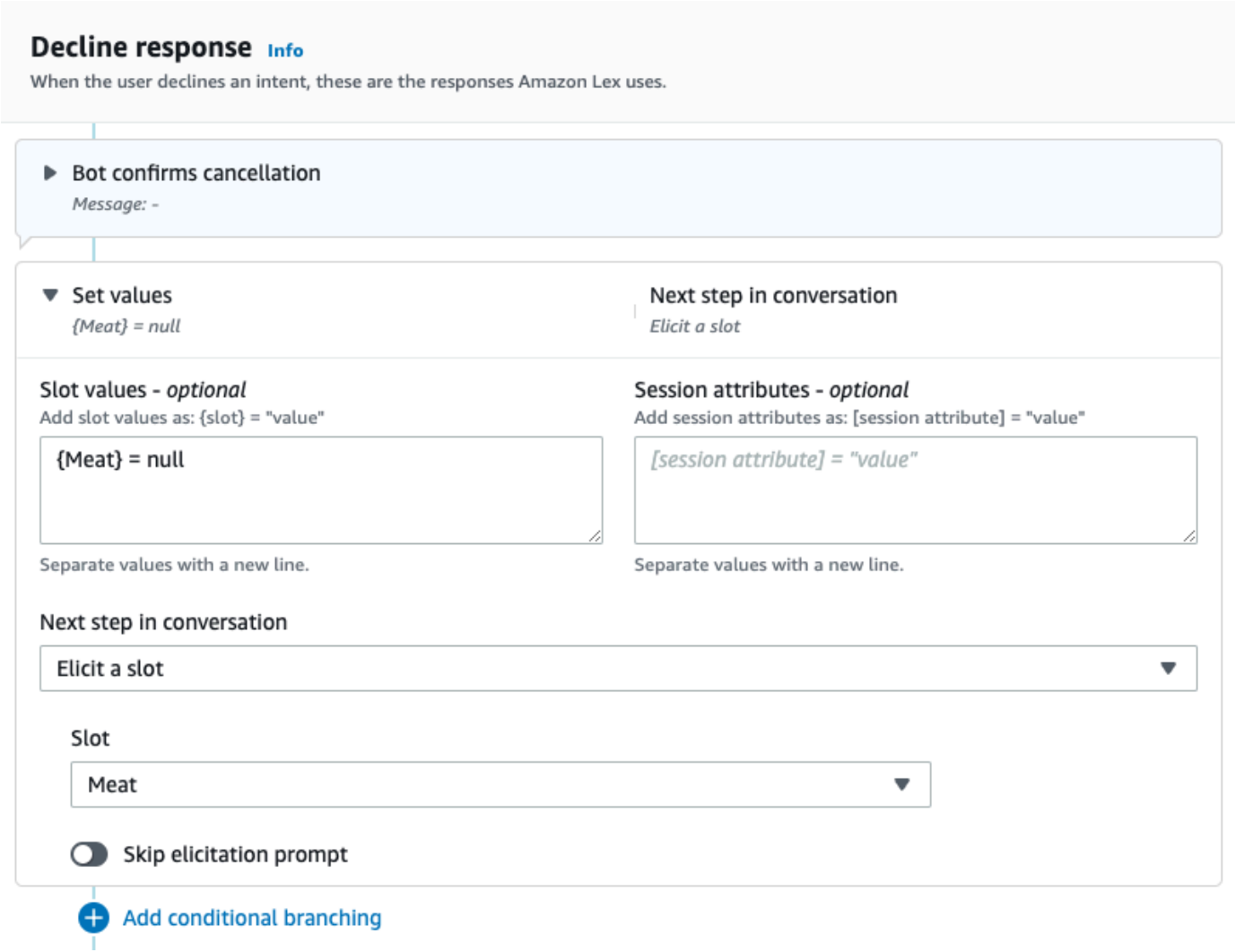
1. Di bagian Konfirmasi editor maksud, pilih panah kanan di sebelah Prompts untuk mengonfirmasi maksud untuk memperluas bagian.
2. Pilih Opsi lanjutan di bagian bawah.
3. Di bagian Tolak respons, pilih panah kanan di sebelah Tetapkan nilai untuk memperluas bagian. Isi bagian ini dengan langkah-langkah berikut, seperti pada gambar di bawah ini:

   a. Tetapkan nilai slot yang ingin Anda dapatkan kembali. **`null`** Dalam contoh ini, kami ingin mendapatkan kembali `Meat` slot, jadi kami masukan **`{Meat} = null`** di bagian Nilai Slot.

   b. Di menu tarik-turun di bawah Langkah berikutnya dalam percakapan, pilih Dapatkan slot.

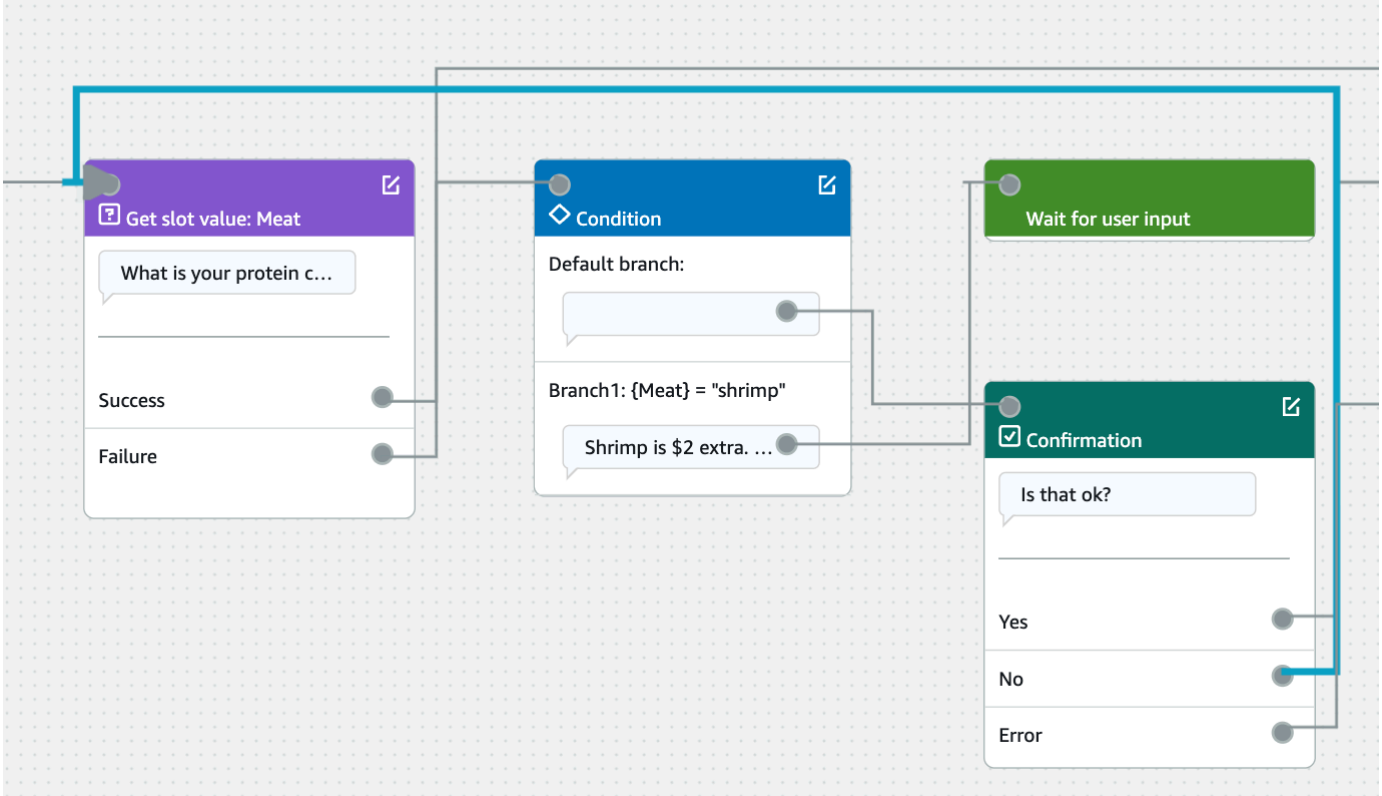
   c. Bagian Slot akan muncul. Di menu tarik-turun di bawahnya, pilih slot yang ingin Anda dapatkan kembali.

   d. Pilih opsi Perbarui untuk mengonfirmasi perubahan Anda.

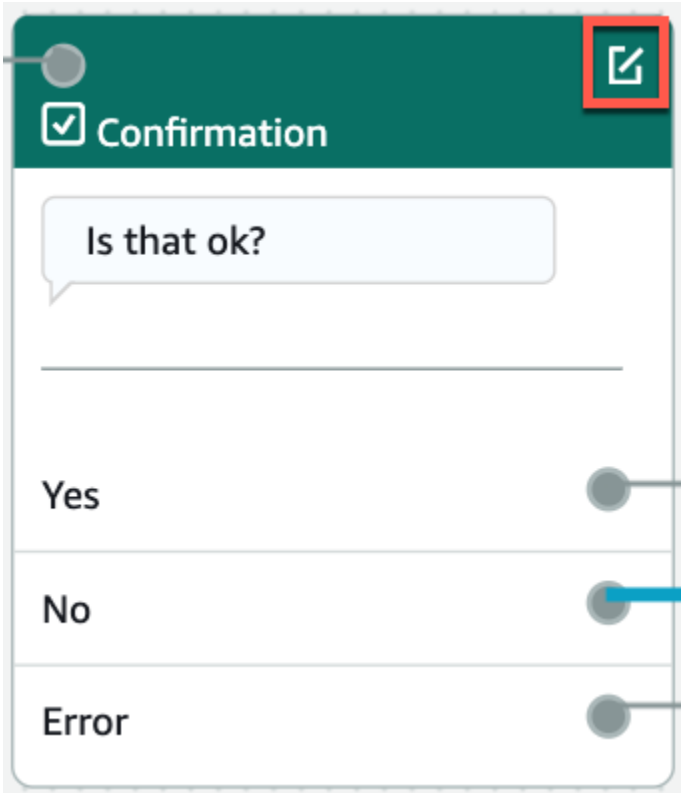


Mereproduksi contoh di atas dengan pembangun percakapan Visual

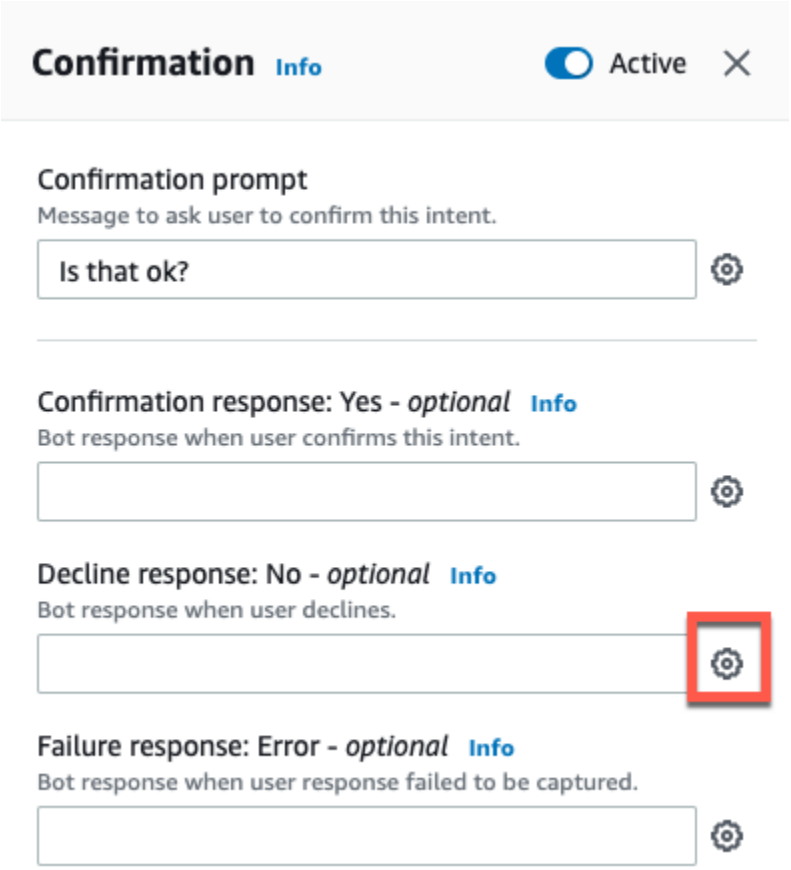
1. Buat koneksi dari No port dari blok Konfirmasi ke port masuk dari Get slot value: Meat block.



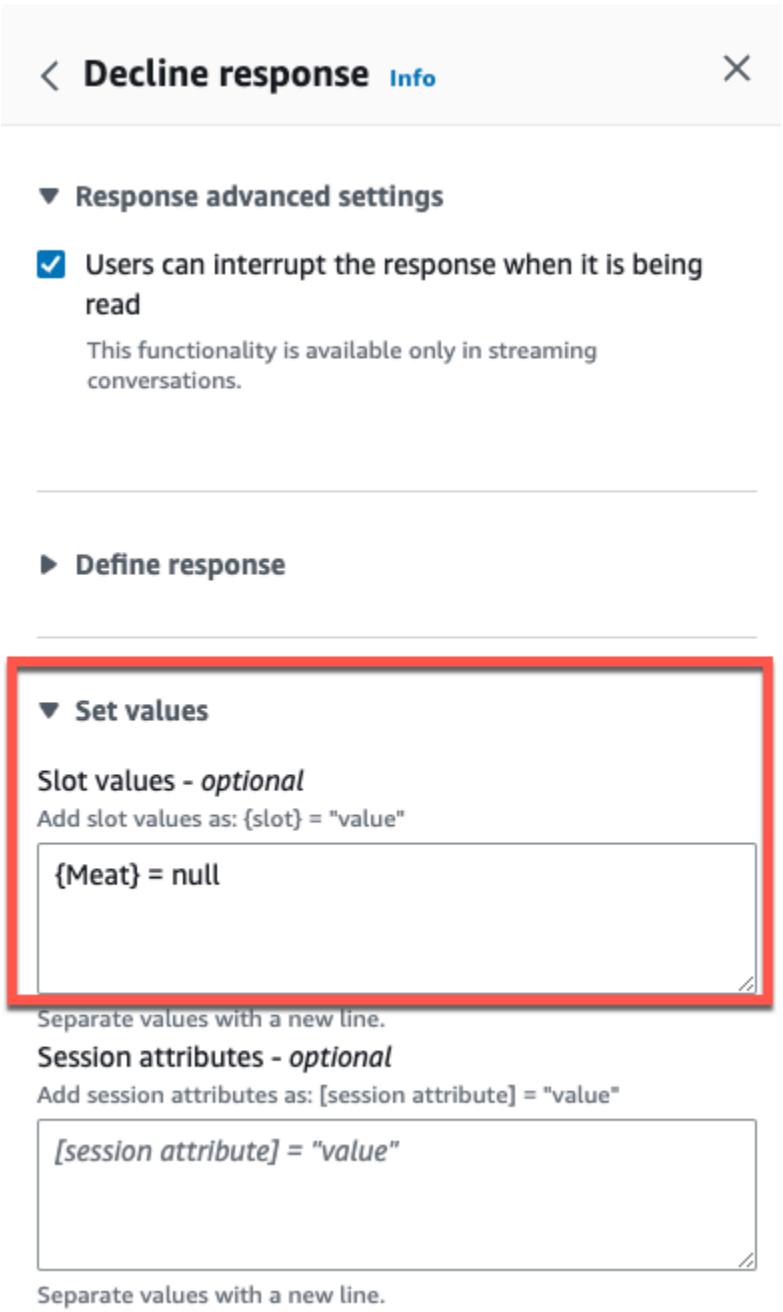
2. Pilih ikon Edit di sudut kanan atas blok Konfirmasi.



3. Pilih ikon roda gigi di sebelah respons bot di bagian respons Decilne.



4. Di bagian Set values, tambahkan “{Meat} = null” di kotak Nilai Slot.



5. Pilih Simpan Maksud.

Menggunakan beberapa nilai dalam slot

> **Note**
>
> Beberapa slot nilai hanya didukung dalam bahasa Inggris (AS).

Untuk beberapa maksud, Anda mungkin ingin menangkap beberapa nilai untuk satu slot. Misalnya, bot pemesanan pizza mungkin memiliki maksud dengan ucapan berikut:

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

Maksudnya mengharapkan bahwa `{toppings}` slot berisi daftar topping yang diinginkan pelanggan di pizza mereka, misalnya “pepperoni dan nanas”.

Untuk mengonfigurasi slot untuk menangkap beberapa nilai, Anda mengatur `allowMultipleValues` bidang pada slot ke true. Anda dapat mengatur bidang menggunakan konsol atau dengan UpdateSlotoperasi CreateSlotatau.

Anda hanya dapat menandai slot dengan jenis slot khusus sebagai slot multi-nilai.

Untuk slot multi-nilai, Amazon Lex V2 mengembalikan daftar nilai slot sebagai respons terhadap RecognizeUtteranceoperasi RecognizeTextatau. Berikut ini adalah informasi slot yang dikembalikan untuk ucapan “Saya ingin pizza dengan pepperoni dan nanas” dari bot. OrderPizza

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

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

Slot multi-nilai selalu mengembalikan daftar nilai. Ketika ucapan hanya berisi satu nilai, daftar nilai yang dikembalikan hanya berisi satu respons.

Amazon Lex V2 mengenali beberapa nilai yang dipisahkan oleh spasi, koma (,), dan konjungsi “dan”. Slot multi-nilai bekerja dengan input teks dan suara.

Anda dapat menggunakan slot multi-nilai dalam petunjuk. Misalnya, Anda dapat mengatur prompt konfirmasi untuk maksud

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

Ketika Amazon Lex V2 mengirimkan prompt ke pengguna, ia mengirimkan “Apakah Anda ingin saya memesan pizza pepperoni dan nanas Anda?”

Slot multi-nilai mendukung nilai default tunggal. Jika beberapa nilai default disediakan, Amazon Lex V2 mengisi slot dengan hanya nilai pertama yang tersedia. Untuk informasi selengkapnya, lihat Menggunakan nilai slot default dalam maksud untuk bot Lex V2 Anda.

Anda dapat menggunakan slot obfuscation untuk menutupi nilai slot multi-nilai di log percakapan. Saat Anda mengaburkan nilai slot, nilai masing-masing nilai slot diganti dengan nama slot. Lihat informasi yang lebih lengkap di Mengaburkan nilai slot dalam log percakapan dari Lex V2.

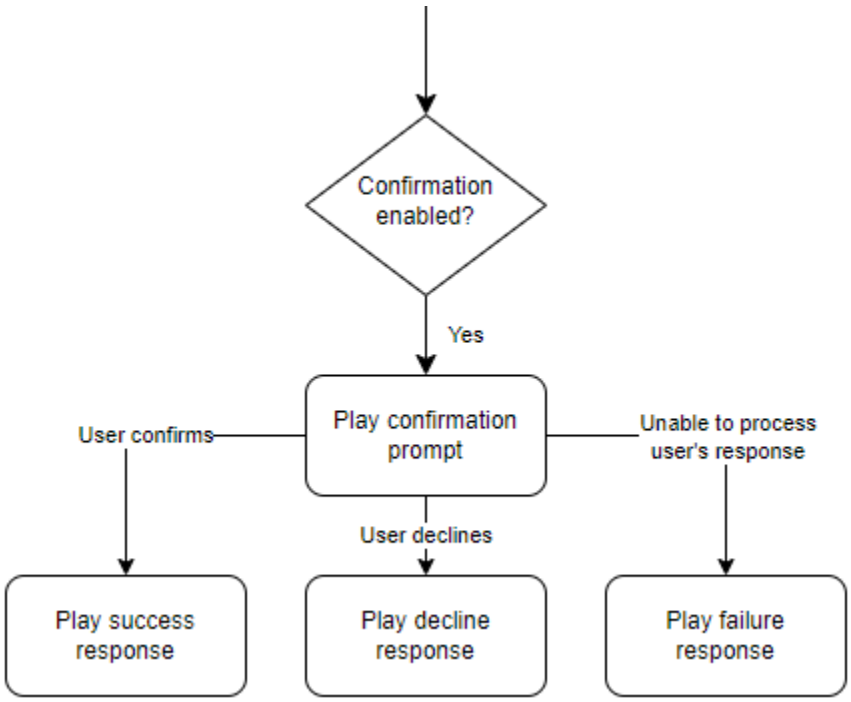
Konfirmasi

Setelah percakapan dengan pengguna selesai dan nilai slot untuk intent terisi, Anda dapat mengonfigurasi prompt konfirmasi untuk menanyakan kepada pengguna apakah nilai slot sudah benar. Misalnya, bot yang menjadwalkan janji temu layanan untuk mobil mungkin meminta pengguna dengan hal-hal berikut:

Saya punya layanan untuk Honda Civic 2017 Anda yang dijadwalkan pada 25 Maret pukul 15.00. Apakah itu semua benar?

Anda dapat menentukan 3 jenis tanggapan terhadap prompt konfirmasi:

- Respons konfirmasi - Respons ini dikirim ke pengguna saat pengguna mengonfirmasi maksud. Misalnya, setelah pengguna menjawab “ya” ke prompt “apakah Anda ingin melakukan pemesanan?”
- Respons tolak - Respons ini dikirim ke pengguna saat pengguna menolak maksud. Misalnya, setelah pengguna membalas “tidak” ke prompt “apakah Anda ingin melakukan pemesanan?”
- Respons kegagalan - Respons ini dikirim ke pengguna ketika prompt konfirmasi tidak dapat diproses. Misalnya, jika respons pengguna tidak dapat dipahami atau tidak dapat diselesaikan menjadi ya atau tidak.

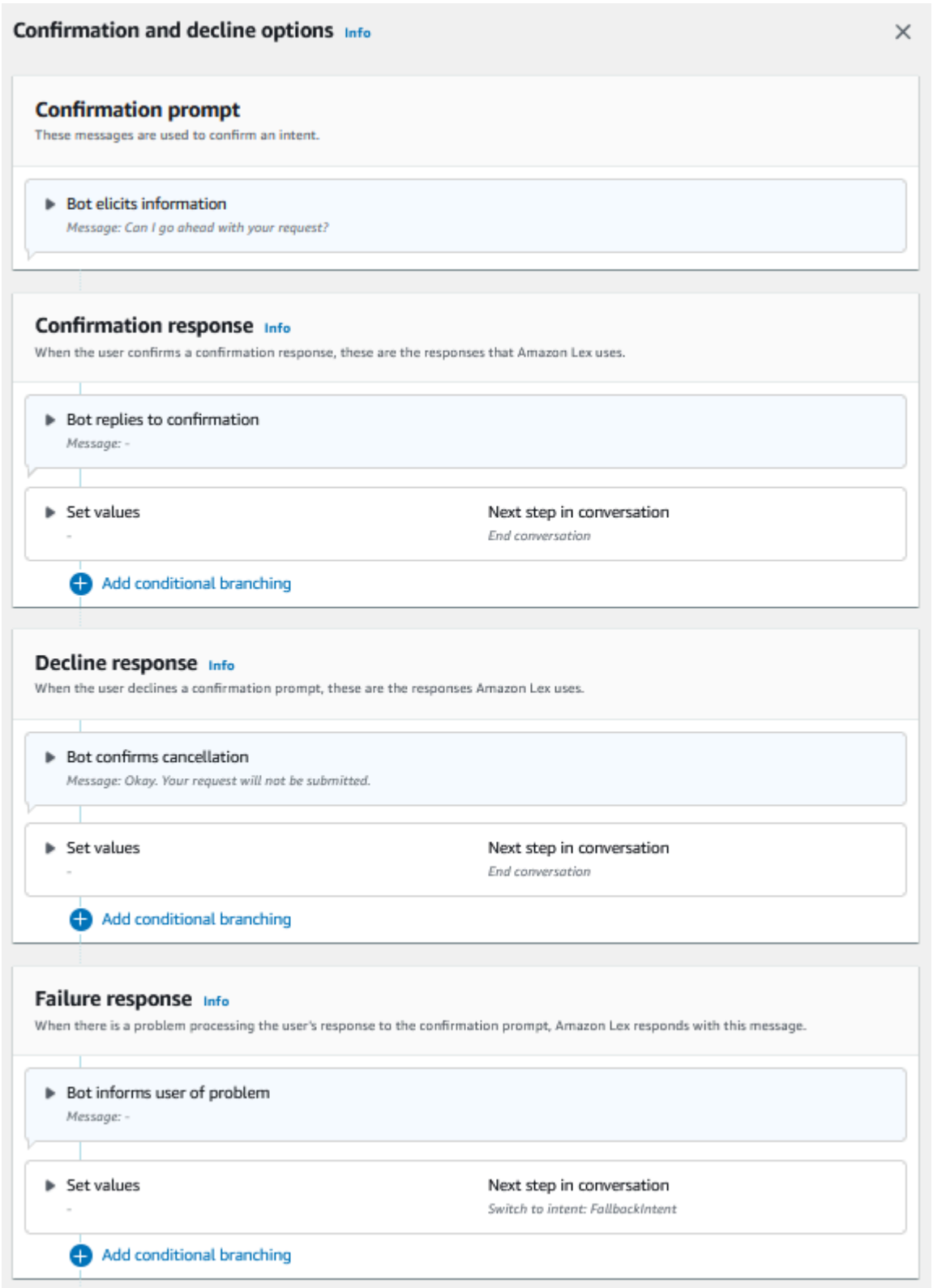


Jika Anda tidak menentukan prompt konfirmasi, Amazon Lex V2 pindah ke langkah pemenuhan atau respons penutupan.

Anda dapat menetapkan nilai, mengonfigurasi langkah berikutnya, dan menerapkan kondisi yang sesuai dengan setiap respons untuk merancang alur percakapan. Dengan tidak adanya kondisi atau langkah eksplisit berikutnya, Amazon Lex V2 pindah ke langkah pemenuhan.

Anda juga dapat mengaktifkan hook kode dialog untuk memvalidasi informasi yang ditangkap dalam intent sebelum mengirimnya untuk pemenuhan. Untuk menggunakan hook kode, aktifkan hook kode dialog di opsi lanjutan prompt konfirmasi. Selain itu, konfigurasikan langkah selanjutnya dari status sebelumnya untuk mengeksekusi hook kode dialog. Untuk informasi selengkapnya, lihat Memanggil hook kode dialog.

> **Note**
>
> Jika Anda menggunakan hook kode untuk memicu langkah konfirmasi saat runtime, Anda harus menandai langkah konfirmasi sebagai Aktif pada waktu pembuatan.


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**
>
> Pada 17 Agustus 2022, Amazon Lex V2 merilis perubahan pada cara percakapan dikelola dengan pengguna. Perubahan ini memberi Anda kontrol lebih besar atas jalur yang diambil pengguna melalui percakapan. Untuk informasi selengkapnya, lihat Perubahan alur percakapan di Amazon Lex V2. Bot yang dibuat sebelum 17 Agustus 2022 tidak mendukung pesan kait kode dialog, menyetel nilai, mengonfigurasi langkah selanjutnya, dan menambahkan kondisi.

Menggunakan fungsi Lambda untuk memvalidasi intent.

Anda dapat menentukan hook kode Lambda untuk memvalidasi intent sebelum mengirimnya untuk pemenuhan. Untuk menggunakan hook kode, aktifkan hook kode dialog di opsi lanjutan prompt konfirmasi.

Saat Anda menggunakan kait kode, Anda dapat menentukan tindakan yang dilakukan Amazon Lex V2 setelah hook kode berjalan. Anda dapat membuat tiga jenis tanggapan:

- Respon sukses - Dikirim ke pengguna ketika hook kode selesai dengan sukses.
- Respons kegagalan - Dikirim ke pengguna ketika hook kode tidak berjalan dengan sukses atau ketika hook kode kembali `Failure` dalam respons.
- Respons batas waktu - Dikirim ke pengguna saat kait kode tidak selesai dalam periode batas waktu yang dikonfigurasi.

Pemenuhan

Setelah semua nilai slot disediakan oleh pengguna untuk maksud tersebut, Amazon Lex V2 memenuhi permintaan pengguna. Anda dapat mengonfigurasi opsi berikut untuk pemenuhan.

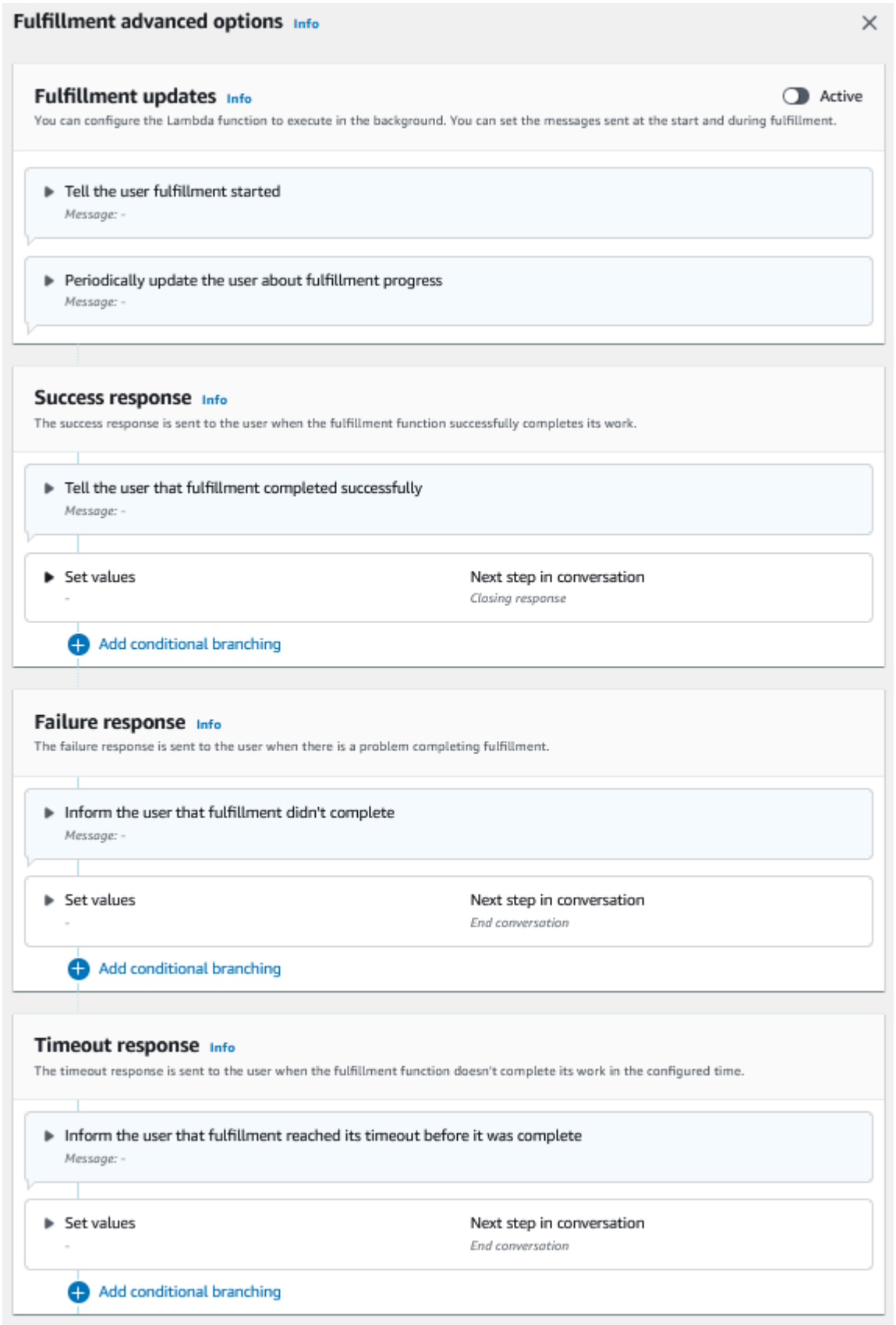
- Hook kode pemenuhan - Anda dapat menggunakan opsi ini untuk mengontrol pemenuhan pemanggilan Lambda. Jika opsi dinonaktifkan, pemenuhan berhasil tanpa menjalankan fungsi Lambda.
- Pembaruan pemenuhan - Anda dapat mengaktifkan pembaruan pemenuhan untuk fungsi Lambda yang membutuhkan waktu lebih dari beberapa detik untuk diselesaikan, sehingga pengguna tahu bahwa prosesnya sedang berlangsung. Untuk informasi selengkapnya, lihat Mengkonfigurasi pembaruan kemajuan pemenuhan untuk bot Lex V2 Anda. Fungsionalitas ini hanya tersedia untuk streaming percakapan.

- Respons pemenuhan — Anda dapat mengonfigurasi respons sukses, respons kegagalan, dan respons batas waktu. Respons yang sesuai dikembalikan ke pengguna berdasarkan status pemenuhan pemanggilan Lambda.

Ada tiga kemungkinan tanggapan pemenuhan:

- Respon sukses — Pesan yang dikirim ketika pemenuhan Lambda selesai dengan sukses.
- Respons kegagalan — Pesan yang dikirim jika pemenuhan gagal atau Lambda tidak dapat diselesaikan karena alasan tertentu.
- Respons batas waktu - Pesan yang dikirim jika fungsi Lambda pemenuhan tidak selesai dalam batas waktu yang dikonfigurasi.

Anda dapat menetapkan nilai, mengonfigurasi langkah berikutnya, dan menerapkan kondisi yang sesuai dengan setiap respons untuk merancang alur percakapan. Dengan tidak adanya kondisi atau langkah eksplisit berikutnya, Amazon Lex V2 beralih ke respons penutupan.


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

> **Note**
>
> Pada 17 Agustus 2022, Amazon Lex V2 merilis perubahan pada cara percakapan dikelola dengan pengguna. Perubahan ini memberi Anda kontrol lebih besar atas jalur yang diambil pengguna melalui percakapan. Untuk informasi selengkapnya, lihat Perubahan alur percakapan di Amazon Lex V2. Bot yang dibuat sebelum 17 Agustus 2022 tidak mendukung pesan kait kode dialog, menyetel nilai, mengonfigurasi langkah selanjutnya, dan menambahkan kondisi.

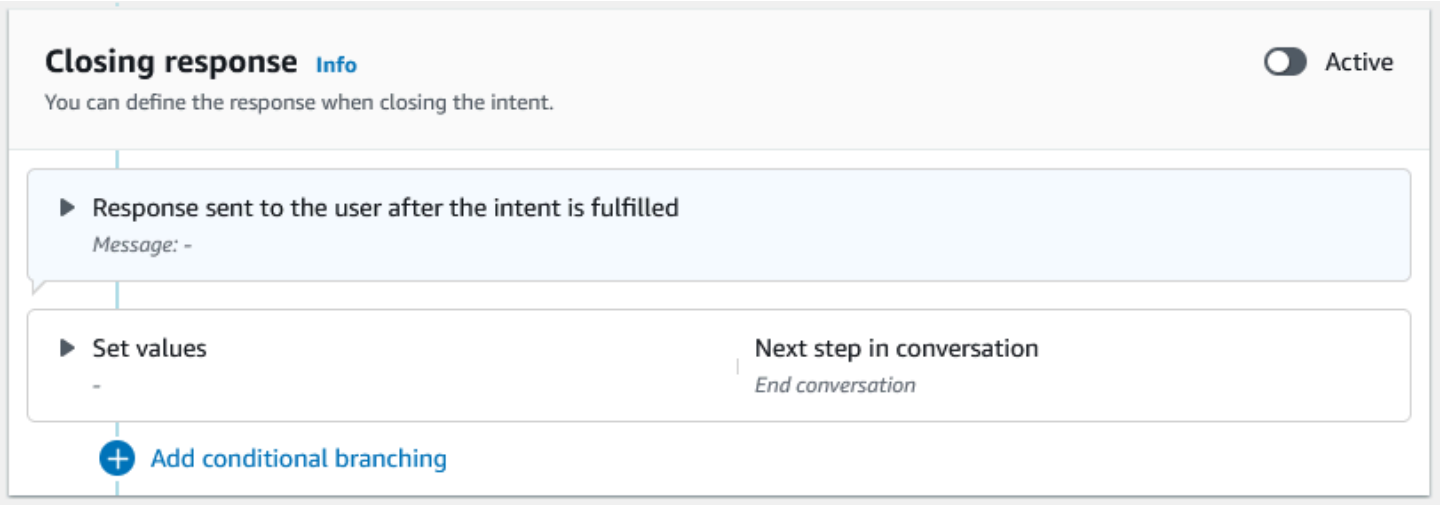
Tanggapan penutupan

Respons penutupan dikirim ke pengguna Anda setelah maksud mereka terpenuhi. Anda dapat menggunakan respons penutup untuk mengakhiri percakapan, atau Anda dapat menggunakannya untuk memberi tahu pengguna bahwa mereka dapat melanjutkan dengan maksud lain. Misalnya, dalam bot pemesanan perjalanan, Anda dapat mengatur respons penutupan untuk maksud kamar hotel buku untuk ini:

```
Baiklah, aku sudah memesan kamar hotelmu. Apakah ada hal lain yang bisa saya bantu?
```

Anda dapat menetapkan nilai, mengonfigurasi langkah selanjutnya, dan menerapkan kondisi setelah respons penutupan untuk merancang jalur percakapan. Dengan tidak adanya kondisi atau langkah eksplisit berikutnya, Amazon Lex V2 mengakhiri percakapan.

Jika Anda tidak memberikan respons penutup, atau jika tidak ada kondisi yang benar, Amazon Lex V2 mengakhiri percakapan dengan bot Anda.

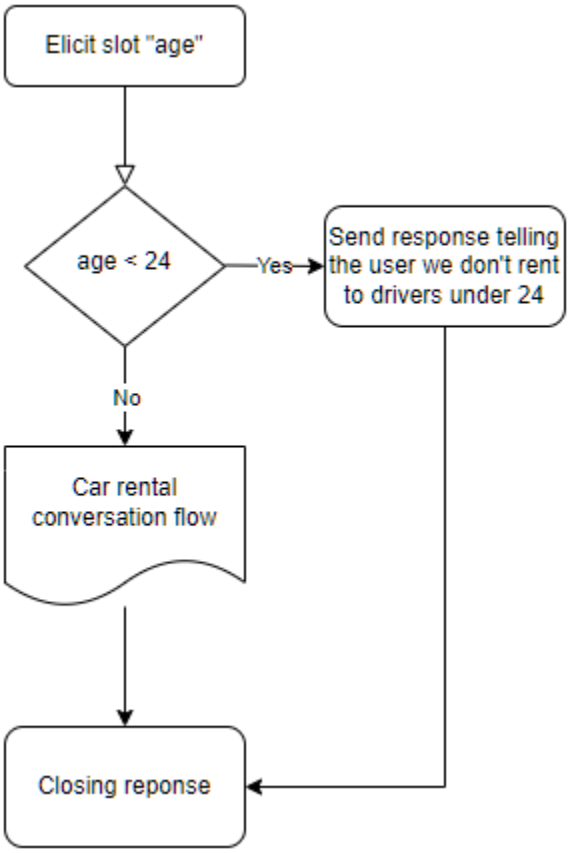


> **Note**
>
> Pada 17 Agustus 2022, Amazon Lex V2 merilis perubahan pada cara percakapan dikelola dengan pengguna. Perubahan ini memberi Anda kontrol lebih besar atas jalur yang diambil pengguna melalui percakapan. Untuk informasi selengkapnya, lihat Perubahan alur percakapan di Amazon Lex V2. Bot yang dibuat sebelum 17 Agustus 2022 tidak mendukung pesan kait kode dialog, menyetel nilai, mengonfigurasi langkah selanjutnya, dan menambahkan kondisi.

Membuat jalur percakapan

Biasanya, Amazon Lex V2 mengelola aliran percakapan dengan pengguna Anda. Untuk bot sederhana, aliran default bisa cukup untuk menciptakan pengalaman yang baik bagi pengguna Anda. Namun, untuk bot yang lebih kompleks, Anda mungkin ingin mengendalikan percakapan dan mengarahkan aliran ke jalur yang lebih kompleks.

Misalnya, dalam bot yang memesan persewaan mobil, Anda mungkin tidak menyewakan kepada pengemudi yang lebih muda. Dalam hal ini, Anda dapat membuat kondisi yang memeriksa untuk melihat apakah pengemudi di bawah usia tertentu, dan jika demikian, lompat ke respons penutupan.



Untuk merancang interaksi seperti itu, Anda dapat mengonfigurasi langkah berikutnya di setiap titik dalam percakapan, mengevaluasi kondisi, menetapkan nilai, dan memanggil kait kode.

Percabangan bersyarat membantu Anda membuat jalur untuk pengguna melalui interaksi yang kompleks. Anda dapat menggunakan cabang bersyarat kapan saja Anda melewati kendali percakapan ke bot Anda. Misalnya, Anda dapat membuat kondisi sebelum bot memunculkan nilai slot pertama, Anda dapat membuat kondisi antara memunculkan setiap nilai slot, atau Anda dapat membuat kondisi sebelum bot menutup percakapan. Untuk daftar tempat yang dapat Anda tambahkan kondisi, lihatMenambahkan maksud.

Saat Anda membuat bot, Amazon Lex V2 membuat jalur default melalui percakapan berdasarkan urutan prioritas slot. Untuk menyesuaikan jalur percakapan, Anda dapat mengubah langkah berikutnya kapan saja dalam percakapan. Untuk informasi selengkapnya, lihat Konfigurasikan langkah selanjutnya dalam percakapan.

Untuk membuat jalur alternatif berdasarkan kondisi, Anda dapat menggunakan cabang bersyarat kapan saja dalam percakapan. Misalnya, Anda dapat membuat kondisi sebelum bot memunculkan nilai slot pertama. Anda dapat membuat kondisi antara memunculkan setiap nilai slot, atau Anda

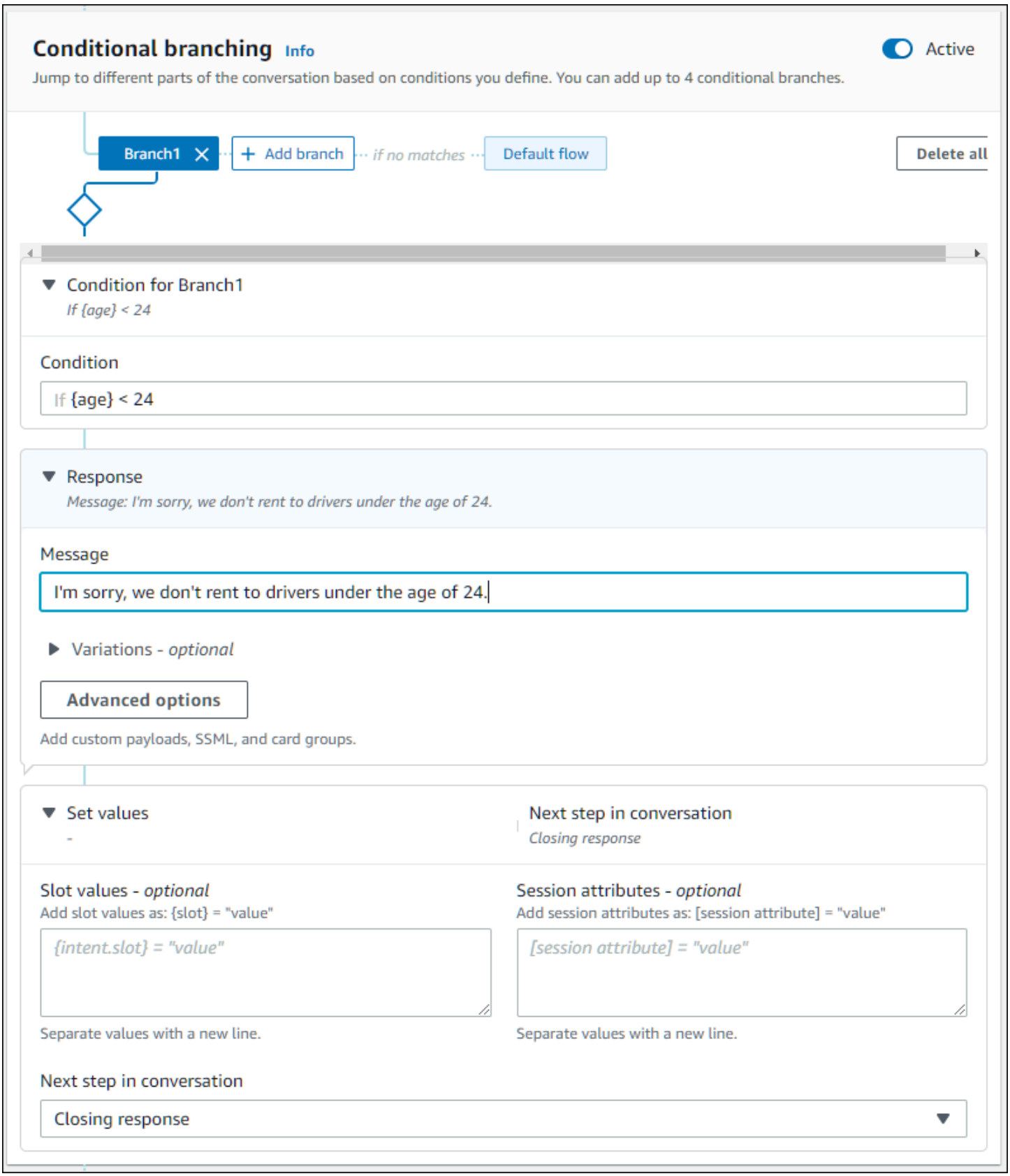
dapat membuat kondisi sebelum bot menutup percakapan. Untuk daftar tempat yang memungkinkan Anda menambahkan kondisi, lihatTambahkan kondisi ke percakapan cabang.

Anda dapat mengatur kondisi berdasarkan nilai slot, atribut sesi, mode input dan transkrip input, atau respons dari Amazon Kendra.

Anda dapat mengatur nilai atribut slot dan sesi di setiap titik dalam percakapan. Untuk informasi selengkapnya, lihat Tetapkan nilai selama percakapan.

Anda juga dapat mengatur tindakan berikutnya ke hook kode dialog untuk menjalankan fungsi Lambda. Untuk informasi selengkapnya, lihat Memanggil hook kode dialog.

Gambar berikut menunjukkan pembuatan jalur untuk slot di konsol. Dalam contoh ini, Amazon Lex V2 akan memperoleh slot “usia”. Jika nilai slot kurang dari 24, Amazon Lex V2 melompat ke respons penutupan, jika tidak Amazon Lex V2 akan mengikuti jalur default.


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

> **Note**
>
> Pada 17 Agustus 2022, Amazon Lex V2 merilis perubahan pada cara percakapan dikelola dengan pengguna. Perubahan ini memberi Anda kontrol lebih besar atas jalur yang diambil pengguna melalui percakapan. Untuk informasi selengkapnya, lihat Perubahan alur percakapan di Amazon Lex V2. Bot yang dibuat sebelum 17 Agustus 2022 tidak mendukung pesan kait kode dialog, menyetel nilai, mengonfigurasi langkah selanjutnya, dan menambahkan kondisi.

Konfigurasikan langkah selanjutnya dalam percakapan

Anda dapat mengonfigurasi langkah berikutnya di setiap tahap percakapan untuk merancang percakapan. Biasanya, Amazon Lex V2 secara otomatis mengonfigurasi langkah default berikutnya untuk setiap tahap percakapan sesuai urutan berikut.

Respon Awal → Elisitasi Slot → Konfirmasi (jika aktif) → Pemenuhan (jika aktif) → Respon Penutupan (jika aktif) → Akhiri percakapan

Anda dapat memodifikasi langkah default berikutnya dan mendesain percakapan berdasarkan pengalaman pengguna yang diharapkan. Langkah-langkah berikut selanjutnya dapat dikonfigurasi pada setiap tahap percakapan:

Lompat ke

- Respon awal — Percakapan dimulai kembali dari awal niat. Anda dapat memilih untuk melewati respons awal saat mengonfigurasi langkah berikutnya
- Dapatkan slot — Anda dapat memperoleh slot apa pun dalam maksud.
- Evaluasi kondisi — Anda dapat mengevaluasi kondisi dan percakapan cabang pada setiap langkah percakapan.
- Memanggil kode dialog hook - Anda dapat menjalankan logika bisnis pada langkah apa pun.
- Konfirmasi maksud — Pengguna akan diminta untuk mengonfirmasi maksud.
- Memenuhi niat — Pemenuhan niat akan dimulai sebagai langkah berikutnya.
- Respons penutupan - Respons penutupan akan dikembalikan ke pengguna.

Beralih ke

- Maksud — Anda dapat beralih ke maksud yang berbeda dan melanjutkan percakapan untuk maksud ini. Secara opsional, Anda dapat melewatkan respons awal intent saat melakukan transisi.
- Maksud: slot khusus — Anda dapat langsung memperoleh slot tertentu dalam maksud yang berbeda jika Anda telah menangkap beberapa nilai slot dalam maksud saat ini.

Tunggu masukan pengguna — Bot menunggu pengguna memberikan input untuk mengenali maksud baru apa pun. Anda dapat mengonfigurasi petunjuk seperti “Apakah ada hal lain yang dapat saya bantu?” sebelum menetapkan langkah selanjutnya. Bot akan berada dalam status `ElicitIntent` dialog.

Akhiri percakapan — Percakapan dengan bot ditutup.

> **Note**
>
> Pada 17 Agustus 2022, Amazon Lex V2 merilis perubahan pada cara percakapan dikelola dengan pengguna. Perubahan ini memberi Anda kontrol lebih besar atas jalur yang diambil pengguna melalui percakapan. Untuk informasi selengkapnya, lihat Perubahan alur percakapan di Amazon Lex V2. Bot yang dibuat sebelum 17 Agustus 2022 tidak mendukung pesan kait kode dialog, menyetel nilai, mengonfigurasi langkah selanjutnya, dan menambahkan kondisi.

Tetapkan nilai selama percakapan

Amazon Lex V2 menyediakan kemampuan untuk mengatur nilai slot dan nilai atribut sesi di setiap langkah percakapan. Anda kemudian dapat menggunakan nilai-nilai ini selama percakapan untuk mengevaluasi kondisi atau menggunakannya selama pemenuhan maksud.

Anda dapat mengatur nilai slot untuk maksud saat ini. Jika langkah selanjutnya dalam percakapan adalah memanggil intent lain, Anda dapat menetapkan nilai slot dari intent baru.

Jika slot yang ditetapkan tidak diisi, atau jika jalur JSON tidak dapat diuraikan, maka atribut akan disetel ke. `null`

Gunakan sintaks berikut saat menggunakan nilai slot dan atribut sesi:

- Nilai slot — mengelilingi nama slot dengan kawat gigi (“{}”). Untuk nilai slot di intent saat ini, Anda hanya perlu menggunakan nama slot. Misalnya, `{slot}`. Jika Anda menetapkan nilai di intent

berikutnya, Anda harus menggunakan nama maksud dan nama slot untuk mengidentifikasi slot. Misalnya, `{intent.slot}`.

Contoh:

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

Nilai slot dapat berupa salah satu dari yang berikut:

- string konstan
- jalur JSON yang mengacu pada blok transkripsi dalam respons Amazon Lex (untuk en-US dan en-GB)
- atribut sesi

Contoh:

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

> **Note**
>
> Nilai slot juga dapat diatur ke`null`. Jika Anda perlu mendapatkan kembali nilai slot yang telah diisi, Anda harus menetapkan nilainya `null` sebelum meminta pelanggan untuk nilai slot lagi. Jika slot yang ditetapkan tidak diisi, atau jika jalur JSON tidak dapat diuraikan, maka atribut akan disetel ke. `null`

- Atribut sesi — mengelilingi nama atribut dengan tanda kurung siku (“[]”). Misalnya, `[sessionAttribute]`.

  Contoh:

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

  Nilai atribut sesi dapat berupa salah satu dari berikut ini:

- string konstan
- jalur JSON yang mengacu pada blok transkripsi dalam respons Amazon Lex (untuk en-US dan en-GB)
- referensi nilai slot

> **Note**
>
> Jika slot yang ditetapkan tidak diisi, atau jika jalur JSON tidak dapat diuraikan, maka atribut akan disetel ke. `null`

> **Note**
>
> Pada 17 Agustus 2022, Amazon Lex V2 merilis perubahan pada cara percakapan dikelola dengan pengguna. Perubahan ini memberi Anda kontrol lebih besar atas jalur yang diambil pengguna melalui percakapan. Untuk informasi selengkapnya, lihat Perubahan alur percakapan di Amazon Lex V2. Bot yang dibuat sebelum 17 Agustus 2022 tidak mendukung pesan kait kode dialog, menyetel nilai, mengonfigurasi langkah selanjutnya, dan menambahkan kondisi.

Tambahkan kondisi ke percakapan cabang

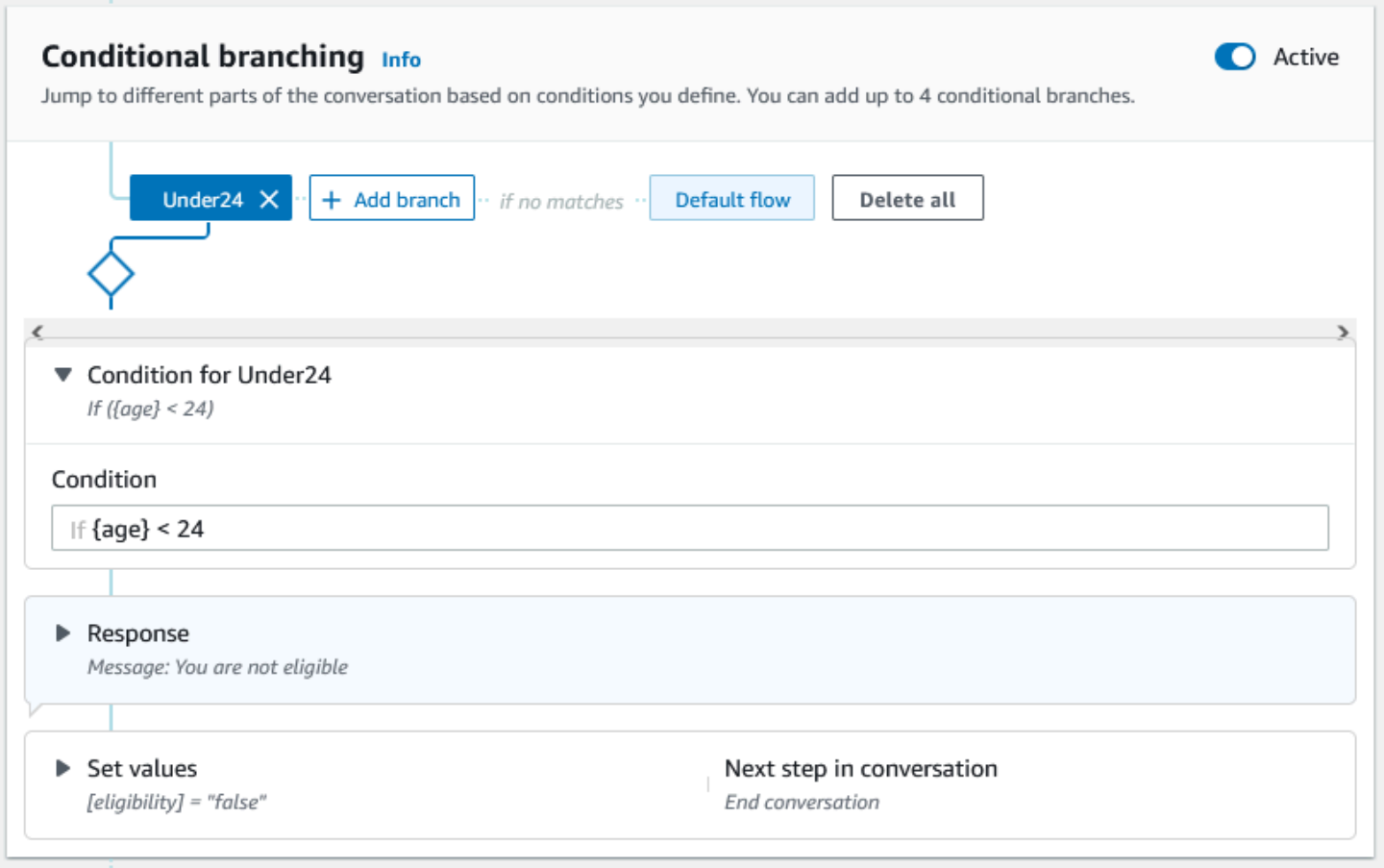
Anda dapat menggunakan percabangan bersyarat untuk mengontrol jalur yang diambil pelanggan Anda melalui percakapan dengan bot Anda. Anda dapat mem-cabang percakapan berdasarkan nilai slot, atribut sesi, isi mode input dan bidang transkrip input, atau respons dari Amazon Kendra.

Anda dapat menentukan hingga empat cabang. Setiap cabang memiliki kondisi yang harus dipenuhi agar Amazon Lex V2 mengikuti cabang itu. Jika tidak ada cabang yang kondisinya terpenuhi, cabang default diikuti.

Saat Anda mendefinisikan cabang, Anda menentukan tindakan yang harus dilakukan Amazon Lex V2 jika kondisi yang sesuai dengan cabang tersebut dievaluasi menjadi true. Anda dapat menentukan salah satu tindakan berikut:

- Tanggapan dikirim ke pengguna.
- Nilai slot untuk diterapkan ke slot.

- Nilai atribut sesi untuk sesi saat ini.
- Langkah selanjutnya dalam percakapan. Untuk informasi selengkapnya, lihat Membuat jalur percakapan.



Setiap cabang kondisional memiliki ekspresi Boolean yang harus dipenuhi untuk Amazon Lex V2 untuk mengikuti cabang. Ada operator perbandingan dan Boolean, fungsi, dan operator kuantifier yang dapat Anda gunakan untuk kondisi Anda. Misalnya, kondisi berikut mengembalikan nilai true jika slot {age} kurang dari 24.

```
{age} < 24
```

Kondisi berikut mengembalikan nilai true jika slot multi-nilai {toppings} berisi kata “nanas”.

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

Anda dapat menggabungkan beberapa operator perbandingan dengan operator Boolean untuk kondisi yang lebih kompleks. Misalnya, kondisi berikut mengembalikan nilai true jika nilai slot {make}

adalah “Honda” dan nilai slot {model} adalah “Civic”. Gunakan tanda kurung untuk mengatur urutan evaluasi.

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

Topik berikut memberikan rincian tentang operator dan fungsi cabang bersyarat.

> **Note**
>
> Pada 17 Agustus 2022, Amazon Lex V2 merilis perubahan pada cara percakapan dikelola dengan pengguna. Perubahan ini memberi Anda kontrol lebih besar atas jalur yang diambil pengguna melalui percakapan. Untuk informasi selengkapnya, lihat Perubahan alur percakapan di Amazon Lex V2. Bot yang dibuat sebelum 17 Agustus 2022 tidak mendukung pesan kait kode dialog, menyetel nilai, mengonfigurasi langkah selanjutnya, dan menambahkan kondisi.

Topik

- Operator perbandingan
- Operator Boolean
- Operator kuantifier
- Fungsi
- Contoh ekspresi bersyarat

Operator perbandingan

Amazon Lex V2 mendukung operator perbandingan berikut untuk kondisi:

- Sama dengan (=)
- Tidak sama (! =)
- Kurang dari (<)
- Kurang dari atau sama (<=)
- Lebih besar dari (>)
- Lebih besar dari atau sama (>=)

Saat menggunakan operator perbandingan, ia menggunakan aturan berikut.

- Sisi kiri harus menjadi referensi. Misalnya, untuk referensi nilai slot, Anda menggunakan`{slotName}`. Untuk mereferensikan nilai atribut sesi, Anda menggunakan`[attribute]`. Untuk mode input dan transkrip input, Anda menggunakan `$.inputMode` dan`$.inputTranscript`.
- Sisi kanan harus konstan dan tipe yang sama dengan sisi kiri.
- Ekspresi apa pun yang mereferensikan atribut yang belum disetel diperlakukan sebagai tidak valid, dan tidak dievaluasi.
- Ketika Anda membandingkan slot multi-nilai, nilai yang digunakan adalah daftar dipisahkan koma dari semua nilai yang ditafsirkan.

Perbandingan didasarkan pada jenis slot referensi. Mereka diselesaikan sebagai berikut:

- String — string dibandingkan berdasarkan representasi ASCII mereka. Perbandingannya tidak peka huruf besar/kecil.
- Nomor — slot berbasis angka dikonversi dari representasi string ke angka dan kemudian dibandingkan.
- Tanggal/Waktu — slot berbasis waktu dibandingkan berdasarkan deret waktu. Tanggal atau waktu sebelumnya dianggap lebih kecil. Untuk jangka waktu, periode yang lebih pendek dianggap lebih kecil.

Operator Boolean

Amazon Lex V2 mendukung operator Boolean untuk menggabungkan operator perbandingan. Mereka memungkinkan Anda membuat pernyataan yang mirip dengan berikut ini:

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

Anda dapat menggunakan operator Boolean berikut:

- DAN (&&)
- ATAU (||)
- TIDAK (!)

Operator kuantifier

Operator kuantifier mengevaluasi elemen urutan dan menentukan apakah satu atau lebih elemen memenuhi kondisi.

- CONTAINS — menentukan apakah nilai yang ditentukan terkandung dalam slot multi-nilai dan mengembalikan true jika itu. Misalnya, `{toppings} CONTAINS "pineapple"` mengembalikan true jika pengguna memesan nanas di pizza mereka.

Fungsi

Fungsi harus diawali dengan string`fn.`. Argumen untuk fungsi adalah referensi ke slot, atribut sesi, atau atribut permintaan. Amazon Lex V2 menyediakan dua fungsi untuk mendapatkan informasi dari nilai slot, SessionAttribute, atau RequestAttribute.

- Fn.count () — menghitung jumlah nilai dalam slot multi-nilai.

  Misalnya, jika slot `{toppings}` berisi nilai “pepperoni, nanas”:

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

- fn.is_set () — nilai true jika slot, atribut sesi, atau atribut permintaan diatur dalam sesi saat ini.

  Berdasarkan contoh sebelumnya:

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

- fn.length () - nilai adalah panjang dari nilai atribut sesi, nilai slot, atau atribut slot yang diatur dalam sesi saat ini. Fungsi ini tidak mendukung slot multi-nilai atau slot komposit.

  Contoh:

  Jika slot `{credit-card-number}` berisi nilai “123456781234":

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

Contoh ekspresi bersyarat

Berikut adalah beberapa contoh ekspresi kondisional. CATATAN: `$.` mewakili titik masuk ke respons Amazon Lex V2 JSON. Nilai berikut `$.` akan diuraikan dalam respons Amazon Lex V2 untuk mengambil nilainya. Ekspresi bersyarat menggunakan referensi jalur JSON ke blok transkripsi di

respons Amazon Lex V2 hanya akan didukung di lokal yang sama yang mendukung skor transkripsi ASR.

Tipe nilai	Kasus penggunaan	Ekspresi bersyarat
Slot khusus	`pizzaSize` nilai slot sama dengan besar	`{pizzaSize} = "large"`
Slot khusus	`pizzaSize` sama dengan besar atau sedang	`{pizzaSize} = "large"`ATAU `{pizzaSize} = "medium"`
Slot khusus	Ekspresi dengan `( )` dan `AND/OR`	`{pizzaType} = "pepperoni"` ATAU `{pizzaSize} = "medium"` ATAU `{pizzaSize} = "small"`
Slot khusus (Slot Multi-Nilai)	Periksa apakah salah satu toppingnya adalah Onion	`{toppings} CONTAINS "Onion"`
Slot khusus (Slot Multi-Nilai)	Jumlah topping lebih dari 3	`fn.COUNT({topping}) > 2`
`AMAZON.AlphaNumeric`	`bookingID` adalah ABC123	`{bookingID} = "ABC123"`
`AMAZON.Number`	nilai slot usia lebih besar dari 30	`{age} > 30`
`AMAZON.Number`	nilai slot usia sama dengan 10	`{age} = 10`
`AMAZON.Date`	`dateOfBirth` nilai slot sebelum 1990	`{dateOfBirth} < "1990-10-01"`
`AMAZON.State`	`destinationState` nilai slot sama dengan Washington	`{destinationState} = "washington"`

Tipe nilai	Kasus penggunaan	Ekspresi bersyarat
AMAZON.Country	destinationCountry nilai slot bukan Amerika Serikat	{destinationCountry} != "united states"
AMAZON.FirstName	firstName Nilai slot adalah John	{firstName} = "John"
AMAZON.PhoneNumber	phoneNumber nilai slot adalah 716767891932	{phoneNumer} = 716767891932
AMAZON.Percentage	Periksa apakah persentase nilai slot lebih besar dari atau sama dengan 78	{percentage} >= 78
AMAZON.EmailAddress	emailAddress nilai slot adalah userA@hmail.com	{emailAddress} = "userA@hmail.com"
AMAZON.LastName	lastNamenilai slot adalah Doe	{lastName} = "Doe"
AMAZON.City	Nilai slot kota sama dengan Seattle	{city} = "Seattle"
AMAZON.Time	Waktu setelah jam 8 malam	{time} > "20:00"
AMAZON.StreetName	streetName nilai slot adalah Boren Avenue	{streetName} = "boren avenue"
AMAZON.Duration	travelDuration nilai slot kurang dari 2 jam	{travelDuration} < P2H
Modus masukan	Mode input adalah ucapan	$.inputMode = "Speech"
Transkrip masukan	Transkrip masukan sama dengan “Saya ingin pizza besar”	$.inputTranscript = "I want a large pizza"

Tipe nilai	Kasus penggunaan	Ekspresi bersyarat
Atribut sesi	periksa atribut customer_subscription_type	`[customer_subcription_type] = "yearly"`
Atribut permintaan	periksa bendera retry_enabled	`((retry_enabled)) = "TRUE"`
Tanggapan Kendra	Tanggapan Kendra berisi FAQ	`fn.IS_SET(((x-amz-lex:kendra-search-response-question_answer-question-1)))`
Ekspresi bersyarat dengan transkripsi	Ekspresi bersyarat menggunakan transkripsi jalur JSON	`$.transcriptions[0].transcriptionConfidence < 0.8 AND $.transcriptions[1].transcriptionConfidence > 0.5`
Tetapkan atribut sesi	Atur atribut sesi menggunakan transkripsi jalur JSON dan nilai slot	`[sessionAttribute] = "$.transcriptions...." AND [sessionAttribute] = "{<slotName>}"`
Tetapkan nilai slot	Tetapkan nilai slot menggunakan atribut sesi dan transkripsi jalur JSON	`{slotName} = [<sessionAttribute>] AND {slotName} = "$.transcriptions...."`

> **Note**
>
> `slotName`mengacu pada nama slot di bot Amazon Lex V2. Jika slot tidak diselesaikan (null), atau jika slot tidak ada, maka tugas diabaikan saat runtime. `sessionAttribute`mengacu pada nama atribut sesi yang ditetapkan oleh pelanggan pada waktu pembuatan.

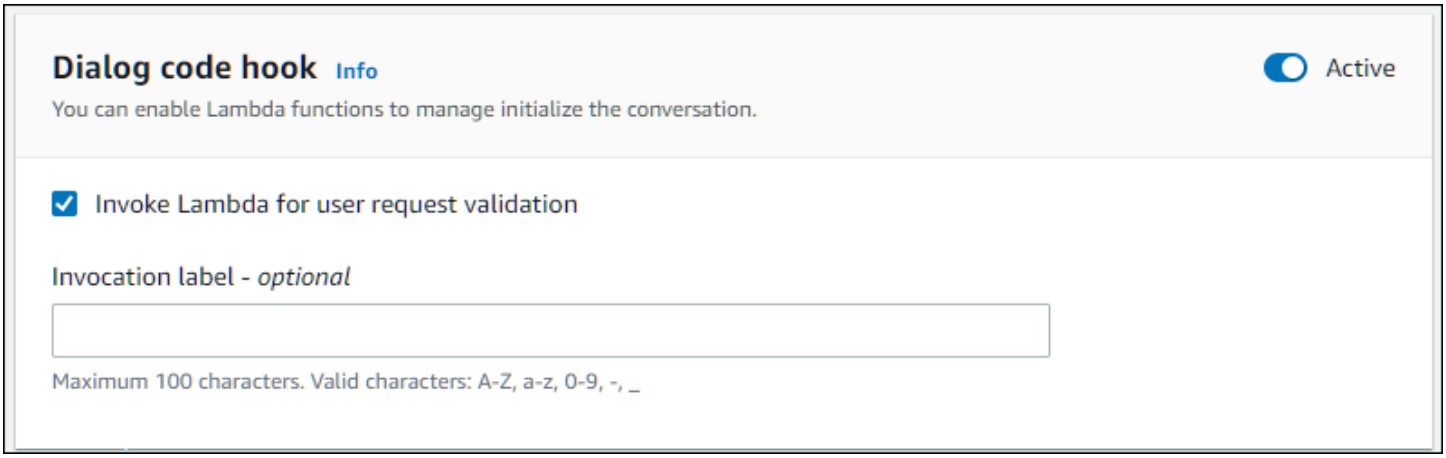
Memanggil hook kode dialog

Pada setiap langkah dalam percakapan ketika Amazon Lex V2 mengirim pesan ke pengguna, Anda dapat menggunakan fungsi Lambda sebagai langkah berikutnya dalam percakapan. Anda dapat menggunakan fungsi untuk mengimplementasikan logika bisnis berdasarkan keadaan percakapan saat ini.

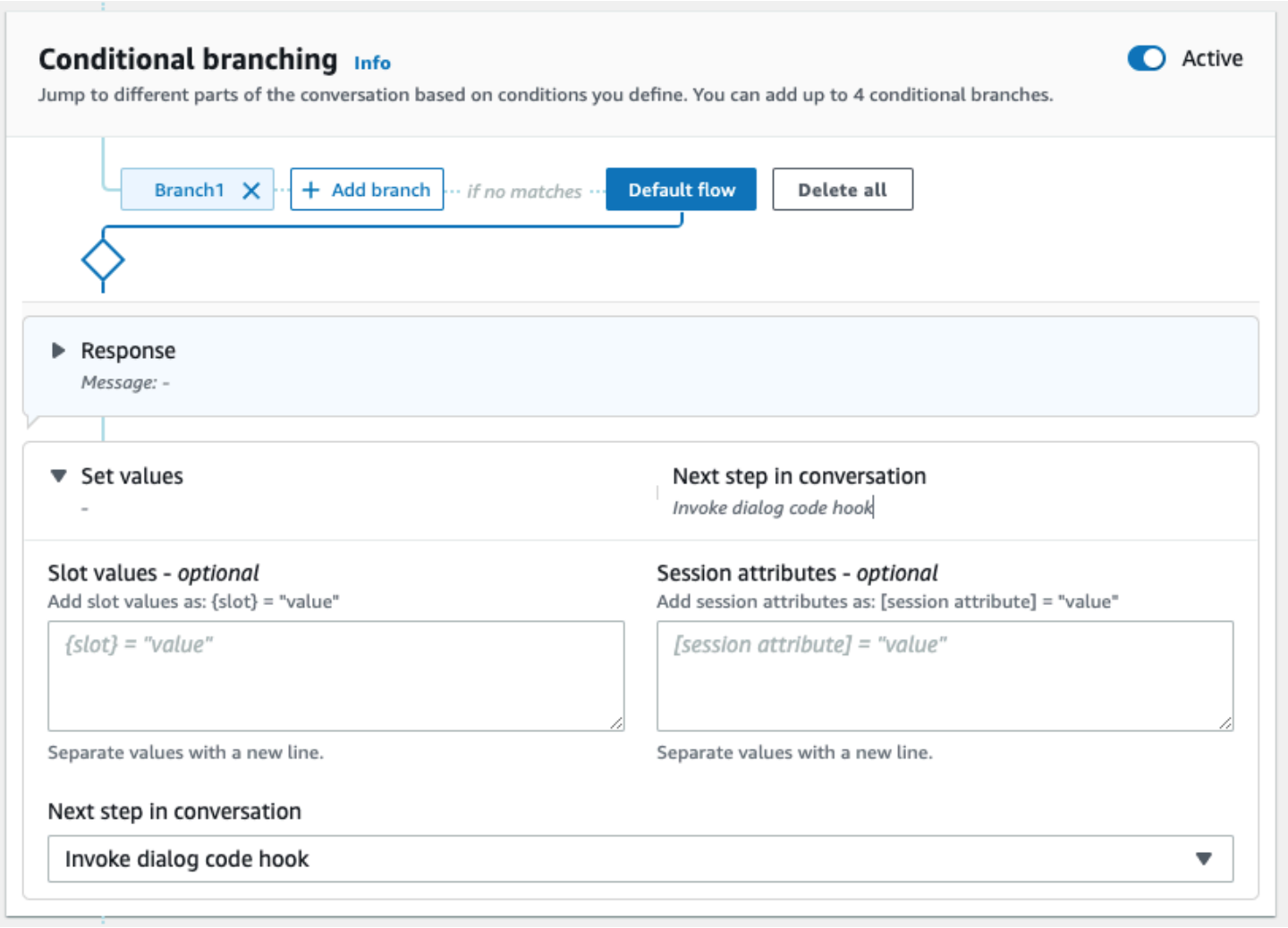
Fungsi Lambda yang berjalan dikaitkan dengan alias bot yang Anda gunakan. Untuk menjalankan fungsi Lambda di semua kait kode dialog di intent Anda, Anda harus memilih Gunakan fungsi Lambda untuk menginisialisasi dan memvalidasi maksud. Untuk informasi selengkapnya tentang memilih fungsi Lambda, lihat. Membuat AWS Lambda fungsi untuk bot Amazon Lex V2 Anda

Ada dua langkah untuk menggunakan fungsi Lambda. Pertama, Anda harus mengaktifkan hook kode dialog kapan saja dalam percakapan. Kedua, Anda harus mengatur langkah berikutnya dalam percakapan untuk menggunakan hook kode dialog.

Gambar berikut menunjukkan kode dialog hook diaktifkan.

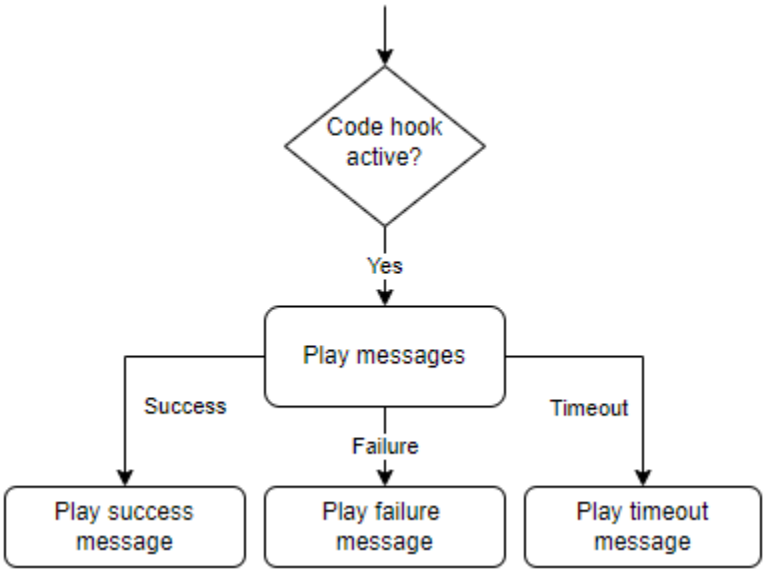


Selanjutnya, atur kode hook sebagai tindakan selanjutnya untuk langkah percakapan. Anda dapat melakukan ini dengan mengonfigurasi langkah berikutnya dalam percakapan ke hook kode dialog Invoke. Gambar berikut menunjukkan cabang bersyarat di mana memanggil hook kode dialog adalah langkah berikutnya untuk jalur default percakapan.



Saat kait kode aktif, Anda dapat mengatur tiga respons untuk kembali ke pengguna:

- Sukses - Dikirim ketika fungsi Lambda selesai dengan sukses.
- Kegagalan - Dikirim jika ada masalah dengan menjalankan fungsi Lambda, atau fungsi Lambda mengembalikan nilai. `intent.state` `Failed`
- Timeout - Dikirim jika fungsi Lambda tidak selesai dalam periode batas waktu yang dikonfigurasi.



Pilih kait kode dialog Lambda dan kemudian pilih Opsi lanjutan untuk melihat tiga opsi respons yang sesuai dengan pemanggilan fungsi Lambda. Anda dapat menetapkan nilai, mengonfigurasi langkah berikutnya, dan menerapkan kondisi yang sesuai dengan setiap respons untuk merancang alur percakapan. Dengan tidak adanya kondisi atau langkah eksplisit berikutnya, Amazon Lex V2 memutuskan langkah selanjutnya berdasarkan keadaan percakapan saat ini.

Pada halaman Opsi lanjutan Anda juga dapat memilih untuk mengaktifkan atau menonaktifkan pemanggilan fungsi Lambda Anda. Ketika fungsi diaktifkan, hook kode dialog dipanggil dengan pemanggilan Lambda, diikuti oleh pesan sukses, kegagalan atau batas waktu berdasarkan hasil pemanggilan Lambda. Ketika fungsi dinonaktifkan, Amazon Lex V2 tidak menjalankan fungsi Lambda dan melanjutkan seolah-olah hook kode dialog berhasil.

Anda juga dapat mengatur label pemanggilan yang dikirim ke fungsi Lambda saat dipanggil oleh pesan ini. Anda dapat menggunakan ini untuk membantu mengidentifikasi bagian fungsi Lambda Anda untuk dijalankan.

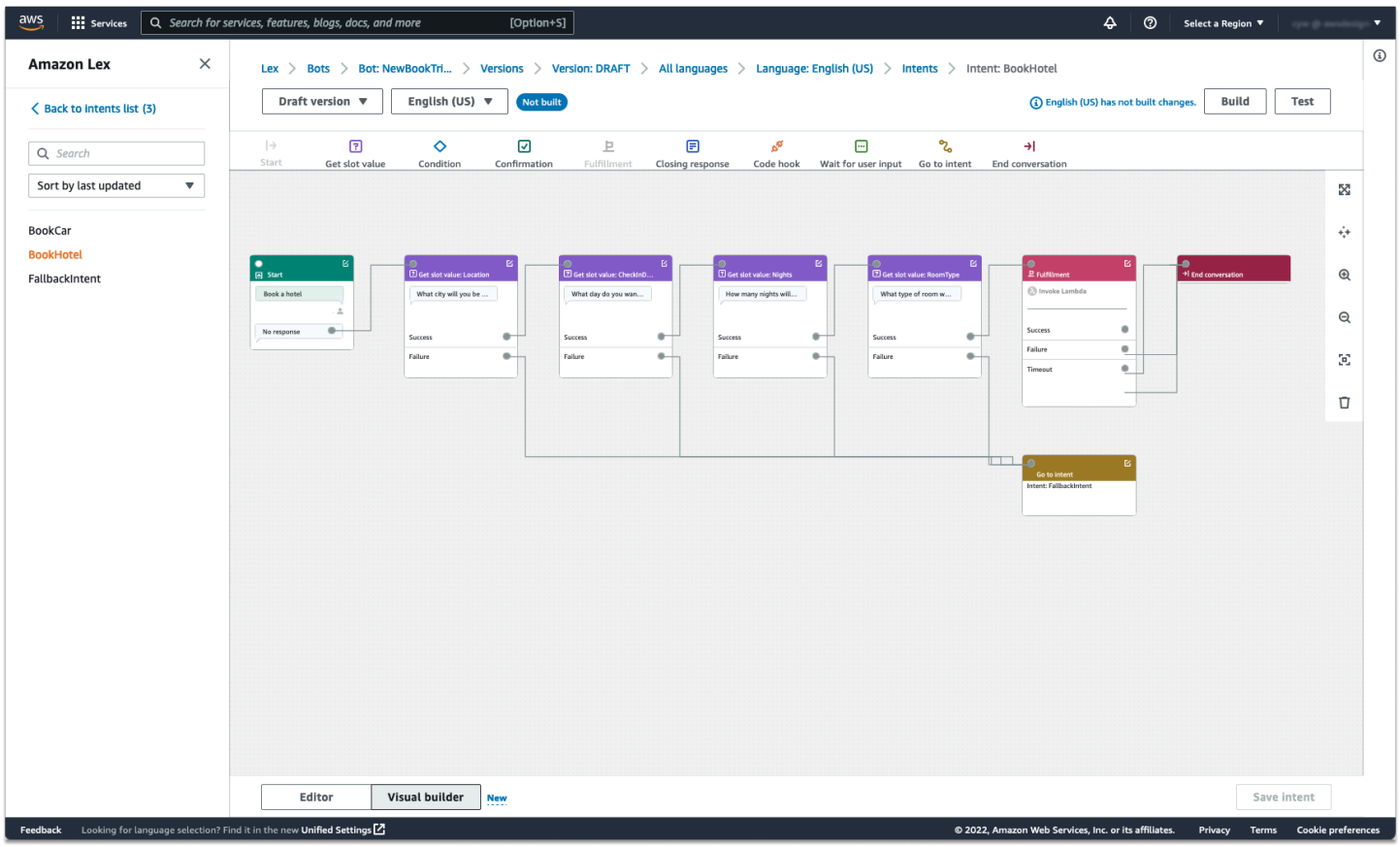
> **Note**
>
> Pada 17 Agustus 2022, Amazon Lex V2 merilis perubahan pada cara percakapan dikelola dengan pengguna. Perubahan ini memberi Anda kontrol lebih besar atas jalur yang diambil pengguna melalui percakapan. Untuk informasi selengkapnya, lihat Perubahan alur percakapan di Amazon Lex V2. Bot yang dibuat sebelum 17 Agustus 2022 tidak mendukung pesan kait kode dialog, menyetel nilai, mengonfigurasi langkah selanjutnya, dan menambahkan kondisi.

Menggunakan pembangun percakapan Visual

Pembuat percakapan visual adalah pembuat percakapan seret dan lepas untuk merancang dan memvisualisasikan jalur percakapan dengan mudah dengan menggunakan maksud dalam lingkungan visual yang kaya.

Untuk mengakses pembuat percakapan visual

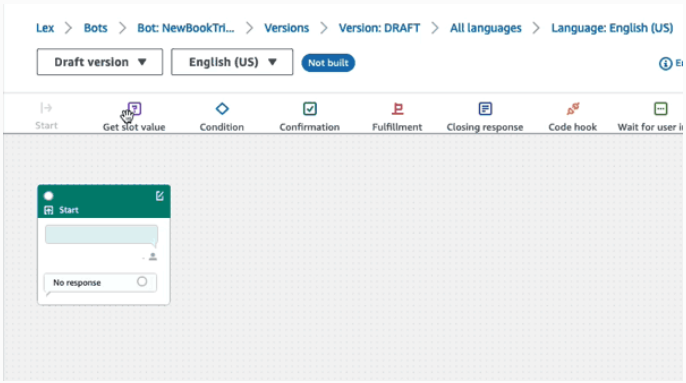
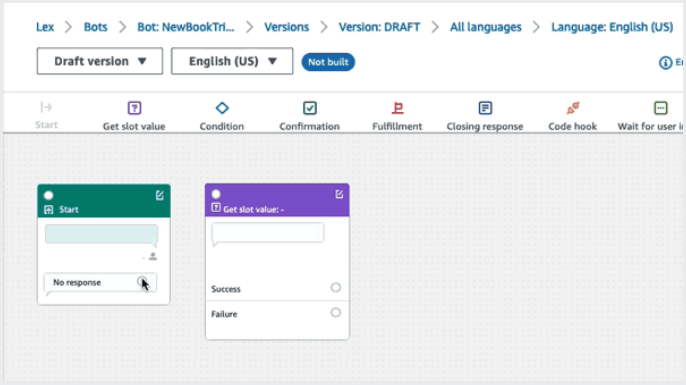
1. Di konsol Amazon Lex V2, pilih bot dan pilih Maksud dari panel navigasi kiri.
2. Buka editor maksud dengan salah satu cara berikut:
   - Pilih Tambahkan maksud di sudut kanan atas bagian Intent, lalu pilih untuk menambahkan maksud kosong atau intent bawaan.
   - Pilih nama maksud dari bagian Intent.
3. Di editor maksud, pilih Pembuat visual di panel di bagian bawah layar untuk mengakses pembangun percakapan Visual.
4. Untuk kembali ke antarmuka editor intent menu, pilih Editor.

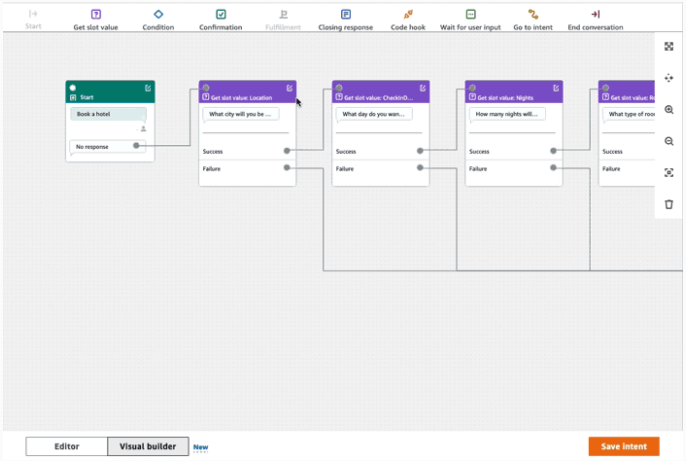
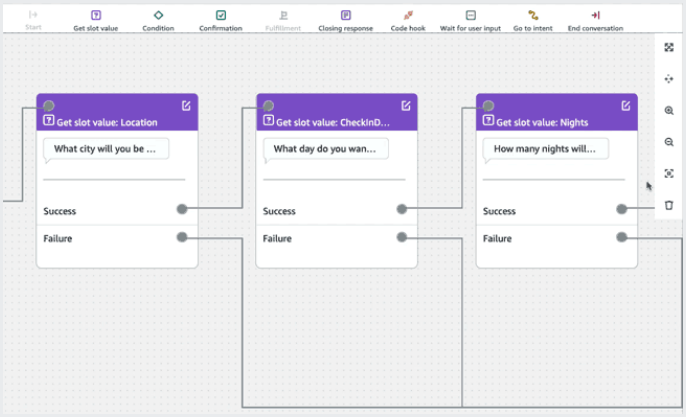
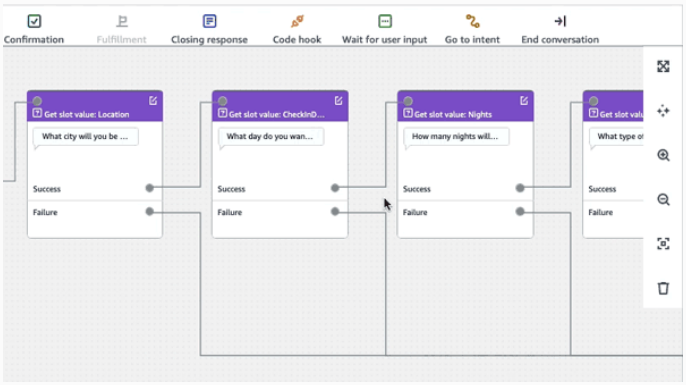


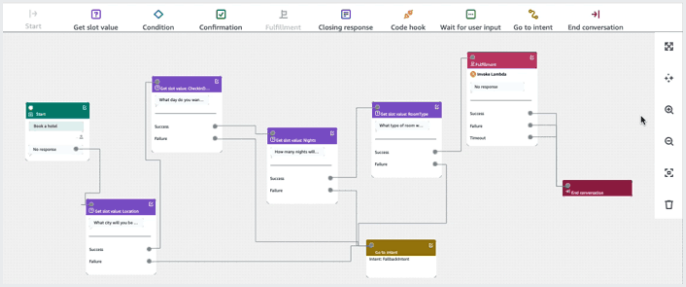
Pembuat percakapan visual menawarkan antarmuka pengguna yang lebih intuitif dengan kemampuan untuk memvisualisasikan dan memodifikasi alur percakapan. Dengan menyeret dan menjatuhkan blok, Anda dapat memperluas alur yang ada atau menyusun ulang langkah-langkah percakapan. Anda dapat mengembangkan alur percakapan dengan percabangan yang kompleks tanpa menulis kode Lambda apa pun.

Perubahan ini membantu memisahkan desain alur percakapan dari logika bisnis lain di Lambda. Pembuat percakapan visual dapat digunakan bersama dengan editor maksud yang ada dan dapat digunakan untuk membangun alur percakapan. Namun, disarankan untuk menggunakan tampilan editor visual untuk alur percakapan yang lebih kompleks.

Saat Anda menyimpan intent, Amazon Lex V2 dapat menyambungkan intent secara otomatis ketika menentukan bahwa ada koneksi yang tidak terjawab, Amazon Lex V2 menyarankan koneksi, atau Anda dapat memilih koneksi Anda sendiri untuk blok tersebut.

Tindakan	Contoh
Menambahkan blok ke ruang kerja	
Membuat koneksi antar blok	

Tindakan	Contoh
Membuka panel konfigurasi pada blok	
Zoom agar pas	
Menghapus blok dari alur percakapan	

Tindakan	Contoh
Bersihkan ruang kerja secara otomatis	

Terminologi:

Blok — Unit bangunan dasar dari alur percakapan. Setiap blok memiliki fungsi khusus untuk menangani kasus penggunaan percakapan yang berbeda.

Port — Setiap blok berisi port, yang dapat digunakan untuk menghubungkan satu blok ke blok lainnya. Blok dapat berisi port input dan port output. Setiap port output mewakili variasi fungsional tertentu dari blok (seperti kesalahan, batas waktu, atau keberhasilan).

Edge — Edge adalah koneksi antara port output dari satu blok ke port input blok lain. Ini adalah bagian dari cabang dalam alur percakapan.

Alur percakapan — Satu set blok yang dihubungkan oleh tepi yang menggambarkan interaksi tingkat maksud dengan pelanggan.

Blok

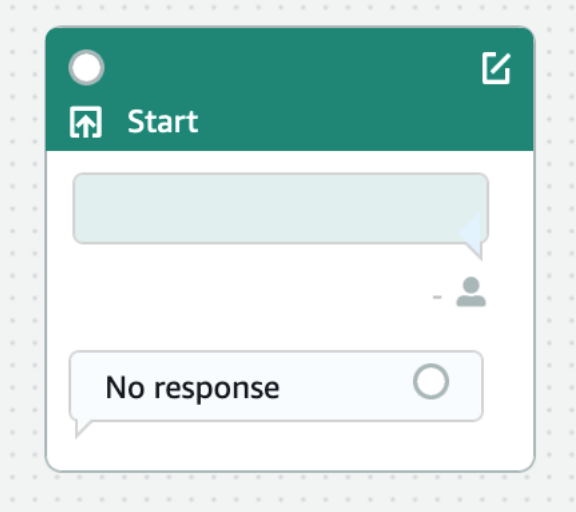
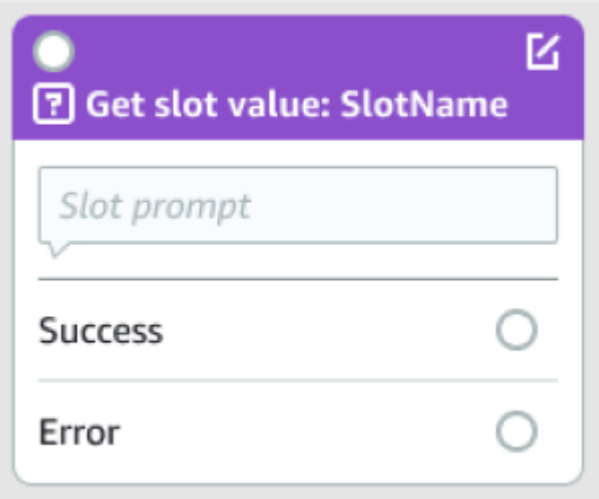
Blok adalah blok bangunan dari desain alur percakapan. Mereka mewakili status yang berbeda dalam maksud, yang mencakup dari awal maksud, input pengguna, hingga penutupan.

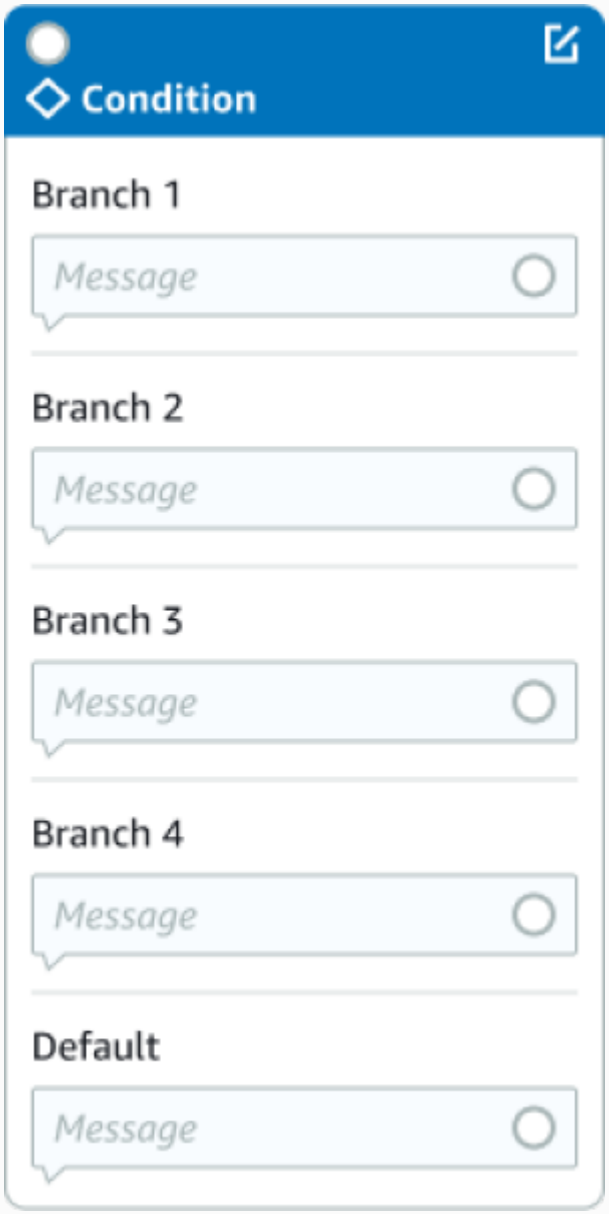
Setiap blok memiliki titik masuk dan satu atau banyak titik keluar berdasarkan jenis blok. Setiap titik keluar dapat dikonfigurasi dengan pesan yang sesuai saat percakapan berlangsung melalui titik keluar. Untuk blok dengan beberapa titik keluar, titik keluar berhubungan dengan status yang sesuai dengan node. Untuk node kondisi, titik keluar mewakili kondisi yang berbeda.

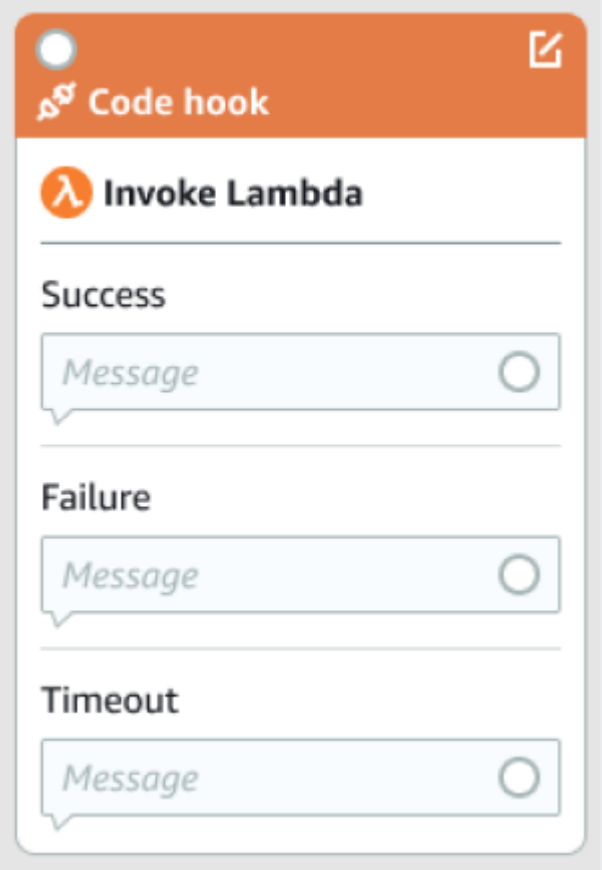
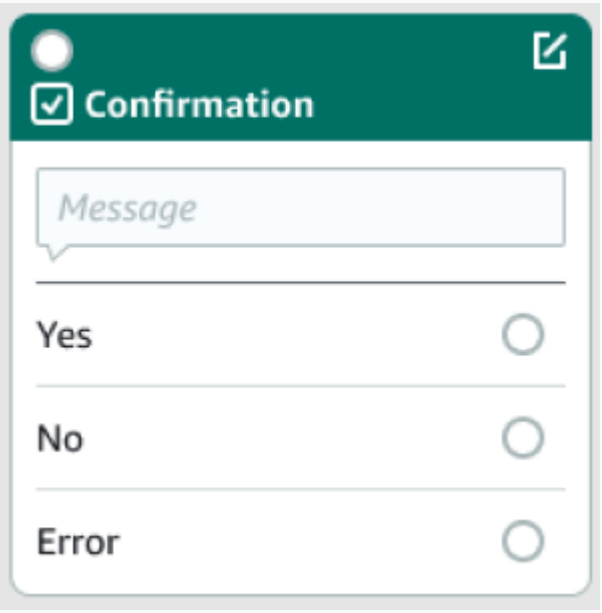
Setiap blok memiliki panel konfigurasi, yang terbuka dengan mengklik ikon Edit di sudut kanan atas blok. Panel konfigurasi berisi bidang terperinci yang dapat dikonfigurasi agar sesuai dengan setiap blok.

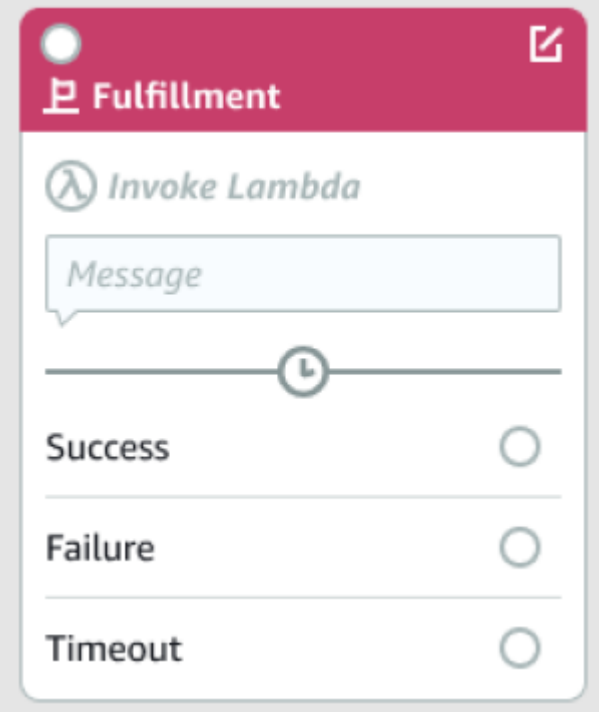
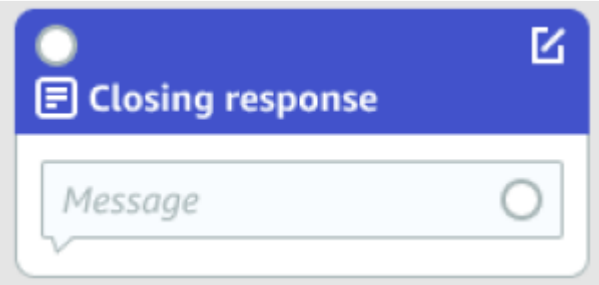
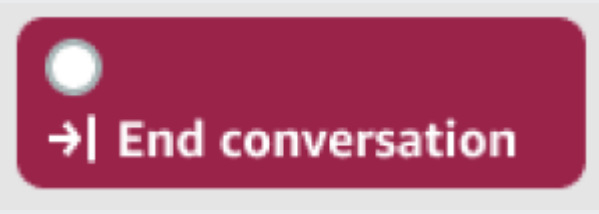
Permintaan bot dan pesan dapat dikonfigurasi langsung pada node dengan menyeret blok baru, atau mereka dapat dimodifikasi dalam panel kanan, bersama dengan atribut lain dari blok.

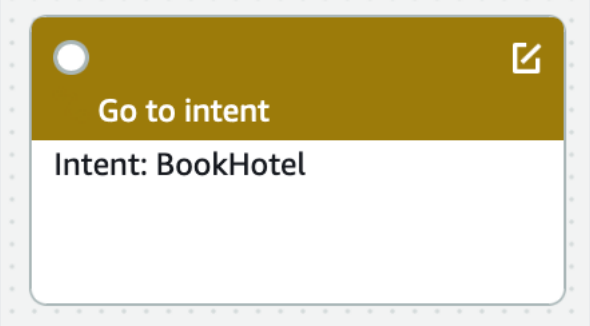
Jenis blok — Berikut adalah jenis blok yang dapat Anda gunakan dengan pembuat percakapan visual.

Jenis Blok	Blokir
Mulai — Akar atau blok pertama dari alur percakapan. Blok ini juga dapat dikonfigurasi sedemikian rupa sehingga bot dapat mengirim respons awal (pesan maksud telah dikenali) . Untuk informasi selengkapnya, lihat Respon awal.	
Dapatkan nilai slot — Blok ini mencoba memperoleh nilai untuk satu slot. Blok ini memiliki pengaturan untuk menunggu respons pelanggan terhadap prompt elicitation slot. Untuk informasi selengkapnya, lihat Slot.	

Jenis Blok	Blokir
Kondisi - Blok ini berisi kondisional. Ini berisi hingga 4 cabang khusus (dengan kondisi) dan satu cabang default. Untuk informasi selengkap nya, lihat Tambahkan kondisi ke percakapan cabang.	

Jenis Blok	Blokir
Kait kode dialog - Blok ini menangani pemanggilan fungsi dialog Lambda. Blok ini berisi respons bot berdasarkan dialog fungsi Lambda yang berhasil, gagal, atau habis waktu. Untuk informasi selengkapnya, lihat Memanggil hook kode dialog.	
Konfirmasi — Blok ini menanyakan pelanggan sebelum memenuhi maksud. Ini berisi respons bot berdasarkan pelanggan yang mengataka n ya atau tidak pada prompt konfirmasi. Untuk informasi selengkapnya, lihat Konfirmasi.	

Jenis Blok	Blokir
Pemenuhan — Blok ini menangani pemenuhan niat, biasanya setelah elisitasi slot. Ini dapat dikonfigurasi untuk memanggil fungsi Lambda, serta merespons dengan pesan, jika pemenuhan berhasil atau gagal. Untuk informasi selengkapnya, lihat Pemenuhan.	
Respons penutupan — Blok ini memungkinkan bot untuk merespons dengan pesan sebelum mengakhiri percakapan. Untuk informasi selengkapnya, lihat Tanggapan penutupan.	
Akhiri percakapan — Blok ini menunjukkan akhir alur percakapan.	
Tunggu masukan pengguna — Blok ini dapat digunakan untuk menangkap input dari pelanggan dan beralih ke maksud lain berdasarkan ucapan.	

Jenis Blok	Blokir
Pergi ke intent — Blok ini dapat digunakan untuk pergi ke maksud baru, atau untuk secara langsung mendapatkan slot tertentu dari maksud itu.	

Jenis port

Semua blok berisi satu port input, yang digunakan untuk menghubungkan blok induknya. Percakapan hanya dapat mengalir ke port input blok tertentu dari port output blok induknya. Namun, blok dapat berisi nol, satu, atau banyak port output. Blok tanpa port output menandakan akhir alur percakapan di intent saat ini (`GoToIntent`,`EndConversation`,`WaitForUserInput`).

Aturan desain niat:

- Semua aliran dalam maksud dimulai dengan blok awal.
- Pesan yang sesuai dengan setiap titik keluar bersifat opsional.
- Anda dapat mengonfigurasi blok untuk menetapkan nilai yang sesuai dengan setiap titik keluar di panel konfigurasi.
- Hanya satu blok awal, konfirmasi, pemenuhan, dan penutupan yang dapat ada dalam satu aliran dalam suatu maksud. Beberapa kondisi, kait kode dialog, dapatkan nilai slot, akhiri percakapan, transfer, dan tunggu blok input pengguna mungkin ada.
- Blok kondisi tidak dapat memiliki koneksi langsung ke blok kondisi. Hal yang sama berlaku untuk hook kode dialog.
- Aliran melingkar diperbolehkan tiga blok, tetapi konektor masuk ke Start Intent tidak diperbolehkan.
- Slot opsional tidak memiliki konektor masuk atau koneksi keluar dan terutama digunakan untuk menangkap data apa pun yang ada selama elisitasi maksud. Setiap slot lain yang merupakan bagian dari jalur percakapan harus menjadi slot wajib.

Blok:

- Blok awal harus memiliki tepi keluar.

- Setiap blok nilai slot get harus memiliki tepi keluar dari port sukses, jika slot diperlukan.
- Setiap blok kondisi harus memiliki tepi keluar dari setiap cabang jika blok aktif.
- Blok kondisi tidak dapat memiliki lebih dari satu induk.
- Blok kondisi aktif harus memiliki tepi yang masuk.
- Setiap blok hook kode aktif harus memiliki tepi keluar dari setiap port: keberhasilan, kegagalan, dan batas waktu.
- Blok kait kode aktif harus memiliki tepi yang masuk.
- Blok konfirmasi aktif harus memiliki tepi yang masuk.
- Blok pemenuhan aktif harus memiliki tepi yang masuk.
- Blok penutupan aktif harus memiliki tepi yang masuk.
- Blok kondisi harus memiliki setidaknya satu cabang non-default.
- Blok go to intent harus memiliki maksud yang ditentukan.

Tepi:

- Blok kondisi tidak dapat dihubungkan ke blok kondisi lain.
- Blok kait kode tidak dapat dihubungkan ke blok kait kode lain.
- Blok kondisi hanya dapat dihubungkan ke nol atau satu blok kait kode.
- Koneksi (code hook -> condition -> code hook) tidak valid.
- Blok pemenuhan tidak dapat memiliki blok kait kode sebagai anak.
- Blok kondisi, yang merupakan anak dari blok pemenuhan, tidak dapat memiliki anak blok kait kode.
- Blok penutup tidak dapat memiliki blok kait kode sebagai anak.
- Blok kondisi yang merupakan anak dari blok penutup tidak dapat memiliki anak blok kait kode.
- Blok nilai awal, konfirmasi, atau dapatkan slot dapat memiliki tidak lebih dari satu blok kait kode dalam rantai ketergantungannya.

> **Note**
>
> Pada 17 Agustus 2022, Amazon Lex V2 merilis perubahan pada cara percakapan dikelola dengan pengguna. Perubahan ini memberi Anda kontrol lebih besar atas jalur yang diambil pengguna melalui percakapan. Untuk informasi selengkapnya, lihat Perubahan alur percakapan di Amazon Lex V2. Bot yang dibuat sebelum 17 Agustus 2022 tidak

mendukung pesan kait kode dialog, menyetel nilai, mengonfigurasi langkah selanjutnya, dan menambahkan kondisi.

Maksud bawaan

Untuk tindakan umum, Anda dapat menggunakan pustaka maksud bawaan standar. Untuk membuat intent dari intent bawaan, pilih intent bawaan di konsol, dan beri nama baru. Maksud baru memiliki konfigurasi maksud dasar, seperti contoh ucapan.

Dalam implementasi saat ini, Anda tidak dapat melakukan hal berikut:

- Menambahkan atau menghapus contoh ucapan dari maksud dasar
- Konfigurasikan slot untuk maksud bawaan

Untuk menambahkan intent bawaan ke bot

1. Masuk ke AWS Management Console dan buka konsol Amazon Lex di https://console.aws.amazon.com/lex/.
2. Pilih bot untuk menambahkan maksud bawaan.
3. Di menu sebelah kiri, pilih bahasa dan kemudian pilih Maksud.
4. Pilih Tambahkan maksud, lalu pilih Gunakan maksud bawaan.
5. Di Intent bawaan, pilih intent yang akan digunakan.
6. Beri maksud nama, lalu pilih Tambah.
7. Gunakan editor intent untuk mengonfigurasi intent seperti yang diperlukan untuk bot Anda.

Topik

- 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**
>
> Sebelum Anda dapat memanfaatkan fitur AI generatif, Anda harus memenuhi prasyarat berikut
>
> 1. Untuk informasi tentang harga untuk menggunakan Amazon Bedrock, lihat harga Amazon Bedrock.
> 2. Aktifkan kemampuan AI generatif untuk lokal bot Anda. Untuk melakukannya, ikuti langkah-langkah diOptimalkan pembuatan dan kinerja bot Lex V2 dengan menggunakan AI generatif.

Mengaktifkan Agen Amazon Bedrock yang ditentukan dalam maksud untuk menanggapi permintaan pelanggan dan mengaktifkan alur kerja agen untuk mencapai tugas yang ditentukan. Fitur ini tersedia di semua lokal yang didukung Amazon Lex V2 dan semua wilayah komersial di mana Amazon Lex V2 dan Amazon Bedrock Agents hadir.

Jika maksud ini mengesampingkan`FallbackIntent`, maksud diaktifkan ketika ucapan tidak diklasifikasikan ke dalam maksud lain yang ada di bot, jika tidak maka hanya diaktifkan ketika ucapan diklasifikasikan ke dalam maksud ini. Penting untuk dicatat bahwa maksud ini tidak akan diaktifkan untuk ucapan yang terlewat saat memunculkan nilai slot.

Setelah dikenali oleh bot Amazon Lex V2 Anda`AMAZON.BedrockAgentIntent`, mengaktifkan yang ditentukan `BedrockAgent` atau `BedrockKnowledgeBase` untuk menanggapi pelanggan. Jika Anda menggunakan Agen Bedrock Amazon, percakapan tetap ada `BedrockAgentIntent` dan permintaan pengguna diteruskan ke Agen, hingga Agen Bedrock Amazon menentukan bahwa percakapan ditandai. `FINISH` Hanya setelah itu, Amazon Lex V2

mengasumsikan kontrol percakapan dan mematuhi langkah selanjutnya yang ditentukan dalam. `AMAZON.BedrockAgentIntent`

Menanggapi pertanyaan pelanggan dengan menggunakan Agen dan Pangkalan Pengetahuan Amazon Bedrock untuk menjawab pertanyaan pelanggan dan memberikan tanggapan terperinci.

> **Warning**
>
> Anda tidak dapat menggunakan ucapan `AMAZON.BedrockAgentIntent` tanpa sampel, `AMAZON.QnAIntent` tanpa ucapan sampel dan `AMAZON.KendraSearchIntent` di lokal bot yang sama.

Jika Anda memilih maksud ini, Anda mengonfigurasi bidang berikut dan kemudian pilih Tambah untuk menambahkan maksud.

- Amazon Bedrock Agent Id - Pengidentifikasi Agen Batuan Dasar Amazon. Pilih Agen Bedrock yang ingin Anda gunakan.
- Amazon Bedrock Agent Alias Id - Pengidentifikasi Alias dari Agen Batuan Dasar Amazon.

> **Important**
>
> Saat membuat Agen Bedrock Amazon untuk digunakan dengan Amazon Lex V2, verifikasi bahwa Input Pengguna di bawah Pengaturan Tambahan adalah`ENABLED`. Pengaturan ini sangat penting untuk memungkinkan Agen mengajukan pertanyaan klarifikasi atau tindak lanjut, dan memungkinkan Amazon Lex V2 untuk mendelegasikan kembali ke Agen untuk menyelesaikan tugas yang sesuai.

(Opsional) Anda juga dapat menambahkan BedrockAgentIntent dengan opsi ini:

- Model Amazon Bedrock - Pilih penyedia dan model pondasi yang akan digunakan untuk maksud ini. Saat ini, beberapa model Anthropic Claude didukung.
- Basis Pengetahuan Amazon Bedrock - Jika Anda memilih opsi ini, tentukan ID Basis Pengetahuan Amazon Bedrock. Anda dapat menemukan ID dengan memeriksa halaman detail Pangkalan Pengetahuan Amazon Bedrock di konsol, atau dengan mengirimkan `GetKnowledgeBase` permintaan.

Tanggapan dari BedrockAgentIntent akan disimpan ke dalam sesi dan meminta atribut seperti yang ditunjukkan di bawah ini:

- `x-amz-lex:bedrock-agent-search-response`— Tanggapan dari Amazon Bedrock Agent untuk pertanyaan atau ucapan.
- `x-amz-lex:bedrock-knowledge-base-search-response-source`— Menunjuk ke dokumen, atau daftar dokumen, yang digunakan untuk menghasilkan respons jika menggunakan konfigurasi Pangkalan Pengetahuan Amazon Bedrock.
- `x-amz-lex:bedrock-agent-action-group-invocation-input`- Objek yang berisi nilai masukan yang dikumpulkan oleh kelompok aksi agen. Untuk informasi selengkapnya tentang grup aksi agen, lihat ActionGroupInvocationInput.
- `x-amz-lex:bedrock-agent-knowledge-base-lookup-input`- Objek yang berisi rincian terkait pencarian Amazon Bedrock Knowledge Base.
- `x-amz-lex:bedrock-agent-agent-collaborator-details`— Objek yang berisi rincian input dan output dari sub agen yang dipanggil sebagai bagian dari pemanggilan kolaborasi multi-agen.

Untuk informasi selengkapnya, lihat Menggunakan BedrockAgentIntent untuk menggunakan Agen Batuan Dasar di Amazon Lex.

AMAZON.CancelIntent

Menanggapi kata dan frasa yang menunjukkan pengguna ingin membatalkan interaksi saat ini. Aplikasi Anda dapat menggunakan intent ini untuk menghapus nilai jenis slot dan atribut lainnya sebelum mengakhiri interaksi dengan pengguna.

Ucapan umum:

- membatalkan
- tidak pernah keberatan
- lupakan saja

AMAZON.FallbackIntent

Jika input pengguna ke intent tidak seperti yang diharapkan bot, Anda dapat mengonfigurasi Amazon Lex V2 untuk memanggil maksud fallback. Misalnya, jika input pengguna “Saya ingin memesan

permen” tidak cocok dengan maksud di `OrderFlowers` bot Anda, Amazon Lex V2 memanggil maksud fallback untuk menangani respons.

Jenis `AMAZON.FallbackIntent` intent bawaan ditambahkan ke bot Anda secara otomatis saat Anda membuat bot menggunakan konsol atau saat Anda menambahkan lokal ke bot menggunakan operasi. CreateBotLocale

Memanggil maksud fallback menggunakan dua langkah. Pada langkah pertama maksud fallback dicocokkan berdasarkan input dari pengguna. Ketika maksud fallback dicocokkan, cara bot berperilaku bergantung pada jumlah percobaan ulang yang dikonfigurasi untuk prompt.

Amazon Lex V2 cocok dengan maksud mundur dalam situasi ini:

- Input pengguna ke intent tidak cocok dengan input yang diharapkan bot
- Input audio adalah noise, atau input teks tidak dikenali sebagai kata-kata.
- Input pengguna bersifat ambigu dan Amazon Lex V2 tidak dapat menentukan maksud mana yang akan dipanggil.

Maksud fallback dipanggil saat:

- Intent tidak mengenali input pengguna sebagai nilai slot setelah jumlah percobaan yang dikonfigurasi.
- Intent tidak mengenali input pengguna sebagai respons terhadap prompt konfirmasi setelah jumlah percobaan yang dikonfigurasi.

Anda tidak dapat menambahkan yang berikut ini ke maksud fallback:

- Ucapan
- Slot
- Prompt konfirmasi

Menggunakan Fungsi Lambda dengan Maksud Fallback

Ketika maksud fallback dipanggil, respons bergantung pada pengaturan `fulfillmentCodeHook` parameter ke operasi. CreateIntent Bot melakukan salah satu hal berikut:

- Mengembalikan informasi maksud ke aplikasi klien.

- Memanggil validasi alias dan pemenuhan fungsi Lambda. Ini memanggil fungsi dengan variabel sesi yang diatur untuk sesi.

Untuk informasi selengkapnya tentang menyetel respons saat maksud fallback dipanggil, lihat `fulfillmentCodeHook` parameter operasi. CreateIntent

Jika Anda menggunakan fungsi Lambda dengan maksud fallback, Anda dapat menggunakan fungsi ini untuk memanggil maksud lain atau untuk melakukan beberapa bentuk komunikasi dengan pengguna, seperti mengumpulkan nomor panggilan balik atau membuka sesi dengan perwakilan layanan pelanggan.

Maksud fallback dapat dipanggil beberapa kali dalam sesi yang sama. Misalnya, fungsi Lambda Anda menggunakan tindakan `ElicitIntent` dialog untuk meminta pengguna untuk maksud yang berbeda. Jika Amazon Lex V2 tidak dapat menyimpulkan maksud pengguna setelah jumlah percobaan yang dikonfigurasi, Amazon Lex V2 akan memanggil maksud fallback lagi. Ini juga memanggil maksud fallback ketika pengguna tidak merespons dengan nilai slot yang valid setelah jumlah percobaan yang dikonfigurasi.

Anda dapat mengonfigurasi fungsi Lambda untuk melacak berapa kali intent fallback dipanggil menggunakan variabel sesi. Fungsi Lambda Anda dapat mengambil tindakan yang berbeda jika dipanggil lebih banyak dari ambang batas yang Anda tetapkan dalam fungsi Lambda Anda. Untuk informasi selengkapnya tentang variabel sesi, lihatMengatur atribut sesi untuk bot Lex V2 Anda.

AMAZON.HelpIntent

Menanggapi kata atau frasa yang menunjukkan pengguna membutuhkan bantuan saat berinteraksi dengan bot Anda. Ketika intent ini dipanggil, Anda dapat mengonfigurasi fungsi atau aplikasi Lambda Anda untuk memberikan informasi tentang kemampuan bot Anda, mengajukan pertanyaan lanjutan tentang area bantuan, atau menyerahkan interaksi ke agen manusia.

Ucapan umum:

- bantuan
- tolong aku
- dapatkah Anda membantu saya

AMAZON.KendraSearchIntent

Untuk mencari dokumen yang telah Anda indeks dengan Amazon Kendra, gunakan intent. `AMAZON.KendraSearchIntent` Jika Amazon Lex V2 tidak dapat menentukan tindakan selanjutnya dalam percakapan dengan pengguna, Amazon Lex V2 akan memicu maksud pencarian.

`AMAZON.KendraSearchIntent`Ini hanya tersedia di wilayah Inggris (AS) (en-AS) dan di Wilayah AS Timur (Virginia N.), AS Barat (Oregon) dan Eropa (Irlandia).

Amazon Kendra adalah layanan machine-learning-based pencarian yang mengindeks dokumen bahasa alami seperti dokumen PDF atau file Microsoft Word. Ini dapat mencari dokumen yang diindeks dan mengembalikan jenis tanggapan berikut ke pertanyaan:

- Jawaban
- Entri dari FAQ yang mungkin menjawab pertanyaan
- Dokumen yang terkait dengan pertanyaan

Untuk contoh menggunakan`AMAZON.KendraSearchIntent`, lihatContoh: Membuat Bot FAQ untuk Indeks Amazon Kendra.

Jika Anda mengonfigurasi `AMAZON.KendraSearchIntent` intent untuk bot Anda, Amazon Lex V2 akan memanggil intent kapan pun intent tidak dapat menentukan ucapan pengguna untuk maksud. Jika tidak ada tanggapan dari Amazon Kendra, percakapan berlanjut seperti yang dikonfigurasi di bot.

> **Note**
>
> Amazon Lex V2 saat ini tidak mendukung elisitasi slot `AMAZON.KendraSearchIntent` selama. Jika Amazon Lex V2 tidak dapat menentukan ucapan pengguna untuk slot, itu memanggil file. `AMAZON.FallbackIntent`

Saat Anda menggunakan `AMAZON.KendraSearchIntent` with `AMAZON.FallbackIntent` di bot yang sama, Amazon Lex V2 menggunakan intent sebagai berikut:

1. Amazon Lex V2 menyebut`AMAZON.KendraSearchIntent`. Maksudnya menyebut operasi Amazon `Query` Kendra.
2. Jika Amazon Kendra mengembalikan respons, Amazon Lex V2 menampilkan hasilnya kepada pengguna.

3. Jika tidak ada tanggapan dari Amazon Kendra, Amazon Lex V2 meminta kembali pengguna. Tindakan selanjutnya tergantung pada respons dari pengguna.
   - Jika respons dari pengguna berisi ucapan yang dikenali Amazon Lex V2, seperti mengisi nilai slot atau mengonfirmasi maksud, percakapan dengan pengguna dilanjutkan seperti yang dikonfigurasi untuk bot.
   - Jika respons dari pengguna tidak berisi ucapan yang dikenali Amazon Lex V2, Amazon Lex V2 akan melakukan panggilan lain ke operasi tersebut. `Query`
4. Jika tidak ada respons setelah jumlah percobaan ulang yang dikonfigurasi, Amazon Lex V2 memanggil `AMAZON.FallbackIntent` dan mengakhiri percakapan dengan pengguna.

Ada tiga cara untuk menggunakan `AMAZON.KendraSearchIntent` untuk membuat permintaan ke Amazon Kendra:

- Biarkan maksud pencarian membuat permintaan untuk Anda. Amazon Lex V2 menyebut Amazon Kendra dengan ucapan pengguna sebagai string pencarian. Saat membuat intent, Anda dapat menentukan string filter kueri yang membatasi jumlah respons yang dikembalikan Amazon Kendra. Amazon Lex V2 menggunakan filter dalam permintaan kueri.
- Tambahkan parameter kueri tambahan ke permintaan untuk mempersempit hasil pencarian menggunakan fungsi Lambda Anda. Anda menambahkan `kendraQueryFilterString` bidang yang berisi parameter kueri Amazon Kendra ke tindakan `delegate` dialog. Saat Anda menambahkan parameter kueri ke permintaan dengan fungsi Lambda, parameter tersebut lebih diutamakan daripada filter kueri yang Anda tentukan saat Anda membuat maksud.
- Buat kueri baru menggunakan fungsi Lambda. Anda dapat membuat permintaan kueri Amazon Kendra lengkap yang dikirimkan Amazon Lex V2. Anda menentukan kueri di `kendraQueryRequestPayload` bidang dalam tindakan `delegate` dialog. `kendraQueryRequestPayload`Lapangan lebih diutamakan di atas lapangan. `kendraQueryFilterString`

Untuk menentukan `queryFilterString` parameter saat Anda membuat bot, atau untuk menentukan `kendraQueryFilterString` bidang saat Anda memanggil `delegate` tindakan dalam dialog fungsi Lambda, Anda menentukan string yang digunakan sebagai filter atribut untuk kueri Amazon Kendra. Jika string bukan filter atribut yang valid, Anda akan mendapatkan `InvalidBotConfigException` pengecualian saat runtime. Untuk informasi selengkapnya tentang filter atribut, lihat Menggunakan atribut dokumen untuk memfilter kueri di Panduan Pengembang Amazon Kendra.

Untuk memiliki kontrol atas kueri yang dikirimkan Amazon Lex V2 ke Amazon Kendra, Anda dapat menentukan kueri di `kendraQueryRequestPayload` bidang di fungsi Lambda Anda. Jika kueri tidak valid, Amazon Lex V2 mengembalikan `InvalidLambdaResponseException` pengecualian. Untuk informasi selengkapnya, lihat Operasi kueri di Panduan Pengembang Amazon Kendra.

Untuk contoh cara menggunakan`AMAZON.KendraSearchIntent`, lihatContoh: Membuat Bot FAQ untuk Indeks Amazon Kendra.

Kebijakan IAM untuk Pencarian Amazon Kendra

Untuk menggunakan `AMAZON.KendraSearchIntent` intent, Anda harus menggunakan peran yang menyediakan kebijakan AWS Identity and Access Management (IAM) yang memungkinkan Amazon Lex V2 untuk mengambil peran runtime yang memiliki izin untuk memanggil maksud Amazon Kendra. `Query` Pengaturan IAM yang Anda gunakan bergantung pada apakah Anda membuat `AMAZON.KendraSearchIntent` menggunakan konsol Amazon Lex V2, atau menggunakan AWS SDK atau (AWS Command Line Interface AWS CLI). Saat menggunakan konsol, Anda dapat memilih antara menambahkan izin untuk memanggil Amazon Kendra ke peran terkait layanan Amazon Lex V2 atau menggunakan peran khusus untuk memanggil operasi Amazon Kendra. `Query` Saat Anda menggunakan AWS CLI atau SDK untuk membuat intent, Anda harus menggunakan peran khusus untuk memanggil operasi. `Query`

Melampirkan Izin

Anda dapat menggunakan konsol untuk melampirkan izin untuk mengakses operasi Amazon `Query` Kendra ke peran default Amazon Lex V2 terkait layanan. Saat melampirkan izin ke peran terkait layanan, Anda tidak perlu membuat dan mengelola peran runtime secara khusus untuk terhubung ke indeks Amazon Kendra.

Pengguna, peran, atau grup yang Anda gunakan untuk mengakses konsol Amazon Lex V2 harus memiliki izin untuk mengelola kebijakan peran. Lampirkan kebijakan IAM berikut ke peran akses konsol. Saat Anda memberikan izin ini, peran tersebut memiliki izin untuk mengubah kebijakan peran terkait layanan yang ada.

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": "*"
    }
]
}
```

Menentukan Peran

Anda dapat menggunakan konsol, AWS CLI, atau API untuk menentukan peran runtime yang akan digunakan saat memanggil operasi Amazon Kendra. `Query`

Pengguna, peran, atau grup yang Anda gunakan untuk menentukan peran runtime harus memiliki `iam:PassRole` izin. Kebijakan berikut mendefinisikan izin. Anda dapat menggunakan kunci konteks `iam:AssociatedResourceArn` dan `iam:PassedToService` kondisi untuk membatasi cakupan izin lebih lanjut. Untuk informasi selengkapnya, lihat IAM dan AWS STS Condition Context Keys di Panduan Pengguna AWS Identity and Access Management.

JSON

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

Peran runtime yang perlu digunakan Amazon Lex V2 untuk memanggil Amazon Kendra harus memiliki `kendra:Query` izin. Saat Anda menggunakan peran IAM yang ada untuk izin memanggil operasi Amazon `Query` Kendra, peran tersebut harus memiliki kebijakan berikut yang dilampirkan.

Anda dapat menggunakan konsol IAM, IAM API, atau AWS CLI untuk membuat kebijakan dan melampirkannya ke peran. Petunjuk ini menggunakan AWS CLI untuk membuat peran dan kebijakan.

> **Note**
>
> Kode berikut diformat untuk Linux dan macOS. Untuk Windows, ganti karakter kelanjutan baris Linux (\) dengan tanda sisipan (^).

Untuk menambahkan izin operasi Kueri ke peran

1. Buat dokumen yang disebut **`KendraQueryPolicy.json`** di direktori saat ini, tambahkan kode berikut ke dalamnya, dan simpan
2. Di AWS CLI, jalankan perintah berikut untuk membuat kebijakan IAM untuk menjalankan operasi Amazon Kendra. `Query`

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

3. Lampirkan kebijakan ke peran IAM yang Anda gunakan untuk memanggil `Query` operasi.

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

Anda dapat memilih untuk memperbarui peran terkait layanan Amazon Lex V2 atau menggunakan peran yang Anda buat saat membuat `AMAZON.KendraSearchIntent` untuk bot Anda. Prosedur berikut menunjukkan bagaimana memilih peran IAM untuk digunakan.

Untuk menentukan peran runtime untuk AMAZON.KendraSearchIntent

1. Masuk ke AWS Management Console dan buka konsol Amazon Lex di https://console.aws.amazon.com/lex/.
2. Pilih bot yang ingin Anda tambahkan. `AMAZON.KendraSearchIntent`

3. Pilih plus (+) di sebelah Intent.
4. Di Tambah maksud, pilih Cari maksud yang ada.
5. Di Maksud pencarian, masukkan **`AMAZON.KendraSearchIntent`** lalu pilih Tambah.
6. Di Salin maksud bawaan, masukkan nama untuk maksud, seperti**`KendraSearchIntent`**, lalu pilih Tambah.
7. Buka bagian kueri Amazon Kendra.
8. Untuk peran IAM pilih salah satu opsi berikut:

   - Untuk memperbarui peran terkait layanan Amazon Lex V2 untuk mengaktifkan bot Anda menanyakan indeks Amazon Kendra, pilih Tambahkan izin Amazon Kendra.
   - Untuk menggunakan peran yang memiliki izin untuk memanggil `Query` operasi Amazon Kendra, pilih Gunakan peran yang ada.

Menggunakan Atribut Permintaan dan Sesi sebagai Filter

Untuk memfilter respons dari Amazon Kendra ke item yang terkait dengan percakapan saat ini, gunakan atribut sesi dan permintaan sebagai filter dengan menambahkan `queryFilterString` parameter saat Anda membuat bot. Anda menentukan placeholder untuk atribut saat membuat intent, lalu Amazon Lex V2 mengganti nilai sebelum memanggil Amazon Kendra. Untuk informasi selengkapnya tentang atribut permintaan, lihatMengatur atribut permintaan untuk bot Lex V2 Anda. Untuk informasi selengkapnya tentang atribut sesi, lihatMengatur atribut sesi untuk bot Lex V2 Anda.

Berikut ini adalah contoh `queryFilterString` parameter yang menggunakan string untuk memfilter kueri Amazon Kendra.

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

Berikut ini adalah contoh `queryFilterString` parameter yang menggunakan atribut sesi dipanggil `"SourceURI"` untuk memfilter kueri Amazon Kendra.

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

Berikut ini adalah contoh `queryFilterString` parameter yang menggunakan atribut permintaan dipanggil `"DepartmentName"` untuk memfilter kueri Amazon Kendra.

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

`AMAZON.KendraSearchIntent`gFilter menggunakan format yang sama dengan filter pencarian Amazon Kendra. Untuk informasi selengkapnya, lihat Menggunakan atribut dokumen untuk memfilter hasil penelusuran di panduan pengembang Amazon Kendra.

String filter kueri yang digunakan dengan huruf kecil `AMAZON.KendraSearchIntent` harus menggunakan huruf kecil untuk huruf pertama dari setiap filter. Misalnya, berikut ini adalah filter kueri yang valid untuk file`AMAZON.KendraSearchIntent`.

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

Menggunakan Respon Pencarian

Amazon Kendra mengembalikan respons ke pencarian sebagai tanggapan dari pernyataan niat. `IntentClosingSetting` Maksud harus memiliki `closingResponse` pernyataan kecuali fungsi Lambda menghasilkan pesan respons penutup.

Amazon Kendra memiliki lima jenis tanggapan.

- Dua tanggapan berikut memerlukan FAQ yang akan disiapkan untuk indeks Amazon Kendra Anda. Untuk detail selengkapnya, lihat Menambahkan pertanyaan dan jawaban langsung ke indeks.
  - `x-amz-lex:kendra-search-response-question_answer-question-<N>`— Pertanyaan dari FAQ yang cocok dengan pencarian.

  - `x-amz-lex:kendra-search-response-question_answer-answer-<N>`— Jawaban dari FAQ yang cocok dengan pencarian.
- Tiga tanggapan berikut memerlukan sumber data yang akan disiapkan untuk indeks Amazon Kendra Anda. Untuk detail selengkapnya, lihat Membuat sumber data.
  - `x-amz-lex:kendra-search-response-document-<N>`— Kutipan dari dokumen dalam indeks yang terkait dengan teks ucapan.
  - `x-amz-lex:kendra-search-response-document-link-<N>`— URL dokumen dalam indeks yang terkait dengan teks ucapan.
  - `x-amz-lex:kendra-search-response-answer-<N>`— Kutipan dari dokumen dalam indeks yang menjawab pertanyaan.

Tanggapan dikembalikan dalam `request` atribut. Ada hingga lima tanggapan untuk setiap atribut, bernomor 1 hingga 5. Untuk informasi selengkapnya tentang tanggapan, lihat Jenis respons di Panduan Pengembang Amazon Kendra.

`closingResponse`Pernyataan harus memiliki satu atau lebih grup pesan. Setiap grup pesan berisi satu atau beberapa pesan. Setiap pesan dapat berisi satu atau beberapa variabel placeholder yang diganti dengan atribut permintaan dalam respons dari Amazon Kendra. Harus ada setidaknya satu pesan dalam grup pesan di mana semua variabel dalam pesan diganti dengan nilai atribut permintaan dalam respons runtime, atau harus ada pesan dalam grup tanpa variabel placeholder. Atribut permintaan diatur dengan tanda kurung ganda (“((” “))”). Pesan grup pesan berikut cocok dengan respons apa pun dari Amazon Kendra:

- “Saya menemukan pertanyaan FAQ untuk Anda: ((x-amz-lex: kendra-search-response-question _jawaban-pertanyaan-1)), dan jawabannya adalah ((: _jawaban-jawab-1))” x-amz-lex kendra-search-response-question
- “Saya menemukan kutipan dari dokumen yang bermanfaat: ((x-amz-lex: kendra-search-response-document -1))”
- “Saya pikir jawaban atas pertanyaan Anda adalah ((x-amz-lex: kendra-search-response-answer -1))”

Menggunakan Fungsi Lambda untuk Mengelola Permintaan dan Respons

`AMAZON.KendraSearchIntent`Maksud dapat menggunakan hook kode dialog dan hook kode pemenuhan untuk mengelola permintaan ke Amazon Kendra dan responsnya. Gunakan fungsi

Lambda kait kode dialog saat Anda ingin memodifikasi kueri yang Anda kirim ke Amazon Kendra, dan kode pemenuhan menghubungkan fungsi Lambda saat Anda ingin memodifikasi respons.

Membuat Query dengan Hook Kode Dialog

Anda dapat menggunakan hook kode dialog untuk membuat kueri untuk dikirim ke Amazon Kendra. Menggunakan hook kode dialog adalah opsional. Jika Anda tidak menentukan hook kode dialog, Amazon Lex V2 akan membuat kueri dari ucapan pengguna dan menggunakan `queryFilterString` yang Anda berikan saat mengonfigurasi intent, jika Anda memberikannya.

Anda dapat menggunakan dua bidang dalam respons kait kode dialog untuk memodifikasi permintaan ke Amazon Kendra:

- `kendraQueryFilterString`— Gunakan string ini untuk menentukan filter atribut untuk permintaan Amazon Kendra. Anda dapat memfilter kueri menggunakan salah satu bidang indeks yang ditentukan dalam indeks Anda. Untuk struktur string filter, lihat Menggunakan atribut dokumen untuk memfilter kueri di Panduan Pengembang Amazon Kendra. Jika string filter yang ditentukan tidak valid, Anda akan mendapatkan `InvalidLambdaResponseException` pengecualian. `kendraQueryFilterString`String mengesampingkan string kueri apa pun yang ditentukan dalam `queryFilterString` konfigurasi untuk maksud.
- `kendraQueryRequestPayload`— Gunakan string ini untuk menentukan kueri Amazon Kendra. Kueri Anda dapat menggunakan salah satu fitur Amazon Kendra. Jika Anda tidak menentukan kueri yang valid, Anda mendapatkan `InvalidLambdaResponseException` pengecualian. Untuk informasi selengkapnya, lihat Kueri di Panduan Pengembang Amazon Kendra.

Setelah Anda membuat filter atau string kueri, Anda mengirim respons ke Amazon Lex V2 dengan `dialogAction` bidang respons yang disetel ke`delegate`. Amazon Lex V2 mengirimkan kueri ke Amazon Kendra dan kemudian mengembalikan respons kueri ke hook kode pemenuhan.

Menggunakan Hook Kode Pemenuhan untuk Respons

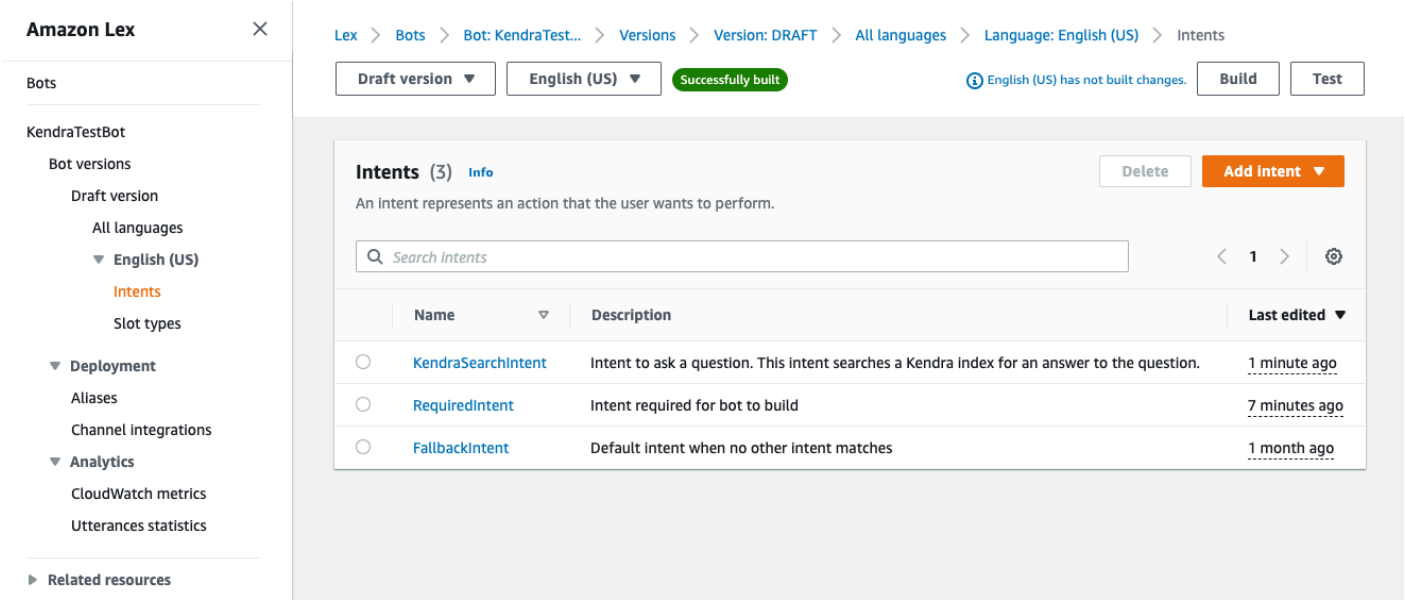
Setelah Amazon Lex V2 mengirimkan kueri ke Amazon Kendra, respons kueri dikembalikan ke fungsi Lambda `AMAZON.KendraSearchIntent` pemenuhan. Acara input ke kait kode berisi respons lengkap dari Amazon Kendra. Data kueri berada dalam struktur yang sama dengan yang dikembalikan oleh operasi Amazon Kendra`Query`. Untuk informasi selengkapnya, lihat Sintaks respons kueri di Panduan Pengembang Amazon Kendra.

Kait kode pemenuhan adalah opsional. Jika tidak ada, atau jika kait kode tidak mengembalikan pesan dalam respons, Amazon Lex V2 menggunakan `closingResponse` pernyataan untuk tanggapan.

Contoh: Membuat Bot FAQ untuk Indeks Amazon Kendra

Contoh ini membuat bot Amazon Lex V2 yang menggunakan indeks Amazon Kendra untuk memberikan jawaban atas pertanyaan pengguna. Bot FAQ mengelola dialog untuk pengguna. Ini menggunakan `AMAZON.KendraSearchIntent` maksud untuk menanyakan indeks dan menyajikan respons kepada pengguna. Berikut adalah ringkasan tentang bagaimana Anda akan membuat bot FAQ Anda menggunakan indeks Amazon Kendra:

1. Buat bot yang akan berinteraksi dengan pelanggan Anda untuk mendapatkan jawaban dari bot Anda.
2. Buat maksud khusus. Karena `AMAZON.KendraSearchIntent` dan `AMAZON.FallbackIntent` merupakan maksud cadangan, bot Anda memerlukan setidaknya satu maksud lain yang harus berisi setidaknya satu ucapan. Maksud ini memungkinkan bot Anda untuk membangun, tetapi tidak digunakan sebaliknya. Bot FAQ Anda akan berisi setidaknya tiga maksud, seperti pada gambar di bawah ini:



3. Tambahkan `AMAZON.KendraSearchIntent` intent ke bot Anda dan konfigurasikan agar berfungsi dengan indeks Amazon Kendra Anda.
4. Uji bot dengan membuat kueri dan memverifikasi bahwa hasil dari indeks Amazon Kendra Anda adalah dokumen yang menjawab kueri.

Prasyarat

Sebelum Anda dapat menggunakan contoh ini, Anda perlu membuat indeks Amazon Kendra. Untuk informasi selengkapnya, lihat Memulai konsol Amazon Kendra di Panduan Pengembang Amazon

Kendra. Untuk contoh ini, pilih kumpulan data sampel (Contoh dokumentasi AWS) sebagai sumber data Anda.

Untuk membuat bot FAQ:

1. Masuk ke AWS Management Console dan buka konsol Amazon Lex di https://console.aws.amazon.com/lex/.
2. Di panel navigasi, pilih Bots.
3. Pilih Buat bot.
    a. Untuk metode Creation, pilih Create a blank bot.
    b. Di bagian konfigurasi Bot, beri bot nama yang menunjukkan tujuannya, seperti**KendraTestBot**, dan deskripsi opsional. Nama harus unik di akun Anda.
    c. Di bagian Izin IAM, pilih Buat peran dengan izin Amazon Lex dasar. Ini akan membuat peran AWS Identity and Access Management (IAM) dengan izin yang dibutuhkan Amazon Lex V2 untuk menjalankan bot Anda.
    d. Di bagian Children's Online Privacy Protection Act (COPPA), pilih No.
    e. Di bagian batas waktu sesi Idle dan Pengaturan lanjutan, tinggalkan pengaturan default dan pilih Berikutnya.
    f. Sekarang Anda berada di bagian Tambahkan bahasa ke bot. Di menu di bawah Interaksi suara, pilih Tidak Ada. Ini hanya aplikasi berbasis teks. Tinggalkan pengaturan default untuk bidang yang tersisa.
    g. Pilih Selesai. Amazon Lex V2 membuat bot Anda dan maksud default yang disebut NewIntent, dan membawa Anda ke halaman untuk mengonfigurasi maksud ini

Agar berhasil membangun bot, Anda harus membuat setidaknya satu maksud yang terpisah dari `AMAZON.FallbackIntent` dan. `AMAZON.KendraSearchIntent` Maksud ini diperlukan untuk membangun bot Amazon Lex V2 Anda, tetapi tidak digunakan untuk respons FAQ. Maksud ini harus mengandung setidaknya satu contoh ucapan dan ucapan tersebut tidak boleh berlaku untuk pertanyaan apa pun yang diajukan pelanggan Anda.

Untuk membuat maksud yang diperlukan:

1. Di bagian Detail maksud, beri nama maksud, seperti. **`RequiredIntent`**
2. Di bagian Sample ujaran, ketikkan ucapan di kotak di sebelah Tambahkan ucapan, seperti. **`Required utterance`** Kemudian pilih Tambahkan ucapan.

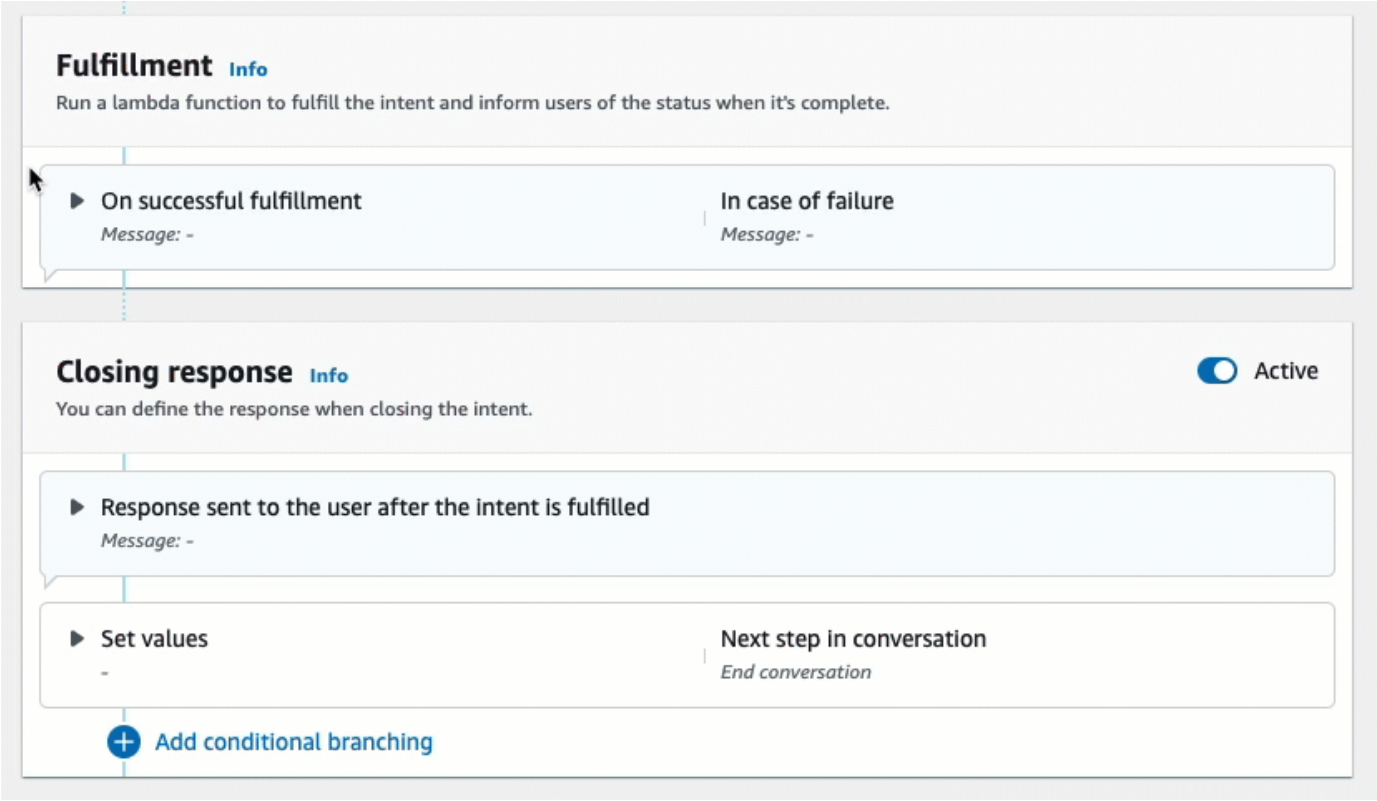
3. Pilih Simpan maksud.

Buat maksud untuk mencari indeks Amazon Kendra dan pesan respons yang harus dikembalikan.

Untuk membuat AMAZON. KendraSearchIntent pesan maksud dan respons:

1. Pilih Kembali ke daftar maksud di panel navigasi untuk kembali ke halaman Intent untuk bot Anda. Pilih Tambah maksud dan pilih Gunakan maksud bawaan dari menu tarik-turun.
2. Di kotak yang muncul, pilih menu di bawah Maksud bawaan. Masukkan **`AMAZON.KendraSearchIntent`** di bilah pencarian dan kemudian pilih dari daftar.
3. Berikan maksud nama, seperti**`KendraSearchIntent`**.
4. Dari menu tarik-turun indeks Amazon Kendra, pilih indeks yang ingin Anda cari. Indeks yang Anda buat di bagian Prasyarat harus tersedia.
5. Pilih Tambahkan.
6. Di editor maksud, gulir ke bawah ke bagian Pemenuhan, pilih panah kanan untuk memperluas bagian, dan tambahkan pesan berikut di kotak di bawah Pada pemenuhan yang berhasil:

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



Untuk informasi selengkapnya tentang Respons Penelusuran Amazon Kendra, lihat Menggunakan Respons Penelusuran.

7. Pilih Simpan maksud, lalu pilih Build untuk membangun bot. Saat bot sudah siap, spanduk di bagian atas layar berubah menjadi hijau dan menampilkan pesan sukses.

Terakhir, gunakan jendela pengujian konsol untuk menguji respons dari bot Anda.

Untuk menguji bot FAQ Anda:

1. Setelah bot berhasil dibangun, pilih Uji.
2. Masukkan **`What is Amazon Kendra?`** di jendela uji konsol. Verifikasi bahwa bot merespons dengan tautan.
3. Untuk informasi selengkapnya tentang mengonfigurasi`AMAZON.KendraSearchIntent`, lihat AMAZON.KendraSearchIntentdan KendraConfiguration.

AMAZON.PauseIntent

Menanggapi kata dan frasa yang memungkinkan pengguna untuk menjeda interaksi dengan bot sehingga mereka dapat kembali ke sana nanti. Fungsi atau aplikasi Lambda Anda perlu menyimpan data maksud dalam variabel sesi, atau Anda perlu menggunakan GetSessionoperasi untuk mengambil data maksud saat melanjutkan maksud saat ini.

Ucapan umum:

- jeda
- jeda itu

AMAZON.QnAIntent

> **Note**
>
> Sebelum Anda dapat memanfaatkan fitur AI generatif, Anda harus memenuhi prasyarat berikut
>
> 1. Untuk informasi tentang harga untuk menggunakan Amazon Bedrock, lihat harga Amazon Bedrock.
> 2. Aktifkan kemampuan AI generatif untuk lokal bot Anda. Untuk melakukannya, ikuti langkah-langkah diOptimalkan pembuatan dan kinerja bot Lex V2 dengan menggunakan AI generatif.

Menanggapi pertanyaan pelanggan dengan menggunakan Amazon Bedrock FM untuk mencari dan meringkas tanggapan FAQ. Maksud ini diaktifkan ketika ucapan tidak diklasifikasikan ke dalam maksud lain yang ada di bot. Perhatikan bahwa maksud ini tidak akan diaktifkan untuk ucapan yang terlewat saat memunculkan nilai slot. Setelah dikenali,`AMAZON.QnAIntent`, menggunakan model Amazon Bedrock yang ditentukan untuk mencari Basis Pengetahuan Amazon Bedrock yang dikonfigurasi dan menanggapi pertanyaan pelanggan.

> **Warning**
>
> Anda tidak dapat menggunakan `AMAZON.QnAIntent` dan `AMAZON.KendraSearchIntent` di lokal bot yang sama.

Opsi toko pengetahuan berikut tersedia. Anda harus sudah membuat toko pengetahuan dan mengindeks dokumen di dalamnya.

- OpenSearch Domain layanan - Berisi dokumen yang diindeks. Untuk membuat domain, ikuti langkah-langkah di Membuat dan mengelola domain OpenSearch Layanan Amazon.
- Indeks Amazon Kendra - Berisi dokumen FAQ yang diindeks. Untuk membuat indeks Amazon Kendra, ikuti langkah-langkah di Membuat indeks.
- Basis Pengetahuan Amazon Bedrock - Berisi sumber data yang diindeks. Untuk menyiapkan Basis Pengetahuan Batuan Dasar Amazon, ikuti langkah-langkah di Membangun Basis Pengetahuan.

Jika Anda memilih maksud ini, Anda mengonfigurasi bidang berikut dan kemudian pilih Tambah untuk menambahkan maksud.

- Model batuan dasar — Pilih penyedia dan model pondasi yang akan digunakan untuk maksud ini. Pastikan untuk memeriksa model terbaru yang tersedia dan jadwal penghentian serta merencanakan migrasi yang sesuai. Untuk informasi selengkapnya, lihat Siklus hidup model.
- Toko pengetahuan — Pilih sumber dari mana Anda ingin model menarik informasi dari untuk menjawab pertanyaan pelanggan. Sumber-sumber berikut tersedia.
  - OpenSearch— Konfigurasikan bidang berikut.
    - Endpoint domain — Menyediakan endpoint domain yang Anda buat untuk domain atau yang diberikan kepada Anda setelah pembuatan domain.
    - Nama indeks — Berikan indeks untuk mencari. Untuk informasi selengkapnya, lihat Mengindeks data di OpenSearch Layanan Amazon.
    - Pilih bagaimana Anda ingin mengembalikan respons kepada pelanggan.
      - Respons yang tepat - Ketika opsi ini diaktifkan, nilai di bidang Jawaban digunakan sebagaimana adanya untuk respons bot. Model dasar Amazon Bedrock yang dikonfigurasi digunakan untuk memilih konten jawaban yang tepat apa adanya, tanpa sintesis atau ringkasan konten apa pun. Tentukan nama bidang tanya jawab yang dikonfigurasi dalam OpenSearch database.
      - Sertakan bidang - Mengembalikan jawaban yang dihasilkan oleh model menggunakan bidang yang Anda tentukan. Tentukan nama hingga lima bidang yang dikonfigurasi dalam OpenSearch database. Gunakan titik koma (;) untuk memisahkan bidang.
  - Amazon Kendra - Konfigurasikan bidang berikut.
    - Indeks Amazon Kendra - Pilih indeks Amazon Kendra yang Anda ingin bot Anda cari.

- Filter Amazon Kendra — Untuk membuat filter, pilih kotak centang ini. Untuk informasi selengkapnya tentang format JSON filter penelusuran Amazon Kendra, lihat Menggunakan atribut dokumen untuk memfilter hasil penelusuran.
- Respons yang tepat - Untuk membiarkan bot Anda mengembalikan respons persis yang dikembalikan oleh Amazon Kendra, pilih kotak centang ini. Jika tidak, model Amazon Bedrock yang Anda pilih menghasilkan respons berdasarkan hasil.

> **Note**
>
> Untuk menggunakan fitur ini, Anda harus terlebih dahulu menambahkan pertanyaan FAQ ke indeks Anda dengan mengikuti langkah-langkah di Menambahkan pertanyaan yang sering diajukan (FAQs) ke indeks.

- Basis Pengetahuan Amazon Bedrock - Jika Anda memilih opsi ini, tentukan ID Basis Pengetahuan Amazon Bedrock. Anda dapat menemukan ID dengan memeriksa halaman detail Pangkalan Pengetahuan Amazon Bedrock di konsol, atau dengan mengirimkan GetKnowledgeBasepermintaan.
  - Respons yang tepat - Ketika opsi ini diaktifkan, nilai di bidang Jawaban digunakan sebagaimana adanya untuk respons bot. Model dasar Amazon Bedrock yang dikonfigurasi digunakan untuk memilih konten jawaban yang tepat apa adanya, tanpa sintesis atau ringkasan konten apa pun. Untuk menggunakan respons yang tepat untuk Pangkalan Pengetahuan Amazon Bedrock, Anda perlu melakukan hal berikut:
    - Buat file JSON individual dengan setiap file yang berisi bidang jawaban yang berisi respons yang tepat yang perlu dikembalikan ke pengguna akhir.
    - Saat mengindeks dokumen-dokumen ini di Basis Pengetahuan Batuan Dasar, pilih Strategi Chunking sebagai No Chunking..
    - Tentukan bidang jawaban di Amazon Lex V2, sebagai bidang Jawaban di Pangkalan Pengetahuan Batuan Dasar.

Tanggapan dari Qn AIntent akan disimpan ke dalam atribut permintaan seperti yang ditunjukkan di bawah ini:

- `x-amz-lex:qnA-search-response`— Tanggapan dari Qn AIntent terhadap pertanyaan atau ucapan.
- `x-amz-lex:qnA-search-response-source`— Menunjuk ke dokumen atau daftar dokumen yang digunakan untuk menghasilkan respons.

- `x-amz-lex:qna-additional-context`— Konteks tambahan yang digunakan oleh Qn AIntent untuk menghasilkan respons.

Konfigurasi model tambahan

Saat Amazon.qn AIntent dipanggil, ia menggunakan templat prompt default yang menggabungkan instruksi dan konteks dengan kueri pengguna untuk membuat prompt yang dikirim ke model untuk pembuatan respons. Anda juga dapat memberikan prompt khusus atau memperbarui prompt default agar sesuai dengan kebutuhan Anda.

Anda dapat merekayasa template prompt dengan alat-alat berikut:

Placeholder prompt — Variabel yang telah ditentukan sebelumnya di Amazon.qn AIntent untuk Amazon Bedrock yang diisi secara dinamis saat runtime selama panggilan batuan dasar. Dalam prompt sistem, Anda dapat melihat placeholder ini dikelilingi oleh simbol. $ Daftar berikut menjelaskan placeholder yang dapat Anda gunakan:

Variabel	Digantikan oleh	Model	Diperlukan?
$query_hasil$	Hasil yang diambil untuk kueri pengguna dari Knowledge Store	Model Batuan Dasar Terpilih	Ya
$output_instruksi$	Instruksi yang mendasari untuk memformat generasi respons dan kutipan. Berbeda dengan model. Jika Anda menentukan instruksi pemformatan Anda sendiri, kami sarankan Anda menghapus placehold er ini.	Model Batuan Dasar Terpilih	Tidak
$additional_context$	Konteks tambahan yang digunakan	Model Batuan Dasar Terpilih	Tidak

Variabel	Digantikan oleh	Model	Diperlukan?
	oleh Qn AIntent untuk menghasilkan respons		
$locale$	Bahasa di mana bot akan menjawab pertanyaan pelanggan	Model Batuan Dasar Terpilih	Tidak

Prompt default yang digunakan adalah:

```
$query_results$

$additional_context$

Please only follow the instructions in <instruction> tags below.
<instruction>
Given the conversation history, <additional_context> and <Context>:
(1) first, identify the user query intent and classify it as one of the categories:
 FAQ_QUERY, OTHER_QUERY, GIBBERISH, GREETINGS, AFFIRMATION, CHITCHAT, or MISC;
(2) second, if the intent is FAQ_QUERY, predict the most relevant grounding passage(s)
 by providing the passage id(s) or output CANNOTANSWER;
(3) then, generate a concise, to-the-point FAQ-style response in $locale$ locale
 ONLY USING the grounding content in <Context> and <additional_context>; or output
 CANNOTANSWER if the user query/request cannot be directly answered with the grounding
 content. DO NOT mention about the grounding passages such as ids or other meta data;
 do not create new content not presented in <Context>. Do NOT respond to query that is
 ill-intented or off-topic;
(4) lastly, provide the confidence level of the above prediction as LOW, MID or HIGH.
</instruction>

$output_instruction$
```

$output_instruction$ diganti dengan:

```
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**
>
> Jika Anda memutuskan untuk tidak menggunakan instruksi default, maka output apa pun yang diberikan LLM akan dikembalikan apa adanya kembali ke pengguna akhir.
> Instruksi keluaran harus berisi<text></text> dan<passageId></passageId> tag dan instruksi untuk LLM untuk mengembalikan passageID untuk memberikan respons dan atribusi sumber.

Dukungan konteks tambahan melalui atribut sesi

Anda dapat meneruskan konteks tambahan ke `AMAZON.QnAIntent` saat runtime melalui atribut `x-amz-lex:qna-additional-context` session. Ini memungkinkan Anda untuk memberikan informasi tambahan yang dapat digunakan model di samping hasil penyimpanan pengetahuan saat menghasilkan respons. Konteks tambahan dimasukkan ke dalam template prompt melalui `$additional_context$` placeholder.

Contoh:

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

Dukungan pemfilteran metadata Amazon Bedrock Knowledge Base melalui atribut sesi

Anda dapat meneruskan filter metadata Amazon Bedrock Knowledge Base sebagai bagian dari atribut session. `x-amz-lex:bkb-retrieval-filter`

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

> **Note**
>
> Anda perlu menggunakan Amazon Bedrock Knowledge Base sebagai penyimpanan Data untuk Qn AIntent untuk menggunakan filter ini. Untuk informasi selengkapnya, lihat Metadata filtering

Konfigurasi inferensi

Anda dapat menentukan konfigurasi inferensi yang akan digunakan saat melakukan panggilan ke LLM menggunakan atribut session:

- suhu: tipe Integer
- TopP
- MaxTokens

Contoh:

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

Dukungan Bedrock Guardrails melalui atribut waktu pembuatan dan sesi

- Dengan menggunakan Konsol di Buildtime — Sediakan GuardrailsIdentifier dan. GuardrailsVersion Pelajari lebih lanjut di bagian Konfigurasi Model Tambahan.
- Dengan menggunakan atribut Session — Anda juga dapat menentukan konfigurasi Guardrails menggunakan atribut sesi: `x-amz-lex:bedrock-guardrails-identifier` dan. `x-amz-lex:bedrock-guardrails-version`

Untuk informasi lebih lanjut tentang penggunaan Bedrock Guardrails, lihat Guardrails.

AMAZON.QnAIntent (multiple use support)

Anda dapat memilih untuk memiliki beberapa Amazon.qn AIntents dalam satu lokal. Amazon Lex V2 mendukung hingga 5 Amazon.qn AIntents dalam lokal bot.

Amazon.qn AIntent dapat dipicu jika salah satu kasus berikut benar:

- Jika lokal bot hanya berisi 1 Amazon.qn AIntent dan maksud itu tidak mengandung contoh ucapan, maka itu diaktifkan ketika ucapan tidak diklasifikasikan ke dalam maksud lain yang ada di bot. Maksud ini diaktifkan ketika ucapan tidak diklasifikasikan ke dalam maksud lain yang ada di bot. Perhatikan bahwa maksud ini tidak akan diaktifkan untuk ucapan yang terlewat saat memunculkan nilai slot.

  > **Note**
  >
  > Jika respons dari FM tidak memuaskan atau panggilan ke FM gagal, Amazon Lex V2 kemudian memanggil. `AMAZON.FallbackIntent`

- Jika Amazon.qn AIntent memang berisi contoh ucapan, maka itu hanya diaktifkan ketika Amazon Lex V2 mengenali bahwa pengguna ingin memulai maksud tersebut berdasarkan input pengguna.

  > **Note**
  >
  > Jika respons dari FM tidak memuaskan atau panggilan ke FM gagal, Amazon Lex V2 memanggil kegagalan langkah berikutnya, yang ditentukan dalam blok pemenuhan.

> **Note**
>
> Jika `botLocale` memiliki lebih dari 1 Amazon.qnAIntent, maka setiap Amazon.qn AIntent harus memiliki setidaknya 1 contoh ucapan.

AMAZON.QinConnectIntent

> **Note**
>
> Untuk menggunakan kemampuan AI generatif menggunakan Amazon Q In Connect, Anda harus menyelesaikan prasyarat berikut:
>
> 1. Arahkan ke konsol Amazon Connect dan buat instans Anda, jika Anda belum memilikinya, lihat Memulai Amazon Connect.

2. Aktifkan Amazon Q di Connect untuk instans Anda, lihat Mengaktifkan Amazon Q di Connect untuk instans Anda.

AMAZON. QinConnectIntent menanggapi pertanyaan pelanggan dengan menggunakan evolusi Amazon Connect Wisdom yang disempurnakan LLM yang memberikan rekomendasi waktu nyata untuk membantu pelanggan dan agen pusat kontak menyelesaikan masalah pelanggan dengan cepat dan akurat. Maksud ini diaktifkan ketika ucapan tidak diklasifikasikan ke dalam maksud lain yang ada di bot. Perhatikan bahwa maksud ini tidak akan diaktifkan untuk ucapan yang terlewat saat memunculkan nilai slot. Setelah dikenali, AMAZON. QinConnectIntent, menggunakan Q yang ditentukan di domain Connect untuk mencari Basis Pengetahuan Amazon Bedrock yang dikonfigurasi dan menanggapi pertanyaan pelanggan.

Note

- Anda tidak dapat menggunakan Amazon. QinConnectIntent bersama dengan Amazon.qn AIntent di lokal bot yang sama.
- Jika Anda memilih bahasa lain selain Bahasa Inggris AS, Anda harus menyesuaikan permintaan layanan mandiri (`SELF_SERVICE_PRE_PROCESSING`dan`SELF_SERVICE_ANSWER_GENERATION`) untuk merespons dalam bahasa yang ditentukan. Untuk informasi selengkapnya tentang cara menyesuaikan prompt, lihat Menyesuaikan Amazon Q di Connect.

Jika Anda memilih maksud ini, Anda perlu mengonfigurasi bidang berikut dan kemudian pilih Simpan Maksud untuk menambahkan maksud ke bot.

- Konfigurasi Amazon Q In Connect - Berikan Nama Sumber Daya Amazon (ARN) dari asisten Amazon Q in Connect. Asisten Pola ARN:. `^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}$>`

Tanggapan dari QinConnectIntent akan disimpan ke dalam atribut permintaan seperti yang ditunjukkan di bawah ini:

- `x-amz-lex:q-in-connect-response`— Tanggapan dari QinConnectIntent pertanyaan atau ucapan.

Atribut Sesi Dikembalikan dari QinConnectIntent

Interaksi dengan QinConnect intent menyediakan data tambahan tentang percakapan melalui atribut sesi.

1. `x-amz-lex:q-in-connect:session-arn`— Pengenal unik untuk sesi yang dibuat dengan Amazon Q In Connect selama percakapan.
2. `x-amz-lex:q-in-connect:conversation-status`— Status percakapan saat ini dengan QinConnect asisten atau domain. Ada tiga nilai yang mungkin untuk status ini:
   - `CLOSED`
   - `READY`
   - `PROCESSING`
3. `x-amz-lex:q-in-connect:conversation-status-reason`— Memberikan alasan untuk status saat ini yang dilaporkan dengan atribut di atas. Alasan yang mungkin adalah sebagai berikut:
   - `SUCCESS`— Menunjukkan bahwa pelanggan tidak memiliki apa-apa lagi untuk ditanyakan dan pertanyaan telah berhasil dijawab.
   - `FAILED`— Menunjukkan kegagalan saat menjawab pertanyaan pelanggan. Ini sebagian besar disebabkan oleh kegagalan untuk memahami pertanyaan pelanggan.
   - `REJECTED`— Menunjukkan bahwa asisten menolak untuk menjawab pertanyaan pelanggan, dan merekomendasikan agar pertanyaan ditangani di luar interaksi bot, seperti berbicara dengan orang atau agen, untuk mendapatkan informasi lebih lanjut.

> **Note**
>
> Saat bot dengan QinConnectIntent dipanggil selama interaksi pelanggan yang didorong oleh instans Amazon Connect, sesi Anda harus dibuat dan diteruskan dari instans Amazon Connect. Untuk membuat sesi, Amazon Connect Flows dapat dikonfigurasi dengan Amazon Q di langkah Connect.

Batasan

- Anda tidak dapat menggunakan Amazon. QinConnectIntent bersama dengan maksud tanpa ucapan tertentu seperti Amazon.qn, AMAZONAIntent. BedrockAgentIntent di lokal bot yang sama.
- Saat bot dengan QinConnectIntent dipanggil selama interaksi pelanggan yang didorong oleh instans Amazon Connect, sesi Anda harus dibuat dan diteruskan dari instans Amazon Connect. Untuk membuat sesi, Amazon Connect Flows dapat dikonfigurasi dengan langkah Amazon Q In Connect.
- Tidak boleh ada lebih dari satu AMAZON. QinConnectIntent per bot lokal.
- Domain Amazon Q in Connect digunakan dengan AMAZON. QinConnectIntent harus berada di Wilayah AWS yang sama dengan bot Amazon Lex V2.

Izin

Jika QinConnect Intent digunakan dalam bot Amazon Lex V2, dan bot menggunakan Service Linked Role (SLR), layanan Amazon Lex V2 memiliki izin untuk memperbarui kebijakan yang sesuai tentang peran untuk mengintegrasikannya dengan asisten Q in Connect. Jika bot menggunakan peran IAM khusus, maka pengguna perlu menambahkan izin ini secara manual ke peran IAM mereka.

Peran Tertaut Layanan akan diperbarui dengan izin berikut jika QinConnect maksud ditambahkan. Kebijakan baru akan ditambahkan untuk QinConnect akses:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Sid": "QInConnectAssistantPolicy",
            "Action": [
                "wisdom:CreateSession",
                "wisdom:GetAssistant"
            ],
            "Resource": [
                "arn:aws:wisdom:*:accountId:assistant/assistantId",
                "arn:aws:wisdom:*:accountId:assistant/assistantId/*"
            ]
        },
        {
            "Effect": "Allow",
            "Sid": "QInConnectSessionsPolicy",
```

```
            "Action": [
                "wisdom:SendMessage",
                "wisdom:GetNextMessage"
            ],
            "Resource": [
                "arn:aws:wisdom:*:accountId:session/assistantId/*"
            ]
        },
        {
            "Sid": "QInConnectKmsCMKPolicy",
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt",
                "kms:GenerateDataKey"
            ],
            "Resource": [
                "arn:aws:kms:region:accountId:key/keyId"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:ResourceAccount": "accountId",
                    "kms:ViaService": "wisdom.region.amazonaws.com",
                    "kms:EncryptionContext:aws:wisdom:assistant:arn":
 ["arn:aws:wisdom:region:accountId:assistant/assistantId"]
                }
            }
        }
    ]
}
```

> **Note**
>
> `QInConnectKmsCMKPolicy`Pernyataan ini hanya diperlukan jika Anda menggunakan kunci KMS yang dikelola pelanggan dengan asisten Amazon Q in Connect.

Kebijakan Kepercayaan

```
{
    "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

Menanggapi kata dan frasa yang memungkinkan pengguna untuk mengulangi pesan sebelumnya. Aplikasi Anda perlu menggunakan fungsi Lambda untuk menyimpan informasi intent sebelumnya dalam variabel sesi, atau Anda perlu menggunakan GetSessionoperasi untuk mendapatkan informasi intent sebelumnya.

Ucapan umum:

- ulangi
- katakan itu lagi
- ulangi itu

AMAZON.ResumeIntent

Menanggapi kata dan frasa yang memungkinkan pengguna untuk melanjutkan maksud yang dijeda sebelumnya. Fungsi atau aplikasi Lambda Anda harus mengelola informasi yang diperlukan untuk melanjutkan maksud sebelumnya.

Ucapan umum:

- melanjutkan
- terus
- terus berjalan

AMAZON.StartOverIntent

Menanggapi kata dan frasa yang memungkinkan pengguna untuk berhenti memproses maksud saat ini dan memulai kembali dari awal. Anda dapat menggunakan fungsi Lambda Anda atau `PutSession` operasi untuk mendapatkan nilai slot pertama lagi.

Ucapan umum:

- mulai dari awal
- restart
- mulai lagi

AMAZON.StopIntent

Menanggapi kata dan frasa yang menunjukkan bahwa pengguna ingin berhenti memproses maksud saat ini dan mengakhiri interaksi dengan bot. Fungsi atau aplikasi Lambda Anda harus menghapus atribut dan nilai jenis slot yang ada dan kemudian mengakhiri interaksi.

Ucapan umum:

- berhenti
- off
- diam

Menambahkan jenis slot

Jenis slot menentukan nilai yang dapat diberikan pengguna untuk variabel intent Anda. Anda menentukan jenis slot untuk setiap bahasa sehingga nilainya spesifik untuk bahasa itu. Misalnya, untuk jenis slot yang mencantumkan warna cat, Anda dapat memasukkan nilai "" red dalam bahasa Inggris, "rouge" dalam bahasa Prancis, dan "rojo" dalam bahasa Spanyol.

Topik ini menjelaskan cara membuat jenis slot khusus yang memberikan nilai untuk slot maksud Anda. Anda juga dapat menggunakan tipe slot bawaan untuk nilai standar. Misalnya, Anda dapat menggunakan jenis slot bawaan `AMAZON.Country` untuk daftar negara di dunia.

Untuk membuat jenis slot

1. Masuk ke AWS Management Console dan buka konsol Amazon Lex di https://console.aws.amazon.com/lex/.
2. Dari daftar bot, pilih bot yang ingin Anda tambahkan bahasa, lalu pilih Struktur percakapan dan kemudian Semua bahasa.
3. Pilih bahasa untuk menambahkan jenis slot, lalu pilih jenis Slot.
4. Pilih Tambahkan jenis slot, beri nama jenis slot Anda, lalu pilih Tambah.
5. Di editor jenis slot, tambahkan detail jenis slot Anda.
   - Resolusi nilai slot — Menentukan bagaimana nilai slot diselesaikan. Jika Anda memilih Perluas nilai, Amazon Lex V2 menggunakan nilai sebagai nilai representatif untuk pelatihan. Jika Anda menggunakan Batasi nilai slot, nilai yang diizinkan untuk slot dibatasi untuk yang Anda berikan.
   - Nilai tipe slot — Nilai untuk slot. Jika Anda memilih Batasi nilai slot, Anda dapat menambahkan sinonim untuk nilainya. Misalnya, untuk nilai “sepak bola” Anda dapat menambahkan sinonim “sepak bola.” Jika pengguna memasukkan “sepak bola” dalam percakapan dengan bot Anda, nilai sebenarnya dari slot tersebut adalah “sepak bola.”
   - Gunakan nilai slot sebagai kosakata khusus - Aktifkan opsi ini untuk membantu meningkatkan pengenalan nilai slot dan sinonim dalam percakapan audio. Jangan aktifkan opsi ini ketika nilai slot adalah istilah umum, seperti “ya,” “tidak,” “satu,” “dua,” “tiga,” dll.
6. Pilih Simpan jenis slot.

Amazon Lex V2 menawarkan jenis slot berikut:

Topik

- Built-in jenis slot
- Jenis slot khusus
- Jenis slot tata bahasa
- Jenis slot komposit

Built-in jenis slot

Amazon Lex mendukung tipe slot bawaan yang menentukan bagaimana data dalam slot dikenali dan ditangani. Anda dapat membuat slot jenis ini dalam maksud Anda. Ini menghilangkan kebutuhan

untuk membuat nilai enumerasi untuk data slot yang umum digunakan seperti tanggal, waktu, dan lokasi. Built-in Jenis slot tidak memiliki versi.

Jenis Slot	Deskripsi Singkat	Lokal yang Didukung	
AMAZON.AlphaNumeric	Mengenali kata-kata yang terdiri dari huruf dan angka.		
AMAZON.City	Mengakui kata-kata yang mewakili kota.	Semua lokal	
AMAZON.Confirmation	Mengenali kata-kata yang berarti 'Ya', 'Tidak', 'Mungkin', dan 'Tidak tahu' dan mengubahnya menjadi format standar (Yes/No/Maybe/Dontidak tahu).	Semua lokal	
AMAZON.Country	Mengakui kata-kata yang mewakili suatu negara.	Semua lokal	
AMAZON.Currency	Mengenali kata-kata yang mewakili nilai mata uang dan mengubahnya menjadi singkatan dan nilai mata uang standar.	Semua lokal	
AMAZON.Date	Mengenali kata-kata yang mewakili tanggal dan mengubahn	Semua lokal	

Jenis Slot	Deskripsi Singkat	Lokal yang Didukung
	ya menjadi format standar.	
AMAZON.Duration	Mengenali kata-kata yang mewakili durasi dan mengubahn ya menjadi format standar.	Semua lokal
AMAZON.EmailAddress	Mengenali kata-kata yang mewakili alamat email dan mengubahn ya menjadi alamat email standar.	Semua lokal
AMAZON.FirstName	Mengenali kata-kata yang mewakili nama depan.	Semua lokal
AMAZON.LastName	Mengenali kata-kata yang mewakili nama belakang.	Semua lokal
AMAZON.Number	Mengenali kata-kata numerik dan mengubahnya menjadi digit.	Semua lokal
AMAZON.Percentage	Mengenali kata-kata yang mewakili persentase dan mengubahnya menjadi angka dan tanda persen (%).	Semua lokal

Jenis Slot	Deskripsi Singkat	Lokal yang Didukung
AMAZON.PhoneNumber	Mengenali kata-kata yang mewakili nomor telepon dan mengubahnya menjadi string numerik.	Semua lokal
AMAZON.State	Mengakui kata-kata yang mewakili suatu negara.	Semua lokal
AMAZON.StreetName	Mengenali kata-kata yang mewakili nama jalan.	Semua lokal
AMAZON.Time	Mengenali kata-kata yang menunjukkan waktu dan mengubahnya menjadi format waktu.	Semua lokal
AMAZON.UKPostalCode	Mengenali kata-kata yang mewakili kode pos Inggris dan mengubahnya menjadi bentuk standar.	Hanya bahasa Inggris (Inggris) (en-GB)
AMAZON.FreeFormInput	Mengenali string yang terdiri dari kata atau karakter apa pun.	Semua lokal

AMAZON.AlphaNumeric

Mengenali string yang terdiri dari huruf dan angka, seperti. **`APQ123`**

Anda dapat menggunakan jenis `AMAZON.AlphaNumeric` slot untuk string yang berisi:

- Karakter abjad, seperti **`ABC`**
- Karakter numerik, seperti **`123`**
- Kombinasi karakter alfanumerik, seperti **`ABC123`**

Jenis `AMAZON.AlphaNumeric` slot mendukung input menggunakan gaya ejaan. Anda dapat menggunakan gaya ejaan demi huruf dan ejaan demi kata untuk membantu pelanggan Anda memasukkan huruf. Untuk informasi selengkapnya, lihat Menangkap nilai slot dengan gaya ejaan selama percakapan.

Anda dapat menambahkan ekspresi reguler ke jenis `AMAZON.AlphaNumeric` slot untuk memvalidasi nilai yang dimasukkan untuk slot. Misalnya, Anda dapat menggunakan ekspresi reguler untuk memvalidasi:

- Kode pos Kanada
- Nomor SIM
- Nomor identifikasi kendaraan

Gunakan ekspresi reguler standar. Amazon Lex V2 mendukung karakter berikut dalam ekspresi reguler:

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

Amazon Lex V2 juga mendukung karakter Unicode dalam ekspresi reguler. Bentuknya adalah`\u`*`Unicode`*. Gunakan empat digit untuk mewakili karakter Unicode. Misalnya, `[\u0041-\u005A]` setara dengan [A-Z].

Operator ekspresi reguler berikut tidak didukung:

- Repeater tak terbatas: *, +, atau {x,} tanpa batas atas.
- Kartu liar (.)

Panjang maksimum ekspresi reguler adalah 300 karakter. Panjang maksimum string yang disimpan dalam jenis `AMAZON.AlphaNumeric` slot yang menggunakan ekspresi reguler adalah 30 karakter.

Berikut ini adalah beberapa contoh ekspresi reguler.

- String alfanumerik, seperti **APQ123** atau**APQ1**: `[A-Z]{3}[0-9]{1,3}` atau yang lebih dibatasi `[A-DP-T]{3} [1-5]{1,3}`
- Format Internasional Surat Prioritas Layanan Pos AS, seperti**CP123456789US**: `CP[0-9]{9}US`
- Nomor routing bank, seperti**123456789**: `[0-9]{9}`

Untuk mengatur ekspresi reguler untuk jenis slot, gunakan konsol atau CreateSlotTypeoperasi. Ekspresi reguler divalidasi saat Anda menyimpan jenis slot. Jika ekspresi tidak valid, Amazon Lex V2 mengembalikan pesan kesalahan.

Saat Anda menggunakan ekspresi reguler dalam jenis slot, Amazon Lex V2 memeriksa input ke slot jenis itu terhadap ekspresi reguler. Jika input cocok dengan ekspresi, nilai diterima untuk slot. Jika input tidak cocok, Amazon Lex V2 meminta pengguna untuk mengulangi input.

AMAZON.City

Menyediakan daftar kota lokal dan dunia. Jenis slot mengenali variasi umum nama kota. Amazon Lex V2 tidak mengonversi dari variasi ke nama resmi.

Contoh:

- New York
- Reykjavik
- Tokyo
- Versailles

AMAZON.Confirmation

Jenis slot ini mengenali frasa input yang sesuai dengan frasa dan kata 'Ya', 'Tidak', 'Mungkin', dan 'Tidak tahu' untuk Amazon Lex V2 dan mengubahnya menjadi salah satu dari empat nilai. Ini dapat digunakan untuk menangkap konfirmasi atau pengakuan dari pengguna. Berdasarkan nilai akhir yang diselesaikan, Anda dapat membuat kondisi untuk merancang beberapa jalur percakapan.

Contoh:

jika {confirmation} = “Ya”, penuhi maksudnya

lain, dapatkan slot lain

Contoh:

- Ya: Ya, Ya, Ok, Tentu, saya memilikinya, saya setuju...
- Tidak: Tidak, Negatif, Tidak, Lupakan, Saya akan menolak, Tidak mungkin...
- Mungkin: Mungkin, Mungkin, Kadang-kadang, saya mungkin, Itu bisa benar...
- Tidak tahu: Entah, Tidak diketahui, Tidak tahu, Tidak yakin tentang itu, Siapa tahu...

Per 17 Agustus 2023, jika ada jenis slot khusus yang ada bernama “Konfirmasi”, nama harus diubah untuk menghindari konflik dengan Konfirmasi slot bawaan. Di navigasi sisi kiri di konsol Lex, buka jenis slot (untuk jenis slot khusus yang ada bernama Konfirmasi) dan perbarui nama jenis slot. Nama jenis slot baru tidak boleh “Konfirmasi,” yang merupakan kata kunci yang dicadangkan untuk jenis slot konfirmasi bawaan.

AMAZON.Country

Nama-nama negara di seluruh dunia. Contoh:

- Australia
- Germany
- Jepang
- Amerika Serikat
- Uruguay

AMAZON.Currency

Mengkonversi kata-kata yang mewakili mata uang menjadi standar ISO 4217 kode mata uang alfabet dan angka. Amazon Lex V2 mengenali mata uang tetapi tidak mengonversi dari satu mata uang ke mata uang lainnya.

Untuk informasi selengkapnya, lihat Kode mata uang - ISO 4217 di situs web Organisasi Internasional untuk Standardisasi (ISO).

Mata uang yang diwakili disusun sebagai berikut: `{Unit} {Amount}`

- {Unit} mengacu pada unit mata uang tertentu (misalnya, USD).

- {Jumlah} menunjukkan nilai moneter, diformat menjadi dua tempat desimal (misalnya, 300,00).

Contoh (semua contoh di bawah ini menggunakan lokal en-US; lokal yang berbeda dapat menghasilkan hasil yang berbeda):

- “3USD”: USD 3,00
- “USD300": USD 300,00
- “3 sen”: USD 0,30
- “$1,56": USD 1,56
- “5c”: USD 0,05
- “1 dolar”: USD 1,00
- “lima lima belas”: USD 515,00
- “lima dolar lima belas sen”: USD 5,15
- “5 usd dan 1/2 “: USD 5,50

AMAZON.Date

Mengonversi kata-kata yang mewakili tanggal menjadi format tanggal.

Tanggal disediakan untuk maksud Anda dalam format ISO-8601 tanggal. Tanggal yang diterima niat Anda di slot dapat bervariasi tergantung pada frasa spesifik yang diucapkan oleh pengguna.

- Ucapan yang dipetakan ke tanggal tertentu, seperti “hari ini,” “sekarang,” atau “dua puluh lima November,” dikonversi ke tanggal lengkap:. `2020-11-25` Ini default ke tanggal pada atau setelah tanggal saat ini.
- Ucapan yang dipetakan ke future week, seperti “minggu depan,” dikonversi ke tanggal hari terakhir minggu ini. Dalam ISO-8601 format, minggu dimulai pada hari Senin dan berakhir pada hari Minggu. Misalnya, jika hari ini 2020-11-25, “minggu depan” dikonversi menjadi. `2020-11-29` Tanggal yang dipetakan ke minggu saat ini atau sebelumnya dikonversi ke hari pertama dalam seminggu. Misalnya, jika hari ini 2020-11-25, “minggu lalu” dikonversi menjadi. `2020-11-16`
- Ucapan yang dipetakan ke bulan masa depan, tetapi bukan hari tertentu, seperti “bulan depan,” dikonversi ke hari terakhir bulan itu. Misalnya, jika hari ini 2020-11-25, “bulan depan” dikonversi menjadi. `2020-12-31` Untuk tanggal yang dipetakan ke bulan saat ini atau sebelumnya, konversi ke hari pertama bulan itu. Misalnya, jika hari ini 2020-11-25, “bulan ini” dipetakan ke. `2020-11-01`

- Ucapan yang dipetakan ke tahun masa depan, tetapi bukan bulan atau hari tertentu, seperti “tahun depan,” dikonversi ke hari terakhir tahun berikutnya. Misalnya, jika hari ini 2020-11-25, “tahun depan” dikonversi menjadi. `2021-12-31` Untuk tanggal yang dipetakan ke tahun saat ini atau tahun sebelumnya, konversi ke hari pertama tahun ini. Misalnya, jika hari ini 2020-11-25, “tahun lalu” dikonversi menjadi. `2019-01-01`

AMAZON.Duration

Mengkonversi kata-kata yang menunjukkan durasi menjadi durasi numerik.

Durasi diselesaikan ke format berdasarkan format ISO-8601 durasi,`PnYnMnWnDTnHnMnS`. Ini `P` menunjukkan bahwa ini adalah durasi, `n` adalah nilai numerik, dan huruf kapital yang mengikuti `n` adalah elemen tanggal atau waktu tertentu. Misalnya, `P3D` berarti 3 hari. A `T` digunakan untuk menunjukkan bahwa nilai yang tersisa mewakili elemen waktu daripada elemen tanggal.

Contoh:

- “sepuluh menit”: `PT10M`
- “lima jam”: `PT5H`
- “tiga hari”: `P3D`
- “empat puluh lima detik”: `PT45S`
- “delapan minggu”: `P8W`
- “tujuh tahun”: `P7Y`
- “lima jam sepuluh menit”: `PT5H10M`
- “dua tahun tiga jam sepuluh menit”: `P2YT3H10M`

AMAZON.EmailAddress

Mengenali kata-kata yang mewakili alamat email yang diberikan sebagai nama pengguna @domain. Alamat dapat menyertakan karakter khusus berikut dalam nama pengguna: garis bawah (_), tanda hubung (-), titik (.), dan tanda plus (+).

Jenis `AMAZON.EmailAddress` slot mendukung input menggunakan gaya ejaan. Anda dapat menggunakan gaya ejaan demi huruf dan ejaan demi kata untuk membantu pelanggan Anda memasukkan alamat email. Untuk informasi selengkapnya, lihat Menangkap nilai slot dengan gaya ejaan selama percakapan.

AMAZON.FirstName

Nama depan yang umum digunakan. Jenis slot ini mengenali nama formal, nama panggilan informal, dan nama yang terdiri dari lebih dari satu kata. Nama yang dikirim ke intent Anda adalah nilai yang dikirim oleh pengguna. Amazon Lex V2 tidak mengonversi dari nama panggilan ke nama resmi.

Untuk nama depan yang terdengar sama tetapi dieja berbeda, Amazon Lex V2 mengirimkan maksud Anda satu bentuk umum.

Jenis `AMAZON.FirstName` slot mendukung input menggunakan gaya ejaan. Anda dapat menggunakan gaya ejaan demi huruf dan ejaan demi kata untuk membantu pelanggan Anda memasukkan nama. Untuk informasi selengkapnya, lihat Menangkap nilai slot dengan gaya ejaan selama percakapan.

Contoh:

- Emily
- John
- Sophie
- Anil Kumar

AMAZON.FirstName juga mengembalikan daftar nama yang terkait erat berdasarkan nilai aslinya. Anda dapat menggunakan daftar nilai yang diselesaikan untuk memulihkan dari kesalahan ketik, mengonfirmasi nama dengan pengguna, atau melakukan pencarian database untuk nama yang valid di direktori pengguna Anda.

Misalnya, input “John” dapat mengakibatkan pengembalian nama terkait tambahan seperti “John J” dan “John-Paul”.

Berikut ini menunjukkan format respons untuk tipe slot `AMAZON.FirstName` bawaan:

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

```
}
```

AMAZON.LastName

Nama belakang yang umum digunakan. Untuk nama yang terdengar sama yang dieja berbeda, Amazon Lex V2 mengirimkan maksud Anda satu bentuk umum.

Jenis `AMAZON.LastName` slot mendukung input menggunakan gaya ejaan. Anda dapat menggunakan gaya ejaan demi huruf dan ejaan demi kata untuk membantu pelanggan Anda memasukkan nama. Untuk informasi selengkapnya, lihat Menangkap nilai slot dengan gaya ejaan selama percakapan.

Contoh:

- Brosky
- Dasher
- Evers
- Parres
- Welt

AMAZON.LastName juga mengembalikan daftar nama yang terkait erat berdasarkan nilai aslinya. Anda dapat menggunakan daftar nilai yang diselesaikan untuk memulihkan dari kesalahan ketik, mengonfirmasi nama dengan pengguna, atau melakukan pencarian database untuk nama yang valid di direktori pengguna Anda.

Misalnya, input “Smith” dapat mengakibatkan pengembalian nama terkait tambahan seperti “Smyth” dan “Smithe”.

Berikut ini menunjukkan format respons untuk tipe slot `AMAZON.LastName` bawaan:

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

```
}
```

AMAZON.Number

Mengkonversi kata atau angka yang mengekspresikan angka menjadi digit, termasuk angka desimal. Tabel berikut menunjukkan bagaimana jenis `AMAZON.Number` slot menangkap kata-kata numerik.

Input	Respons
seratus dua puluh tiga poin empat lima	123.45
seratus dua puluh tiga titik empat lima	123.45
titik empat dua	0,42
poin empat puluh dua	0,42
232.998	232.998
50	50
-15	-15
minus 15	-15
minus lima belas poin dua empat lima	-15,245

AMAZON.Percentage

Mengkonversi kata dan simbol yang mewakili persentase menjadi nilai numerik dengan tanda persen (%).

Jika pengguna memasukkan angka tanpa tanda persen atau kata “persen,” nilai slot diatur ke nomor tersebut. Tabel berikut menunjukkan bagaimana jenis `AMAZON.Percentage` slot menangkap persentase.

Input	Respons
50 persen	50%

Input	Respons
0,4 persen	0,4%
23,5%	23,5%
dua puluh lima persen	25%

AMAZON.PhoneNumber

Mengkonversi angka atau kata-kata yang mewakili nomor telepon ke dalam format string tanpa tanda baca sebagai berikut.

Tipe	Deskripsi	Input	Hasil
Nomor internasional dengan tanda plus (+) terkemuka	Nomor 11 digit dengan tanda plus terkemuka.	+61 7 4445 1061 +1 (509) 555-1212	`+61744431061` `+15095551212`
Nomor internasional tanpa tanda plus (+) terkemuka	Angka 11 digit tanpa tanda plus utama	1 (509) 555-1212 61 7 4445 1061	`15095551212` `61744451061`
Nomor nasional	10 digit nomor tanpa kode internasional	(03) 5115 4444 (509) 555-1212	`0351154444` `5095551212`
Nomor lokal	nomor telepon tanpa kode internasional atau kode area	555-1212	`5551212`

AMAZON.State

Nama-nama wilayah geografis dan politik di dalam negara.

Contoh:

- Bayern

- Prefektur Fukushima
- Pasifik Barat Laut
- Queensland
- Wales

AMAZON.StreetName

Nama-nama jalan dalam alamat jalan yang khas. Ini hanya termasuk nama jalan, bukan nomor rumah.

Contoh:

- Jalan Canberra
- Jalan Depan
- Jalan Pasar

AMAZON.Time

Mengubah kata-kata yang mewakili waktu menjadi nilai waktu. `AMAZON.Time`dapat menyelesaikan waktu yang tepat, nilai ambigu, dan rentang waktu. Nilai slot dapat diselesaikan ke rentang waktu berikut:

- SAYA
- PM
- MO (pagi)
- AF (sore)
- EV (malam)
- NI (malam)

Saat pengguna memasukkan waktu yang ambigu, Amazon Lex V2 menggunakan `slots` atribut peristiwa Lambda untuk meneruskan resolusi untuk waktu yang ambigu ke fungsi Lambda Anda. Misalnya, jika bot Anda meminta pengguna untuk waktu pengiriman, pengguna dapat merespons dengan mengatakan “jam 10.” Kali ini ambigu. Itu berarti 10:00 AM atau 10:00 PM. Dalam hal ini, nilai di `interpretedValue` lapangan adalah`null`, dan `resolvedValues` bidang berisi dua

kemungkinan resolusi saat itu. Amazon Lex V2 memasukkan yang berikut ini ke dalam fungsi Lambda:

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

Saat pengguna merespons dengan waktu yang tidak ambigu, Amazon Lex V2 mengirimkan waktu ke fungsi Lambda Anda di `interpretedValue` bidang atribut `slots` peristiwa Lambda. Misalnya, jika pengguna Anda merespons prompt untuk waktu pengiriman dengan "10:00 AM," Amazon Lex V2 memasukkan yang berikut ke dalam fungsi Lambda:

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

Saat pengguna merespons prompt untuk waktu pengiriman dengan “di pagi hari,” Amazon Lex V2 memasukkan hal berikut ke dalam fungsi Lambda:

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

```
    }
}
```

Untuk informasi selengkapnya tentang data yang dikirim dari Amazon Lex V2 ke fungsi Lambda, lihat. AWS Lambdaformat acara masukan untuk Lex V2

AMAZON.UKPostalCode

Mengonversi kata-kata yang mewakili kode pos Inggris ke format standar untuk kode pos di Britania Raya. Jenis `AMAZON.UKPostalCode` slot memvalidasi dan menyelesaikan kode pos ke satu set format standar, tetapi tidak memeriksa untuk memastikan bahwa kode pos valid. Aplikasi Anda harus memvalidasi kode pos.

Jenis `AMAZON.UKPostalCode` slot hanya tersedia di bahasa Inggris (UK) (en-GB) lokal.

Jenis `AMAZON.UKPostalCode` slot mendukung input menggunakan gaya ejaan. Anda dapat menggunakan gaya ejaan demi huruf dan ejaan demi kata untuk membantu pelanggan Anda memasukkan huruf. Untuk informasi selengkapnya, lihat Menangkap nilai slot dengan gaya ejaan selama percakapan.

Jenis slot hanya mengenali format kode pos yang valid yang tercantum di bawah ini, yang digunakan di Britania Raya. Format yang valid adalah (“A” menampilkan kembali huruf, dan “9" mewakili digit):

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

Untuk input teks, pengguna dapat memasukkan campuran huruf besar dan kecil. Pengguna dapat menggunakan atau menghilangkan ruang dalam kode pos. Nilai yang diselesaikan akan selalu menyertakan ruang di lokasi yang tepat untuk kode pos.

Untuk input lisan, pengguna dapat berbicara karakter individu, atau mereka dapat menggunakan pengucapan huruf ganda, seperti “ganda A” atau “ganda 9". Mereka juga dapat menggunakan pengucapan dua digit, seperti “sembilan puluh sembilan” untuk “99".

> **Note**
>
> Tidak semua kode pos Inggris diakui. Hanya format yang tercantum di atas yang didukung.

AMAZON.FreeFormInput

`AMAZON.FreeFormInput`dapat digunakan untuk menangkap input formulir gratis dari pengguna akhir. Ini mengenali string yang terdiri dari kata-kata atau karakter. Nilai yang diselesaikan adalah seluruh ucapan input.

Contoh:

Bot: Harap berikan umpan balik dari pengalaman panggilan Anda.

Pengguna: Saya mendapat jawaban atas semua pertanyaan saya, dan saya dapat menyelesaikan transaksi.

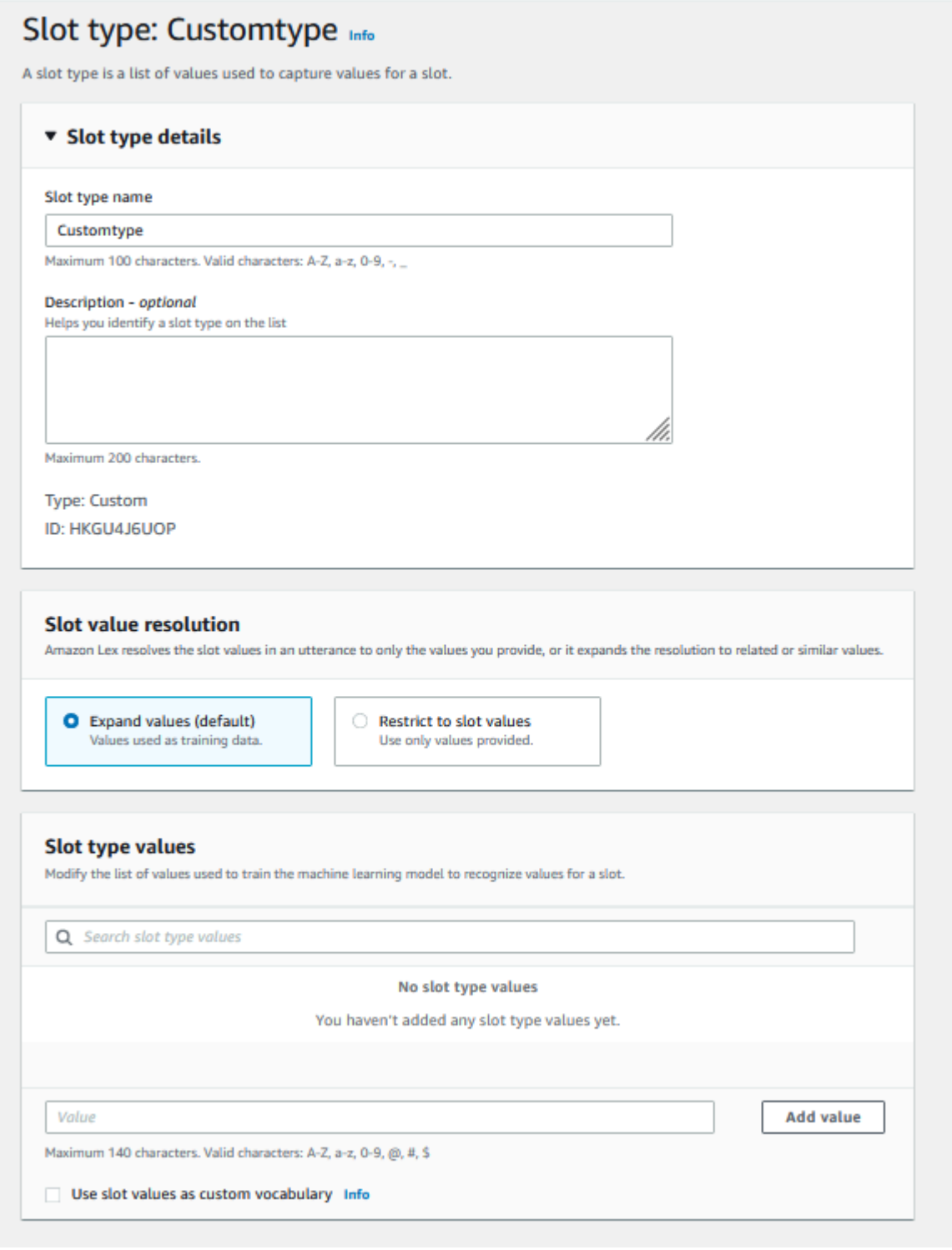
Catatan:

- `AMAZON.FreeFormInput`dapat digunakan untuk menangkap input formulir gratis apa adanya dari pengguna akhir.
- `AMAZON.FreeFormInput`tidak dapat digunakan dalam ucapan sampel maksud.
- `AMAZON.FreeFormInput`tidak dapat memiliki contoh ucapan slot.
- `AMAZON.FreeFormInput`hanya dikenali ketika diminta untuk.
- `AMAZON.FreeFormInput`tidak mendukung menunggu dan melanjutkan.
- `AMAZON.FreeFormInput`saat ini tidak didukung di saluran Connect Customer Chat.
- Ketika `AMAZON.FreeFormInput` slot ditimbulkan, tidak `FallbackIntent` akan dipicu.
- Ketika `AMAZON.FreeFormInput` slot ditimbulkan, tidak akan ada sakelar maksud.

Jenis slot khusus

Untuk setiap intent, Anda dapat menentukan parameter yang menunjukkan informasi yang dibutuhkan intent untuk memenuhi permintaan pengguna. Parameter ini, atau slot, memiliki tipe. Jenis slot adalah daftar nilai yang digunakan Amazon Lex V2 untuk melatih model pembelajaran mesin untuk mengenali nilai slot. Misalnya, Anda dapat menentukan jenis slot yang disebut `Genres` dengan nilai-nilai seperti “komedi,” “petualangan,” “dokumenter,” dan sebagainya. Anda dapat

menentukan sinonim untuk nilai jenis slot. Misalnya, Anda dapat mendefinisikan sinonim “lucu” dan “lucu” untuk nilai “komedi.”



Anda dapat mengonfigurasi jenis slot untuk memperluas nilai slot. Nilai slot akan digunakan sebagai data pelatihan dan model akan menyelesaikan slot dengan nilai yang diberikan oleh pengguna jika

mirip dengan nilai slot dan sinonim dari nilai-nilai tersebut. Ini adalah perilaku default. Amazon Lex V2 menyimpan daftar kemungkinan resolusi untuk slot. Setiap entri dalam daftar memberikan nilai yang diselesaikan yang diakui Amazon Lex V2 sebagai kemungkinan tambahan untuk slot. Nilai yang diselesaikan adalah upaya terbaik untuk mencocokkan nilai slot. Daftar ini berisi hingga lima nilai.

Atau, Anda dapat mengonfigurasi jenis slot untuk membatasi resolusi ke nilai slot. Dalam hal ini, model akan menyelesaikan nilai slot yang dimasukkan oleh pengguna ke nilai slot yang ada hanya jika sama dengan nilai slot itu atau itu adalah sinonim. Misalnya, jika pengguna memasukkan “lucu” itu akan menyelesaikan nilai slot “komedi.”

Ketika nilai yang dimasukkan oleh pengguna adalah sinonim dari nilai jenis slot, model mengembalikan nilai jenis slot sebagai entri pertama dalam daftar. `resolvedValues` Misalnya, jika pengguna memasukkan “lucu,” model mengisi `originalValue` bidang dengan nilai “lucu” dan entri pertama di bidang ResolvedValues dengan “komedi.” Anda dapat mengkonfigurasi `valueSelectionStrategy` ketika Anda membuat atau memperbarui jenis slot dengan CreateSlotTypeoperasi sehingga nilai slot diisi dengan nilai pertama dalam daftar resolusi.

Jenis slot khusus mendukung input menggunakan gaya ejaan. Anda dapat menggunakan spell-by-letter dan spell-by-word gaya untuk membantu pelanggan Anda memasukkan huruf. Untuk informasi selengkapnya, lihat Menangkap nilai slot dengan gaya ejaan selama percakapan.

Jika Anda menggunakan fungsi Lambda, peristiwa input ke fungsi tersebut menyertakan daftar resolusi yang disebut. `resolvedValues` Contoh berikut menunjukkan bagian slot input ke fungsi Lambda:

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

Untuk setiap jenis slot, Anda dapat menentukan maksimum 10.000 nilai dan sinonim. Setiap bot dapat memiliki jumlah total 50.000 nilai jenis slot dan sinonim. Misalnya, Anda dapat memiliki 5 jenis

slot, masing-masing dengan 5.000 nilai dan 5.000 sinonim, atau Anda dapat memiliki 10 jenis slot, masing-masing dengan 2.500 nilai dan 2.500 sinonim.

Jenis slot khusus tidak boleh memiliki nama yang sama dengan jenis slot bawaan. Misalnya, jenis slot khusus tidak boleh diberi nama dengan kata kunci yang dicadangkan dari Tanggal, Nomor, atau Konfirmasi. Kata kunci ini disediakan untuk jenis slot bawaan. Untuk daftar semua jenis slot bawaan, lihatBuilt-in jenis slot.

Jenis slot tata bahasa

> **Important**
>
> Amazon Lex V2 memberikan akses ke jenis slot tata bahasa untuk setiap akun AWS. Jika akun Anda tidak memiliki akses, Amazon Lex V2 menolak upaya untuk menambahkan jenis slot tata bahasa ke bot. Amazon Lex V2 juga menolak build untuk lokal yang menyertakan jenis slot tata bahasa. Untuk meminta akses ke akun Anda, hubungi Dukungan AWS. Jika Anda tidak perlu menulis tata bahasa Anda sendiri, Anda dapat menggunakan jenis slot khusus sebagai gantinya.

Dengan jenis slot tata bahasa, Anda dapat membuat tata bahasa Anda sendiri dalam format XML sesuai spesifikasi SRGS untuk mengumpulkan informasi dalam percakapan. Amazon Lex V2 mengenali ucapan yang cocok dengan aturan yang ditentukan dalam tata bahasa. Anda juga dapat memberikan aturan interpretasi semantik menggunakan tag ECMAScript dalam file tata bahasa. Amazon Lex kemudian mengembalikan properti yang ditetapkan dalam tag sebagai nilai yang diselesaikan saat kecocokan terjadi.

Anda hanya dapat membuat jenis slot tata bahasa dalam bahasa Inggris (Australia), Inggris (Inggris), dan Inggris (AS) lokal.

Ada dua bagian untuk jenis slot tata bahasa. Yang pertama adalah tata bahasa itu sendiri yang ditulis menggunakan format spesifikasi SRGS. Tata bahasa menafsirkan ucapan dari pengguna. Jika ucapan diterima oleh tata bahasa itu dicocokkan, jika tidak maka ditolak. Jika ucapan dicocokkan, itu diteruskan ke skrip jika ada.

Yang kedua adalah bagian dari jenis slot tata bahasa adalah skrip opsional yang ditulis dalam ECMAScript yang mengubah input ke nilai terselesaikan yang dikembalikan oleh jenis slot. Misalnya, Anda dapat menggunakan skrip untuk mengubah angka yang diucapkan menjadi digit. <tag>Pernyataan ECMAScript dilampirkan dalam elemen.

Contoh berikut adalah dalam format XML sesuai spesifikasi SRGS yang menunjukkan tata bahasa valid yang diterima oleh Amazon Lex V2. Ini mendefinisikan jenis slot tata bahasa yang menerima nomor kartu dan menentukan apakah mereka untuk akun reguler atau premium. Untuk informasi selengkapnya tentang sintaks yang dapat diterima, lihat Definisi tata bahasa dan top Format skrip iknya.

```
<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>
```

Tata bahasa di atas hanya menerima dua jenis nomor kartu: 7241 atau 7001. Keduanya dapat diawali secara opsional dengan “nomor kartu”. Ini juga berisi tag ECMAScript yang dapat digunakan untuk interpretasi semantik. Dengan interpretasi semantik, ucapan “kartu nomor tujuh dua empat satu” akan mengembalikan objek berikut:

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

```
}
```

Objek ini dikembalikan sebagai JSON-serialized string dalam `resolvedValues` objek yang dikembalikan oleh StartConversation operasi RecognizeText RecognizeUtterance,, dan.

Menambahkan jenis slot tata bahasa

Untuk menambahkan jenis slot tata bahasa

1. Unggah definisi XML dari jenis slot Anda ke bucket S3. Catat nama bucket dan path ke file.

   > **Note**
   >
   > Ukuran file maksimum adalah 100 KB.

2. Masuk ke Konsol Manajemen AWS dan buka konsol Amazon Lex di https://console.aws.amazon.com/lex/.
3. Dari menu kiri, pilih Bots dan kemudian pilih bot untuk menambahkan jenis slot tata bahasa.
4. Pilih Lihat bahasa, lalu pilih bahasa untuk menambahkan jenis slot tata bahasa.
5. Pilih Lihat jenis slot.
6. Pilih Tambahkan jenis slot, lalu pilih Tambahkan jenis slot tata bahasa.
7. Beri nama jenis slot, lalu pilih Tam bah.
8. Pilih bucket S3 yang berisi file definisi Anda dan masukkan jalur ke file tersebut. Pilih Simpan jenis slot.

Definisi tata bahasa

Topik ini menunjukkan bagian-bagian spesifikasi SRGS yang didukung Amazon Lex V2. Semua aturan didefinisikan dalam spesifikasi SRGS. Untuk informasi selengkapnya, lihat rekomendasi spesifikasi tata bahasa pengenalan ucapan versi 1.0 W3C.

Topik

- Deklarasi header
- Elemen XML yang didukung
- Token

- Referensi aturan
- Urutan dan enkapsulasi
- Pengulangan
- Bahasa
- Tanda
- Bobot

Dokumen ini mencakup materi yang disalin dan diturunkan dari W3C Speech Recognition Grammar Specification Version 1.0 (tersedia di https://www.w3.org/TR/speech-grammar/). Informasi kutipan berikut:

Hak C ipta © 2004 W3C® (MIT, ERCIM, Keio , Semua Hak Dilindungi. Kewaj iban W3C, merek dagang, penggunaan dokumen, dan aturan lisensi perangkat lunak berlaku.

Dokumen spesifikasi SRGS, Rekom endasi W3C, tersedia dari W3C di bawah lisensi berikut.

Teks lisensi

Lisensi

Dengan menggunakan and/or menyalin dokumen ini, atau dokumen W3C dari mana pernyataan ini ditautkan, Anda (penerima lisensi) setuju bahwa Anda telah membaca, memahami, dan akan mematuhi syarat dan ketentuan berikut:

Izin untuk menyalin, dan mendistribusikan isi dokumen ini, atau dokumen W3C dari mana pernyataan ini ditautkan, dalam media apa pun untuk tujuan apa pun dan tanpa biaya atau royalti dengan ini diberikan, asalkan Anda menyertakan hal-hal berikut pada SEMUA salinan dokumen, atau bagiannya, yang Anda gunakan:

- Tautan atau URL ke dokumen W3C asli.
- Pemberitahuan hak cipta yang sudah ada sebelumnya dari penulis asli, atau jika tidak ada, pemberitahuan (hiperteks lebih disukai, tetapi representasi tekstual diizinkan) dari bentuk: “Hak Cipta © [$date-of-document] World Wide Web Consortium, (MIT, ERCIM, Keio, Beihang). http://www.ercim.org/ http://www.keio.ac.jp/ http://ev.buaa.edu.cn/ http://www.w3.org/Consortium/Legal/2015/doc-license”
- Jika ada, STATUS dokumen W3C.

Ketika ruang memungkinkan, penyertaan teks lengkap PEMBERITAH UAN ini harus disediakan. Kami meminta agar atribusi kepengarangan disediakan dalam perangkat lunak, dokumen, atau item atau produk lain yang Anda buat sesuai dengan implementasi isi dokumen ini, atau bagiannya.

Tidak ada hak untuk membuat modifikasi atau turunan dari dokumen W3C yang diberikan sesuai dengan lisensi ini, kecuali sebagai berikut: Untuk memfasilitasi implementasi spesifikasi teknis yang ditetapkan dalam dokumen ini, siapa pun dapat menyiapkan dan mendistribusikan karya turunan dan bagian dari dokumen ini dalam perangkat lunak, dalam materi pendukung yang menyertainya perangkat lunak, dan dalam dokumentasi perangkat lunak, ASALKAN semua karya tersebut menyertakan pemberitahuan di bawah ini. NAMUN, publikasi karya turunan dari dokumen ini untuk digunakan sebagai spesifikasi teknis secara tegas dilarang.

Selain itu, “Komponen Kode” —IDL Web di bagian yang ditandai dengan jelas sebagai IDL Web; dan W3C-defined markup (HTML, CSS, dan sebagainya) dan kode bahasa pemrograman komputer yang ditandai dengan jelas sebagai contoh kode— dilisensikan di bawah Lisensi Perangkat Lunak W3C.

Pemberitahannya adalah:

“Hak Cipta © 2015 W3C® (MIT, ERCIM, Keio, Beihang). Perangkat lunak atau dokumen ini mencakup materi yang disalin dari atau berasal dari [judul dan URI dokumen W3C].”

Penafian

DOKUMEN INI DISEDIAKAN “SEBAGAIMANA ADANYA,” DAN PEMEGANG HAK CIPTA TIDAK MEMBUAT PERNYATAAN ATAU JAMINAN, TERSURAT ATAU TERSIRAT, TERMASUK, TETAPI TIDAK TERBATAS PADA, JAMINAN KELAYAKAN DIPERDAGANGKAN, KESESUAIAN UNTUK TUJUAN TERTENTU NON-INFRINGEMENT, ATAU HAK; BAHWA ISI DOKUMEN SESUAI UNTUK TUJUAN APA PUN; ATAU BAHWA PENERAPAN KONTEN TERSEBUT TIDAK AKAN MELANGGAR PATEN PIHAK KETIGA, HAK CIPTA, MEREK DAGANG ATAU HAK LAINNYA.

PEMEGANG HAK CIPTA TIDAK AKAN BERTANGGUNG JAWAB ATAS KERUSAKAN LANGSUNG, TIDAK LANGSUNG, KHUSUS ATAU KONSEKUENSIAL YANG TIMBUL DARI PENGGUNAAN DOKUMEN ATAU KINERJA ATAU IMPLEMENTASI ISINYA.

Nama dan merek dagang pemegang hak cipta TIDAK boleh digunakan dalam iklan atau publisitas yang berkaitan dengan dokumen ini atau isinya tanpa izin tertulis sebelumnya yang spesifik. Hak cipta dalam dokumen ini akan selalu berada di tangan pemegang hak cipta.

Deklarasi header

Tabel berikut menunjukkan deklarasi header yang didukung oleh jenis slot tata bahasa. Untuk informasi selengkapnya, lihat Deklarasi header tata bahasa dalam rekomendasi W3C spesifikasi tata bahasa pengenalan ucapan versi 1.

Pernyataan	Persyaratan spesifikasi	Formulir XML	Dukungan Amazon Lex	Spesifikasi
Versi tata bahasa	Diperlukan	4.3: `version` atribut pada `grammar` elemen	Diperlukan	SRGS
Ruang nama XML	Diperlukan (hanya XML)	4.3: `xmlns` atribut pada `grammar` elemen	Diperlukan	SRGS
Jenis dokumen	Diperlukan (hanya XML)	4.3: TIPE DOKUMEN XML	Disarankan	SRGS
Pengkodean karakter	Disarankan	4.4: `encoding` atribut dalam deklarasi XML	Disarankan	SRGS
Bahasa	Diperlukan dalam mode suara Diabaikan dalam mode DTMF	4.5: `xml:lang` atribut pada `grammar` elemen	Diperlukan dalam mode suara Diabaikan dalam mode DTMF	SRGS
Modus	Opsional	4.6: `mode` atribut pada `grammar` elemen	Opsional	SRGS

Pernyataan	Persyaratan spesifikasi	Formulir XML	Dukungan Amazon Lex	Spesifikasi
Aturan akar	Opsional	4.7: `root` atribut pada `grammar` elemen	Diperlukan	SRGS
Format tag	Opsional	4.8: `tag-format` atribut pada `grammar` elemen	String literal dan ECMAScript didukung	SRGS, SISR
URI dasar	Opsional	4.9: `xml:base` atribut pada `grammar` elemen	Opsional	SRGS
Leksikon pengucapan	Opsional, beberapa diizinkan	4.10: elemen `lexicon`	Tidak didukung	SRGS, TOLONG
Metadata	Opsional, beberapa diizinkan	4.11.1: elemen `meta`	Diperlukan	SRGS
Metadata XML	Opsional, hanya XML	4.11.2: elemen `metadata`	Opsional	SRGS
Tag	Opsional, beberapa diizinkan	4.12: elemen `tag`	Tag global tidak didukung	SRGS

Contoh

```
<?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">
```

Elemen XML yang didukung

Amazon Lex V2 mendukung elemen XML berikut untuk tata bahasa kustom:

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

Token

Tabel berikut menunjukkan spesifikasi token yang didukung oleh jenis slot tata bahasa. Untuk informasi selengkapnya, lihat Token dalam rekomendasi W3C spesifikasi tata bahasa pengenalan ucapan versi 1.

Jenis token	Contoh	Didukung?
Token tunggal yang tidak dikutip	halo	Ya
Token tunggal yang tidak dikutip: non-alfabet	2	Ya
Token yang dikutip tunggal, tidak ada spasi putih	"halo"	Ya, jatuhkan tanda kutip ganda jika hanya berisi satu token

Jenis token	Contoh	Didukung?
Dua token dibatasi oleh spasi putih	selamat perjalanan	Ya
Empat token dibatasi oleh spasi putih	ini adalah ujian	Ya
Token tunggal yang dikutip, termasuk spasi putih	“San Francisco	Tidak
Token XML tunggal dalam <token>tag	<token>San Fransisco</token>	Tidak (sama dengan token yang dikutip tunggal dengan spasi putih)

Catatan

- Token kutipan tunggal termasuk spasi putih — Spesifikasi mengharuskan kata-kata yang terlampir dalam tanda kutip ganda diperlakukan sebagai token tunggal. Amazon Lex V2 memperlakukannya sebagai token yang dibatasi spasi putih.
- Token XML tunggal di <token>- Spesifikasi membutuhkan kata-kata yang dibatasi oleh <token>untuk mewakili satu token. Amazon Lex V2 memperlakukannya sebagai token yang dibatasi spasi putih.
- Amazon Lex V2 menampilkan kesalahan validasi ketika salah satu penggunaan ditemukan dalam tata bahasa Anda.

Contoh

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

Referensi aturan

Tabel berikut merangkum berbagai bentuk referensi aturan yang dimungkinkan dalam dokumen tata bahasa. Untuk informasi selengkapnya, lihat Referensi aturan dalam rekomendasi W3C spesifikasi tata bahasa pengenalan ucapan versi 1.

Jenis referensi	Formulir XML	Didukung
2.2.1 Referensi aturan lokal eksplisit	`<ruleref uri="#rulename"/>`	Ya
2.2.2 Referensi eksplisit ke aturan bernama dari tata bahasa yang diidentifikasi oleh URI https://www.w3.org/TR/speech-grammar/#term-uri	`<ruleref uri="grammarURI#rulename"/>`	Tidak
2.2.2 Referensi implisit ke aturan akar tata bahasa yang diidentifikasi oleh URI https://www.w3.org/TR/speech-grammar/#term-uri	`<ruleref uri="grammarURI"/>`	Tidak
2.2.2 Referensi eksplisit ke aturan bernama tata bahasa yang diidentifikasi oleh URI dengan tipe media	`<ruleref uri="grammarURI#rulename" type="media-type"/>`	Tidak
2.2.2 Referensi implisit ke aturan akar tata bahasa yang diidentifikasi oleh URI dengan tipe media	`<ruleref uri="grammarURI" type="media-type"/>`	Tidak
2.2.3 Definisi aturan khusus	`<ruleref special="NULL"/>` `<ruleref special="VOID"/>`	Tidak

Jenis referensi	Formulir XML	Didukung
	`<ruleref special=" GARBAGE"/>`	

Catatan

1. Grammar URI adalah URI eksternal. Misalnya, `http://grammar.example.com/world-cities.grxml`.
2. Jenis media dapat berupa:
   - `application/srgs+xml`
   - `text/plain`

Contoh

```
<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>
```

Urutan dan enkapsulasi

Contoh berikut menunjukkan urutan yang didukung. Untuk informasi selengkapnya, lihat Ur utan dan enkapsulasi dalam rekomendasi W3C spesifikasi tata bahasa pengenalan ucapan versi 1.

Contoh

```
<!-- 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>
```

Pengulangan

Tabel berikut menunjukkan ekspansi berulang yang didukung untuk aturan. Untuk informasi selengkapnya, lihat Pengul angan dalam spesifikasi tata bahasa pengenalan ucapan versi 1 rekomendasi W3C.

Formulir XML Contoh	Perilaku	Didukung?
ulangi = “n” ulangi = “6"	Ekspresi yang terkandung diulang tepat “n” kali. “n” harus “0" atau bilangan bulat positif.	Ya
ulangi = “m-n” ulangi = “4-6”	Ekspansi yang terkandung diulang antara “m” dan “n” kali (inklusif). “m” dan “n” keduanya harus “0" atau bilangan bulat positif, dan “m” harus kurang dari atau sama dengan “n”.	Ya

Formulir XML Contoh	Perilaku	Didukung?
ulangi = “m-” ulangi = “3-”	Ekspansi yang terkandung diulang “m” kali atau lebih (inklusif). “m” harus “0" atau bilangan bulat postif. Misalnya, “3-” menyatakan bahwa ekspansi yang terkandung dapat terjadi tiga, empat, lima, atau lebih kali.	Ya
ulangi = “0-1"	Ekspansi yang terkandung adalah opsional.	Ya
<item repeat="2-4" repeat-prob="0.8">		Tidak

Bahasa

Diskusi berikut berlaku untuk pengidentifikasi bahasa yang diterapkan pada tata bahasa. Untuk informasi selengkapnya, lihat Bahasa dalam spesifikasi tata bahasa pengenalan ucapan versi 1 Rekom endasi W3C.

Secara default tata bahasa adalah dokumen bahasa tunggal dengan pengidentifikasi bahasa yang disediakan dalam deklarasi bahasa di header tata bahasa. Semua token dalam tata bahasa itu, kecuali dinyat akan lain, akan ditangani sesuai dengan bahasa tata bahasa. Grammar-level Deklarasi bahasa tidak didukung.

Dalam contoh berikut:

1. Deklarasi header tata bahasa untuk bahasa “en-US” didukung oleh Amazon Lex V2.
2. Item-level lampiran bahasa (disorot dalam*red*) tidak didukung. Amazon Lex V2 menampilkan kesalahan validasi jika lampiran bahasa berbeda dari deklarasi header.

```
<?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>
```

Tanda

Diskusi berikut berlaku untuk tag yang ditentukan untuk tata bahasa. Untuk informasi selengkapnya, lihat Tag dalam spesifikasi tata bahasa pengenalan ucapan versi 1 rekomendasi W3C.

Berdasarkan spesifikasi SRGS, tag dapat didefinisikan dengan cara berikut:

1. Sebagai bagian dari deklarasi header seperti yang dijelaskan dalamDeklarasi header.
2. Sebagai bagian dari <rule>definisi.

Format tag berikut didukung:

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

Format tag berikut tidak didukung:

- `swi-semantics/1.0`(Nuansa eksklusif)

Contoh

```
<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>
```

Bobot

Anda dapat menambahkan atribut bo bot ke elemen. Berat adalah nilai float point positif yang mewakili sejauh mana frasa dalam item ditingkatkan selama pengenalan ucapan. Untuk informasi selengkapnya, lihat Bobot dalam spesifikasi tata bahasa pengenalan ucapan versi 1 rekomendasi W3C.

Bobot harus lebih besar dari 0 dan kurang dari atau sama dengan 10, dan hanya dapat memiliki satu tempat desimal. Jika bobotnya lebih besar dari 0 dan kurang dari 1, frasa tersebut ditingkatkan secara negatif. Jika bobotnya lebih besar dari 1 dan kurang dari atau sama dengan 10, frasa tersebut

ditingkatkan secara positif. Berat 1 sama dengan tidak memberikan bobot sama sekali, dan tidak ada peningkatan untuk frasa tersebut.

Menetapkan bobot yang sesuai untuk item untuk meningkatkan kinerja pengenalan suara adalah tugas yang sulit. Berikut adalah beberapa tips yang dapat Anda ikuti untuk menetapkan bobot:

- Mulailah dengan tata bahasa tanpa bobot item yang ditetapkan.
- Tentukan pola mana dalam pidato yang sering salah diidentifikasi.
- Terapkan nilai yang berbeda untuk bobot sampai Anda melihat peningkatan kinerja pengenalan ucapan, dan tidak ada regresi.

Contoh 1

Misalnya, jika Anda memiliki tata bahasa untuk bandara, dan Anda melihat bahwa New York sering salah diidentifikasi sebagai Newark, Anda dapat secara positif meningkatkan New York dengan memberikannya bobot 5.

```
<rule> id="airport">
    <one-of>
        <item>
            Boston
            <tag>out="Boston"</tag>
        </item>
        <item weight="5">
            New York
            <tag>out="New York"</tag>
        </item>
        <item>
            Newark
            <tag>out="Newark"</tag>
        </item>
    </one-of>
</rule>
```

Contoh 2

Misalnya, Anda memiliki tata bahasa untuk kode reservasi maskapai yang dimulai dengan alfabet bahasa Inggris diikuti oleh tiga digit. Kode reservasi kemungkinan besar dimulai dengan B atau D,

tetapi Anda mengamati bahwa B sering salah diidentifikasi sebagai P, dan D sebagai T. Anda dapat secara positif meningkatkan B dan D.

```
<rule> id="alphabet">
    <one-of>
        <item>A<tag>out.letters+='A';</tag></item>
        <item weight="3.5">B<tag>out.letters+='B';</tag></item>
        <item>C<tag>out.letters+='C';</tag></item>
        <item weight="2.9">D<tag>out.letters+='D';</tag></item>
        <item>E<tag>out.letters+='E';</tag></item>
        <item>F<tag>out.letters+='F';</tag></item>
        <item>G<tag>out.letters+='G';</tag></item>
        <item>H<tag>out.letters+='H';</tag></item>
        <item>I<tag>out.letters+='I';</tag></item>
        <item>J<tag>out.letters+='J';</tag></item>
        <item>K<tag>out.letters+='K';</tag></item>
        <item>L<tag>out.letters+='L';</tag></item>
        <item>M<tag>out.letters+='M';</tag></item>
        <item>N<tag>out.letters+='N';</tag></item>
        <item>O<tag>out.letters+='O';</tag></item>
        <item>P<tag>out.letters+='P';</tag></item>
        <item>Q<tag>out.letters+='Q';</tag></item>
        <item>R<tag>out.letters+='R';</tag></item>
        <item>S<tag>out.letters+='S';</tag></item>
        <item>T<tag>out.letters+='T';</tag></item>
        <item>U<tag>out.letters+='U';</tag></item>
        <item>V<tag>out.letters+='V';</tag></item>
        <item>W<tag>out.letters+='W';</tag></item>
        <item>X<tag>out.letters+='X';</tag></item>
        <item>Y<tag>out.letters+='Y';</tag></item>
        <item>Z<tag>out.letters+='Z';</tag></item>
    </one-of>
</rule>
```

Format skrip

Amazon Lex V2 mendukung fitur ECMAScript berikut untuk mendefinisikan tata bahasa.

Amazon Lex V2 mendukung fitur ECMAScript berikut saat menentukan tag dalam tata bahasa. `tag-format`harus dikirim ke `semantics/1.0` ketika tag ECMAScript digunakan dalam tata bahasa. Untuk informasi lebih lanjut, lihat spesifikasi bahasa 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">
```

Topik

- Pernyataan variabel
- Ekspresi
- Jika pernyataan
- Pernyataan Switch
- Deklarasi fungsi
- Pernyataan iterasi
- Pernyataan blokir
- Komentar
- Pernyataan yang tidak didukung

Dokumen ini berisi materi dari standar ECMAScript (tersedia di https://www.ecma-international.org/publications-and-standards/standards/ecma-262/). Dokumen spesifikasi bahasa ECMAScript tersedia dari Ecma International di bawah lisensi berikut.

Teks lisensi

© 2020 Ecma Internasional

Dokumen ini dapat disalin, diterbitkan, dan didistribusikan kepada orang lain, dan karya turunan tertentu darinya dapat disiapkan, disalin, diterbitkan, dan didistribusikan, secara keseluruhan atau sebagian, asalkan pemberitahuan hak cipta di atas dan Lisensi Hak Cipta dan Penafian ini disertakan pada semua salinan dan karya turunan tersebut. Satu-satunya karya turunan yang diizinkan berdasarkan Lisensi dan Penafian Hak Cipta ini adalah:

(i) karya yang menggabungkan semua atau sebagian dari dokumen ini untuk tujuan memberikan komentar atau penjelasan (seperti versi dokumen beranotasi),

(ii) karya yang menggabungkan semua atau sebagian dari dokumen ini untuk tujuan menggabungkan fitur-fitur yang menyediakan aksesibilitas,

(iii) terjemahan dokumen ini ke dalam bahasa selain bahasa Inggris dan ke dalam format yang berbeda dan

(iv) bekerja dengan memanfaatkan spesifikasi ini dalam produk yang sesuai standar dengan menerapkan (misalnya, dengan menyalin dan menempel seluruhnya atau sebagian) fungsionalitas di dalamnya.

Namun, isi dokumen ini sendiri tidak boleh dimodifikasi dengan cara apa pun, termasuk dengan menghapus pemberitahuan hak cipta atau referensi ke Ecma International, kecuali sebagaimana diperlukan untuk menerjemahkannya ke dalam bahasa selain bahasa Inggris atau ke dalam format yang berbeda.

Versi resmi dokumen Ecma International adalah versi bahasa Inggris di situs web Ecma International. Dalam hal terjadi perbedaan antara versi terjemahan dan versi resmi, versi resmi akan mengatur.

Izin terbatas yang diberikan di atas bersifat abadi dan tidak akan dicabut oleh Ecma International atau penerus atau penugasnya. Dokumen ini dan informasi yang terkandung di sini disediakan atas dasar “SEBAGAIMANA ADANYA” dan ECMA INTERNATIONAL MENYANGKAL SEMUA JAMINAN, TERSURAT ATAU TERSIRAT, TERMASUK TETAPI TIDAK TERBATAS PADA JAMINAN APA PUN BAHWA PENGGUNAAN INFORMASI DI SINI TIDAK AKAN MELANGGAR HAK KEPEMILIKAN APA PUN ATAU JAMINAN TERSIRAT TENTANG KELAYAKAN PEDAGANG ATAU KESESUAIAN UNTUK TUJUAN TERTENTU.”

Pernyataan variabel

Pernyataan variabel mendefinisikan satu atau lebih variabel.

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

Ekspresi

Anda dapat menambahkan string ekspresi untuk menjalankan fungsi di Amazon Lex V2. Tabel ini menunjukkan sintaks dan contoh yang dapat digunakan dalam ekspresi SRGS.

Jenis ekspresi	Sintaksis	Contoh	Didukung?
Ekspresi reguler literal	String literal berisi karakter khusus regex yang valid	`"^\d\.$"`	Tidak

Jenis ekspresi	Sintaksis	Contoh	Didukung?
Fungsi	`function functionName(parameters) { functionBody}`	`var x = function calc() { return 10; }`	Tidak
Delete	`delete expression`	`delete obj.property;`	Tidak
Kosong	`void expression`	`void (2 == '2');`	Tidak
Tipeof	`typeof expression`	`typeof 42;`	Tidak
Indeks Anggota	`expression [ expressions ]`	`var fruits = ["apple"]; fruits[0];`	Ya
Anggota dot	`expression . identifier`	`out.value`	Ya
Argumen	`expression (arguments)`	`new Date('1994-10-11')`	Ya
Kenaikan pasca	`expression++`	`var x=10; x++;`	Ya
Pasca pengurangan	`expression--`	`var x=10; x--;`	Ya
Peningkatan pra-awal	`++expression`	`var x=10; ++x;`	Ya
Pra-dekrementasi	`--expression`	`var x=10; --x;`	Ya

Jenis ekspresi	Sintaksis	Contoh	Didukung?
Unary plus/Unary minus	`+expression / -expression`	`+x / -x;`	Ya
Sedikit tidak	`~ expression`	`const a = 5; consol e.log( ~a );`	Ya
Logis tidak	`! expression`	`!(a > 0 || b > 0)`	Ya
Multiplikatif	`expression ('*' | '/' | '%') expression`	`(x + y) * (a / b)`	Ya
Aditif	`expression ('+' | '-') expression`	`(a + b) - (a - (a + b))`	Ya
Pergeseran bit	`expression ('<<' | '>>' | '>>>') expression`	`(a >> b) >>> c`	Ya
Relatif	`expression ('<' | '>' | '<=' | '>=') expression`	`if (a > b) { ... }`	Ya
Masuk	`expression in expression`	`fruits[0] in otherFruits;`	Ya
Kesetaraan	`expression ('==' | '!=' | '===' | '!==') expression`	`if (a == b) { ... }`	Ya

Jenis ekspresi	Sintaksis	Contoh	Didukung?
Bit dan/xor/atau	`expression ('&' | '^' | '|') expression`	`a & b / a ^ b / a | b`	Ya
Logis dan/atau	`expression ('&&' | '||') expression`	`if (a && (b ||c)) { ...}`	Ya
Ternary	`expression ? expression : expression`	`a > b ? obj.prop : 0`	Ya
Penugasan	`expression = expression`	`out.value = "string";`	Ya
Operator penugasan	`expression ('*=' | '/=' | '+=' | '-=' | '%=') expression`	`a *= 10;`	Ya
Operator penugasan bitwise	`expression ('<<=' | '>>=' | '>>>=' | '&=' | '^=' | '|=') expression`	`a <<= 10;`	Ya

Jenis ekspresi	Sintaksis	Contoh	Didukung?
Pengidentifikasi	`identifierSequence` dimana IdentifierSequence adalah urutan karakter yang valid https://developer.mozilla.org/en-US/docs/Glossary/Identifier	`fruits=[10, 20, 30];`	Ya
Null literal	`null`	`x = null;`	Ya
Literal Boolean	`true | false`	`x = true;`	Ya
String literal	`'string' / "string"`	`a = 'hello', b = "world";`	Ya
Desimal literal	`integer [.] digits [exponent]`	`111.11 e+12`	Ya
Hex literal	`0 (x | X)[0-9a-fA-F]`	`0x123ABC`	Ya
Oktal literal	`0 [0-7]`	`"051"`	Ya
Array literal	`[ expression, ... ]`	`v = [a, b, c];`	Ya
Objek literal	`{property: value, ...}`	`out = {value: 1, flag: false};`	Ya

Jenis ekspresi	Sintaksis	Contoh	Didukung?
Ditandai dengan tanda kurung	`( expressions )`	`x + (x + y)`	Ya

Jika pernyataan

Anda dapat menambahkan pernyataan if untuk melakukan fungsi di Amazon Lex V2. Contoh ini menunjukkan sintaks yang dapat digunakan dalam ekspresi SRGS.

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

Catatan: Dalam contoh sebelumnya, `expressions` dan `statements` harus salah satu yang didukung dari dokumen ini.

Pernyataan Switch

Anda dapat menambahkan pernyataan switch untuk menjalankan fungsi di Amazon Lex V2. Contoh ini menunjukkan sintaks yang dapat digunakan dalam ekspresi SRGS.

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

Catatan: Dalam contoh sebelumnya, `expressions` dan `statements` harus salah satu yang didukung dari dokumen ini.

Deklarasi fungsi

Anda dapat menambahkan deklarasi fungsi untuk menjalankan fungsi di Amazon Lex V2. Contoh ini menunjukkan sintaks yang dapat digunakan dalam ekspresi SRGS.

```
function functionIdentifier([parameterList, ...]) {
     <function body>
}
```

Pernyataan iterasi

Pernyataan iterasi dapat berupa salah satu dari berikut ini:

```
// 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
}
```

Pernyataan blokir

Anda dapat menambahkan blok pernyataan untuk melakukan fungsi di Amazon Lex V2. Contoh ini menunjukkan sintaks yang dapat digunakan dalam ekspresi SRGS.

```
{
   statements
}

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

```
    }
}
```

Catatan: Dalam contoh sebelumnya, `statements` disediakan dalam blok harus salah satu yang didukung dari dokumen ini.

Komentar

Anda dapat menambahkan komentar di Amazon Lex V2. Contoh ini menunjukkan sintaks yang dapat digunakan dalam ekspresi SRGS.

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

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

Pernyataan yang tidak didukung

Amazon Lex V2 tidak mendukung fitur ECMAScript berikut.

Topik

- Pernyataan kosong
- Lanjutkan pernyataan
- Pernyataan istirahat
- Pernyataan pengembalian
- Lempar pernyataan
- Pernyataan coba
- Pernyataan Debugger
- Pernyataan berlabel
- Deklarasi kelas

Pernyataan kosong

Pernyataan kosong digunakan untuk tidak memberikan pernyataan. Berikut ini adalah sintaks untuk pernyataan kosong:

```
;
```

Lanjutkan pernyataan

Pernyataan lanjutan tanpa label didukung dengan[Pernyataan iterasi](). Pernyataan lanjutan dengan label tidak didukung.

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

Pernyataan istirahat

Pernyataan break tanpa label didukung dengan[Pernyataan iterasi](). Pernyataan break dengan label tidak didukung.

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

Pernyataan pengembalian

```
return expression;
```

Lempar pernyataan

Pernyataan lempar digunakan untuk melempar pengecualian yang ditentukan pengguna.

```
throw expression;
```

Pernyataan coba

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

```
}
finally {
  statements
}
```

Pernyataan Debugger

Pernyataan debugger digunakan untuk memanggil fungsionalitas debugging yang disediakan oleh lingkungan.

```
debugger;
```

Pernyataan berlabel

Pernyataan berlabel dapat digunakan dengan `continue` pernyataan `break` atau.

```
label:
   statements


// Example
let str = '';

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

console.log(str);
```

Deklarasi kelas

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

Tata bahasa industri

Tata bahasa industri adalah sekumpulan file XML untuk digunakan dengan jenis slot tata bahasa. Anda dapat menggunakannya untuk memberikan pengalaman pengguna akhir yang konsisten dengan cepat saat Anda memigrasikan alur kerja respons suara interaktif ke Amazon Lex V2. Anda dapat memilih dari berbagai tata bahasa pra-bangun di tiga domain: layanan keuangan, asuransi, dan telekomunikasi. Ada juga seperangkat tata bahasa generik yang dapat Anda gunakan sebagai titik awal untuk tata bahasa Anda sendiri.

Tata bahasa berisi aturan untuk mengumpulkan informasi dan tag ECMAScript untuk interpretasi semantik.

Tata bahasa untuk layanan keuangan (download)

Tata bahasa berikut didukung untuk layanan keuangan: nomor rekening dan routing, nomor kartu kredit dan pinjaman, skor kredit, tanggal pembukaan dan penutupan akun, dan nomor Jaminan Soal.

Nomor rekening

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">


        <!-- Test Cases

        Grammar will support the following inputs:

            Scenario 1:
                Input: My account number is A B C 1 2 3 4
                Output: ABC1234

            Scenario 2:
                Input: My account number is 1 2 3 4 A B C
                Output: 1234ABC

            Scenario 3:
```

```
                Input: Hmm My account number is 1 2 3 4 A B C 1
                Output: 123ABC1
        -->

        <rule id="main" scope="public">
            <tag>out=""</tag>
            <item><ruleref uri="#alphanumeric"/><tag>out +=
rules.alphanumeric.alphanum;</tag></item>
            <item repeat="0-1"><ruleref uri="#alphabets"/><tag>out +=
rules.alphabets.letters;</tag></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out +=
rules.digits.numbers</tag></item>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
            <one-of>
                <item repeat="0-1">account number is</item>
                <item repeat="0-1">Account Number</item>
                <item repeat="0-1">Here is my Account Number </item>
                <item repeat="0-1">Yes, It is</item>
                <item repeat="0-1">Yes It is</item>
                <item repeat="0-1">Yes It's</item>
                <item repeat="0-1">My account Id is</item>
                <item repeat="0-1">This is the account Id</item>
                <item repeat="0-1">account Id</item>
            </one-of>
        </rule>

        <rule id="hesitation">
          <one-of>
             <item>Hmm</item>
             <item>Mmm</item>
             <item>My</item>
          </one-of>
        </rule>

        <rule id="alphanumeric" scope="public">
            <tag>out.alphanum=""</tag>
            <item><ruleref uri="#alphabets"/><tag>out.alphanum +=
rules.alphabets.letters;</tag></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out.alphanum +=
rules.digits.numbers</tag></item>
        </rule>
```

```
        <rule id="alphabets">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.letters=""</tag>
            <tag>out.firstOccurence=""</tag>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out.firstOccurence +=
rules.digits.numbers; out.letters += out.firstOccurence;</tag></item>
            <item repeat="1-">
                <one-of>
                    <item>A<tag>out.letters+='A';</tag></item>
                    <item>B<tag>out.letters+='B';</tag></item>
                    <item>C<tag>out.letters+='C';</tag></item>
                    <item>D<tag>out.letters+='D';</tag></item>
                    <item>E<tag>out.letters+='E';</tag></item>
                    <item>F<tag>out.letters+='F';</tag></item>
                    <item>G<tag>out.letters+='G';</tag></item>
                    <item>H<tag>out.letters+='H';</tag></item>
                    <item>I<tag>out.letters+='I';</tag></item>
                    <item>J<tag>out.letters+='J';</tag></item>
                    <item>K<tag>out.letters+='K';</tag></item>
                    <item>L<tag>out.letters+='L';</tag></item>
                    <item>M<tag>out.letters+='M';</tag></item>
                    <item>N<tag>out.letters+='N';</tag></item>
                    <item>O<tag>out.letters+='O';</tag></item>
                    <item>P<tag>out.letters+='P';</tag></item>
                    <item>Q<tag>out.letters+='Q';</tag></item>
                    <item>R<tag>out.letters+='R';</tag></item>
                    <item>S<tag>out.letters+='S';</tag></item>
                    <item>T<tag>out.letters+='T';</tag></item>
                    <item>U<tag>out.letters+='U';</tag></item>
                    <item>V<tag>out.letters+='V';</tag></item>
                    <item>W<tag>out.letters+='W';</tag></item>
                    <item>X<tag>out.letters+='X';</tag></item>
                    <item>Y<tag>out.letters+='Y';</tag></item>
                    <item>Z<tag>out.letters+='Z';</tag></item>
                </one-of>
            </item>
        </rule>

        <rule id="digits">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.numbers=""</tag>
            <item repeat="1-10">
                <one-of>
```

```
                    <item>0<tag>out.numbers+=0;</tag></item>
                    <item>1<tag>out.numbers+=1;</tag></item>
                    <item>2<tag>out.numbers+=2;</tag></item>
                    <item>3<tag>out.numbers+=3;</tag></item>
                    <item>4<tag>out.numbers+=4;</tag></item>
                    <item>5<tag>out.numbers+=5;</tag></item>
                    <item>6<tag>out.numbers+=6;</tag></item>
                    <item>7<tag>out.numbers+=7;</tag></item>
                    <item>8<tag>out.numbers+=8;</tag></item>
                    <item>9<tag>out.numbers+=9;</tag></item>
                </one-of>
            </item>
        </rule>
</grammar>
```

Nomor perutean

```
<?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>
```

Nomor kartu kredit

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="digits"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: My credit card number is 1 2 3 4 5 6 7 8 9 1 2 3 4 5 6 7
                 Output: 1234567891234567

             Scenario 2:
                 Input: card number 1 2 3 4 5 6 7 8 9 1 2 3 4 5 6 7
                 Output: 1234567891234567

         -->

        <rule id="digits">
            <tag>out=""</tag>
            <item><ruleref uri="#singleDigit"/><tag>out += rules.singleDigit.digit;</
tag></item>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
           <one-of>
              <item repeat="0-1">My credit card number is</item>
```

```
            <item repeat="0-1">card number</item>
         </one-of>
      </rule>

      <rule id="hesitation">
          <one-of>
             <item>Hmm</item>
             <item>Mmm</item>
             <item>My</item>
          </one-of>
        </rule>

      <rule id="singleDigit">
          <item repeat="0-1"><ruleref uri="#text"/></item>
          <tag>out.digit=""</tag>
          <item repeat="16">
              <one-of>
                  <item>0<tag>out.digit+=0;</tag></item>
                  <item>zero<tag>out.digit+=0;</tag></item>
                  <item>1<tag>out.digit+=1;</tag></item>
                  <item>one<tag>out.digit+=1;</tag></item>
                  <item>2<tag>out.digit+=2;</tag></item>
                  <item>two<tag>out.digit+=2;</tag></item>
                  <item>3<tag>out.digit+=3;</tag></item>
                  <item>three<tag>out.digit+=3;</tag></item>
                  <item>4<tag>out.digit+=4;</tag></item>
                  <item>four<tag>out.digit+=4;</tag></item>
                  <item>5<tag>out.digit+=5;</tag></item>
                  <item>five<tag>out.digit+=5;</tag></item>
                  <item>6<tag>out.digit+=6;</tag></item>
                  <item>six<tag>out.digit+=5;</tag></item>
                  <item>7<tag>out.digit+=7;</tag></item>
                  <item>seven<tag>out.digit+=7;</tag></item>
                  <item>8<tag>out.digit+=8;</tag></item>
                  <item>eight<tag>out.digit+=8;</tag></item>
                  <item>9<tag>out.digit+=9;</tag></item>
                  <item>nine<tag>out.digit+=9;</tag></item>
              </one-of>
          </item>
      </rule>
</grammar>
```

ID Pinjaman

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">


        <!-- Test Cases

        Grammar will support the following inputs:

            Scenario 1:
                Input: My loan Id is A B C 1 2 3 4
                Output: ABC1234
        -->

        <rule id="main" scope="public">
            <tag>out=""</tag>
            <item><ruleref uri="#alphanumeric"/><tag>out +=
 rules.alphanumeric.alphanum;</tag></item>
            <item repeat="0-1"><ruleref uri="#alphabets"/><tag>out +=
 rules.alphabets.letters;</tag></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out +=
 rules.digits.numbers</tag></item>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
            <one-of>
                <item repeat="0-1">my loan number is</item>
                <item repeat="0-1">The loan number</item>
                <item repeat="0-1">The loan is </item>
                <item repeat="0-1">The number is</item>
                <item repeat="0-1">loan number</item>
                <item repeat="0-1">loan number of</item>
                <item repeat="0-1">loan Id is</item>
                <item repeat="0-1">My loan Id is</item>
            </one-of>
```

```
        </rule>

        <rule id="hesitation">
          <one-of>
             <item>Hmm</item>
             <item>Mmm</item>
             <item>My</item>
          </one-of>
        </rule>

        <rule id="alphanumeric" scope="public">
            <tag>out.alphanum=""</tag>
            <item><ruleref uri="#alphabets"/><tag>out.alphanum +=
rules.alphabets.letters;</tag></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out.alphanum +=
rules.digits.numbers</tag></item>
        </rule>

        <rule id="alphabets">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.letters=""</tag>
            <tag>out.firstOccurence=""</tag>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out.firstOccurence +=
rules.digits.numbers; out.letters += out.firstOccurence;</tag></item>
            <item repeat="1-">
                <one-of>
                    <item>A<tag>out.letters+='A';</tag></item>
                    <item>B<tag>out.letters+='B';</tag></item>
                    <item>C<tag>out.letters+='C';</tag></item>
                    <item>D<tag>out.letters+='D';</tag></item>
                    <item>E<tag>out.letters+='E';</tag></item>
                    <item>F<tag>out.letters+='F';</tag></item>
                    <item>G<tag>out.letters+='G';</tag></item>
                    <item>H<tag>out.letters+='H';</tag></item>
                    <item>I<tag>out.letters+='I';</tag></item>
                    <item>J<tag>out.letters+='J';</tag></item>
                    <item>K<tag>out.letters+='K';</tag></item>
                    <item>L<tag>out.letters+='L';</tag></item>
                    <item>M<tag>out.letters+='M';</tag></item>
                    <item>N<tag>out.letters+='N';</tag></item>
                    <item>O<tag>out.letters+='O';</tag></item>
                    <item>P<tag>out.letters+='P';</tag></item>
                    <item>Q<tag>out.letters+='Q';</tag></item>
                    <item>R<tag>out.letters+='R';</tag></item>
```

```
                    <item>S<tag>out.letters+='S';</tag></item>
                    <item>T<tag>out.letters+='T';</tag></item>
                    <item>U<tag>out.letters+='U';</tag></item>
                    <item>V<tag>out.letters+='V';</tag></item>
                    <item>W<tag>out.letters+='W';</tag></item>
                    <item>X<tag>out.letters+='X';</tag></item>
                    <item>Y<tag>out.letters+='Y';</tag></item>
                    <item>Z<tag>out.letters+='Z';</tag></item>
                </one-of>
            </item>
        </rule>

        <rule id="digits">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.numbers=""</tag>
            <item repeat="1-10">
                <one-of>
                    <item>0<tag>out.numbers+=0;</tag></item>
                    <item>1<tag>out.numbers+=1;</tag></item>
                    <item>2<tag>out.numbers+=2;</tag></item>
                    <item>3<tag>out.numbers+=3;</tag></item>
                    <item>4<tag>out.numbers+=4;</tag></item>
                    <item>5<tag>out.numbers+=5;</tag></item>
                    <item>6<tag>out.numbers+=6;</tag></item>
                    <item>7<tag>out.numbers+=7;</tag></item>
                    <item>8<tag>out.numbers+=8;</tag></item>
                    <item>9<tag>out.numbers+=9;</tag></item>
                </one-of>
            </item>
        </rule>
</grammar>
```

Skor kredit

```
<?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>
```

Tanggal pembukaan rekening

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
```

```
                Input: I opened account on July Two Thousand and Eleven
                Output: 07/11

             Scenario 2:
                Input: I need account number opened on July Two Thousand and Eleven
                Output: 07/11

         -->

        <rule id="main" scope="public">
            <tag>out=""</tag>
            <item repeat="1-10">
                <item repeat="1"><ruleref uri="#months"/><tag>out = out +
 rules.months.mon + "/";</tag></item>
                <one-of>
                    <item><ruleref uri="#thousands"/><tag>out += rules.thousands;</
tag></item>
                    <item repeat="0-1"><ruleref uri="#digits"/><tag>out +=
 rules.digits;</tag></item>
                    <item repeat="0-1"><ruleref uri="#teens"/><tag>out +=
 rules.teens;</tag></item>
                    <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out +=
 rules.above_twenty;</tag></item>
                </one-of>
            </item>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
           <one-of>
              <item repeat="0-1">I opened account on </item>
              <item repeat="0-1">I need account number opened on </item>
           </one-of>
        </rule>

        <rule id="hesitation">
            <one-of>
               <item>Hmm</item>
               <item>Mmm</item>
               <item>My</item>
            </one-of>
          </rule>
        <rule id="months">
           <item repeat="0-1"><ruleref uri="#text"/></item>
```

```
            <tag>out.mon=""</tag>
            <one-of>
                <item>january<tag>out.mon+="01";</tag></item>
                <item>february<tag>out.mon+="02";</tag></item>
                <item>march<tag>out.mon+="03";</tag></item>
                <item>april<tag>out.mon+="04";</tag></item>
                <item>may<tag>out.mon+="05";</tag></item>
                <item>june<tag>out.mon+="06";</tag></item>
                <item>july<tag>out.mon+="07";</tag></item>
                <item>august<tag>out.mon+="08";</tag></item>
                <item>september<tag>out.mon+="09";</tag></item>
                <item>october<tag>out.mon+="10";</tag></item>
                <item>november<tag>out.mon+="11";</tag></item>
                <item>december<tag>out.mon+="12";</tag></item>
                <item>jan<tag>out.mon+="01";</tag></item>
                <item>feb<tag>out.mon+="02";</tag></item>
                <item>aug<tag>out.mon+="08";</tag></item>
                <item>sept<tag>out.mon+="09";</tag></item>
                <item>oct<tag>out.mon+="10";</tag></item>
                <item>nov<tag>out.mon+="11";</tag></item>
                <item>dec<tag>out.mon+="12";</tag></item>
            </one-of>
        </rule>

        <rule id="digits">
            <one-of>
                <item>zero<tag>out=0;</tag></item>
                <item>one<tag>out=1;</tag></item>
                <item>two<tag>out=2;</tag></item>
                <item>three<tag>out=3;</tag></item>
                <item>four<tag>out=4;</tag></item>
                <item>five<tag>out=5;</tag></item>
                <item>six<tag>out=6;</tag></item>
                <item>seven<tag>out=7;</tag></item>
                <item>eight<tag>out=8;</tag></item>
                <item>nine<tag>out=9;</tag></item>
            </one-of>
        </rule>

        <rule id="teens">
            <one-of>
                <item>ten<tag>out=10;</tag></item>
                <item>eleven<tag>out=11;</tag></item>
                <item>twelve<tag>out=12;</tag></item>
```

```
                <item>thirteen<tag>out=13;</tag></item>
                <item>fourteen<tag>out=14;</tag></item>
                <item>fifteen<tag>out=15;</tag></item>
                <item>sixteen<tag>out=16;</tag></item>
                <item>seventeen<tag>out=17;</tag></item>
                <item>eighteen<tag>out=18;</tag></item>
                <item>nineteen<tag>out=19;</tag></item>
            </one-of>
        </rule>

        <rule id="thousands">
            <item>two thousand<!--<tag>out=2000;</tag>--></item>
            <item repeat="0-1">and</item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out = rules.digits;</tag></
item>
            <item repeat="0-1"><ruleref uri="#teens"/><tag>out = rules.teens;</tag></
item>
            <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out =
 rules.above_twenty;</tag></item>
        </rule>

        <rule id="above_twenty">
            <one-of>
                <item>twenty<tag>out=20;</tag></item>
                <item>thirty<tag>out=30;</tag></item>
                <item>forty<tag>out=40;</tag></item>
                <item>fifty<tag>out=50;</tag></item>
                <item>sixty<tag>out=60;</tag></item>
                <item>seventy<tag>out=70;</tag></item>
                <item>eighty<tag>out=80;</tag></item>
                <item>ninety<tag>out=90;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>
</grammar>
```

Tanggal pembayaran otomatis

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
```

```
                              http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: I want to schedule auto pay for twenty five Dollar
                 Output: $25

             Scenario 2:
                 Input: Setup automatic payments for twenty five dollars
                 Output: $25

         -->

        <rule id="main" scope="public">
            <tag>out="$"</tag>
            <one-of>
                <item><ruleref uri="#sub_hundred"/><tag>out += rules.sub_hundred.sh;</
tag></item>
                <item><ruleref uri="#subThousands"/><tag>out += rules.subThousands;</
tag></item>
            </one-of>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
           <one-of>
              <item repeat="0-1">I want to schedule auto pay for</item>
              <item repeat="0-1">Setup automatic payments for twenty five dollars</
item>
              <item repeat="0-1">Auto pay amount of</item>
              <item repeat="0-1">Set it up for</item>
           </one-of>
        </rule>

        <rule id="hesitation">
            <one-of>
               <item>Hmm</item>
```

```
            <item>Mmm</item>
            <item>My</item>
        </one-of>
      </rule>

    <rule id="digits">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <tag>out.num = 0;</tag>
        <one-of>
            <item>0<tag>out.num+=0;</tag></item>
            <item>1<tag>out.num+=1;</tag></item>
            <item>2<tag>out.num+=2;</tag></item>
            <item>3<tag>out.num+=3;</tag></item>
            <item>4<tag>out.num+=4;</tag></item>
            <item>5<tag>out.num+=5;</tag></item>
            <item>6<tag>out.num+=6;</tag></item>
            <item>7<tag>out.num+=7;</tag></item>
            <item>8<tag>out.num+=8;</tag></item>
            <item>9<tag>out.num+=9;</tag></item>
            <item>one<tag>out.num+=1;</tag></item>
            <item>two<tag>out.num+=2;</tag></item>
            <item>three<tag>out.num+=3;</tag></item>
            <item>four<tag>out.num+=4;</tag></item>
            <item>five<tag>out.num+=5;</tag></item>
            <item>six<tag>out.num+=6;</tag></item>
            <item>seven<tag>out.num+=7;</tag></item>
            <item>eight<tag>out.num+=8;</tag></item>
            <item>nine<tag>out.num+=9;</tag></item>
        </one-of>
        <item repeat="0-1"><ruleref uri="#currency"/></item>
    </rule>

    <rule id="teens">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <tag>out.teen = 0;</tag>
        <one-of>
            <item>ten<tag>out.teen+=10;</tag></item>
            <item>eleven<tag>out.teen+=11;</tag></item>
            <item>twelve<tag>out.teen+=12;</tag></item>
            <item>thirteen<tag>out.teen+=13;</tag></item>
            <item>fourteen<tag>out.teen+=14;</tag></item>
            <item>fifteen<tag>out.teen+=15;</tag></item>
            <item>sixteen<tag>out.teen+=16;</tag></item>
            <item>seventeen<tag>out.teen+=17;</tag></item>
```

```
                <item>eighteen<tag>out.teen+=18;</tag></item>
                <item>nineteen<tag>out.teen+=19;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#currency"/></item>
        </rule>

        <rule id="above_twenty">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.tens = 0;</tag>
            <one-of>
                <item>twenty<tag>out.tens+=20;</tag></item>
                <item>thirty<tag>out.tens+=30;</tag></item>
                <item>forty<tag>out.tens+=40;</tag></item>
                <item>fifty<tag>out.tens+=50;</tag></item>
                <item>sixty<tag>out.tens+=60;</tag></item>
                <item>seventy<tag>out.tens+=70;</tag></item>
                <item>eighty<tag>out.tens+=80;</tag></item>
                <item>ninety<tag>out.tens+=90;</tag></item>
                <item>hundred<tag>out.tens+=100;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#currency"/></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out.tens +=
 rules.digits.num;</tag></item>
        </rule>

        <rule id="currency">
            <one-of>
                <item repeat="0-1">dollars</item>
                <item repeat="0-1">Dollars</item>
                <item repeat="0-1">dollar</item>
                <item repeat="0-1">Dollar</item>
            </one-of>
        </rule>


        <rule id="sub_hundred">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.sh = 0;</tag>
            <one-of>
                <item><ruleref uri="#teens"/><tag>out.sh += rules.teens.teen;</tag></
item>
                <item>
                    <ruleref uri="#above_twenty"/><tag>out.sh +=
 rules.above_twenty.tens;</tag>
```

```
                </item>
                <item><ruleref uri="#digits"/><tag>out.sh += rules.digits.num;</tag></
item>
            </one-of>
         </rule>

        <rule id="subThousands">
            <ruleref uri="#sub_hundred"/><tag>out = (100 * rules.sub_hundred.sh);</tag>
            hundred
            <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out +=
 rules.above_twenty.tens;</tag></item>
            <item repeat="0-1"><ruleref uri="#teens"/><tag>out += rules.teens.teen;</
tag></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits.num;</
tag></item>
            <item repeat="0-1"><ruleref uri="#currency"/></item>
        </rule>
</grammar>
```

Tanggal kedaluwarsa kartu kredit

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="dateCardExpiration"
         mode="voice"
         tag-format="semantics/1.0">

        <rule id="dateCardExpiration" scope="public">
            <tag>out="";</tag>
            <item repeat="1"><ruleref uri="#months"/><tag>out = out + rules.months;</
tag></item>
            <item repeat="1"><ruleref uri="#year"/><tag>out += " " + rules.year.yr;</
tag></item>
        </rule>

        <!-- Test Cases

        Grammar will support the following inputs:
```

```
            Scenario 1:
                Input: My card expiration date is july eleven
                Output: 07 2011

            Scenario 2:
                Input: My card expiration date is may twenty six
                Output: 05 2026

        -->

        <rule id="text">
           <item repeat="0-1"><ruleref uri="#hesitation"/></item>
           <one-of>
              <item repeat="0-1">My card expiration date is </item>
              <item repeat="0-1">Expiration date is </item>
           </one-of>
        </rule>

        <rule id="hesitation">
          <one-of>
             <item>Hmm</item>
             <item>Mmm</item>
             <item>My</item>
          </one-of>
        </rule>

        <rule id="months">
           <item repeat="0-1"><ruleref uri="#text"/></item>
           <one-of>
               <item>january<tag>out="01";</tag></item>
               <item>february<tag>out="02";</tag></item>
               <item>march<tag>out="03";</tag></item>
               <item>april<tag>out="04";</tag></item>
               <item>may<tag>out="05";</tag></item>
               <item>june<tag>out="06";</tag></item>
               <item>july<tag>out="07";</tag></item>
               <item>august<tag>out="08";</tag></item>
               <item>september<tag>out="09";</tag></item>
               <item>october<tag>out="10";</tag></item>
               <item>november<tag>out="11";</tag></item>
               <item>december<tag>out="12";</tag></item>
               <item>jan<tag>out="01";</tag></item>
               <item>feb<tag>out="02";</tag></item>
               <item>aug<tag>out="08";</tag></item>
```

```
            <item>sept<tag>out="09";</tag></item>
            <item>oct<tag>out="10";</tag></item>
            <item>nov<tag>out="11";</tag></item>
            <item>dec<tag>out="12";</tag></item>
            <item>1<tag>out="01";</tag></item>
            <item>2<tag>out="02";</tag></item>
            <item>3<tag>out="03";</tag></item>
            <item>4<tag>out="04";</tag></item>
            <item>5<tag>out="05";</tag></item>
            <item>6<tag>out="06";</tag></item>
            <item>7<tag>out="07";</tag></item>
            <item>8<tag>out="08";</tag></item>
            <item>9<tag>out="09";</tag></item>
            <item>ten<tag>out="10";</tag></item>
            <item>eleven<tag>out="11";</tag></item>
            <item>twelve<tag>out="12";</tag></item>
        </one-of>
    </rule>

     <rule id="digits">
         <item repeat="0-1"><ruleref uri="#text"/></item>
         <one-of>
             <item>0<tag>out=0;</tag></item>
             <item>1<tag>out=1;</tag></item>
             <item>2<tag>out=2;</tag></item>
             <item>3<tag>out=3;</tag></item>
             <item>4<tag>out=4;</tag></item>
             <item>5<tag>out=5;</tag></item>
             <item>6<tag>out=6;</tag></item>
             <item>7<tag>out=7;</tag></item>
             <item>8<tag>out=8;</tag></item>
             <item>9<tag>out=9;</tag></item>
             <item>one<tag>out=1;</tag></item>
             <item>two<tag>out=2;</tag></item>
             <item>three<tag>out=3;</tag></item>
             <item>four<tag>out=4;</tag></item>
             <item>five<tag>out=5;</tag></item>
             <item>six<tag>out=6;</tag></item>
             <item>seven<tag>out=7;</tag></item>
             <item>eight<tag>out=8;</tag></item>
             <item>nine<tag>out=9;</tag></item>
         </one-of>
     </rule>
```

```
        <rule id="year">
          <tag>out.yr="20"</tag>
          <one-of>
              <item><ruleref uri="#teens"/><tag>out.yr += rules.teens;</tag></item>
              <item><ruleref uri="#above_twenty"/><tag>out.yr += rules.above_twenty;</
tag></item>
          </one-of>
        </rule>

        <rule id="teens">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <one-of>
                <item>ten<tag>out=10;</tag></item>
                <item>eleven<tag>out=11;</tag></item>
                <item>twelve<tag>out=12;</tag></item>
                <item>thirteen<tag>out=13;</tag></item>
                <item>fourteen<tag>out=14;</tag></item>
                <item>fifteen<tag>out=15;</tag></item>
                <item>sixteen<tag>out=16;</tag></item>
                <item>seventeen<tag>out=17;</tag></item>
                <item>eighteen<tag>out=18;</tag></item>
                <item>nineteen<tag>out=19;</tag></item>
                <item>10<tag>out=10;</tag></item>
                <item>11<tag>out=11;</tag></item>
                <item>12<tag>out=12;</tag></item>
                <item>13<tag>out=13;</tag></item>
                <item>14<tag>out=14;</tag></item>
                <item>15<tag>out=15;</tag></item>
                <item>16<tag>out=16;</tag></item>
                <item>17<tag>out=17;</tag></item>
                <item>18<tag>out=18;</tag></item>
                <item>19<tag>out=19;</tag></item>
            </one-of>
        </rule>

        <rule id="above_twenty">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <one-of>
                <item>twenty<tag>out=20;</tag></item>
                <item>thirty<tag>out=30;</tag></item>
                <item>forty<tag>out=40;</tag></item>
                <item>fifty<tag>out=50;</tag></item>
                <item>sixty<tag>out=60;</tag></item>
                <item>seventy<tag>out=70;</tag></item>
```

```
                <item>eighty<tag>out=80;</tag></item>
                <item>ninety<tag>out=90;</tag></item>
                <item>20<tag>out=20;</tag></item>
                <item>30<tag>out=30;</tag></item>
                <item>40<tag>out=40;</tag></item>
                <item>50<tag>out=50;</tag></item>
                <item>60<tag>out=60;</tag></item>
                <item>70<tag>out=70;</tag></item>
                <item>80<tag>out=80;</tag></item>
                <item>90<tag>out=90;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>
</grammar>
```

Tanggal pernyataan

```
<?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>
```

Tanggal transaksi

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
```

```
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: My last incorrect transaction date is july twenty three
                 Output: 07/23

             Scenario 2:
                 Input: My last incorrect transaction date is july fifteen
                 Output: 07/15

         -->

        <rule id="main" scope="public">
            <tag>out=""</tag>
            <item repeat="1-10">
                <item><ruleref uri="#months"/><tag>out= rules.months.mon + "/";</tag></
item>
                <one-of>
                    <item><ruleref uri="#digits"/><tag>out+= rules.digits;</tag></item>
                    <item><ruleref uri="#teens"/><tag>out+= rules.teens;</tag></item>
                    <item><ruleref uri="#above_twenty"/><tag>out+=
 rules.above_twenty;</tag></item>
                </one-of>
            </item>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
           <one-of>
              <item repeat="0-1">My last incorrect transaction date is</item>
              <item repeat="0-1">It is</item>
           </one-of>
        </rule>
        <rule id="hesitation">
            <one-of>
               <item>Hmm</item>
               <item>Mmm</item>
               <item>My</item>
            </one-of>
          </rule>
```

```
<rule id="months">
   <item repeat="0-1"><ruleref uri="#text"/></item>
   <tag>out.mon=""</tag>
   <one-of>
       <item>january<tag>out.mon+="01";</tag></item>
       <item>february<tag>out.mon+="02";</tag></item>
       <item>march<tag>out.mon+="03";</tag></item>
       <item>april<tag>out.mon+="04";</tag></item>
       <item>may<tag>out.mon+="05";</tag></item>
       <item>june<tag>out.mon+="06";</tag></item>
       <item>july<tag>out.mon+="07";</tag></item>
       <item>august<tag>out.mon+="08";</tag></item>
       <item>september<tag>out.mon+="09";</tag></item>
       <item>october<tag>out.mon+="10";</tag></item>
       <item>november<tag>out.mon+="11";</tag></item>
       <item>december<tag>out.mon+="12";</tag></item>
       <item>jan<tag>out.mon+="01";</tag></item>
       <item>feb<tag>out.mon+="02";</tag></item>
       <item>aug<tag>out.mon+="08";</tag></item>
       <item>sept<tag>out.mon+="09";</tag></item>
       <item>oct<tag>out.mon+="10";</tag></item>
       <item>nov<tag>out.mon+="11";</tag></item>
       <item>dec<tag>out.mon+="12";</tag></item>
   </one-of>
</rule>

<rule id="digits">
    <item repeat="0-1"><ruleref uri="#text"/></item>
    <one-of>
        <item>0<tag>out=0;</tag></item>
        <item>1<tag>out=1;</tag></item>
        <item>2<tag>out=2;</tag></item>
        <item>3<tag>out=3;</tag></item>
        <item>4<tag>out=4;</tag></item>
        <item>5<tag>out=5;</tag></item>
        <item>6<tag>out=6;</tag></item>
        <item>7<tag>out=7;</tag></item>
        <item>8<tag>out=8;</tag></item>
        <item>9<tag>out=9;</tag></item>
        <item>first<tag>out=01;</tag></item>
        <item>second<tag>out=02;</tag></item>
        <item>third<tag>out=03;</tag></item>
        <item>fourth<tag>out=04;</tag></item>
```

```
            <item>fifth<tag>out=05;</tag></item>
            <item>sixth<tag>out=06;</tag></item>
            <item>seventh<tag>out=07;</tag></item>
            <item>eighth<tag>out=08;</tag></item>
            <item>ninth<tag>out=09;</tag></item>
            <item>one<tag>out=1;</tag></item>
            <item>two<tag>out=2;</tag></item>
            <item>three<tag>out=3;</tag></item>
            <item>four<tag>out=4;</tag></item>
            <item>five<tag>out=5;</tag></item>
            <item>six<tag>out=6;</tag></item>
            <item>seven<tag>out=7;</tag></item>
            <item>eight<tag>out=8;</tag></item>
            <item>nine<tag>out=9;</tag></item>
        </one-of>
    </rule>

    <rule id="teens">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <one-of>
            <item>ten<tag>out=10;</tag></item>
            <item>tenth<tag>out=10;</tag></item>
            <item>eleven<tag>out=11;</tag></item>
            <item>twelve<tag>out=12;</tag></item>
            <item>thirteen<tag>out=13;</tag></item>
            <item>fourteen<tag>out=14;</tag></item>
            <item>fifteen<tag>out=15;</tag></item>
            <item>sixteen<tag>out=16;</tag></item>
            <item>seventeen<tag>out=17;</tag></item>
            <item>eighteen<tag>out=18;</tag></item>
            <item>nineteen<tag>out=19;</tag></item>
            <item>tenth<tag>out=10;</tag></item>
            <item>eleventh<tag>out=11;</tag></item>
            <item>twelveth<tag>out=12;</tag></item>
            <item>thirteenth<tag>out=13;</tag></item>
            <item>fourteenth<tag>out=14;</tag></item>
            <item>fifteenth<tag>out=15;</tag></item>
            <item>sixteenth<tag>out=16;</tag></item>
            <item>seventeenth<tag>out=17;</tag></item>
            <item>eighteenth<tag>out=18;</tag></item>
            <item>nineteenth<tag>out=19;</tag></item>
        </one-of>
    </rule>
```

```
        <rule id="above_twenty">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <one-of>
                <item>twenty<tag>out=20;</tag></item>
                <item>thirty<tag>out=30;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>
</grammar>
```

Jumlah transfer

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: I want to transfer twenty five Dollar
                 Output: $25

             Scenario 2:
                 Input: transfer twenty five dollars
                 Output: $25

         -->

        <rule id="main" scope="public">
            <tag>out="$"</tag>
            <one-of>
                <item><ruleref uri="#sub_hundred"/><tag>out += rules.sub_hundred.sh;</
tag></item>
```

```
                <item><ruleref uri="#subThousands"/><tag>out += rules.subThousands;</
tag></item>
            </one-of>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
           <one-of>
              <item repeat="0-1">I want to transfer</item>
              <item repeat="0-1">transfer</item>
              <item repeat="0-1">make a transfer for</item>
           </one-of>
        </rule>

        <rule id="hesitation">
            <one-of>
               <item>Hmm</item>
               <item>Mmm</item>
               <item>My</item>
            </one-of>
          </rule>

        <rule id="digits">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.num = 0;</tag>
            <one-of>
                <item>0<tag>out.num+=0;</tag></item>
                <item>1<tag>out.num+=1;</tag></item>
                <item>2<tag>out.num+=2;</tag></item>
                <item>3<tag>out.num+=3;</tag></item>
                <item>4<tag>out.num+=4;</tag></item>
                <item>5<tag>out.num+=5;</tag></item>
                <item>6<tag>out.num+=6;</tag></item>
                <item>7<tag>out.num+=7;</tag></item>
                <item>8<tag>out.num+=8;</tag></item>
                <item>9<tag>out.num+=9;</tag></item>
                <item>one<tag>out.num+=1;</tag></item>
                <item>two<tag>out.num+=2;</tag></item>
                <item>three<tag>out.num+=3;</tag></item>
                <item>four<tag>out.num+=4;</tag></item>
                <item>five<tag>out.num+=5;</tag></item>
                <item>six<tag>out.num+=6;</tag></item>
                <item>seven<tag>out.num+=7;</tag></item>
                <item>eight<tag>out.num+=8;</tag></item>
```

```
                <item>nine<tag>out.num+=9;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#currency"/></item>
        </rule>

        <rule id="teens">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.teen = 0;</tag>
            <one-of>
                <item>ten<tag>out.teen+=10;</tag></item>
                <item>eleven<tag>out.teen+=11;</tag></item>
                <item>twelve<tag>out.teen+=12;</tag></item>
                <item>thirteen<tag>out.teen+=13;</tag></item>
                <item>fourteen<tag>out.teen+=14;</tag></item>
                <item>fifteen<tag>out.teen+=15;</tag></item>
                <item>sixteen<tag>out.teen+=16;</tag></item>
                <item>seventeen<tag>out.teen+=17;</tag></item>
                <item>eighteen<tag>out.teen+=18;</tag></item>
                <item>nineteen<tag>out.teen+=19;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#currency"/></item>
        </rule>

        <rule id="above_twenty">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.tens = 0;</tag>
            <one-of>
                <item>twenty<tag>out.tens+=20;</tag></item>
                <item>thirty<tag>out.tens+=30;</tag></item>
                <item>forty<tag>out.tens+=40;</tag></item>
                <item>fifty<tag>out.tens+=50;</tag></item>
                <item>sixty<tag>out.tens+=60;</tag></item>
                <item>seventy<tag>out.tens+=70;</tag></item>
                <item>eighty<tag>out.tens+=80;</tag></item>
                <item>ninety<tag>out.tens+=90;</tag></item>
                <item>hundred<tag>out.tens+=100;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#currency"/></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out.tens +=
rules.digits.num;</tag></item>
        </rule>

        <rule id="currency">
            <one-of>
```

```
                <item repeat="0-1">dollars</item>
                <item repeat="0-1">Dollars</item>
                <item repeat="0-1">dollar</item>
                <item repeat="0-1">Dollar</item>
            </one-of>
        </rule>


        <rule id="sub_hundred">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.sh = 0;</tag>
            <one-of>
                <item><ruleref uri="#teens"/><tag>out.sh += rules.teens.teen;</tag></
item>
                <item>
                    <ruleref uri="#above_twenty"/><tag>out.sh +=
 rules.above_twenty.tens;</tag>
                </item>
                <item><ruleref uri="#digits"/><tag>out.sh += rules.digits.num;</tag></
item>
            </one-of>
         </rule>

        <rule id="subThousands">
            <ruleref uri="#sub_hundred"/><tag>out = (100 * rules.sub_hundred.sh);</tag>
            hundred
            <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out +=
 rules.above_twenty.tens;</tag></item>
            <item repeat="0-1"><ruleref uri="#teens"/><tag>out += rules.teens.teen;</
tag></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits.num;</
tag></item>
            <item repeat="0-1"><ruleref uri="#currency"/></item>
        </rule>
</grammar>
```

Nomor Jaminan Soal

```
<?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>
```

Tata bahasa untuk asuransi (download)

Tata bahasa berikut didukung untuk domain asuransi: nomor klaim dan polis, nomor SIM dan plat nomor, tanggal kedaluwarsa, tanggal mulai dan tanggal perpanjangan, klaim dan jumlah polis.

ID Klaim

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="digits"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: My claim number is One Five Four Two
                 Output: 1542

             Scenario 2:
                 Input: Claim number One Five Four Four
                 Output: 1544

         -->

        <rule id="digits">
            <tag>out=""</tag>
            <item><ruleref uri="#singleDigit"/><tag>out += rules.singleDigit.digit;</
tag></item>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
            <one-of>
                <item repeat="0-1">My claim number is</item>
                <item repeat="0-1">Claim number</item>
                <item repeat="0-1">This is for claim</item>
            </one-of>
```

```
        </rule>

        <rule id="hesitation">
            <one-of>
               <item>Hmm</item>
               <item>Mmm</item>
               <item>My</item>
            </one-of>
          </rule>

        <rule id="singleDigit">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.digit=""</tag>
            <item repeat="1-10">
                <one-of>
                    <item>0<tag>out.digit+=0;</tag></item>
                    <item>zero<tag>out.digit+=0;</tag></item>
                    <item>1<tag>out.digit+=1;</tag></item>
                    <item>one<tag>out.digit+=1;</tag></item>
                    <item>2<tag>out.digit+=2;</tag></item>
                    <item>two<tag>out.digit+=2;</tag></item>
                    <item>3<tag>out.digit+=3;</tag></item>
                    <item>three<tag>out.digit+=3;</tag></item>
                    <item>4<tag>out.digit+=4;</tag></item>
                    <item>four<tag>out.digit+=4;</tag></item>
                    <item>5<tag>out.digit+=5;</tag></item>
                    <item>five<tag>out.digit+=5;</tag></item>
                    <item>6<tag>out.digit+=6;</tag></item>
                    <item>six<tag>out.digit+=5;</tag></item>
                    <item>7<tag>out.digit+=7;</tag></item>
                    <item>seven<tag>out.digit+=7;</tag></item>
                    <item>8<tag>out.digit+=8;</tag></item>
                    <item>eight<tag>out.digit+=8;</tag></item>
                    <item>9<tag>out.digit+=9;</tag></item>
                    <item>nine<tag>out.digit+=9;</tag></item>
                </one-of>
            </item>
        </rule>
</grammar>
```

ID Kebijakan

```
<?xml version="1.0" encoding="UTF-8" ?>
```

```
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">


        <!-- Test Cases

        Grammar will support the following inputs:

            Scenario 1:
                Input: My policy number is A B C 1 2 3 4
                Output: ABC1234

            Scenario 2:
                Input: This is the policy number 1 2 3 4 A B C
                Output: 1234ABC

            Scenario 3:
                Input: Hmm My policy number is 1 2 3 4 A B C 1
                Output: 123ABC1
        -->

        <rule id="main" scope="public">
            <tag>out=""</tag>
            <item><ruleref uri="#alphanumeric"/><tag>out +=
 rules.alphanumeric.alphanum;</tag></item>
            <item repeat="0-1"><ruleref uri="#alphabets"/><tag>out +=
 rules.alphabets.letters;</tag></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out +=
 rules.digits.numbers</tag></item>
            <item repeat="0-1"><ruleref uri="#thanks"/></item>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
            <one-of>
                <item repeat="0-1">My policy number is</item>
                <item repeat="0-1">This is the policy number</item>
                <item repeat="0-1">Policy number</item>
```

```
                <item repeat="0-1">Yes, It is</item>
                <item repeat="0-1">Yes It is</item>
                <item repeat="0-1">Yes It's</item>
                <item repeat="0-1">My policy Id is</item>
                <item repeat="0-1">This is the policy Id</item>
                <item repeat="0-1">Policy Id</item>
            </one-of>
        </rule>

        <rule id="hesitation">
          <one-of>
             <item>Hmm</item>
             <item>Mmm</item>
             <item>My</item>
          </one-of>
        </rule>

        <rule id="thanks">
            <one-of>
               <item>Thanks</item>
               <item>I think</item>
            </one-of>
          </rule>

        <rule id="alphanumeric" scope="public">
            <tag>out.alphanum=""</tag>
            <item><ruleref uri="#alphabets"/><tag>out.alphanum +=
rules.alphabets.letters;</tag></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out.alphanum +=
rules.digits.numbers</tag></item>
        </rule>

        <rule id="alphabets">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.letters=""</tag>
            <tag>out.firstOccurence=""</tag>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out.firstOccurence +=
rules.digits.numbers; out.letters += out.firstOccurence;</tag></item>
            <item repeat="1-">
                <one-of>
                    <item>A<tag>out.letters+='A';</tag></item>
                    <item>B<tag>out.letters+='B';</tag></item>
                    <item>C<tag>out.letters+='C';</tag></item>
                    <item>D<tag>out.letters+='D';</tag></item>
```

```
                <item>E<tag>out.letters+='E';</tag></item>
                <item>F<tag>out.letters+='F';</tag></item>
                <item>G<tag>out.letters+='G';</tag></item>
                <item>H<tag>out.letters+='H';</tag></item>
                <item>I<tag>out.letters+='I';</tag></item>
                <item>J<tag>out.letters+='J';</tag></item>
                <item>K<tag>out.letters+='K';</tag></item>
                <item>L<tag>out.letters+='L';</tag></item>
                <item>M<tag>out.letters+='M';</tag></item>
                <item>N<tag>out.letters+='N';</tag></item>
                <item>O<tag>out.letters+='O';</tag></item>
                <item>P<tag>out.letters+='P';</tag></item>
                <item>Q<tag>out.letters+='Q';</tag></item>
                <item>R<tag>out.letters+='R';</tag></item>
                <item>S<tag>out.letters+='S';</tag></item>
                <item>T<tag>out.letters+='T';</tag></item>
                <item>U<tag>out.letters+='U';</tag></item>
                <item>V<tag>out.letters+='V';</tag></item>
                <item>W<tag>out.letters+='W';</tag></item>
                <item>X<tag>out.letters+='X';</tag></item>
                <item>Y<tag>out.letters+='Y';</tag></item>
                <item>Z<tag>out.letters+='Z';</tag></item>
            </one-of>
        </item>
    </rule>

    <rule id="digits">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <tag>out.numbers=""</tag>
        <item repeat="1-10">
            <one-of>
                <item>0<tag>out.numbers+=0;</tag></item>
                <item>1<tag>out.numbers+=1;</tag></item>
                <item>2<tag>out.numbers+=2;</tag></item>
                <item>3<tag>out.numbers+=3;</tag></item>
                <item>4<tag>out.numbers+=4;</tag></item>
                <item>5<tag>out.numbers+=5;</tag></item>
                <item>6<tag>out.numbers+=6;</tag></item>
                <item>7<tag>out.numbers+=7;</tag></item>
                <item>8<tag>out.numbers+=8;</tag></item>
                <item>9<tag>out.numbers+=9;</tag></item>
            </one-of>
        </item>
    </rule>
```

```
</grammar>
```

Nomor SIM

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="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>
```

Nomor plat

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">


        <!-- Test Cases

        Grammar will support the following inputs:

            Scenario 1:
                Input: my license plate is A B C D 1 2
                Output: ABCD12

            Scenario 2:
                Input: license plate number A B C 1 2 3 4
                Output: ABC1234

            Scenario 3:
                Input: my plates say A F G K 9 8 7 6 Thanks
                Output: AFGK9876
        -->

        <rule id="main" scope="public">
            <tag>out.licenseNum=""</tag>
            <item><ruleref uri="#alphabets"/><tag>out.licenseNum +=
rules.alphabets.letters;</tag></item>
            <item repeat="0-1"><ruleref uri="#thanks"/></item>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
            <one-of>
                <item repeat="0-1">my license plate is</item>
                <item repeat="0-1">license plate number</item>
                <item repeat="0-1">my plates say</item>
            </one-of>
        </rule>

        <rule id="hesitation">
          <one-of>
             <item>Hmm</item>
```

```
            <item>Mmm</item>
            <item>My</item>
          </one-of>
        </rule>

        <rule id="thanks">
            <one-of>
               <item>Thanks</item>
               <item>I think</item>
            </one-of>
          </rule>

        <rule id="alphabets">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.letters=""</tag>
            <tag>out.firstOccurence=""</tag>
            <item repeat="3-4">
                <one-of>
                    <item>A<tag>out.letters+='A';</tag></item>
                    <item>B<tag>out.letters+='B';</tag></item>
                    <item>C<tag>out.letters+='C';</tag></item>
                    <item>D<tag>out.letters+='D';</tag></item>
                    <item>E<tag>out.letters+='E';</tag></item>
                    <item>F<tag>out.letters+='F';</tag></item>
                    <item>G<tag>out.letters+='G';</tag></item>
                    <item>H<tag>out.letters+='H';</tag></item>
                    <item>I<tag>out.letters+='I';</tag></item>
                    <item>J<tag>out.letters+='J';</tag></item>
                    <item>K<tag>out.letters+='K';</tag></item>
                    <item>L<tag>out.letters+='L';</tag></item>
                    <item>M<tag>out.letters+='M';</tag></item>
                    <item>N<tag>out.letters+='N';</tag></item>
                    <item>O<tag>out.letters+='O';</tag></item>
                    <item>P<tag>out.letters+='P';</tag></item>
                    <item>Q<tag>out.letters+='Q';</tag></item>
                    <item>R<tag>out.letters+='R';</tag></item>
                    <item>S<tag>out.letters+='S';</tag></item>
                    <item>T<tag>out.letters+='T';</tag></item>
                    <item>U<tag>out.letters+='U';</tag></item>
                    <item>V<tag>out.letters+='V';</tag></item>
                    <item>W<tag>out.letters+='W';</tag></item>
                    <item>X<tag>out.letters+='X';</tag></item>
                    <item>Y<tag>out.letters+='Y';</tag></item>
                    <item>Z<tag>out.letters+='Z';</tag></item>
```

```
                </one-of>
            </item>
     <item repeat="0-1"><ruleref uri="#digits"/><tag>out.firstOccurence +=
 rules.digits.numbers; out.letters += out.firstOccurence;</tag></item>
        </rule>

        <rule id="digits">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.numbers=""</tag>
            <item repeat="2-4">
                <one-of>
                    <item>0<tag>out.numbers+=0;</tag></item>
                    <item>1<tag>out.numbers+=1;</tag></item>
                    <item>2<tag>out.numbers+=2;</tag></item>
                    <item>3<tag>out.numbers+=3;</tag></item>
                    <item>4<tag>out.numbers+=4;</tag></item>
                    <item>5<tag>out.numbers+=5;</tag></item>
                    <item>6<tag>out.numbers+=6;</tag></item>
                    <item>7<tag>out.numbers+=7;</tag></item>
                    <item>8<tag>out.numbers+=8;</tag></item>
                    <item>9<tag>out.numbers+=9;</tag></item>
                </one-of>
            </item>
        </rule>
</grammar>
```

Tanggal kedaluwarsa kartu kredit

```
<?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>
```

Tanggal kedaluwarsa kebijakan, day/month /tahun

```
<?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>
```

Tanggal perpanjangan kebijakan, month/year

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: I renewed my policy on July Two Thousand and Eleven
                 Output: 07/11

             Scenario 2:
                 Input: My policy will renew on July Two Thousand and Eleven
                 Output: 07/11

         -->

        <rule id="main" scope="public">
            <tag>out=""</tag>
            <item repeat="1-10">
                <item repeat="1"><ruleref uri="#months"/><tag>out = out +
 rules.months.mon + "/";</tag></item>
                <one-of>
                    <item><ruleref uri="#thousands"/><tag>out += rules.thousands;</
tag></item>
                    <item repeat="0-1"><ruleref uri="#digits"/><tag>out +=
 rules.digits;</tag></item>
                    <item repeat="0-1"><ruleref uri="#teens"/><tag>out +=
 rules.teens;</tag></item>
```

```
                    <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out +=
rules.above_twenty;</tag></item>
                </one-of>
            </item>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
            <one-of>
                <item repeat="0-1">My policy will renew on</item>
                <item repeat="0-1">My policy was renewed on</item>
                <item repeat="0-1">Renew policy on</item>
                <item repeat="0-1">I renewed my policy on</item>
            </one-of>
        </rule>

        <rule id="hesitation">
            <one-of>
               <item>Hmm</item>
               <item>Mmm</item>
               <item>My</item>
            </one-of>
          </rule>

        <rule id="months">
           <item repeat="0-1"><ruleref uri="#text"/></item>
           <tag>out.mon=""</tag>
           <one-of>
               <item>january<tag>out.mon+="01";</tag></item>
               <item>february<tag>out.mon+="02";</tag></item>
               <item>march<tag>out.mon+="03";</tag></item>
               <item>april<tag>out.mon+="04";</tag></item>
               <item>may<tag>out.mon+="05";</tag></item>
               <item>june<tag>out.mon+="06";</tag></item>
               <item>july<tag>out.mon+="07";</tag></item>
               <item>august<tag>out.mon+="08";</tag></item>
               <item>september<tag>out.mon+="09";</tag></item>
               <item>october<tag>out.mon+="10";</tag></item>
               <item>november<tag>out.mon+="11";</tag></item>
               <item>december<tag>out.mon+="12";</tag></item>
               <item>jan<tag>out.mon+="01";</tag></item>
               <item>feb<tag>out.mon+="02";</tag></item>
               <item>aug<tag>out.mon+="08";</tag></item>
               <item>sept<tag>out.mon+="09";</tag></item>
```

```
            <item>oct<tag>out.mon+="10";</tag></item>
            <item>nov<tag>out.mon+="11";</tag></item>
            <item>dec<tag>out.mon+="12";</tag></item>
         </one-of>
      </rule>

      <rule id="digits">
          <one-of>
              <item>zero<tag>out=0;</tag></item>
              <item>one<tag>out=1;</tag></item>
              <item>two<tag>out=2;</tag></item>
              <item>three<tag>out=3;</tag></item>
              <item>four<tag>out=4;</tag></item>
              <item>five<tag>out=5;</tag></item>
              <item>six<tag>out=6;</tag></item>
              <item>seven<tag>out=7;</tag></item>
              <item>eight<tag>out=8;</tag></item>
              <item>nine<tag>out=9;</tag></item>
          </one-of>
      </rule>

      <rule id="teens">
          <one-of>
              <item>ten<tag>out=10;</tag></item>
              <item>eleven<tag>out=11;</tag></item>
              <item>twelve<tag>out=12;</tag></item>
              <item>thirteen<tag>out=13;</tag></item>
              <item>fourteen<tag>out=14;</tag></item>
              <item>fifteen<tag>out=15;</tag></item>
              <item>sixteen<tag>out=16;</tag></item>
              <item>seventeen<tag>out=17;</tag></item>
              <item>eighteen<tag>out=18;</tag></item>
              <item>nineteen<tag>out=19;</tag></item>
          </one-of>
      </rule>

      <rule id="thousands">
          <item>two thousand<!--<tag>out=2000;</tag>--></item>
          <item repeat="0-1">and</item>
          <item repeat="0-1"><ruleref uri="#digits"/><tag>out = rules.digits;</tag></
item>
          <item repeat="0-1"><ruleref uri="#teens"/><tag>out = rules.teens;</tag></
item>
```

```
            <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out =
 rules.above_twenty;</tag></item>
        </rule>

        <rule id="above_twenty">
            <one-of>
                <item>twenty<tag>out=20;</tag></item>
                <item>thirty<tag>out=30;</tag></item>
                <item>forty<tag>out=40;</tag></item>
                <item>fifty<tag>out=50;</tag></item>
                <item>sixty<tag>out=60;</tag></item>
                <item>seventy<tag>out=70;</tag></item>
                <item>eighty<tag>out=80;</tag></item>
                <item>ninety<tag>out=90;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>
</grammar>
```

Tanggal mulai kebijakan

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: I bought my policy on july twenty three
                 Output: 07/23

             Scenario 2:
                 Input: My policy started on july fifteen
                 Output: 07/15
```

```
         -->

        <rule id="main" scope="public">
            <tag>out=""</tag>
            <item repeat="1-10">
                <item><ruleref uri="#months"/><tag>out= rules.months.mon + "/";</tag></
item>
                <one-of>
                    <item><ruleref uri="#digits"/><tag>out+= rules.digits;</tag></item>
                    <item><ruleref uri="#teens"/><tag>out+= rules.teens;</tag></item>
                    <item><ruleref uri="#above_twenty"/><tag>out+=
 rules.above_twenty;</tag></item>
                </one-of>
            </item>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
             <one-of>
                <item repeat="0-1">I bought my policy on</item>
                <item repeat="0-1">I bought policy on</item>
                <item repeat="0-1">My policy started on</item>
             </one-of>
          </rule>

          <rule id="hesitation">
             <one-of>
                <item>Hmm</item>
                <item>Mmm</item>
                <item>My</item>
             </one-of>
           </rule>

        <rule id="months">
           <item repeat="0-1"><ruleref uri="#text"/></item>
           <tag>out.mon=""</tag>
           <one-of>
               <item>january<tag>out.mon+="01";</tag></item>
               <item>february<tag>out.mon+="02";</tag></item>
               <item>march<tag>out.mon+="03";</tag></item>
               <item>april<tag>out.mon+="04";</tag></item>
               <item>may<tag>out.mon+="05";</tag></item>
               <item>june<tag>out.mon+="06";</tag></item>
```

```
            <item>july<tag>out.mon+="07";</tag></item>
            <item>august<tag>out.mon+="08";</tag></item>
            <item>september<tag>out.mon+="09";</tag></item>
            <item>october<tag>out.mon+="10";</tag></item>
            <item>november<tag>out.mon+="11";</tag></item>
            <item>december<tag>out.mon+="12";</tag></item>
            <item>jan<tag>out.mon+="01";</tag></item>
            <item>feb<tag>out.mon+="02";</tag></item>
            <item>aug<tag>out.mon+="08";</tag></item>
            <item>sept<tag>out.mon+="09";</tag></item>
            <item>oct<tag>out.mon+="10";</tag></item>
            <item>nov<tag>out.mon+="11";</tag></item>
            <item>dec<tag>out.mon+="12";</tag></item>
        </one-of>
    </rule>

    <rule id="digits">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <one-of>
            <item>0<tag>out=0;</tag></item>
            <item>1<tag>out=1;</tag></item>
            <item>2<tag>out=2;</tag></item>
            <item>3<tag>out=3;</tag></item>
            <item>4<tag>out=4;</tag></item>
            <item>5<tag>out=5;</tag></item>
            <item>6<tag>out=6;</tag></item>
            <item>7<tag>out=7;</tag></item>
            <item>8<tag>out=8;</tag></item>
            <item>9<tag>out=9;</tag></item>
            <item>first<tag>out=01;</tag></item>
            <item>second<tag>out=02;</tag></item>
            <item>third<tag>out=03;</tag></item>
            <item>fourth<tag>out=04;</tag></item>
            <item>fifth<tag>out=05;</tag></item>
            <item>sixth<tag>out=06;</tag></item>
            <item>seventh<tag>out=07;</tag></item>
            <item>eighth<tag>out=08;</tag></item>
            <item>ninth<tag>out=09;</tag></item>
            <item>one<tag>out=1;</tag></item>
            <item>two<tag>out=2;</tag></item>
            <item>three<tag>out=3;</tag></item>
            <item>four<tag>out=4;</tag></item>
            <item>five<tag>out=5;</tag></item>
            <item>six<tag>out=6;</tag></item>
```

```
                <item>seven<tag>out=7;</tag></item>
                <item>eight<tag>out=8;</tag></item>
                <item>nine<tag>out=9;</tag></item>
            </one-of>
        </rule>

        <rule id="teens">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <one-of>
                <item>ten<tag>out=10;</tag></item>
                <item>tenth<tag>out=10;</tag></item>
                <item>eleven<tag>out=11;</tag></item>
                <item>twelve<tag>out=12;</tag></item>
                <item>thirteen<tag>out=13;</tag></item>
                <item>fourteen<tag>out=14;</tag></item>
                <item>fifteen<tag>out=15;</tag></item>
                <item>sixteen<tag>out=16;</tag></item>
                <item>seventeen<tag>out=17;</tag></item>
                <item>eighteen<tag>out=18;</tag></item>
                <item>nineteen<tag>out=19;</tag></item>
                <item>tenth<tag>out=10;</tag></item>
                <item>eleventh<tag>out=11;</tag></item>
                <item>twelveth<tag>out=12;</tag></item>
                <item>thirteenth<tag>out=13;</tag></item>
                <item>fourteenth<tag>out=14;</tag></item>
                <item>fifteenth<tag>out=15;</tag></item>
                <item>sixteenth<tag>out=16;</tag></item>
                <item>seventeenth<tag>out=17;</tag></item>
                <item>eighteenth<tag>out=18;</tag></item>
                <item>nineteenth<tag>out=19;</tag></item>
            </one-of>
        </rule>

        <rule id="above_twenty">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <one-of>
                <item>twenty<tag>out=20;</tag></item>
                <item>thirty<tag>out=30;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>
</grammar>
```

Jumlah klaim

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: I want to make a claim of one hundre ten dollars
                 Output: $110

             Scenario 2:
                 Input: Requesting claim of Two hundred dollars
                 Output: $200

         -->

        <rule id="main" scope="public">
            <tag>out="$"</tag>
            <one-of>
                <item><ruleref uri="#sub_hundred"/><tag>out += rules.sub_hundred.sh;</
tag></item>
                <item><ruleref uri="#subThousands"/><tag>out += rules.subThousands;</
tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#thanks"/></item>
        </rule>

        <rule id="text">
           <item repeat="0-1"><ruleref uri="#hesitation"/></item>
           <one-of>
              <item repeat="0-1">I want to place a claim for</item>
              <item repeat="0-1">I want to make a claim of</item>
              <item repeat="0-1">I assess damage of</item>
              <item repeat="0-1">Requesting claim of</item>
```

```
        </one-of>
    </rule>

    <rule id="hesitation">
        <one-of>
            <item>Hmm</item>
            <item>Mmm</item>
            <item>My</item>
        </one-of>
      </rule>

      <rule id="thanks">
          <one-of>
              <item>Thanks</item>
              <item>I think</item>
          </one-of>
        </rule>

    <rule id="digits">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <tag>out.num = 0;</tag>
        <one-of>
            <item>0<tag>out.num+=0;</tag></item>
            <item>1<tag>out.num+=1;</tag></item>
            <item>2<tag>out.num+=2;</tag></item>
            <item>3<tag>out.num+=3;</tag></item>
            <item>4<tag>out.num+=4;</tag></item>
            <item>5<tag>out.num+=5;</tag></item>
            <item>6<tag>out.num+=6;</tag></item>
            <item>7<tag>out.num+=7;</tag></item>
            <item>8<tag>out.num+=8;</tag></item>
            <item>9<tag>out.num+=9;</tag></item>
            <item>one<tag>out.num+=1;</tag></item>
            <item>two<tag>out.num+=2;</tag></item>
            <item>three<tag>out.num+=3;</tag></item>
            <item>four<tag>out.num+=4;</tag></item>
            <item>five<tag>out.num+=5;</tag></item>
            <item>six<tag>out.num+=6;</tag></item>
            <item>seven<tag>out.num+=7;</tag></item>
            <item>eight<tag>out.num+=8;</tag></item>
            <item>nine<tag>out.num+=9;</tag></item>
        </one-of>
        <item repeat="0-1"><ruleref uri="#currency"/></item>
    </rule>
```

```
        <rule id="teens">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.teen = 0;</tag>
            <one-of>
                <item>ten<tag>out.teen+=10;</tag></item>
                <item>eleven<tag>out.teen+=11;</tag></item>
                <item>twelve<tag>out.teen+=12;</tag></item>
                <item>thirteen<tag>out.teen+=13;</tag></item>
                <item>fourteen<tag>out.teen+=14;</tag></item>
                <item>fifteen<tag>out.teen+=15;</tag></item>
                <item>sixteen<tag>out.teen+=16;</tag></item>
                <item>seventeen<tag>out.teen+=17;</tag></item>
                <item>eighteen<tag>out.teen+=18;</tag></item>
                <item>nineteen<tag>out.teen+=19;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#currency"/></item>
        </rule>

        <rule id="above_twenty">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.tens = 0;</tag>
            <one-of>
                <item>twenty<tag>out.tens+=20;</tag></item>
                <item>thirty<tag>out.tens+=30;</tag></item>
                <item>forty<tag>out.tens+=40;</tag></item>
                <item>fifty<tag>out.tens+=50;</tag></item>
                <item>sixty<tag>out.tens+=60;</tag></item>
                <item>seventy<tag>out.tens+=70;</tag></item>
                <item>eighty<tag>out.tens+=80;</tag></item>
                <item>ninety<tag>out.tens+=90;</tag></item>
                <item>hundred<tag>out.tens+=100;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#currency"/></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out.tens +=
rules.digits.num;</tag></item>
        </rule>

        <rule id="currency">
            <one-of>
                <item repeat="0-1">dollars</item>
                <item repeat="0-1">Dollars</item>
                <item repeat="0-1">dollar</item>
                <item repeat="0-1">Dollar</item>
```

```
            </one-of>
        </rule>


        <rule id="sub_hundred">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.sh = 0;</tag>
            <one-of>
                <item><ruleref uri="#teens"/><tag>out.sh += rules.teens.teen;</tag></
item>
                <item>
                    <ruleref uri="#above_twenty"/><tag>out.sh +=
 rules.above_twenty.tens;</tag>
                </item>
                <item><ruleref uri="#digits"/><tag>out.sh += rules.digits.num;</tag></
item>
            </one-of>
         </rule>

        <rule id="subThousands">
            <ruleref uri="#sub_hundred"/><tag>out = (100 * rules.sub_hundred.sh);</tag>
            hundred
            <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out +=
 rules.above_twenty.tens;</tag></item>
            <item repeat="0-1"><ruleref uri="#teens"/><tag>out += rules.teens.teen;</
tag></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits.num;</
tag></item>
            <item repeat="0-1"><ruleref uri="#currency"/></item>
        </rule>
</grammar>
```

Jumlah premi

```
<?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>
```

Kuantitas kebijakan

```
<?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>
```

Tata bahasa untuk telekomunikasi (download)

Tata bahasa berikut didukung untuk telekomunikasi: Nomor telepon, nomor seri, nomor SIM, kode pos AS, tanggal kedaluwarsa kartu kredit, rencana mulai, tanggal perpanjangan dan kedaluwarsa, tanggal mulai layanan, jumlah peralatan dan jumlah tagihan.

Nomor telepon

```
<?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>
```

Nomor seri

```
<?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>
```

Nomor SIM

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">


        <!-- Test Cases

        Grammar will support the following inputs:

            Scenario 1:
                Input: My SIM number is A B C 1 2 3 4
                Output: ABC1234

            Scenario 2:
                Input: My SIM number is 1 2 3 4 A B C
```

```
                Output: 1234ABC

            Scenario 3:
                Input: My SIM number is 1 2 3 4 A B C 1
                Output: 123ABC1
        -->

        <rule id="main" scope="public">
            <tag>out=""</tag>
            <item><ruleref uri="#alphanumeric"/><tag>out +=
rules.alphanumeric.alphanum;</tag></item>
            <item repeat="0-1"><ruleref uri="#alphabets"/><tag>out +=
rules.alphabets.letters;</tag></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out +=
rules.digits.numbers</tag></item>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
            <one-of>
                <item repeat="0-1">My SIM number is</item>
                <item repeat="0-1">SIM number is</item>
            </one-of>
        </rule>

        <rule id="hesitation">
          <one-of>
             <item>Hmm</item>
             <item>Mmm</item>
             <item>My</item>
          </one-of>
        </rule>

        <rule id="alphanumeric" scope="public">
            <tag>out.alphanum=""</tag>
            <item><ruleref uri="#alphabets"/><tag>out.alphanum +=
rules.alphabets.letters;</tag></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out.alphanum +=
rules.digits.numbers</tag></item>
        </rule>

        <rule id="alphabets">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.letters=""</tag>
```

```
            <tag>out.firstOccurence=""</tag>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out.firstOccurence +=
 rules.digits.numbers; out.letters += out.firstOccurence;</tag></item>
            <item repeat="1-">
                <one-of>
                    <item>A<tag>out.letters+='A';</tag></item>
                    <item>B<tag>out.letters+='B';</tag></item>
                    <item>C<tag>out.letters+='C';</tag></item>
                    <item>D<tag>out.letters+='D';</tag></item>
                    <item>E<tag>out.letters+='E';</tag></item>
                    <item>F<tag>out.letters+='F';</tag></item>
                    <item>G<tag>out.letters+='G';</tag></item>
                    <item>H<tag>out.letters+='H';</tag></item>
                    <item>I<tag>out.letters+='I';</tag></item>
                    <item>J<tag>out.letters+='J';</tag></item>
                    <item>K<tag>out.letters+='K';</tag></item>
                    <item>L<tag>out.letters+='L';</tag></item>
                    <item>M<tag>out.letters+='M';</tag></item>
                    <item>N<tag>out.letters+='N';</tag></item>
                    <item>O<tag>out.letters+='O';</tag></item>
                    <item>P<tag>out.letters+='P';</tag></item>
                    <item>Q<tag>out.letters+='Q';</tag></item>
                    <item>R<tag>out.letters+='R';</tag></item>
                    <item>S<tag>out.letters+='S';</tag></item>
                    <item>T<tag>out.letters+='T';</tag></item>
                    <item>U<tag>out.letters+='U';</tag></item>
                    <item>V<tag>out.letters+='V';</tag></item>
                    <item>W<tag>out.letters+='W';</tag></item>
                    <item>X<tag>out.letters+='X';</tag></item>
                    <item>Y<tag>out.letters+='Y';</tag></item>
                    <item>Z<tag>out.letters+='Z';</tag></item>
                </one-of>
            </item>
        </rule>

        <rule id="digits">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.numbers=""</tag>
            <item repeat="1-10">
                <one-of>
                    <item>0<tag>out.numbers+=0;</tag></item>
                    <item>1<tag>out.numbers+=1;</tag></item>
                    <item>2<tag>out.numbers+=2;</tag></item>
                    <item>3<tag>out.numbers+=3;</tag></item>
```

```
                    <item>4<tag>out.numbers+=4;</tag></item>
                    <item>5<tag>out.numbers+=5;</tag></item>
                    <item>6<tag>out.numbers+=6;</tag></item>
                    <item>7<tag>out.numbers+=7;</tag></item>
                    <item>8<tag>out.numbers+=8;</tag></item>
                    <item>9<tag>out.numbers+=9;</tag></item>
                </one-of>
            </item>
        </rule>
</grammar>
```

Kode pos AS

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="digits"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support 5 digits code and here are couple of examples of valid
 inputs:

             Scenario 1:
                 Input: Mmmm My zipcode is umm One Oh Nine Eight Seven
                 Output: 10987

             Scenario 2:
                 Input: My zipcode is One Oh Nine Eight Seven
                 Output: 10987

         -->

        <rule id="digits">
            <tag>out=""</tag>
            <item><ruleref uri="#singleDigit"/><tag>out += rules.singleDigit.digit;</
tag></item>
        </rule>
```

```
<rule id="text">
    <item repeat="0-1"><ruleref uri="#hesitation"/></item>
   <one-of>
      <item repeat="0-1">My zipcode is</item>
      <item repeat="0-1">Zipcode is</item>
      <item repeat="0-1">It is</item>
   </one-of>
</rule>

<rule id="hesitation">
    <one-of>
       <item>Hmm</item>
       <item>Mmm</item>
       <item>My</item>
    </one-of>
  </rule>

<rule id="singleDigit">
    <item repeat="0-1"><ruleref uri="#text"/></item>
    <tag>out.digit=""</tag>
    <item repeat="5">
        <one-of>
            <item>0<tag>out.digit+=0;</tag></item>
            <item>zero<tag>out.digit+=0;</tag></item>
            <item>Oh<tag>out.digit+=0;</tag></item>
            <item>1<tag>out.digit+=1;</tag></item>
            <item>one<tag>out.digit+=1;</tag></item>
            <item>2<tag>out.digit+=2;</tag></item>
            <item>two<tag>out.digit+=2;</tag></item>
            <item>3<tag>out.digit+=3;</tag></item>
            <item>three<tag>out.digit+=3;</tag></item>
            <item>4<tag>out.digit+=4;</tag></item>
            <item>four<tag>out.digit+=4;</tag></item>
            <item>5<tag>out.digit+=5;</tag></item>
            <item>five<tag>out.digit+=5;</tag></item>
            <item>6<tag>out.digit+=6;</tag></item>
            <item>six<tag>out.digit+=5;</tag></item>
            <item>7<tag>out.digit+=7;</tag></item>
            <item>seven<tag>out.digit+=7;</tag></item>
            <item>8<tag>out.digit+=8;</tag></item>
            <item>eight<tag>out.digit+=8;</tag></item>
            <item>9<tag>out.digit+=9;</tag></item>
            <item>nine<tag>out.digit+=9;</tag></item>
```

```
                </one-of>
            </item>
        </rule>
</grammar>
```

Tanggal kedaluwarsa kartu kredit

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="dateCardExpiration"
         mode="voice"
         tag-format="semantics/1.0">

        <rule id="dateCardExpiration" scope="public">
            <tag>out="";</tag>
            <item repeat="1"><ruleref uri="#months"/><tag>out = out + rules.months;</
tag></item>
            <item repeat="1"><ruleref uri="#year"/><tag>out += " " + rules.year.yr;</
tag></item>
        </rule>

        <!-- Test Cases

        Grammar will support the following inputs:

            Scenario 1:
                Input: My card expiration date is july eleven
                Output: 07 2011

            Scenario 2:
                Input: My card expiration date is may twenty six
                Output: 05 2026

        -->

        <rule id="text">
           <item repeat="0-1"><ruleref uri="#hesitation"/></item>
           <one-of>
              <item repeat="0-1">My card expiration date is </item>
```

```
    </one-of>
  </rule>

  <rule id="hesitation">
    <one-of>
      <item>Hmm</item>
      <item>Mmm</item>
      <item>My</item>
    </one-of>
  </rule>

  <rule id="months">
    <item repeat="0-1"><ruleref uri="#text"/></item>
    <one-of>
        <item>january<tag>out="01";</tag></item>
        <item>february<tag>out="02";</tag></item>
        <item>march<tag>out="03";</tag></item>
        <item>april<tag>out="04";</tag></item>
        <item>may<tag>out="05";</tag></item>
        <item>june<tag>out="06";</tag></item>
        <item>july<tag>out="07";</tag></item>
        <item>august<tag>out="08";</tag></item>
        <item>september<tag>out="09";</tag></item>
        <item>october<tag>out="10";</tag></item>
        <item>november<tag>out="11";</tag></item>
        <item>december<tag>out="12";</tag></item>
        <item>jan<tag>out="01";</tag></item>
        <item>feb<tag>out="02";</tag></item>
        <item>aug<tag>out="08";</tag></item>
        <item>sept<tag>out="09";</tag></item>
        <item>oct<tag>out="10";</tag></item>
        <item>nov<tag>out="11";</tag></item>
        <item>dec<tag>out="12";</tag></item>
        <item>1<tag>out="01";</tag></item>
        <item>2<tag>out="02";</tag></item>
        <item>3<tag>out="03";</tag></item>
        <item>4<tag>out="04";</tag></item>
        <item>5<tag>out="05";</tag></item>
        <item>6<tag>out="06";</tag></item>
        <item>7<tag>out="07";</tag></item>
        <item>8<tag>out="08";</tag></item>
        <item>9<tag>out="09";</tag></item>
        <item>ten<tag>out="10";</tag></item>
        <item>eleven<tag>out="11";</tag></item>
```

```
                <item>twelve<tag>out="12";</tag></item>
            </one-of>
        </rule>

        <rule id="digits">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <one-of>
                <item>0<tag>out=0;</tag></item>
                <item>1<tag>out=1;</tag></item>
                <item>2<tag>out=2;</tag></item>
                <item>3<tag>out=3;</tag></item>
                <item>4<tag>out=4;</tag></item>
                <item>5<tag>out=5;</tag></item>
                <item>6<tag>out=6;</tag></item>
                <item>7<tag>out=7;</tag></item>
                <item>8<tag>out=8;</tag></item>
                <item>9<tag>out=9;</tag></item>
                <item>one<tag>out=1;</tag></item>
                <item>two<tag>out=2;</tag></item>
                <item>three<tag>out=3;</tag></item>
                <item>four<tag>out=4;</tag></item>
                <item>five<tag>out=5;</tag></item>
                <item>six<tag>out=6;</tag></item>
                <item>seven<tag>out=7;</tag></item>
                <item>eight<tag>out=8;</tag></item>
                <item>nine<tag>out=9;</tag></item>
            </one-of>
        </rule>

        <rule id="year">
          <tag>out.yr="20"</tag>
          <one-of>
              <item><ruleref uri="#teens"/><tag>out.yr += rules.teens;</tag></item>
              <item><ruleref uri="#above_twenty"/><tag>out.yr += rules.above_twenty;</
tag></item>
          </one-of>
        </rule>

        <rule id="teens">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <one-of>
                <item>ten<tag>out=10;</tag></item>
                <item>eleven<tag>out=11;</tag></item>
                <item>twelve<tag>out=12;</tag></item>
```

```
                <item>thirteen<tag>out=13;</tag></item>
                <item>fourteen<tag>out=14;</tag></item>
                <item>fifteen<tag>out=15;</tag></item>
                <item>sixteen<tag>out=16;</tag></item>
                <item>seventeen<tag>out=17;</tag></item>
                <item>eighteen<tag>out=18;</tag></item>
                <item>nineteen<tag>out=19;</tag></item>
                <item>10<tag>out=10;</tag></item>
                <item>11<tag>out=11;</tag></item>
                <item>12<tag>out=12;</tag></item>
                <item>13<tag>out=13;</tag></item>
                <item>14<tag>out=14;</tag></item>
                <item>15<tag>out=15;</tag></item>
                <item>16<tag>out=16;</tag></item>
                <item>17<tag>out=17;</tag></item>
                <item>18<tag>out=18;</tag></item>
                <item>19<tag>out=19;</tag></item>
            </one-of>
        </rule>

        <rule id="above_twenty">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <one-of>
                <item>twenty<tag>out=20;</tag></item>
                <item>thirty<tag>out=30;</tag></item>
                <item>forty<tag>out=40;</tag></item>
                <item>fifty<tag>out=50;</tag></item>
                <item>sixty<tag>out=60;</tag></item>
                <item>seventy<tag>out=70;</tag></item>
                <item>eighty<tag>out=80;</tag></item>
                <item>ninety<tag>out=90;</tag></item>
                <item>20<tag>out=20;</tag></item>
                <item>30<tag>out=30;</tag></item>
                <item>40<tag>out=40;</tag></item>
                <item>50<tag>out=50;</tag></item>
                <item>60<tag>out=60;</tag></item>
                <item>70<tag>out=70;</tag></item>
                <item>80<tag>out=80;</tag></item>
                <item>90<tag>out=90;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>
```

```
</grammar>
```

Tanggal kedaluwarsa rencana, day/month /tahun

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: My plan expires on July Five Two Thousand and Eleven
                 Output: 07/5/11

             Scenario 2:
                 Input: My plan will expire on July Sixteen Two Thousand and Eleven
                 Output: 07/16/11

             Scenario 3:
                 Input: My plan will expire on July Thirty Two Thousand and Eleven
                 Output: 07/30/11
         -->

         <rule id="main" scope="public">
             <tag>out=""</tag>
             <item>
                 <item repeat="1"><ruleref uri="#months"/><tag>out = out +
 rules.months.mon + "/";</tag></item>
                 <one-of>
                   <item><ruleref uri="#digits"/><tag>out += rules.digits + "/";</
tag></item>
                   <item><ruleref uri="#teens"/><tag>out += rules.teens+ "/";</tag></
item>
                   <item><ruleref uri="#above_twenty"/><tag>out += rules.above_twenty+
 "/";</tag></item>
```

```
                </one-of>
                <one-of>
                    <item><ruleref uri="#thousands"/><tag>out += rules.thousands;</
tag></item>
                    <item repeat="0-1"><ruleref uri="#digits"/><tag>out +=
 rules.digits;</tag></item>
                    <item repeat="0-1"><ruleref uri="#teens"/><tag>out +=
 rules.teens;</tag></item>
                    <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out +=
 rules.above_twenty;</tag></item>
                </one-of>
             </item>
        </rule>

        <rule id="text">
          <item repeat="0-1"><ruleref uri="#hesitation"/></item>
          <one-of>
               <item repeat="0-1">My plan expires on</item>
               <item repeat="0-1">My plan expired on</item>
               <item repeat="0-1">My plan will expire on</item>
           </one-of>
        </rule>

        <rule id="hesitation">
           <one-of>
              <item>Hmm</item>
              <item>Mmm</item>
              <item>My</item>
           </one-of>
         </rule>

        <rule id="months">
           <tag>out.mon=""</tag>
        <item repeat="0-1"><ruleref uri="#text"/></item>

           <one-of>
             <item>january<tag>out.mon+="01";</tag></item>
             <item>february<tag>out.mon+="02";</tag></item>
             <item>march<tag>out.mon+="03";</tag></item>
             <item>april<tag>out.mon+="04";</tag></item>
             <item>may<tag>out.mon+="05";</tag></item>
             <item>june<tag>out.mon+="06";</tag></item>
             <item>july<tag>out.mon+="07";</tag></item>
             <item>august<tag>out.mon+="08";</tag></item>
```

```
            <item>september<tag>out.mon+="09";</tag></item>
            <item>october<tag>out.mon+="10";</tag></item>
            <item>november<tag>out.mon+="11";</tag></item>
            <item>december<tag>out.mon+="12";</tag></item>
            <item>jan<tag>out.mon+="01";</tag></item>
            <item>feb<tag>out.mon+="02";</tag></item>
            <item>aug<tag>out.mon+="08";</tag></item>
            <item>sept<tag>out.mon+="09";</tag></item>
            <item>oct<tag>out.mon+="10";</tag></item>
            <item>nov<tag>out.mon+="11";</tag></item>
            <item>dec<tag>out.mon+="12";</tag></item>
        </one-of>
    </rule>

    <rule id="digits">
        <one-of>
            <item>zero<tag>out=0;</tag></item>
            <item>one<tag>out=1;</tag></item>
            <item>two<tag>out=2;</tag></item>
            <item>three<tag>out=3;</tag></item>
            <item>four<tag>out=4;</tag></item>
            <item>five<tag>out=5;</tag></item>
            <item>six<tag>out=6;</tag></item>
            <item>seven<tag>out=7;</tag></item>
            <item>eight<tag>out=8;</tag></item>
            <item>nine<tag>out=9;</tag></item>
        </one-of>
    </rule>

    <rule id="teens">
        <one-of>
            <item>ten<tag>out=10;</tag></item>
            <item>eleven<tag>out=11;</tag></item>
            <item>twelve<tag>out=12;</tag></item>
            <item>thirteen<tag>out=13;</tag></item>
            <item>fourteen<tag>out=14;</tag></item>
            <item>fifteen<tag>out=15;</tag></item>
            <item>sixteen<tag>out=16;</tag></item>
            <item>seventeen<tag>out=17;</tag></item>
            <item>eighteen<tag>out=18;</tag></item>
            <item>nineteen<tag>out=19;</tag></item>
        </one-of>
    </rule>
```

```
        <rule id="thousands">
            <item>two thousand</item>
            <item repeat="0-1">and</item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out = rules.digits;</
tag></item>
            <item repeat="0-1"><ruleref uri="#teens"/><tag>out = rules.teens;</tag></
item>
            <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out =
 rules.above_twenty;</tag></item>
        </rule>

        <rule id="above_twenty">
            <one-of>
                <item>twenty<tag>out=20;</tag></item>
                <item>thirty<tag>out=30;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>
 </grammar>
```

Tanggal perpanjangan rencana, month/year

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: My plan will renew on July Two Thousand and Eleven
                 Output: 07/11

             Scenario 2:
                 Input: Renew plan on July Two Thousand and Eleven
```

```
                Output: 07/11

         -->

        <rule id="main" scope="public">
            <tag>out=""</tag>
            <item repeat="1-10">
                <item repeat="1"><ruleref uri="#months"/><tag>out = out +
 rules.months.mon + "/";</tag></item>
                <one-of>
                    <item><ruleref uri="#thousands"/><tag>out += rules.thousands;</
tag></item>
                    <item repeat="0-1"><ruleref uri="#digits"/><tag>out +=
 rules.digits;</tag></item>
                    <item repeat="0-1"><ruleref uri="#teens"/><tag>out +=
 rules.teens;</tag></item>
                    <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out +=
 rules.above_twenty;</tag></item>
                </one-of>
            </item>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
            <one-of>
                <item repeat="0-1">My plan will renew on</item>
                <item repeat="0-1">My plan was renewed on</item>
                <item repeat="0-1">Renew plan on</item>
            </one-of>
        </rule>

        <rule id="hesitation">
            <one-of>
               <item>Hmm</item>
               <item>Mmm</item>
               <item>My</item>
            </one-of>
          </rule>

        <rule id="months">
           <item repeat="0-1"><ruleref uri="#text"/></item>
           <tag>out.mon=""</tag>
           <one-of>
               <item>january<tag>out.mon+="01";</tag></item>
```

```
            <item>february<tag>out.mon+="02";</tag></item>
            <item>march<tag>out.mon+="03";</tag></item>
            <item>april<tag>out.mon+="04";</tag></item>
            <item>may<tag>out.mon+="05";</tag></item>
            <item>june<tag>out.mon+="06";</tag></item>
            <item>july<tag>out.mon+="07";</tag></item>
            <item>august<tag>out.mon+="08";</tag></item>
            <item>september<tag>out.mon+="09";</tag></item>
            <item>october<tag>out.mon+="10";</tag></item>
            <item>november<tag>out.mon+="11";</tag></item>
            <item>december<tag>out.mon+="12";</tag></item>
            <item>jan<tag>out.mon+="01";</tag></item>
            <item>feb<tag>out.mon+="02";</tag></item>
            <item>aug<tag>out.mon+="08";</tag></item>
            <item>sept<tag>out.mon+="09";</tag></item>
            <item>oct<tag>out.mon+="10";</tag></item>
            <item>nov<tag>out.mon+="11";</tag></item>
            <item>dec<tag>out.mon+="12";</tag></item>
        </one-of>
    </rule>

    <rule id="digits">
        <one-of>
            <item>zero<tag>out=0;</tag></item>
            <item>one<tag>out=1;</tag></item>
            <item>two<tag>out=2;</tag></item>
            <item>three<tag>out=3;</tag></item>
            <item>four<tag>out=4;</tag></item>
            <item>five<tag>out=5;</tag></item>
            <item>six<tag>out=6;</tag></item>
            <item>seven<tag>out=7;</tag></item>
            <item>eight<tag>out=8;</tag></item>
            <item>nine<tag>out=9;</tag></item>
        </one-of>
    </rule>

    <rule id="teens">
        <one-of>
            <item>ten<tag>out=10;</tag></item>
            <item>eleven<tag>out=11;</tag></item>
            <item>twelve<tag>out=12;</tag></item>
            <item>thirteen<tag>out=13;</tag></item>
            <item>fourteen<tag>out=14;</tag></item>
            <item>fifteen<tag>out=15;</tag></item>
```

```
                <item>sixteen<tag>out=16;</tag></item>
                <item>seventeen<tag>out=17;</tag></item>
                <item>eighteen<tag>out=18;</tag></item>
                <item>nineteen<tag>out=19;</tag></item>
            </one-of>
        </rule>

        <rule id="thousands">
            <item>two thousand</item>
            <item repeat="0-1">and</item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out = rules.digits;</tag></
item>
            <item repeat="0-1"><ruleref uri="#teens"/><tag>out = rules.teens;</tag></
item>
            <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out =
 rules.above_twenty;</tag></item>
        </rule>

        <rule id="above_twenty">
            <one-of>
                <item>twenty<tag>out=20;</tag></item>
                <item>thirty<tag>out=30;</tag></item>
                <item>forty<tag>out=40;</tag></item>
                <item>fifty<tag>out=50;</tag></item>
                <item>sixty<tag>out=60;</tag></item>
                <item>seventy<tag>out=70;</tag></item>
                <item>eighty<tag>out=80;</tag></item>
                <item>ninety<tag>out=90;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>
</grammar>
```

Rencanakan tanggal mulai, month/day

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
```

```
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: My plan will start on july twenty three
                 Output: 07/23

             Scenario 2:
                 Input: My plan will start on july fifteen
                 Output: 07/15

         -->

        <rule id="main" scope="public">
            <tag>out=""</tag>
            <item repeat="1-10">
                <item><ruleref uri="#months"/><tag>out= rules.months.mon + "/";</tag></
item>
                <one-of>
                    <item><ruleref uri="#digits"/><tag>out+= rules.digits;</tag></item>
                    <item><ruleref uri="#teens"/><tag>out+= rules.teens;</tag></item>
                    <item><ruleref uri="#above_twenty"/><tag>out+=
 rules.above_twenty;</tag></item>
                </one-of>
            </item>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
             <one-of>
                <item repeat="0-1">My plan started on</item>
                <item repeat="0-1">My plan will start on</item>
                <item repeat="0-1">I paid it on</item>
                <item repeat="0-1">I paid bill for</item>
             </one-of>
          </rule>

          <rule id="hesitation">
             <one-of>
                <item>Hmm</item>
```

```
            <item>Mmm</item>
            <item>My</item>
          </one-of>
       </rule>

    <rule id="months">
       <item repeat="0-1"><ruleref uri="#text"/></item>
       <tag>out.mon=""</tag>
       <one-of>
           <item>january<tag>out.mon+="01";</tag></item>
           <item>february<tag>out.mon+="02";</tag></item>
           <item>march<tag>out.mon+="03";</tag></item>
           <item>april<tag>out.mon+="04";</tag></item>
           <item>may<tag>out.mon+="05";</tag></item>
           <item>june<tag>out.mon+="06";</tag></item>
           <item>july<tag>out.mon+="07";</tag></item>
           <item>august<tag>out.mon+="08";</tag></item>
           <item>september<tag>out.mon+="09";</tag></item>
           <item>october<tag>out.mon+="10";</tag></item>
           <item>november<tag>out.mon+="11";</tag></item>
           <item>december<tag>out.mon+="12";</tag></item>
           <item>jan<tag>out.mon+="01";</tag></item>
           <item>feb<tag>out.mon+="02";</tag></item>
           <item>aug<tag>out.mon+="08";</tag></item>
           <item>sept<tag>out.mon+="09";</tag></item>
           <item>oct<tag>out.mon+="10";</tag></item>
           <item>nov<tag>out.mon+="11";</tag></item>
           <item>dec<tag>out.mon+="12";</tag></item>
       </one-of>
    </rule>

    <rule id="digits">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <one-of>
            <item>0<tag>out=0;</tag></item>
            <item>1<tag>out=1;</tag></item>
            <item>2<tag>out=2;</tag></item>
            <item>3<tag>out=3;</tag></item>
            <item>4<tag>out=4;</tag></item>
            <item>5<tag>out=5;</tag></item>
            <item>6<tag>out=6;</tag></item>
            <item>7<tag>out=7;</tag></item>
            <item>8<tag>out=8;</tag></item>
            <item>9<tag>out=9;</tag></item>
```

```
            <item>first<tag>out=01;</tag></item>
            <item>second<tag>out=02;</tag></item>
            <item>third<tag>out=03;</tag></item>
            <item>fourth<tag>out=04;</tag></item>
            <item>fifth<tag>out=05;</tag></item>
            <item>sixth<tag>out=06;</tag></item>
            <item>seventh<tag>out=07;</tag></item>
            <item>eighth<tag>out=08;</tag></item>
            <item>ninth<tag>out=09;</tag></item>
            <item>one<tag>out=1;</tag></item>
            <item>two<tag>out=2;</tag></item>
            <item>three<tag>out=3;</tag></item>
            <item>four<tag>out=4;</tag></item>
            <item>five<tag>out=5;</tag></item>
            <item>six<tag>out=6;</tag></item>
            <item>seven<tag>out=7;</tag></item>
            <item>eight<tag>out=8;</tag></item>
            <item>nine<tag>out=9;</tag></item>
        </one-of>
    </rule>

    <rule id="teens">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <one-of>
            <item>ten<tag>out=10;</tag></item>
            <item>tenth<tag>out=10;</tag></item>
            <item>eleven<tag>out=11;</tag></item>
            <item>twelve<tag>out=12;</tag></item>
            <item>thirteen<tag>out=13;</tag></item>
            <item>fourteen<tag>out=14;</tag></item>
            <item>fifteen<tag>out=15;</tag></item>
            <item>sixteen<tag>out=16;</tag></item>
            <item>seventeen<tag>out=17;</tag></item>
            <item>eighteen<tag>out=18;</tag></item>
            <item>nineteen<tag>out=19;</tag></item>
            <item>tenth<tag>out=10;</tag></item>
            <item>eleventh<tag>out=11;</tag></item>
            <item>twelveth<tag>out=12;</tag></item>
            <item>thirteenth<tag>out=13;</tag></item>
            <item>fourteenth<tag>out=14;</tag></item>
            <item>fifteenth<tag>out=15;</tag></item>
            <item>sixteenth<tag>out=16;</tag></item>
            <item>seventeenth<tag>out=17;</tag></item>
            <item>eighteenth<tag>out=18;</tag></item>
```

```
                <item>nineteenth<tag>out=19;</tag></item>
            </one-of>
        </rule>

        <rule id="above_twenty">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <one-of>
                <item>twenty<tag>out=20;</tag></item>
                <item>thirty<tag>out=30;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>
</grammar>
```

Tanggal mulai layanan, month/day

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: My plan starts on july twenty three
                 Output: 07/23

             Scenario 2:
                 Input: I want to activate on july fifteen
                 Output: 07/15

         -->

        <rule id="main" scope="public">
            <tag>out=""</tag>
```

```
            <item repeat="1-10">
                <item><ruleref uri="#months"/><tag>out= rules.months.mon + "/";</tag></
item>
                <one-of>
                    <item><ruleref uri="#digits"/><tag>out+= rules.digits;</tag></item>
                    <item><ruleref uri="#teens"/><tag>out+= rules.teens;</tag></item>
                    <item><ruleref uri="#above_twenty"/><tag>out+=
 rules.above_twenty;</tag></item>
                </one-of>
            </item>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
             <one-of>
                <item repeat="0-1">My plan starts on</item>
                <item repeat="0-1">I want to start my plan on</item>
                <item repeat="0-1">Activation date of</item>
                <item repeat="0-1">Start activation on</item>
                <item repeat="0-1">I want to activate on</item>
                <item repeat="0-1">Activate plan starting</item>
                <item repeat="0-1">Starting</item>
                <item repeat="0-1">Start on</item>
             </one-of>
          </rule>

          <rule id="hesitation">
             <one-of>
                <item>Hmm</item>
                <item>Mmm</item>
                <item>My</item>
             </one-of>
           </rule>

        <rule id="months">
           <item repeat="0-1"><ruleref uri="#text"/></item>
           <tag>out.mon=""</tag>
           <one-of>
               <item>january<tag>out.mon+="01";</tag></item>
               <item>february<tag>out.mon+="02";</tag></item>
               <item>march<tag>out.mon+="03";</tag></item>
               <item>april<tag>out.mon+="04";</tag></item>
               <item>may<tag>out.mon+="05";</tag></item>
               <item>june<tag>out.mon+="06";</tag></item>
```

```
            <item>july<tag>out.mon+="07";</tag></item>
            <item>august<tag>out.mon+="08";</tag></item>
            <item>september<tag>out.mon+="09";</tag></item>
            <item>october<tag>out.mon+="10";</tag></item>
            <item>november<tag>out.mon+="11";</tag></item>
            <item>december<tag>out.mon+="12";</tag></item>
            <item>jan<tag>out.mon+="01";</tag></item>
            <item>feb<tag>out.mon+="02";</tag></item>
            <item>aug<tag>out.mon+="08";</tag></item>
            <item>sept<tag>out.mon+="09";</tag></item>
            <item>oct<tag>out.mon+="10";</tag></item>
            <item>nov<tag>out.mon+="11";</tag></item>
            <item>dec<tag>out.mon+="12";</tag></item>
        </one-of>
    </rule>

    <rule id="digits">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <one-of>
            <item>0<tag>out=0;</tag></item>
            <item>1<tag>out=1;</tag></item>
            <item>2<tag>out=2;</tag></item>
            <item>3<tag>out=3;</tag></item>
            <item>4<tag>out=4;</tag></item>
            <item>5<tag>out=5;</tag></item>
            <item>6<tag>out=6;</tag></item>
            <item>7<tag>out=7;</tag></item>
            <item>8<tag>out=8;</tag></item>
            <item>9<tag>out=9;</tag></item>
            <item>first<tag>out=01;</tag></item>
            <item>second<tag>out=02;</tag></item>
            <item>third<tag>out=03;</tag></item>
            <item>fourth<tag>out=04;</tag></item>
            <item>fifth<tag>out=05;</tag></item>
            <item>sixth<tag>out=06;</tag></item>
            <item>seventh<tag>out=07;</tag></item>
            <item>eighth<tag>out=08;</tag></item>
            <item>ninth<tag>out=09;</tag></item>
            <item>one<tag>out=1;</tag></item>
            <item>two<tag>out=2;</tag></item>
            <item>three<tag>out=3;</tag></item>
            <item>four<tag>out=4;</tag></item>
            <item>five<tag>out=5;</tag></item>
            <item>six<tag>out=6;</tag></item>
```

```
                <item>seven<tag>out=7;</tag></item>
                <item>eight<tag>out=8;</tag></item>
                <item>nine<tag>out=9;</tag></item>
            </one-of>
        </rule>

        <rule id="teens">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <one-of>
                <item>ten<tag>out=10;</tag></item>
                <item>tenth<tag>out=10;</tag></item>
                <item>eleven<tag>out=11;</tag></item>
                <item>twelve<tag>out=12;</tag></item>
                <item>thirteen<tag>out=13;</tag></item>
                <item>fourteen<tag>out=14;</tag></item>
                <item>fifteen<tag>out=15;</tag></item>
                <item>sixteen<tag>out=16;</tag></item>
                <item>seventeen<tag>out=17;</tag></item>
                <item>eighteen<tag>out=18;</tag></item>
                <item>nineteen<tag>out=19;</tag></item>
                <item>tenth<tag>out=10;</tag></item>
                <item>eleventh<tag>out=11;</tag></item>
                <item>twelveth<tag>out=12;</tag></item>
                <item>thirteenth<tag>out=13;</tag></item>
                <item>fourteenth<tag>out=14;</tag></item>
                <item>fifteenth<tag>out=15;</tag></item>
                <item>sixteenth<tag>out=16;</tag></item>
                <item>seventeenth<tag>out=17;</tag></item>
                <item>eighteenth<tag>out=18;</tag></item>
                <item>nineteenth<tag>out=19;</tag></item>
            </one-of>
        </rule>

        <rule id="above_twenty">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <one-of>
                <item>twenty<tag>out=20;</tag></item>
                <item>thirty<tag>out=30;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>
</grammar>
```

Kuantitas peralatan

```
<?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>
```

Jumlah tagihan

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

                 Input: I want to make a payment of one hundred ten dollars
                 Output: $110

         -->

        <rule id="main" scope="public">
            <tag>out="$"</tag>
            <one-of>
                <item><ruleref uri="#sub_hundred"/><tag>out += rules.sub_hundred.sh;</
tag></item>
                <item><ruleref uri="#subThousands"/><tag>out += rules.subThousands;</
tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#thanks"/></item>
        </rule>

        <rule id="text">
```

```
    <item repeat="0-1"><ruleref uri="#hesitation"/></item>
    <one-of>
       <item repeat="0-1">I want to make a payment for</item>
       <item repeat="0-1">I want to make a payment of</item>
       <item repeat="0-1">Pay a total of</item>
       <item repeat="0-1">Paying</item>
       <item repeat="0-1">Pay bill for </item>
    </one-of>
</rule>

<rule id="hesitation">
    <one-of>
       <item>Hmm</item>
       <item>Mmm</item>
       <item>My</item>
    </one-of>
  </rule>

  <rule id="thanks">
      <one-of>
         <item>Thanks</item>
         <item>I think</item>
      </one-of>
    </rule>

<rule id="digits">
    <item repeat="0-1"><ruleref uri="#text"/></item>
    <tag>out.num = 0;</tag>
    <one-of>
        <item>0<tag>out.num+=0;</tag></item>
        <item>1<tag>out.num+=1;</tag></item>
        <item>2<tag>out.num+=2;</tag></item>
        <item>3<tag>out.num+=3;</tag></item>
        <item>4<tag>out.num+=4;</tag></item>
        <item>5<tag>out.num+=5;</tag></item>
        <item>6<tag>out.num+=6;</tag></item>
        <item>7<tag>out.num+=7;</tag></item>
        <item>8<tag>out.num+=8;</tag></item>
        <item>9<tag>out.num+=9;</tag></item>
        <item>one<tag>out.num+=1;</tag></item>
        <item>two<tag>out.num+=2;</tag></item>
        <item>three<tag>out.num+=3;</tag></item>
        <item>four<tag>out.num+=4;</tag></item>
        <item>five<tag>out.num+=5;</tag></item>
```

```
                <item>six<tag>out.num+=6;</tag></item>
                <item>seven<tag>out.num+=7;</tag></item>
                <item>eight<tag>out.num+=8;</tag></item>
                <item>nine<tag>out.num+=9;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#currency"/></item>
        </rule>

        <rule id="teens">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.teen = 0;</tag>
            <one-of>
                <item>ten<tag>out.teen+=10;</tag></item>
                <item>eleven<tag>out.teen+=11;</tag></item>
                <item>twelve<tag>out.teen+=12;</tag></item>
                <item>thirteen<tag>out.teen+=13;</tag></item>
                <item>fourteen<tag>out.teen+=14;</tag></item>
                <item>fifteen<tag>out.teen+=15;</tag></item>
                <item>sixteen<tag>out.teen+=16;</tag></item>
                <item>seventeen<tag>out.teen+=17;</tag></item>
                <item>eighteen<tag>out.teen+=18;</tag></item>
                <item>nineteen<tag>out.teen+=19;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#currency"/></item>
        </rule>

        <rule id="above_twenty">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.tens = 0;</tag>
            <one-of>
                <item>twenty<tag>out.tens+=20;</tag></item>
                <item>thirty<tag>out.tens+=30;</tag></item>
                <item>forty<tag>out.tens+=40;</tag></item>
                <item>fifty<tag>out.tens+=50;</tag></item>
                <item>sixty<tag>out.tens+=60;</tag></item>
                <item>seventy<tag>out.tens+=70;</tag></item>
                <item>eighty<tag>out.tens+=80;</tag></item>
                <item>ninety<tag>out.tens+=90;</tag></item>
                <item>hundred<tag>out.tens+=100;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#currency"/></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out.tens +=
rules.digits.num;</tag></item>
        </rule>
```

```
        <rule id="currency">
            <one-of>
                <item repeat="0-1">dollars</item>
                <item repeat="0-1">Dollars</item>
                <item repeat="0-1">dollar</item>
                <item repeat="0-1">Dollar</item>
            </one-of>
        </rule>


        <rule id="sub_hundred">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.sh = 0;</tag>
            <one-of>
                <item><ruleref uri="#teens"/><tag>out.sh += rules.teens.teen;</tag></
item>
                <item>
                    <ruleref uri="#above_twenty"/><tag>out.sh +=
 rules.above_twenty.tens;</tag>
                </item>
                <item><ruleref uri="#digits"/><tag>out.sh += rules.digits.num;</tag></
item>
            </one-of>
         </rule>

        <rule id="subThousands">
            <ruleref uri="#sub_hundred"/><tag>out = (100 * rules.sub_hundred.sh);</tag>
            hundred
            <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out +=
 rules.above_twenty.tens;</tag></item>
            <item repeat="0-1"><ruleref uri="#teens"/><tag>out += rules.teens.teen;</
tag></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits.num;</
tag></item>
            <item repeat="0-1"><ruleref uri="#currency"/></item>
        </rule>
</grammar>
```

Tata bahasa generik (download)

Kami menyediakan tata bahasa generik berikut: alfanumerik, mata uang, tanggal (mm/dd/yy), angka, salam, keraguan, dan agen.

Alfanumerik

```
<?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>
```

Mata Uang

```
<?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>
```

Tanggal, dd/mm

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

        <rule id="main" scope="public">
            <tag>out=""</tag>
            <item repeat="1-10">
                <one-of>
                    <item><ruleref uri="#digits"/><tag>out += rules.digits + " ";</
tag></item>
                    <item><ruleref uri="#teens"/><tag>out += rules.teens+ " ";</tag></
item>
                    <item><ruleref uri="#above_twenty"/><tag>out += rules.above_twenty+
 " ";</tag></item>
                </one-of>
                <item><ruleref uri="#months"/><tag>out = out + rules.months;</tag></
item>
            </item>
        </rule>

        <rule id="months">
           <one-of>
             <item>january<tag>out="january";</tag></item>
             <item>february<tag>out="february";</tag></item>
             <item>march<tag>out="march";</tag></item>
             <item>april<tag>out="april";</tag></item>
             <item>may<tag>out="may";</tag></item>
             <item>june<tag>out="june";</tag></item>
             <item>july<tag>out="july";</tag></item>
             <item>august<tag>out="august";</tag></item>
             <item>september<tag>out="september";</tag></item>
             <item>october<tag>out="october";</tag></item>
             <item>november<tag>out="november";</tag></item>
```

```
            <item>december<tag>out="december";</tag></item>
            <item>jan<tag>out="january";</tag></item>
            <item>feb<tag>out="february";</tag></item>
            <item>aug<tag>out="august";</tag></item>
            <item>sept<tag>out="september";</tag></item>
            <item>oct<tag>out="october";</tag></item>
            <item>nov<tag>out="november";</tag></item>
            <item>dec<tag>out="december";</tag></item>
        </one-of>
    </rule>

    <rule id="digits">
        <one-of>
            <item>0<tag>out=0;</tag></item>
            <item>1<tag>out=1;</tag></item>
            <item>2<tag>out=2;</tag></item>
            <item>3<tag>out=3;</tag></item>
            <item>4<tag>out=4;</tag></item>
            <item>5<tag>out=5;</tag></item>
            <item>6<tag>out=6;</tag></item>
            <item>7<tag>out=7;</tag></item>
            <item>8<tag>out=8;</tag></item>
            <item>9<tag>out=9;</tag></item>
            <item>first<tag>out=1;</tag></item>
            <item>second<tag>out=2;</tag></item>
            <item>third<tag>out=3;</tag></item>
            <item>fourth<tag>out=4;</tag></item>
            <item>fifth<tag>out=5;</tag></item>
            <item>sixth<tag>out=6;</tag></item>
            <item>seventh<tag>out=7;</tag></item>
            <item>eighth<tag>out=8;</tag></item>
            <item>ninth<tag>out=9;</tag></item>
            <item>one<tag>out=1;</tag></item>
            <item>two<tag>out=2;</tag></item>
            <item>three<tag>out=3;</tag></item>
            <item>four<tag>out=4;</tag></item>
            <item>five<tag>out=5;</tag></item>
            <item>six<tag>out=6;</tag></item>
            <item>seven<tag>out=7;</tag></item>
            <item>eight<tag>out=8;</tag></item>
            <item>nine<tag>out=9;</tag></item>
        </one-of>
    </rule>
```

```
        <rule id="teens">
            <one-of>
                <item>ten<tag>out=10;</tag></item>
                <item>tenth<tag>out=10;</tag></item>
                <item>eleven<tag>out=11;</tag></item>
                <item>twelve<tag>out=12;</tag></item>
                <item>thirteen<tag>out=13;</tag></item>
                <item>fourteen<tag>out=14;</tag></item>
                <item>fifteen<tag>out=15;</tag></item>
                <item>sixteen<tag>out=16;</tag></item>
                <item>seventeen<tag>out=17;</tag></item>
                <item>eighteen<tag>out=18;</tag></item>
                <item>nineteen<tag>out=19;</tag></item>
                <item>tenth<tag>out=10;</tag></item>
                <item>eleventh<tag>out=11;</tag></item>
                <item>twelveth<tag>out=12;</tag></item>
                <item>thirteenth<tag>out=13;</tag></item>
                <item>fourteenth<tag>out=14;</tag></item>
                <item>fifteenth<tag>out=15;</tag></item>
                <item>sixteenth<tag>out=16;</tag></item>
                <item>seventeenth<tag>out=17;</tag></item>
                <item>eighteenth<tag>out=18;</tag></item>
                <item>nineteenth<tag>out=19;</tag></item>
            </one-of>
        </rule>

        <rule id="above_twenty">
            <one-of>
                <item>twenty<tag>out=20;</tag></item>
                <item>thirty<tag>out=30;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>
</grammar>
```

Tanggal, mm/yy

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
```

```
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

        <rule id="main" scope="public">
            <tag>out=""</tag>
            <item repeat="1-10">
                <item repeat="1"><ruleref uri="#months"/><tag>out = out +
 rules.months.mon + " ";</tag></item>
                <one-of>
                    <item><ruleref uri="#thousands"/><tag>out += rules.thousands;</
tag></item>
                    <item repeat="0-1"><ruleref uri="#digits"/><tag>out +=
 rules.digits;</tag></item>
                    <item repeat="0-1"><ruleref uri="#teens"/><tag>out +=
 rules.teens;</tag></item>
                    <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out +=
 rules.above_twenty;</tag></item>
                </one-of>
            </item>
        </rule>

        <rule id="months">
           <tag>out.mon=""</tag>
           <one-of>
               <item>january<tag>out.mon+="january";</tag></item>
               <item>february<tag>out.mon+="february";</tag></item>
               <item>march<tag>out.mon+="march";</tag></item>
               <item>april<tag>out.mon+="april";</tag></item>
               <item>may<tag>out.mon+="may";</tag></item>
               <item>june<tag>out.mon+="june";</tag></item>
               <item>july<tag>out.mon+="july";</tag></item>
               <item>august<tag>out.mon+="august";</tag></item>
               <item>september<tag>out.mon+="september";</tag></item>
               <item>october<tag>out.mon+="october";</tag></item>
               <item>november<tag>out.mon+="november";</tag></item>
               <item>december<tag>out.mon+="december";</tag></item>
               <item>jan<tag>out.mon+="january";</tag></item>
               <item>feb<tag>out.mon+="february";</tag></item>
               <item>aug<tag>out.mon+="august";</tag></item>
               <item>sept<tag>out.mon+="september";</tag></item>
               <item>oct<tag>out.mon+="october";</tag></item>
               <item>nov<tag>out.mon+="november";</tag></item>
```

```
            <item>dec<tag>out.mon+="december";</tag></item>
        </one-of>
    </rule>

    <rule id="digits">
        <one-of>
            <item>zero<tag>out=0;</tag></item>
            <item>one<tag>out=1;</tag></item>
            <item>two<tag>out=2;</tag></item>
            <item>three<tag>out=3;</tag></item>
            <item>four<tag>out=4;</tag></item>
            <item>five<tag>out=5;</tag></item>
            <item>six<tag>out=6;</tag></item>
            <item>seven<tag>out=7;</tag></item>
            <item>eight<tag>out=8;</tag></item>
            <item>nine<tag>out=9;</tag></item>
        </one-of>
    </rule>

    <rule id="teens">
        <one-of>
            <item>ten<tag>out=10;</tag></item>
            <item>eleven<tag>out=11;</tag></item>
            <item>twelve<tag>out=12;</tag></item>
            <item>thirteen<tag>out=13;</tag></item>
            <item>fourteen<tag>out=14;</tag></item>
            <item>fifteen<tag>out=15;</tag></item>
            <item>sixteen<tag>out=16;</tag></item>
            <item>seventeen<tag>out=17;</tag></item>
            <item>eighteen<tag>out=18;</tag></item>
            <item>nineteen<tag>out=19;</tag></item>
        </one-of>
    </rule>

    <!--  <rule id="singleDigit">
        <item><ruleref uri="#digits"/><tag>out += rules.digits;</tag></item>
    </rule> -->

    <rule id="thousands">
        <!-- <item>
            <ruleref uri="#digits"/>
            <tag>out = (1000 * rules.digits);</tag>
            thousand
        </item> -->
```

```
            <item>two thousand<tag>out=2000;</tag></item>
            <item repeat="0-1">and</item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
            <item repeat="0-1"><ruleref uri="#teens"/><tag>out += rules.teens;</tag></
item>
            <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out +=
 rules.above_twenty;</tag></item>
        </rule>

        <rule id="above_twenty">
            <one-of>
                <item>twenty<tag>out=20;</tag></item>
                <item>thirty<tag>out=30;</tag></item>
                <item>forty<tag>out=40;</tag></item>
                <item>fifty<tag>out=50;</tag></item>
                <item>sixty<tag>out=60;</tag></item>
                <item>seventy<tag>out=70;</tag></item>
                <item>eighty<tag>out=80;</tag></item>
                <item>ninety<tag>out=90;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>

</grammar>
```

Tanggal, 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>
               <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>
```

Angka, digit

```
<?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>
```

Bilangan, ordinal

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

        <rule id="main" scope="public">
            <tag>out="";</tag>
            <one-of>
              <item repeat="1"><ruleref uri="#digits"/><tag>out+= rules.digits;</tag></
item>
              <item repeat="1"><ruleref uri="#teens"/><tag>out+= rules.teens;</tag></
item>
              <item repeat="1"><ruleref uri="#above_twenty"/><tag>out+=
 rules.above_twenty;</tag></item>
            </one-of>
```

```
        </rule>

        <rule id="digits">
            <one-of>
                <item>0<tag>out=0;</tag></item>
                <item>1<tag>out=1;</tag></item>
                <item>2<tag>out=2;</tag></item>
                <item>3<tag>out=3;</tag></item>
                <item>4<tag>out=4;</tag></item>
                <item>5<tag>out=5;</tag></item>
                <item>6<tag>out=6;</tag></item>
                <item>7<tag>out=7;</tag></item>
                <item>8<tag>out=8;</tag></item>
                <item>9<tag>out=9;</tag></item>
                <item>one<tag>out=1;</tag></item>
                <item>two<tag>out=2;</tag></item>
                <item>three<tag>out=3;</tag></item>
                <item>four<tag>out=4;</tag></item>
                <item>five<tag>out=5;</tag></item>
                <item>six<tag>out=6;</tag></item>
                <item>seven<tag>out=7;</tag></item>
                <item>eight<tag>out=8;</tag></item>
                <item>nine<tag>out=9;</tag></item>
            </one-of>
        </rule>

        <rule id="teens">
            <one-of>
                <item>ten<tag>out=10;</tag></item>
                <item>eleven<tag>out=11;</tag></item>
                <item>twelve<tag>out=12;</tag></item>
                <item>thirteen<tag>out=13;</tag></item>
                <item>fourteen<tag>out=14;</tag></item>
                <item>fifteen<tag>out=15;</tag></item>
                <item>sixteen<tag>out=16;</tag></item>
                <item>seventeen<tag>out=17;</tag></item>
                <item>eighteen<tag>out=18;</tag></item>
                <item>nineteen<tag>out=19;</tag></item>
                <item>10<tag>out=10;</tag></item>
                <item>11<tag>out=11;</tag></item>
                <item>12<tag>out=12;</tag></item>
                <item>13<tag>out=13;</tag></item>
                <item>14<tag>out=14;</tag></item>
                <item>15<tag>out=15;</tag></item>
```

```
                <item>16<tag>out=16;</tag></item>
                <item>17<tag>out=17;</tag></item>
                <item>18<tag>out=18;</tag></item>
                <item>19<tag>out=19;</tag></item>
            </one-of>
        </rule>

        <rule id="above_twenty">
            <one-of>
                <item>twenty<tag>out=20;</tag></item>
                <item>thirty<tag>out=30;</tag></item>
                <item>forty<tag>out=40;</tag></item>
                <item>fifty<tag>out=50;</tag></item>
                <item>sixty<tag>out=60;</tag></item>
                <item>seventy<tag>out=70;</tag></item>
                <item>eighty<tag>out=80;</tag></item>
                <item>ninety<tag>out=90;</tag></item>
                <item>20<tag>out=20;</tag></item>
                <item>30<tag>out=30;</tag></item>
                <item>40<tag>out=40;</tag></item>
                <item>50<tag>out=50;</tag></item>
                <item>60<tag>out=60;</tag></item>
                <item>70<tag>out=70;</tag></item>
                <item>80<tag>out=80;</tag></item>
                <item>90<tag>out=90;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>

</grammar>
```

Agen

```
<?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>
```

Salam

```
<?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>
```

Keraguan

```
<?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>
```

Jenis slot komposit

Slot komposit adalah kombinasi dari dua atau lebih slot yang menangkap beberapa bagian informasi dalam satu input pengguna. Misalnya, Anda dapat mengonfigurasi bot untuk mendapatkan lokasi dengan meminta “kota dan negara bagian atau kode pos”. Sebaliknya, ketika percakapan dikonfigurasi untuk menggunakan jenis slot terpisah menghasilkan pengalaman percakapan yang kaku (“Apa itu kota?” diikuti dengan “Apa kode pos?”). Dengan slot komposit, Anda dapat menangkap semua informasi melalui satu slot. Slot komposit adalah kombinasi slot yang disebut subslot, seperti kota, negara bagian, dan kode pos.

Anda dapat menggunakan kombinasi jenis slot Amazon Lex yang tersedia (built-in) dan slot Anda sendiri (slot khusus). Anda dapat merancang ekspresi logis untuk menangkap informasi dalam subslot yang diperlukan. Misalnya: kota dan negara bagian atau kode pos.

Jenis slot komposit hanya tersedia di en-US dan lokal dengan dukungan fitur terbatas melalui AI Generatif.

Membuat tipe slot komposit

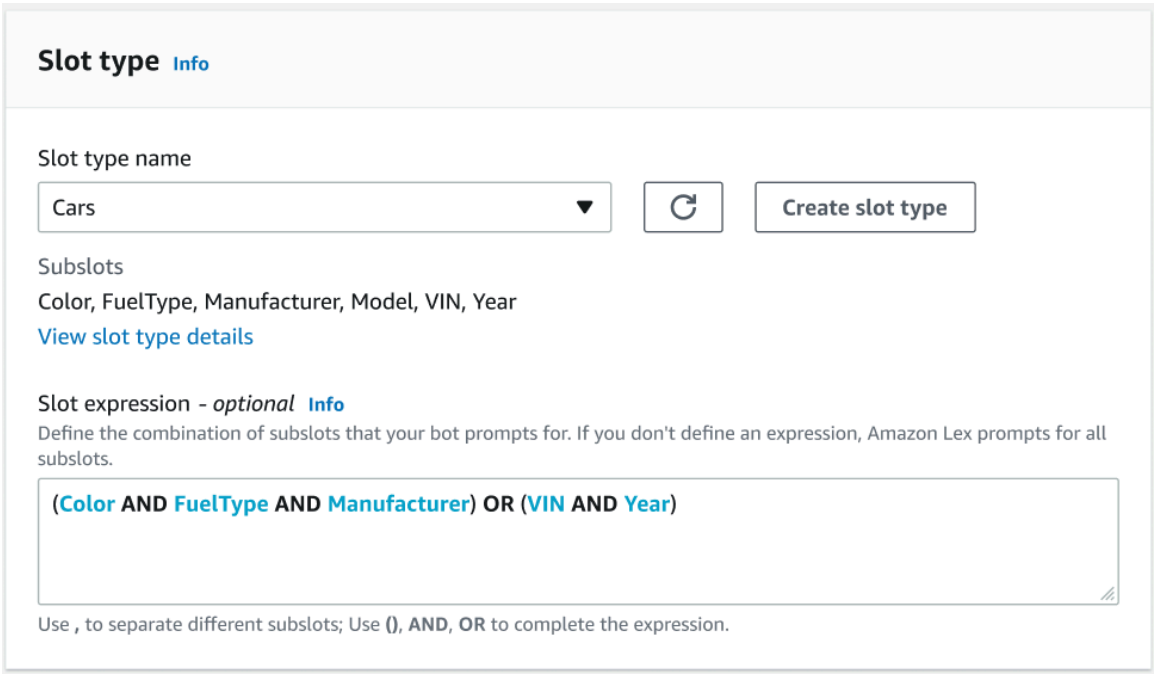
Untuk menggunakan subslot dalam slot komposit, Anda harus terlebih dahulu mengkonfigurasi jenis slot komposit. Untuk melakukannya, gunakan langkah-langkah menambahkan konsol tipe slot atau operasi API. Setelah Anda memilih nama dan deskripsi untuk jenis slot komposit, Anda harus memberikan informasi untuk subslot. Untuk informasi selengkapnya tentang menambahkan jenis slot, lihat Menambahkan jenis slot

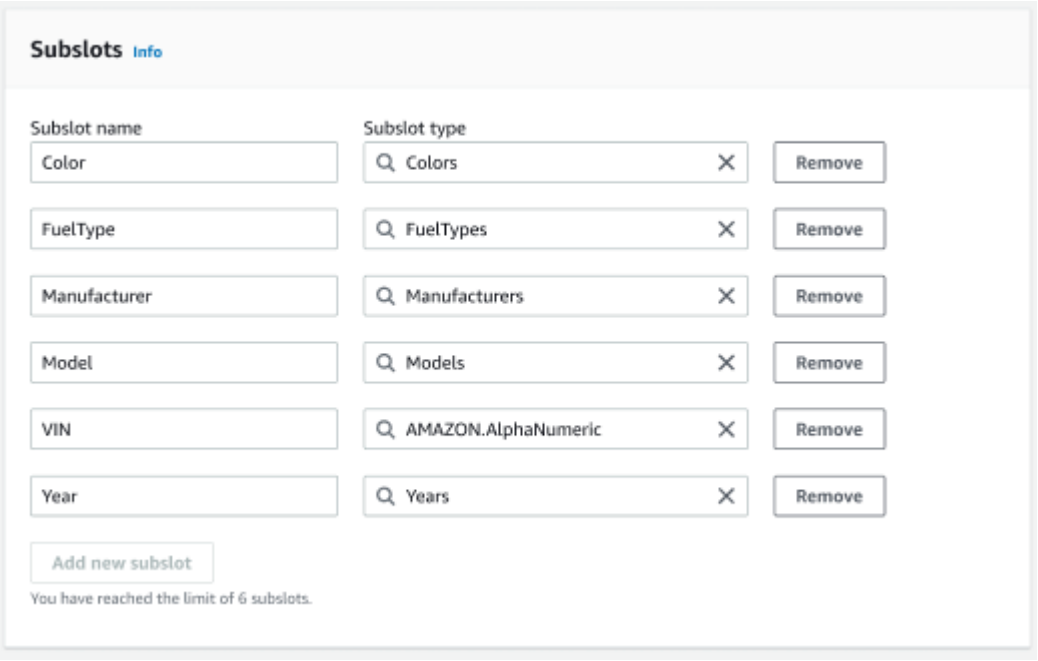
Subslot

Jenis slot komposit memerlukan konfigurasi slot yang mendasarinya, yang disebut subslot. Jika Anda ingin mendapatkan beberapa informasi dari pelanggan dalam satu permintaan, konfigurasikan kombinasi subslot. Misalnya: kota, negara bagian, dan kode pos. Anda dapat menambahkan hingga 6 subslot untuk slot komposit.

Slot jenis slot tunggal dapat digunakan untuk menambahkan subslot ke jenis slot komposit. Namun, Anda tidak dapat menggunakan jenis slot komposit sebagai tipe slot untuk subslot.

Gambar-gambar berikut adalah ilustrasi dari slot komposit “Mobil”, yang merupakan kombinasi dari subslot: Warna,, FuelType Produsen, Model, VIN, dan Tahun.





Pembuat ekspresi

Untuk mendorong pemenuhan slot komposit, Anda dapat menggunakan pembangun ekspresi secara opsional. Dengan pembangun ekspresi, Anda dapat merancang ekspresi slot logis untuk menangkap nilai subslot yang diperlukan dalam urutan yang diinginkan. Sebagai bagian dari ekspresi boolean, Anda dapat menggunakan operator seperti AND dan OR. Berdasarkan ekspresi yang dirancang, ketika subslot yang diperlukan terpenuhi, slot komposit dianggap terpenuhi.

Menggunakan tipe slot komposit

Untuk beberapa maksud, Anda mungkin ingin menangkap slot yang berbeda sebagai bagian dari satu slot. Misalnya, bot penjadwalan perawatan mobil mungkin memiliki maksud dengan ucapan berikut:

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

Maksudnya mengharapkan bahwa slot komposit {car} berisi daftar slot, yang terdiri dari detail mobil. Misalnya, “Toyota Camry Putih 2021”.

Slot komposit berbeda dari slot multi-nilai. Slot komposit terdiri dari beberapa slot, masing-masing dengan nilainya sendiri. Sedangkan, slot multi-nilai adalah slot tunggal yang dapat berisi daftar nilai. Untuk informasi lebih lanjut tentang slot multi-nilai lihat, Menggunakan beberapa nilai dalam slot

Untuk slot komposit, Amazon Lex mengembalikan nilai untuk setiap subslot dalam respons terhadap `RecognizeUtterance` operasi `RecognizeText` atau. Berikut ini adalah informasi slot yang

dikembalikan untuk ucapan: “Saya ingin menjadwalkan layanan untuk “2021 White Toyota Camry” saya dari bot. CarService

```
"slots": {
    "CarType": {
        "value": {
            "originalValue": "White Toyota Camry 2021",
            "interpretedValue": "White Toyota Camry 2021",
            "resolvedValues": [
                "white Toyota Camry 2021"
            ]
        },
        "subSlots": {
            "Color": {
                "value": {
                    "originalValue": "White",
                    "interpretedValue": "White",
                    "resolvedValues": [
                        "white"
                    ]
                },
                "shape": "Scalar"
            },
            "Manufacturer": {
                "value": {
                    "originalValue": "Toyota",
                    "interpretedValue": "Toyota",
                    "resolvedValues": [
                        "Toyota"
                    ]
                },
                "shape": "Scalar"
            },
            "Model": {
                "value": {
                    "originalValue": "Camry",
                    "interpretedValue": "Camry",
                    "resolvedValues": [
                       "Camry"
                    ]
                },
                "shape": "Scalar"
            },
```

```
            "Year": {
                "value": {
                    "originalValue": "2021",
                    "interpretedValue": "2021",
                    "resolvedValues": [
                        "2021"
                    ]
                },
                "shape": "Scalar"
            }
        }
    },
    ...
}
```

Slot komposit dapat ditimbulkan pada putaran pertama atau putaran ke-n percakapan. Berdasarkan nilai input yang diberikan, slot komposit dapat menghasilkan subslot yang diperlukan yang tersisa.

Slot komposit selalu mengembalikan nilai untuk setiap subslot. Ketika ucapan tidak mengandung nilai yang dapat dikenali untuk subslot tertentu, tidak ada respons yang dikembalikan untuk subslot tertentu.

Slot komposit bekerja dengan input teks dan suara.

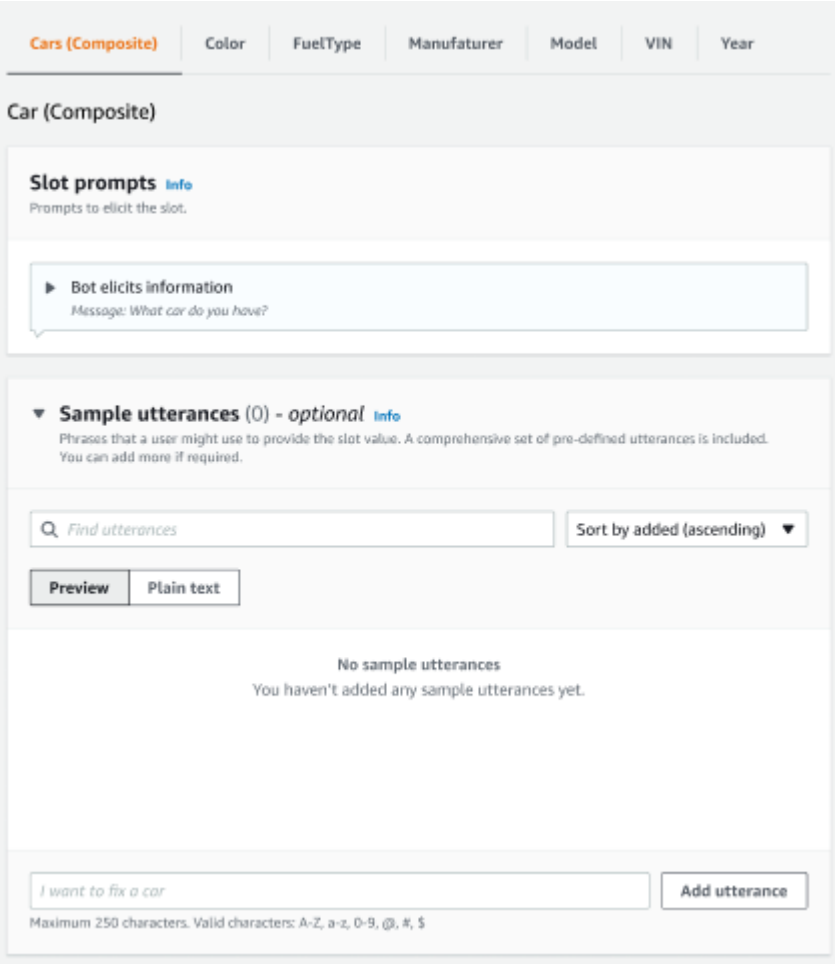
Saat menambahkan slot ke maksud, slot komposit hanya tersedia sebagai jenis slot khusus.

Anda dapat menggunakan slot Komposit dalam petunjuk. Misalnya, Anda dapat mengatur prompt konfirmasi untuk maksud.

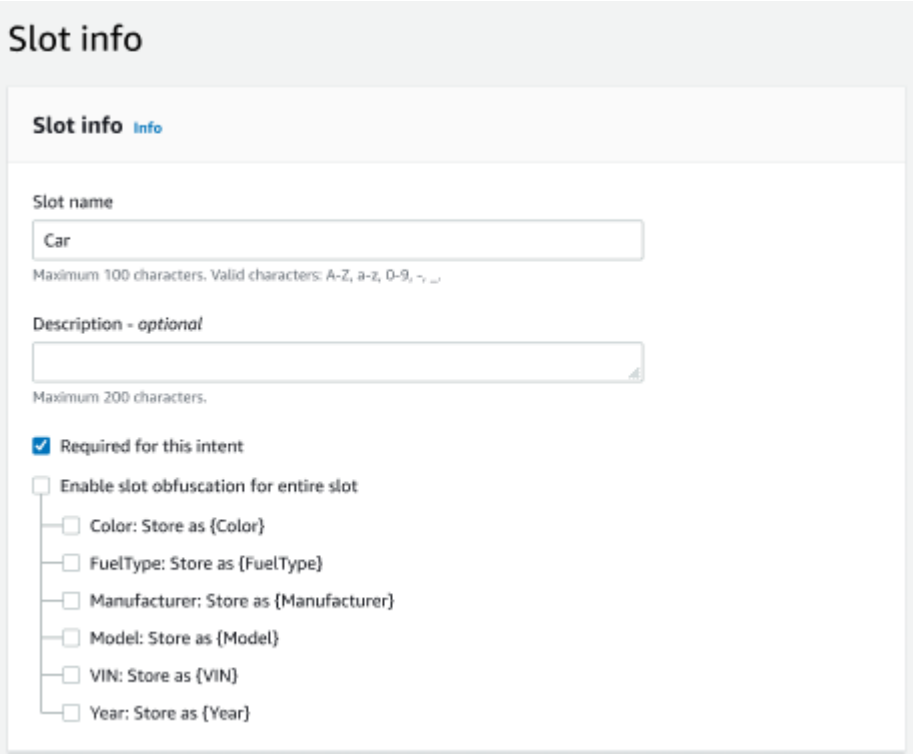
```
Would you like me to schedule service for your 2021 White Toyota Camry?
```

Ketika Amazon Lex mengirimkan prompt kepada pengguna, ia mengirimkan “Apakah Anda ingin saya menjadwalkan layanan untuk Toyota Camry Putih 2021 Anda?”

Setiap subslot dikonfigurasi sebagai slot. Anda dapat menambahkan petunjuk slot untuk mendapatkan subslot dan contoh ucapan. Anda dapat mengaktifkan tunggu dan melanjutkan untuk subslot serta nilai default. Untuk informasi selengkapnya, lihat Menggunakan nilai slot default dalam maksud untuk bot Lex V2 Anda



Anda dapat menggunakan pengaburan slot untuk menutupi seluruh slot komposit dalam log percakapan. Harap dicatat bahwa pengaburan slot diterapkan pada tingkat slot komposit dan ketika diaktifkan, nilai untuk subslot milik slot komposit dikaburkan. Saat Anda mengaburkan nilai slot, nilai masing-masing nilai slot diganti dengan nama slot. Untuk informasi selengkapnya, lihat Mengaburkan nilai slot dalam log percakapan dari Lex V2.



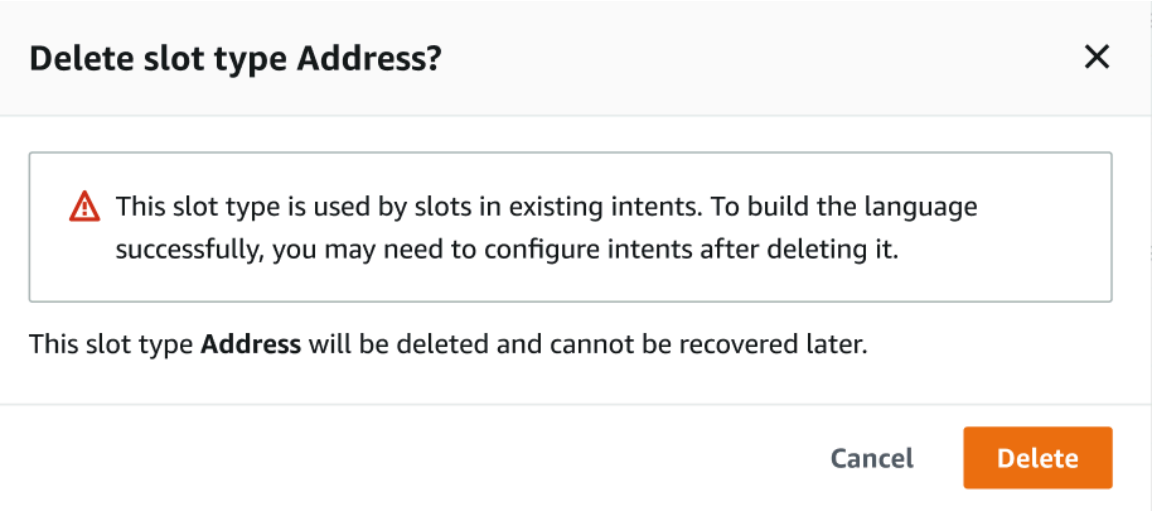
Mengedit jenis slot komposit

Anda dapat mengedit subslot dari dalam konfigurasi slot komposit untuk memodifikasi nama subslot dan jenis slot. Namun, ketika slot komposit digunakan oleh maksud, Anda harus mengedit maksud sebelum memodifikasi subslot.

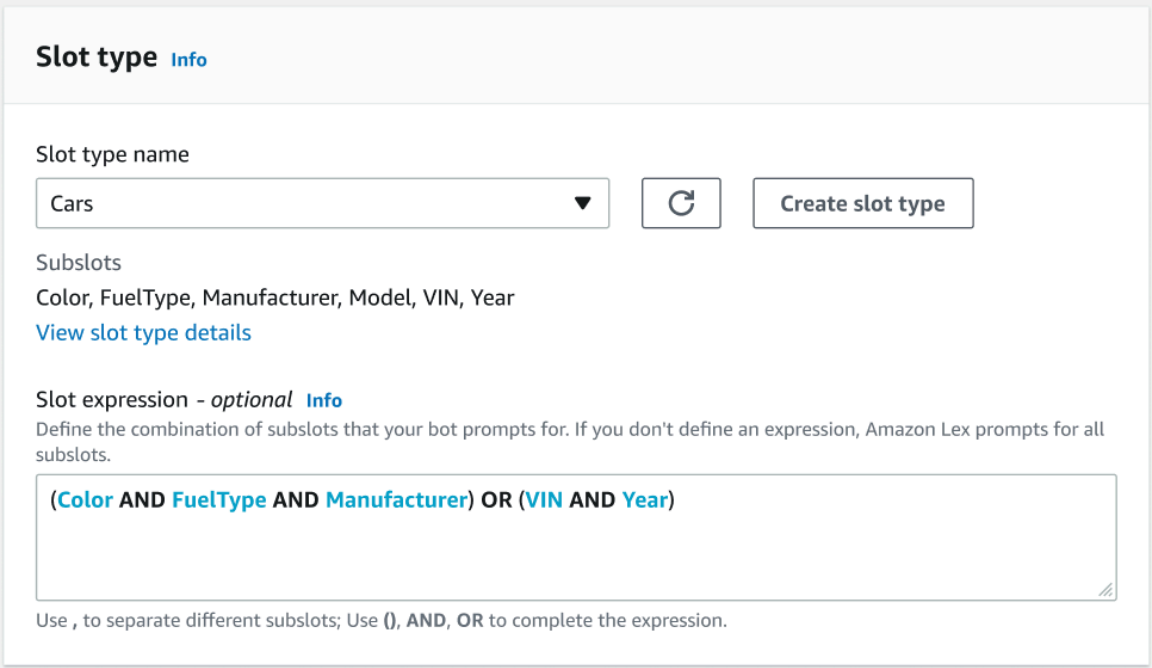
> Existing intents use this slot type. To build the language successfully, you may need to configure those intents after editing sub slots.

Menghapus jenis slot komposit

Anda dapat menghapus subslot dari dalam konfigurasi slot komposit. Harap dicatat bahwa ketika subslot digunakan dalam maksud, subslot masih dihapus dari maksud itu.



Ekspresi slot di pembuat ekspresi memberikan peringatan untuk menginformasikan tentang subslot yang dihapus.



Menguji bot menggunakan konsol

Konsol Amazon Lex V2 berisi jendela uji yang dapat Anda gunakan untuk menguji interaksi dengan bot Anda. Anda menggunakan jendela uji untuk melakukan percakapan uji dengan bot Anda dan untuk melihat tanggapan yang diterima aplikasi Anda dari bot.

Ada dua jenis pengujian yang dapat Anda lakukan dengan bot Anda. Yang pertama, pengujian ekspres, memungkinkan Anda untuk menguji bot Anda dengan frasa yang tepat yang Anda gunakan

untuk membuat bot. Misalnya, jika Anda menambahkan ucapan “Saya ingin mengambil bunga” ke maksud Anda, Anda dapat menguji bot menggunakan frasa yang tepat.

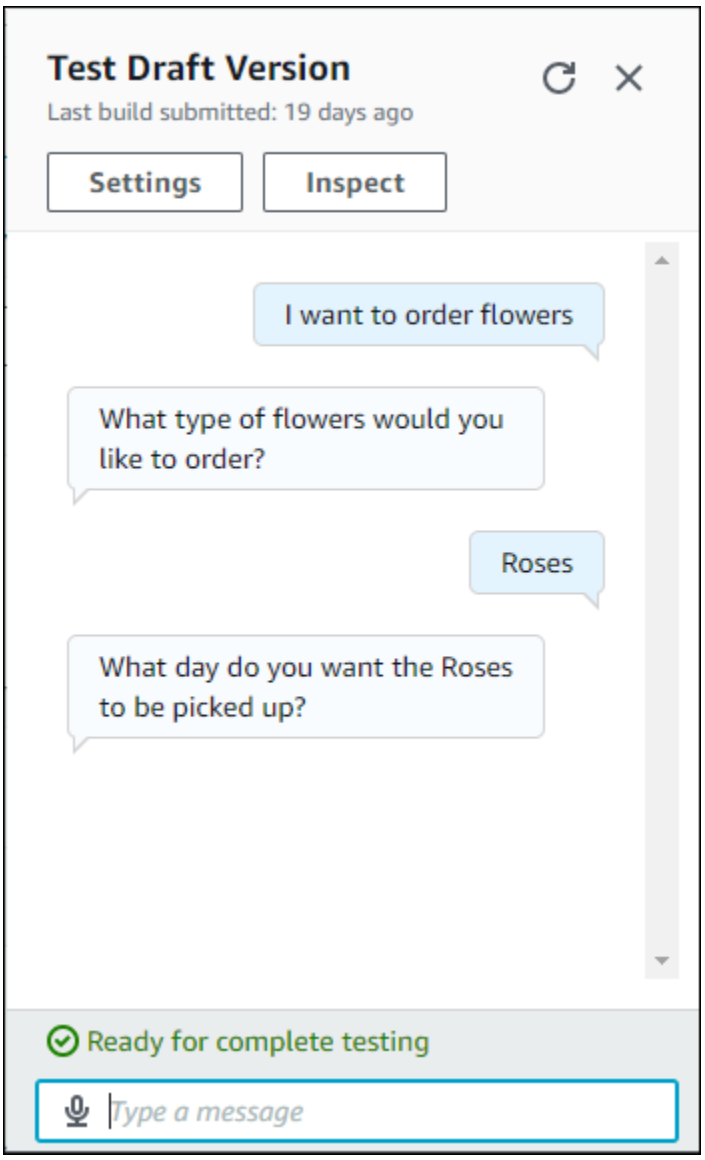
Tipe kedua, pengujian lengkap, memungkinkan Anda untuk menguji bot Anda menggunakan frasa yang terkait dengan ucapan yang Anda konfigurasi. Misalnya, Anda dapat menggunakan frasa “Bisakah saya memesan bunga” untuk memulai percakapan dengan bot Anda.

Anda menguji bot menggunakan alias dan bahasa tertentu. Jika Anda menguji versi pengembangan bot, Anda menggunakan `TestBotAlias` alias untuk pengujian.

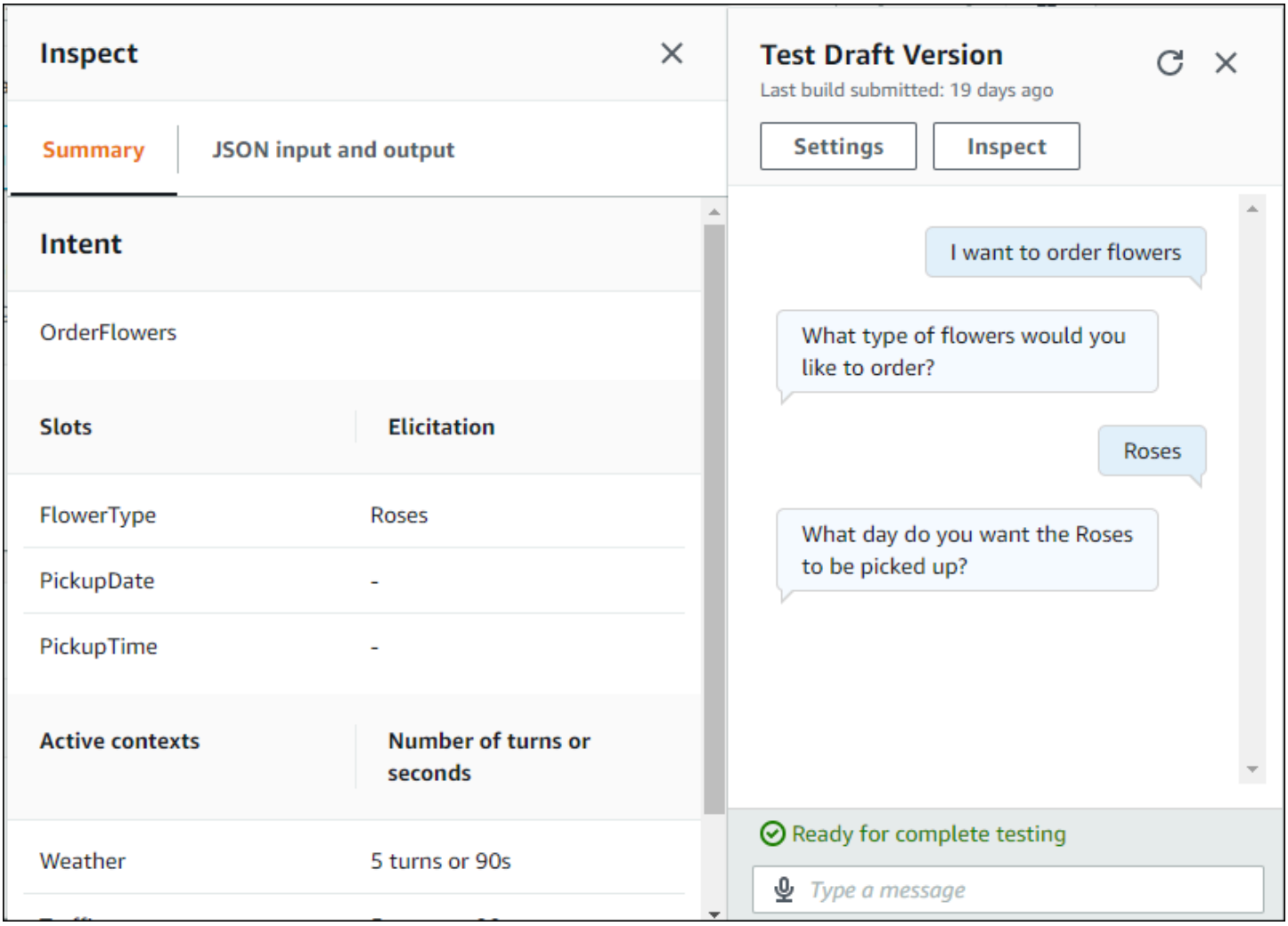
Untuk membuka jendela uji

1. Masuk ke AWS Management Console dan buka konsol Amazon Lex di https://console.aws.amazon.com/lex/.
2. Pilih bot yang akan diuji dari daftar bot.
3. Dari menu sebelah kiri, pilih Alias.
4. Dari daftar alias, pilih alias untuk diuji.
5. Dari Bahasa, pilih tombol radio bahasa yang akan diuji, lalu pilih Uji.

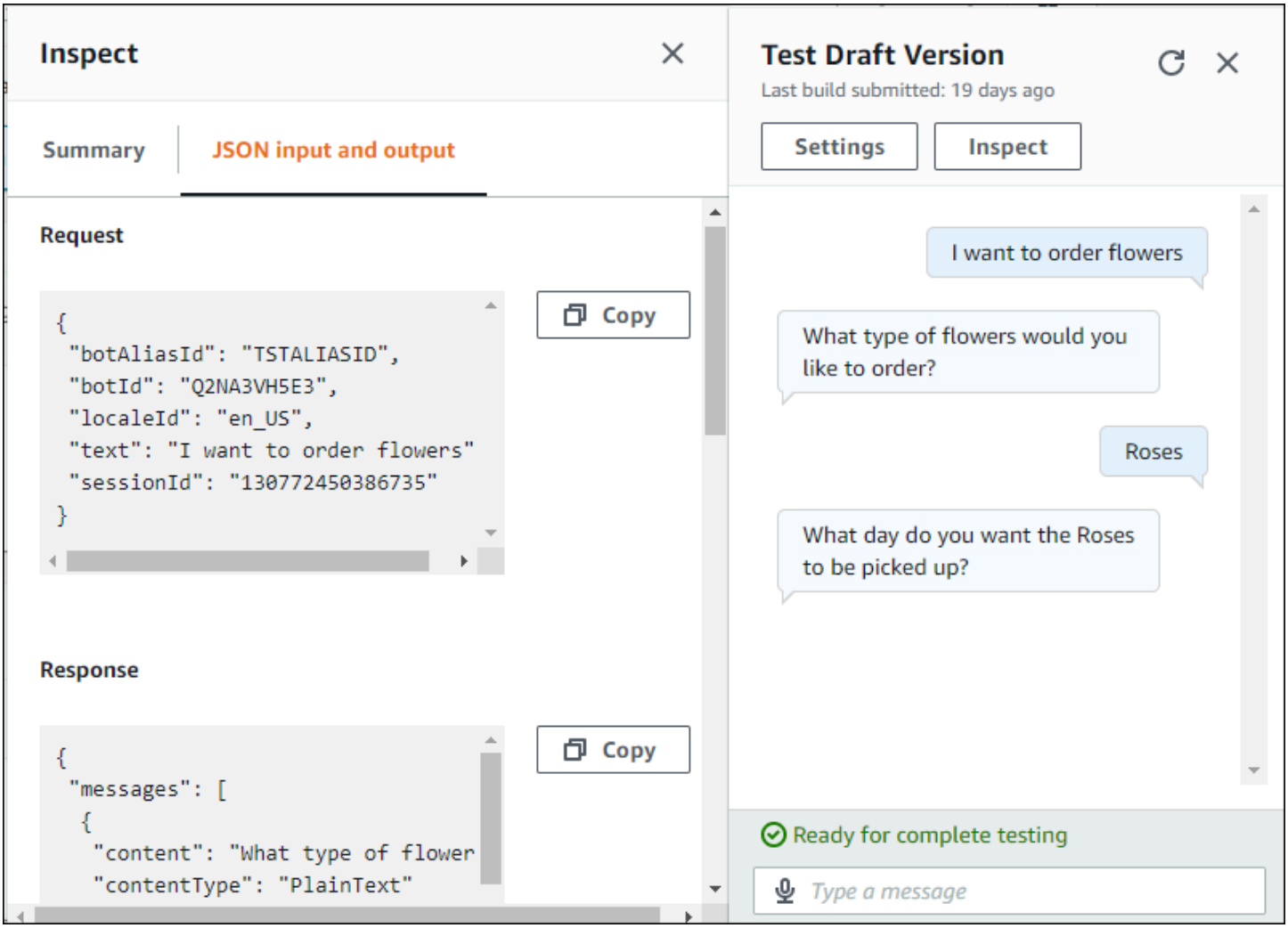
Setelah Anda memilih Uji, jendela tes terbuka di konsol. Anda dapat menggunakan jendela uji untuk berinteraksi dengan bot Anda, seperti yang ditunjukkan pada grafik berikut.



Selain percakapan, Anda juga dapat memilih Periksa di jendela pengujian untuk melihat respons yang dikembalikan dari bot. Tampilan pertama menunjukkan ringkasan informasi yang dikembalikan dari bot Anda ke jendela pengujian.



Anda juga dapat menggunakan jendela inspeksi pengujian untuk melihat struktur JSON yang dikirim antara bot dan jendela pengujian. Anda dapat melihat permintaan dari jendela pengujian dan respons dari Amazon Lex V2.


Inspect
Summary
JSON input and output
Request
{
"botAliasId": "TSTALIASID",
"botId": "Q2NA3VH5E3",
"localeId": "en_US",
"text": "I want to order flowers"
"sessionId": "130772450386735"
}
Copy
Response
{
"messages": [
{
"content": "What type of flower
"contentType": "PlainText"
Copy
Test Draft Version
Last build submitted: 19 days ago
Settings
Inspect
I want to order flowers
What type of flowers would you like to order?
Roses
What day do you want the Roses to be picked up?
Ready for complete testing
Type a message

Optimalkan pembuatan dan kinerja bot Lex V2 dengan menggunakan AI generatif

Manfaatkan kemampuan AI generatif Amazon Bedrock untuk mengotomatiskan dan mempercepat proses pembuatan bot Amazon Lex V2 Anda. Anda dapat melakukan proses berikut dengan bantuan Amazon Bedrock.

> **Note**
>
> Fitur-fitur ini menggunakan AI generatif. Saat Anda menggunakan layanan, ingatlah bahwa itu mungkin memberikan tanggapan yang tidak akurat atau tidak pantas. Untuk informasi selengkapnya, lihat Kebijakan AI Bertanggung Jawab AWS.
> Didukung oleh Amazon Bedrock: AWS menerapkan deteksi penyalahgunaan otomatis. Karena fitur AI generatif Amazon Lex V2 dibangun di Amazon Bedrock, pengguna mewarisi kontrol yang diterapkan di Amazon Bedrock untuk menegakkan keselamatan, keamanan, dan penggunaan AI yang bertanggung jawab.

- Buat bot baru dan isi dengan maksud dan jenis slot yang relevan secara efisien menggunakan deskripsi bahasa alami.
- Secara otomatis menghasilkan contoh ucapan untuk maksud bot Anda.
- Tingkatkan kinerja resolusi slot bot Anda.
- Buat maksud untuk membantu menjawab pertanyaan pelanggan Anda.
- Gunakan Amazon Bedrock Agents dan Amazon Bedrock Knowledge Bases untuk membantu menjawab pertanyaan pelanggan Anda.

Anda dapat mengaktifkan kemampuan AI generatif untuk Amazon Lex V2 baik melalui konsol atau API.

> **Note**
>
> Sebelum Anda dapat memanfaatkan fitur AI generatif, Anda harus memenuhi prasyarat berikut

1. Untuk informasi tentang harga untuk menggunakan Amazon Bedrock, lihat harga Amazon Bedrock.
2. Aktifkan kemampuan AI generatif untuk lokal bot Anda. Untuk melakukannya, ikuti langkah-langkah diOptimalkan pembuatan dan kinerja bot Lex V2 dengan menggunakan AI generatif.

Using the console

1. Masuk ke AWS Management Console dan buka konsol Amazon Lex V2 di https://console.aws.amazon.com/lexv2/rumah.
2. Pilih bot dan lokal di bot yang ingin Anda aktifkan kemampuan AI generatif.
3. Di bagian Generative AI configurations, pilih Configure.
4. Alihkan tombol Diaktifkan untuk setiap fitur yang ingin Anda aktifkan. Pilih model dan versi yang ingin Anda gunakan untuk fitur itu. Mengaktifkan fitur dapat dikenakan biaya tambahan. Untuk informasi tentang harga untuk menggunakan Amazon Bedrock, lihat harga Amazon Bedrock. Untuk mempelajari lebih lanjut tentang fitur, pilih topik yang sesuai dari daftar di bawah ini. Pilih Simpan setelah Anda mengaktifkan fitur yang ingin Anda aktifkan. Spanduk sukses hijau muncul untuk mengkonfirmasi bahwa kemampuan dihidupkan.

Using the API

1. Untuk mengaktifkan kemampuan AI generatif untuk bot baru, gunakan CreateBotoperasi untuk membuat bot baru.
2. Kirim CreateBotLocalepermintaan, modifikasi `generativeAISettings` objek seperlunya. Jika Anda mengaktifkan kemampuan untuk bot yang ada, kirim UpdateBotLocalepermintaan sebagai gantinya.

    a. Untuk mengaktifkan penggunaan pembuat bot deskriptif, modifikasi `descriptiveBotBuilder` objek. Tentukan model pondasi yang akan digunakan di `modelArn` lapangan dan atur `enabled` nilainya`True`.

    b. Untuk mengaktifkan peningkatan resolusi slot, modifikasi `slotResolutionImprovement` objek. Tentukan model pondasi yang akan digunakan di `modelArn` lapangan dan atur `enabled` nilainya`True`.

c. Untuk mengaktifkan pembuatan ucapan sampel, modifikasi objek. `sampleUtteranceGeneration` Tentukan model pondasi yang akan digunakan di `modelArn` lapangan dan atur `enabled` nilainya`True`.

Topik

- Gunakan deskripsi untuk membangun bot di Lex V2 dengan pembuat bot deskriptif
- Gunakan generasi ucapan untuk menghasilkan contoh ucapan untuk pengenalan maksud
- Menggunakan resolusi slot berbantuan untuk memperjelas nilai slot di Amazon Lex V2
- Menggunakan BedrockAgentIntent untuk menggunakan Agen Batuan Dasar Amazon di Amazon Lex V2
- Tingkatkan klasifikasi maksud dan resolusi slot di Lex V2 dengan bantuan NLU
- Selesaikan masukan pengguna yang ambigu dengan Disambiguasi Intent
- Optimalkan Bot menggunakan Bot Analyzer yang didukung AI
- Amazon.qn AIntent

Gunakan deskripsi untuk membangun bot di Lex V2 dengan pembuat bot deskriptif

> **Note**
>
> Sebelum Anda dapat memanfaatkan fitur AI generatif, Anda harus memenuhi prasyarat berikut
>
> 1. Untuk informasi tentang harga untuk menggunakan Amazon Bedrock, lihat harga Amazon Bedrock.
> 2. Aktifkan kemampuan AI generatif untuk lokal bot Anda. Untuk melakukannya, ikuti langkah-langkah diOptimalkan pembuatan dan kinerja bot Lex V2 dengan menggunakan AI generatif.

Pembangun bot deskriptif memungkinkan Anda memanfaatkan akses Amazon Bedrock ke model bahasa besar untuk meningkatkan efisiensi proses pembuatan bot. Anda memberikan prompt menggunakan bahasa alami yang mencakup tujuan bot dan tindakan yang harus dilakukan. Amazon Lex V2 memanfaatkan kemampuan Amazon Bedrock untuk menghasilkan maksud dan jenis slot

yang relevan untuk bot Anda berdasarkan deskripsi Anda. Setelah Anda memilih maksud dan jenis slot yang ingin Anda simpan, Anda kemudian dapat mengulangi bot untuk memodifikasinya ke kasus penggunaan spesifik Anda. Pembangun bot deskriptif menghemat waktu Anda dengan membiarkan Anda menghindari keharusan membuat maksud dan jenis slot untuk bot secara manual.

Pembuat bot deskriptif tersedia di lokal bahasa Inggris (lihat lokal yang dimulai dengan tabel `en_` di). Bahasa dan lokal yang didukung oleh Amazon Lex V2

Sebelum Anda membuat bot Anda, lakukan hal berikut.

1. Periksa apakah peran Anda memiliki izin yang benar dengan meninjau langkah-langkah di. Izin diperlukan untuk membuat bot dengan deskripsi bahasa alami di Lex V2
2. Tentukan deskripsi yang akan digunakan. Anda dapat merujuk ke Contoh deskripsi bot untuk pembuat bot deskriptif untuk deskripsi bot sampel.

Buat bot dengan menggunakan bahasa alami untuk menggambarkan apa yang seharusnya bisa dilakukan bot. Amazon Lex V2 memanggil model Amazon Bedrock untuk menghasilkan maksud dan jenis slot yang sesuai dengan kasus penggunaan bot Anda. Anda dapat membuat bot dengan konsol atau API.

Console

Buat bot menggunakan pembuat bot deskriptif

1. Masuk ke AWS Management Console dan buka konsol Amazon Lex V2 di https://console.aws.amazon.com/lexv2/rumah.
2. Di halaman Bots, pilih Buat bot.
3. Untuk metode Creation, pilih Descriptive Bot Builder - GenAI.
4. Beri bot Anda nama dan deskripsi opsional, konfigurasikan izin IAM, dan pilih apakah bot Anda tunduk pada COPPA atau tidak. Kemudian pilih Berikutnya.
5. Pilih bahasa untuk membuat bot, suara untuk bot, dan ambang kepercayaan untuk klasifikasi maksud (untuk informasi selengkapnya, lihatMenggunakan skor keyakinan niat untuk meningkatkan pemilihan maksud dengan Lex V2.
6. Di bawah Descriptive Bot Builder - GenAI, berikan deskripsi untuk bot yang ingin Anda buat. Deskripsi Anda harus terperinci dan tepat untuk membantu menghasilkan maksud yang sesuai dan memadai untuk bot Anda. Sertakan daftar tindakan untuk meningkatkan proses pembuatan maksud.

7. Pilih penyedia model dan model di bawah Pilih model.
8. Untuk membuat bot di lokal lain, pilih Tambahkan bahasa lain. Setelah selesai menambahkan bahasa, pilih Selesai. Amazon Lex V2 membuat bot Anda dan pembuat bot deskriptif menghasilkan maksud dan slot untuk itu. Ketika lokal telah dihasilkan, spanduk berubah dari biru menjadi hijau. Pilih Tinjau untuk melihat maksud dan jenis slot yang dihasilkan.

> **Note**
>
> Pembuat bot deskriptif saat ini hanya tersedia di lokal bahasa Inggris. Namun, Anda dapat menyalin bot ke lokal non-Inggris setelah membuatnya.

Tinjau maksud dan jenis slot yang dihasilkan dan tambahkan ke bot Anda

1. Jika ada cukup maksud dan jenis slot yang berlaku untuk kasus penggunaan bot Anda, Anda dapat meninjau maksud yang dihasilkan.
   a. Tinjau maksud yang dihasilkan.
      i. Pilih kotak centang di sebelah maksud untuk menghapusnya dari daftar maksud untuk ditambahkan ke bot.
      ii. Pilih nama maksud untuk melihat ucapan Sampel dan Slot yang dihasilkan untuk maksud tersebut.
      iii. Secara default, semua ucapan dan slot dipilih. Pilih kotak centang untuk menghapus item tersebut dari intent. Pilih Tambahkan ke pilihan untuk menyimpan item yang dicentang dalam maksud.
   b. Tinjau jenis slot yang dihasilkan.
      i. Pilih kotak centang di sebelah jenis slot untuk menghapusnya dari daftar maksud untuk ditambahkan ke bot.
      ii. Anda dapat menambahkan nilai ke jenis slot setelah Anda menambahkannya ke bot
2. Ketika Anda puas dengan maksud dan jenis slot Anda, pilih Tambahkan maksud dan jenis slot di bagian atas halaman untuk menambahkan maksud dan jenis slot ke bot Anda.
3. Ketika sumber daya selesai ditambahkan, spanduk sukses hijau muncul. Buka jenis Intent dan Slot untuk mengedit yang dihasilkan dan untuk menambahkan lebih banyak nilai.
4. Jika maksud yang Dihasilkan dan jenis slot yang Dihasilkan sebagian besar tidak dapat diterapkan pada bot yang ingin Anda buat, lakukan langkah-langkah berikut.

a. Pilih Generasi baru di bagian Detail pembuat bot deskriptif.

b. Tulis ulang prompt dan pilih Buat ulang untuk menghasilkan maksud dan jenis slot baru. Hasilnya berbeda jika Anda menggunakan model yang berbeda.

> **Important**
>
> Tidak ada jaminan bahwa maksud dan slot yang sama akan dihasilkan. Anda dikenakan biaya setiap kali Anda meregenerasi maksud dan jenis slot.

API

Buat bot menggunakan deskripsi bahasa alami

Saat Anda menggunakan pembuat bot deskriptif melalui API, itu membuat definisi bot dalam file.zip di bucket Amazon S3. Anda mengunduh file ini dan mengimpor definisi bot ke Amazon Lex V2 untuk membuat bot Anda.

1. Kirim CreateBotpermintaan untuk membuat bot baru. Kemudian kirim CreateBotLocalepermintaan untuk membuat lokal untuk bot.
2. Kirim StartBotResourceGenerationpermintaan, tentukan ID, versi, dan lokal bot. Anda dapat menggunakan `DRAFT` untuk versi bot. Berikan prompt Anda di `generationInputPrompt` lapangan. Deskripsi Anda harus terperinci dan tepat untuk membantu menghasilkan maksud yang sesuai dan memadai untuk bot Anda. Sertakan daftar tindakan untuk meningkatkan proses pembuatan maksud.
3. Catat `generationId` dalam tanggapannya.
4. Kirim DescribeBotResourceGenerationpermintaan menggunakan yang `generationId` Anda terima dalam `StartBotResourceGeneration` tanggapan. Sertakan ID bot, versi, dan lokal.
5. Jika `generationStatus` dalam `DescribeBotResourceGeneration` responsnya`Complete`, `generatedBotLocaleUrl` bidang juga akan diisi. Gunakan URI Amazon S3 ini untuk mengunduh definisi bot dengan mengikuti langkah-langkah di Mengunduh objek.

Periksa definisi bot yang dihasilkan dan impor

1. Gunakan URI Amazon S3 dari `generationStatus` dalam `DescribeBotResourceGeneration` respons untuk mengunduh definisi bot dengan mengikuti langkah-langkah di Mengunduh objek.
2. Anda dapat langsung memodifikasi konten yang dihasilkan untuk kasus penggunaan khusus bot Anda dengan mengedit file. Anda juga dapat mengirim `StartBotResourceGeneration` permintaan lain untuk meregenerasi maksud dan slot.

   > **Important**
   >
   > Tidak ada jaminan bahwa maksud dan slot yang sama akan dihasilkan. Anda dikenakan biaya setiap kali Anda meregenerasi maksud dan jenis slot.

3. Untuk mengimpor definisi bot, ikuti langkah-langkah diMengimpor bot di Lex V2.
4. Setelah mengimpor, Anda dapat memodifikasi maksud dan slot yang dihasilkan dengan menggunakan UpdateIntent, UpdateSlot, dan UpdateSlotTypeoperasi.

Untuk membuat daftar metadata tentang semua item yang dihasilkan untuk lokal bot, gunakan operasi. ListBotResourceGenerations Gunakan salah satu `generationId` nilai yang dikembalikan dalam `DescribeBotResourcGeneration` permintaan untuk mengambil URI Amazon S3 untuk definisi bot yang dihasilkan.

Topik

- Contoh deskripsi bot untuk pembuat bot deskriptif
- Izin diperlukan untuk membuat bot dengan deskripsi bahasa alami di Lex V2

Contoh deskripsi bot untuk pembuat bot deskriptif

Berikut adalah beberapa contoh deskripsi bot bermanfaat yang dapat Anda gunakan dengan pembuat bot deskriptif di Amazon Lex V2.

Industri	Contoh prompt
Jasa keuangan	Kami adalah layanan kartu keuangan yang membantu pengguna melakukan tugas-tug as ketika mereka menerima kartu baru seperti

Industri	Contoh prompt
	mengaktifkan kartu, email atau mengirim pin, memverifikasi kartu baru (menggunakan kode pos). Kami juga membantu mereka dengan tugas-tugas yang terkait dengan kartu mereka yang ada, seperti menanyakan tentang manfaat kartu kredit, melaporkan kartu yang hilang, meminta kartu baru, mengatur ulang pin kartu, atau membayar tagihan kartu.
Layanan makanan	Saya ingin bot membantu pelanggan memesan makanan (menggunakan ID item, jumlah, ukuran), memeriksa status pesanan, dan membatalkan pesanan. Gunakan ID Pesanan untuk mengindeks pesanan.
Maskapai	Kami adalah domain maskapai penerbang an yang membantu pengguna memesan tiket penerbangan, memeriksa detail reservasi, mendapatkan tanda terima untuk penerbangan yang dipesan, menanyakan status penerbang an, menjadwal ulang penerbangan yang dipesan, mendapatkan detail penerbangan, dan membatalkan penerbangan yang dipesan. Anda juga dapat menghasilkan intent tambahan jika mereka membantu mendukung fungsi dalam deskripsi domain.

Industri	Contoh prompt
Asuransi	Tujuan: Kami adalah perusahaan asuransi yang menjual polis asuransi mobil, rumah dan anuitas. Saya ingin bot yang dapat memeriksa status klaim, mengajukan klaim, melakukan pembayaran kebijakan, dan membatalkan kebijakan. Kami menggunakan policy_id dan 4 SSN terakhir untuk identifikasi dan validasi akun. Saya berharap bot memiliki setidaknya maksud dan slot berikut: otentikasi - policy_id , SSNpolicy tipe last4: mobil, rumah, anuitista tus kebijakan: periksa saldo, periksa tanggal jatuh tempo, periksa pertanggunganmelakukan pembayaran: pembayaran satu kali, angsuran, jumlah
Manajemen kendaraan	Kami sedang membangun bot Pencarian Mobil Derek yang membantu pengemudi di kota yang mobilnya telah ditarik untuk menemukan di mana mobil itu berada. Bot ini harus menanyakan alamat atau lokasi dari mana mobil ditarik, dan rincian tentang kendaraan seperti plat nomor dan merek, model, dan tahun mobil. Bot harus membalas dengan lokasi tempat parkir mobil yang ditarik, dan jam operasi.

Industri	Contoh prompt
Perjalanan	Saya seorang agen perjalanan dan saya ingin bot untuk membantu pelanggan saya memesan perjalanan ke Disney. Disney memiliki beberapa taman di seluruh dunia untuk dipilih, dan juga memiliki hotel, makan, dan hiburan khusus yang dapat dipesan. Pengguna bot harus dapat mengubah atau membatalkan pemesanan mereka. Pemesanan harus mencakup minimal taman, tanggal, dan hotel. Termasuk makan atau hiburan adalah opsional dan dapat ditambahkan atau diubah nanti.

Izin diperlukan untuk membuat bot dengan deskripsi bahasa alami di Lex V2

- Untuk mengakses fitur ini di konsol Amazon Lex V2, pastikan peran konsol Anda memiliki `bedrock:ListFoundationModels` dan `bedrock:ListInferenceProfiles` izin.
- Peran IAM yang terkait dengan bot harus memiliki `bedrock:InvokeModel` izin. Saat Anda mengaktifkan fitur dengan konsol Amazon Lex, kebijakan akan ditambahkan secara otomatis ke peran bot asalkan bot Anda menggunakan peran terkait layanan yang dihasilkan oleh Amazon Lex.

  JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "bedrock:InvokeModel"
            ],
            "Resource": [
                "arn:aws:bedrock:us-east-1::foundation-model/model-id"
            ]
        }
```

```
    ]
}
```

Gunakan generasi ucapan untuk menghasilkan contoh ucapan untuk pengenalan maksud

> **Note**
>
> Sebelum Anda dapat memanfaatkan fitur AI generatif, Anda harus memenuhi prasyarat berikut
>
> 1. Untuk informasi tentang harga untuk menggunakan Amazon Bedrock, lihat harga Amazon Bedrock.
> 2. Aktifkan kemampuan AI generatif untuk lokal bot Anda. Untuk melakukannya, ikuti langkah-langkah diOptimalkan pembuatan dan kinerja bot Lex V2 dengan menggunakan AI generatif.

Gunakan pembuatan ucapan untuk mengotomatiskan pembuatan contoh ucapan untuk maksud Anda. Alih-alih memasukkan contoh ucapan secara manual, Amazon Lex V2 menghasilkan contoh ucapan untuk Anda berdasarkan nama maksud, deskripsi, dan contoh ucapan yang ada, sehingga Anda dapat mengurangi waktu dan upaya yang Anda habiskan dalam menemukan dan menulis ucapan sampel Anda sendiri. Setelah Amazon Lex V2 menghasilkan ucapan, Anda dapat mengedit dan menghapus ucapannya. Gunakan alat ini untuk mempercepat pembuatan ucapan sampel untuk proses pengenalan maksud.

Untuk memungkinkan generasi ucapan, ikuti langkah-langkah di Optimalkan pembuatan dan kinerja bot Lex V2 dengan menggunakan AI generatif untuk mengaktifkan kemampuan AI generatif.

Untuk mengakses fitur ini di konsol Amazon Lex V2, pastikan peran konsol Anda memiliki `bedrock:ListFoundationModelsbedrock:ListInferenceProfiles`, dan `bedrock:InvokeModel` izin.

Anda dapat menghasilkan ucapan dengan konsol atau API.

Console

1. Arahkan ke bagian Contoh ucapan dari maksud apa pun di bot Anda (di pembuat percakapan Visual, ada di blok Mulai).
2. Pilih tombol Generate utterances untuk menghasilkan 5 contoh ucapan. Jika maksud Anda memiliki lebih dari 25 contoh ucapan, tombol Hasilkan ucapan menjadi dinonaktifkan.
3. Ucapan yang dihasilkan ditampilkan dengan spanduk hijau yang membedakan ucapan yang dihasilkan dari ucapan yang ada.
4. Arahkan kursor ke ucapan untuk menampilkan opsi untuk mengedit, menghapus, dan mengurutkan ucapan yang dihasilkan.

API

1. Kirim GenerateBotElementpermintaan, isi maksud dan ID bot, versi, dan lokal yang ingin Anda hasilkan contoh ucapan.
2. Respons mengembalikan daftar SampleUtteranceobjek, yang masing-masing berisi ucapan yang dihasilkan.
3. Untuk menambahkan ucapan ke intent, kirim UpdateIntentpermintaan dan tambahkan ucapan ke bidang. `sampleUtterances`

Menggunakan resolusi slot berbantuan untuk memperjelas nilai slot di Amazon Lex V2

> **Note**
>
> Sebelum Anda dapat memanfaatkan fitur AI generatif, Anda harus memenuhi prasyarat berikut
>
> 1. Untuk informasi tentang harga untuk menggunakan Amazon Bedrock, lihat harga Amazon Bedrock.
> 2. Aktifkan kemampuan AI generatif untuk lokal bot Anda. Untuk melakukannya, ikuti langkah-langkah diOptimalkan pembuatan dan kinerja bot Lex V2 dengan menggunakan AI generatif.

Anda dapat meningkatkan akurasi beberapa slot bawaan dalam alur percakapan bot Anda dengan menggunakan resolusi slot berbantuan. Resolusi slot berbantuan menggunakan model bahasa besar Amazon Bedrock (LLMs) untuk meningkatkan pengenalan beberapa slot bawaan, yang menghasilkan interpretasi yang lebih baik dari respons pelanggan selama elisitasi slot. Untuk ucapan yang tidak dapat diselesaikan secara normal, Amazon Lex V2 akan mencoba menyelesaikannya untuk kedua kalinya menggunakan Amazon Bedrock.

Resolusi slot berbantuan memungkinkan Anda menggunakan kekuatan model fondasi Amazon Bedrock untuk meningkatkan akurasi slot bawaan berikut:

- `AMAZON.Alphanumeric`tanpa dukungan regex
- `AMAZON.City`
- `AMAZON.Country`
- `AMAZON.Date`
- `AMAZON.Number`
- `AMAZON.PhoneNumber`
- `AMAZON.Confirmation`

Anda dapat mengaktifkan resolusi slot berbantuan untuk maksud apa pun yang menggunakan slot bawaan yang tercantum di atas. Resolusi slot berbantuan tidak berlaku untuk slot khusus atau slot bawaan Amazon yang tidak tercantum di atas.

Anda dapat mengumpulkan data tentang peningkatan akurasi setelah Anda mengaktifkan resolusi slot berbantuan di bot Amazon Lex V2 Anda dengan menggunakan log dan metrik percakapan.

- Log percakapan - Interpretasi akan memiliki `interpretationSource` sepertiBedrock, jika Amazon Bedrock digunakan untuk menyelesaikan slot.
- CloudWatch metrik - Metrik akan diterbitkan di bawah dimensi yang tercantum dalam metrik. CloudWatch Untuk mempelajari lebih lanjut, lihat Memantau Amazon Lex dengan Amazon CloudWatch.

Untuk menggunakan pembuat bot deskriptif, pastikan bahwa peran IAM Anda memiliki izin yang tepat dengan mengikuti langkah-langkah di. Izin diperlukan di Lex V2 untuk resolusi slot berbantuan

Topik

- Contoh resolusi slot berbantuan yang digunakan dalam Lex V2

- Aktifkan resolusi slot berbantuan di layar konfigurasi Generative AI
- Aktifkan resolusi slot berbantuan dalam pengaturan slot di Lex V2
- Izin diperlukan di Lex V2 untuk resolusi slot berbantuan

Contoh resolusi slot berbantuan yang digunakan dalam Lex V2

Di bawah ini adalah beberapa contoh di mana resolusi slot berbantuan mampu secara cerdas menyelesaikan ucapan pengguna menjadi nilai.

Amazon.number

Vertikal	SlotType	slotName	SlotPrompt	ucapan	Nilai Terselesaikan
Perjalanan	Amazon.number	numberOfNightsTinggal	Berapa malam Anda menginap untuk perjalanan?	Seminggu penuh, 7 malam.	7
Perbankan	Amazon.number	numberOfPeopleOnTheAccount	Berapa banyak orang yang ada di akun?	Aku dan istriku.	2
Perjalanan	Amazon.number	numberOfStops	Berapa banyak berhenti?	Sekali di Jepang. Sekali di LA.	2

AMAZON. AlphaNumeric

Vertikal	SlotType	slotName	SlotPrompt	ucapan	Nilai Terselesaikan
Penyewaan Mobil	Amazon.alfanumerik	TransactionID	Apa id transaksi Anda?	Saya percaya itu wiski alfa gema delapan tiga empat sembilan romeo juliet.	AWE8349RJ
Perjalanan	Amazon.alfanumerik	Kode Konfirmasi	Berapa nomor konfirmasi untuk reservasi Anda?	Nomor konfirmasi adalah BLT2UE.	BLT2UE

Amazon.tanggal

Vertikal	SlotType	slotName	SlotPrompt	ucapan	Nilai Terselesaikan	CurrentDate
Penyewaan Mobil	Amazon.tanggal	DueDate	Kapan perjanjian sewa akan kedaluwarsa?	Sewa naik pada tanggal 1 bulan depan.	2023-12-01	2023-11-09
Perjalanan	Amazon.tanggal	Tanggal Pengembalian	Kapan kau kembali?	Kemudian hari ini sekitar 7.	2023-11-09	2023-11-09

AMAZON. PhoneNumber

Vertikal	SlotType	slotName	SlotPrompt	ucapan	Nilai Terselesaikan
Asuransi	AMAZON.PhoneNumber	Pemegang Polis	Berapa nomor telepon pemegang polis?	Nomor telepon untuk pemegang polis adalah 123-456-7890.	1234567890
Eceran	AMAZON.PhoneNumber	PhoneLookup	Berapa nomor telepon Anda sehingga saya dapat menemukan akun Anda?	Saya pikir itu di bawah 413-570-9617, biarkan saya periksa kembali.	4135709617

Amazon.negara

Vertikal	SlotType	slotName	SlotPrompt	ucapan	Nilai Terselesaikan
Perjalanan	Amazon.negara	NativeCountry	Apa negara asal Anda?	Saya orang India.	India
Perbankan	Amazon.negara	CountryItinerary	Negara mana yang akan Anda kunjungi dengan kartu debit Anda?	Saya akan bepergian ke New Delhi.	India

Amazon.kota

Vertikal	Jenis Slot	Niat	Pertanyaan	Respons	Nilai Terselesaikan
Asuransi	Amazon.kota	policyHolderCity	Di kota mana pemegang polis tinggal?	Saya tinggal di Springfield.	Springfield
Perjalanan	Amazon.kota	DestinasiKota	Kota mana yang Anda kunjungi?	Saya bepergian ke Tokyo.	Tokyo

Amazon.konfirmasi

Vertikal	SlotType	slotName	SlotPrompt	ucapan	Nilai Terselesaikan
Asuransi	Amazon.konfirmasi	Kebijakan Kedaluwarsa	Apakah polis asuransi sudah kedaluwarsa?	Ya, sayangnya sudah kedaluwarsa.	Ya
Perbankan	Amazon.konfirmasi	HasInvestasi	Apakah Anda memiliki investasi?	Saya belum berinvestasi dalam apa pun.	Tidak

Aktifkan resolusi slot berbantuan di layar konfigurasi Generative AI

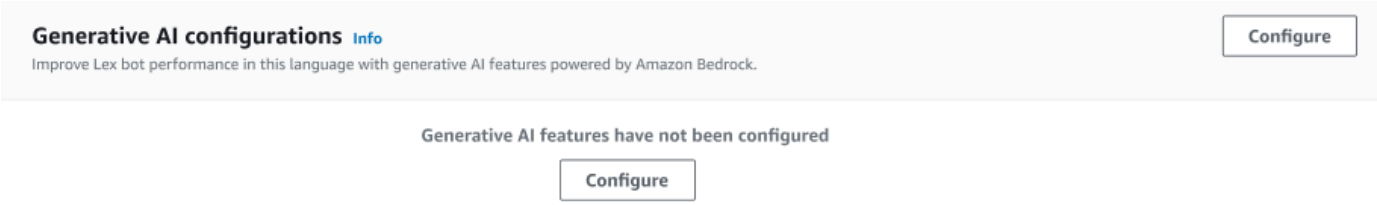
Anda dapat mengaktifkan resolusi slot berbantuan untuk slot bawaan yang didukung dengan menavigasi ke layar Generative AI.

Jika slot adalah slot bawaan yang didukung, Anda akan memiliki opsi untuk mengaktifkan resolusi slot yang dibantu di tingkat slot.

1. Masuk ke AWS Management Console dan buka konsol Amazon Lex V2 di https://console.aws.amazon.com/lexv2/rumah.
2. Di panel navigasi di bawah Bots, pilih bot yang ingin Anda gunakan untuk resolusi slot berbantuan.
3. Pilih bahasa Inggris (AS) untuk bot yang ingin Anda aktifkan.
4. Buka bagian konfigurasi Generative AI di layar.
5. Pilih Buka Amazon Bedrock untuk mendaftar dan mengaktifkan fitur, jika fitur belum diaktifkan.

> **Note**
>
> Jika Anda tidak memiliki akses ke model fondasi Amazon Bedrock, Anda harus melihat Pergi ke Amazon Bedrock. Klik Pergi ke Amazon Bedrock untuk pergi ke halaman Amazon Bedrock di mana Anda dapat mendaftar untuk akses ke model foundation. Resolusi slot berbantuan saat ini mendukung Anthropic Claude. Kami menyarankan menggunakan Anthropic Claude untuk hasil terbaik.

6. Jika Anda sudah memiliki akses ke model Amazon Bedrock Foundation, Anda akan melihat tombol Configure. Klik tombol ini untuk membuka halaman konfigurasi AI generatif untuk mengaktifkan fitur AI generatif di Lex.



7. Di sudut kanan atas kotak, gerakkan slider ke kanan untuk memilih pengaturan Diaktifkan.
8. Pilih tombol Aktifkan untuk mengaktifkan resolusi slot berbantuan untuk slot yang dipilih.
9. Anda dapat menonaktifkan resolusi slot berbantuan dengan memilih slot dari daftar dan memilih tombol Nonaktifkan.

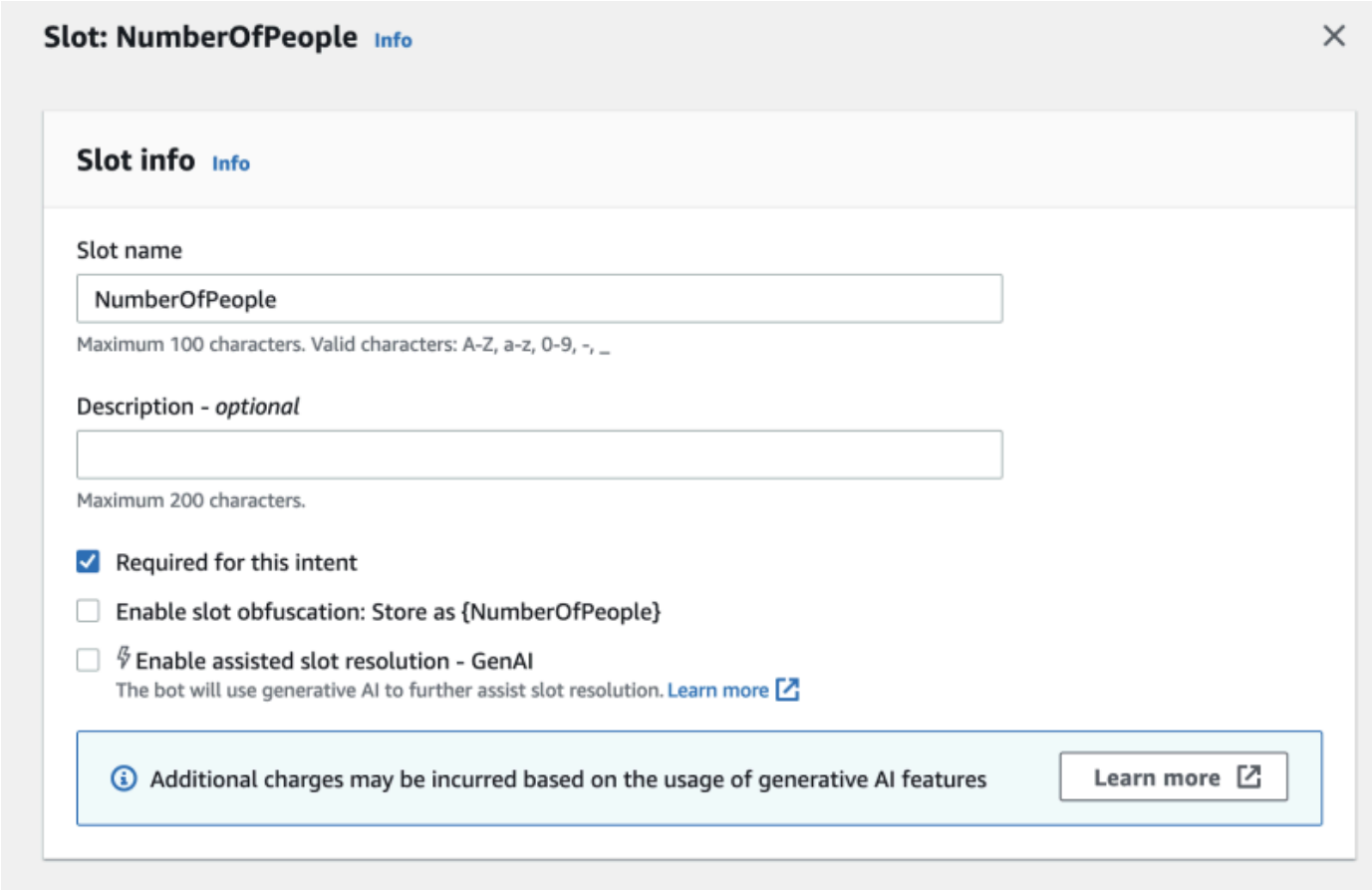
Aktifkan resolusi slot berbantuan dalam pengaturan slot di Lex V2

Anda dapat mengaktifkan resolusi slot berbantuan untuk slot bawaan yang didukung dengan menavigasi ke level slot untuk setiap maksud yang memiliki slot. Slot harus menjadi salah satu slot bawaan yang didukung yang tercantum di atas untuk memiliki opsi mengaktifkan resolusi slot

berbantuan. Jika slot tidak memiliki opsi untuk mengaktifkan resolusi slot berbantuan, opsi akan berwarna abu-abu.

> **Note**
>
> Anda harus terlebih dahulu mengaktifkan fitur resolusi slot berbantuan pada panel Generative AI untuk mengaktifkan fitur untuk slot individual.

1. Masuk ke AWS Management Console dan buka konsol Amazon Lex V2 di https://console.aws.amazon.com/lexv2/ rumah.
2. Di panel navigasi di bawah Bots, pilih bot yang ingin Anda gunakan untuk resolusi slot yang dibantu.
3. Di bawah Semua bahasa, pilih Bahasa Inggris (AS) untuk memperluas daftar.
4. Di panel sisi kiri, pilih Maksud untuk melihat daftar maksud di bot yang Anda pilih.
5. Di layar Maksud, pilih maksud yang berisi slot yang ingin Anda ubah.
6. Pilih nama maksud untuk melihat slot untuk maksud itu.
7. Pilih tombol Opsi Lanjutan di bagian Slot.
8. Pilih kotak centang untuk Aktifkan resolusi slot berbantuan untuk mengaktifkan fitur.



9. Pilih tombol Perbarui Slot di sudut kanan bawah layar. Ini akan mengaktifkan resolusi slot berbantuan untuk slot yang telah Anda pilih.

Anda dapat mengaktifkan resolusi slot berbantuan untuk slot bawaan yang didukung dengan melakukan panggilan API.

- Ikuti langkah-langkah di Optimalkan pembuatan dan kinerja bot Lex V2 dengan menggunakan AI generatif untuk mengaktifkan resolusi slot berbantuan untuk lokal bot Anda.
- Kirim UpdateSlotpermintaan, tentukan slot yang ingin Anda aktifkan resolusi slot berbantuan. Di `slotResolutionSetting` lapangan, tetapkan `slotResolutionStrategy` nilainya sebagai`EnhancedFallback`. Untuk membuat slot baru dengan resolusi slot berbantuan diaktifkan, kirim CreateSlotpermintaan sebagai gantinya.

Izin diperlukan di Lex V2 untuk resolusi slot berbantuan

- Untuk mengakses fitur ini di konsol Amazon Lex V2, pastikan peran konsol Anda memiliki `bedrock:ListFoundationModels` dan `bedrock:ListInferenceProfiles` izin.

- Peran IAM yang terkait dengan bot harus memiliki `bedrock:InvokeModel` izin. Saat Anda mengaktifkan fitur dengan konsol Amazon Lex V2, kebijakan akan ditambahkan secara otomatis ke peran bot asalkan bot Anda menggunakan peran terkait layanan yang dihasilkan oleh Amazon Lex V2.

  JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "bedrock:InvokeModel"
            ],
            "Resource": [
                "arn:aws:bedrock:us-east-1::foundation-model/modelId"
            ]
        }
    ]
}
```

Menggunakan BedrockAgentIntent untuk menggunakan Agen Batuan Dasar Amazon di Amazon Lex V2

> **Note**
>
> Sebelum Anda dapat memanfaatkan fitur AI generatif, Anda harus memenuhi prasyarat berikut
>
> 1. Untuk informasi tentang harga untuk menggunakan Amazon Bedrock, lihat harga Amazon Bedrock.
> 2. Aktifkan kemampuan AI generatif untuk lokal bot Anda. Untuk melakukannya, ikuti langkah-langkah diOptimalkan pembuatan dan kinerja bot Lex V2 dengan menggunakan AI generatif.

Anda dapat memanfaatkan Amazon Bedrock Agents untuk menangani beban kerja kompleks yang diminta oleh pelanggan tanpa harus melalui proses definisi tugas yang komprehensif. Amazon Lex V2 menawarkan built-in `AMAZON.BedrockAgentIntent` yang dapat Anda tambahkan ke bot Anda. Maksud ini memanfaatkan kemampuan AI generatif dari Amazon Bedrock dengan mengenali permintaan pelanggan, menganalisisnya, memberi alasan, dan akhirnya merespons. Ini juga memiliki kemampuan untuk mengajukan pertanyaan tindak lanjut untuk mencapai tugas yang diperlukan (misalnya, gambar Anda mendefinisikan agen ritel yang dapat memeriksa status pesanan pelanggan. Ketika pelanggan meminta status pesanan, agen pertama meminta `customerId` atau terkait `emailId` untuk mengambil rincian, dan akhirnya merespons dengan status pesanan yang benar). Anda juga dapat memutuskan untuk mengintegrasikan AMAZON Anda. BedrockAgentIntent dengan Basis Pengetahuan Batuan Dasar untuk secara langsung menjawab pertanyaan pelanggan.

Pastikan peran IAM Anda memiliki izin yang tepat untuk mengakses AMAZON. BedrockAgentIntent dengan mengikuti langkah-langkah di Izin untuk AMAZON. BedrockAgentIntent

Untuk memanfaatkan AMAZON. BedrockAgentIntent Anda harus telah mendirikan salah satu toko pengetahuan berikut.

- Agen Batuan Dasar Amazon — Untuk informasi selengkapnya, lihat Membuat Agen Batuan Dasar.
- Basis Pengetahuan Amazon Bedrock — Untuk informasi selengkapnya, lihat Membangun Basis Pengetahuan.

Untuk menggunakan AMAZON. BedrockAgentIntent, pastikan bahwa peran IAM Anda memiliki izin yang tepat dengan mengikuti langkah-langkah di Izin yang diperlukan di Lex V2 untuk Intent Agen Batuan Dasar.

Topik

- Aktifkan Intent Agen Batuan Dasar di layar konfigurasi Generative AI
- Aktifkan Intent Agen Batuan Dasar dengan menambahkan intent bawaan ke bot Anda
- Izin diperlukan di Lex V2 untuk Intent Agen Batuan Dasar
- Permintaan sampel dengan atribut sesi

Aktifkan Intent Agen Batuan Dasar di layar konfigurasi Generative AI

Anda dapat mengaktifkan Bedrock Agent Intent dengan menavigasi ke layar Generative AI.

1. Masuk ke AWS Management Console dan buka konsol Amazon Lex V2 di https://console.aws.amazon.com/lexv2/rumah.
2. Di panel navigasi di bawah Bots, pilih bot yang ingin Anda gunakan untuk Bedrock Agent Intent.
3. Pilih bahasa untuk bot yang ingin Anda aktifkan.
4. Buka bagian konfigurasi Generative AI di layar, dan pilih Configure.
5. Di bagian BedrockAgentIntent konfigurasi, pilih Buat BedrockAgent maksud.

Aktifkan Intent Agen Batuan Dasar dengan menambahkan intent bawaan ke bot Anda

Anda dapat mengaktifkan Intent Agen Batuan Dasar dengan menambahkan intent bawaan ke bot Amazon Lex V2 Anda.

> **Note**
>
> Anda harus terlebih dahulu mengaktifkan fitur Bedrock Agent Intent pada panel Generative AI untuk mengaktifkan fitur untuk bot individual.

1. Masuk ke AWS Management Console dan buka konsol Amazon Lex V2 di https://console.aws.amazon.com/lexv2/ rumah.
2. Di panel navigasi di bawah Bots, pilih bot yang ingin Anda gunakan untuk Intent Agen Batuan Dasar.
3. Di bawah Semua bahasa, pilih Bahasa Inggris (AS) untuk memperluas daftar.
4. Pilih Tambah maksud dan pilih Gunakan maksud bawaan dari menu tarik-turun.
5. Untuk detail lebih lanjut tentang konfigurasi untuk AMAZON. BedrockAgentIntent, lihatAMAZON.BedrockAgentIntent.

Izin diperlukan di Lex V2 untuk Intent Agen Batuan Dasar

- Untuk mengakses fitur ini di konsol Amazon Lex V2, pastikan peran konsol Anda memiliki `bedrock:ListFoundationModels` dan `bedrock:ListInferenceProfiles` izin.
- Peran IAM yang terkait dengan bot harus memiliki izin yang diperlukan untuk. `AMAZON.BedrockAgentIntent`

Peran bot harus memiliki izin untuk menelepon `bedrock:InvokeAgent` jika menggunakan Agen Amazon Bedrock, `bedrock:InvokeModel` dan `bedrock:RetrieveAndGenerate` jika menggunakan Pangkalan Pengetahuan Amazon Bedrock dalam intent. Anda juga harus melampirkan pernyataan untuk setiap agen yang Anda tentukan di bot Anda `AMAZON.BedrockAgentIntent` (lihat Izin untuk mengakses Agen Amazon Bedrock atau Izin untuk mengakses pernyataan Pangkalan Pengetahuan Amazon Bedrock dalam kebijakan di bawah).

Saat Anda mengaktifkan fitur dengan konsol Amazon Lex, kebijakan akan ditambahkan secara otomatis ke peran bot asalkan bot Anda menggunakan peran terkait layanan yang dihasilkan oleh Amazon Lex.

Permintaan sampel dengan atribut sesi

Contoh berikut menunjukkan cara memanggil AMAZON. BedrockAgentIntent dan menunjukkan atribut sesi dan permintaan yang diisi dalam respons. Atribut ini berisi data respons Agen Bedrock dan dapat digunakan untuk mengakses output agen, sumber Pangkalan Pengetahuan Amazon Bedrock, dan detail pemanggilan grup tindakan.

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

Dalam contoh ini, atribut sesi menunjukkan cara BedrockAgentIntent mengisi data respons termasuk jawaban agen, sumber Pangkalan Pengetahuan Amazon Bedrock yang digunakan, pemanggilan grup tindakan, dan detail pencarian Pangkalan Pengetahuan Amazon Bedrock yang digunakan untuk menghasilkan respons.

Untuk informasi selengkapnya, lihat AMAZON. BedrockAgentIntent.

Tingkatkan klasifikasi maksud dan resolusi slot di Lex V2 dengan bantuan NLU

Assisted NLU adalah fitur yang menggunakan Large Language Models (LLMs) untuk meningkatkan klasifikasi maksud Amazon Lex V2 dan kemampuan resolusi slot. Ini meningkatkan akurasi saat tetap berada dalam maksud dan slot yang dikonfigurasi bot Anda. Fitur ini tidak menghasilkan atau memodifikasi konten bot apa pun. Fitur ini membantu meningkatkan akurasi keseluruhan sistem NLU, menghasilkan pengalaman percakapan yang lebih mulus dan efektif bagi pengguna.

Fitur NLU yang dibantu tersedia dalam bahasa Inggris, Spanyol, Portugis, Catalan, Prancis, Italia, dan Jerman. Secara khusus, ini mendukung lokal yang dimulai denganen_,, pt_ (PT_BRes_, PT_pt), (Ca_ES), (fr_CA, fr_FR), ca_ (IT_IT), fr_ (de_AT, de_DE), it_ (Zh_CN, Zh_hk),, de_ dan. zh_ ja_JP ko_KR Untuk daftar lengkap lokal yang didukung, lihat tabel diBahasa dan lokal yang didukung oleh Amazon Lex V2.

Gunakan NLU yang dibantu untuk meningkatkan klasifikasi maksud dan resolusi slot. Amazon Lex V2 memanggil model Amazon Bedrock untuk membantu mengklasifikasikan maksud dan menyelesaikan jenis slot yang sesuai dengan kasus penggunaan bot Anda. Anda dapat mengaktifkan NLU yang dibantu untuk bot Anda dengan konsol.

Mode NLU yang Dibantu

Dalam mode Primer, Lex akan secara default menggunakan LLM sebagai sarana utama pemrosesan input pengguna untuk menentukan Intent pengguna, serta untuk mengisi nilai slot.

Dalam mode Fallback, Lex akan menggunakan LLM untuk menentukan maksud pengguna jika skor kepercayaan yang ditentukan oleh NLU lebih rendah dari ambang batas yang dikonfigurasi atau perutean ke FallbackIntent, serta untuk menentukan nilai slot dari input pengguna jika NLU tradisional tidak menangkap nilai.

Console

Menggunakan NLU yang dibantu dengan bot Amazon Lex V2 Anda

1. Masuk ke AWS Management Console dan buka konsol Amazon Lex V2 di https://console.aws.amazon.com/lexv2/rumah.
2. Di halaman Bots, pilih bot yang ingin Anda gunakan dengan NLU yang dibantu.
3. Pada halaman Bot Locale, klik Configure di bawah bagian Assisted NLU.

4. Di bawah bagian fitur AI generatif Runtime, Anda dapat melihat fitur Assisted NLU. Gunakan tombol sakelar di sampingnya untuk mengaktifkan fitur LLM Assisted NLU. Anda kemudian dapat memilih mode Primer atau Fallback, dan klik Simpan.
5. Verifikasi bahwa fitur LLM Assisted NLU diaktifkan di bawah bagian NLU Bantuan di halaman Bot Locale.
6. Bangun bot untuk melihat perubahan tercermin dalam bot Anda di Runtime.
7. Setelah pembuatan bot selesai, Anda dapat menggunakan panel uji di konsol atau menjalankan set pengujian untuk melihat peningkatan setelah mengaktifkan fitur LLM Assisted NLU.

Panduan untuk meningkatkan akurasi bot Anda saat menggunakan Fitur NLU Berbantuan LLM

Praktik terbaik berikut dapat membantu Anda memaksimalkan efektivitas fitur NLU yang Dibantu:

1. Make Intent Names Self-Exploratory — Gunakan nama yang segera menyampaikan tindakan atau tujuan dari maksud tersebut. Misalnya, jika Anda membuat maksud untuk memesan penerbangan, cukup sebut saja "BookFlight”.
2. Jaga Nama Tetap Bersih dan Sederhana — Hindari menambahkan awalan, sufiks, atau kata-kata yang tidak perlu ke maksud dan nama slot Anda. Elemen tambahan seperti “Dev” atau “Test” dapat membingungkan LLM dan membuat tujuannya kurang jelas.
3. Berikan Deskripsi Terperinci — Untuk setiap maksud dan slot khusus, sertakan deskripsi singkat namun informatif. Ini membantu menjelaskan penggunaan dan konteks spesifiknya, sehingga lebih mudah bagi manusia dan LLM untuk memahami tujuannya.

> **Note**
>
> Saat Anda mengaktifkan fitur ini, data Anda mungkin diproses di seluruh Wilayah AWS. Untuk informasi lebih lanjut tentang Inferensi Lintas Wilayah, lihat. https://docs.aws.amazon.com/bedrock/latest/userguide/cross-region-inference.html

> **Important**
>
> Aktifkan fitur ini dalam versi draf bot. Uji sebelum menggunakannya dalam alias produksi.

Menonaktifkan NLU yang Dibantu

Untuk menonaktifkan fitur NLU yang dibantu, ikuti langkah-langkah berikut:

1. Masuk ke AWS Management Console dan buka konsol Amazon Lex V2 di https://console.aws.amazon.com/lexv2/rumah.
2. Di halaman Bots, pilih bot Anda.
3. Pada halaman Bot Locale, klik Configure di bawah bagian Assisted NLU.
4. Di bawah bagian fitur AI generatif Runtime, matikan fitur NLU Bantuan, dan klik Simpan.
5. Bangun bot untuk menerapkan perubahan.

Selesaikan masukan pengguna yang ambigu dengan Disambiguasi Intent

Disambiguasi Intent adalah peningkatan pada NLU Berbantuan yang membantu menyelesaikan masukan pengguna yang ambigu ketika beberapa intent dapat cocok. Saat diaktifkan, sistem menyajikan pertanyaan klarifikasi kepada pengguna, membantu mereka menentukan maksud sebenarnya untuk meningkatkan akurasi percakapan. Sistem menggunakan model bahasa besar (LLM) yang menganalisis nama maksud dan deskripsi sebagai konteks, dan berdasarkan ambiguitas ucapan pengguna, mengembalikan maksud yang paling mungkin cocok. LLM mengevaluasi apakah input pengguna jelas cocok dengan satu maksud atau beberapa maksud dan cukup ambigu untuk memerlukan disambiguasi, kemudian memberikan maksud kandidat.

Fitur Intent Disambiguation tersedia dalam bahasa Inggris, Spanyol, Portugis, Catalan, Prancis, Italia, Jerman, China, Jepang, dan Korea. Secara khusus, ini mendukung lokal yang dimulai denganen_,, pt_ (PT_BRes_, PT_pt), (Ca_ES), (fr_CA, fr_FR), ca_ (IT_IT), fr_ (de_AT, de_DE), it_ (Zh_CN, Zh_hk),, de_ dan. zh_ ja_JP ko_KR Untuk daftar lengkap lokal yang didukung, lihat tabel diBahasa dan lokal yang didukung oleh Amazon Lex V2.

Anda dapat mengonfigurasi opsi berikut untuk Disambiguasi Intent:

Jumlah Opsi Intent

Konfigurasikan jumlah maksimum intent (2-5) untuk disajikan kepada pengguna saat disambiguasi diperlukan. Pengaturan ini menentukan berapa banyak opsi intent yang akan ditampilkan kepada pengguna saat sistem mendeteksi input yang ambigu. Nilai defaultnya adalah 3, yang memberikan

keseimbangan yang baik antara memberi pengguna opsi yang cukup sambil menjaga pemilihan tetap dapat dikelola.

Pesan Disambiguasi

Berikan pesan khusus yang akan ditampilkan sebelum menyajikan opsi disambiguasi kepada pengguna. Pesan ini membantu mengatur konteks bagi pengguna dan dapat disesuaikan agar sesuai dengan nada dan merek bot Anda. Jika tidak ditentukan, pesan default akan digunakan.

Nama Tampilan Maksud

Konfigurasikan nama tampilan yang mudah digunakan untuk maksud Anda guna meningkatkan pengalaman disambiguasi. Ini disarankan jika nama maksud Anda bersifat teknis atau tidak cocok untuk ditampilkan kepada pengguna akhir. Nama tampilan akan ditampilkan kepada pengguna selama disambiguasi, bukan nama maksud teknis.

Console

Menggunakan Intent Disambiguation dengan bot Amazon Lex V2

1. Masuk ke AWS Management Console dan buka konsol Amazon Lex V2 di https://console.aws.amazon.com/lexv2/rumah.
2. Di halaman Bots, pilih bot yang ingin Anda gunakan dengan Intent Disambiguation.
3. Pada halaman Bot Locale, klik Configure di bawah bagian Assisted NLU.
4. Aktifkan NLU Berbantuan dan pilih mode Primer atau Fallback (Disambiguasi Intent berfungsi dengan kedua mode).
5. Di bagian Intent Disambiguation dalam konfigurasi Assisted NLU, gunakan tombol toggle untuk mengaktifkan fitur Intent Disambiguation.
6. Konfigurasikan pengaturan opsional berikut:
   - Jumlah opsi Intent: Pilih jumlah maksimum intent (2-5) untuk ditampilkan kepada pengguna selama disambiguasi. Defaultnya adalah 3.
   - Pesan Disambiguasi: Berikan pesan khusus yang akan ditampilkan saat menyajikan opsi maksud. Jika tidak ditentukan, pesan default akan digunakan.
7. Klik Simpan untuk menerapkan konfigurasi.
8. Konfigurasikan Nama Tampilan Intent secara opsional untuk pengalaman pengguna yang lebih baik:

a. Arahkan ke setiap intent di bot yang ingin Anda konfigurasikan.
b. Di Halaman Editor Maksud, cari bidang Nama tampilan.
c. Masukkan nama yang mudah digunakan yang akan ditampilkan kepada pengguna selama disambiguasi, bukan nama Intent.

9. Bangun bot untuk melihat perubahan yang tercermin dalam bot Anda saat runtime.

Panduan untuk meningkatkan efektivitas bot Anda saat menggunakan Fitur Disambiguasi Intent

Praktik terbaik berikut dapat membantu Anda memaksimalkan efektivitas fitur Disambiguasi Intent:

1. Hapus Nama dan Deskripsi Maksud: Pastikan nama maksud dan deskripsi bersih, jelas, dan tidak tumpang tindih dengan maksud lain, karena ini adalah masukan utama yang diberikan kepada LLM untuk disambiguasi.
2. Nama Tampilan Deskriptif: Jika nama Intent saat ini bersifat teknis, gunakan nama tampilan deskriptif yang mengkomunikasikan tujuan maksud dengan jelas. Nama tampilan harus menyerupai atau cocok dengan nama maksud.
3. Maksud Maksimum yang Sesuai: Tetapkan jumlah maksimum opsi maksud berdasarkan preferensi dan pengujian Anda.
4. Pesan Kustom: Buat pesan disambiguasi ringkas yang mengakui masukan pengguna dan mengarah ke opsi maksud.
5. Skenario Pengujian: Uji dengan ucapan ambigu untuk memastikan pengalaman disambiguasi terasa alami dengan nama maksud atau nama tampilan maksud dan pesan khusus, dan verifikasi bahwa opsi maksud yang benar disajikan selama prompt disambiguasi.

> **Important**
>
> Aktifkan fitur ini dalam versi draf bot. Uji sebelum menggunakannya dalam alias produksi.

Menonaktifkan Disambiguasi Intent

Untuk menonaktifkan fitur Intent Disambiguation, ikuti langkah-langkah berikut:

1. Masuk ke AWS Management Console dan buka konsol Amazon Lex V2 di https://console.aws.amazon.com/lexv2/rumah.

2. Di halaman Bots, pilih bot Anda.
3. Pada halaman Bot Locale, klik Configure di bawah bagian Assisted NLU.
4. Di bagian Intent Disambiguation dalam konfigurasi Assisted NLU, matikan fitur Intent Disambiguation, lalu klik Simpan.
5. Bangun bot untuk menerapkan perubahan.

Optimalkan Bot menggunakan Bot Analyzer yang didukung AI

Analisis konfigurasi bot Amazon Lex V2 Anda terhadap praktik terbaik AWS menggunakan rekomendasi yang didukung AI. Bot Analyzer menggunakan kemampuan AI generatif Amazon Bedrock untuk mengidentifikasi masalah konfigurasi dan memberikan panduan yang dapat ditindaklanjuti untuk meningkatkan klasifikasi maksud dan kinerja resolusi slot.

Bot Analyzer secara otomatis mengevaluasi konfigurasi intent bot Anda dan memberikan rekomendasi untuk:

- Tingkatkan pemisahan maksud - Identifikasi dan selesaikan maksud umum yang mengelompokkan beberapa konsep
- Hilangkan tumpang tindih maksud - Mendeteksi makna dan frasa serupa di antara intent yang menyebabkan kesalahan perutean
- Optimalkan penggunaan slot - Merekomendasikan penggunaan slot yang tepat untuk menggabungkan maksud serupa dan meningkatkan ekstraksi entitas
- Tingkatkan kualitas ucapan - Analisis cakupan ucapan sampel dan keragaman untuk klasifikasi niat yang lebih baik

Prasyarat

Sebelum menggunakan Bot Analyzer, pastikan:

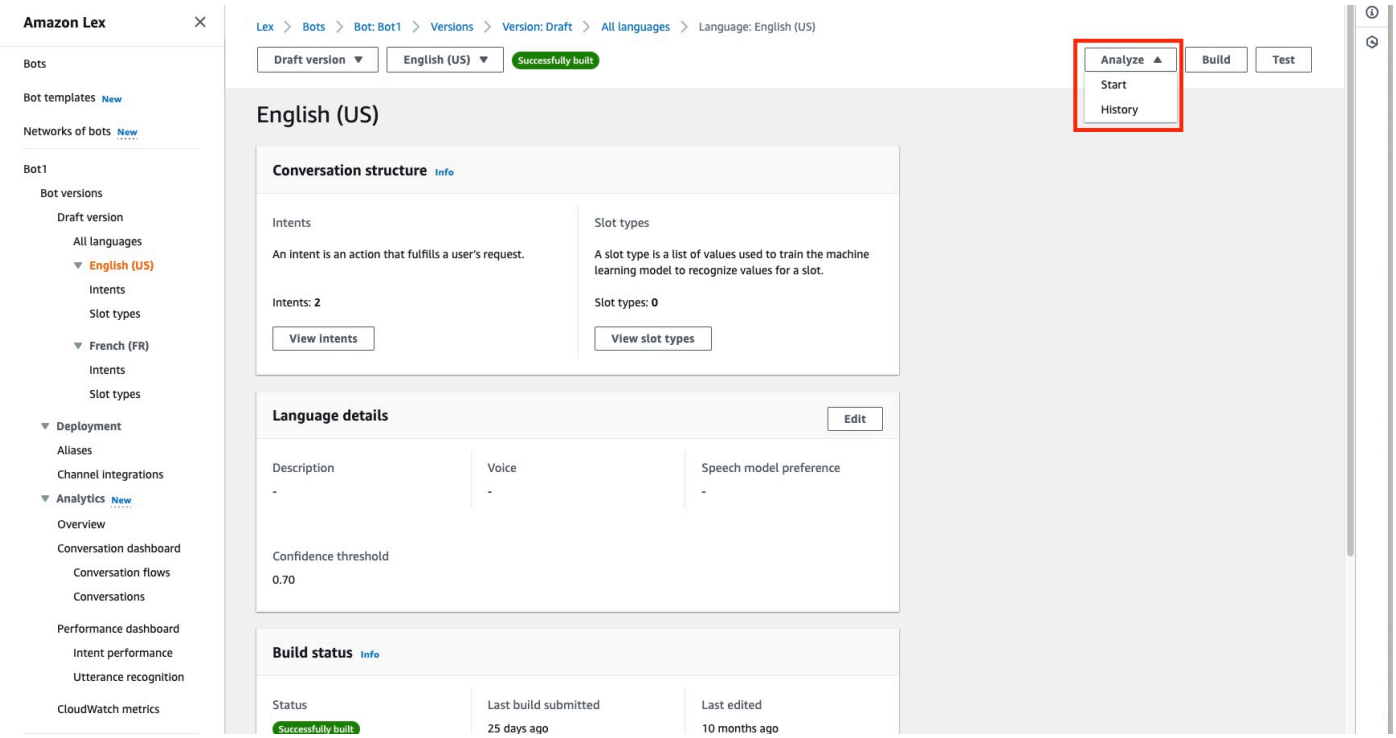
- Lokal bot Anda berhasil dibangun
- Versi bot Anda untuk analisis adalah `DRAFT`
- Lokal bot Anda adalah salah satu lokal bahasa Inggris yang didukung:`en_AU`,,,`en_GB`, `en_IN` `en_US` `en_ZA`

Anda dapat menggunakan Bot Analyzer dengan konsol atau API.

Console

Mulai analisis

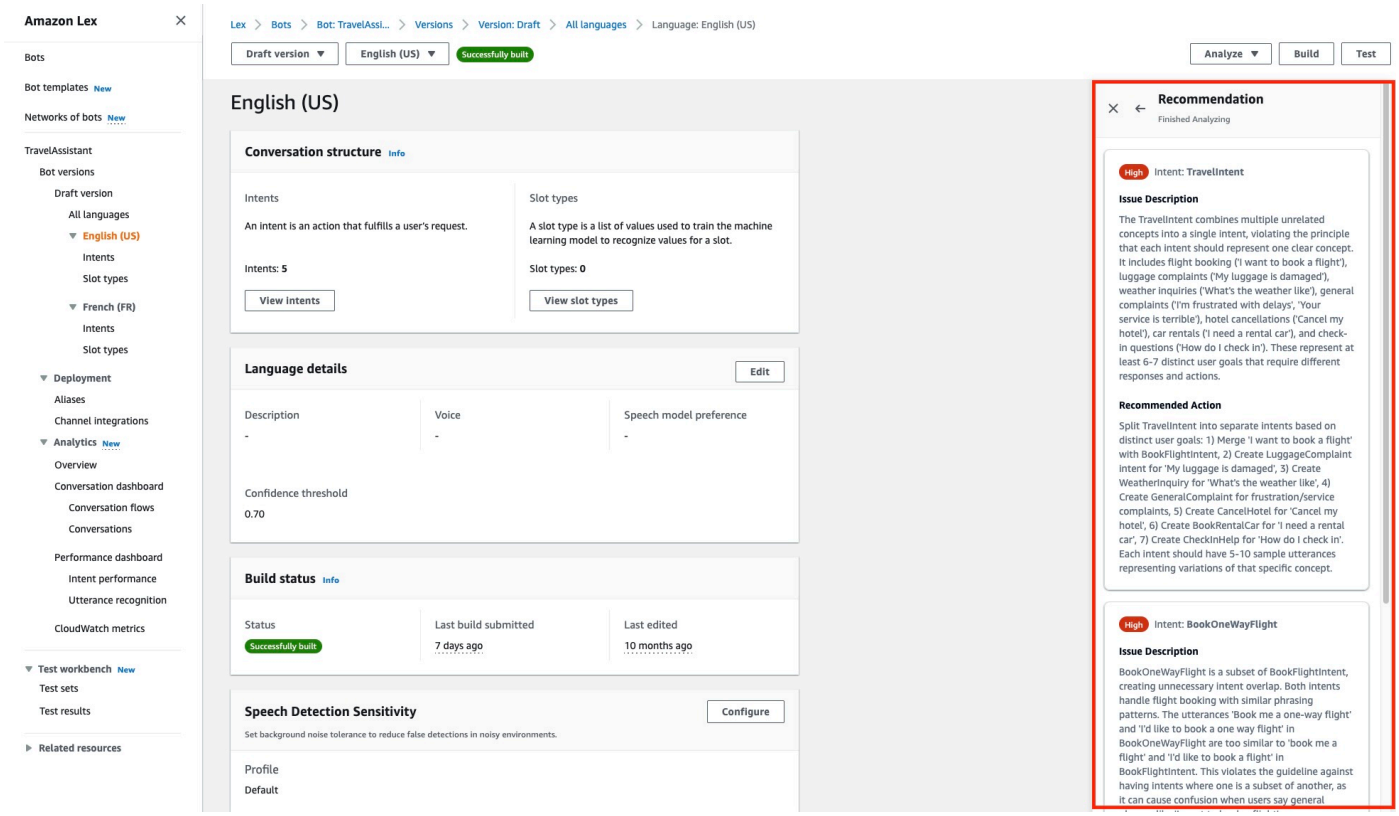
1. Masuk ke AWS Management Console dan buka konsol Amazon Lex V2 di https://console.aws.amazon.com/lexv2/rumah.
2. Pilih bot Anda dan arahkan ke lokal bot yang ingin Anda analisis.
3. Di editor lokal bot, klik menu tarik-turun Analisis.
4. Pilih Mulai untuk memulai analisis.



Analisis biasanya selesai dalam beberapa menit. Selama analisis, tombol Mulai berubah menjadi Berhenti Menganalisis jika Anda perlu membatalkan.

Lihat Rekomendasi

Setelah analisis selesai, rekomendasi muncul di panel Rekomendasi di sisi kanan.



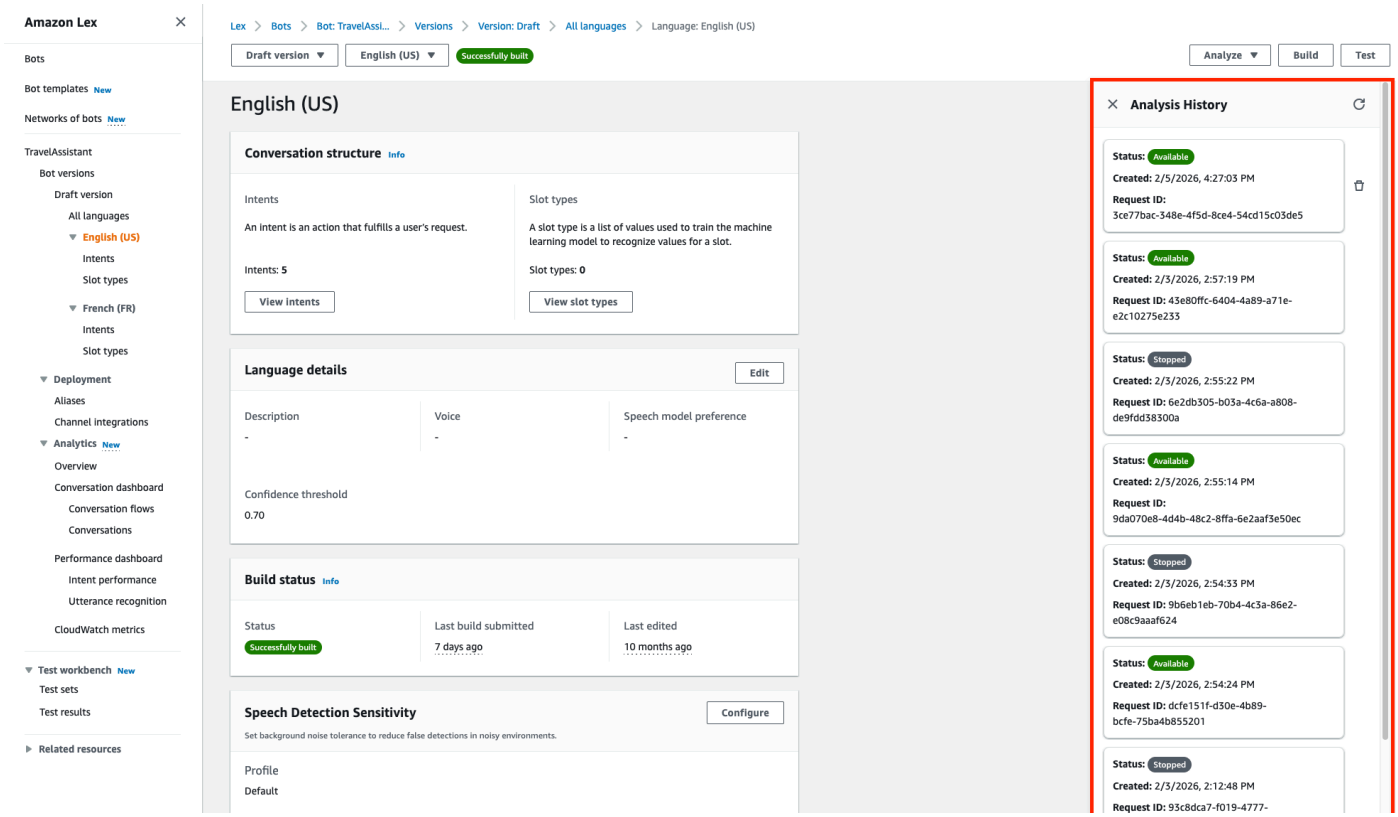
Setiap rekomendasi meliputi:

- Prioritas - Tingkat keparahan Tinggi, Sedang, atau Rendah
- Lokasi Masalah - Maksud spesifik yang terpengaruh
- Deskripsi Masalah - Masalah konfigurasi apa yang terdeteksi
- Perbaikan yang Diusulkan - Langkah-langkah yang dapat ditindaklanjuti untuk menyelesaikan masalah

Lihat riwayat analisis

Untuk melihat analisis sebelumnya:

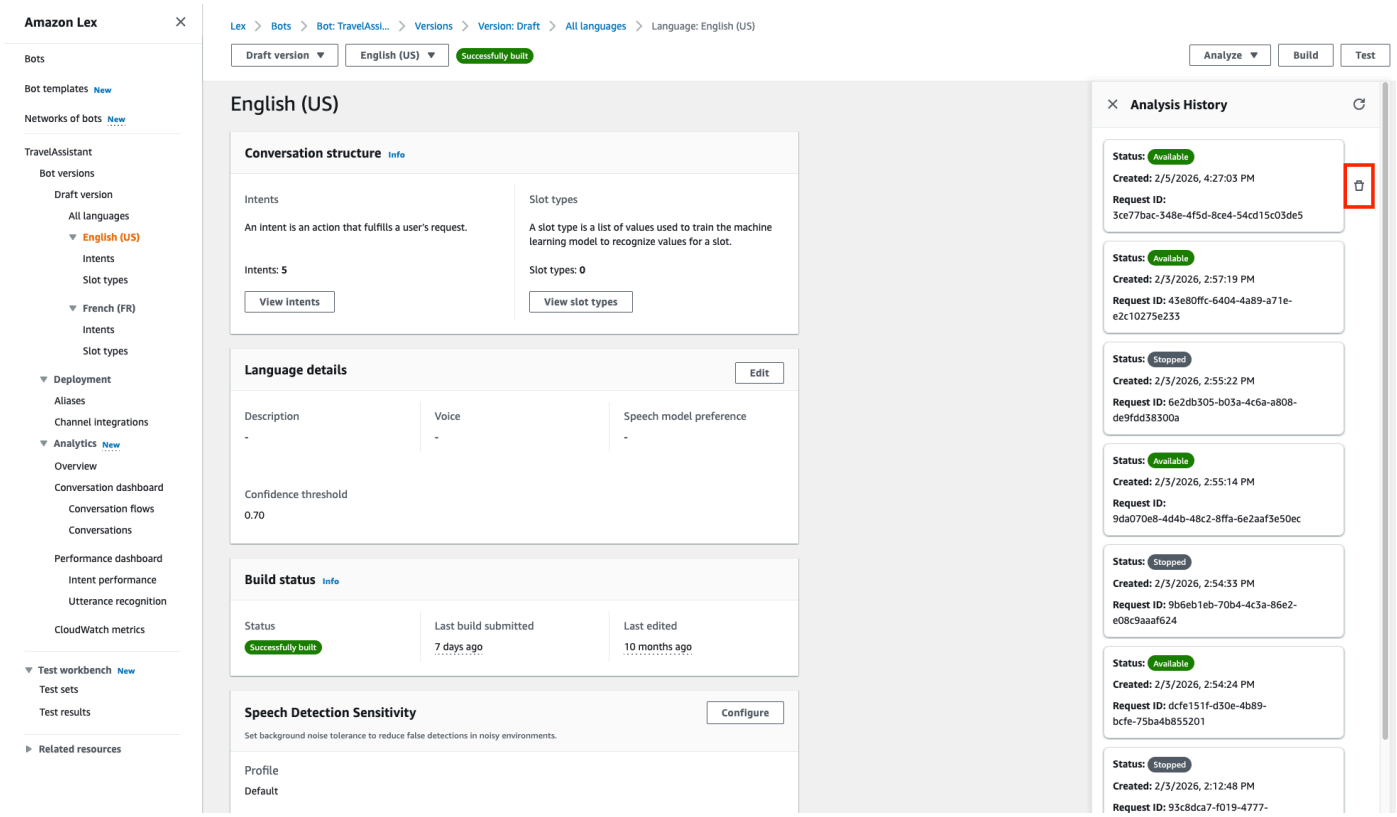
1. Klik menu dropdown Analisis.
2. Pilih Riwayat.
3. Panel Analysis History menampilkan permintaan analisis sebelumnya dengan status dan stempel waktunya.



Hapus Rekomendasi

Untuk menghapus hasil analisis:

1. Klik menu dropdown Analisis.
2. Pilih Hapus.
3. Konfirmasikan penghapusan rekomendasi saat ini.



API

Mulai analisis

Kirim `StartBotAnalyzer` permintaan untuk memulai analisis untuk lokal bot Anda. Respons mengembalikan status HTTP 202 dengan file. `botAnalyzerRequestId` Catat ID ini dan Anda akan memerlukannya untuk memeriksa status analisis dan mengambil rekomendasi.

Periksa status analisis dan ambil rekomendasi

Kirim `DescribeBotAnalyzerRecommendation` permintaan menggunakan `botAnalyzerRequestId` dari langkah sebelumnya. Sertakan `botId` di jalur permintaan.

Jika `botAnalyzerStatus` dalam responsnya`Available`, analisis selesai dan `botAnalyzerRecommendationList` bidang akan diisi dengan rekomendasi. Setiap rekomendasi meliputi:

- `issueLocation`- Lokasi di mana masalah terdeteksi
- `priority`- Tingkat keparahan Tinggi, Sedang, atau Rendah
- `issueDescription`- Detail tentang masalah konfigurasi

- `proposedFix`- Panduan yang dapat ditindaklanjuti untuk menyelesaikan masalah

Hentikan analisis yang sedang berlangsung

Jika Anda perlu membatalkan analisis yang sedang berlangsung, kirim `StopBotAnalyzer` permintaan dengan `botId` dan`botAnalyzerRequestId`.

Lihat riwayat analisis

Untuk mengambil daftar analisis sebelumnya untuk lokal bot, kirim permintaan. `ListBotAnalyzerHistory` Tentukan `botId` dan `localeId` untuk melihat semua permintaan analisis sebelumnya dengan status dan stempel waktu mereka.

Hapus rekomendasi

Untuk menghapus hasil analisis, kirim `DeleteBotAnalyzerRecommendation` permintaan dengan `botId` dan`botAnalyzerRequestId`. Ini secara permanen menghapus rekomendasi yang terkait dengan analisis itu.

> **Note**
>
> Rekomendasi dihapus secara otomatis setelah 15 hari.

Topik Terkait

- Praktik terbaik untuk membuat model interaksi Amazon Lex
- Tingkatkan klasifikasi maksud dan resolusi slot di Lex V2 dengan bantuan NLU
- Menggunakan resolusi slot berbantuan untuk memperjelas nilai slot di Amazon Lex V2

Amazon.qn AIntent

> **Note**
>
> Sebelum Anda dapat memanfaatkan fitur AI generatif, Anda harus memenuhi prasyarat berikut
>
> 1. Untuk informasi tentang harga untuk menggunakan Amazon Bedrock, lihat harga Amazon Bedrock.

2. Aktifkan kemampuan AI generatif untuk lokal bot Anda. Untuk melakukannya, ikuti langkah-langkah diOptimalkan pembuatan dan kinerja bot Lex V2 dengan menggunakan AI generatif.

Anda dapat memanfaatkan Amazon Bedrock FMs untuk membantu menjawab pertanyaan pelanggan dalam percakapan bot. Amazon Lex V2 menawarkan built-in `AMAZON.QnAIntent` yang dapat Anda tambahkan ke bot Anda. Maksud ini memanfaatkan kemampuan AI generatif dari Amazon Bedrock dengan mengenali pertanyaan pelanggan dan mencari jawaban dari toko pengetahuan berikut (misalnya,). **`Can you provide me details on the baggage limits for my international flight?`** Fitur ini mengurangi kebutuhan untuk mengonfigurasi pertanyaan dan jawaban menggunakan dialog berorientasi tugas dalam maksud Amazon Lex V2. Maksud ini juga mengenali pertanyaan tindak lanjut (misalnya,**`What about domestic flight?`**) berdasarkan riwayat percakapan dan memberikan jawaban yang sesuai.

Pastikan bahwa peran IAM Anda memiliki izin yang tepat untuk mengakses `AMAZON.QnAIntent` dengan mengikuti langkah-langkah di. Izin untuk AMAZON.QnAIntent

Untuk mengambil keuntungan dari `AMAZON.QnAIntent` Anda harus telah mendirikan salah satu toko pengetahuan berikut.

- Basis data OpenSearch Layanan Amazon — Untuk informasi selengkapnya, lihat Membuat dan mengelola domain OpenSearch Layanan Amazon.
- Indeks Amazon Kendra — Untuk informasi selengkapnya, lihat Membuat indeks.
- Basis pengetahuan Amazon Bedrock — Untuk informasi selengkapnya, lihat Membangun basis pengetahuan.

Anda dapat mengatur `AMAZON.QnAIntent` dalam salah satu dari dua cara:

Untuk mengatur menggunakan konfigurasi Generative AI

1. Di konsol Amazon Lex V2, pilih Bots dari panel navigasi kiri dan pilih bot yang ingin Anda tambahkan intent dari bagian Bots.
2. Dari panel navigasi kiri, pilih bahasa yang ingin Anda tambahkan intent.
3. Di bagian Generative AI configurations, pilih Configure.
4. Di bagian konfigurasi QnA, pilih Buat maksud QnA.

Untuk mengatur dengan menambahkan intent bawaan ke bot Anda

1. Di konsol Amazon Lex V2, pilih Bots dari panel navigasi kiri dan pilih bot yang ingin Anda tambahkan intent dari bagian Bots.
2. Dari panel navigasi kiri, pilih Maksud di bawah bahasa yang ingin Anda tambahkan intent.
3. Pilih Tambah maksud dan pilih Gunakan maksud bawaan dari menu tarik-turun.
4. Untuk detail selengkapnya tentang konfigurasi`AMAZON.QnAIntent`, lihatAMAZON.QnAIntent.

> **Note**
>
> `AMAZON.QnAIntent`Ini diaktifkan ketika ucapan tidak diklasifikasikan ke dalam maksud lain yang ada di bot. Maksud ini diaktifkan ketika ucapan tidak diklasifikasikan ke dalam maksud lain yang ada di bot. Perhatikan bahwa maksud ini tidak akan diaktifkan untuk ucapan yang tidak terjawab saat memunculkan nilai slot. Setelah dikenali, `AMAZON.QnAIntent` menggunakan model Amazon Bedrock yang ditentukan untuk mencari basis pengetahuan yang dikonfigurasi dan menanggapi pertanyaan pelanggan.

Topik

- Izin untuk AMAZON.QnAIntent

Izin untuk AMAZON.QnAIntent

Untuk mengakses fitur ini di konsol Amazon Lex V2, pastikan peran konsol Anda memiliki `bedrock:ListFoundationModels` dan `bedrock:ListInferenceProfiles` izin.

Peran IAM yang terkait dengan bot harus memiliki izin berikut yang diperlukan untuk. `AMAZON.QnAIntent` Peran bot harus memiliki izin untuk meneleponbedrock:InvokeModel. Anda juga harus melampirkan pernyataan untuk setiap penyimpanan data yang Anda tentukan di bot Anda `AMAZON.QnAIntent` (lihat`Permissions to access Amazon Kendra index`,`Permissions to access OpenSearch Service index`, dan `Permissions to access knowledge base in Amazon Bedrock` pernyataan dalam kebijakan di bawah). Saat Anda mengaktifkan fitur dengan konsol Amazon Lex, kebijakan akan secara otomatis ditambahkan ke peran bot asalkan bot Anda menggunakan peran terkait layanan yang dihasilkan oleh Amazon Lex.

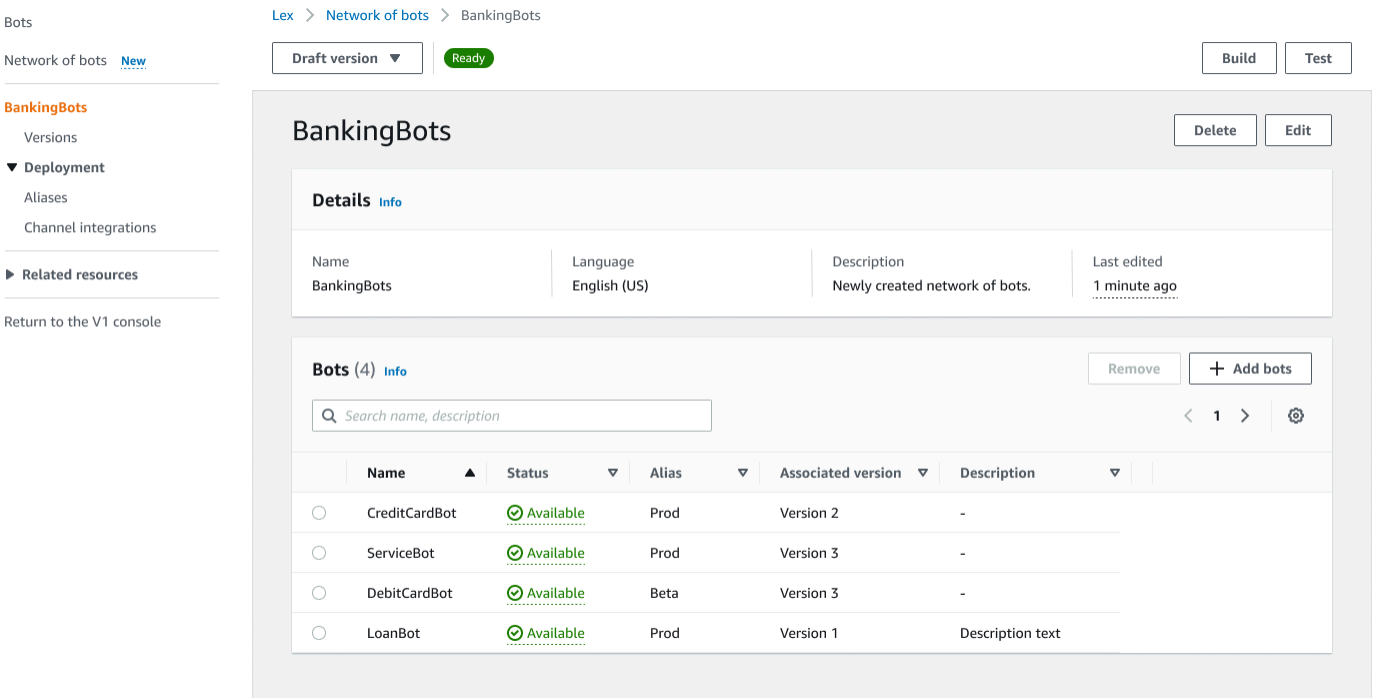
Membuat jaringan bot untuk bot Lex V2 Anda

Network of Bots memungkinkan perusahaan untuk memberikan pengalaman pengguna terpadu di beberapa bot. Dengan Network of Bots, perusahaan dapat menambahkan beberapa bot ke satu jaringan untuk memungkinkan manajemen siklus hidup bot yang fleksibel dan independen. Jaringan mengekspos satu antarmuka terpadu ke pengguna akhir dan merutekan permintaan ke bot yang sesuai berdasarkan input pengguna.

Tim dapat berkolaborasi untuk membuat jaringan bot untuk memenuhi berbagai kebutuhan bisnis dengan memelihara dan menambahkan bot ke jaringan saat bot yang ditingkatkan digunakan untuk produksi. Pengembang dapat menyederhanakan dan mempercepat penyebaran dan peningkatan dengan mengintegrasikan beberapa bot ke dalam satu jaringan.

Jaringan bot saat ini hanya tersedia dalam bahasa en-US.

> **Note**
>
> Saat ini, jaringan bot terbatas pada satu akun. Anda tidak dapat menambahkan bot dari akun lain.



Buat jaringan bot untuk bot Lex V2 Anda

Masuk ke AWS Management Console dan buka konsol Amazon Lex V2 di https://console.aws.amazon.com/lexv2/home. Pilih Jaringan bot dari menu samping. Anda harus membangun setidaknya satu bot untuk membuat jaringan bot.

Langkah 1: Konfigurasikan jaringan pengaturan bot

1. Di bagian Detail, masukkan nama jaringan Anda dan berikan deskripsi opsional.
2. Di bagian izin IAM, pilih peran AWS Identity and Access Management (IAM) yang memberikan izin Amazon Lex V2 untuk mengakses layanan AWS lainnya, seperti Amazon. CloudWatch Anda dapat meminta Amazon Lex V2 membuat peran, atau Anda dapat memilih peran yang ada dengan CloudWatch izin. Lihat Identitas dan manajemen akses untuk Amazon Lex V2 untuk informasi selengkapnya.
3. Di bagian Children's Online Privacy Protection Act (COPPA), pilih respons yang sesuai. Untuk informasi selengkapnya, lihat DataPrivacy.
4. Di bagian batas waktu sesi Idle, pilih durasi yang Amazon Lex V2 menyimpan sesi dengan pengguna terbuka. Amazon Lex V2 mempertahankan variabel sesi selama sesi sehingga bot Anda dapat melanjutkan percakapan dengan variabel yang sama. Lihat Mengatur batas waktu sesi untuk informasi selengkapnya.
5. Di bagian Tambahkan pengaturan bahasa, pilih suara untuk bot Anda untuk berinteraksi dengan pengguna. Anda dapat mengetik frasa dalam sampel Suara dan memilih Putar untuk mendengarkan suara.
6. Di bagian Pengaturan lanjutan, tambahkan tag yang membantu mengidentifikasi bot secara opsional. Tag dapat digunakan untuk mengontrol akses dan memantau sumber daya. Lihat Menandai sumber daya untuk informasi selengkapnya.
7. Pilih Berikutnya untuk membuat jaringan bot dan pindah ke menambahkan bot.

Langkah 2: Tambahkan bot

1. Di bagian Bots, pilih + Tambahkan bot.
2. Modal Add bots akan muncul. Pilih bot untuk ditambahkan dari menu dropdown Bot dan alias bot yang ingin Anda gunakan dari menu dropdown Alias.

   Alias harus menunjuk ke versi bot bernomor dan bukan ke versi draf. Anda dapat menambahkan hingga 5 bot. Bot dapat ditambahkan hingga 25 jaringan yang berbeda.

3. Pilih + Tambahkan bot untuk menambahkan lebih banyak bot ke jaringan Anda. Untuk menghapus bot, pilih Hapus di sebelah bot yang ingin Anda hapus. Setelah selesai menambahkan bot, pilih Simpan untuk menutup modal.
4. Pilih Simpan untuk menyelesaikan pembuatan jaringan Anda.

Kelola jaringan bot Anda untuk bot Lex V2 Anda

Setelah membuat jaringan bot Anda, Anda akan dibawa ke halaman di mana Anda dapat mengelola dan membangun jaringan Anda. Atau Anda dapat mencapai halaman ini dengan memilih Jaringan bot di menu samping, dan memilih nama jaringan yang akan dikelola.

1. Untuk mengedit informasi untuk jaringan Anda, pilih Edit di atas bagian Detail. Untuk menghapus jaringan, pilih Hapus di atas bagian Detail.
2. Di bagian Bots, Anda dapat menambahkan lebih banyak bot dengan memilih + Tambahkan bot. Anda juga dapat menambahkan bot jika Anda menavigasi ke halaman Bots di menu samping di konsol Amazon Lex V2. Alihkan tombol radio di sebelah bot yang ingin Anda tambahkan, dan pilih Tambahkan ke jaringan bot dari menu tarik-turun Tindakan.

   Dari menu dropdown Network of bots di modal yang muncul, pilih jaringan tempat Anda ingin menambahkan bot. Kemudian pilih alias bot yang ingin Anda gunakan dari menu dropdown alias Bot. Pilih Tambahkan untuk menambahkan bot ke jaringan yang Anda pilih.
3. Anda dapat menghapus bot dari jaringan Anda dengan mengaktifkan tombol radio di sebelah bot dan memilih Hapus.
4. Setelah selesai mengonfigurasi jaringan, pilih Build di kanan atas untuk membangun jaringan Anda. Mungkin perlu beberapa menit untuk membangunnya. Jika build berhasil, spanduk sukses hijau muncul di bagian atas halaman.
5. Setelah jaringan dibangun, Anda dapat memilih Uji di kanan atas agar jendela obrolan muncul di sudut kanan bawah. Anda dapat menggunakan jendela obrolan ini untuk berkomunikasi dengan bot jaringan Anda dan memastikan alur percakapan dan transisi dikonfigurasi dengan benar.

> **Note**
>
> Jika Anda menambahkan, menghapus, atau memperbarui bot dalam jaringan Anda, Anda harus membangun kembali jaringan.

Versi jaringan bot Anda untuk Lex V2

Anda dapat membuat versi yang berbeda dari jaringan bot Anda. Untuk mengelola versi, pilih jaringan Anda dari menu samping di konsol Amazon Lex V2 dan pilih Versi.

1. Pilih Buat Versi untuk membuat versi baru dari jaringan bot Anda. Anda dapat menambahkan deskripsi opsional. Pilih Buat untuk membuat versi.
2. Saat Anda mengaktifkan tombol radio di sebelah versi jaringan bot Anda, Anda dapat memilih Alias asosiasi dengan versi untuk mengarahkan alias ke versi ini.
3. Untuk mengelola versi jaringan Anda, pilih nama versi di bagian Versi. Di halaman berikut, Anda dapat mengedit detail versi dan mengelola bot dalam versi dan alias terkait.

Alias untuk jaringan bot Anda untuk Lex V2

Anda dapat menggunakan alias untuk menyebarkan jaringan Anda. Untuk mengelola alias, pilih jaringan Anda dari menu samping di konsol Amazon Lex V2 dan pilih Alias.

1. Pilih Buat alias untuk membuat alias baru.
2. Berikan nama alias dan Deskripsi opsional di bagian detail Alias. Anda dapat memilih versi untuk mengaitkan alias dengan bagian Associate with a version dan menambahkan tag di bagian Tag. Pilih Buat untuk membuat alias.
3. Untuk mengelola alias untuk jaringan Anda, pilih nama alias di bagian Alias. Di halaman berikut, Anda dapat mengedit detail alias dan mengelola tag, integrasi saluran, dan kebijakan berbasis sumber daya. Anda juga dapat melihat riwayat hubungannya dengan versi jaringan.

Integrasi saluran untuk jaringan bot Lex V2 Anda

Untuk mengintegrasikan jaringan bot Anda dengan platform perpesanan, pilih jaringan bot Anda dari menu samping di konsol Amazon Lex V2. Kemudian pilih Integrasi saluran.

1. Pilih Tambahkan saluran untuk mengintegrasikan jaringan Anda dengan saluran baru.
2. Di bagian Platform, pilih platform yang ingin Anda gunakan bot Anda di Select platform. Peran IAM akan dibuat. Pilih kunci dari menu tarik-turun di bawah tombol KMS untuk melindungi informasi Anda.
3. Di saluran konfigurasi Integrasi, masukkan Nama dan Deskripsi opsional. Pilih Alias dari menu dropdown.

4. Dapatkan SID akun Anda dan token otentikasi dari platform dan isi bidang Token SID dan Otentikasi Akun. Lihat Mengintegrasikan bot Anda untuk informasi selengkapnya.
5. Pilih Buat untuk menyelesaikan integrasi saluran.

> **Note**
>
> Jaringan bot saat ini tidak tersedia dalam suara atau obrolan Connect Customer.

Menyebarkan bot dari Lex V2 untuk lingkungan produksi Anda

Setelah membuat dan menguji bot Anda, bot siap digunakan untuk berinteraksi dengan pelanggan Anda. Di bagian ini, pelajari cara membuat versi bot Anda saat Anda telah melakukan pembaruan. Gunakan alias untuk menunjuk ke berbagai versi bot Anda saat mereka siap untuk diterapkan. Pelajari cara mengintegrasikan bot Anda dengan platform perpesanan, aplikasi seluler, dan situs web.

Topik

- Pembuatan versi dan alias dengan bot Lex V2 Anda
- Menggunakan aplikasi Java untuk berinteraksi dengan bot Amazon Lex V2
- Gunakan Ketahanan Global untuk menyebarkan bot ke Wilayah lain
- Mengintegrasikan bot Amazon Lex V2 dengan platform perpesanan
- Mengintegrasikan bot Amazon Lex V2 dengan pusat kontak

Pembuatan versi dan alias dengan bot Lex V2 Anda

Amazon Lex V2 mendukung pembuatan versi dan alias bot dan jaringan bot sehingga Anda dapat mengontrol implementasi yang digunakan aplikasi klien Anda. Versi bertindak sebagai snapshot bernomor dari pekerjaan Anda. Anda dapat mengarahkan alias ke versi bot Anda yang ingin tersedia untuk pelanggan Anda. Di sela-sela pembuatan versi, Anda dapat terus memperbarui `Draft` versi bot Anda tanpa memengaruhi pengalaman pengguna.

Versi

Amazon Lex V2 mendukung pembuatan versi bot sehingga Anda dapat mengontrol implementasi yang digunakan aplikasi klien Anda. Versi adalah snapshot bernomor dari pekerjaan Anda yang dapat Anda buat untuk digunakan di berbagai bagian alur kerja Anda, seperti pengembangan, penerapan beta, dan produksi.

Versi Draf bot Lex V2 Anda

Saat Anda membuat bot Amazon Lex V2 hanya ada satu versi, `Draft` versinya.

`Draft`adalah salinan bot Anda yang berfungsi. Anda hanya dapat memperbarui `Draft` versi dan sampai Anda membuat versi pertama Anda, `Draft` adalah satu-satunya versi bot yang Anda miliki.

`Draft`Versi bot Anda dikaitkan dengan`TestBotAlias`. `TestBotAlias`Seharusnya hanya digunakan untuk pengujian manual. Amazon Lex V2 membatasi jumlah permintaan runtime yang dapat Anda buat untuk `TestBotAlias` alias bot.

Membuat versi untuk bot Lex V2 Anda

Saat Anda membuat versi bot Amazon Lex V2, Anda membuat snapshot bot bernomor sehingga Anda dapat menggunakan bot seperti yang ada saat versi dibuat. Setelah Anda membuat versi numerik, itu akan tetap sama saat Anda terus mengerjakan versi draf aplikasi Anda.

Saat membuat versi, Anda dapat memilih lokal yang akan disertakan dalam versi. Anda tidak perlu memilih semua lokal di bot. Selain itu, saat Anda membuat versi, Anda dapat memilih lokal dari versi sebelumnya. Misalnya, jika Anda memiliki tiga versi bot, Anda dapat memilih satu lokal dari `Draft` versi dan satu dari versi dua saat Anda membuat versi empat.

Jika Anda menghapus lokal dari `Draft` versi, itu tidak dihapus dari versi bernomor.

Jika versi bot tidak digunakan selama enam bulan, Amazon Lex V2 akan menandai versi tidak aktif. Saat versi tidak aktif, Anda tidak dapat menggunakan operasi runtime dengan bot. Untuk membuat bot aktif, bangun kembali semua bahasa yang terkait dengan versi.

Memperbarui bot Amazon Lex V2

Anda hanya dapat memperbarui `Draft` versi bot Amazon Lex V2. Versi tidak dapat diubah. Anda dapat membuat versi baru kapan saja setelah memperbarui sumber daya di konsol atau dengan CreateBotVersionoperasi.

Menghapus bot atau versi Amazon Lex V2

Amazon Lex V2 mendukung penghapusan bot atau versi menggunakan konsol atau salah satu operasi API:

- DeleteBot
- DeleteBotVersion

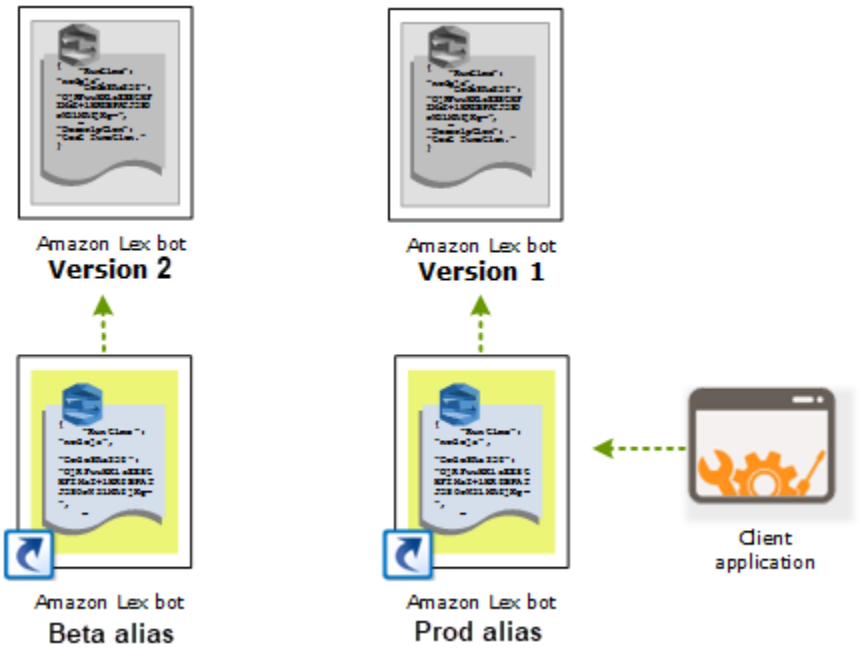
Alias untuk bot Lex V2 Anda

Bot Amazon Lex V2 mendukung alias. Alias adalah pointer ke versi bot tertentu. Dengan alias, Anda dapat dengan mudah memperbarui versi yang digunakan aplikasi klien Anda. Misalnya, Anda dapat mengarahkan alias ke versi 1 bot Anda. Ketika Anda siap untuk memperbarui bot, Anda membuat versi 2 dan mengubah alias untuk menunjuk ke versi baru. Karena aplikasi Anda menggunakan alias alih-alih versi tertentu, semua klien Anda mendapatkan fungsionalitas baru tanpa perlu diperbarui.

Alias adalah penunjuk ke versi tertentu dari bot Amazon Lex V2. Gunakan alias untuk memungkinkan aplikasi klien menggunakan versi bot tertentu tanpa memerlukan aplikasi untuk melacak versi mana yang ada.

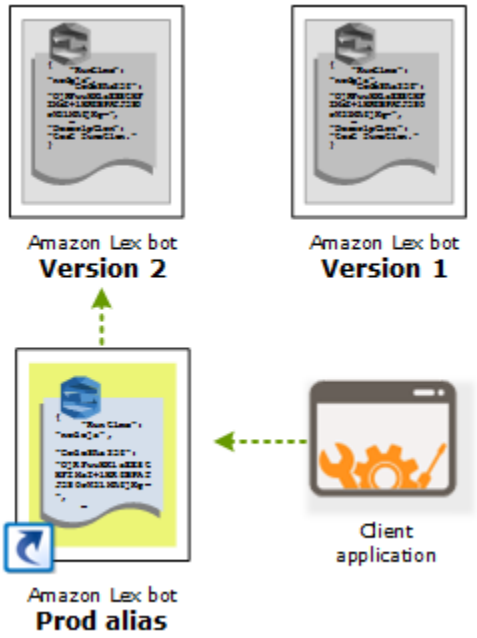
Saat Anda membuat bot, Amazon Lex V2 membuat alias `TestBotAlias` yang disebut yang dapat Anda gunakan untuk menguji bot Anda. `TestBotAlias`Alias selalu dikaitkan dengan `Draft` versi bot Anda. Anda hanya boleh menggunakan `TestBotAlias` alias untuk pengujian, Amazon Lex V2 membatasi jumlah permintaan runtime yang dapat Anda buat untuk alias.

Contoh berikut menunjukkan dua versi bot Amazon Lex V2, versi 1 dan versi 2. Masing-masing versi bot ini memiliki alias terkait, BETA dan PROD, masing-masing. Aplikasi klien menggunakan alias PROD untuk mengakses bot.



Saat Anda membuat versi bot kedua, Anda dapat memperbarui alias untuk menunjuk ke versi baru bot menggunakan konsol atau UpdateBotAliasoperasi. Ketika Anda mengubah alias, semua aplikasi

klien Anda menggunakan versi baru. Jika ada masalah dengan versi baru, Anda dapat memutar kembali ke versi sebelumnya hanya dengan mengubah alias untuk menunjuk ke versi itu.



Ketika Anda mengatur aplikasi klien Anda untuk memanggil Amazon Lex Runtime V2 APIs untuk memungkinkan pelanggan berinteraksi dengan bot Anda, Anda menggunakan alias yang menunjukkan versi yang Anda ingin pelanggan Anda gunakan.

> **Note**
>
> Meskipun Anda dapat menguji `Draft` versi bot di konsol, kami menyarankan bahwa ketika Anda mengintegrasikan bot dengan aplikasi klien Anda, Anda terlebih dahulu membuat versi dan membuat alias yang menunjuk ke versi itu. Gunakan alias dalam aplikasi klien Anda untuk alasan yang dijelaskan di bagian ini. Saat Anda memperbarui alias, Amazon Lex V2 akan menggunakan versi saat ini untuk semua sesi yang sedang berlangsung. Sesi baru menggunakan versi baru.

Menggunakan aplikasi Java untuk berinteraksi dengan bot Amazon Lex V2

AWS SDK untuk Java 2.0 menyediakan antarmuka yang dapat Anda gunakan dari aplikasi Java Anda untuk berinteraksi dengan bot Anda. Gunakan SDK for Java untuk membangun aplikasi klien bagi pengguna.

Aplikasi berikut berinteraksi dengan OrderFlowers bot yang Anda buat. Latihan 1: Buat chatbot dari template Ini menggunakan `LexRuntimeV2Client` dari SDK for Java SDK untuk memanggil RecognizeTextoperasi untuk melakukan percakapan dengan bot.

Output dari percakapan terlihat seperti ini:

```
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.
```

Untuk struktur JSON yang dikirim antara aplikasi klien dan bot Amazon Lex V2, lihatLatihan 2: Tinjau alur percakapan.

Untuk menjalankan aplikasi, Anda harus memberikan informasi berikut:

- BotID — Pengidentifikasi yang ditetapkan ke bot saat Anda membuatnya. Anda dapat melihat ID bot di konsol Amazon Lex V2 di halaman Pengaturan bot.
- botAliasId — Pengidentifikasi yang ditetapkan ke alias bot saat Anda membuatnya. Anda dapat melihat ID alias bot di konsol Amazon Lex V2 di halaman Alias. Jika Anda tidak dapat melihat ID alias dalam daftar, pilih ikon roda gigi di kanan atas dan nyalakan ID Alias.
- LocaleID — Pengidentifikasi lokal yang Anda gunakan untuk bot Anda. Untuk daftar lokal, lihatBahasa dan lokal yang didukung oleh Amazon Lex V2.
- AccessKey dan SecretKey — Kunci otentikasi untuk akun Anda. Jika Anda tidak memiliki satu set kunci, buat kunci menggunakan konsol AWS Identity and Access Management.
- SessionID — Pengenal untuk sesi dengan bot Amazon Lex V2. Dalam hal ini, kode menggunakan UUID acak.
- Jika bot Anda tidak berada di Wilayah AS Timur (Virginia N.), pastikan Anda mengubah Wilayah.

Aplikasi menggunakan fungsi yang dipanggil `getRecognizeTextRequest` untuk membuat permintaan individual ke bot. Fungsi ini membangun permintaan dengan parameter yang diperlukan untuk dikirim ke Amazon Lex V2.

```
package com.lex.recognizetext.sample;

import software.amazon.awssdk.auth.credentials.AwsBasicCredentials;
import software.amazon.awssdk.auth.credentials.AwsCredentialsProvider;
import software.amazon.awssdk.auth.credentials.StaticCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.lexruntimev2.LexRuntimeV2Client;
import software.amazon.awssdk.services.lexruntimev2.model.RecognizeTextRequest;
import software.amazon.awssdk.services.lexruntimev2.model.RecognizeTextResponse;

import java.net.URISyntaxException;
import java.util.UUID;


/**
 * This is a sample application to interact with a bot using RecognizeText API.
 */
public class OrderFlowersSampleApplication {

    public static void main(String[] args) throws URISyntaxException,
 InterruptedException {
        String botId = "";
        String botAliasId = "";
        String localeId = "en_US";
        String accessKey = "";
        String secretKey = "";
        String sessionId = UUID.randomUUID().toString();
        Region region = Region.US_EAST_1; // pick an appropriate region

        AwsBasicCredentials awsCreds = AwsBasicCredentials.create(accessKey,
 secretKey);
        AwsCredentialsProvider awsCredentialsProvider =
 StaticCredentialsProvider.create(awsCreds);

        LexRuntimeV2Client lexV2Client = LexRuntimeV2Client
                .builder()
                .credentialsProvider(awsCredentialsProvider)
                .region(region)
```

```
                .build();

        // utterance 1
        String userInput = "I would like to order flowers";
        RecognizeTextRequest recognizeTextRequest = getRecognizeTextRequest(botId,
botAliasId, localeId, sessionId, userInput);
        RecognizeTextResponse recognizeTextResponse =
lexV2Client.recognizeText(recognizeTextRequest);

        System.out.println("User : " + userInput);
        recognizeTextResponse.messages().forEach(message -> {
            System.out.println("Bot : " + message.content());
        });

        // utterance 2
        userInput = "1 dozen roses";
        recognizeTextRequest = getRecognizeTextRequest(botId, botAliasId, localeId,
sessionId, userInput);
        recognizeTextResponse = lexV2Client.recognizeText(recognizeTextRequest);

        System.out.println("User : " + userInput);
        recognizeTextResponse.messages().forEach(message -> {
            System.out.println("Bot : " + message.content());
        });

        // utterance 3
        userInput = "next monday";
        recognizeTextRequest = getRecognizeTextRequest(botId, botAliasId, localeId,
sessionId, userInput);
        recognizeTextResponse = lexV2Client.recognizeText(recognizeTextRequest);

        System.out.println("User : " + userInput);
        recognizeTextResponse.messages().forEach(message -> {
            System.out.println("Bot : " + message.content());
        });

        // utterance 4
        userInput = "5 in evening";
        recognizeTextRequest = getRecognizeTextRequest(botId, botAliasId, localeId,
sessionId, userInput);
        recognizeTextResponse = lexV2Client.recognizeText(recognizeTextRequest);

        System.out.println("User : " + userInput);
        recognizeTextResponse.messages().forEach(message -> {
```

```
            System.out.println("Bot : " + message.content());
        });

        // utterance 5
        userInput = "Yes";
        recognizeTextRequest = getRecognizeTextRequest(botId, botAliasId, localeId,
 sessionId, userInput);
        recognizeTextResponse = lexV2Client.recognizeText(recognizeTextRequest);

        System.out.println("User : " + userInput);
        recognizeTextResponse.messages().forEach(message -> {
            System.out.println("Bot : " + message.content());
        });
    }

    private static RecognizeTextRequest getRecognizeTextRequest(String botId, String
 botAliasId, String localeId, String sessionId, String userInput) {
        RecognizeTextRequest recognizeTextRequest = RecognizeTextRequest.builder()
                .botAliasId(botAliasId)
                .botId(botId)
                .localeId(localeId)
                .sessionId(sessionId)
                .text(userInput)
                .build();
        return recognizeTextRequest;
    }
}
```

Gunakan Ketahanan Global untuk menyebarkan bot ke Wilayah lain

Ketahanan Global memungkinkan Anda mereplikasi bot di wilayah sekunder. Wilayah sekunder dapat dibuat aktif dengan replikasi otomatis bot pengguna di kedua wilayah. Anda akan memiliki wilayah cadangan jika terjadi pemadaman regional. Setelah Ketahanan Global aktif, bot baru yang dibuat direplikasi di wilayah AWS kedua.

> **Note**
>
> Fitur ini hanya tersedia untuk instans Amazon Connect dan Amazon Lex V2 yang dibuat di Wilayah AS Timur (Virginia N.) dan AS Barat (Oregon), dan Wilayah Eropa Barat (London) dan Eropa Tengah (Frankfurt).
> Untuk mendapatkan akses ke fitur ini, hubungi Arsitek Solusi Amazon Connect atau Manajer Akun Teknis Anda.

Setelah mengaktifkan fitur ini, Anda dapat mengotomatiskan replikasi bot Amazon Lex V2 dan sumber daya, versi, dan aliasnya di seluruh wilayah AWS yang dipasangkan dalam waktu dekat. Dengan fitur ini, Anda dapat memantau nomor versi bot asli dan replika untuk memastikan bahwa replika bot tetap sinkron dengan bot asli. Saat mengaktifkan replikasi, Anda dapat mengaktifkan wilayah AWS yang telah ditentukan sebelumnya tempat bot direplikasi (wilayah didasarkan pada pasangan yang telah ditentukan sebelumnya). Setiap pembaruan ke bot sumber di wilayah sumber secara otomatis diperbarui ke bot yang direplikasi di wilayah kedua.

> **Note**
>
> Saat Ketahanan Global diaktifkan untuk bot, semua alias yang ada dan versi terkaitnya direplikasi di wilayah replika. Versi yang tidak terkait dengan alias sebelum mengaktifkan replikasi direplikasi ketika mereka dikaitkan dengan alias. Setiap versi dan alias yang dibuat setelah mengaktifkan replikasi, secara otomatis direplikasi. Pengguna dapat menggunakan `ListBotVersionReplicas` dan `ListBotAliasReplicas` meninjau status replikasi masing-masing versi dan alias individu. Mutasi bot bersifat uni-directional dari bot ke replika nya. Pengguna tidak dapat memodifikasi bot replika karena selalu sinkron dengan bot.

Informasi tambahan tentang penggunaan Ketahanan Global:

- Ketahanan Global saat ini hanya bekerja dengan pasangan wilayah yang telah ditentukan sebelumnya.

us-east-1	us-west-2
eu-west-2	eu-central-1

- Saat Ketahanan Global diaktifkan untuk bot, semua alias yang ada dan versi terkaitnya direplikasi di wilayah replika. Versi yang tidak terkait dengan alias sebelum mengaktifkan replikasi, direplikasi ketika mereka dikaitkan dengan alias. Setiap versi dan alias yang dibuat setelah mengaktifkan replikasi, akan direplikasi secara otomatis. Pengguna dapat menggunakan `ListBotVersionReplicas` dan `ListBotAliasReplicas` mengetahui status replikasi untuk setiap versi individu dan alias. Mutasi bot bersifat uni-directional dari bot ke replika nya. Pengguna tidak dapat memodifikasi bot replika, karena selalu disinkronkan dengan bot.
- Alias apa pun dapat dikaitkan dengan versi apa pun. Jika versi belum direplikasi, itu akan direplikasi selama asosiasi dengan Alias.

Pembatasan:

- Ketahanan Global tidak mereplikasi bot yang dibuat dengan slot yang menggunakan LLM seperti CFAQ dan Utterance Generation.
- Ketahanan Global tidak mereplikasi Jaringan Bot, tetapi bot apa pun yang merupakan bagian dari Jaringan Bot masih dapat direplikasi secara individual.

Topik

- Izin untuk mereplikasi bot dan mengelola replika bot di Lex V2
- Menyebarkan Ketahanan Global dengan bot Lex V2 Anda

Izin untuk mereplikasi bot dan mengelola replika bot di Lex V2

Jika peran IAM memiliki AmazonLexFullAccesskebijakan yang dilampirkan, ia dapat membuat dan mengelola replika bot.

Jika Anda lebih suka membuat peran dengan izin minimal untuk Ketahanan Global, gunakan kebijakan berikut, yang berisi pernyataan berikut.

- Izin untuk mengakses peran terkait layanan Amazon Lex V2 untuk replikasi bot.
- Izin untuk mengizinkan Amazon Lex V2 membuat peran terkait layanan untuk replikasi bot atas nama Anda.
- Izin untuk memanggil replikasi APIs bot.

Anda dapat membatasi izin lebih lanjut dengan memodifikasinya sebagai berikut.

- Ganti * dengan bot atau alias bot tertentu IDs untuk membatasi izin ke bot atau alias bot tertentu.
- Gunakan subset `lex BotReplica` tindakan untuk membatasi peran pada tindakan tertentu.

Sebagai contoh, lihat Izinkan pengguna membuat dan melihat replika bot, tetapi tidak menghapusnya.

Menyebarkan Ketahanan Global dengan bot Lex V2 Anda

Ketahanan Global memungkinkan Anda mereplikasi bot di wilayah sekunder. Wilayah sekunder dapat dibuat aktif dengan replikasi otomatis bot pengguna di kedua wilayah. Anda akan memiliki wilayah cadangan jika terjadi pemadaman regional. Setelah Ketahanan Global aktif, bot baru yang dibuat direplikasi di wilayah AWS kedua.

Panel informasi Ketahanan Global menampilkan detail tentang penerapan Anda

Anda dapat mengakses informasi berikut di panel Ketahanan Global:

- Rincian sumber — Informasi tentang wilayah sumber bot Anda, jenis replika, tanggal diaktifkan replikasi, dan versi terakhir dibuat. Gunakan informasi ini untuk melacak iterasi bot Anda.
- Detail replikasi - Setelah membuat replika bot Anda, Anda dapat melacak wilayah yang direplikasi, jenis replika, tanggal disinkronkan versi terakhir, dan versi replikasi terakhir. Gunakan informasi ini untuk melacak sinkronisasi replika bot Anda.
- Wilayah sumber - Wilayah tempat Ketahanan Global diaktifkan. Anda dapat membuat perubahan di wilayah sumber untuk mereplikasi bot di kedua wilayah.
- Jenis replika - Menunjukkan apakah bot hanya dibaca atau dapat membaca dan menulis berdasarkan wilayah.
- Wilayah replika - Wilayah sekunder yang digunakan untuk mereplikasi bot sumber Anda untuk Ketahanan Global. Ketahanan Global saat ini hanya bekerja dengan IAD/PDX dan pasangan LDN/FRA regional.
- Tanggal diaktifkan replikasi - Tanggal dan waktu replika bot diaktifkan.
- Versi terakhir dibuat — Versi bot terakhir yang terkait dengan replika di wilayah sumber.

Mengaktifkan Ketahanan Global untuk bot Lex V2 Anda

Sebelum mengaktifkan Ketahanan Global di konsol Amazon Lex V2, Anda harus memastikan bahwa pengguna yang mengaktifkan replikasi bot memiliki izin untuk membuat Peran Tertaut Layanan

(SLR). Ketahanan Global akan menggunakan kredensil FAS ini untuk membuat SLR di akun yang diaktifkan saat dipanggil. CreateReplica Untuk informasi selengkapnya tentang menyiapkan SLR untuk Ketahanan Global di Amazon Lex V2, lihat kebijakan terkelola AWS:. AmazonLexFullAccess

> **Note**
>
> Fitur ini hanya tersedia untuk instans Amazon Connect dan Amazon Lex V2 yang dibuat di Wilayah AS Timur (Virginia N.) dan AS Barat (Oregon), dan Wilayah Eropa Barat (London) dan Eropa Tengah (Frankfurt).
> Untuk mendapatkan akses ke fitur ini, hubungi Arsitek Solusi Amazon Connect atau Manajer Akun Teknis Anda.

Aktifkan Ketahanan Global dan siapkan replikasi bot untuk wilayah kedua:

1. Masuk ke AWS Management Console dan buka konsol Amazon Lex di https://console.aws.amazon.com/lex/.
2. Pilih bot yang ingin Anda tiru dari navigasi Bots di panel navigasi sisi kiri.
3. Pilih Deployment > Global Resiliency.
4. Pilih tombol Buat replika di sudut kanan atas jendela untuk membuat versi draf bot Anda.

   > **Note**
   >
   > Periksa untuk memastikan Anda tidak memiliki bot di wilayah sekunder dengan nama yang sama dengan bot yang ingin Anda tiru. (Bot Anda harus diberi nama unik).

5. Buka Ketahanan Global, Klik Buat Replika - Tindakan ini membuat versi draf bot Anda. (Anda tidak perlu kembali ke tab Ketahanan Global kecuali untuk meninjau status atau melihat detail build future).

   > **Note**
   >
   > Anda juga dapat membuat bot Alias untuk replikasi di Global Resiliency dengan membuka Alias dan memilih Create New Alias for Global Resiliency enabled bot. Semua alias akan direplikasi, bahkan jika dibuat sebelum replikasi diaktifkan.

6. Buka Alias - Buat Alias Baru untuk bot yang diaktifkan Ketahanan Global. Semua alias akan direplikasi, bahkan jika mereka dibuat sebelum replikasi diaktifkan.

7. Buka Versi - Buat Versi Baru untuk bot yang diaktifkan Ketahanan Global. Versi apa pun yang terkait dengan alias akan direplikasi, bahkan jika itu dibuat sebelum replikasi diaktifkan.

> **Note**
>
> Pelanggan masih memiliki kendali penuh untuk mengelola kebijakan dan tag berbasis sumber daya mereka untuk bot yang direplikasi. Fungsi Lambda dan Grup CloudWatch Log perlu diterapkan di kedua wilayah dengan pengidentifikasi yang sama. Pengguna tidak perlu mengaitkan fungsi lambda lagi di wilayah replika.

Menonaktifkan Ketahanan Global

Anda dapat menonaktifkan Ketahanan Global kapan saja dengan memilih tombol Nonaktifkan Ketahanan Global. Tindakan ini menghentikan bot sumber Anda dan alias serta versi apa pun yang terkait dengannya agar tidak direplikasi di wilayah lain.

Menggunakan APIs dengan Ketahanan Global untuk bot Lex V2 Anda

Anda dapat melakukan panggilan API dalam Ketahanan Global menggunakan yang berikut ini. APIs Informasi tambahan tentang Ketahanan Global APIs dan Amazon Lex V2 dapat ditemukan di Panduan API Amazon Lex V2.

- CreateBotReplica

  Aktifkan Ketahanan Global dan buat bot yang direplikasi. Membutuhkan`replicaRegion`.

  Untuk informasi selengkapnya, lihat CreateBotReplicadi Panduan API Amazon Lex V2.
- DeleteBotReplica

  Nonaktifkan Ketahanan Global dan hapus bot yang direplikasi. Membutuhkan `replicaRegion` dan`botId`.

  Untuk informasi selengkapnya, lihat DeleteBotReplicadi Panduan API Amazon Lex V2.
- ListBotReplicas

  Buat daftar bot yang direplikasi di zona sekunder. Membutuhkan`botId`.

  Untuk informasi selengkapnya, lihat ListBotReplicasdi Panduan API Amazon Lex V2.

- DescribeBotReplica

  Ringkasan informasi untuk bot yang direplikasi. Membutuhkan `replicaRegion` dan`botId`.

  Untuk informasi selengkapnya, lihat DescribeBotReplicadi Panduan API Amazon Lex V2.

Mengintegrasikan bot Amazon Lex V2 dengan platform perpesanan

Bagian ini menjelaskan cara mengintegrasikan bot Amazon Lex V2 dengan platform perpesanan Facebook, Slack, dan Twilio. Jika Anda belum memiliki bot Amazon Lex V2, buat satu. Dalam topik ini, kami berasumsi bahwa Anda menggunakan bot yang Anda buatLatihan 1: Buat chatbot dari template. Namun, Anda dapat menggunakan bot apa pun.

> **Note**
>
> Saat menyimpan konfigurasi Facebook, Slack, atau Twilio Anda, Amazon Lex V2 menggunakan file untuk mengenkripsi informasi. AWS KMS key Pertama kali Anda membuat saluran ke salah satu platform perpesanan ini, Amazon Lex V2 membuat kunci terkelola pelanggan default (`aws/lex`) di akun AWS Anda atau Anda dapat memilih kunci yang dikelola pelanggan Anda sendiri. Amazon Lex V2 hanya mendukung tombol simetris. Lihat informasi selengkapnya di Panduan Developer AWS Key Management Service.

Saat platform perpesanan mengirimkan permintaan ke Amazon Lex V2, platform tersebut menyertakan informasi khusus platform sebagai atribut permintaan untuk fungsi Lambda Anda. Gunakan atribut ini untuk menyesuaikan cara bot Anda berperilaku. Untuk informasi selengkapnya, lihat Mengatur atribut permintaan untuk bot Lex V2 Anda.

Atribut permintaan umum untuk platform perpesanan

Atribut	Deskripsi
x-amz-lex:saluran:platform	Salah satu nilai berikut: • `Facebook` • `Slack` • `Twilio`

Mengintegrasikan bot Amazon Lex V2 dengan Facebook Messenger

Anda dapat meng-host bot Amazon Lex V2 Anda di Facebook Messenger. Ketika Anda melakukannya, pengguna Facebook dapat berinteraksi dengan bot Anda untuk memenuhi maksud.

Sebelum memulai, Anda harus mendaftar untuk akun pengembang Facebook di https://developers.facebook.com.

Anda perlu melakukan langkah-langkah berikut:

Topik

- Langkah 1: Buat aplikasi Facebook
- Langkah 2: Integrasikan Facebook Messenger dengan bot Amazon Lex V2
- Langkah 3: Selesaikan integrasi Facebook dengan bot Lex V2 Anda
- Langkah 4: Uji integrasi dengan Facebook Messenger

Langkah 1: Buat aplikasi Facebook

Di portal pengembang Facebook, buat aplikasi Facebook dan halaman Facebook.

Untuk membuat aplikasi Facebook

1. https://developers.facebook.comTerbuka/aplikasi
2. Pilih Buat Aplikasi.
3. Di halaman Buat Aplikasi, pilih Bisnis, lalu pilih Berikutnya.
4. Untuk bidang Tambahkan nama aplikasi, email kontak Aplikasi, dan Akun Bisnis, buat pilihan yang sesuai untuk aplikasi Anda. Pilih Buat Aplikasi untuk melanjutkan.
5. Dari Tambahkan Produk ke Aplikasi Anda, pilih Mengatur dari ubin Messenger.
6. Di bagian Access Tokens, pilih Tambah atau Hapus halaman.
7. Pilih halaman yang akan digunakan dengan aplikasi Anda, lalu pilih Berikutnya.
8. Untuk Apa yang diizinkan dilakukan aplikasi, biarkan default lalu pilih Selesai.
9. Di halaman konfirmasi, pilih OK.
10. Di bagian Access Tokens, pilih Generate Token, lalu salin token. Anda memasukkan token ini di konsol Amazon Lex V2.
11. Dari menu kiri, pilih Pengaturan dan kemudian pilih Dasar.

12. Untuk Rahasia Aplikasi, pilih Tampilkan dan kemudian salin rahasianya. Anda memasukkan token ini di konsol Amazon Lex V2.

Langkah berikutnya

Langkah 2: Integrasikan Facebook Messenger dengan bot Amazon Lex V2

Langkah 2: Integrasikan Facebook Messenger dengan bot Amazon Lex V2

Pada langkah ini Anda menautkan bot Amazon Lex V2 Anda dengan Facebook.

1. Masuk ke AWS Management Console dan buka konsol Amazon Lex di https://console.aws.amazon.com/lex/.
2. Dari daftar bot, pilih bot Amazon Lex V2 yang Anda buat.
3. Di menu sebelah kiri, pilih Integrasi saluran dan kemudian pilih Tambah saluran.
4. Di Buat saluran, lakukan hal berikut:

    a. Untuk Platform, pilih Facebook.

    b. Untuk kebijakan Identitas, pilih AWS KMS kunci untuk melindungi informasi saluran. Kunci default disediakan oleh Amazon Lex V2.

    c. Untuk konfigurasi Integrasi, berikan nama saluran dan deskripsi opsional. Pilih alias yang menunjuk ke versi bot yang akan digunakan, dan pilih bahasa yang didukung saluran.

    d. Untuk konfigurasi tambahan, masukkan yang berikut ini:

    - Alias — String yang mengidentifikasi aplikasi yang memanggil Amazon Lex V2. Anda dapat menggunakan string apa pun. Rekam string ini, Anda memasukkannya di konsol pengembang Facebook.
    - Token akses halaman — Token akses halaman yang Anda salin dari konsol pengembang Facebook.
    - Kunci rahasia aplikasi — Kunci rahasia yang Anda salin dari konsol pengembang Facebook.

    e. Pilih Buat

    f. Amazon Lex V2 menunjukkan daftar saluran untuk bot Anda. Dari daftar, pilih saluran yang baru saja Anda buat.

    g. Dari URL Callback, rekam URL callback. Anda memasukkan URL ini di konsol pengembang Facebook.

Langkah berikutnya

Langkah 3: Selesaikan integrasi Facebook dengan bot Lex V2 Anda

Langkah 3: Selesaikan integrasi Facebook dengan bot Lex V2 Anda

Pada langkah ini, gunakan konsol pengembang Facebook untuk menyelesaikan integrasi dengan Amazon Lex V2.

Untuk menyelesaikan integrasi Facebook Messenger

1. https://developers.facebook.comTerbuka/aplikasi
2. Dari daftar aplikasi, pilih aplikasi yang Anda integrasikan dengan Facebook Messenger.
3. Di menu sebelah kiri, pilih Messenger, lalu pilih Pengaturan.
4. Di bagian Webhooks:
    a. Pilih Tambahkan URL Panggilan Balik.
    b. Di Edit URL Callback, masukkan yang berikut ini:
        - URL Callback — Masukkan URL callback yang Anda rekam dari konsol Amazon Lex V2.
        - Verifikasi Token — Masukkan alias yang Anda masukkan di konsol Amazon Lex V2.
    c. Pilih Verifikasi dan Simpan.
    d. Pilih Tambahkan langganan di bawah Webhook di sebelah halaman Anda.
    e. Di jendela yang muncul, pilih `messages` lalu klik Simpan.

Langkah berikutnya

Langkah 4: Uji integrasi dengan Facebook Messenger

Langkah 4: Uji integrasi dengan Facebook Messenger

Anda sekarang dapat memulai percakapan dari Facebook Messenger dengan bot Amazon Lex V2 Anda.

Untuk menguji integrasi antara Facebook Messenger dan bot Amazon Lex V2

1. Buka halaman Facebook yang Anda kaitkan dengan bot Anda di langkah 1.
2. Di jendela Messenger, gunakan ucapan pengujian yang disediakan di. Latihan 1: Buat chatbot dari template

Mengintegrasikan bot Amazon Lex V2 dengan Slack

Topik ini memberikan instruksi untuk mengintegrasikan bot Amazon Lex V2 dengan aplikasi perpesanan Slack. Anda harus melakukan langkah-langkah berikut ini:

Topik

- Langkah 1: Mendaftar untuk Slack dan membuat tim Slack
- Langkah 2: Buat aplikasi Slack
- Langkah 3: Integrasikan aplikasi Slack dengan bot Amazon Lex V2
- Langkah 4: Selesaikan integrasi Slack dengan bot Lex V2 Anda
- Langkah 5: Uji integrasi antara bot Lex V2 Anda dan Slack

Langkah 1: Mendaftar untuk Slack dan membuat tim Slack

Daftar untuk akun Slack dan buat tim Slack. Untuk petunjuk, lihat Menggunakan Slack. Di bagian selanjutnya Anda membuat aplikasi Slack, yang dapat diinstal oleh tim Slack mana pun.

Langkah berikutnya

Langkah 2: Buat aplikasi Slack

Langkah 2: Buat aplikasi Slack

Di bagian ini, Anda melakukan hal berikut:

1. Buat aplikasi Slack di Slack API Console.
2. Konfigurasikan aplikasi untuk menambahkan pesan interaktif ke bot Anda.

Di akhir bagian ini, Anda mendapatkan kredensi aplikasi (ID Klien, Rahasia Klien, dan Token Verifikasi). Pada langkah berikutnya, Anda menggunakan informasi ini untuk mengintegrasikan bot di konsol Amazon Lex V2.

Untuk membuat aplikasi Slack

1. Masuk ke Slack API Console di https://api.slack.com.
2. Membuat aplikasi.

Setelah Anda berhasil membuat aplikasi, Slack menampilkan halaman Informasi Dasar untuk aplikasi.

3. Konfigurasikan fitur aplikasi sebagai berikut:

   - Di menu sebelah kiri, pilih Interaktivitas & Pintasan.

     - Pilih sakelar untuk mengaktifkan komponen interaktif.
     - Di kotak Permintaan URL, tentukan URL yang valid. Misalnya, Anda dapat menggunakan**`https://slack.com`**.

       > **Note**
       >
       > Untuk saat ini, masukkan URL yang valid untuk mendapatkan token verifikasi yang Anda butuhkan di langkah berikutnya. Anda akan memperbarui URL ini setelah menambahkan asosiasi saluran bot di konsol Amazon Lex.

     - Pilih Simpan Perubahan.

4. Di menu sebelah kiri, di Pengaturan, pilih Informasi Dasar. Catat kredensi aplikasi berikut:

   - ID Klien
   - Rahasia Klien
   - Token Verifikasi

Langkah berikutnya

Langkah 3: Integrasikan aplikasi Slack dengan bot Amazon Lex V2

Langkah 3: Integrasikan aplikasi Slack dengan bot Amazon Lex V2

Di bagian ini, integrasikan aplikasi Slack yang Anda buat dengan bot Amazon Lex V2 yang Anda buat dengan menggunakan integrasi Saluran.

1. Masuk ke AWS Management Console dan buka konsol Amazon Lex di https://console.aws.amazon.com/lex/.
2. Dari daftar bot, pilih bot Amazon Lex V2 yang Anda buat.
3. Di menu sebelah kiri, pilih Integrasi saluran dan kemudian pilih Tambah saluran.
4. Di Buat saluran, lakukan hal berikut:

a. Untuk Platform, pilih Slack.

b. Untuk kebijakan Identitas, pilih AWS KMS kunci untuk melindungi informasi saluran. Kunci default disediakan oleh Amazon Lex V2.

c. Untuk konfigurasi Integrasi, berikan nama saluran dan deskripsi opsional. Pilih alias yang menunjuk ke versi bot yang akan digunakan, dan pilih bahasa yang didukung saluran.

> **Note**
>
> Jika bot Anda tersedia dalam berbagai bahasa, Anda harus membuat saluran yang berbeda dan aplikasi yang berbeda untuk setiap bahasa.

d. Untuk konfigurasi tambahan, masukkan yang berikut ini:

- ID Klien — masukkan ID klien dari Slack.
- Rahasia klien — masukkan rahasia klien dari Slack.
- Token verifikasi — masukkan token verifikasi dari Slack.
- URL halaman sukses — URL halaman yang harus dibuka Slack saat pengguna diautentikasi. Biasanya Anda membiarkan ini kosong.

5. Pilih Buat untuk membuat saluran.
6. Amazon Lex V2 menunjukkan daftar saluran untuk bot Anda. Dari daftar, pilih saluran yang baru saja Anda buat.
7. Dari URL Callback, catat titik akhir dan titik akhir. OAuth

Langkah berikutnya

Langkah 4: Selesaikan integrasi Slack dengan bot Lex V2 Anda

Langkah 4: Selesaikan integrasi Slack dengan bot Lex V2 Anda

Di bagian ini, gunakan konsol Slack API untuk menyelesaikan integrasi dengan aplikasi Slack.

1. Masuk ke konsol Slack API di https://api.slack.com. Pilih aplikasi yang Anda buatLangkah 2: Buat aplikasi Slack.
2. Perbarui fitur OAuth & Izin sebagai berikut:

a. Di menu sebelah kiri, pilih OAuth & Izin.

b. Di URLs bagian Redirect, tambahkan OAuth titik akhir yang disediakan Amazon Lex pada langkah sebelumnya. Pilih Tambah, lalu pilih Simpan URLs.

c. Di bagian Bot Token Scopes, tambahkan dua izin dengan tombol Add an OAuth Scope. Filter daftar dengan teks berikut:

   - **`chat:write`**
   - **`team:read`**

3. Perbarui fitur Interaktivitas & Pintasan dengan memperbarui nilai URL Permintaan ke titik akhir yang disediakan Amazon Lex pada langkah sebelumnya. Masukkan titik akhir yang Anda simpan di langkah 3, lalu pilih Simpan Perubahan.
4. Berlangganan fitur Langganan Acara sebagai berikut:

   - Aktifkan acara dengan memilih opsi Aktif.
   - Tetapkan nilai URL Permintaan ke titik akhir yang disediakan Amazon Lex pada langkah sebelumnya.
   - Di bagian Berlangganan Acara Bot, pilih Tambahkan Acara Pengguna Bot dan tambahkan acara **`message.im`** bot untuk mengaktifkan pesan langsung antara pengguna akhir dan bot Slack.
   - Simpan perubahan.
5. Aktifkan pengiriman pesan dari tab pesan sebagai berikut:

   - Dari menu sebelah kiri, pilih App Home.
   - Di bagian Tampilkan Tab, pilih Izinkan pengguna mengirim perintah garis miring dan pesan dari tab pesan.
6. Pilih Kelola Distribusi di bawah Pengaturan. Pilih Tambahkan ke Slack untuk menginstal aplikasi. Jika Anda diautentikasi ke beberapa ruang kerja, pertama-tama pilih ruang kerja yang benar di sudut kanan atas dari daftar drop-down. Selanjutnya, pilih Izinkan untuk mengotorisasi bot untuk menanggapi pesan.

> **Note**
>
> Jika Anda membuat perubahan pada pengaturan aplikasi Slack nanti, Anda harus mengulang sublangkah ini.

Langkah berikutnya

Langkah 5: Uji integrasi antara bot Lex V2 Anda dan Slack

Langkah 5: Uji integrasi antara bot Lex V2 Anda dan Slack

Sekarang gunakan jendela browser untuk menguji integrasi Slack dengan bot Amazon Lex V2 Anda.

Untuk menguji aplikasi Slack Anda

1. Luncurkan Slack. Dari menu kiri, di bagian Direct Messages, pilih bot Anda. Jika Anda tidak melihat bot Anda, pilih ikon plus (+) di sebelah Direct Message untuk mencarinya.
2. Terlibat dalam obrolan dengan aplikasi Slack Anda. Bot Anda merespons pesan.

   Jika Anda membuat bot menggunakanLatihan 1: Buat chatbot dari template, Anda dapat menggunakan contoh percakapan dari latihan itu.

Mengintegrasikan bot Amazon Lex V2 dengan Twilio SMS

Topik ini memberikan instruksi untuk mengintegrasikan bot Amazon Lex V2 dengan layanan pesan sederhana Twilio (SMS). Anda harus melakukan langkah-langkah berikut ini:

Topik

- Langkah 1: Buat akun SMS Twilio
- Langkah 2: Integrasikan titik akhir layanan pesan Twilio dengan bot Amazon Lex V2
- Langkah 3: Selesaikan integrasi Twilio antara bot Lex V2 Anda dan Twilio
- Langkah 4: Uji integrasi antara bot Lex V2 Anda dan Twilio

Langkah 1: Buat akun SMS Twilio

Mendaftar untuk akun Twilio dan catat informasi akun berikut:

- AKUN SID
- TOKEN AUTENTIKASI

Untuk petunjuk pendaftaran, lihat https://www.twilio.com/console.

Langkah berikutnya

Langkah 2: Integrasikan titik akhir layanan pesan Twilio dengan bot Amazon Lex V2

Langkah 2: Integrasikan titik akhir layanan pesan Twilio dengan bot Amazon Lex V2

1. Masuk ke AWS Management Console dan buka konsol Amazon Lex di https://console.aws.amazon.com/lex/.
2. Dari daftar bot, pilih bot Amazon Lex V2 yang Anda buat.
3. Di menu sebelah kiri, pilih Integrasi saluran dan kemudian pilih Tambah saluran.
4. Di Buat saluran, lakukan hal berikut:
    a. Untuk Platform, pilih Twilio.
    b. Untuk kebijakan Identitas, pilih AWS KMS kunci untuk melindungi informasi saluran. Kunci default disediakan oleh Amazon Lex V2.
    c. Untuk konfigurasi Integrasi, berikan nama saluran dan deskripsi opsional. Pilih alias yang menunjuk ke versi bot yang akan digunakan, dan pilih bahasa yang didukung saluran.
    d. Untuk konfigurasi tambahan, masukkan SID akun dan token otentikasi dari dasbor Twilio.
5. Pilih Buat.
6. Dari daftar saluran, pilih saluran yang baru saja Anda buat.
7. Salin URL Callback.

Langkah berikutnya

Langkah 3: Selesaikan integrasi Twilio antara bot Lex V2 Anda dan Twilio

Langkah 3: Selesaikan integrasi Twilio antara bot Lex V2 Anda dan Twilio

Gunakan konsol Twilio untuk menyelesaikan integrasi bot Amazon Lex V2 Anda dengan Twilio SMS.

1. Buka konsol Twilio di https://www.twilio.com /console.
2. Dari menu sebelah kiri, pilih Semua Produk & Layanan, lalu pilih Nomor Telepon.
3. Jika Anda memiliki nomor telepon, pilihlah. Jika Anda tidak memiliki nomor telepon, pilih Beli Nomor untuk mendapatkannya.
4. Di bagian Pesan, di PESAN MASUK, masukkan URL panggilan balik dari konsol Amazon Lex V2.
5. Pilih Simpan.

Langkah berikutnya

Langkah 4: Uji integrasi antara bot Lex V2 Anda dan Twilio

Langkah 4: Uji integrasi antara bot Lex V2 Anda dan Twilio

Gunakan ponsel Anda untuk menguji integrasi antara Twilio SMS dan bot Anda. Menggunakan ponsel Anda, kirim pesan ke nomor Twilio.

Jika Anda membuat bot menggunakanLatihan 1: Buat chatbot dari template, Anda dapat menggunakan contoh percakapan dari latihan itu.

Mengintegrasikan bot Amazon Lex V2 dengan pusat kontak

Anda dapat mengintegrasikan bot Amazon Lex V2 dengan pusat kontak Anda untuk mengaktifkan kasus penggunaan swalayan menggunakan API streaming Amazon Lex V2. Gunakan bot ini sebagai agen respons suara interaktif (IVR) di telepon atau sebagai chatbot berbasis teks yang terintegrasi ke dalam pusat kontak Anda. Untuk informasi selengkapnya tentang API streaming, lihatStreaming percakapan ke bot Amazon Lex V2.

Dengan API streaming, Anda dapat mengaktifkan fitur-fitur berikut:

- Interupsi (“barge-in”) — Penelepon dapat mengganggu bot dan menjawab pertanyaan sebelum prompt selesai. Untuk informasi selengkapnya, lihat Mengaktifkan bot Amazon Lex V2 Anda terganggu oleh pengguna.
- Tunggu dan lanjutkan — Penelepon dapat menginstruksikan bot untuk menunggu jika mereka membutuhkan waktu untuk mengambil informasi tambahan selama panggilan, seperti nomor kartu kredit atau ID pemesanan. Untuk informasi selengkapnya, lihat Mengaktifkan bot Amazon Lex V2 untuk menunggu pengguna memberikan informasi lebih lanjut selama jeda.
- Dukungan DTMF - Penelepon dapat memberikan informasi melalui pidato atau DTMF secara bergantian.
- Dukungan SSML - Anda dapat mengonfigurasi permintaan bot Amazon Lex V2 menggunakan tag SSML untuk kontrol yang lebih besar atas pembuatan ucapan dari teks. Untuk informasi selengkapnya, lihat Menghasilkan pidato dari dokumen SSML di panduan pengembang Amazon Polly.
- Batas waktu yang dapat dikonfigurasi — Anda dapat mengonfigurasi berapa lama menunggu pelanggan selesai berbicara sebelum Amazon Lex V2 mengumpulkan masukan ucapan mereka,

seperti menjawab pertanyaan ya atau tidak, atau memberikan tanggal atau nomor kartu kredit. Untuk informasi selengkapnya, lihat Mengkonfigurasi batas waktu untuk menangkap input pengguna dengan bot Lex V2.

- Pembaruan kemajuan pemenuhan - Anda dapat mengonfigurasi bot untuk merespons dengan beberapa pesan berdasarkan status pemenuhan selama eksekusi logika bisnis untuk pemenuhan maksud. Anda dapat mengatur bot untuk merespons dengan pesan ketika pemenuhan dimulai dan selesai, dan menyediakan pembaruan berkala untuk fungsi Lambda yang berjalan lama. Untuk informasi selengkapnya, lihat Mengkonfigurasi pembaruan kemajuan pemenuhan untuk bot Lex V2 Anda.

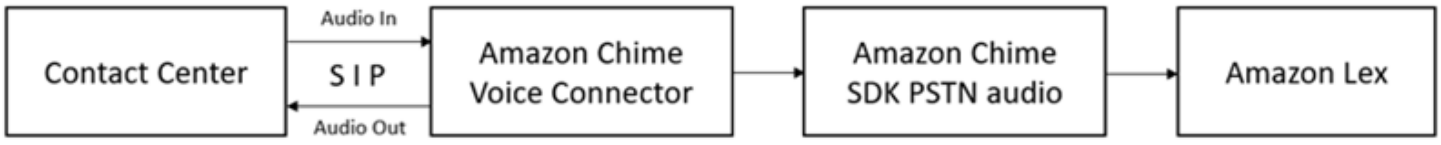
Amazon Chime SDK

Gunakan Amazon Chime SDK untuk menambahkan kemampuan audio, video, berbagi layar, dan pesan real-time ke aplikasi web atau seluler Anda. Amazon Chime SDK menyediakan layanan audio public switched telephone network (PSTN) sehingga Anda dapat membangun aplikasi telepon khusus dengan suatu fungsi. AWS Lambda

Audio Amazon Chime PSTN terintegrasi dengan Amazon Lex V2. Anda dapat menggunakan integrasi ini untuk mengakses bot Amazon Lex V2 sebagai sistem respons suara interaktif (IVR) di pusat kontak untuk interaksi audio. Gunakan ini untuk mengintegrasikan Amazon Lex V2 menggunakan layanan audio PSTN dalam skenario berikut.

Integrasi pusat kontak —Anda dapat menggunakan Amazon Chime Voice Connector dan layanan audio Amazon Chime SDK PSTN untuk mengakses bot Amazon Lex V2. Gunakan mereka dalam aplikasi pusat kontak apa pun yang menggunakan protokol inisiasi sesi (SIP) untuk komunikasi suara. Integrasi ini menambahkan pengalaman percakapan suara bahasa alami ke pusat kontak lokal atau berbasis Internet yang ada dengan dukungan SIP. Untuk daftar platform pusat kontak yang didukung, lihat sumber daya konektor suara Amazon Chime.

Diagram berikut menunjukkan integrasi antara pusat kontak menggunakan SIP dan Amazon Lex V2.



Dukungan telepon langsung —Anda dapat membuat solusi IVR yang disesuaikan untuk mengakses bot Amazon Lex V2 secara langsung menggunakan nomor telepon yang disediakan di Amazon Chime SDK.

Untuk informasi selengkapnya, lihat topik berikut di panduan Amazon Chime SDK.

- Integrasi SIP menggunakan konektor suara Amazon Chime
- Menggunakan layanan audio Amazon Chime SDK PSTN
- Mengintegrasikan audio Amazon Chime PSTN dengan Amazon Lex V2

Saat Amazon Chime SDK mengirimkan permintaan ke Amazon Lex V2, itu menyertakan informasi khusus platform ke fungsi Lambda dan log percakapan Anda. Gunakan informasi ini untuk menentukan aplikasi pusat kontak yang mengirimkan lalu lintas ke bot Anda.

Atribut permintaan umum	Nilai
x-amz-lex:saluran:platform	`Amazon Chime SDK PSTN Audio`

Connect Pelanggan

Connect Customer adalah pusat kontak cloud omnichannel. Anda dapat mengatur pusat kontak dalam beberapa langkah, menambahkan agen yang berlokasi di mana saja, dan mulai terlibat dengan pelanggan Anda. Untuk informasi selengkapnya, lihat Memulai Connect Customer di panduan administrator Connect Customer.

Anda dapat membuat pengalaman yang dipersonalisasi untuk pelanggan Anda menggunakan komunikasi omnichannel. Misalnya, Anda dapat menawarkan obrolan dan kontak suara berdasarkan preferensi pelanggan dan perkiraan waktu tunggu. Sementara itu agen dapat menangani semua pelanggan hanya dari satu antarmuka. Misalnya, mereka dapat mengobrol dengan pelanggan, dan membuat atau menanggapi tugas saat mereka diarahkan ke mereka.

Anda dapat menggunakan Connect Customer untuk interaksi audio dengan pelanggan Anda, atau Connect Customer Chat untuk interaksi teks saja.

Untuk informasi selengkapnya, lihat topik berikut di panduan Connect Customer administrator.

- Apa itu Connect Customer
- Tambahkan bot Amazon Lex V2
- Connect Customer mendapatkan blok kontak masukan pelanggan

Saat pusat kontak mengirimkan permintaan ke Amazon Lex V2, itu menyertakan informasi khusus platform sebagai atribut permintaan ke fungsi Lambda dan log percakapan Anda. Gunakan informasi ini untuk menentukan aplikasi pusat kontak mana yang mengirimkan lalu lintas ke bot Anda.

Atribut permintaan umum untuk Amazon Connect

Atribut	Nilai
x-amz-lex:saluran:platform	Salah satu nilai berikut: • `Connect` • `Connect Chat`

Awan Genesys

Genesys Cloud adalah rangkaian layanan cloud untuk komunikasi perusahaan, kolaborasi, dan manajemen pusat kontak. Genesys Cloud dibangun di atas AWS dan menggunakan lingkungan cloud terdistribusi yang menyediakan akses aman ke organisasi di sekitar pekerjaan.

Untuk informasi selengkapnya, lihat halaman berikut di situs web Genesys Cloud.

- Tentang pusat kontak Genesys Cloud
- Tentang integrasi Amazon Lex V2

Ketika pusat kontak mengirim permintaan ke Amazon Lex V2, itu menyertakan informasi khusus platform sebagai atribut permintaan ke fungsi Lambda dan log percakapan Anda. Gunakan informasi ini untuk menentukan aplikasi pusat kontak mana yang mengirimkan lalu lintas ke bot Anda.

Atribut permintaan umum untuk Genesys Cloud

Atribut	Nilai
x-amz-lex:saluran:platform	• `Genesys Cloud`

Pelajari selengkapnya

- Hidupkan pusat kontak Anda dengan Amazon Lex dan Genesys Cloud

Memahami percakapan bot Amazon Lex V2

Setelah Anda membangun bot, Anda mengintegrasikan aplikasi klien Anda dengan operasi runtime Amazon Lex V2 untuk mengadakan percakapan dengan bot Anda. Saat pengguna memulai percakapan dengan bot Anda, Amazon Lex V2 membuat sesi. Sesi merangkum informasi yang dipertukarkan antara aplikasi Anda dan bot. Untuk informasi selengkapnya, lihat Memahami sesi bot Amazon Lex V2.

Percakapan tipikal melibatkan arus bolak-balik antara pengguna dan bot. Contoh:

```
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.
```

Gunakan operasi RecognizeTextatau RecognizeUtteranceAPI untuk mengelola percakapan sendiri. Gunakan operasi StartConversationAPI untuk memungkinkan Amazon Lex V2 mengelola percakapan untuk Anda.

Untuk mengelola percakapan, Anda harus mengirim ucapan pengguna ke bot hingga percakapan mencapai akhir yang logis. Percakapan saat ini ditangkap dalam status sesi. Status sesi diperbarui setelah setiap ucapan pengguna. Status sesi berisi status percakapan saat ini dan dikembalikan oleh bot sebagai respons terhadap setiap ucapan pengguna.

Percakapan dapat dilakukan di salah satu negara bagian berikut:

- ElicitIntent— Menunjukkan bahwa bot belum menentukan maksud pengguna.
- ElicitSlot— Menunjukkan bahwa bot telah mendeteksi maksud pengguna dan mengumpulkan informasi yang diperlukan untuk memenuhi maksud.
- ConfirmIntent— Menunjukkan bahwa bot sedang menunggu pengguna untuk mengkonfirmasi bahwa informasi yang dikumpulkan sudah benar.
- Ditutup - Menunjukkan bahwa maksud pengguna telah selesai dan bahwa percakapan dengan bot mencapai akhir yang logis.

Pengguna dapat menentukan intent baru setelah intent pertama selesai. Untuk informasi selengkapnya, lihat Konteks percakapan dengan bot Lex V2 Anda.

Sebuah maksud dapat memiliki salah satu dari status berikut:

- InProgress— Menunjukkan bahwa bot mengumpulkan informasi yang diperlukan untuk menyelesaikan maksud. Ini dalam hubungannya dengan keadaan `ElicitSlot` percakapan.
- Menunggu — Menunjukkan bahwa pengguna meminta bot untuk menunggu ketika bot meminta informasi untuk slot tertentu.
- Terpenuhi - Menunjukkan bahwa logika bisnis dalam fungsi Lambda yang terkait dengan maksud berjalan dengan sukses.
- ReadyForFulfillment— Menunjukkan bahwa bot mengumpulkan semua informasi yang diperlukan untuk memenuhi maksud dan bahwa aplikasi klien dapat menjalankan logika bisnis pemenuhan.
- Gagal - Menunjukkan bahwa maksud telah gagal.

Lihat topik berikut untuk mempelajari cara menggunakan Amazon Lex V2 APIs untuk mengelola konteks percakapan dan sesi antara bot dan pengguna Anda.

Topik

- Konteks percakapan dengan bot Lex V2 Anda
- Memahami sesi bot Amazon Lex V2

Konteks percakapan dengan bot Lex V2 Anda

Konteks percakapan adalah informasi yang disediakan pengguna, aplikasi klien, atau fungsi Lambda ke bot Amazon Lex V2 untuk memenuhi maksud. Konteks percakapan mencakup data slot yang disediakan pengguna, atribut permintaan yang ditetapkan oleh aplikasi klien, dan atribut sesi yang dibuat oleh aplikasi klien dan fungsi Lambda.

Topik

- Mengatur konteks maksud untuk bot Lex V2 Anda
- Menggunakan nilai slot default dalam maksud untuk bot Lex V2 Anda
- Mengatur atribut sesi untuk bot Lex V2 Anda
- Mengatur atribut permintaan untuk bot Lex V2 Anda

- Mengatur batas waktu sesi
- Berbagi informasi antar maksud dengan bot Lex V2 Anda
- Mengatur atribut kompleks di bot Lex V2 Anda

Mengatur konteks maksud untuk bot Lex V2 Anda

Anda dapat memiliki maksud pemicu Amazon Lex V2 berdasarkan konteks. Konteks adalah variabel status yang dapat dikaitkan dengan maksud saat Anda mendefinisikan bot. Anda mengonfigurasi konteks untuk maksud saat membuat maksud menggunakan konsol atau menggunakan operasi. CreateIntent Anda hanya dapat menggunakan konteks dalam bahasa Inggris (AS) (en-US) lokal.

Ada dua jenis hubungan untuk konteks, konteks keluaran dan konteks input. Konteks keluaran menjadi aktif ketika maksud terkait terpenuhi. Konteks keluaran dikembalikan ke aplikasi Anda dalam respons dari RecognizeUtteranceoperasi RecognizeTextatau, dan diatur untuk sesi saat ini. Setelah konteks diaktifkan, konteks tetap aktif untuk jumlah putaran atau batas waktu yang dikonfigurasi saat konteks ditentukan.

Konteks input menentukan kondisi di mana maksud dapat dikenali. Maksud hanya dapat dikenali selama percakapan ketika semua konteks inputnya aktif. Maksud tanpa konteks input selalu memenuhi syarat untuk pengakuan.

Amazon Lex V2 secara otomatis mengelola siklus hidup konteks yang diaktifkan dengan memenuhi maksud dengan konteks keluaran. Anda juga dapat mengatur konteks aktif dalam panggilan ke `RecognizeText` atau `RecognizeUtterance` operasi.

Anda juga dapat mengatur konteks percakapan menggunakan fungsi Lambda untuk intent. Konteks keluaran dari Amazon Lex V2 dikirim ke peristiwa input fungsi Lambda. Fungsi Lambda dapat mengirim konteks dalam responsnya. Untuk informasi selengkapnya, lihat Mengintegrasikan AWS Lambda fungsi ke bot Amazon Lex V2 Anda.

Misalnya, Anda memiliki maksud untuk memesan mobil sewaan yang dikonfigurasi untuk mengembalikan konteks keluaran yang disebut “book_car_filled”. Ketika intent terpenuhi, Amazon Lex V2 menyetel variabel konteks keluaran “book_car_fulfilled”. Karena “book_car_fulfilled” adalah konteks aktif, maksud dengan konteks “book_car_fulfilled” yang ditetapkan sebagai konteks input sekarang dipertimbangkan untuk dikenali, selama ucapan pengguna diakui sebagai upaya untuk memperoleh maksud tersebut. Anda dapat menggunakan ini untuk maksud yang hanya masuk akal setelah memesan mobil, seperti mengirim email tanda terima atau memodifikasi reservasi.

Konteks keluaran maksud untuk bot Lex V2 Anda

Amazon Lex V2 membuat konteks keluaran intent aktif saat intent terpenuhi. Anda dapat menggunakan konteks keluaran untuk mengontrol maksud yang memenuhi syarat untuk menindaklanjuti maksud saat ini.

Setiap konteks memiliki daftar parameter yang dipertahankan dalam sesi. Parameter adalah nilai slot untuk maksud yang terpenuhi. Anda dapat menggunakan parameter ini untuk mengisi otomatis nilai slot untuk intent lainnya. Untuk informasi selengkapnya, lihat Menggunakan nilai slot default dalam maksud untuk bot Lex V2 Anda.

Anda mengonfigurasi konteks keluaran saat membuat intent dengan konsol atau dengan CreateIntentoperasi. Anda dapat mengonfigurasi intent dengan lebih dari satu konteks keluaran. Ketika maksud terpenuhi, semua konteks output diaktifkan dan dikembalikan dalam respons atau. RecognizeTextRecognizeUtterance

Saat Anda menentukan konteks keluaran, Anda juga menentukan waktunya untuk hidup, lamanya waktu atau jumlah putaran yang konteksnya disertakan dalam tanggapan dari Amazon Lex V2. Giliran adalah satu permintaan dari aplikasi Anda ke Amazon Lex V2. Setelah jumlah putaran atau waktu kedaluwarsa, konteksnya tidak lagi aktif.

Aplikasi Anda dapat menggunakan konteks output sesuai kebutuhan. Misalnya, aplikasi Anda dapat menggunakan konteks output untuk:

- Ubah perilaku aplikasi berdasarkan konteksnya. Misalnya, aplikasi perjalanan dapat memiliki tindakan yang berbeda untuk konteks “book_car_fulfilled” daripada “rental_hotel_fulfilled.”
- Kembalikan konteks output ke Amazon Lex V2 sebagai konteks masukan untuk ucapan berikutnya. Jika Amazon Lex V2 mengenali ucapan sebagai upaya untuk mendapatkan maksud, Amazon Lex V2 menggunakan konteks untuk membatasi maksud yang dapat dikembalikan ke maksud dengan konteks yang ditentukan.

Masukan konteks maksud untuk bot Lex V2 Anda

Anda menetapkan konteks input untuk membatasi titik dalam percakapan tempat maksud dikenali. Maksud tanpa konteks input selalu memenuhi syarat untuk dikenali.

Anda menyetel konteks input yang ditanggapi intent menggunakan konsol atau operasi. `CreateIntent` Sebuah intent dapat memiliki lebih dari satu konteks masukan.

Untuk maksud dengan lebih dari satu konteks masukan, semua konteks harus aktif untuk memicu maksud. Anda dapat mengatur konteks input saat Anda memanggil RecognizeText, RecognizeUtterance, atau PutSessionoperasi.

Anda dapat mengonfigurasi slot dalam maksud untuk mengambil nilai default dari konteks aktif saat ini. Nilai default digunakan saat Amazon Lex V2 mengenali maksud baru tetapi tidak menerima nilai slot. Anda menentukan nama konteks dan nama slot dalam formulir `#context-name.parameter-name` saat Anda menentukan slot. Untuk informasi selengkapnya, lihat Menggunakan nilai slot default dalam maksud untuk bot Lex V2 Anda.

Menggunakan nilai slot default dalam maksud untuk bot Lex V2 Anda

Saat Anda menggunakan nilai default, Anda menentukan sumber untuk nilai slot yang akan diisi untuk maksud baru ketika tidak ada slot yang disediakan oleh input pengguna. Sumber ini dapat berupa dialog sebelumnya, atribut permintaan atau sesi, atau nilai tetap yang Anda tetapkan pada waktu pembuatan.

Anda dapat menggunakan berikut ini sebagai sumber untuk nilai default Anda.

- Dialog sebelumnya (konteks) - #context -name.parameter-name
- Atribut sesi - [atribut-nama]
- Atribut permintaan — <attribute-name>
- Nilai tetap - Nilai apa pun yang tidak cocok dengan sebelumnya

Saat Anda menggunakan CreateIntentoperasi untuk menambahkan slot ke intent, Anda dapat menambahkan daftar nilai default. Nilai default digunakan dalam urutan yang terdaftar. Misalnya, Anda memiliki maksud dengan slot dengan definisi berikut:

```
"slots": [
    {
        "botId": "string",
        "defaultValueSpec": {
            "defaultValueList": [
                {
                    "defaultValue": "#book-car-fulfilled.startDate"
                },
                {
                    "defaultValue": "[reservationStartDate]"
                }
```

```
            ]
        },
        Other slot configuration settings
    }
]
```

Ketika maksud dikenali, slot bernama “reservation-start-date” memiliki nilai yang ditetapkan ke salah satu dari berikut ini.

1. Jika konteks “book-car-filled” aktif, nilai parameter “startDate” digunakan sebagai nilai default.
2. Jika konteks “book-car-filled” tidak aktif, atau jika parameter “startDate” tidak disetel, nilai atribut sesi “reservasiStartDate” digunakan sebagai nilai default.
3. Jika tidak satu pun dari dua nilai default pertama yang digunakan, maka slot tidak memiliki nilai default dan Amazon Lex V2 akan memperoleh nilai seperti biasa.

Jika nilai default digunakan untuk slot, slot tidak ditimbulkan bahkan jika diperlukan.

Mengatur atribut sesi untuk bot Lex V2 Anda

Atribut sesi berisi informasi khusus aplikasi yang diteruskan antara bot dan aplikasi klien selama sesi. Amazon Lex V2 meneruskan atribut sesi ke semua fungsi Lambda yang dikonfigurasi untuk bot. Jika fungsi Lambda menambahkan atau memperbarui atribut sesi, Amazon Lex V2 meneruskan informasi baru kembali ke aplikasi klien.

Gunakan atribut sesi dalam fungsi Lambda Anda untuk menginisialisasi bot dan untuk menyesuaikan prompt dan kartu respons. Contoh:

- Inisialisasi — Dalam bot pemesanan pizza, aplikasi klien melewati lokasi pengguna sebagai atribut sesi dalam panggilan pertama ke RecognizeUtteranceoperasi RecognizeTextatau. Misalnya, `"Location": "111 Maple Street"`. Fungsi Lambda menggunakan informasi ini untuk menemukan restoran pizza terdekat untuk melakukan pemesanan.
- Personalisasi prompt - Konfigurasikan prompt dan kartu respons untuk merujuk ke atribut sesi. Misalnya, “Hei [FirstName], topping apa yang Anda inginkan?” Jika Anda meneruskan nama depan pengguna sebagai atribut session (`{"FirstName": "Vivian"}`), Amazon Lex mengganti nama untuk placeholder. Kemudian mengirimkan prompt yang dipersonalisasi kepada pengguna, “Hai Vivian, topping mana yang Anda inginkan?”

Atribut sesi bertahan selama sesi berlangsung. Amazon Lex V2 menyimpannya di penyimpanan data terenkripsi hingga sesi berakhir. Klien dapat membuat atribut sesi dalam permintaan dengan memanggil RecognizeUtteranceoperasi RecognizeTextatau dengan `sessionAttributes` bidang yang disetel ke nilai. Fungsi Lambda dapat membuat atribut sesi dalam respons. Setelah klien atau fungsi Lambda membuat atribut session, nilai atribut yang disimpan digunakan kapan saja aplikasi klien tidak menyertakan `sessionAttribute` bidang dalam permintaan ke Amazon Lex V2.

Misalnya, Anda memiliki dua atribut sesi,`{"x": "1", "y": "2"}`. Jika klien memanggil `RecognizeUtterance` operasi `RecognizeText` atau tanpa menentukan `sessionAttributes` bidang, Amazon Lex V2 memanggil fungsi Lambda dengan atribut `{"x": 1, "y": 2}` sesi tersimpan (). Jika fungsi Lambda tidak menampilkan atribut sesi, Amazon Lex V2 mengembalikan atribut sesi tersimpan ke aplikasi klien.

Jika aplikasi klien atau fungsi Lambda melewati atribut sesi, Amazon Lex V2 memperbarui atribut sesi yang disimpan. Melewati nilai yang ada, seperti `{"x": 2}`, memperbarui nilai yang disimpan. Jika Anda melewatkan satu set atribut sesi baru, seperti`{"z": 3}`, nilai yang ada akan dihapus dan hanya nilai baru yang disimpan. Ketika peta kosong,`{}`, dilewatkan, nilai yang disimpan dihapus.

Untuk mengirim atribut sesi ke Amazon Lex V2, Anda membuat peta string-ke-string atribut. Berikut ini menunjukkan cara memetakan atribut sesi:

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

Untuk `RecognizeText` operasi, Anda memasukkan peta ke dalam badan permintaan menggunakan `sessionAttributes` bidang `sessionState` struktur, sebagai berikut:

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

Untuk `RecognizeUtterance` operasi, Anda base64 menyandikan peta, dan kemudian mengirimkannya sebagai bagian dari header. `x-amz-lex-session-state`

Jika Anda mengirim data biner atau terstruktur dalam atribut sesi, Anda harus terlebih dahulu mengubah data menjadi string sederhana. Untuk informasi selengkapnya, lihat Mengatur atribut kompleks di bot Lex V2 Anda.

Atribut sesi yang telah ditentukan

Amazon Lex menyediakan atribut sesi yang telah ditentukan sebelumnya. Atribut ini mengatur cara Amazon Lex memproses informasi yang dikirim ke bot Anda. Atribut bertahan untuk seluruh sesi. Semua atribut yang telah ditentukan ada di `x-amz-lex:` namespace.

Meningkatkan kode hook Lambda timeout

Untuk meningkatkan nilai batas waktu Lambda hook kode di bot Anda, yaitu 30 detik secara default, gunakan atribut session. `x-amz-lex:codehook-timeout-ms` Mengatur nilai atribut sesi dalam milidetik. Batas waktu maksimum adalah 120 detik.

Misalnya, jika pengguna menyetel “x-amz-lex:codehook-timeout-ms”: “90000", batas waktu tunggu Lambda kait kode akan menjadi 90 detik.

Mengatur atribut permintaan untuk bot Lex V2 Anda

Atribut permintaan berisi informasi khusus permintaan dan hanya berlaku untuk permintaan saat ini. Aplikasi klien mengirimkan informasi ini ke Amazon Lex V2. Gunakan atribut permintaan untuk meneruskan informasi yang tidak perlu bertahan selama seluruh sesi. Anda dapat membuat atribut permintaan Anda sendiri atau Anda dapat menggunakan atribut yang telah ditentukan sebelumnya. Untuk mengirim atribut permintaan, gunakan `x-amz-lex-request-attributes` header dalam RecognizeUtteranceatau `requestAttributes` bidang dalam RecognizeTextpermintaan. Karena atribut permintaan tidak bertahan di seluruh permintaan seperti atribut sesi, atribut tersebut tidak dikembalikan `RecognizeUtterance` atau ditanggapi`RecognizeText`.

> **Note**
>
> Untuk mengirim informasi yang tetap ada di seluruh permintaan, gunakan atribut sesi.

Mengatur atribut permintaan yang ditentukan pengguna untuk setiap permintaan bot Lex V2

Atribut permintaan yang ditentukan pengguna adalah data yang Anda kirim ke bot Anda di setiap permintaan. Anda mengirim informasi di `amz-lex-request-attributes` header

`RecognizeUtterance` permintaan atau di `requestAttributes` bidang `RecognizeText` permintaan.

Untuk mengirim atribut permintaan ke Amazon Lex V2, Anda membuat peta string-ke-string atribut. Berikut ini menunjukkan cara memetakan atribut permintaan:

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

Untuk `PostText` operasi, Anda memasukkan peta ke dalam badan permintaan menggunakan `requestAttributes` bidang, sebagai berikut:

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

Untuk `PostContent` operasi, Anda base64 menyandikan peta, dan kemudian mengirimkannya sebagai header. `x-amz-lex-request-attributes`

Jika Anda mengirim data biner atau terstruktur dalam atribut permintaan, Anda harus terlebih dahulu mengubah data menjadi string sederhana. Untuk informasi selengkapnya, lihat Mengatur atribut kompleks di bot Lex V2 Anda.

Amazon Lex V2 menyediakan atribut permintaan yang telah ditentukan untuk mengelola cara memproses informasi yang dikirim ke bot Anda. Atribut tidak bertahan untuk seluruh sesi, jadi Anda harus mengirim atribut yang telah ditentukan di setiap permintaan. Semua atribut yang telah ditentukan ada di `x-amz-lex:` namespace.

Selain atribut yang telah ditentukan berikut, Amazon Lex menyediakan atribut yang telah ditentukan untuk platform perpesanan. Untuk daftar atribut tersebut, lihat Menerapkan Bot Amazon Lex di Platform Perpesanan.

Mengatur Jenis Respons

Jika Anda memiliki dua aplikasi klien yang memiliki kemampuan berbeda, Anda mungkin perlu membatasi format pesan dalam respons. Misalnya, Anda mungkin ingin membatasi pesan yang

dikirim ke klien Web ke teks biasa, tetapi memungkinkan klien seluler untuk menggunakan teks biasa dan Speech Synthesis Markup Language (SSML). Untuk mengatur format pesan yang dikembalikan oleh `PostContent` dan `PostText` operasi, gunakan atribut permintaan x-amz-lex:accept-content-types".

Anda dapat mengatur atribut ke kombinasi jenis pesan berikut:

- `PlainText`— Pesan berisi UTF-8 teks biasa.
- `SSML`— Pesan berisi teks yang diformat untuk output suara.
- `CustomPayload`— Pesan berisi format khusus yang telah Anda buat untuk klien Anda. Anda dapat menentukan payload untuk memenuhi kebutuhan aplikasi Anda.

Amazon Lex V2 hanya mengembalikan pesan dengan jenis yang ditentukan di bidang Pesan respons. Anda dapat mengatur lebih dari satu nilai dengan memisahkan nilai dengan koma. Jika Anda menggunakan grup pesan, setiap grup pesan harus berisi setidaknya satu pesan dari jenis yang ditentukan. Jika tidak, Anda mendapatkan `NoUsableMessageException` kesalahan. Untuk informasi selengkapnya, lihat Grup Pesan.

Mengatur atribut permintaan yang telah ditentukan di bot Lex V2 Anda

Amazon Lex V2 menyediakan atribut permintaan yang telah ditentukan untuk mengelola cara memproses informasi yang dikirim ke bot Anda. Atribut tidak bertahan untuk seluruh sesi, jadi Anda harus mengirim atribut yang telah ditentukan di setiap permintaan. Semua atribut yang telah ditentukan ada di `x-amz-lex:` namespace.

Menonaktifkan sakelar maksud di bot Lex V2 Anda

Untuk mengontrol apakah pengguna dapat beralih antar intent selama konfirmasi maksud atau elisitasi slot, gunakan atribut request. `x-amz-lex:intent-switch` Saat disetel ke`DISABLE`, atribut ini mencegah pengguna memicu maksud yang berbeda saat mereka berada di tengah menyelesaikan alur intent saat ini.

Misalnya, jika pengguna sedang dalam proses pemesanan penerbangan dan sedang diminta untuk rincian penerbangan, maka ucapan seperti "periksa cuaca" atau "pesan hotel" - yang biasanya dapat memicu maksud lain - akan diabaikan, memastikan percakapan tetap terfokus pada proses pemesanan saat ini.

Mengatur batas waktu sesi

Amazon Lex menyimpan informasi konteks—data slot dan atribut sesi—hingga sesi percakapan berakhir. Untuk mengontrol berapa lama sesi berlangsung untuk bot, atur batas waktu sesi. Secara default, durasi sesi adalah 5 menit, tetapi Anda dapat menentukan durasi antara 0 dan 1.440 menit (24 jam).

Misalnya, Anda membuat `ShoeOrdering` bot yang mendukung maksud seperti `OrderShoes` dan`GetOrderStatus`. Ketika Amazon Lex mendeteksi bahwa maksud pengguna adalah memesan sepatu, ia meminta data slot. Misalnya, ia meminta ukuran sepatu, warna, merek, dll. Jika pengguna menyediakan beberapa data slot tetapi tidak menyelesaikan pembelian sepatu, Amazon Lex mengingat semua data slot dan atribut sesi untuk seluruh sesi. Jika pengguna kembali ke sesi sebelum kedaluwarsa, mereka dapat memberikan data slot yang tersisa, dan menyelesaikan pembelian.

Di konsol Amazon Lex V2, Anda mengatur batas waktu sesi saat membuat bot. Dengan antarmuka baris perintah AWS (AWS CLI) atau API, Anda mengatur batas waktu saat membuat bot dengan CreateBotoperasi dengan menyetel bidang idle. SessionTTLInSeconds

Berbagi informasi antar maksud dengan bot Lex V2 Anda

Amazon Lex V2 mendukung berbagi informasi antar maksud. Untuk berbagi antar maksud, gunakan konteks keluaran atau atribut sesi.

Untuk menggunakan konteks keluaran, Anda menentukan konteks keluaran saat membuat atau memperbarui intent. Ketika intent terpenuhi, respons dari Amazon Lex V2 berisi nilai konteks dan slot dari intent sebagai parameter konteks. Anda dapat menggunakan parameter ini sebagai nilai default dalam maksud berikutnya atau dalam kode aplikasi Anda atau fungsi Lambda.

Untuk menggunakan atribut sesi, Anda mengatur atribut di Lambda atau kode aplikasi Anda. Misalnya, pengguna `ShoeOrdering` bot memulai dengan memesan sepatu. Bot terlibat dalam percakapan dengan pengguna, mengumpulkan data slot, seperti ukuran sepatu, warna, dan merek. Saat pengguna melakukan pemesanan, fungsi Lambda yang memenuhi urutan akan menetapkan atribut `orderNumber` sesi, yang berisi nomor urut. Untuk mendapatkan status pesanan, pengguna menggunakan `GetOrderStatus` intent. Bot dapat meminta pengguna untuk data slot, seperti nomor pesanan dan tanggal pesanan. Ketika bot memiliki informasi yang diperlukan, bot mengembalikan status pesanan.

Jika Anda berpikir bahwa pengguna Anda mungkin beralih maksud selama sesi yang sama, Anda dapat mendesain bot Anda untuk mengembalikan status pesanan terbaru. Alih-alih meminta pengguna untuk informasi pesanan lagi, Anda menggunakan atribut `orderNumber` session untuk berbagi informasi di seluruh intent dan memenuhi `GetOrderStatus` intent. Bot melakukan ini dengan mengembalikan status pesanan terakhir yang ditempatkan pengguna.

Mengatur atribut kompleks di bot Lex V2 Anda

Atribut sesi dan permintaan adalah peta string-ke-string atribut dan nilai. Dalam banyak kasus, Anda dapat menggunakan peta string untuk mentransfer nilai atribut antara aplikasi klien Anda dan bot. Namun, dalam beberapa kasus, Anda mungkin perlu mentransfer data biner atau struktur kompleks yang tidak dapat dengan mudah dikonversi ke peta string. Misalnya, objek JSON berikut mewakili array dari tiga kota terpadat di Amerika Serikat:

```
{
   "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"
         }
      }
   ]
}
```

Array data ini tidak diterjemahkan dengan baik ke peta string-ke-string. Dalam kasus seperti itu, Anda dapat mengubah objek menjadi string sederhana sehingga Anda dapat mengirimkannya ke bot Anda dengan RecognizeUtteranceoperasi RecognizeTextdan.

Misalnya, jika Anda menggunakan JavaScript, Anda dapat menggunakan `JSON.stringify` operasi untuk mengonversi objek ke JSON, dan `JSON.parse` operasi untuk mengonversi teks JSON menjadi JavaScript objek:

```
// 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);
```

Untuk mengirim atribut dengan `RecognizeUtterance` operasi, Anda harus base64 menyandikan atribut sebelum Anda menambahkannya ke header permintaan, seperti yang ditunjukkan dalam kode berikut: JavaScript

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

Anda dapat mengirim data biner ke `RecognizeText` dan `RecognizeUtterance` operasi dengan terlebih dahulu mengonversi data ke string yang dikodekan base64, dan kemudian mengirim string sebagai nilai dalam atribut sesi:

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

Memahami sesi bot Amazon Lex V2

Saat pengguna memulai percakapan dengan bot Anda, Amazon Lex V2 membuat sesi. Informasi yang dipertukarkan antara aplikasi Anda dan Amazon Lex V2 membentuk status sesi untuk percakapan. Saat Anda membuat permintaan, sesi diidentifikasi oleh pengenal yang Anda tentukan. Untuk informasi selengkapnya tentang pengenal sesi, lihat `sessionId` bidang dalam RecognizeUtteranceoperasi RecognizeTextatau.

Anda dapat mengubah status sesi yang dikirim antara aplikasi Anda dan bot Anda. Misalnya, Anda dapat membuat dan memodifikasi atribut sesi yang berisi informasi khusus tentang sesi, dan Anda

dapat mengubah alur percakapan dengan menyetel konteks dialog untuk menafsirkan ucapan berikutnya.

Ada tiga cara agar Anda dapat memperbarui status sesi.

- Lulus informasi sesi sebaris sebagai bagian dari panggilan ke `RecognizeText` atau `RecognizeUtterance` operasi.
- Gunakan fungsi Lambda dengan `RecognizeText` atau `RecognizeUtterance` operasi yang dipanggil setelah setiap putaran percakapan. Untuk informasi selengkapnya, lihat Mengintegrasikan AWS Lambda fungsi ke bot Amazon Lex V2 Anda. Yang lainnya adalah menggunakan API runtime Amazon Lex V2 di aplikasi Anda untuk membuat perubahan pada status sesi.
- Gunakan operasi yang memungkinkan Anda mengelola informasi sesi untuk percakapan dengan bot Anda. Operasi adalah PutSessionoperasi, GetSessionoperasi, dan DeleteSessionoperasi. Anda menggunakan operasi ini untuk mendapatkan informasi tentang status sesi pengguna Anda dengan bot Anda, dan untuk memiliki kontrol halus atas status.

Gunakan `GetSession` operasi saat Anda ingin mendapatkan status sesi saat ini. Operasi mengembalikan status sesi saat ini, termasuk status dialog dengan pengguna Anda, atribut sesi apa pun yang telah ditetapkan, dan nilai slot untuk maksud saat ini, dan maksud lain yang diidentifikasi Amazon Lex V2 sebagai kemungkinan maksud yang cocok dengan ucapan pengguna.

`PutSession`Operasi ini memungkinkan Anda untuk secara langsung memanipulasi status sesi saat ini. Anda dapat mengatur sesi, termasuk jenis tindakan dialog yang akan dilakukan bot selanjutnya dan pesan yang dikirimkan Amazon Lex V2 ke pengguna. Ini memberi Anda kendali atas aliran percakapan dengan bot. Atur `type` bidang tindakan dialog `Delegate` agar Amazon Lex V2 menentukan tindakan selanjutnya untuk bot.

Anda dapat menggunakan `PutSession` operasi untuk membuat sesi baru dengan bot dan mengatur maksud bahwa bot harus memulai. Anda juga dapat menggunakan `PutSession` operasi untuk mengubah dari satu maksud ke maksud lainnya. Saat membuat sesi atau mengubah intent, Anda juga dapat mengatur status sesi, seperti nilai slot dan atribut sesi. Ketika intent baru selesai, Anda memiliki opsi untuk memulai ulang maksud sebelumnya.

Respons dari `PutSession` operasi berisi informasi yang sama dengan `RecognizeUtterance` operasi. Anda dapat menggunakan informasi ini untuk meminta pengguna untuk informasi berikutnya, seperti yang Anda lakukan dengan respons dari `RecognizeUtterance` operasi.

Gunakan `DeleteSession` operasi untuk menghapus sesi yang ada dan mulai lagi dengan sesi baru. Misalnya, ketika Anda menguji bot Anda, Anda dapat menggunakan `DeleteSession` operasi untuk menghapus sesi pengujian dari bot Anda.

Operasi sesi bekerja dengan fungsi Lambda pemenuhan Anda. Misalnya, jika fungsi Lambda Anda kembali `Failed` sebagai status pemenuhan, Anda dapat menggunakan `PutSession` operasi untuk menyetel jenis tindakan dialog ke `close` dan `fulfillmentState` `ReadyForFulfillment` untuk mencoba lagi langkah pemenuhan.

Berikut adalah beberapa hal yang dapat Anda lakukan dengan operasi sesi:

- Mintalah bot memulai percakapan alih-alih menunggu pengguna.
- Beralih maksud selama percakapan.
- Kembali ke intent sebelumnya.
- Mulai atau mulai ulang percakapan di tengah interaksi.
- Validasi nilai slot dan minta bot kembali meminta nilai yang tidak valid.

Masing-masing dijelaskan lebih lanjut di bawah ini.

Memulai sesi baru

Jika Anda ingin bot memulai percakapan dengan pengguna Anda, Anda dapat menggunakan `PutSession` operasi.

- Buat maksud selamat datang tanpa slot dan pesan kesimpulan yang meminta pengguna untuk menyatakan maksud. Misalnya, “Apa yang ingin Anda pesan? Anda bisa mengatakan 'Pesan minuman' atau 'Pesan pizza. '”
- Panggil `PutSession` operasi. Tetapkan nama maksud ke nama maksud selamat datang Anda dan setel tindakan dialog ke. `Delegate`
- Amazon Lex akan merespons dengan permintaan dari niat selamat datang Anda untuk memulai percakapan dengan pengguna Anda.

Beralih maksud

Anda dapat menggunakan `PutSession` operasi untuk beralih dari satu maksud ke maksud lainnya. Anda juga dapat menggunakannya untuk beralih kembali ke intent sebelumnya. Anda dapat menggunakan `PutSession` operasi untuk mengatur atribut sesi atau nilai slot untuk maksud baru.

- Panggil `PutSession` operasi. Tetapkan nama maksud ke nama maksud baru dan setel tindakan dialog ke. `Delegate` Anda juga dapat mengatur nilai slot atau atribut sesi apa pun yang diperlukan untuk maksud baru.
- Amazon Lex akan memulai percakapan dengan pengguna menggunakan maksud baru.

Melanjutkan niat sebelumnya

Untuk melanjutkan intent sebelumnya, Anda menggunakan `GetSession` operasi untuk mendapatkan status intent, lakukan interaksi yang diperlukan, lalu gunakan `PutSession` operasi untuk menyetel intent ke status dialog sebelumnya.

- Panggil `GetSession` operasi. Simpan status niat.
- Lakukan interaksi lain, seperti memenuhi niat yang berbeda.
- Menggunakan informasi yang disimpan informasi untuk maksud sebelumnya, panggil `PutSession` operasi. Ini akan mengembalikan pengguna ke intent sebelumnya di tempat yang sama dalam percakapan.

Dalam beberapa kasus mungkin perlu untuk melanjutkan percakapan pengguna Anda dengan bot Anda. Misalnya, katakan bahwa Anda telah membuat bot layanan pelanggan. Aplikasi Anda menentukan bahwa pengguna perlu berbicara dengan perwakilan layanan pelanggan. Setelah berbicara dengan pengguna, perwakilan dapat mengarahkan percakapan kembali ke bot dengan informasi yang mereka kumpulkan.

Untuk melanjutkan sesi, gunakan langkah-langkah yang mirip dengan ini:

- Aplikasi Anda menentukan bahwa pengguna perlu berbicara dengan perwakilan layanan pelanggan.
- Gunakan `GetSession` operasi untuk mendapatkan status dialog intent saat ini.
- Perwakilan layanan pelanggan berbicara kepada pengguna dan menyelesaikan masalah.
- Gunakan `PutSession` operasi untuk menyetel status dialog maksud. Ini mungkin termasuk menyetel nilai slot, menyetel atribut sesi, atau mengubah maksud.
- Bot melanjutkan percakapan dengan pengguna.

Memvalidasi nilai slot

Anda dapat memvalidasi tanggapan terhadap bot Anda menggunakan aplikasi klien Anda. Jika respons tidak valid, Anda dapat menggunakan `PutSession` operasi untuk mendapatkan respons baru dari pengguna Anda. Misalnya, bot pemesanan bunga Anda hanya dapat menjual tulip, mawar, dan bunga lili. Jika pengguna memesan anyelir, aplikasi Anda dapat melakukan hal berikut:

- Periksa nilai slot yang dikembalikan dari `PostContent` respons `PostText` atau.
- Jika nilai slot tidak valid, hubungi `PutSession` operasi. Aplikasi Anda harus menghapus nilai slot, mengatur `slotToElicit` bidang, dan menetapkan `dialogAction.type` nilainya`elicitSlot`. Secara opsional, Anda dapat mengatur `messageFormat` bidang `message` dan jika Anda ingin mengubah pesan yang digunakan Amazon Lex untuk memperoleh nilai slot.

Mengintegrasikan AWS Lambda fungsi ke bot Amazon Lex V2 Anda

Dengan AWS Lambdafungsi, Anda dapat memperluas dan mengontrol perilaku bot Amazon Lex V2 Anda dengan lebih baik melalui fungsi khusus yang Anda tentukan. Amazon Lex V2 menggunakan satu fungsi Lambda per alias bot per bahasa, bukan satu fungsi Lambda untuk setiap maksud. Sebelum memulai, tentukan bidang mana dalam acara input yang ingin Anda ambil informasi dan bidang mana dalam respons yang ingin Anda manipulasi dan kembalikan dari fungsi Lambda Anda

Untuk mengintegrasikan fungsi Lambda dengan bot Amazon Lex V2 Anda, lakukan langkah-langkah berikut:

1. Buat fungsi dalam AWS Lambda menggunakan bahasa pemrograman pilihan Anda dan tulis skrip Anda.
2. Pastikan bahwa fungsi mengembalikan struktur yang cocok dengan format respons.
3. Menyebarkan fungsi Lambda.
4. Kaitkan fungsi Lambda dengan alias bot Amazon Lex V2 dengan operasi konsol atau API.
5. Pilih tahapan percakapan di mana Anda ingin menjalankan fungsi Lambda Anda dengan konsol atau operasi API.
6. Bangun bot Amazon Lex V2 Anda dan uji apakah fungsi Lambda berfungsi sebagaimana dimaksud. Debug fungsi Anda dengan bantuan Amazon CloudWatch.

Topik

- AWS Lambdaformat acara masukan untuk Lex V2
- AWS Lambdaformat respons untuk Lex V2
- Struktur umum dalam AWS Lambda fungsi untuk Amazon Lex V2
- Membuat AWS Lambda fungsi untuk bot Amazon Lex V2 Anda
- Mendebug fungsi CloudWatch Lambda menggunakan log Log

AWS Lambdaformat acara masukan untuk Lex V2

Langkah pertama dalam mengintegrasikan fungsi Lambda ke dalam bot Amazon Lex V2 Anda adalah memahami bidang dalam acara Amazon Lex V2 dan untuk menentukan informasi dari bidang

ini yang ingin Anda gunakan saat menulis skrip Anda. Objek JSON berikut menunjukkan format umum peristiwa Amazon Lex V2 yang diteruskan ke fungsi Lambda:

> **Note**
>
> Format input dapat berubah tanpa perubahan yang sesuai dengan format`messageVersion`. Kode Anda seharusnya tidak menimbulkan kesalahan jika ada bidang baru.

```
{
    "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 Niat 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 Niat for details about the structure
        },
        "prompt": {
            "attempt": string
        }
    },
    "requestAttributes": {
        string: string,
        ...
    },
    "sessionState": {
        // see Status sesi for details about the structure
    },
    "transcriptions": [
        {
            "transcription": string,
            "transcriptionConfidence": number,
            "resolvedContext": {
                "intent": string
            },
            "resolvedSlots": {
                slot name: {
                    // see Slot for details about the structure
                },
                ...
            }
        },
        ...
    ]
}
```

Setiap bidang dalam acara input dijelaskan di bawah ini:

messageVersion

Versi pesan yang mengidentifikasi format data peristiwa yang masuk ke fungsi Lambda dan format respons yang diharapkan dari fungsi Lambda.

> **Note**
>
> Anda mengonfigurasi nilai ini saat menentukan maksud. Dalam implementasi saat ini, Amazon Lex V2 hanya mendukung pesan versi 1.0. Oleh karena itu, konsol mengasumsikan nilai default 1.0 dan tidak menampilkan versi pesan.

invocationSource

Hook kode yang disebut fungsi Lambda. Nilai-nilai berikut dimungkinkan:

`DialogCodeHook`- Amazon Lex V2 menyebut fungsi Lambda setelah input dari pengguna.

`FulfillmentCodeHook`— Amazon Lex V2 menyebut fungsi Lambda setelah mengisi semua slot yang diperlukan dan maksudnya siap untuk dipenuhi.

InputMode

Mode ucapan pengguna. Kemungkinan nilainya adalah sebagai berikut:

`DTMF`— Pengguna memasukkan ucapan menggunakan keypad touch-tone (Dual Tone Multi-Frequency).

`Speech`— Pengguna mengucapkan ucapannya.

`Text`— Pengguna mengetik ucapan.

responseContentType

Mode respons bot terhadap pengguna. `text/plain; charset=utf-8`menunjukkan bahwa ucapan terakhir ditulis, sedangkan nilai yang dimulai dengan `audio` menunjukkan bahwa ucapan terakhir diucapkan.

sessionId

Pengidentifikasi sesi alfanumerik yang digunakan untuk percakapan.

inputTranscript

Transkripsi input dari pengguna.

- Untuk input teks, ini adalah teks yang diketik pengguna. Untuk input DTMF, ini adalah kunci yang dimasukkan pengguna.
- Untuk input ucapan, ini adalah teks yang digunakan Amazon Lex V2 untuk mengonversi ucapan pengguna untuk memanggil maksud atau mengisi slot.

InvocationLabel

Nilai yang menunjukkan respons yang memanggil fungsi Lambda. Anda dapat mengatur label pemanggilan untuk respons awal, slot, dan respons konfirmasi.

bot

Informasi tentang bot yang memproses permintaan, yang terdiri dari bidang-bidang berikut:

- id — Pengenal yang ditetapkan ke bot saat Anda membuatnya. Anda dapat melihat ID bot di konsol Amazon Lex V2 di halaman Pengaturan bot.
- Nama — Nama yang Anda berikan pada bot saat Anda membuatnya.
- LocaleID — Pengenal lokal yang Anda gunakan untuk bot Anda. Untuk daftar lokal, lihatBahasa dan lokal yang didukung oleh Amazon Lex V2.
- versi — Versi bot yang memproses permintaan.
- AliaSid — Pengidentifikasi yang ditetapkan ke alias bot saat Anda membuatnya. Anda dapat melihat ID alias bot di konsol Amazon Lex V2 di halaman Alias. Jika Anda tidak dapat melihat ID alias dalam daftar, pilih ikon roda gigi di kanan atas dan nyalakan ID Alias.
- AliasName — Nama yang Anda berikan alias bot.

interpretasi

Daftar informasi tentang maksud yang Amazon Lex V2 anggap mungkin cocok dengan ucapan pengguna. Setiap item adalah struktur yang memberikan informasi tentang kecocokan ucapan dengan maksud, dengan format berikut:

```
{
```

```
    "intent": {
        // see Niat 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
        }
    }
}
```

Bidang dalam struktur adalah sebagai berikut:

- intent — Struktur yang berisi informasi tentang maksud. Lihat Niat untuk detail tentang struktur.
- NLUConfidence — Skor yang menunjukkan seberapa yakin Amazon Lex V2 bahwa maksud tersebut cocok dengan maksud pengguna.
- SentimentResponse — Analisis sentimen respons, yang berisi bidang-bidang berikut:
  - sentimen — Menunjukkan apakah sentimen ucapan ituPOSITIVE,,, NEGATIVE atau. NEUTRAL MIXED
  - SentimentScore — Struktur yang memetakan setiap sentimen ke angka yang menunjukkan seberapa yakin Amazon Lex V2 bahwa ucapan tersebut menyampaikan sentimen itu.
- InterpretationSource - Menunjukkan apakah slot diselesaikan oleh Amazon Lex V2 atau Amazon Bedrock.

proposedNextState

Jika fungsi Lambda menyetel `sessionState` to`Delegate`, bidang ini akan muncul dan menampilkan proposal Amazon Lex V2 untuk langkah selanjutnya dalam percakapan. `dialogAction` Jika tidak, status berikutnya tergantung pada pengaturan yang Anda kembalikan dalam respons dari fungsi Lambda Anda. Struktur ini hanya ada jika kedua pernyataan berikut benar:

1. `invocationSource`Nilainya adalah `DialogCodeHook`
2. Yang diprediksi `type` `dialogAction` adalah`ElicitSlot`.

Anda dapat menggunakan informasi ini untuk menambahkan `runtimeHints` pada titik yang tepat dalam percakapan. Lihat Meningkatkan pengenalan nilai slot dengan petunjuk runtime dalam percakapan untuk informasi lebih lanjut. `proposedNextState`adalah struktur yang berisi bidang-bidang berikut:

Struktur `proposedNextState` adalah sebagai berikut:

```
"proposedNextState": {
    "dialogAction": {
        "slotToElicit": string,
        "type": "Close | ConfirmIntent | Delegate | ElicitIntent | ElicitSlot"
    },
    "intent": {
        // see Niat for details about the structure
    },
    "prompt": {
        "attempt": string
    }
}
```

- DialogAction - Berisi informasi tentang langkah selanjutnya yang diusulkan Amazon Lex V2. Bidang dalam struktur adalah sebagai berikut:
  - slotToElicit— Slot untuk memperoleh berikutnya seperti yang diusulkan oleh Amazon Lex V2. Bidang ini hanya muncul jika `type` ada`ElicitSlot`.
  - type — Langkah selanjutnya dalam percakapan seperti yang diusulkan oleh Amazon Lex V2. Nilai-nilai berikut dimungkinkan:

    `Delegate`- Amazon Lex V2 menentukan tindakan selanjutnya.

    `ElicitIntent`— Tindakan selanjutnya adalah untuk mendapatkan maksud dari pengguna.

    `ElicitSlot`— Tindakan selanjutnya adalah mendapatkan nilai slot dari pengguna.

    `Close`— Mengakhiri proses pemenuhan maksud dan menunjukkan bahwa tidak akan ada respons dari pengguna.

    `ConfirmIntent`— Tindakan selanjutnya adalah bertanya kepada pengguna apakah slotnya benar dan niat siap untuk dipenuhi.
- maksud — Maksud bahwa bot telah menentukan bahwa pengguna mencoba untuk memenuhi. Lihat Niat untuk detail tentang struktur.

- prompt — Struktur yang berisi bidang`attempt`, yang memetakan ke nilai yang menentukan berapa kali Amazon Lex V2 telah mendorong pengguna untuk slot berikutnya. Nilai yang mungkin adalah `Initial` untuk upaya pertama dan`Retry1`,`Retry2`,`Retry3`,`Retry4`, dan `Retry5` untuk upaya selanjutnya.

requestAttributes

Struktur yang berisi atribut permintaan khusus yang dikirim klien dalam permintaan. Gunakan atribut permintaan untuk meneruskan informasi yang tidak perlu bertahan selama seluruh sesi. Jika tidak ada atribut permintaan, nilainya akan menjadi null. Untuk informasi selengkapnya, lihat Mengatur atribut permintaan untuk bot Lex V2 Anda.

SessionState

Keadaan percakapan saat ini antara pengguna dan bot Amazon Lex V2 Anda. Lihat Status sesi untuk detail tentang struktur.

transkripsi

Daftar transkripsi yang Amazon Lex V2 anggap mungkin cocok dengan ucapan pengguna. Untuk informasi selengkapnya, lihat Menggunakan skor kepercayaan transkripsi suara untuk meningkatkan percakapan dengan bot Lex V2 Anda. Setiap item adalah objek dengan format berikut, berisi informasi tentang satu kemungkinan transkripsi:

```
{
    "transcription": string,
    "transcriptionConfidence": number,
    "resolvedContext": {
        "intent": string
    },
    "resolvedSlots": {
        slot name: {
            // see Slot for details about the structure
        },
        ...
    }
}
```

Bidang dijelaskan di bawah ini:

- transkripsi — Transkripsi yang Amazon Lex V2 anggap mungkin cocok dengan ucapan audio pengguna.
- TranscriptionConfidence — Skor yang menunjukkan seberapa yakin Amazon Lex V2 bahwa intent cocok dengan maksud pengguna.
- ResolvedContext - Struktur yang berisi bidang`intent`, yang memetakan ke maksud yang berkaitan dengan ucapan tersebut.
- ResolvedSlots — Struktur yang kuncinya adalah nama setiap slot yang diselesaikan dengan ucapan. Setiap nama slot memetakan ke struktur yang berisi informasi tentang slot itu. Lihat Slot untuk detail tentang struktur.

AWS Lambdaformat respons untuk Lex V2

Langkah kedua dalam mengintegrasikan fungsi Lambda ke bot Amazon Lex V2 Anda adalah memahami bidang dalam respons fungsi Lambda dan menentukan parameter mana yang ingin Anda manipulasi. Objek JSON berikut menunjukkan format umum respons Lambda yang dikembalikan ke Amazon Lex V2:

```
{
    "sessionState": {
        // see Status sesi 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,
        ...
    }
}
```

Setiap bidang dalam tanggapan dijelaskan di bawah ini:

SessionState

Keadaan percakapan antara pengguna dan bot Amazon Lex V2 Anda yang ingin Anda kembalikan. Lihat Status sesi untuk detail tentang struktur. Bidang ini selalu diperlukan.

pesan

Daftar pesan yang dikembalikan Amazon Lex V2 ke pelanggan untuk giliran percakapan berikutnya. Jika yang `contentType` Anda berikan adalah `PlainTextCustomPayload`,, atau`SSML`, tulis pesan yang ingin Anda kembalikan ke pelanggan di `content` lapangan. Jika yang `contentType` Anda berikan adalah`ImageResponseCard`, berikan detail kartu di `imageResponseCard` lapangan. Jika Anda tidak menyediakan pesan, Amazon Lex V2 menggunakan pesan yang sesuai yang ditentukan saat bot dibuat.

`messages`Bidang diperlukan jika `dialogAction.type` adalah `ElicitIntent` atau`ConfirmIntent`.

Setiap item dalam daftar adalah struktur dalam format berikut, berisi informasi tentang pesan untuk kembali ke pengguna. Inilah contohnya:

```
{
    "contentType": "CustomPayload | ImageResponseCard | PlainText | SSML",
    "content": string,
    "imageResponseCard": {
        "title": string,
        "subtitle": string,
        "imageUrl": string,
        "buttons": [
            {
                "text": string,
                "value": string
            },
```

```
            ...
        ]
    }
}
```

Deskripsi untuk setiap bidang disediakan di bawah ini:

- ContentType — Jenis pesan yang akan digunakan.

  `CustomPayload`— String respons yang dapat Anda sesuaikan untuk menyertakan data atau metadata untuk aplikasi Anda.

  `ImageResponseCard`— Gambar dengan tombol yang dapat dipilih pelanggan. Untuk informasi selengkapnya, lihat ImageResponseCard.

  `PlainText`— String teks biasa.

  `SSML`— String yang menyertakan Speech Synthesis Markup Language untuk menyesuaikan respons audio.
- konten — Pesan untuk dikirim ke pengguna. Gunakan bidang ini jika jenis pesan adalah`PlainText`,`CustomPayload`, atau`SSML`.
- imageResponseCard— Berisi definisi kartu respons untuk ditampilkan kepada pengguna. Gunakan bidang ini jika jenis pesannya`ImageResponseCard`. Peta ke struktur yang berisi bidang-bidang berikut:
  - judul — Judul kartu respons.
  - subtitle — Permintaan bagi pengguna untuk memilih tombol.
  - ImageUrl — Tautan ke gambar untuk kartu.
  - tombol — Daftar struktur yang berisi informasi tentang tombol. Setiap struktur berisi `text` bidang dengan teks yang akan ditampilkan dan `value` bidang dengan nilai untuk dikirim ke Amazon Lex V2 jika pelanggan memilih tombol itu. Anda dapat menyertakan hingga tiga tombol.

requestAttributes

Struktur yang berisi atribut permintaan khusus untuk respons terhadap pelanggan. Untuk informasi selengkapnya, lihat Mengatur atribut permintaan untuk bot Lex V2 Anda. Bidang ini bersifat opsional.

Bidang yang diperlukan dalam respons

Minimal, respons Lambda harus menyertakan `sessionState` objek. Di dalamnya, berikan `dialogAction` objek dan tentukan `type` bidangnya. Bergantung pada `type` `dialogAction` yang Anda berikan, mungkin ada bidang lain yang diperlukan untuk respons Lambda. Persyaratan ini dijelaskan sebagai berikut, di samping contoh kerja minimal:

Mendelegasikan

Delegasi memungkinkan Amazon Lex V2 menentukan langkah selanjutnya. Tidak ada bidang lain yang diperlukan.

```
{
    "sessionState": {
        "dialogAction": {
            "type": "Delegate"
    }
}
```

ElicitIntent

ElicitIntentmeminta pelanggan untuk menyatakan niat. Anda harus menyertakan setidaknya satu pesan di `messages` bidang untuk meminta pemunculan maksud.

```
{
    "sessionState": {
        "dialogAction": {
            "type": "ElicitIntent"
    },
    "messages": [
        {
            "contentType": PlainText,
            "content": "How can I help you?"
        }
    ]
}
```

ElicitSlot

ElicitSlotmeminta pelanggan untuk memberikan nilai slot. Anda harus memasukkan nama slot di `slotToElicit` bidang di `dialogAction` objek. Anda juga harus memasukkan `sessionState` objek `intent` dalam. `name`

```
{`
    "sessionState": {
        "dialogAction": {
            "slotToElicit": "OriginCity",
            "type": "ElicitSlot"
        },
        "intent": {
            "name": "BookFlight"
        }
    }
}
```

ConfirmIntent

ConfirmIntentmengkonfirmasikan nilai slot pelanggan dan apakah maksudnya siap dipenuhi. Anda harus memasukkan `sessionState` objek `intent` dalam dan yang `slots` akan dikonfirmasi. `name` Anda juga harus menyertakan setidaknya satu pesan di `messages` bidang untuk meminta pengguna konfirmasi nilai slot. Pesan Anda harus meminta respons “ya” atau “tidak”. Jika pengguna menjawab “ya”, Amazon Lex V2 menyetel intent ke. `confirmationState` `Confirmed` Jika pengguna merespons “tidak”, Amazon Lex V2 menyetel intent ke. `confirmationState` `Denied`

```
{
    "sessionState": {
        "dialogAction": {
            "type": "ConfirmIntent"
        },
        "intent": {
            "name": "BookFlight",
            "slots": {
                "DepartureDate": {
                    "value": {
                        "originalValue": "tomorrow",
                        "interpretedValue": "2023-05-09",
                        "resolvedValues": [
                            "2023-05-09"
                    ]
                 }
                },
                "DestinationCity": {
                    "value": {
                        "originalValue": "sf",
                        "interpretedValue": "sf",
```

```
                        "resolvedValues": [
                            "sf"
                        ]
                    }
                },
                "OriginCity": {
                    "value": {
                        "originalValue": "nyc",
                        "interpretedValue": "nyc",
                        "resolvedValues": [
                            "nyc"
                        ]
                    }
                }
            }
        }
    },
    "messages": [
        {
            "contentType": PlainText,
            "content": "Okay, you want to fly from {OriginCity} to \
            {DestinationCity} on {DepartureDate}. Is that correct?"
        }
    ]
}
```

Tutup

Tutup mengakhiri proses pemenuhan maksud dan menunjukkan bahwa tidak ada tanggapan lebih lanjut yang diharapkan dari pengguna. Anda harus memasukkan `name` dan `state` dari `intent` dalam `sessionState` objek. Status maksud yang kompatibel adalah`Failed`,`Fulfilled`, dan`InProgress`.

```
"sessionState": {
    "dialogAction": {
        "type": "Close"
    },
    "intent": {
        "name": "BookFlight",
        "state": "Failed | Fulfilled | InProgress"
    }
}
```

Struktur umum dalam AWS Lambda fungsi untuk Amazon Lex V2

Dalam respons Lambda, ada sejumlah struktur yang berulang. Rincian tentang struktur umum ini disediakan di bagian ini.

Niat

```
"intent": {
    "confirmationState": "Confirmed | Denied | None",
    "name": string,
    "slots": {
        // see Slot 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
       }
}
```

`intent`Bidang dipetakan ke objek dengan bidang berikut:

ConfirmationState

Menunjukkan apakah pengguna telah mengonfirmasi slot untuk maksud tersebut dan maksud siap untuk dipenuhi. Nilai-nilai berikut dimungkinkan:

`Confirmed`— Pengguna mengonfirmasi bahwa nilai slot sudah benar.

`Denied`— Pengguna menunjukkan bahwa nilai slot salah.

`None`— Pengguna belum mencapai tahap konfirmasi.

name

Nama niat.

slot

Informasi tentang slot yang diperlukan untuk memenuhi maksud. Lihat Slot untuk detail tentang struktur.

status

Menunjukkan status pemenuhan untuk maksud tersebut. Nilai-nilai berikut dimungkinkan:

`Failed`— Bot gagal memenuhi niat.

`Fulfilled`— Bot telah menyelesaikan pemenuhan niat.

`FulfillmentInProgress`— Bot berada di tengah memenuhi niat.

`InProgress`— Bot berada di tengah memunculkan nilai slot yang diperlukan untuk memenuhi maksud.

`ReadyForFulfillment`— Bot telah memunculkan semua nilai slot untuk maksud tersebut dan siap untuk memenuhi maksud tersebut.

`Waiting`— Bot sedang menunggu respons dari pengguna (terbatas pada percakapan streaming).

kendraResponse

Berisi informasi tentang hasil permintaan pencarian Kendra. Bidang ini hanya muncul jika maksudnya adalah a`KendraSearchIntent`. Lihat sintaks respons dalam panggilan Query API untuk Kendra untuk informasi selengkapnya.

Slot

`slots`Bidang ada dalam `intent` struktur dan dipetakan ke struktur yang kuncinya adalah nama slot untuk maksud itu. Jika slot bukan slot multi-nilai (lihat Menggunakan beberapa nilai dalam slot untuk lebih jelasnya), itu dipetakan ke struktur dengan format berikut. Perhatikan bahwa `shape` adalah`Scalar`.

```
{
    slot name: {
        "shape": "Scalar",
        "value": {
            "originalValue": string,
            "interpretedValue": string,
            "resolvedValues": [
                string,
                ...
            ]
        }
    }
```

```
}
```

Jika slot adalah slot multi-nilai, objek yang dipetakan berisi bidang lain yang disebut`values`, yang dipetakan ke daftar struktur, masing-masing berisi informasi tentang slot yang membentuk slot multi-nilai. Format setiap objek dalam daftar cocok dengan objek yang slot reguler dipetakan. Perhatikan bahwa `shape` adalah`List`, tetapi slot `shape` komponen di bawah `values` adalah`Scalar`.

```
{
    slot name: {
    "shape": "List",
    "value": {
        "originalValue": string,
        "interpretedValue": string,
        "resolvedValues": [
            string,
            ...
        ]
    },
    "values": [
        {
            "shape": "Scalar",
            "value": {
                "originalValue": string,
                "interpretedValue": string,
                "resolvedValues": [
                    string,
                    ...
                ]
            }
        },
        {
            "shape": "Scalar",
            "value": {
                "originalValue": string,
                "interpretedValue": string,
                "resolvedValues": [
                    string,
                    ...
                ]
            }
        },
        ...
    ]
```

```
}
```

Bidang dalam objek slot dijelaskan di bawah ini:

bentuk

Bentuk slotnya. Nilai ini adalah `List` jika ada beberapa nilai dalam slot (lihat Menggunakan beberapa nilai dalam slot untuk lebih jelasnya) dan `Scalar` sebaliknya.

nilai

Objek yang berisi informasi tentang nilai yang diberikan pengguna untuk slot dan interpretasi Amazon Lex V2, dalam format berikut:

```
{
    "originalValue": string,
    "interpretedValue": string,
    "resolvedValues": [
        string,
        ...
    ]
}
```

Bidang dijelaskan di bawah ini:

- OriginalValue — Bagian dari respons pengguna terhadap elisitasi slot yang ditentukan Amazon Lex V2 relevan dengan nilai slot.
- InterpretedValue — Nilai yang ditentukan Amazon Lex V2 untuk slot, mengingat input pengguna.
- ResolvedValues — Daftar nilai yang ditentukan Amazon Lex V2 adalah kemungkinan resolusi untuk input pengguna.

values

Daftar objek yang berisi informasi tentang slot yang membentuk slot multi-nilai. Format setiap objek cocok dengan slot normal, dengan `shape` dan `value` bidang yang dijelaskan di atas. `values`hanya muncul jika slot terdiri dari beberapa nilai (lihat Menggunakan beberapa nilai dalam slot untuk lebih jelasnya). Objek JSON berikut menunjukkan dua slot komponen:

```
"values": [
    {
        "shape": "Scalar",
```

```
        "value": {
            "originalValue": string,
            "interpretedValue": string,
            "resolvedValues": [
                string,
                ...
            ]
        }
    },
    {
        "shape": "Scalar",
        "value": {
            "originalValue": string,
            "interpretedValue": string,
            "resolvedValues": [
                string,
                ...
            ]
        }
    },
    ...
]
```

Status sesi

`sessionState`Bidang dipetakan ke objek yang berisi informasi tentang keadaan percakapan dengan pengguna. Bidang aktual yang muncul di objek bergantung pada jenis tindakan dialog. Lihat Bidang yang diperlukan dalam respons kolom wajib dalam respons Lambda. Format `sessionState` objek adalah sebagai berikut:

```
"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 Niat for details about the structure
    },
    "originatingRequestId": string
}
```

Bidang dijelaskan di bawah ini:

ActiveContext

Daftar objek yang berisi informasi tentang konteks yang digunakan pengguna dalam sesi. Gunakan konteks untuk memfasilitasi dan mengontrol pengenalan niat. Untuk informasi lebih lanjut tentang konteks, lihat. Mengatur konteks maksud untuk bot Lex V2 Anda Setiap objek diformat sebagai berikut:

```
{
    "name": string,
```

```
    "contextAttributes": {
        string: string
    },
    "timeToLive": {
        "timeToLiveInSeconds": number,
        "turnsToLive": number
    }
}
```

Bidang dijelaskan di bawah ini:

- nama — Nama konteksnya.
- ContextAttributes — Objek yang berisi nama-nama atribut untuk konteks dan nilai-nilai yang dipetakan.
- timeToLiveObjek yang menentukan berapa lama konteksnya tetap aktif. Objek ini dapat berisi salah satu atau kedua bidang berikut:
  - timeToLiveInSeconds— Jumlah detik yang konteksnya tetap aktif.
  - turnsToLive— Jumlah putaran yang konteksnya tetap aktif.

sessionAttributes

Peta key/value pasangan yang mewakili informasi konteks khusus sesi. Untuk informasi selengkapnya, lihat Mengatur atribut sesi untuk bot Lex V2 Anda. Objek diformat sebagai berikut:

```
{
    string: string,
    ...
}
```

RuntimeHints

Memberikan petunjuk untuk frasa yang akan digunakan pelanggan untuk slot untuk meningkatkan pengenalan audio. Nilai-nilai yang Anda berikan dalam petunjuk meningkatkan pengenalan audio dari nilai-nilai tersebut di atas kata-kata yang terdengar serupa. Format `runtimeHints` objek adalah sebagai berikut:

```
{
    "slotHints": {
        intent name: {
```

```
            slot name: {
                "runtimeHintValues": [
                    {
                        "phrase": string
                    },
                    ...
                ]
            },
            ...
        },
        ...
    }
}
```

`slotHints`Bidang memetakan ke objek yang bidangnya adalah nama maksud di bot. Setiap nama maksud memetakan ke objek yang bidangnya adalah nama slot untuk maksud itu. Setiap nama slot memetakan ke struktur dengan satu bidang`runtimeHintValues`, yang merupakan daftar objek. Setiap objek berisi `phrase` bidang yang memetakan ke petunjuk.

dialogAction

Menentukan tindakan selanjutnya untuk Amazon Lex V2 untuk mengambil. Format objek adalah sebagai berikut:

```
{
    "slotElicitationStyle": "Default | SpellByLetter | SpellByWord",
    "slotToElicit": string,
    "type": "Close | ConfirmIntent | Delegate | ElicitIntent | ElicitSlot"
}
```

Bidang dijelaskan di bawah ini:

- slotElicitationStyle— Menentukan bagaimana Amazon Lex V2 menafsirkan input audio dari pengguna jika `type` dari `dialogAction` adalah`ElicitSlot`. Untuk informasi selengkapnya, lihat Menangkap nilai slot dengan gaya ejaan selama percakapan. Nilai-nilai berikut dimungkinkan:

  `Default`— Amazon Lex V2 menafsirkan input audio secara default untuk memenuhi slot.

  `SpellByLetter`— Amazon Lex V2 mendengarkan ejaan pengguna dari nilai slot.

  `SpellByWord`— Amazon Lex V2 mendengarkan ejaan pengguna dari nilai slot menggunakan kata-kata yang terkait dengan setiap huruf (misalnya, “a as in apple”).

- slotToElicit— Mendefinisikan slot yang akan diperoleh dari pengguna jika dari adalah`type`. `dialogAction` `ElicitSlot`
- type — Mendefinisikan tindakan yang harus dijalankan bot. Nilai-nilai berikut dimungkinkan:

  `Delegate`— Memungkinkan Amazon Lex V2 menentukan langkah selanjutnya.

  `ElicitIntent`— Meminta pelanggan untuk menyatakan niat.

  `ConfirmIntent`— Mengonfirmasi nilai slot pelanggan dan apakah niat siap untuk dipenuhi.

  `ElicitSlot`— Meminta pelanggan untuk memberikan nilai slot untuk suatu maksud.

  `Close`Mengakhiri proses pemenuhan niat.

maksud

Lihat Niat untuk struktur `intent` lapangan.

originatingRequestId

Pengidentifikasi unik untuk permintaan tersebut. Bidang ini opsional untuk respons Lambda.

Membuat AWS Lambda fungsi untuk bot Amazon Lex V2 Anda

Untuk membuat fungsi Lambda untuk bot Amazon Lex V2 Anda, akses AWS Lambda dari AWS Management Console dan buat fungsi baru. Anda dapat merujuk ke panduan AWS Lambda pengembang untuk detail selengkapnyaAWS Lambda.

1. Masuk ke AWS Management Console dan buka AWS Lambda konsol di https://console.aws.amazon.com/lambda/.
2. Pilih Fungsi di sidebar kiri.
3. Pilih Buat fungsi.
4. Anda dapat memilih Author dari awal untuk memulai dengan kode minimal, Gunakan cetak biru untuk memilih kode sampel untuk kasus penggunaan umum dari daftar, atau Gambar kontainer untuk memilih gambar kontainer yang akan digunakan untuk fungsi Anda. Jika Anda memilih Penulis dari awal, lanjutkan dengan langkah-langkah berikut:

   a. Berikan fungsi Anda nama Fungsi yang bermakna untuk menggambarkan apa yang dilakukannya.

b. Pilih bahasa dari menu tarik-turun di bawah Runtime untuk menulis fungsi Anda.

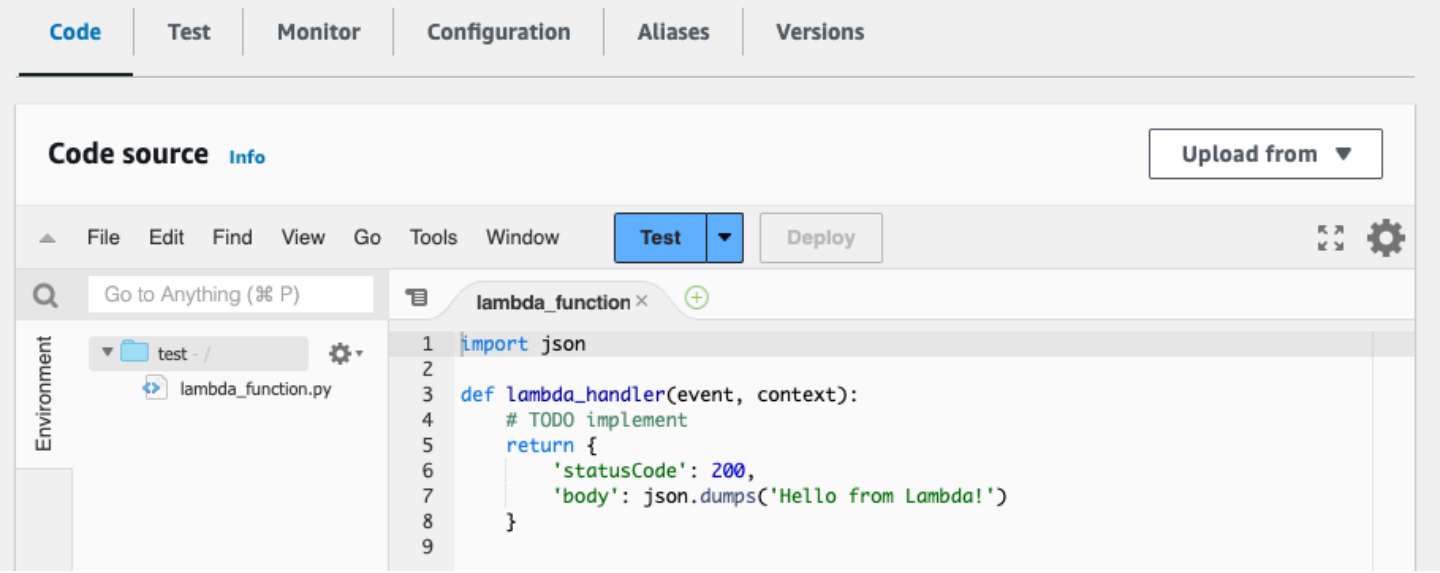
c. Pilih set instruksi Arsitektur untuk fungsi Anda.

d. Secara default, Lambda membuat peran dengan izin dasar. Untuk menggunakan peran yang ada atau membuat peran menggunakan templat kebijakan AWS, perluas menu Ubah peran eksekusi default dan pilih opsi.

e. Perluas menu Pengaturan lanjutan untuk mengonfigurasi lebih banyak opsi.

5. Pilih Buat fungsi.

Gambar berikut menunjukkan apa yang Anda lihat ketika Anda membuat fungsi baru dari awal:



Fungsi handler Lambda berbeda tergantung pada bahasa yang Anda gunakan. Minimal mengambil objek `event` JSON sebagai argumen. Anda dapat melihat bidang di Amazon Lex V2 `event` yang disediakan diAWS Lambdaformat acara masukan untuk Lex V2. Ubah fungsi handler untuk akhirnya mengembalikan objek `response` JSON yang cocok dengan format yang dijelaskan dalam. AWS Lambdaformat respons untuk Lex V2

- Setelah Anda selesai menulis fungsi Anda, pilih Deploy untuk mengizinkan fungsi yang akan digunakan.

Ingatlah bahwa Anda dapat mengaitkan setiap alias bot dengan paling banyak satu fungsi Lambda. Namun, Anda dapat menentukan fungsi sebanyak yang Anda butuhkan untuk bot Anda dalam kode Lambda dan memanggil fungsi-fungsi ini dalam fungsi penangan Lambda. Misalnya, sementara semua intent dalam alias bot yang sama harus memanggil fungsi Lambda yang sama, Anda dapat

membuat fungsi router yang mengaktifkan fungsi terpisah untuk setiap intent. Berikut ini adalah contoh fungsi router yang dapat Anda gunakan atau modifikasi untuk aplikasi Anda:

```
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
```

Kapan menggunakan AWS Lambda fungsi dalam percakapan bot Amazon Lex V2

Anda dapat menggunakan fungsi Lambda pada titik-titik berikut dalam percakapan dengan pengguna:

- Dalam respons awal setelah maksud dikenali. Misalnya, setelah pengguna mengatakan mereka ingin memesan pizza.
- Setelah mendapatkan nilai slot dari pengguna. Misalnya, setelah pengguna memberi tahu bot ukuran pizza yang ingin mereka pesan.
- Antara setiap percobaan lagi untuk mendapatkan slot. Misalnya, jika pelanggan tidak menggunakan ukuran pizza yang diakui.
- Saat mengonfirmasi niat. Misalnya, saat mengonfirmasi pesanan pizza.
- Untuk memenuhi niat. Misalnya, untuk memesan pizza.

- Setelah pemenuhan niat, dan sebelum bot Anda menutup percakapan. Misalnya, untuk beralih ke niat memesan minuman.

Topik

- Lampirkan AWS Lambda fungsi ke bot Amazon Lex V2 menggunakan konsol
- Lampirkan AWS Lambda fungsi ke bot Amazon Lex V2 menggunakan operasi API

Lampirkan AWS Lambda fungsi ke bot Amazon Lex V2 menggunakan konsol

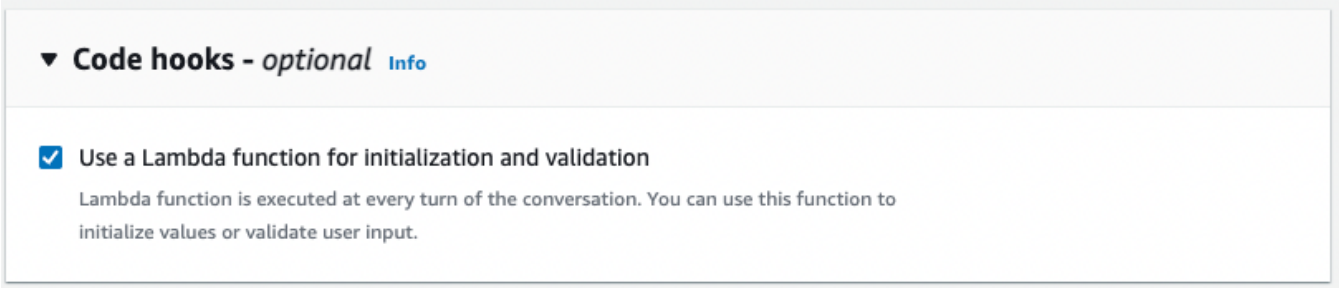
Anda harus terlebih dahulu melampirkan fungsi Lambda ke alias bot Amazon Lex V2 Anda sebelum Anda dapat memanggilnya. Anda hanya dapat melampirkan satu fungsi Lambda dengan setiap alias bot. Lakukan langkah-langkah ini untuk melampirkan fungsi Lambda menggunakan konsol AWS.

1. Masuk ke AWS Management Console dan buka konsol Amazon Lex di https://console.aws.amazon.com/lex/.
2. Pilih Bot dari panel sisi kiri dan dari daftar bot, pilih nama bot yang ingin Anda lampirkan fungsi Lambda.
3. Dari panel sisi kiri, pilih Alias di bawah menu Deployment.
4. Dari daftar alias, pilih nama alias yang ingin Anda lampirkan fungsi Lambda.
5. Di panel Bahasa, pilih bahasa yang Anda inginkan untuk fungsi Lambda. Pilih Kelola bahasa di alias untuk menambahkan bahasa jika tidak ada di panel.
6. Di menu tarik-turun Sumber, pilih nama fungsi Lambda yang ingin Anda lampirkan.
7. Di versi fungsi Lambda atau menu dropdown alias, pilih versi atau alias fungsi Lambda yang ingin Anda gunakan. Kemudian pilih Simpan. Fungsi Lambda yang sama digunakan untuk semua maksud dalam bahasa yang didukung oleh bot.

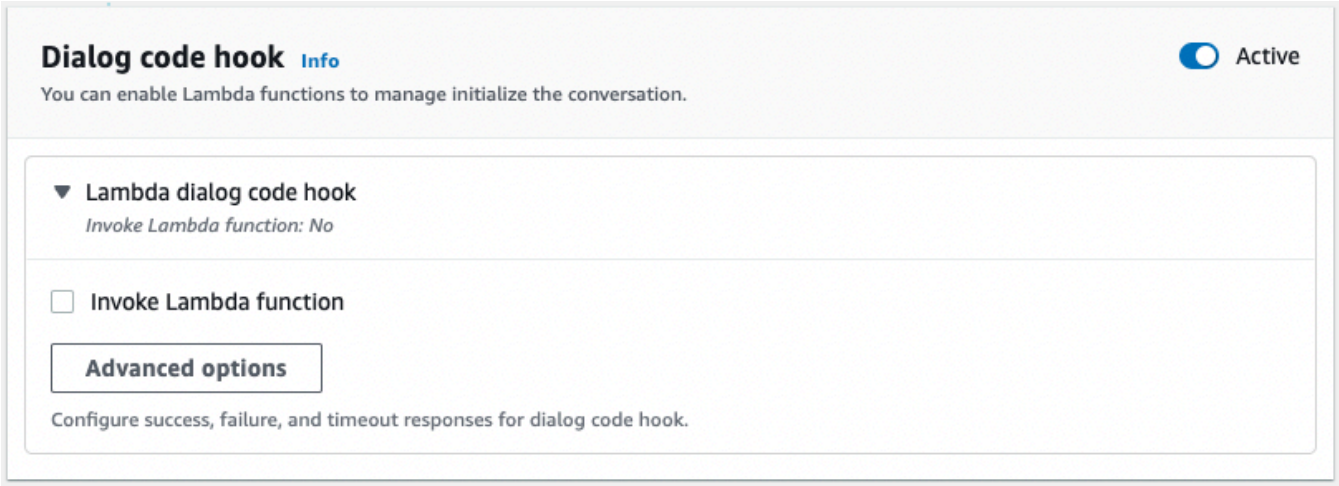
Menyetel intent Amazon Lex V2 untuk menjalankan fungsi Lambda menggunakan konsol

1. Setelah memilih bot, pilih Maksud di menu sebelah kiri di bawah bahasa bot yang ingin Anda gunakan untuk menjalankan fungsi Lambda.
2. Pilih maksud di mana Anda ingin menjalankan fungsi Lambda untuk membuka editor maksud.
3. Ada dua opsi untuk mengatur hook kode Lambda:

1. Untuk menjalankan fungsi Lambda setelah setiap langkah percakapan, gulir ke bagian Kait kode di bagian bawah editor maksud dan pilih kotak centang Gunakan fungsi Lambda untuk inisialisasi dan validasi, seperti pada gambar berikut:



2. Atau, gunakan bagian hook kode dialog di tahapan percakapan untuk menjalankan fungsi Lambda. Bagian hook kode dialog muncul sebagai berikut:



   Ada dua cara untuk mengontrol bagaimana Amazon Lex V2 memanggil hook kode untuk respons:

   - Alihkan tombol Aktif untuk menandainya sebagai aktif atau tidak aktif. Ketika hook kode aktif, Amazon Lex V2 akan memanggil hook kode. Ketika hook kode tidak aktif, Amazon Lex V2 tidak menjalankan hook kode.
   - Perluas bagian kait kode dialog Lambda dan pilih kotak centang fungsi Invoke Lambda untuk menandainya sebagai diaktifkan atau dinonaktifkan. Anda hanya dapat mengaktifkan atau menonaktifkan hook kode ketika ditandai aktif. Ketika ditandai diaktifkan, hook kode dijalankan secara normal. Ketika dinonaktifkan, hook kode tidak dipanggil dan Amazon Lex V2 bertindak seolah-olah hook kode berhasil dikembalikan. Untuk mengonfigurasi respons setelah hook kode dialog berhasil, gagal, atau habis waktu, pilih Opsi lanjutan

   Kait kode Lambda dapat dipanggil pada tahap percakapan berikut:

- Untuk menjalankan fungsi sebagai respons awal, gulir ke bagian Respons Awal, perluas panah di sebelah Respons untuk mengakui permintaan pengguna, dan pilih Opsi lanjutan. Temukan bagian kait kode dialog di bagian bawah menu yang muncul.
- Untuk menjalankan fungsi setelah elicitation slot, gulir ke bagian Slots, perluas panah di sebelah Prompt for slot yang relevan, dan pilih Opsi lanjutan. Temukan bagian kait kode dialog di dekat bagian bawah menu yang muncul, tepat di atas nilai Default.

  Anda juga dapat memanggil fungsi setelah setiap elisitasi. Untuk melakukan ini, perluas Bot memperoleh informasi di bagian Petunjuk Slot, pilih Opsi prompt lainnya, dan pilih kotak centang di samping Memanggil kait kode Lambda setelah setiap elicitation.
- Untuk menjalankan fungsi konfirmasi maksud, gulir ke bagian Konfirmasi, perluas panah di sebelah Prompts untuk mengonfirmasi maksud, dan pilih Opsi lanjutan. Temukan bagian kait kode dialog di bagian bawah menu yang muncul.
- Untuk memanggil fungsi untuk pemenuhan maksud, gulir ke bagian Pemenuhan. Alihkan tombol Aktif untuk mengatur hook kode menjadi aktif. Perluas panah di sebelah On successful fulfillment, dan pilih Advanced options. Pilih kotak centang di samping Gunakan fungsi Lambda untuk pemenuhan di bawah bagian kait kode Lambda Pemenuhan untuk mengatur kait kode ke diaktifkan.

4. Setelah Anda mengatur tahapan percakapan untuk menjalankan fungsi Lambda, Bangun bot lagi untuk menguji fungsinya.

Lampirkan AWS Lambda fungsi ke bot Amazon Lex V2 menggunakan operasi API

Anda harus terlebih dahulu melampirkan fungsi Lambda ke alias bot Amazon Lex V2 Anda sebelum Anda dapat memanggilnya. Anda hanya dapat mengaitkan satu fungsi Lambda dengan setiap alias bot. Lakukan langkah-langkah ini untuk melampirkan fungsi Lambda menggunakan operasi API.

Jika Anda membuat alias bot baru, gunakan CreateBotAliasoperasi untuk melampirkan fungsi Lambda. Untuk melampirkan fungsi Lambda ke alias bot yang ada, gunakan operasi. UpdateBotAlias Ubah `botAliasLocaleSettings` bidang untuk memuat pengaturan yang benar:

```
{
    "botAliasLocaleSettings" : {
        locale: {
            "codeHookSpecification": {
                "lambdaCodeHook": {
```

```
                    "codeHookInterfaceVersion": "1.0",
                    "lambdaARN": "arn:aws:lambda:region:account-id:function:function-
name"
                }
            },
            "enabled": true
        },
        ...
    }
}
```

1. `botAliasLocaleSettings`Bidang memetakan ke objek yang kuncinya adalah lokal di mana Anda ingin melampirkan fungsi Lambda. Lihat Bahasa dan lokal yang didukung daftar lokal yang didukung dan kode yang merupakan kunci yang valid.
2. Untuk menemukan fungsi `lambdaARN` for Lambda, buka AWS Lambda konsol di https://console.aws.amazon.com/lambda/rumah, pilih Fungsi di bilah sisi kiri, dan pilih fungsi yang akan dikaitkan dengan alias bot. Di sisi kanan ikhtisar Fungsi, temukan di `lambdaARN` bawah Fungsi ARN. Ini harus berisi wilayah, ID akun, dan nama fungsi.
3. Untuk mengizinkan Amazon Lex V2 memanggil fungsi Lambda untuk alias, setel `enabled` bidang ke. `true`

Menyetel intent Amazon Lex V2 untuk menjalankan fungsi Lambda menggunakan operasi API

Untuk mengatur pemanggilan fungsi Lambda selama intent, gunakan CreateIntentoperasi jika Anda membuat intent baru, atau UpdateIntentoperasi jika Anda menjalankan fungsi dalam intent yang ada. Bidang yang mengontrol pemanggilan fungsi Lambda dalam operasi intent adalah`dialogCodeHook`,, dan. `initialResponseSetting` `intentConfirmationSetting` `fulfillmentCodeHook`

Jika Anda menjalankan fungsi selama elisitasi slot, gunakan CreateSlotoperasi jika Anda membuat slot baru, atau UpdateSlotoperasi untuk memanggil fungsi di slot yang ada. Bidang yang mengontrol pemanggilan fungsi Lambda dalam operasi slot adalah objek. `slotCaptureSetting` `valueElicitationSetting`

1. Untuk mengatur hook kode dialog Lambda agar berjalan setelah setiap putaran percakapan, setel `enabled` bidang DialogCodeHookSettingsobjek berikut di `dialogCodeHook` bidang menjadi: `true`

   ```
   "dialogCodeHook": {
   ```

```
    "enabled": boolean
}
```

2. Atau, Anda dapat mengatur hook kode dialog Lambda agar berjalan hanya pada titik-titik tertentu dalam percakapan dengan memodifikasi `codeHook` and/or `elicitationCodeHook` bidang dalam struktur yang sesuai dengan tahapan percakapan di mana Anda ingin menjalankan fungsi tersebut. Untuk menggunakan hook kode dialog Lambda untuk pemenuhan maksud, gunakan `fulfillmentCodeHook` bidang dalam operasi atau. CreateIntentUpdateIntent Struktur dan penggunaan ketiga jenis kait kode ini adalah sebagai berikut:

CodeHook

`codeHook`Bidang mendefinisikan pengaturan untuk hook kode untuk dijalankan pada tahap tertentu dalam percakapan. Ini adalah DialogCodeHookInvocationSettingobjek dengan struktur berikut:

```
"codeHook": {
    "active": boolean,
    "enableCodeHookInvocation": boolean,
    "invocationLabel": string,
    "postCodeHookSpecification": PostDialogCodeHookInvocationSpecification object,
}
```

- Ubah `active` bidang menjadi `true` Amazon Lex V2 untuk memanggil kait kode pada saat itu dalam percakapan.
- Ubah `enableCodeHookInvocation` bidang menjadi `true` Amazon Lex V2 untuk memungkinkan hook kode berjalan normal. Jika Anda menandainya`false`, Amazon Lex V2 bertindak seolah-olah pengait kode berhasil dikembalikan.
- `invocationLabel`Ini menunjukkan langkah dialog dari mana kode hook dipanggil.
- Gunakan `postCodeHookSpecification` bidang untuk menentukan tindakan dan pesan yang terjadi setelah hook kode berhasil, gagal, atau habis waktu.

elicitationCodeHook

`elicitationCodeHook`Bidang mendefinisikan pengaturan untuk hook kode untuk dijalankan jika slot atau slot perlu ditimbulkan kembali. Skenario ini dapat terjadi jika elisitasi slot gagal atau konfirmasi maksud ditolak. `elicitationCodeHook`Bidang adalah ElicitationCodeHookInvocationSettingobjek dengan struktur berikut:

```
"elicitationCodeHook": {
    "enableCodeHookInvocation": boolean,
    "invocationLabel": string
}
```

- Ubah `enableCodeHookInvocation` bidang menjadi `true` Amazon Lex V2 untuk memungkinkan hook kode berjalan normal. Jika Anda menandainya`false`, Amazon Lex V2 bertindak seolah-olah pengait kode berhasil dikembalikan.
- `invocationLabel`Ini menunjukkan langkah dialog dari mana kode hook dipanggil.

fulfillmentCodeHook

`fulfillmentCodeHook`Bidang mendefinisikan pengaturan untuk hook kode yang akan dijalankan untuk memenuhi maksud. Ini memetakan ke FulfillmentCodeHookSettingsobjek berikut:

```
"fulfillmentCodeHook": {
    "active": boolean,
    "enabled": boolean,
    "fulfillmentUpdatesSpecification": FulfillmentUpdatesSpecification object,
    "postFulfillmentStatusSpecification": PostFulfillmentStatusSpecification object
}
```

- Ubah `active` bidang menjadi `true` Amazon Lex V2 untuk memanggil kait kode pada saat itu dalam percakapan.
- Ubah `enabled` bidang menjadi `true` Amazon Lex V2 untuk memungkinkan hook kode berjalan normal. Jika Anda menandainya`false`, Amazon Lex V2 bertindak seolah-olah pengait kode berhasil dikembalikan.
- Gunakan `fulfillmentUpdatesSpecification` bidang untuk menentukan pesan yang muncul untuk memperbarui pengguna selama pemenuhan maksud dan waktu yang terkait dengannya.
- Gunakan `postFulfillmentStatusSpecification` bidang untuk menentukan pesan dan tindakan yang terjadi setelah hook kode berhasil, gagal, atau habis waktu.

Anda dapat memanggil hook kode Lambda pada titik-titik berikut dalam percakapan dengan menyetel bidang `enableCodeHookInvocation` and`enabled`/`active`ke: `true`

Selama respon awal

Untuk memanggil fungsi Lambda dalam respons awal setelah maksud dikenali, gunakan struktur `codeHook` di `initialResponse` bidang operasi atau. CreateIntentUpdateIntent `initialResponse`Bidang memetakan ke InitialResponseSettingobjek berikut:

```
"initialResponse": {
    "codeHook": {
        "active": boolean,
        "enableCodeHookInvocation": boolean,
        "invocationLabel": string,
        "postCodeHookSpecification": PostDialogCodeHookInvocationSpecification object,
    },
    "initialResponse": FulfillmentUpdatesSpecification object,
    "nextStep": PostFulfillmentStatusSpecification object,
    "conditional": ConditionalSpecification object
}
```

Setelah elisitasi slot atau selama elisitasi ulang slot

Untuk memanggil fungsi Lambda setelah memunculkan nilai slot, gunakan bidang `slotCaptureSetting` dalam bidang operasi `valueElicitation` atau. CreateSlotUpdateSlot `slotCaptureSetting`Bidang memetakan ke SlotCaptureSettingobjek berikut:

```
"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
}
```

- Untuk menjalankan fungsi Lambda setelah elisitasi slot berhasil, gunakan bidang. `codeHook`
- Untuk menjalankan fungsi Lambda setelah elisitasi slot gagal dan Amazon Lex V2 mencoba lagi elisitasi slot, gunakan bidang tersebut. `elicitationCodeHook`

Setelah konfirmasi maksud atau penolakan

Untuk memanggil fungsi Lambda saat mengonfirmasi maksud, gunakan bidang `intentConfirmationSetting` operasi atau. CreateIntentUpdateIntent `intentConfirmation`Bidang memetakan ke IntentConfirmationSettingobjek berikut:

```
"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
}
```

- Untuk menjalankan fungsi Lambda setelah pengguna mengonfirmasi maksud dan slotnya, gunakan bidang tersebut. `codeHook`
- Untuk menjalankan fungsi Lambda setelah pengguna menolak konfirmasi maksud dan Amazon Lex V2 mencoba lagi elicitation slot, gunakan bidang tersebut. `elicitationCodeHook`

Selama pemenuhan niat

Untuk memanggil fungsi Lambda untuk memenuhi maksud, gunakan bidang dalam `fulfillmentCodeHook` CreateIntentoperasi atau. UpdateIntent `fulfillmentCodeHook`Bidang memetakan ke FulfillmentCodeHookSettingsobjek berikut:

```
{
    "active": boolean,
    "enabled": boolean,
    "fulfillmentUpdatesSpecification": FulfillmentUpdatesSpecification object,
    "postFulfillmentStatusSpecification": PostFulfillmentStatusSpecification object
}
```

3. Setelah Anda mengatur tahapan percakapan untuk menjalankan fungsi Lambda, gunakan `BuildBotLocale` operasi untuk membangun kembali bot untuk menguji fungsi.

Mendebug fungsi CloudWatch Lambda menggunakan log Log

Amazon CloudWatch Logs adalah alat untuk melacak panggilan API dan metrik yang dapat Anda gunakan untuk membantu men-debug fungsi Lambda Anda. Saat Anda menguji bot Anda di konsol atau dengan panggilan API CloudWatch , catat setiap langkah percakapan. Jika Anda menggunakan fungsi cetak dalam kode Lambda Anda, CloudWatch tampilkan juga.

Untuk melihat CloudWatch log untuk fungsi Lambda Anda

1. Masuk ke AWS Management Console dan buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di bawah menu Log di bilah sisi kiri, pilih Grup log.
3. Pilih grup log fungsi Lambda Anda, yang seharusnya dari format. `/aws/lambda/`*`function-name`*
4. Daftar aliran Log berisi log untuk setiap sesi dengan bot. Pilih aliran log untuk melihatnya.
5. Dalam daftar peristiwa Log, pilih panah kanan di sebelah Timestamp untuk memperluas detail acara tersebut. Apa pun yang Anda cetak dari kode Lambda Anda akan muncul sebagai peristiwa log. Gunakan informasi ini untuk men-debug kode Anda.
6. Setelah Anda men-debug kode Anda, ingatlah untuk Menerapkan fungsi Lambda dan, jika Anda menggunakan konsol, untuk memuat ulang jendela Uji sebelum menguji ulang perilaku bot.

Menyesuaikan interaksi bot dengan pengguna di Lex V2

Pelajari tentang fitur berikut yang dapat Anda gunakan untuk menyesuaikan interaksi bot dengan pengguna Anda dengan memperluas dan menyesuaikan perilaku default mereka:

Topik

- Menganalisis sentimen ucapan pengguna dalam percakapan dengan bot Anda
- Menggunakan skor kepercayaan diri untuk meningkatkan akurasi percakapan
- Menyesuaikan transkripsi ucapan untuk digunakan dengan bot Lex V2 Anda
- Menggunakan pengisi audio untuk meningkatkan respons bot

Menganalisis sentimen ucapan pengguna dalam percakapan dengan bot Anda

Anda dapat menggunakan analisis sentimen untuk menentukan sentimen yang diungkapkan dalam ucapan pengguna. Dengan informasi sentimen Anda dapat mengelola alur percakapan atau melakukan analisis pasca-panggilan. Misalnya, jika sentimen pengguna negatif, Anda dapat membuat aliran untuk menyerahkan percakapan ke agen manusia.

Amazon Lex terintegrasi dengan Amazon Comprehend untuk mendeteksi sentimen pengguna. Respons dari Amazon Comprehend menunjukkan apakah sentimen keseluruhan teks positif, netral, negatif, atau campuran. Respons berisi sentimen yang paling mungkin untuk ucapan pengguna dan skor untuk masing-masing kategori sentimen. Skor mewakili kemungkinan bahwa sentimen terdeteksi dengan benar.

Anda mengaktifkan analisis sentimen untuk bot menggunakan konsol atau dengan menggunakan Amazon Lex API. Anda mengaktifkan analisis sentimen pada alias untuk bot. Di konsol Amazon Lex:

1. Pilih alias.
2. Di Detail, pilih Edit.
3. Pilih Aktifkan analisis sentimen untuk analisis sentimen aktif atau nonaktif.
4. Pilih Konfirmasi untuk menyimpan perubahan Anda.

Jika Anda menggunakan API, panggil CreateBotAliasoperasi dengan `detectSentiment` bidang yang disetel `ketrue`.

Ketika analisis sentimen diaktifkan, respons dari RecognizeTextdan RecognizeUtteranceoperasi mengembalikan bidang yang disebut `sentimentResponse` dalam `interpretations` struktur dengan metadata lainnya. `sentimentResponse`Bidang ini memiliki dua bidang`sentimentScore`, `sentiment` dan, yang berisi hasil analisis sentimen. Jika Anda menggunakan fungsi Lambda, `sentimentResponse` bidang tersebut disertakan dalam data peristiwa yang dikirim ke fungsi Anda.

Berikut ini adalah contoh `sentimentResponse` bidang yang dikembalikan sebagai bagian dari `RecognizeText` atau `RecognizeUtterance` respons.

```
sentimentResponse {
    "sentimentScore": {
        "mixed": 0.030585512690246105,
        "positive": 0.94992071056365967,
        "neutral": 0.0141543131828308,
        "negative": 0.00893945890665054
    },
    "sentiment": "POSITIVE"
}
```

Amazon Lex memanggil Amazon Comprehend atas nama Anda untuk menentukan sentimen dalam setiap ucapan yang diproses oleh bot. Dengan mengaktifkan analisis sentimen, Anda menyetujui persyaratan dan perjanjian layanan untuk Amazon Comprehend. Untuk informasi selengkapnya tentang harga Amazon Comprehend, lihat Harga Amazon Comprehend.

Untuk informasi selengkapnya tentang cara kerja analisis sentimen Amazon Comprehend, lihat Menentukan sentimen di Panduan Pengembang Amazon Comprehend.

Menggunakan skor kepercayaan diri untuk meningkatkan akurasi percakapan

Ada dua langkah yang digunakan Amazon Lex V2 untuk menentukan apa yang dikatakan pengguna. Yang pertama, pengenalan suara otomatis (ASR), membuat transkrip ucapan audio pengguna. Yang kedua, pemahaman bahasa alami (NLU), menentukan arti ucapan pengguna untuk mengenali maksud pengguna atau nilai slot.

Secara default, Amazon Lex V2 mengembalikan hasil yang paling mungkin dari ASR dan NLU. Kadang-kadang mungkin sulit bagi Amazon Lex V2 untuk menentukan hasil yang paling mungkin. Dalam hal ini, ia mengembalikan beberapa hasil yang mungkin bersama dengan skor kepercayaan yang menunjukkan seberapa besar kemungkinan hasilnya benar. Skor kepercayaan adalah peringkat

yang diberikan Amazon Lex V2 yang menunjukkan kepercayaan relatif yang dimilikinya dalam hasilnya. Skor kepercayaan berkisar dari 0,0 hingga 1,0.

Anda dapat menggunakan pengetahuan domain Anda dengan skor kepercayaan untuk membantu menentukan interpretasi yang benar dari hasil ASR atau NLU.

Skor kepercayaan ASR, atau transkripsi, adalah peringkat tentang seberapa yakin Amazon Lex V2 bahwa transkripsi tertentu benar. Skor kepercayaan NLU, atau maksud, adalah peringkat tentang seberapa yakin Amazon Lex V2 bahwa maksud yang ditentukan oleh transkripsi teratas adalah benar. Gunakan skor kepercayaan yang paling sesuai dengan aplikasi Anda.

Topik

- Menggunakan skor keyakinan niat untuk meningkatkan pemilihan maksud dengan Lex V2
- Menggunakan skor kepercayaan transkripsi suara untuk meningkatkan percakapan dengan bot Lex V2 Anda

Menggunakan skor keyakinan niat untuk meningkatkan pemilihan maksud dengan Lex V2

Saat pengguna mengucapkan ucapan, Amazon Lex V2 menggunakan pemahaman bahasa alami (NLU) untuk memahami permintaan pengguna dan mengembalikan maksud yang tepat. Secara default Amazon Lex V2 mengembalikan maksud yang paling mungkin ditentukan oleh bot Anda.

Dalam beberapa kasus mungkin sulit bagi Amazon Lex V2 untuk menentukan maksud yang paling mungkin. Misalnya, pengguna mungkin membuat ucapan ambigu, atau mungkin ada dua maksud yang serupa. Untuk membantu menentukan maksud yang tepat, Anda dapat menggabungkan pengetahuan domain Anda dengan skor kepercayaan NLU dalam daftar interpretasi. Skor kepercayaan diri adalah peringkat yang diberikan Amazon Lex V2 yang menunjukkan betapa yakinnya bahwa niat adalah niat yang benar.

Untuk menentukan perbedaan antara dua maksud dalam interpretasi, Anda dapat membandingkan skor kepercayaan mereka. Misalnya, jika satu niat memiliki skor kepercayaan 0,95 dan yang lain memiliki skor 0,65, maksud pertama mungkin benar. Namun, jika satu maksud memiliki skor 0,75 dan yang lain memiliki skor 0,72, ada ambiguitas antara dua maksud yang mungkin dapat Anda bedakan menggunakan pengetahuan domain dalam aplikasi Anda.

Anda juga dapat menggunakan skor kepercayaan untuk membuat aplikasi pengujian yang menentukan apakah perubahan pada ucapan maksud membuat perbedaan dalam perilaku bot.

Misalnya, Anda bisa mendapatkan skor kepercayaan untuk maksud bot menggunakan serangkaian ucapan, lalu memperbarui maksud dengan ucapan baru. Anda kemudian dapat memeriksa skor kepercayaan diri untuk melihat apakah ada peningkatan.

Skor kepercayaan yang dikembalikan Amazon Lex V2 adalah nilai komparatif. Anda tidak harus mengandalkan mereka sebagai skor absolut. Nilai dapat berubah berdasarkan peningkatan Amazon Lex V2.

Amazon Lex V2 mengembalikan maksud yang paling mungkin dan hingga 4 maksud alternatif dengan skor terkait dalam `interpretations` struktur di setiap respons. Kode JSON berikut menunjukkan `interpretations` struktur dalam respon dari RecognizeTextoperasi:

```
"interpretations": [
   {
      "intent": {
         "confirmationState": "string",
         "name": "string",
         "slots": {
            "string" : {
               "value": {
                  "interpretedValue": "string",
                  "originalValue": "string",
                  "resolvedValues": [ "string" ]
               }
            }
         },
         "state": "string"
      },
      "nluConfidence": number
   }
]
```

AMAZON. FallbackIntent

Amazon Lex V2 kembali `AMAZON.FallbackIntent` sebagai tujuan utama dalam dua situasi:

1. Jika skor kepercayaan dari semua maksud yang mungkin kurang dari ambang kepercayaan. Anda dapat menggunakan ambang batas default atau Anda dapat mengatur ambang batas Anda sendiri. Jika Anda memiliki `AMAZON.KendraSearchIntent` konfigurasi, Amazon Lex V2 mengembalikannya juga dalam situasi ini.

2. Jika keyakinan interpretasi untuk `AMAZON.FallbackIntent` lebih tinggi dari kepercayaan interpretasi dari semua maksud lainnya.

Perhatikan bahwa Amazon Lex V2 tidak menampilkan skor kepercayaan untuk`AMAZON.FallbackIntent`.

Menetapkan dan mengubah ambang kepercayaan

Ambang batas kepercayaan harus berupa angka antara 0,00 dan 1,00. Anda dapat mengatur ambang batas untuk setiap bahasa di bot Anda dengan cara berikut:

Menggunakan konsol Amazon Lex V2

- Untuk mengatur ambang batas saat Anda menambahkan bahasa ke bot Anda dengan Tambahkan bahasa, Anda dapat memasukkan nilai yang Anda inginkan di panel ambang skor Keyakinan.
- Untuk memperbarui ambang batas, Anda dapat memilih Edit di panel Detail bahasa dalam bahasa untuk bot Anda. Kemudian masukkan nilai yang Anda inginkan di panel ambang skor Keyakinan.

Menggunakan operasi API

- Untuk mengatur ambang batas, atur `nluIntentConfidenceThreshold` parameter CreateBotLocaleoperasi.
- Untuk memperbarui ambang kepercayaan, atur `nluIntentConfidenceThreshold` parameter UpdateBotLocaleoperasi.

Manajemen Sesi

Untuk mengubah maksud yang digunakan Amazon Lex V2 dalam percakapan dengan pengguna, Anda dapat menggunakan respons dari fungsi Lambda kait kode dialog, atau Anda dapat menggunakan APIs manajemen sesi di aplikasi kustom Anda.

Menggunakan fungsi Lambda dengan bot Lex V2 Anda

Saat Anda menggunakan fungsi Lambda, Amazon Lex V2 memanggilnya dengan struktur JSON yang berisi input ke fungsi tersebut. Struktur JSON berisi bidang bernama `currentIntent` yang berisi maksud yang diidentifikasi Amazon Lex V2 sebagai maksud yang paling mungkin untuk ucapan pengguna. Struktur JSON juga mencakup `alternativeIntents` bidang yang berisi hingga empat intent tambahan yang dapat memenuhi maksud pengguna. Setiap intent menyertakan bidang yang

disebut `nluIntentConfidenceScore` yang berisi skor kepercayaan yang ditetapkan Amazon Lex V2 untuk maksud tersebut.

Untuk menggunakan intent alternatif, Anda menentukannya dalam `ConfirmIntent` atau tindakan `ElicitSlot` dialog dalam fungsi Lambda Anda.

Untuk informasi selengkapnya, lihat Mengintegrasikan AWS Lambda fungsi ke bot Amazon Lex V2 Anda.

Menggunakan Session Management API dengan bot Lex V2

Untuk menggunakan intent yang berbeda dari intent saat ini, gunakan operasi. PutSession Misalnya, jika Anda memutuskan bahwa alternatif pertama lebih disukai daripada maksud yang dipilih Amazon Lex V2, Anda dapat menggunakan `PutSession` operasi untuk mengubah maksud sehingga maksud berikutnya yang berinteraksi dengan pengguna adalah yang Anda pilih.

Lihat informasi yang lebih lengkap di Memahami sesi bot Amazon Lex V2.

Menggunakan skor kepercayaan transkripsi suara untuk meningkatkan percakapan dengan bot Lex V2 Anda

Saat pengguna membuat ucapan suara, Amazon Lex V2 menggunakan pengenalan suara otomatis (ASR) untuk mentranskripsikan permintaan pengguna sebelum ditafsirkan. Secara default, Amazon Lex V2 menggunakan transkripsi audio yang paling mungkin untuk interpretasi.

Dalam beberapa kasus mungkin ada lebih dari satu kemungkinan transkripsi audio. Misalnya, pengguna mungkin membuat ucapan dengan suara ambigu, seperti “Nama saya John” yang mungkin dipahami sebagai “Nama saya Juan.” Dalam hal ini, Anda dapat menggunakan teknik disambiguasi atau menggabungkan pengetahuan domain Anda dengan skor kepercayaan transkripsi untuk membantu menentukan transkripsi mana dalam daftar transkripsi yang benar.

Amazon Lex V2 menyertakan transkripsi teratas dan hingga dua transkripsi alternatif untuk input pengguna dalam permintaan ke fungsi kait kode Lambda Anda. Setiap transkripsi berisi skor kepercayaan bahwa itu adalah transkripsi yang benar. Setiap transkripsi juga mencakup nilai slot apa pun yang disimpulkan dari input pengguna.

Anda dapat membandingkan skor kepercayaan dari dua transkripsi untuk menentukan apakah ada ambiguitas di antara keduanya. Misalnya, jika satu transkripsi memiliki skor kepercayaan 0,95 dan yang lainnya memiliki skor kepercayaan 0,65, transkripsi pertama mungkin benar dan ambiguitas di antara keduanya rendah. Jika kedua transkripsi memiliki skor kepercayaan 0,75 dan

0,72, ambiguitas di antara keduanya tinggi. Anda mungkin dapat membedakan antara mereka menggunakan pengetahuan domain Anda.

Misalnya, jika nilai slot yang disimpulkan dalam dua transkrip dengan skor kepercayaan 0,75 dan 0,72 adalah “John” dan “Juan”, Anda dapat menanyakan pengguna di database Anda untuk keberadaan nama-nama ini dan menghilangkan salah satu transkripsi. Jika “John” bukan pengguna di database Anda dan “Juan” adalah, Anda dapat menggunakan hook kode dialog untuk mengubah nilai slot yang disimpulkan untuk nama depan menjadi “Juan.”

Skor kepercayaan yang dikembalikan Amazon Lex V2 adalah nilai komparatif. Jangan mengandalkan mereka sebagai skor absolut. Nilai dapat berubah berdasarkan peningkatan Amazon Lex V2.

Skor kepercayaan transkripsi audio hanya tersedia dalam bahasa Inggris (GB) (en_GB) dan Inggris (AS) (en_US). Skor kepercayaan hanya didukung untuk input audio 8 kHz. Skor kepercayaan transkripsi tidak disediakan untuk input audio dari jendela pengujian di konsol Amazon Lex V2 karena menggunakan input audio 16 kHz.

> **Note**
>
> Sebelum Anda dapat menggunakan skor kepercayaan transkripsi audio dengan bot yang ada, Anda harus membangun kembali bot terlebih dahulu. Versi bot yang ada tidak mendukung skor kepercayaan transkripsi. Anda harus membuat versi baru bot untuk menggunakannya.

Anda dapat menggunakan skor kepercayaan diri untuk beberapa pola desain percakapan:

- Jika skor kepercayaan tertinggi turun di bawah ambang batas karena lingkungan yang bising atau kualitas sinyal yang buruk, Anda dapat meminta pengguna dengan pertanyaan yang sama untuk menangkap audio berkualitas lebih baik.
- Jika beberapa transkripsi memiliki skor kepercayaan yang sama untuk nilai slot, seperti “John” dan “Juan,” Anda dapat membandingkan nilai dengan database yang sudah ada sebelumnya untuk menghilangkan input, atau Anda dapat meminta pengguna untuk memilih salah satu dari dua nilai tersebut. Misalnya, “katakan 1 untuk Yohanes atau katakan 2 untuk Juan.”
- Jika logika bisnis Anda memerlukan pengalihan maksud berdasarkan kata kunci tertentu dalam transkrip alternatif dengan skor kepercayaan dekat dengan transkrip teratas, Anda dapat mengubah maksud menggunakan fungsi Lambda kait kode dialog atau menggunakan operasi manajemen sesi. Untuk informasi selengkapnya, lihat Manajemen sesi.

Amazon Lex V2 mengirimkan struktur JSON berikut dengan hingga tiga transkripsi untuk masukan pengguna ke fungsi kait kode Lambda Anda:

```
"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"
                            ]
                        }
                    }
                ]
            }
```

```
            }
        }
    ]
```

Struktur JSON berisi teks transkripsi, maksud yang diselesaikan untuk ucapan, dan nilai untuk setiap slot yang terdeteksi dalam ucapan. Untuk input pengguna teks, transkripsi berisi transkrip tunggal dengan skor kepercayaan 1.0.

Isi transkrip tergantung pada pergantian percakapan dan maksud yang diakui.

Untuk giliran pertama, elisitasi maksud, Amazon Lex V2 menentukan tiga transkripsi teratas. Untuk transkripsi teratas, ia mengembalikan maksud dan nilai slot yang disimpulkan dalam transkripsi.

Pada belokan berikutnya, elisitasi slot, hasilnya tergantung pada maksud yang disimpulkan untuk masing-masing transkripsi, sebagai berikut.

- Jika maksud yang disimpulkan untuk transkrip teratas sama dengan giliran sebelumnya dan semua transkrip lainnya memiliki maksud yang sama, maka
  - Semua transkrip berisi nilai slot yang disimpulkan.

- Jika maksud yang disimpulkan untuk transkrip teratas berbeda dari giliran sebelumnya dan semua transkrip lainnya memiliki maksud sebelumnya, maka
  - Transkrip teratas berisi nilai slot yang disimpulkan untuk maksud baru.
  - Transkrip lain memiliki maksud sebelumnya dan nilai slot yang disimpulkan untuk maksud sebelumnya.

- Jika maksud yang disimpulkan untuk transkrip teratas berbeda dari giliran sebelumnya, satu transkrip sama dengan maksud sebelumnya, dan satu transkrip adalah maksud yang berbeda, maka
  - Transkrip teratas berisi maksud baru yang disimpulkan dan nilai slot yang disimpulkan dalam ucapan.
  - Transkrip yang memiliki maksud yang disimpulkan sebelumnya berisi nilai slot yang disimpulkan untuk maksud tersebut.
  - Transkrip dengan maksud yang berbeda tidak memiliki nama maksud yang disimpulkan dan tidak ada nilai slot yang disimpulkan.

- Jika maksud yang disimpulkan untuk transkrip teratas berbeda dari giliran sebelumnya dan semua transkrip lainnya memiliki maksud yang berbeda, maka
  - Transkrip teratas berisi maksud baru yang disimpulkan dan nilai slot yang disimpulkan dalam ucapan.
  - Transkrip lain tidak mengandung maksud yang disimpulkan dan tidak ada nilai slot yang disimpulkan.

- Jika maksud yang disimpulkan untuk dua transkrip teratas adalah sama dan berbeda dari giliran sebelumnya, dan transkrip ketiga adalah maksud yang berbeda, maka
  - Dua transkrip teratas berisi maksud baru yang disimpulkan dan nilai slot yang disimpulkan dalam ucapan.
  - Transkrip ketiga tidak memiliki nama maksud dan tidak ada nilai slot yang diselesaikan.

Manajemen sesi

Untuk mengubah maksud yang digunakan Amazon Lex V2 dalam percakapan dengan pengguna, gunakan respons dari fungsi Lambda kait kode dialog Anda. Atau Anda dapat menggunakan manajemen sesi APIs dalam aplikasi khusus Anda.

Menggunakan fungsi Lambda dengan bot Lex V2 Anda

Saat Anda menggunakan fungsi Lambda, Amazon Lex V2 memanggilnya dengan struktur JSON yang berisi input ke fungsi tersebut. Struktur JSON berisi bidang yang disebut `transcriptions` yang berisi kemungkinan transkripsi yang telah ditentukan Amazon Lex V2 untuk ucapan tersebut. `transcriptions`Bidang berisi satu hingga tiga kemungkinan transkripsi, masing-masing dengan skor kepercayaan.

Untuk menggunakan intent dari transkripsi alternatif, Anda menentukannya dalam `ConfirmIntent` atau tindakan `ElicitSlot` dialog dalam fungsi Lambda Anda. Untuk menggunakan nilai slot dari transkripsi alternatif, tetapkan nilai di `intent` bidang dalam respons fungsi Lambda Anda. Untuk informasi selengkapnya, lihat Mengintegrasikan AWS Lambda fungsi ke bot Amazon Lex V2 Anda.

Contoh kode menggunakan Lambda dengan Lex V2

Contoh kode berikut adalah fungsi Lambda Python yang menggunakan transkripsi audio untuk meningkatkan pengalaman percakapan bagi pengguna.

Untuk menggunakan kode contoh, Anda harus memiliki:

- Bot dengan satu bahasa, baik Inggris (GB) (en_GB) atau Inggris (AS) (en_US).
- Satu niat, OrderBirthStone. Pastikan bahwa fungsi Use a Lambda untuk inisialisasi dan validasi dipilih di bagian Code hooks dari definisi intent.
- Maksudnya harus memiliki dua slot, "BirthMonth" dan “Nama,” keduanya jenis`AMAZON.AlphaNumeric`.
- Sebuah alias dengan fungsi Lambda didefinisikan. Untuk informasi selengkapnya, lihat Membuat AWS Lambda fungsi untuk bot Amazon Lex V2 Anda.

```
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)
```

Menggunakan API manajemen sesi untuk memilih intent atau nilai slot yang berbeda

Untuk menggunakan intent yang berbeda dari intent saat ini, gunakan operasi. PutSession Misalnya, jika Anda memutuskan bahwa alternatif pertama lebih disukai daripada maksud yang dipilih Amazon Lex V2, Anda dapat menggunakan `PutSession` operasi untuk mengubah maksud. Dengan begitu maksud berikutnya yang berinteraksi dengan pengguna akan menjadi yang Anda pilih.

Anda juga dapat menggunakan `PutSession` operasi untuk mengubah nilai slot dalam `intent` struktur untuk menggunakan nilai dari transkripsi alternatif.

Lihat informasi yang lebih lengkap di Memahami sesi bot Amazon Lex V2.

Menyesuaikan transkripsi ucapan untuk digunakan dengan bot Lex V2 Anda

Perilaku default bot Anda terkadang dapat mengakibatkan transkripsi ucapan yang tidak akurat. Fitur-fitur berikut tersedia untuk membantu bot Anda mengenali kata atau nama yang kurang umum atau mudah bingung.

Topik

- Mengkonfigurasi preferensi model pengenalan suara
- Meningkatkan pengenalan suara dengan kosakata khusus
- Mengkonfigurasi sensitivitas deteksi aktivitas suara
- Menggunakan diarisasi speaker untuk fokus pada pembicara utama
- Meningkatkan pengenalan nilai slot dengan petunjuk runtime dalam percakapan
- Menangkap nilai slot dengan gaya ejaan selama percakapan

Mengkonfigurasi preferensi model pengenalan suara

Amazon Lex V2 menyediakan berbagai model pengenalan suara yang dapat Anda pilih untuk mengoptimalkan akurasi dan kinerja kemampuan pengenalan suara bot Anda. Anda dapat

mengonfigurasi preferensi model ucapan untuk memilih model yang paling sesuai untuk kasus penggunaan Anda.

Jenis model pengenalan ucapan

Amazon Lex V2 mendukung model pengenalan suara berikut:

Model standar

Model pengenalan suara standar memberikan kinerja pengenalan suara yang andal untuk kasus penggunaan umum. Model ini menawarkan akurasi yang konsisten di berbagai kondisi audio dan cocok untuk sebagian besar aplikasi AI percakapan.

Model saraf

Model pengenalan suara saraf memberikan akurasi yang ditingkatkan dan penanganan yang lebih baik dari pola ucapan alami, aksen, dan kebisingan latar belakang. Model ini menggunakan arsitektur jaringan saraf canggih untuk meningkatkan kinerja pengenalan, terutama di lingkungan audio yang menantang.

Deepgram

Deepgram menyediakan public speech-to-text (STT) API untuk pengguna yang membuat akun dan kunci API. Lihat https://deepgram.com/untuk informasi tentang penawaran umum mereka.

Mengkonfigurasi preferensi model ucapan

Anda dapat mengonfigurasi preferensi model ucapan saat membuat atau memperbarui lokal bot. Pengaturan preferensi model ucapan menentukan model pengenalan mana yang digunakan Amazon Lex V2 untuk memproses input audio untuk bot Anda.

Untuk mengonfigurasi preferensi model ucapan:

1. Di konsol Amazon Lex V2, navigasikan ke bot Anda dan pilih lokal yang ingin Anda konfigurasikan.
2. Di pengaturan lokal bot, cari bagian Pengaturan pengenalan ucapan.
3. Untuk preferensi model Ucapan, pilih salah satu opsi berikut:
   - Standar - Gunakan model pengenalan suara standar untuk kinerja yang andal di seluruh kasus penggunaan umum.
   - Neural - Gunakan model pengenalan ucapan saraf untuk meningkatkan akurasi dan penanganan pola bicara alami yang lebih baik.

- Deepgram - Gunakan API Listen Deepgram untuk pengenalan suara. Untuk petunjuk penyiapan, lihatthe section called “Menyiapkan preferensi model ucapan Deepgram”.

4. Simpan perubahan Anda untuk menerapkan preferensi model ucapan ke lokal bot Anda.

> **Note**
>
> Jika Anda tidak menentukan preferensi model ucapan, Amazon Lex V2 menggunakan model standar secara default.

Memilih model pidato yang tepat

Pertimbangkan faktor-faktor berikut saat memilih model pengenalan suara untuk bot Anda:

- Kualitas audio - Jika bot Anda akan memproses audio dengan kebisingan latar belakang, kualitas audio yang bervariasi, atau kondisi akustik yang menantang, model saraf dapat memberikan akurasi yang lebih baik.
- Keragaman pembicara - Jika bot Anda akan berinteraksi dengan pengguna yang memiliki aksen atau pola bicara yang beragam, kemampuan pemrosesan bahasa alami model saraf yang ditingkatkan dapat meningkatkan kinerja pengenalan.
- Persyaratan kinerja - Model standar memberikan kinerja yang konsisten dan mungkin cukup untuk aplikasi dengan lingkungan audio yang terkontrol dan input ucapan yang jelas.

Anda dapat menguji kedua model dengan kasus penggunaan khusus Anda untuk menentukan mana yang memberikan keseimbangan akurasi dan kinerja terbaik untuk aplikasi Anda.

Menyiapkan preferensi model ucapan Deepgram

Deepgram adalah layanan pengenalan suara pihak ketiga yang menyediakan speech-to-text kemampuan canggih yang didukung AI dengan dukungan untuk pemrosesan real-time dan batch. Deepgram menawarkan akurasi yang ditingkatkan di berbagai kondisi audio, berbagai bahasa, dan model khusus untuk kasus penggunaan yang berbeda. Untuk informasi lebih lanjut tentang penawaran Deepgram, lihat/. https://deepgram.com Untuk menggunakan Deepgram sebagai preferensi model pengenalan suara, Anda harus menyelesaikan proses penyiapan satu kali untuk mengonfigurasi kunci API Deepgram Anda dan menyimpannya dengan aman di AWS Secrets Manager.

> **Important**
>
> Deepgram adalah layanan pihak ketiga dan mungkin tidak mematuhi kerangka peraturan tertentu seperti GDPR, FedRAMP, atau standar kepatuhan lainnya yang dipatuhi oleh layanan AWS. Tinjau dokumentasi kepatuhan Deepgram dan persyaratan organisasi Anda sebelum menggunakan integrasi ini.

Pemilihan titik akhir regional

Amazon Lex V2 secara otomatis memilih titik akhir API Deepgram yang sesuai berdasarkan wilayah AWS Anda untuk mengoptimalkan kinerja dan lokalitas data:

- Wilayah UE: Untuk bot yang digunakan di wilayah AWS dengan `eu-` awalan (seperti`eu-west-1`,, dan`eu-central-1`)`eu-west-2`, Amazon Lex V2 menggunakan titik akhir Deepgram EU (). `api.eu.deepgram.com`
- Semua wilayah lain: Untuk bot yang digunakan di semua wilayah AWS lainnya, Amazon Lex V2 menggunakan titik akhir Deepgram global (). `api.deepgram.com`

Pemilihan titik akhir ini otomatis dan saat ini tidak dapat disesuaikan. Kunci API Deepgram yang sama berfungsi dengan kedua titik akhir. Amazon Lex V2 tidak mendukung `apiTokenRegion` parameter yang disediakan Amazon Connect untuk konfigurasi titik akhir kustom. Untuk informasi selengkapnya tentang opsi konfigurasi titik akhir Amazon Connect, lihat Titik Akhir dan Wilayah untuk penyedia STT pihak ketiga di Panduan Administrator Amazon Connect.

Membuat kunci API Deepgram

Sebelum Anda dapat menggunakan Deepgram dengan Amazon Lex V2, Anda perlu mendapatkan kunci API dari Deepgram.

Untuk membuat kunci API Deepgram:

1. Masuk ke konsol Deepgram di https://console.deepgram.com/.
2. Di panel navigasi kiri, pilih Kunci API.
3. Pilih Buat Kunci API Baru.
4. Ikuti petunjuk untuk membuat kunci API dan menyalinnya untuk digunakan di bagian berikutnya.

> **Important**
>
> Simpan kunci API Anda dengan aman. Anda akan memerlukannya untuk mengonfigurasi AWS Secrets Manager di bagian berikutnya.

Menyimpan kunci API di AWS Secrets Manager

Anda harus menyimpan kunci API Deepgram Anda di AWS Secrets Manager untuk Amazon Lex V2 untuk mengaksesnya dengan aman. Rahasia harus berisi satu pasangan kunci-nilai dengan `apiToken` sebagai kunci dan kunci API Deepgram Anda sebagai nilainya.

> **Important**
>
> Anda harus membuat kunci KMS simetris untuk digunakan dengan rahasia. Kunci KMS AWS terkelola default tidak akan berfungsi dengan Amazon Lex V2.

Untuk menyimpan kunci API Deepgram Anda di Secrets Manager:

1. Buka konsol AWS Secrets Manager di https://console.aws.amazon.com/secretsmanager/.
2. Pilih Simpan rahasia baru.
3. Untuk Tipe rahasia, pilih Tipe rahasia lainnya.
4. Konfigurasikan rahasia menggunakan salah satu metode berikut:
   - Metode pasangan kunci/nilai: Di bawah pasangan kunci/nilai, tambahkan satu pasangan kunci-nilai dengan kunci dan kunci API Deepgram Anda `apiToken` sebagai nilainya.
   - Metode Plaintext: Di bawah Plaintext, masukkan objek JSON dengan struktur berikut:

     ```
     {
       "apiToken": "your-deepgram-api-key-here"
     }
     ```
5. Pilih Berikutnya.
6. Masukkan nama untuk rahasia Anda dan pilih Berikutnya.
7. (Opsional) Konfigurasikan rotasi rahasia jika diperlukan, lalu pilih Berikutnya.

8. Tinjau konfigurasi rahasia Anda dan pilih Store.
9. Setelah rahasia dibuat, arahkan ke rahasia Anda dan salin ARN. Anda akan memerlukan ARN ini saat mengonfigurasi bot Anda.

Mengkonfigurasi kebijakan sumber daya untuk Secrets Manager

Untuk mengizinkan Amazon Lex V2 mengambil kunci API Deepgram Anda, Anda harus melampirkan kebijakan sumber daya ke rahasia Anda.

Berikut ini adalah contoh kebijakan sumber daya yang memungkinkan Amazon Lex V2 untuk mengambil rahasia:

```
{
  "Version": "2012-10-17"       ,
  "Statement": [
    {
      "Sid": "LexTrust",
      "Effect": "Allow",
      "Principal": {
        "Service": "lex.amazonaws.com"
      },
      "Action": "secretsmanager:GetSecretValue",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "<YOUR_ACCOUNT_ID>"
        },
        "ArnLike": {
          "aws:SourceArn": "arn:aws:lex:us-east-1:<YOUR_ACCOUNT_ID>:bot-alias/*/*"
        }
      }
    }
  ]
}
```

Ganti <YOUR_ACCOUNT_ID> dengan ID akun AWS Anda yang sebenarnya dan sesuaikan wilayah dalam pola ARN sesuai kebutuhan untuk penerapan Anda.

Mengkonfigurasi bot Anda untuk menggunakan Deepgram

Setelah menyiapkan kunci API Deepgram Anda di Secrets Manager, Anda dapat mengonfigurasi bot Amazon Lex V2 Anda untuk menggunakan Deepgram untuk pengenalan suara.

Untuk mengkonfigurasi Deepgram untuk bot Anda:

1. Di konsol Amazon Lex V2, navigasikan ke bot Anda dan pilih lokal yang ingin Anda konfigurasikan.
2. Untuk preferensi model Speech, pilih Deepgram.
3. Bidang tambahan akan muncul untuk konfigurasi Deepgram:
   - Model ID (opsional) - Tentukan ID model Deepgram jika Anda ingin menggunakan model tertentu. Untuk model yang tersedia, lihat dokumentasi model Deepgram. Jika dibiarkan kosong, model default API akan digunakan.
   - Rahasia ARN (wajib) - Masukkan ARN rahasia yang Anda buat di AWS Secrets Manager yang berisi kunci API Deepgram Anda.
4. Simpan perubahan Anda untuk menerapkan preferensi model ucapan Deepgram ke lokal bot Anda.

Bot Anda sekarang dikonfigurasi untuk menggunakan Deepgram untuk pengenalan suara. Uji bot Anda untuk memastikan bahwa pengenalan suara berfungsi seperti yang diharapkan dengan integrasi Deepgram.

Memecahkan masalah integrasi Deepgram

Jika Anda mengalami masalah dengan integrasi Deepgram Anda, periksa hal berikut:

- Validitas kunci API: Pastikan kunci API Deepgram Anda valid dan belum kedaluwarsa.
- Konfigurasi rahasia: Verifikasi bahwa rahasia Anda di AWS Secrets Manager berisi nama kunci (`apiToken`) dan nilai kunci API yang benar.
- Kebijakan sumber daya: Konfirmasikan bahwa kebijakan sumber daya pada rahasia Anda memungkinkan Amazon Lex V2 mengaksesnya dengan ID akun dan pola ARN yang benar.
- Kunci KMS: Pastikan Anda menggunakan kunci KMS simetris yang dikelola pelanggan, bukan kunci terkelola AWS default.
- ID Model: Jika Anda menentukan ID model, verifikasi bahwa itu adalah pengidentifikasi model Deepgram yang valid.

Untuk dukungan tambahan, lihat CloudWatch log Amazon Lex V2 atau hubungi AWS Support.

Meningkatkan pengenalan suara dengan kosakata khusus

Anda dapat memberikan Amazon Lex V2 informasi lebih lanjut tentang cara memproses percakapan audio dengan bot dengan membuat kosakata khusus dalam bahasa tertentu. Kosakata khusus adalah daftar frasa tertentu yang Anda ingin Amazon Lex V2 kenali dalam input audio. Ini umumnya adalah kata benda yang tepat atau kata-kata khusus domain yang tidak dikenali Amazon Lex V2.

Misalnya, Anda memiliki bot dukungan teknis. Anda dapat menambahkan “cadangan” ke kosakata khusus untuk membantu bot mentranskripsikan audio dengan benar sebagai “cadangan”, bahkan ketika audio terdengar seperti “berkemas.” Kosakata khusus juga dapat membantu mengenali kata-kata langka dalam audio seperti “solvabilitas” untuk layanan keuangan atau kata benda yang tepat seperti “Cognito” atau “Monitron.”

Dasar-dasar kosakata khusus

- Kosakata khusus bekerja pada transkripsi input audio ke bot. Anda harus memberikan contoh ucapan untuk mengenali maksud atau nilai slot.
- Kosakata khusus unik untuk bahasa tertentu. Anda harus mengonfigurasi kosakata khusus secara independen untuk setiap bahasa. Kosakata khusus didukung untuk bahasa-bahasa berikut: Inggris (Inggris), Inggris (AS), Korea (Korea), Italia (Italia), Jerman (Jerman), Prancis (Kanada), Prancis (Prancis), Spanyol (Spanyol), Spanyol (AS), Jepang (Jepang), Mandarin (RRC), Swedia (Swedia), Arab Teluk (Bersatu) Uni Emirat Arab), Hindi (India), Belanda (Belanda), Norwegia (Norwegia), Polandia (Polandia), Finlandia (Finlandia), Portugis (Portugal), Portugis (Brasil), Catalan (Spanyol).
- Kosakata khusus tersedia dengan integrasi pusat kontak yang didukung oleh Amazon Lex V2. Jendela pengujian di konsol Amazon Lex V2 mendukung kosakata khusus untuk semua bot Amazon Lex V2 yang dibangun pada atau setelah 31 Juli 2022. Jika Anda mengalami masalah dengan kosakata khusus di jendela pengujian, bangun kembali bot dan coba lagi.

Amazon Lex V2 menggunakan kosakata khusus untuk memperoleh maksud dan slot. File kosakata kustom yang sama digunakan untuk maksud dan slot. Anda dapat secara selektif mematikan kemampuan kosakata khusus untuk slot saat Anda menambahkan jenis slot.

Memunculkan maksud — Anda dapat membuat kosakata khusus untuk memunculkan maksud. Frasa ini digunakan untuk transkripsi saat bot Anda menentukan maksud pengguna. Misalnya, jika Anda mengonfigurasi frasa “cadangan” dalam kosakata khusus Anda, Amazon Lex V2 mentranskripsikan input pengguna ke “bisakah Anda mencadangkan foto saya?” —bahkan ketika audio terdengar seperti “bisakah Anda mengemas foto saya.” Anda dapat menentukan tingkat peningkatan untuk

setiap frasa dengan mengonfigurasi bobot 0, 1, 2, atau 3. Anda juga dapat menentukan representasi alternatif untuk frasa dalam pidato akhir ke output teks dengan menambahkan `displayAs` bidang.

Frasa kosakata khusus yang digunakan untuk meningkatkan transkripsi selama elisitasi maksud tidak memengaruhi transkripsi saat memunculkan slot. Untuk informasi selengkapnya tentang membuat kosakata khusus untuk memunculkan maksud, lihat. Membuat kosakata khusus untuk memunculkan maksud dan slot

Memunculkan slot khusus — Anda dapat menggunakan kosakata khusus untuk meningkatkan pengenalan slot untuk percakapan audio. Untuk meningkatkan kemampuan bot Amazon Lex V2 Anda untuk mengenali nilai slot, buat slot khusus dan tambahkan nilai slot ke slot khusus, lalu pilih Gunakan nilai slot sebagai kosakata khusus. Contoh nilai slot termasuk nama produk, katalog, atau kata benda yang tepat. Anda tidak boleh menggunakan kata atau frasa umum seperti “ya” dan “tidak” dalam kosakata khusus.

Setelah nilai slot ditambahkan, nilai-nilai ini digunakan untuk meningkatkan pengenalan slot saat bot mengharapkan input untuk slot khusus. Nilai-nilai ini tidak digunakan untuk transkripsi saat memunculkan maksud. Untuk informasi selengkapnya, lihat Menambahkan jenis slot.

Praktik terbaik untuk membuat kosakata khusus

Memunculkan niat

- Kosakata khusus bekerja paling baik bila digunakan untuk menargetkan kata atau frasa tertentu. Hanya tambahkan kata-kata ke kosakata khusus jika tidak mudah dikenali oleh Amazon Lex V2.
- Putuskan berapa banyak bobot untuk memberikan kata berdasarkan seberapa sering kata tersebut tidak dikenali dalam transkripsi dan seberapa jarang kata tersebut dalam input. Sulit untuk mengucapkan kata-kata membutuhkan bobot yang lebih tinggi.
- Gunakan set tes representatif untuk menentukan apakah berat badan sesuai. Anda dapat mengumpulkan set pengujian audio dengan mengaktifkan pencatatan audio di log percakapan.
- Hindari menggunakan kata-kata pendek seperti “on,” “it,” “to,” “yes,” “no” dalam kosakata khusus.

Memunculkan slot khusus

- Tambahkan nilai ke jenis slot khusus yang Anda harapkan akan dikenali. Tambahkan semua nilai slot yang mungkin untuk jenis slot khusus, tidak peduli seberapa umum atau langka nilai slotnya.
- Aktifkan opsi hanya jika jenis slot khusus berisi daftar nilai katalog atau entitas seperti nama produk atau reksa dana.

- Nonaktifkan opsi jika jenis slot digunakan untuk menangkap frasa umum seperti “ya,” “tidak,” “Saya tidak tahu,” “mungkin,” atau kata-kata umum seperti “satu,” “dua,” “tiga.”
- Batasi jumlah nilai slot dan sinonim hingga 500 atau kurang untuk kinerja terbaik.

Masukkan akronim atau kata lain yang hurufnya harus diucapkan secara individual sebagai huruf tunggal yang dipisahkan oleh titik dan spasi. Jangan gunakan huruf individual kecuali mereka adalah bagian dari frasa, seperti “J. P. Morgan” atau “A.W. S.” Anda dapat menggunakan huruf besar atau kecil untuk mendefinisikan akronim.

Membuat kosakata khusus untuk memunculkan maksud dan slot

Anda dapat menggunakan konsol Amazon Lex V2 untuk membuat dan mengelola kosakata khusus, atau Anda dapat menggunakan operasi Amazon Lex V2 API. Ada 2 cara untuk membuat kosakata khusus melalui konsol:

Konsol

Impor kosakata khusus di konsol:

1. Buka konsol Amazon Lex V2 di https://console.aws.amazon.com/lexv2/rumah
2. Dari daftar bot, pilih bot yang ingin Anda tambahkan kosakata khusus.
3. Pada halaman detail bot, dari bagian Tambahkan bahasa, pilih Lihat bahasa.
4. Dari daftar bahasa, pilih bahasa yang ingin Anda tambahkan kosakata khusus.

Buat kosakata kustom baru langsung melalui konsol:

1. Klik Buat di bagian Kosakata Kustom pada halaman detail bahasa. Ini akan membuka jendela pengeditan tanpa kosakata khusus.
2. Tambahkan input untuk frase, DisplayAs, dan bobot sesuai kebutuhan. Anda selanjutnya dapat melakukan pengeditan sebaris ke item yang ditambahkan dengan memperbarui bidangnya atau menghapusnya dari daftar.
3. Klik Simpan. Harap dicatat: kosakata khusus baru hanya disimpan di bot Anda setelah Anda mengklik Simpan.
4. Anda dapat terus melakukan pengeditan sebaris di halaman ini dan klik Simpan setelah selesai.
5. Halaman ini juga memungkinkan Anda mengimpor, mengekspor, dan menghapus file kosakata khusus dari menu drop-down di kanan atas.

API

Gunakan **`ListCustomVocabularyItems`** API untuk melihat entri kosakata kustom:

1. Gunakan `ListCustomVocabularyItems` operasi untuk melihat entri kosakata kustom. Badan permintaan akan terlihat seperti ini:

```
{
   "maxResults": number,
   "nextToken": "string"
}
```

2. Harap dicatat bahwa `maxResults` dan `nextToken` merupakan bidang opsional untuk badan permintaan.
3. Respons dari `ListCustomVocabularyItems` operasi terlihat seperti ini:

```
{
   "botId": "string",
   "botVersion": "string",
   "localeId": "string",
   "customVocabularyItems": [
        {
            "itemId": "string",
            "phrase": "string",
            "weight": number,
            "displayAs": "string"
        }
   ]
}
```

Gunakan **`BatchCreateCustomVocabularyItem`** API untuk membuat entri kosakata kustom baru:

1. Jika lokal bot Anda belum memiliki kosakata khusus yang dibuat, ikuti langkah-langkah untuk menggunakan StartImportuntuk membuat kosakata khusus.
2. Setelah kosakata khusus dibuat, gunakan `BatchCreateCustomVocabularyItem` operasi untuk membuat entri kosakata khusus baru. Badan permintaan akan terlihat seperti ini:

```
{
   "customVocabularyItemList": [
        {
```

```
            "phrase": "string",
            "weight": number,
            "displayAs": "string"
        }
    ]
}
```

3. Harap dicatat bahwa `weight` dan `displayAs` merupakan bidang opsional untuk badan permintaan.
4. Tanggapan dari `BatchCreateCustomVocabularyItem` akan terlihat seperti ini:

```
{
    "botId": "string",
    "botVersion": "string",
    "localeId": "string",
    "errors": [
        {
            "itemId": "string",
            "errorMessage": "string",
            "errorCode": "string"
        }
    ],
    "resources": [
        {
            "itemId": "string",
            "phrase": "string",
            "weight": number,
            "displayAs": "string"
        }
    ]
}
```

5. Karena ini adalah operasi batch, permintaan tidak akan gagal jika salah satu item gagal dibuat. Daftar kesalahan akan berisi informasi tentang mengapa operasi gagal untuk entri tertentu. Daftar sumber daya akan berisi semua entri yang berhasil dibuat.
6. Untuk`BatchCreateCustomVocabularyItem`, Anda dapat mengharapkan melihat jenis kesalahan ini:

   - `RESOURCE_DOES_NOT_EXIST`: Kosakata khusus tidak ada. Ikuti langkah-langkah untuk membuat kosakata khusus sebelum memanggil operasi ini.
   - `DUPLICATE_INPUT`: Daftar input berisi frase duplikat.

- `RESOURCE_ALREADY_EXISTS`: Frasa yang diberikan untuk entri sudah ada dalam kosakata kustom Anda.
- `INTERNAL_SERVER_FAILURE`: Ada kesalahan di backend saat memproses permintaan Anda. Ini mungkin mengindikasikan pemadaman layanan atau masalah lain.

Gunakan **`BatchDeleteCustomVocabularyItem`** API untuk menghapus entri kosakata kustom yang ada:

1. Jika lokal bot Anda belum memiliki kosakata khusus yang dibuat, ikuti langkah-langkah untuk Gunakan untuk membuat kosakata khusus StartImportuntuk membuatnya.
2. Setelah kosakata kustom dibuat, gunakan `BatchDeleteCustomVocabularyItem` operasi untuk menghapus entri kosakata kustom yang ada. Badan permintaan akan terlihat seperti ini:

```
{
   "customVocabularyItemList": [
        {
            "itemId": "string"
        }
   ]
}
```

3. Tanggapan dari `BatchDeleteCustomVocabularyItem` akan terlihat seperti ini:

```
{
    "botId": "string",
    "botVersion": "string",
    "localeId": "string",
    "errors": [
        {
            "itemId": "string",
            "errorMessage": "string",
            "errorCode": "string"
        }
    ],
    "resources": [
        {
            "itemId": "string",
            "phrase": "string",
            "weight": number,
            "displayAs": "string"
```

```
        }
    ]
}
```

4. Karena ini adalah operasi batch, permintaan tidak akan gagal jika salah satu item gagal dihapus. Daftar kesalahan akan berisi informasi tentang mengapa operasi gagal untuk entri tertentu. Daftar sumber daya akan berisi semua entri yang berhasil dihapus.
5. Untuk`BatchDeleteCustomVocabularyItem`, Anda dapat mengharapkan melihat jenis kesalahan ini:
   - `RESOURCE_DOES_NOT_EXIST`: Entri kosakata khusus yang Anda coba hapus tidak ada.
   - `INTERNAL_SERVER_FAILURE`: Ada kesalahan di backend saat memproses permintaan Anda. Ini mungkin mengindikasikan pemadaman layanan atau masalah lain.

Gunakan **`BatchUpdateCustomVocabularyItem`** API untuk memperbarui entri kosakata kustom yang ada:

1. Jika lokal bot Anda belum memiliki kosakata khusus yang dibuat, ikuti langkah-langkah untuk Gunakan untuk membuat kosakata khusus StartImportuntuk membuat kosakata khusus.
2. Setelah kosakata kustom dibuat, gunakan `BatchUpdateCustomVocabularyItem` operasi untuk memperbarui entri kosakata kustom yang ada. Badan permintaan akan terlihat seperti ini:

```
{
   "customVocabularyItemList": [
        {
            "itemId": "string",
            "phrase": "string",
            "weight": number,
            "displayAs": "string"
        }
   ]
}
```

3. Harap dicatat bahwa `weight` dan `displayAs` merupakan bidang opsional untuk badan permintaan.
4. Tanggapan dari `BatchUpdateCustomVocabularyItem` akan terlihat seperti ini:

```
{
    "botId": "string",
```

```
    "botVersion": "string",
    "localeId": "string",
    "errors": [
        {
            "itemId": "string",
            "errorMessage": "string",
            "errorCode": "string"
        }
    ],
    "resources": [
        {
            "itemId": "string",
            "phrase": "string",
            "weight": number,
            "displayAs": "string"
        }
    ]
}
```

5. Karena ini adalah operasi batch, permintaan tidak akan gagal jika salah satu item gagal dihapus. Daftar kesalahan akan berisi informasi tentang mengapa operasi gagal untuk entri tertentu. Daftar sumber daya akan berisi semua entri yang berhasil diperbarui.
6. Untuk`BatchUpdateCustomVocabularyItem`, Anda dapat mengharapkan melihat jenis kesalahan ini:

   - `RESOURCE_DOES_NOT_EXIST`: Entri kosakata khusus yang Anda coba perbarui tidak ada.
   - `DUPLICATE_INPUT`: Daftar input berisi duplikat ItemIds.
   - `RESOURCE_ALREADY_EXISTS`: Frasa yang diberikan untuk entri sudah ada dalam kosakata kustom Anda.
   - `INTERNAL_SERVER_FAILURE`: Ada kesalahan di backend saat memproses permintaan Anda. Ini mungkin mengindikasikan pemadaman layanan atau masalah lain.

Membuat file kosakata khusus

File kosakata khusus adalah daftar nilai yang dipisahkan tab yang berisi frasa untuk dikenali, bobot untuk memberikan dorongan, dan `displayAs` bidang yang akan menggantikan frasa dalam transkrip ucapan. Frasa dengan nilai dorongan yang lebih tinggi lebih mungkin digunakan saat muncul di input audio.

File kosakata khusus harus diberi nama**`CustomVocabulary.tsv`**, dan harus dikompresi dalam file zip sebelum dapat diimpor. File zip harus berukuran kurang dari 300 MB. Jumlah maksimum frasa dalam kosakata khusus adalah 500.

- frase 1-4 kata yang harus dikenali. Pisahkan kata-kata dalam frasa dengan spasi. Anda tidak dapat memiliki frase duplikat dalam file. Bidang frasa diperlukan.
- bobot — Sejauh mana pengenalan frasa ditingkatkan. Nilainya adalah bilangan bulat 0, 1, 2, atau 3. Jika Anda tidak menentukan bobot, nilai defaultnya adalah 1. Tentukan bobot berdasarkan seberapa sering kata tersebut tidak dikenali dalam transkripsi dan seberapa jarang kata tersebut dalam input. Bobot 0 berarti tidak ada peningkatan yang akan diterapkan dan entri hanya akan digunakan untuk melakukan penggantian menggunakan bidang. `displayAs`
- DisplayAs — Mendefinisikan bagaimana Anda ingin frasa Anda terlihat dalam output transkripsi Anda. Ini adalah bidang opsional dalam kosakata khusus.

File kosakata khusus harus berisi baris header dengan tajuk “frase,” “berat,” dan “DisplayAs”. Header dapat dalam urutan apa pun, tetapi harus mengikuti nomenklatur di atas.

Contoh berikut adalah file kosakata khusus. Karakter tab yang diperlukan untuk memisahkan frasa, berat, dan DisplayAs diwakili oleh teks “[TAB]”. Jika Anda menggunakan contoh ini, ganti teks dengan karakter tab.

```
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
```

Mengkonfigurasi sensitivitas deteksi aktivitas suara

Voice Activity Detection (VAD) adalah teknologi yang menentukan kapan ucapan hadir dalam sinyal audio. Amazon Lex V2 menggunakan VAD untuk mengoptimalkan akurasi pengenalan suara dengan membedakan antara suara yang sebenarnya dan kebisingan latar belakang. Anda dapat mengonfigurasi tingkat sensitivitas VAD untuk meningkatkan kinerja bot Anda di lingkungan akustik yang berbeda.

Memahami tingkat sensitivitas VAD

Amazon Lex V2 menyediakan tiga tingkat sensitivitas VAD yang dapat Anda konfigurasi untuk lokal bot Anda:

Default

Tingkat sensitivitas standar yang cocok untuk sebagian besar lingkungan dengan tingkat kebisingan latar belakang yang khas. Ini adalah pengaturan yang disarankan untuk kasus penggunaan umum.

HighNoiseTolerance

Peningkatan toleransi untuk kebisingan latar belakang sedang. Gunakan pengaturan ini saat bot Anda beroperasi di lingkungan dengan tingkat kebisingan yang konsisten namun moderat, seperti kantor yang sibuk atau lingkungan ritel.

MaximumNoiseTolerance

Toleransi maksimum untuk kebisingan latar belakang tingkat tinggi. Gunakan pengaturan ini untuk lingkungan yang sangat bising seperti pusat panggilan, lantai manufaktur, atau lokasi luar ruangan dengan kebisingan sekitar yang signifikan.

> **Note**
>
> Tingkat toleransi kebisingan yang lebih tinggi dapat mengakibatkan sistem menjadi lebih permisif tentang apa yang dianggapnya sebagai ucapan, yang berpotensi menyebabkan negatif palsu di lingkungan yang sangat tenang. Pilih tingkat sensitivitas yang paling sesuai dengan lingkungan akustik yang Anda harapkan.

Mengkonfigurasi sensitivitas VAD

Anda dapat mengonfigurasi sensitivitas VAD saat membuat atau memperbarui lokal bot menggunakan konsol Amazon Lex V2, AWS CLI, atau. SDKs

Menggunakan konsol Amazon Lex V2

Untuk mengkonfigurasi sensitivitas VAD di konsol

1. Buka konsol Amazon Lex V2 di https://console.aws.amazon.com/lexv2/.

2. Pilih bot Anda dari daftar.
3. Di panel navigasi kiri, pilih bahasa Bot.
4. Pilih bahasa yang ingin Anda konfigurasikan, atau pilih Tambahkan bahasa untuk menambahkan bahasa baru.
5. Di bagian Sensitivitas deteksi ucapan, pilih salah satu opsi berikut:
   - Default - Sensitivitas standar untuk lingkungan tipikal
   - Toleransi kebisingan tinggi - Untuk lingkungan yang cukup bising
   - Toleransi kebisingan maksimum - Untuk lingkungan yang sangat bising
6. Pilih Simpan untuk menerapkan perubahan.

Menggunakan AWS CLI atau SDKs

Anda dapat mengatur sensitivitas VAD menggunakan `speechDetectionSensitivity` parameter dalam operasi API berikut:

- `CreateBotLocale`- Atur sensitivitas VAD saat membuat lokal bot baru
- `UpdateBotLocale`- Ubah sensitivitas VAD untuk lokal bot yang ada
- `DescribeBotLocale`- Lihat pengaturan sensitivitas VAD saat ini

Example Menyetel sensitivitas VAD dengan AWS CLI

```
aws lexv2-models create-bot-locale \
    --bot-id "AIDACKCEVSQ6C2EXAMPLE" \
    --bot-version "DRAFT" \
    --locale-id "en_US" \
    --nlu-intent-confidence-threshold 0.40 \
    --speech-detection-sensitivity "HighNoiseTolerance"
```

Praktik terbaik untuk konfigurasi VAD

- Uji di lingkungan target Anda - Konfigurasikan sensitivitas VAD berdasarkan kondisi akustik akustik aktual di mana bot Anda akan digunakan.
- Mulai dengan Default - Mulailah dengan pengaturan Default dan sesuaikan berdasarkan pengujian kinerja dan umpan balik pengguna.

- Pantau kinerja - Gunakan analisis Amazon Lex V2 dan log percakapan untuk memantau akurasi pengenalan suara dan menyesuaikan sensitivitas VAD sesuai kebutuhan.
- Pertimbangkan kasus penggunaan - Tingkat sensitivitas yang lebih tinggi bermanfaat untuk lingkungan yang bising tetapi mungkin tidak diperlukan untuk lingkungan yang terkendali seperti pusat layanan pelanggan dengan headset.

Menggunakan diarisasi speaker untuk fokus pada pembicara utama

Diarisasi speaker membuat bot Anda tetap berada di speaker utama (paling keras) selama percakapan suara. Ketika suara lain atau obrolan latar belakang mencapai mikrofon, bot Anda tetap fokus pada speaker utama alih-alih bereaksi terhadap suara di sekitarnya. Contohnya termasuk televisi di kamar sebelah, percakapan sampingan di kantor terbuka, atau seseorang yang berbicara di dekat penelepon. Hasilnya adalah percakapan suara yang lebih lancar, dengan lebih sedikit putaran yang dimulai oleh suara latar belakang dan lebih sedikit permintaan yang terganggu oleh mereka.

> **Note**
>
> Diarisasi speaker berlaku untuk streaming percakapan suara yang menggunakan StartConversationoperasi, seperti panggilan telepon. Ini membutuhkan audio saluran tunggal (mono) 8 kHz. Amazon Lex V2 tidak menerapkan diarisasi speaker ke percakapan teks, atau untuk non-streaming RecognizeTextdan operasi. RecognizeUtterance Untuk informasi selengkapnya tentang API streaming, lihatStreaming percakapan ke bot Amazon Lex V2. Diarisasi speaker tersedia di semua Wilayah AWS di mana Amazon Lex V2 tersedia. Untuk daftar Wilayah, lihat titik akhir dan kuota Amazon Lex V2 di Referensi Umum Amazon Web Services.

Cara kerja diarisasi speaker

Amazon Lex V2 selalu menggunakan deteksi aktivitas suara (VAD) untuk menentukan kapan ucapan hadir dalam aliran audio. VAD menjawab pertanyaan “apakah seseorang berbicara?” tetapi bukan “siapa yang berbicara?” , jadi suara terdekat yang cukup keras dapat memulai belokan. Diarisasi speaker adalah pengaturan opsional yang Anda aktifkan untuk lokal bot. Ini menambahkan pemeriksaan kedua di atas VAD, sehingga giliran mulai dan berakhir berdasarkan aktivitas pembicara utama. Untuk informasi lebih lanjut tentang VAD, lihatthe section called “Mengkonfigurasi sensitivitas deteksi aktivitas suara”.

Saat Anda mengaktifkan diarisasi speaker, Amazon Lex V2 menambahkan sinyal kedua ke keputusan itu:

- Aktivitas speaker - Model diarisasi streaming memberi label speaker yang aktif di setiap bingkai audio 100 milidetik. Ini melacak hingga empat speaker bersamaan.
- Pemilihan speaker utama - Amazon Lex V2 melacak kenyaringan setiap speaker yang terdeteksi dan memperlakukan yang paling keras sebagai speaker utama. Pendekatan ini mengasumsikan bahwa penelepon paling dekat dengan mikrofon.
- Keputusan gabungan - Ketika diarisasi menentukan bahwa pembicara selain pembicara utama berbicara, Amazon Lex V2 memperlakukan audio itu sebagai non-ucapan. Audio ini tidak memulai belokan, mengganggu prompt, atau mencapai pengenalan suara. Ketika pembicara utama berbicara, dan ketika diarisasi tidak mengidentifikasi pembicara latar belakang yang berbeda, Amazon Lex V2 menggunakan keputusan VAD. Gerbang diarisasi saat pengambilan ucapan dimulai dan berakhir. Itu tidak memfilter suara dari audio yang telah ditangkap Amazon Lex V2.

Model membedakan satu pembicara dari yang lain dengan karakteristik suara daripada dengan apa yang dikatakan. Oleh karena itu berfungsi untuk semua lokal dan bahasa bot.

Perubahan pembicara utama

Speaker utama tidak berubah karena suara keras yang singkat. Pembicara yang berbeda menjadi pembicara utama hanya setelah menjadi pembicara yang paling keras dilacak, sementara secara aktif berbicara, untuk beberapa frame berturut-turut. Batuk, bantingan pintu, dan seruan singkat tidak mengambil alih percakapan.

Warm-up periode

Model ini membutuhkan sekitar detik pertama audio untuk membuat pelacakan speaker. Selama periode itu, diarisasi tidak menekan ucapan apa pun, sehingga awal pidato penelepon tidak pernah terputus saat model memanas.

Sebelum penelepon pertama kali berbicara

Pembicara utama adalah pembicara paling keras sejauh ini dalam percakapan. Jika speaker latar belakang adalah suara pertama yang dilacak model, speaker itu menjadi pembicara utama. Penelepon mengambil alih setelah berbicara cukup lama untuk menjadi pembicara yang paling keras dilacak. Diarisasi speaker tidak melindungi percakapan selama jendela ini. Percakapan latar belakang yang keras masih dapat memulai giliran atau mengganggu prompt sebelum penelepon berbicara.

Fallback otomatis

Amazon Lex V2 memonitor diarisasi di setiap percakapan. Jika diarisasi mulai menyaring hampir semua pidato yang dideteksi VAD, Amazon Lex V2 pertama-tama mengatur ulang status diarisasi. Jika kondisi berlanjut, Amazon Lex V2 berhenti menerapkan diarisasi untuk sisa giliran pengguna saat ini dan melanjutkan dengan VAD saja. Amazon Lex V2 mengaktifkan kembali diarisasi pada awal belokan berikutnya. Masalah diarisasi tidak pernah menyebabkan percakapan gagal atau dibatalkan.

Apa yang tidak dilakukan diarisasi speaker

- Itu tidak mengidentifikasi siapa pembicara. Tidak ada pendaftaran, tidak ada nama pembicara, dan tidak ada asosiasi dengan akun pengguna. Bot hanya membedakan antara speaker utama dan speaker lainnya.
- Itu tidak memisahkan audio menjadi saluran per-speaker, dan tidak mengembalikan label speaker dalam transkrip. Satu-satunya efek adalah audio Amazon Lex V2 memperlakukan sebagai pidato penelepon.
- Itu tidak menghapus ucapan latar belakang yang tumpang tindih dari audio yang ditangkap saat speaker utama berbicara. Pidato yang tumpang tindih dengan giliran pembicara utama masih ditranskripsikan. Diarisasi mengakhiri pengambilan segera setelah speaker utama berhenti.
- Itu tidak melacak speaker di seluruh percakapan. Setiap percakapan dimulai tanpa riwayat pembicara.
- Ini tidak digunakan tanpa VAD. Diarisasi menyempurnakan keputusan VAD alih-alih menggantinya.
- Ini tidak berlaku untuk bot yang menggunakan agen AI generatif atau titik akhir inferensi khusus.
- Akurasi dapat menurun ketika lebih dari empat orang berbicara pada saat yang sama.

Mengkonfigurasi diarisasi speaker

Diarisasi speaker adalah pengaturan lokal bot, sehingga Anda dapat mengaktifkannya untuk beberapa bahasa bot dan membiarkannya dimatikan untuk orang lain. Anda dapat mengonfigurasinya saat membuat atau memperbarui lokal bot melalui konsol Amazon Lex V2, atau AWS CLI dan SDK.

Using the console

1. Buka konsol Amazon Lex V2 di https://console.aws.amazon.com/lexv2/.

2. Pilih bot Anda dari daftar.
3. Di panel navigasi kiri, pilih bahasa Bot.
4. Pilih bahasa yang ingin Anda konfigurasikan, atau pilih Tambahkan bahasa untuk menambahkan bahasa baru.
5. Di bagian Diarisasi Speaker, pilih Konfigurasi.
6. Nyalakan Aktifkan diarisasi speaker.
7. Pilih Simpan untuk menerapkan perubahan.
8. Bangun bahasa sehingga perubahan berlaku saat runtime.

Halaman detail bahasa menunjukkan status pengaturan saat ini sebagai Diaktifkan atau Dinonaktifkan.

Using the API

Anda dapat mengatur diarisasi speaker menggunakan `speakerDiarizationSettings` parameter dalam operasi API berikut:

- `CreateBotLocale`- Konfigurasikan diarisasi speaker untuk lokal bot baru.
- `UpdateBotLocale`- Ubah diarisasi speaker untuk lokal bot yang ada.
- `DescribeBotLocale`- Lihat konfigurasi diarisasi speaker saat ini.

Struktur memiliki satu anggota yang diperlukan:

`enabled`

Apakah Amazon Lex V2 membatasi deteksi ucapan ke speaker utama untuk lokal bot ini. Nilai yang valid adalah `true` dan `false`.

Anda juga dapat memasukkan `speakerDiarizationSettings` dalam bot lokal file impor, sehingga pengaturan berjalan dengan bot saat Anda mengekspor dan mengimpornya.

Example Aktifkan diarisasi speaker menggunakan AWS CLI

```
aws lexv2-models update-bot-locale \
    --bot-id "ABCDE12345" \
```

```
    --bot-version "DRAFT" \
    --locale-id "en_US" \
    --nlu-intent-confidence-threshold 0.40 \
    --speaker-diarization-settings '{
        "enabled": true
    }'
```

Example Lihat pengaturan saat ini menggunakan AWS CLI

```
aws lexv2-models describe-bot-locale \
    --bot-id "ABCDE12345" \
    --bot-version "DRAFT" \
    --locale-id "en_US" \
    --query "speakerDiarizationSettings"
```

`speakerDiarizationSettings`dikembalikan oleh`DescribeBotLocale`, dan bukan oleh`ListBotLocales`. Untuk mengaudit beberapa lokal, hubungi `DescribeBotLocale` masing-masing.

> **Note**
>
> Setelah Anda mengubah pengaturan, buat lokal bot dengan `BuildBotLocale` operasi. Jika alias Anda menyajikan versi bot bernomor, buat versi baru dari `DRAFT` versi dan perbarui alias untuk menunjuk ke sana, seperti yang Anda lakukan untuk pengaturan lokal bot lainnya. Untuk informasi selengkapnya, lihat the section called “Pembuatan versi dan alias dengan bot Lex V2 Anda”.

Bagaimana Amazon Lex V2 menafsirkan pengaturan

`{"enabled": true}`

Deteksi ucapan terbatas pada pembicara utama.

`{"enabled": false}`

Diarisasi speaker dimatikan. Amazon Lex V2 menggunakan VAD saja.

Tidak diatur

Dimatikan untuk bot yang belum pernah mengonfigurasi pengaturan.

`UpdateBotLocale`menyimpan nilai yang disimpan saat Anda menghilangkan `speakerDiarizationSettings` permintaan, sehingga pembaruan yang mengubah pengaturan lokal bot lainnya tidak mematikan diarisasi speaker. Untuk menonaktifkan fitur untuk lokal bot yang diaktifkan, kirim `{"enabled": false}` secara eksplisit alih-alih menghilangkan bidang.

Praktik terbaik untuk diarisasi pembicara

- Aktifkan di mana lingkungan penelepon berisik - Manfaatnya berasal dari gangguan oleh pembicara lain, seperti penelepon di rumah dengan televisi menyala, penelepon di speakerphone, kantor terbuka, dan ruang konferensi. Dalam panggilan satu-satu yang tenang, fitur ini dirancang agar tidak memiliki efek yang nyata.
- Harapkan suara paling keras untuk diperlakukan sebagai penelepon - Pembicara utama dipilih oleh kenyaringan. Dua orang mungkin berbagi panggilan, seperti penelepon dan anggota keluarga, atau penelepon dan penerjemah. Dalam hal ini, Amazon Lex V2 memperlakukan orang yang lebih tenang sebagai latar belakang sampai orang itu menjadi pembicara yang lebih keras untuk jangka waktu yang berkelanjutan. Biarkan diarisasi speaker dimatikan untuk alur percakapan yang bergantung pada dua orang yang berbicara secara bergantian.
- Matikan tongkang pada prompt pembukaan yang panjang - Diarisasi speaker tidak dapat membantu sebelum pemanggil berbicara. Jika bot Anda terbuka dengan prompt yang panjang dan penelepon Anda sering dalam percakapan latar belakang yang konstan, atur atribut `x-amz-lex:allow-interrupt:*:*` sesi ke `false` prompt itu. Prompt kemudian diputar, dan diarisasi speaker mengambil alih setelah penelepon berbicara. Untuk informasi selengkapnya, lihat the section called “Cara kerja perilaku interupsi di bot Lex V2”.
- Pertahankan pengaturan interupsi Anda apa adanya - Diarisasi speaker mengurangi gangguan yang disebabkan oleh suara lain. Itu tidak mengubah cara kerja interupsi dari penelepon, jadi setel pengaturan tongkang dan batas waktu secara terpisah.
- Uji dengan audio representatif - Validasi perubahan dengan panggilan uji yang berisi kebisingan latar belakang yang sebenarnya ditemui bot Anda, seperti televisi, percakapan yang tumpang tindih, dan obrolan samping. Bandingkan hasilnya dengan panggilan yang sama dengan pengaturan dimatikan sebelum Anda meluncurkannya ke produksi.

- Meluncurkan satu bahasa pada satu waktu - Karena pengaturan berlaku untuk satu lokal bot, Anda dapat mengaktifkannya untuk satu bahasa, meninjau metrik dan log percakapan Anda, dan kemudian memperluas ke bahasa lain.

Meningkatkan pengenalan nilai slot dengan petunjuk runtime dalam percakapan

Dengan petunjuk runtime, Anda dapat memberi Amazon Lex V2 satu set nilai slot berdasarkan konteks untuk mendapatkan pengenalan yang lebih baik dalam percakapan audio dan resolusi slot yang ditingkatkan. Anda dapat menggunakan petunjuk runtime untuk memberikan daftar frasa saat runtime yang menjadi kandidat untuk resolusi nilai slot.

Misalnya, jika pengguna yang berinteraksi dengan bot reservasi penerbangan sering bepergian ke San Francisco, Jakarta, Seoul, dan Moskow, Anda dapat mengonfigurasi petunjuk runtime dengan daftar keempat kota ini ketika mencari tujuan untuk meningkatkan pengakuan untuk kota-kota yang sering bepergian.

Petunjuk runtime hanya tersedia dalam bahasa Inggris (AS) dan Inggris (Inggris). Mereka dapat digunakan dengan jenis slot berikut:

- Jenis slot khusus
- Amazon.kota
- Amazon.negara
- AMAZON. FirstName
- AMAZON. LastName
- Amazon.state
- AMAZON. StreetName

Dasar-dasar petunjuk runtime

- Petunjuk runtime hanya digunakan saat memunculkan nilai slot dari pengguna.
- Saat Anda menggunakan petunjuk runtime, nilai petunjuk lebih disukai daripada nilai yang serupa. Misalnya, untuk bot pemesanan makanan, Anda dapat mengatur daftar item menu sebagai petunjuk runtime sambil memunculkan item makanan di slot khusus untuk memilih “fillet” daripada “kawan” yang terdengar serupa.

- Jika input pengguna berbeda dari nilai yang diberikan dalam petunjuk runtime, input pengguna asli akan digunakan untuk slot.
- Untuk jenis slot khusus, nilai yang diberikan sebagai petunjuk runtime akan digunakan untuk resolusi slot meskipun itu bukan bagian dari slot khusus selama pembuatan bot.
- Petunjuk runtime hanya didukung untuk input audio 8 kHz. Mereka tersedia dengan integrasi pusat kontak yang didukung oleh Amazon Lex V2. Petunjuk runtime tidak disediakan untuk input audio dari jendela pengujian di konsol Amazon Lex V2 karena menggunakan input audio 16 kHz.

> **Note**
>
> Sebelum Anda dapat menggunakan petunjuk runtime dengan bot yang ada, Anda harus terlebih dahulu membangun kembali bot. Versi bot yang ada tidak mendukung petunjuk runtime. Anda harus membuat versi baru bot untuk menggunakannya.

Anda dapat mengirim petunjuk runtime ke Amazon Lex V2 menggunakan PutSession,, RecognizeTextRecognizeUtterance, atau StartConversationoperasi. Anda juga dapat menambahkan petunjuk runtime menggunakan fungsi Lambda.

Anda dapat mengirim petunjuk runtime di awal percakapan untuk mengonfigurasi petunjuk untuk setiap slot yang digunakan di bot, atau Anda dapat mengirim petunjuk sebagai bagian dari status sesi selama percakapan. `runtimeHints`Atribut memetakan slot ke petunjuk untuk slot itu.

Setelah Anda mengirim petunjuk runtime ke Amazon Lex V2, mereka bertahan untuk setiap putaran percakapan hingga sesi berakhir. Jika Anda mengirim `runtimeHints` struktur null, petunjuk yang ada digunakan. Anda dapat memodifikasi petunjuk dengan:

- Mengirim `runtimeHints` struktur baru ke bot. Isi struktur baru menggantikan yang sudah ada.
- Mengirim `runtimeHints` struktur kosong ke bot. Ini menghapus petunjuk runtime untuk bot.

Menambahkan nilai slot dalam konteks

Tambahkan konteks untuk bot Anda dengan memberikan nilai slot yang diharapkan sebagai petunjuk runtime ketika aplikasi Anda memiliki informasi tentang kemungkinan ucapan pengguna berikutnya. Tambahkan hook kode dialog Lambda ke bot Anda (lihat Mengintegrasikan AWS Lambda fungsi ke bot Amazon Lex V2 Anda untuk informasi lebih lanjut) dan gunakan proposedNextStatebidang di

AWS Lambdaformat acara masukan untuk Lex V2 untuk menentukan petunjuk runtime yang harus Anda sertakan untuk meningkatkan percakapan dengan pengguna.

Misalnya, di aplikasi perbankan Anda dapat membuat daftar nama panggilan akun untuk pengguna tertentu, dan kemudian menggunakan daftar tersebut saat memunculkan akun yang ingin diakses pengguna.

Kirim petunjuk runtime di awal percakapan ketika Anda memiliki konteks untuk membantu bot Anda menafsirkan input pengguna. Misalnya, jika Anda memiliki nomor telepon pengguna, Anda dapat menggunakan informasi ini untuk mencari pengguna sehingga Anda dapat menggunakan `StartConversation` operasi `PutSession` atau untuk meneruskan petunjuk nama depan dan belakang ke bot jika Anda meminta nama pengguna untuk memvalidasi kredensialnya.

Selama percakapan, Anda dapat mengumpulkan informasi dari satu nilai slot yang dapat membantu dengan nilai slot lain. Misalnya, dalam aplikasi perawatan mobil ketika Anda memiliki nomor akun pengguna, Anda dapat melakukan pencarian untuk menemukan mobil yang dimiliki pelanggan dan meneruskannya sebagai petunjuk ke slot lain.

Masukkan akronim, atau kata lain yang hurufnya harus diucapkan satu per satu, sebagai huruf tunggal yang dipisahkan oleh titik dan spasi. Jangan gunakan huruf individual kecuali mereka adalah bagian dari frasa, seperti “J. P. Morgan” atau “A.W.S”. Anda dapat menggunakan huruf besar atau kecil untuk mendefinisikan akronim.

Menambahkan petunjuk ke slot

Untuk menambahkan petunjuk runtime ke slot, Anda menggunakan `runtimeHints` struktur yang merupakan bagian dari struktur. `sessionState` Berikut ini adalah contoh `runtimeHints` strukturnya. Ini memberikan petunjuk untuk dua slot, "FirstName" dan "LastName" untuk maksud MakeAppointment "".

```
{
    "sessionState": {
        "intent": {},
        "activeContexts": [],
        "dialogAction": {},
        "originatingRequestId": {},
        "sessionAttributes": {},
        "runtimeHints": {
            "slotHints": {
                "MakeAppointment": {
                    "FirstName": {
```

```
                        "runtimeHintValues": [
                            {
                                "phrase": "John"
                            },
                            {
                                "phrase": "Mary"
                            }
                        ]
                    },
                    "LastName": {
                        "runtimeHintValues": [
                            {
                                "phrase": "Stiles"
                            },
                            {
                                "phrase": "Major"
                            }
                        ]
                    }
                }
            }
        }
    }
}
```

Anda juga dapat menggunakan fungsi Lambda untuk menambahkan petunjuk runtime selama percakapan. Untuk menambahkan petunjuk runtime, Anda menambahkan `runtimeHints` struktur ke status sesi respons yang dikirimkan fungsi Lambda Anda ke Amazon Lex V2. Untuk informasi selengkapnya, lihat AWS Lambdaformat respons untuk Lex V2.

Anda harus menentukan valid `intentName` dan `slotName` dalam permintaan, jika tidak Amazon Lex V2 mengembalikan kesalahan runtime.

Menangkap nilai slot dengan gaya ejaan selama percakapan

Amazon Lex V2 menyediakan slot bawaan untuk menangkap informasi khusus pengguna seperti nama depan, nama belakang, alamat email, atau pengidentifikasi alfanumerik. Misalnya, Anda dapat menggunakan `AMAZON.LastName` slot untuk menangkap nama keluarga seperti “Jackson” atau “Garcia.” Namun, Amazon Lex V2 mungkin bingung dengan nama keluarga yang sulit diucapkan atau yang tidak umum di suatu lokal, seperti “Xiulan.” Untuk menangkap nama-nama tersebut, Anda dapat meminta pengguna untuk memberikan masukan dalam ej aan demi huruf atau ejaan dengan gaya kata.

Amazon Lex V2 menyediakan tiga gaya elicitasi slot untuk Anda gunakan. Saat Anda mengatur gaya elicitasi slot, itu mengubah cara Amazon Lex V2 menafsirkan input dari pengguna.

Eja demi huruf — Dengan gaya ini, Anda dapat menginstruksikan bot untuk mendengarkan ejaan alih-alih seluruh frasa. Misalnya, untuk menangkap nama belakang seperti “Xiulan,” Anda dapat memberi tahu pengguna untuk mengeja nama belakang mereka satu huruf pada satu waktu. Bot akan menangkap ejaan dan menyelesaikan huruf menjadi kata. Misalnya, jika pengguna mengatakan “x i u l a n,” bot menangkap nama belakang sebagai “xiulan.”

Eja dengan kata — Dalam percakapan suara, terutama menggunakan telepon, ada beberapa huruf, seperti “t,” “b,” “p”, yang terdengar mirip. Saat menangkap nilai alfanumerik atau nama ejaan menghasilkan nilai yang salah, Anda dapat meminta pengguna untuk memberikan kata pengidentifikasi bersama huruf. Misalnya, jika respons suara untuk permintaan ID pemesanan adalah “abp123,” bot Anda mungkin mengenali frasa “ab b 123" sebagai gantinya. Jika ini adalah nilai yang salah, Anda dapat meminta pengguna untuk memberikan input sebagai “a seperti di alpha b seperti pada boy p seperti pada peter one two three.” Bot akan menyelesaikan input ke “abp123.”

Saat menggunakan ejaan demi kata, Anda dapat menggunakan format berikut:

Untuk bahasa Inggris (AS), Inggris (Inggris), dan Inggris (Australia):

- “seperti di” (seperti pada apel)
- “untuk” (a untuk apel)
- “seperti” (seperti apel)

Untuk bahasa Jerman (de-DE):

- “siapa” (p seperti pizza)
- “for” (p untuk pizza)

Untuk bahasa Prancis (fr-FR):

- “seperti” (p sebagai paris)
- “untuk” (p untuk paris)

Untuk bahasa Spanyol (es-419, es-US, es-ES):

- “seperti” (atau sebagai serikat)

- “de” (u de union)

Default — Ini adalah gaya alami penangkapan slot menggunakan pengucapan kata. Misalnya, dapat menangkap nama-nama seperti “John Stiles” secara alami. Jika gaya elicitasi slot tidak ditentukan, bot menggunakan gaya default. Untuk tipe slot `AMAZON.UKPostal` kode `AMAZON.AlphaNumeric` dan, gaya default mendukung ejaan dengan input huruf.

Jika nama “Xiulan” diucapkan menggunakan campuran huruf dan kata-kata, seperti “x seperti dalam x-ray i u l seperti di lion a n” gaya elicitasi slot harus diatur ke gaya ejaan demi kata. Gaya ejaan demi huruf tidak akan mengenalinya.

Anda harus membuat antarmuka suara yang menangkap nilai slot dengan gaya percakapan alami untuk pengalaman yang lebih baik. Untuk input yang tidak ditangkap dengan benar menggunakan gaya alami, Anda dapat meminta ulang pengguna dan mengatur gaya elicitasi slot menjadi ejaan demi huruf atau ejaan demi kata.

Anda dapat menggunakan gaya ejaan demi kata dan ejaan demi huruf untuk jenis slot berikut dalam bahasa Inggris (AS), Inggris (Inggris), Inggris (Australia), Jerman (de-DE), Prancis (fr-FR), dan Spanyol (es-419, es-US, es-ES):

- AMAZON.AlphaNumeric
- AMAZON.EmailAddress
- AMAZON.FirstName
- AMAZON.LastName
- AMAZON.UKPostalCode(Inggris (AS), Inggris (Inggris), dan Inggris (Australia) saja)
- Jenis Slot Kustom

Mengaktifkan ejaan

Anda mengaktifkan ejaan demi huruf dan ejaan demi kata saat runtime saat Anda memunculkan slot dari pengguna. Anda dapat mengatur gaya ejaan dengan StartConversation operasi PutSession RecognizeText, RecognizeUtterance,, atau. Anda juga dapat mengaktifkan ejaan demi huruf dan ejaan demi kata menggunakan fungsi Lambda.

Anda mengatur gaya ejaan menggunakan `dialogAction` bidang `sessionState` bidang dalam permintaan salah satu operasi API yang disebutkan di atas atau saat mengonfigurasi respons Lambda (lihat AWS Lambdaformat respons untuk Lex V2 untuk informasi lebih lanjut). Anda hanya

dapat mengatur gaya ketika jenis tindakan dialog adalah `ElicitSlot` dan ketika slot yang akan dimunculkan adalah salah satu jenis slot yang didukung.

Kode JSON berikut menunjukkan `dialogAction` bidang yang disetel untuk menggunakan gaya ejaan demi kata:

```
"dialogAction": {
    "slotElicitationStyle": "SpellByWord",
    "slotToElicit": "BookingId",
    "type": "ElicitSlot"
}
```

Bid `slotElicitationStyle` ang dapat diatur ke`SpellByLetter`,`SpellByWord`, atau`Default`. Jika Anda tidak menentukan nilai, maka nilainya disetel ke`Default`.

> **Note**
>
> Anda tidak dapat mengaktifkan gaya elicitasi ejaan demi huruf atau ejaan demi kata melalui konsol.

Contoh kode menggunakan Lambda dan Lex V2

Mengubah gaya ejaan biasanya dilakukan jika upaya pertama untuk menyelesaikan nilai slot yang tidak berhasil. Contoh kode berikut adalah fungsi Python Lambda yang menggunakan gaya ejaan demi kata pada upaya kedua untuk menyelesaikan slot.

Untuk menggunakan kode contoh, Anda harus memiliki:

- Sebuah bot dengan satu bahasa, Inggris (GB) (en_GB).
- Satu maksud, "CheckAccount" dengan satu contoh ucapan, “Saya ingin memeriksa akun saya”. Pastikan bahwa Gunakan fungsi Lambda untuk inisialisasi dan validasi dipilih di bagian Kode kait dari definisi maksud.
- Maksud harus memiliki satu slot, "PostalCode“, dari tipe `AMAZON.UKPostalCode` bawaan.
- Alias dengan fungsi Lambda yang ditentukan. Lihat informasi yang lebih lengkap di Membuat AWS Lambda fungsi untuk bot Amazon Lex V2 Anda.

```
import json
```

```
import time
import os
import logging

logger = logging.getLogger()
logger.setLevel(logging.DEBUG)

# --- Helpers that build all of the responses ---

def get_slots(intent_request):
    return intent_request['sessionState']['intent']['slots']

def get_session_attributes(intent_request):
    sessionState = intent_request['sessionState']
    if 'sessionAttributes' in sessionState:
        return sessionState['sessionAttributes']
    return {}

def get_slot(intent_request, slotName):
    slots = get_slots(intent_request)
    if slots is not None and slotName in slots and slots[slotName] is not None:
        logger.debug('resolvedValue={}'.format(slots[slotName]['value']
['resolvedValues']))
        return slots[slotName]['value']['resolvedValues']
    else:
        return None

def elicit_slot(session_attributes, intent_request, slots, slot_to_elicit,
 slot_elicitation_style, message):
    return {'sessionState': {'dialogAction': {'type': 'ElicitSlot',
                                              'slotToElicit': slot_to_elicit,
                                              'slotElicitationStyle':
 slot_elicitation_style
                                              },
                             'intent': {'name': intent_request['sessionState']
['intent']['name'],
                                        'slots': slots,
                                        'state': 'InProgress'
                                        },
                             'sessionAttributes': session_attributes,
                             'originatingRequestId': 'REQUESTID'
                             },
            'sessionId': intent_request['sessionId'],
            'messages': [ message ],
```

```
            'requestAttributes': intent_request['requestAttributes']
            if 'requestAttributes' in intent_request else None
            }

def build_validation_result(isvalid, violated_slot, slot_elicitation_style,
 message_content):
    return {'isValid': isvalid,
            'violatedSlot': violated_slot,
            'slotElicitationStyle': slot_elicitation_style,
            'message': {'contentType': 'PlainText',
            'content': message_content}
            }

def GetItemInDatabase(postal_code):
    """
    Perform database check for transcribed postal code. This is a no-op
    check that shows that postal_code can't be found in the database.
    """
    return None

def validate_postal_code(intent_request):

    postal_code = get_slot(intent_request, 'PostalCode')

    if GetItemInDatabase(postal_code) is None:
        return build_validation_result(
            False,
            'PostalCode',
            'SpellByWord',
            "Sorry, I can't find your information. " +
            "To try again, spell out your postal " +
            "code using words, like a as in apple."
        )
    return {'isValid': True}

def check_account(intent_request):
    """
    Performs dialog management and fulfillment for checking an account
    with a postal code. Besides fulfillment, the implementation for this
    intent demonstrates the following:
    1) Use of elicitSlot in slot validation and re-prompting.
    2) Use of sessionAttributes to pass information that can be used to
        guide a conversation.
    """
```

```
    slots = get_slots(intent_request)
    postal_code = get_slot(intent_request, 'PostalCode')
    session_attributes = get_session_attributes(intent_request)

    if intent_request['invocationSource'] == 'DialogCodeHook':
        # Validate the PostalCode slot. If any aren't valid,
        # re-elicit for the value.
        validation_result = validate_postal_code(intent_request)
        if not validation_result['isValid']:
            slots[validation_result['violatedSlot']] = None
            return elicit_slot(
                session_attributes,
                intent_request,
                slots,
                validation_result['violatedSlot'],
                validation_result['slotElicitationStyle'],
                validation_result['message']
            )

        return close(
            intent_request,
            session_attributes,
            'Fulfilled',
            {'contentType': 'PlainText',
             'content': 'Thanks'
             }
        )

def close(intent_request, session_attributes, fulfillment_state, message):
    intent_request['sessionState']['intent']['state'] = fulfillment_state
    return {
        'sessionState': {
            'sessionAttributes': session_attributes,
            'dialogAction': {
                'type': 'Close'
            },
            'intent': intent_request['sessionState']['intent'],
            'originatingRequestId': 'xxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx'
        },
        'messages': [ message ],
        'sessionId': intent_request['sessionId'],
        'requestAttributes': intent_request['requestAttributes'] if 'requestAttributes'
 in intent_request else None
    }
```

```
# --- Intents ---

def dispatch(intent_request):
    """
    Called when the user specifies an intent for this bot.
    """
    intent_name = intent_request['sessionState']['intent']['name']
    response = None

    # Dispatch to your bot's intent handlers
    if intent_name == 'CheckAccount':
        response = check_account(intent_request)

    return response

# --- Main handler ---

def lambda_handler(event, context):
    """
    Route the incoming request based on the intent.

    The JSON body of the request is provided in the event slot.
    """

    # By default, treat the user request as coming from
    # Eastern Standard Time.
    os.environ['TZ'] = 'America/New_York'
    time.tzset()

    logger.debug('event={}'.format(json.dumps(event)))
    response = dispatch(event)
    logger.debug("response={}".format(json.dumps(response)))

    return response
```

Menggunakan pengisi audio untuk meningkatkan respons bot

Pengisi audio memutar audio latar belakang singkat, seperti melodi ringan atau penekanan tombol lembut, selama jeda antara akhir ucapan pengguna dan awal respons bot. Pemrosesan topeng ini menunda dan membuat percakapan suara terasa alami.

> **Note**
>
> Saat peluncuran, pengisi audio tersedia untuk lokal bot yang mendukung interaksi ucapan-ke-ucapan dan telah dikonfigurasi. `unifiedSpeechSettings` Support untuk mode percakapan tambahan akan diluncurkan selama beberapa bulan ke depan.

Jenis pengisi audio yang tersedia

Amazon Lex V2 menyediakan tujuh suara pengisi bawaan, disusun menjadi dua keluarga:

- Melodi - Chipper Chime
- Melodi - Perayapan Penasaran
- Melodi - Riak Meningkat
- Melodi - Ping Pasien
- Melody - Merenungkan Pong
- Mengetik - Tombol Kinetik
- Mengetik - Quiet Qwerty

Gunakan tombol Putar pratinjau audio di konsol Amazon Lex V2 untuk mendengar setiap opsi sebelum Anda menyimpannya ke lokal bot.

Parameter waktu

Anda dapat menyetel tiga parameter waktu untuk mengontrol kapan pengisi audio diputar dan bagaimana transisi ke respons bot:

`startDelayInMilliseconds`

Saatnya menunggu setelah akhir ucapan pengguna sebelum memulai pemutaran pengisi audio. Rentang yang valid adalah `500` `5000` milidetik. Default-nya adalah `1000`.

`minimumPlayDurationInMilliseconds`

Pengisi audio waktu minimum diputar setelah dimulai, bahkan jika respons bot menjadi siap lebih cepat. Rentang yang valid adalah `1000` `5000` milidetik. Default-nya adalah `3000`.

`responseDeliveryDelayInMilliseconds`

Penundaan senyap disisipkan antara akhir pemutaran pengisi audio dan dimulainya respons bot. Rentang yang valid adalah `200` `1000` milidetik. Default-nya adalah `500`.

Mengkonfigurasi pengisi audio

Anda dapat mengonfigurasi pengisi audio saat membuat atau memperbarui lokal bot melalui konsol Amazon Lex V2, perancang AI Percakapan Amazon Connect, atau AWS CLI dan SDK.

Using the console

1. Buka konsol Amazon Lex V2 di https://console.aws.amazon.com/lexv2/.
2. Pilih bot Anda dari daftar.
3. Di panel navigasi kiri, pilih bahasa Bot.
4. Pilih bahasa yang ingin Anda konfigurasikan, atau pilih Tambahkan bahasa untuk menambahkan bahasa baru.
5. Di bagian Pengisi Audio, pilih Aktifkan pengisi audio.
6. Pilih jenis Audio dari melodi atau opsi pengetikan. Gunakan Putar pratinjau audio untuk mendengar pengisi yang dipilih.
7. Sesuaikan penggeser waktu untuk Start delay, Durasi bermain minimum, dan buffer Response sesuai kebutuhan.
8. Pilih Simpan untuk menerapkan perubahan.

Using the Amazon Connect Conversational AI designer

1. Buka situs web admin Amazon Connect dan arahkan ke desainer Conversational AI untuk bot Anda.
2. Buka bahasa (lokal) yang ingin Anda konfigurasikan.
3. Di bagian Pengisi Audio, aktifkan pengisi audio dan pilih jenis audio.
4. Sesuaikan nilai Start delay, Minimum play duration, dan Response buffer.
5. Simpan perubahan Anda. Perancang menerapkan hal yang sama `audioFillerSettings` pada lokal bot Amazon Lex V2 yang mendasarinya.

Using the API

Anda dapat mengatur pengisi audio menggunakan `audioFillerSettings` parameter dalam operasi API berikut:

- `CreateBotLocale`- Konfigurasikan pengisi audio untuk lokal bot baru.
- `UpdateBotLocale`- Ubah pengisi audio untuk lokal bot yang ada.
- `DescribeBotLocale`- Lihat konfigurasi pengisi audio saat ini.

Example Konfigurasikan pengisi audio menggunakan AWS CLI

```
aws lexv2-models update-bot-locale \
    --bot-id "bot-1234567890abcdef0" \
    --bot-version "DRAFT" \
    --locale-id "en_US" \
    --nlu-intent-confidence-threshold 0.40 \
    --audio-filler-settings '{
        "enabled": true,
        "audioType": "MELODY_CHIPPER_CHIME",
        "startDelayInMilliseconds": 1000,
        "minimumPlayDurationInMilliseconds": 3000,
        "responseDeliveryDelayInMilliseconds": 500
    }'
```

Pengisi audio dengan pesan sementara agen AI

Pengisi audio bekerja bersama pesan sementara agen AI. Ketika agen AI mengirim pesan sementara ke penelepon (misalnya, “Biarkan saya mencarinya untuk Anda”), pengatur waktu tunda mulai diukur dari akhir pesan sementara itu daripada dari pemanggilan asli. Ini mencegah pengisi audio tumpang tindih dengan ucapan agen dan memastikan penundaan pengalaman penelepon diukur dari audio terbaru yang mereka dengar.

Pengisi audio dengan dialog dan kait kode pemenuhan

Pengisi audio juga diputar selama celah pemrosesan yang diperkenalkan oleh kait kode dialog Lambda dan kait kode pemenuhan. Parameter waktu yang sama berlaku, sehingga penelepon

mendengar pengalaman yang konsisten apakah bot Anda mendelegasikan pemrosesan ke agen AI, kait kode, atau keduanya pada giliran yang sama.

Praktik terbaik untuk pengisi audio

- Cocokkan pengisi dengan suara merek Anda. Gunakan pengisi melodi untuk pengalaman konsumen atau ritel, dan mengetik pengisi saat pengguna mengharapkan bot secara aktif mengerjakan suatu tugas.
- Setel penundaan mulai ke profil latensi Anda. Jika sebagian besar respons bot lebih cepat dari`startDelayInMilliseconds`, pengisi jarang akan diputar. Turunkan penundaan untuk beban kerja yang berat latensi dan naikkan untuk bot yang merespons cepat.
- Jaga agar durasi bermain minimum tetap pendek untuk bot cepat. Long `minimumPlayDurationInMilliseconds` on a fast bot menambah latensi yang dirasakan dengan memegang pengisi setelah respons siap.
- Uji dengan lalu lintas yang representatif. Validasi pilihan pengisi dan waktu dalam percakapan realistis sebelum diluncurkan ke produksi.

Memantau kinerja bot di Lex V2

Pemantauan penting untuk menjaga keandalan, ketersediaan, dan kinerja chatbot Amazon Lex V2 Anda. Topik ini menjelaskan penggunaan log percakapan untuk memantau percakapan antara pengguna dan chatbot Anda, menggunakan statistik ucapan untuk menentukan ucapan yang dideteksi dan dilewatkan oleh bot Anda, serta cara menggunakan Amazon Logs dan untuk memantau Amazon Lex V2 CloudWatch . AWS CloudTrail Ini juga menjelaskan metrik runtime dan asosiasi saluran Amazon Lex V2.

Gunakan alat dan metrik ini untuk memahami arah dan tindakan apa yang dapat Anda ambil untuk meningkatkan kinerja bot Anda.

Topik

- Mengukur kinerja bisnis dengan Analytics
- Mengaktifkan log percakapan untuk bot Lex V2 Anda
- Kesalahan logging dengan log kesalahan di Lex V2
- Memantau metrik operasional di Lex V2
- Mengevaluasi kinerja bot Lex V2 dengan Test Workbench

Mengukur kinerja bisnis dengan Analytics

Dengan Analytics, Anda dapat mengevaluasi kinerja bot Anda dengan metrik yang terkait dengan tingkat keberhasilan dan kegagalan interaksi bot Anda dengan pelanggan. Anda juga dapat memvisualisasikan pola arus percakapan antara bot dan pelanggan Anda. Analytics merampingkan pengalaman Anda dengan meringkas metrik ini dalam grafik dan bagan. Analytics menyediakan alat untuk membantu Anda memfilter hasil untuk mengidentifikasi masalah dan masalah yang melibatkan maksud, slot, ucapan, dan percakapan. Anda dapat menggunakan data ini untuk mengulangi dan meningkatkan bot Anda untuk menciptakan pengalaman pelanggan yang lebih baik.

> **Note**
>
> Agar pengguna dapat mengakses Analytics, kebijakan Kebijakan terkelola AWS: AmazonLexFullAccess atau kebijakan khusus yang menyertakan izin API analitik harus dilampirkan ke peran IAM mereka. Lihat Mengelola izin akses untuk analitik detail tentang cara menangani izin pengguna dengan kebijakan khusus. Jika Kebijakan terkelola AWS:

AmazonLexReadOnly dilampirkan ke peran IAM pelanggan, kesalahan akan menampilkan izin yang hilang yang perlu Anda tambahkan ke peran IAM pengguna agar mereka dapat mengakses dasbor Analytics.

Untuk mengakses Analytics

1. Masuk ke AWS Management Console dan buka konsol Amazon Lex V2 di https://console.aws.amazon.com/lexv2/home.
2. Di panel navigasi di bawah Bots, pilih bot yang ingin Anda lihat di analitik.
3. Pilih bagian di bawah Analytics yang ingin Anda lihat.

Topik

- Definisi kunci
- Hasil penyaringan
- Ikhtisar: ringkasan kinerja bot Anda
- Dasbor percakapan: ringkasan percakapan bot Anda
- Dasbor kinerja: ringkasan metrik maksud dan ucapan bot Anda
- Menggunakan API untuk analitik
- Mengelola izin akses untuk analitik

Definisi kunci

Topik ini memberikan definisi kunci yang akan membantu Anda menafsirkan analitik bot Anda. Definisi ini terkait dengan kinerja bot Anda dalam empat konteks: Maksud, Slot, Percakapan, dan Ucapan. Bidang berikut relevan dengan banyak metrik kinerja:

- `state`Bidang `Intent` objek.
- `type`Bidang `dialogAction` objek di dalam SessionStateobjek.

Intent

Amazon Lex V2 mengkategorikan maksud dengan cara berikut:

- Sukses — Bot berhasil memenuhi niat. Salah satu situasi berikut ini benar:

  - Maksudnya `state` adalah `ReadyForFulfillment` dan `type` dari `dialogAction` adalah`Close`.
  - Maksudnya `state` adalah `Fulfilled` dan `type` dari `dialogAction` adalah`Close`.
- Gagal — Bot gagal memenuhi maksud. Negara niat. Salah satu situasi berikut ini benar:
  - `state`Maksudnya adalah `Failed` dan `type` of `dialogAction` is `Close` (misalnya, pengguna menolak prompt konfirmasi).
  - Bot beralih ke `AMAZON.FallbackIntent` sebelum intent selesai.
- Beralih — Bot mengenali maksud yang berbeda dan beralih ke maksud itu, sebelum maksud asli dikategorikan sebagai sukses atau gagal .
- Dropped — Pelanggan tidak merespons sebelum niat dikategorikan sebagai sukses atau gagal.

Slot

Amazon Lex V2 mengkategorikan slot dengan cara berikut:

- Sukses — Bot mengisi slot dan berhasil dialihkan ke slot lain atau langkah konfirmasi.
- Gagal — Bot tidak dapat mengisi slot, bahkan setelah mencapai jumlah maksimum percobaan ulang.
- Dropped — Pelanggan tidak merespons atau beralih ke maksud lain sebelum slot dikategorikan sebagai sukses atau gagal.

Percakapan

Saat pelanggan melakukan panggilan runtime ke Amazon Lex V2, mereka menyediakan `sessionId`dan Amazon Lex V2 menghasilkan file `originatingRequestId`. Jika pelanggan tidak merespons dalam batas waktu Session (`idleSessionTTLInSeconds`) yang Anda tetapkan untuk bot, sesi akan kedaluwarsa. Jika pelanggan kembali ke sesi dengan menggunakan yang sama`sessionId`, Amazon Lex V2 menghasilkan yang baru`originatingRequestId`.

Untuk analitik, percakapan adalah kombinasi unik dari a `sessionId` dan an`originatingRequestId`. Amazon Lex V2 mengkategorikan percakapan dengan cara berikut:

- Sukses — Maksud akhir dalam percakapan dikategorikan sebagai sukses.
- Gagal — Maksud akhir dalam percakapan gagal. Percakapan juga gagal jika Amazon Lex V2 default ke file. AMAZON.FallbackIntent

- Jatuh — Pelanggan tidak merespons sebelum percakapan dikategorikan sebagai sukses atau gagal.

Ucapan

Amazon Lex V2 mengkategorikan ucapan dengan cara berikut:

- Terdeteksi - Amazon Lex V2 mengenali ucapan tersebut sebagai upaya untuk memanggil maksud yang dikonfigurasi untuk bot.
- Tidak terjawab - Amazon Lex V2 tidak mengenali ucapannya.

Hasil penyaringan

Di bagian atas setiap halaman, Anda dapat memfilter hasil untuk analitik bot Anda.

Anda dapat memfilter berdasarkan parameter berikut:

- Waktu — Anda dapat memfilter hasil berdasarkan rentang waktu relatif atau absolut. Saat Anda memilih waktu mulai dan berakhir, Amazon Lex V2 mengambil percakapan yang dimulai setelah waktu mulai dan berakhir sebelum waktu berakhir.
  - Rentang relatif — Pilih 1d untuk melihat hasil dari hari sebelumnya, 1w selama seminggu terakhir, atau 1m selama sebulan terakhir.

    Untuk opsi lainnya, pilih Kustom dan pilih durasi di menu Rentang relatif. Untuk kontrol lebih lanjut atas durasi, pilih Rentang kustom, masukkan angka di bidang Durasi, dan pilih Satuan waktu dari menu tarik-turun.
  - Rentang absolut - Pilih Kustom dan pilih menu Rentang absolut untuk memfilter percakapan dalam rentang waktu yang Anda tentukan. Anda dapat memilih tanggal mulai dan berakhir di kalender atau memasukkannya dalam YYYY/MM/DD format.

> **Note**
>
> Rentang waktu analitik memiliki batasan berikut:
>
> - Tanggal mulai harus dalam 365 hari terakhir dari tanggal saat ini.
> - Tanggal akhir tidak boleh lebih dari 1 bulan setelah tanggal mulai.

- Filter bot — Untuk memfilter berdasarkan lokal, alias, dan versi bot Anda, pilih menu tarik-turun berlabel Semua lokal, Semua alias, dan Semua versi.
- Modalitas - Pilih ikon roda gigi dan pilih menu tarik-turun Modalitas untuk memilih apakah akan menampilkan hasil untuk Pidato atau Teks.
- Saluran - Pilih ikon roda gigi dan pilih menu tarik-turun Saluran untuk memilih saluran yang ingin Anda tampilkan hasilnya. Untuk informasi selengkapnya tentang integrasi saluran, lihat Mengintegrasikan bot Amazon Lex V2 dengan platform perpesanan dan pusat kontak Amazon Connect

Ikhtisar: ringkasan kinerja bot Anda

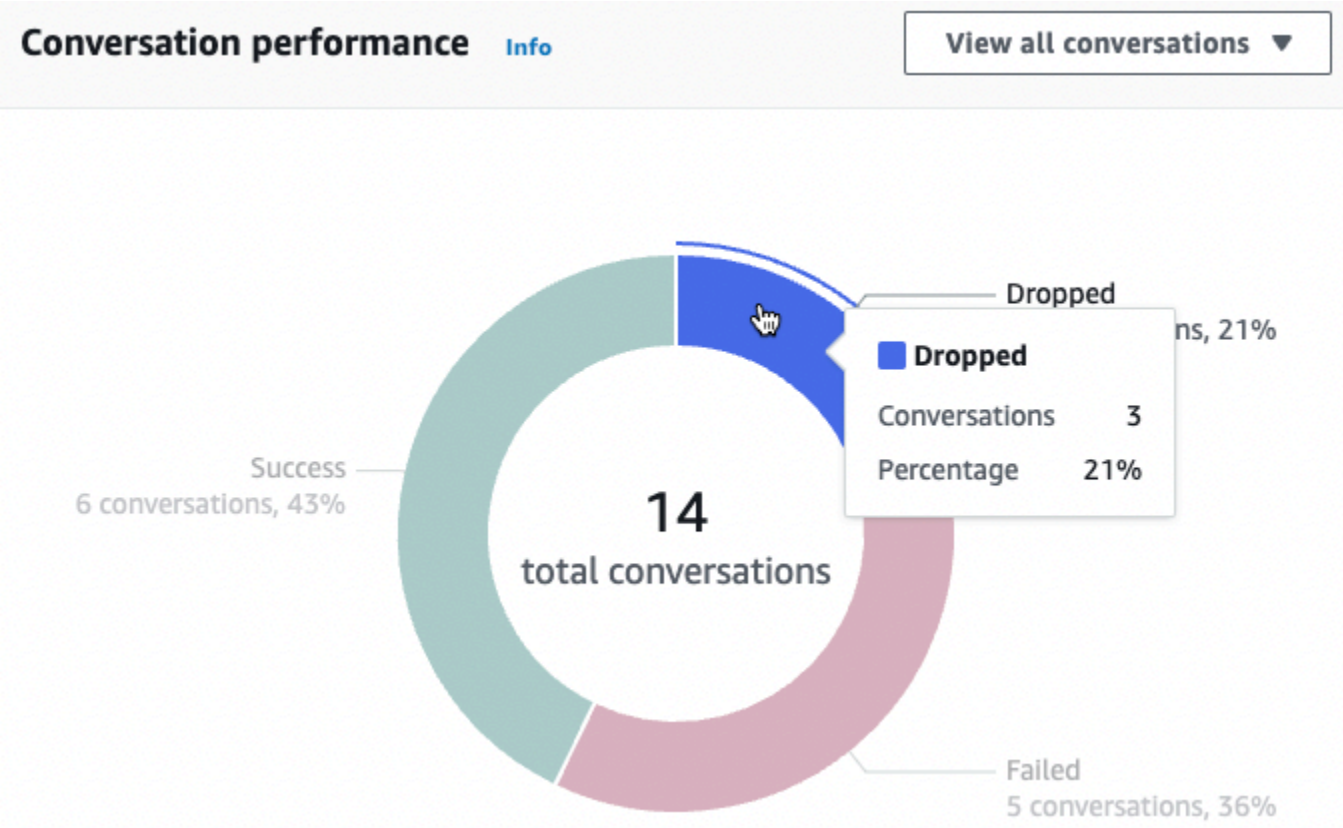
Halaman ikhtisar merangkum kinerja bot Anda pada percakapan, pengenalan ucapan, dan penggunaan maksud. Ikhtisar terdiri dari bagian-bagian berikut:

- Kinerja percakapan
- Tingkat pengakuan ucapan
- Riwayat kinerja percakapan
- Top 5 maksud yang digunakan
- 5 maksud gagal teratas

Kinerja percakapan

Gunakan bagan ini untuk melacak jumlah dan persentase percakapan yang dikategorikan sebagai sukses, gagal, dan turun. Untuk mengakses daftar percakapan, pilih Lihat semua percakapan untuk menampilkan menu tarik-turun. Anda dapat memilih untuk melihat daftar semua percakapan pengguna dengan bot atau memfilter percakapan dengan hasil tertentu (berhasil, gagal, atau terputus). Tautan ini membawa Anda ke subbagian Percakapan di dasbor Percakapan. Untuk informasi selengkapnya, lihat Percakapan.

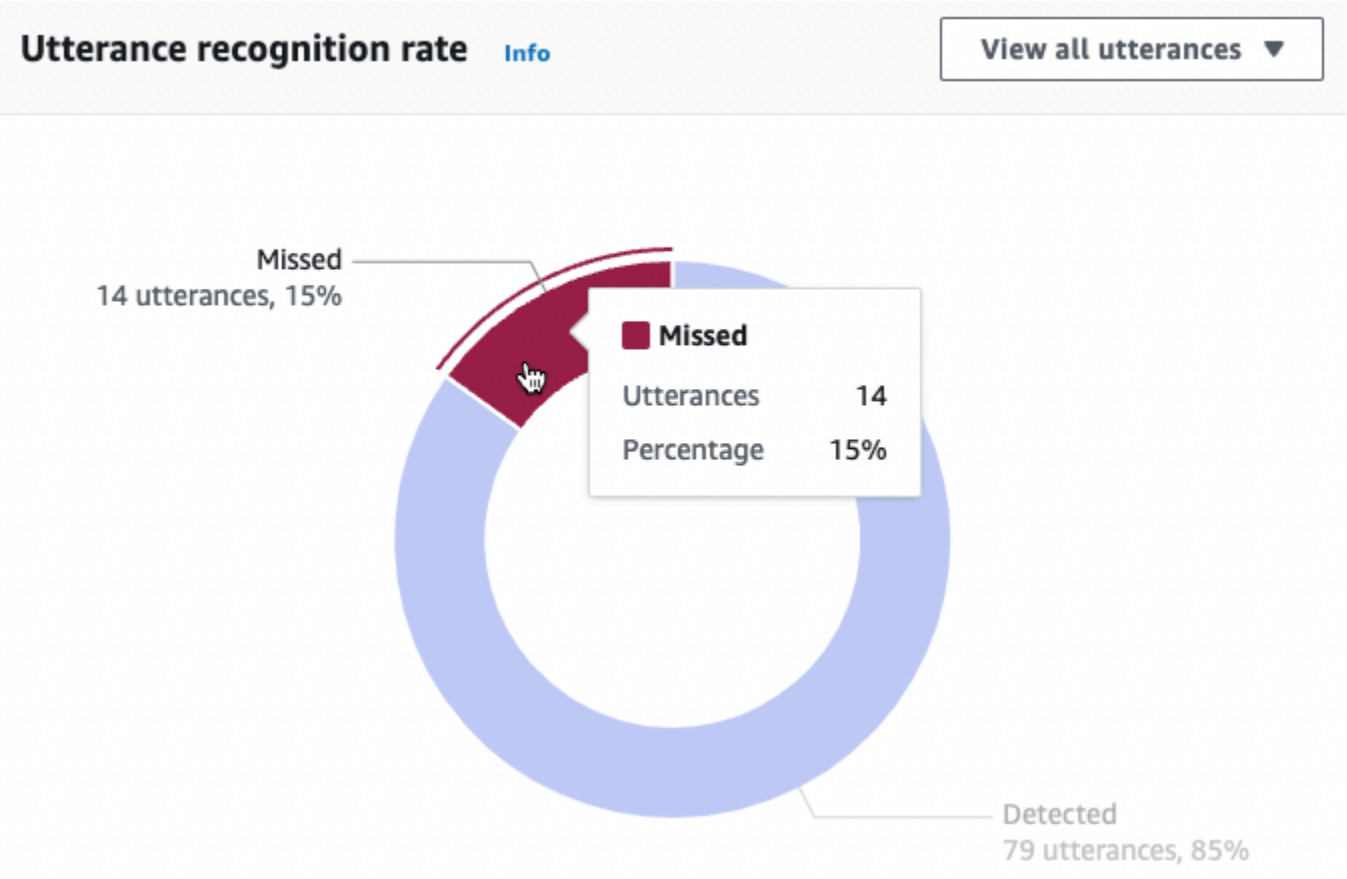
Untuk menampilkan kotak dengan jumlah dan persentase percakapan dengan hasil itu, arahkan kursor ke segmen bagan, seperti pada gambar berikut.



Tingkat pengakuan ucapan

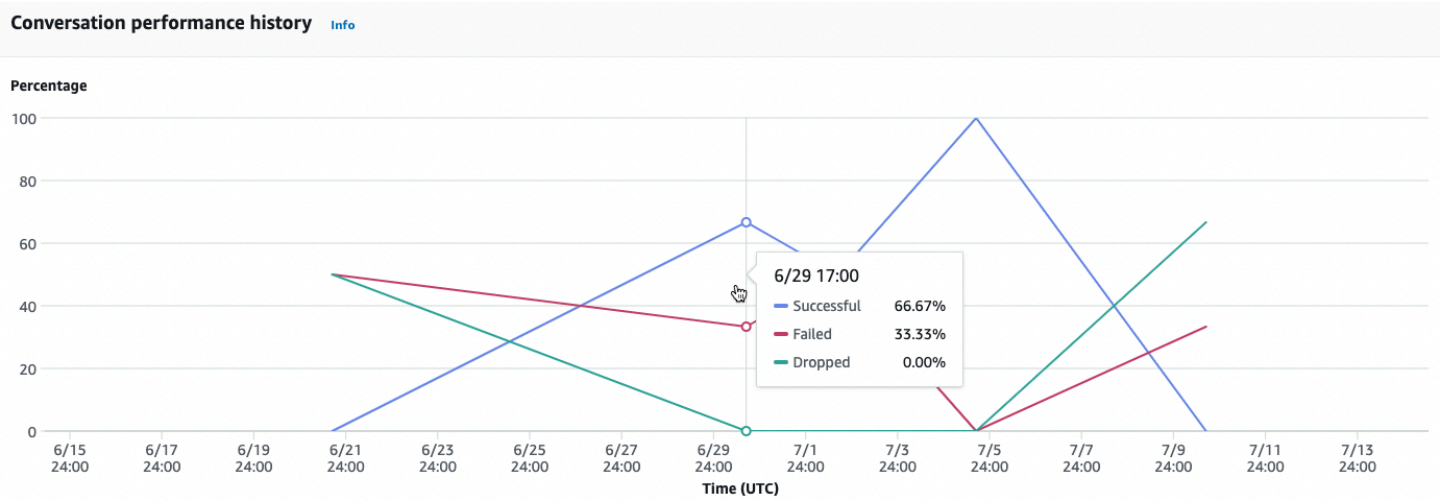
Gunakan bagan ini untuk melacak jumlah dan persentase ucapan yang terdeteksi dan terlewatkan oleh bot Anda. Untuk mengakses daftar ucapan, pilih Lihat ucapan untuk menampilkan menu tarik-turun. Anda dapat memilih untuk melihat daftar semua ucapan pengguna atau memfilter ucapan dengan hasil tertentu (tidak terjawab atau terdeteksi). Tautan ini membawa Anda ke subbagian pengenalan Ucapan dari dasbor Kinerja. Untuk informasi selengkapnya, lihat Lihat Ucapan untuk menavigasi ke. Pengakuan ucapan

Untuk mengungkapkan kotak dengan jumlah dan persentase ucapan, arahkan kursor ke segmen bagan, seperti yang terlihat pada gambar berikut.



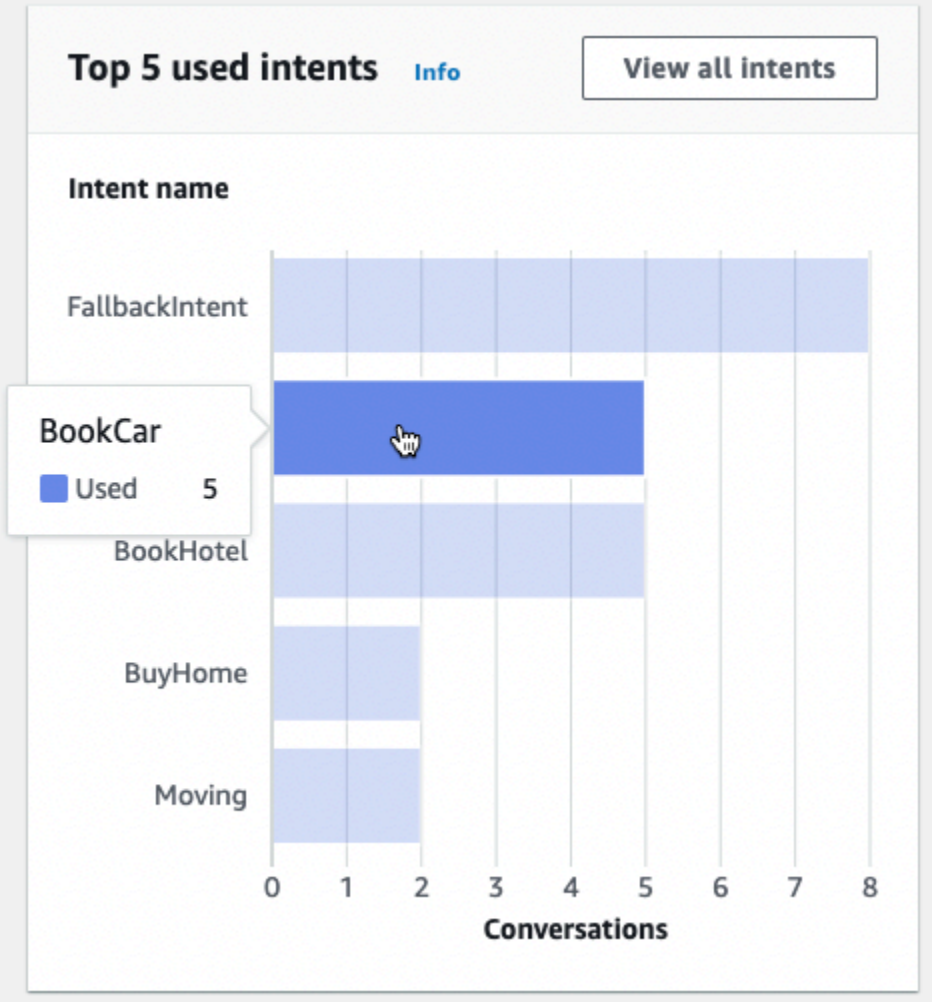
Riwayat kinerja percakapan

Gunakan grafik ini untuk melacak persentase percakapan yang dikategorikan sebagai sukses, gagal, dan turun selama rentang waktu yang Anda tetapkan dalam filter. Untuk melihat persentase percakapan dengan hasil tertentu dalam interval waktu, arahkan kursor ke interval itu, seperti pada gambar berikut.



Top 5 maksud yang digunakan

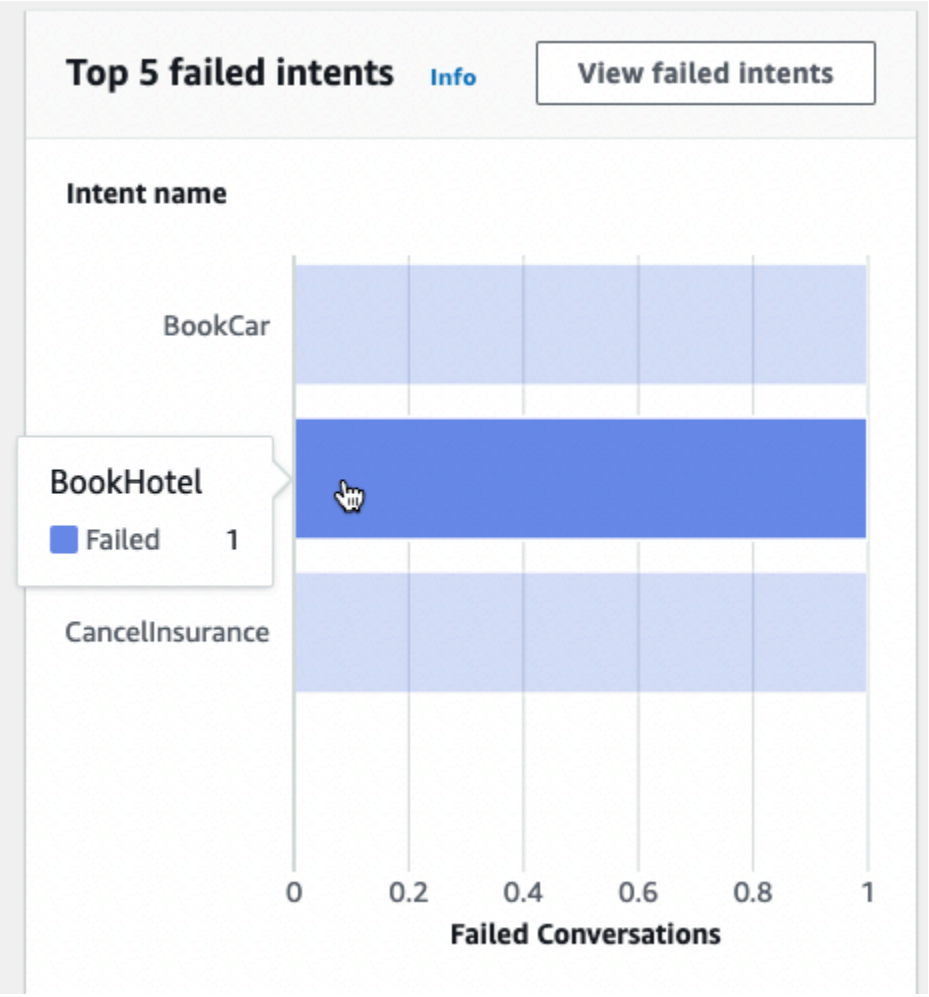
Gunakan bagan ini untuk mengidentifikasi lima maksud teratas yang digunakan pelanggan dengan bot Anda. Arahkan kursor ke bar untuk melihat berapa kali bot Anda mengenali maksud tersebut, seperti pada gambar berikut.



Pilih Lihat semua maksud untuk menavigasi ke subbagian Kinerja Intent dari dasbor Kinerja, tempat Anda dapat melihat metrik kinerja bot Anda dalam memenuhi maksud. Untuk informasi selengkapnya, lihat Kinerja niat.

5 maksud gagal teratas

Gunakan bagan ini untuk mengidentifikasi lima intent teratas yang gagal dipenuhi oleh bot Anda (lihat Intent definisi maksud yang gagal). Arahkan kursor ke bar untuk melihat berapa kali bot Anda gagal memenuhi maksud itu, seperti pada gambar berikut.



Pilih Lihat maksud yang gagal untuk menavigasi ke subbagian kinerja Intent dari dasbor Kinerja, tempat Anda dapat melihat metrik untuk maksud yang gagal dipenuhi oleh bot Anda. Untuk informasi selengkapnya, lihat Kinerja niat.

Dasbor percakapan: ringkasan percakapan bot Anda

Dasbor percakapan memvisualisasikan metrik untuk percakapan pelanggan (lihat Percakapan definisi percakapan) dengan bot Anda.

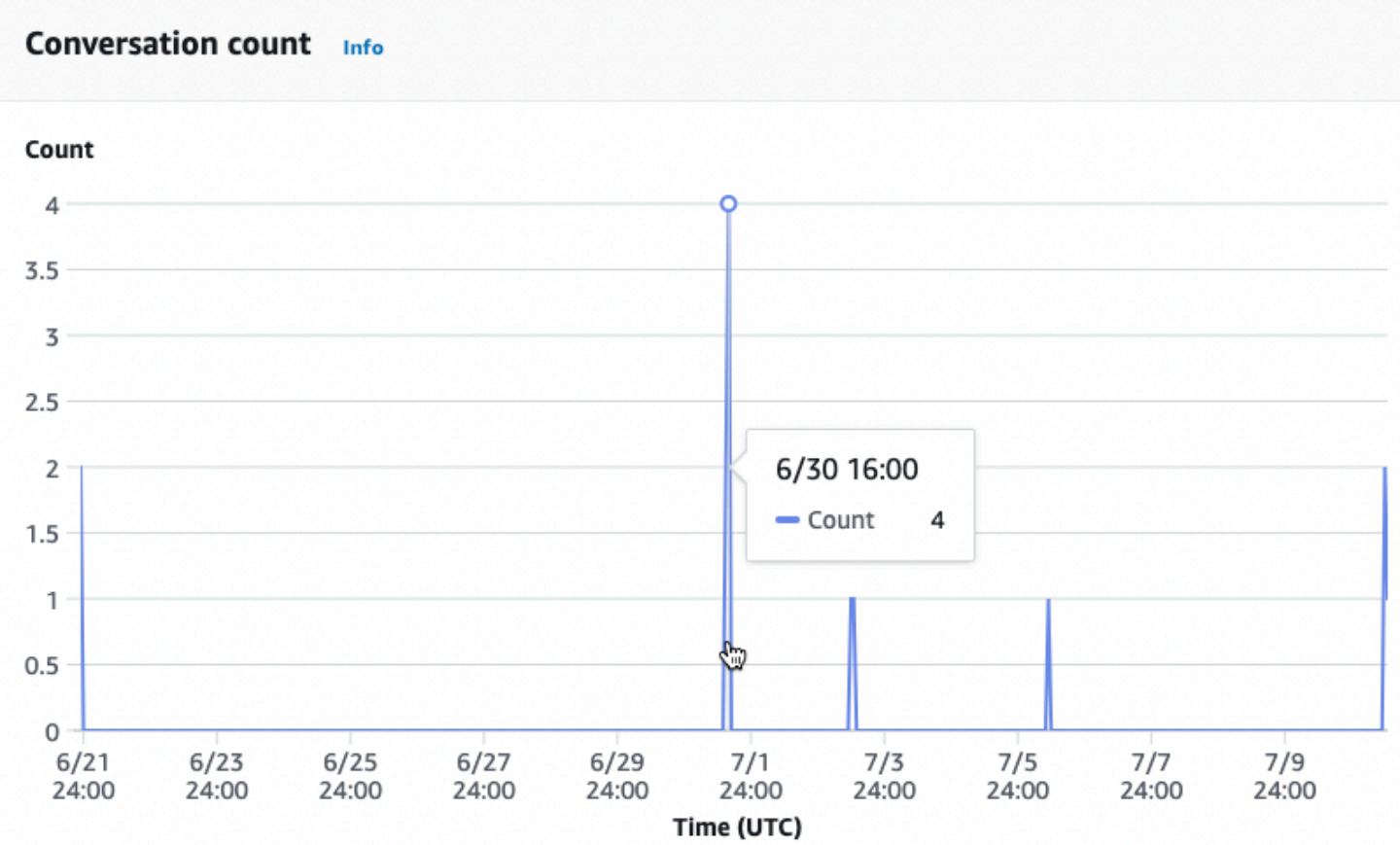
Ringkasan berisi informasi berikut tentang percakapan pengguna dengan bot Anda. Angka-angka dihitung berdasarkan pengaturan filter.

- Total percakapan — Jumlah total percakapan dengan bot.
- Durasi percakapan rata-rata — Waktu rata-rata percakapan pengguna dengan bot dalam hitungan menit dan detik. Formatnya mm: ss.
- Rata-rata putaran per percakapan — Jumlah rata-rata giliran percakapan berlangsung.

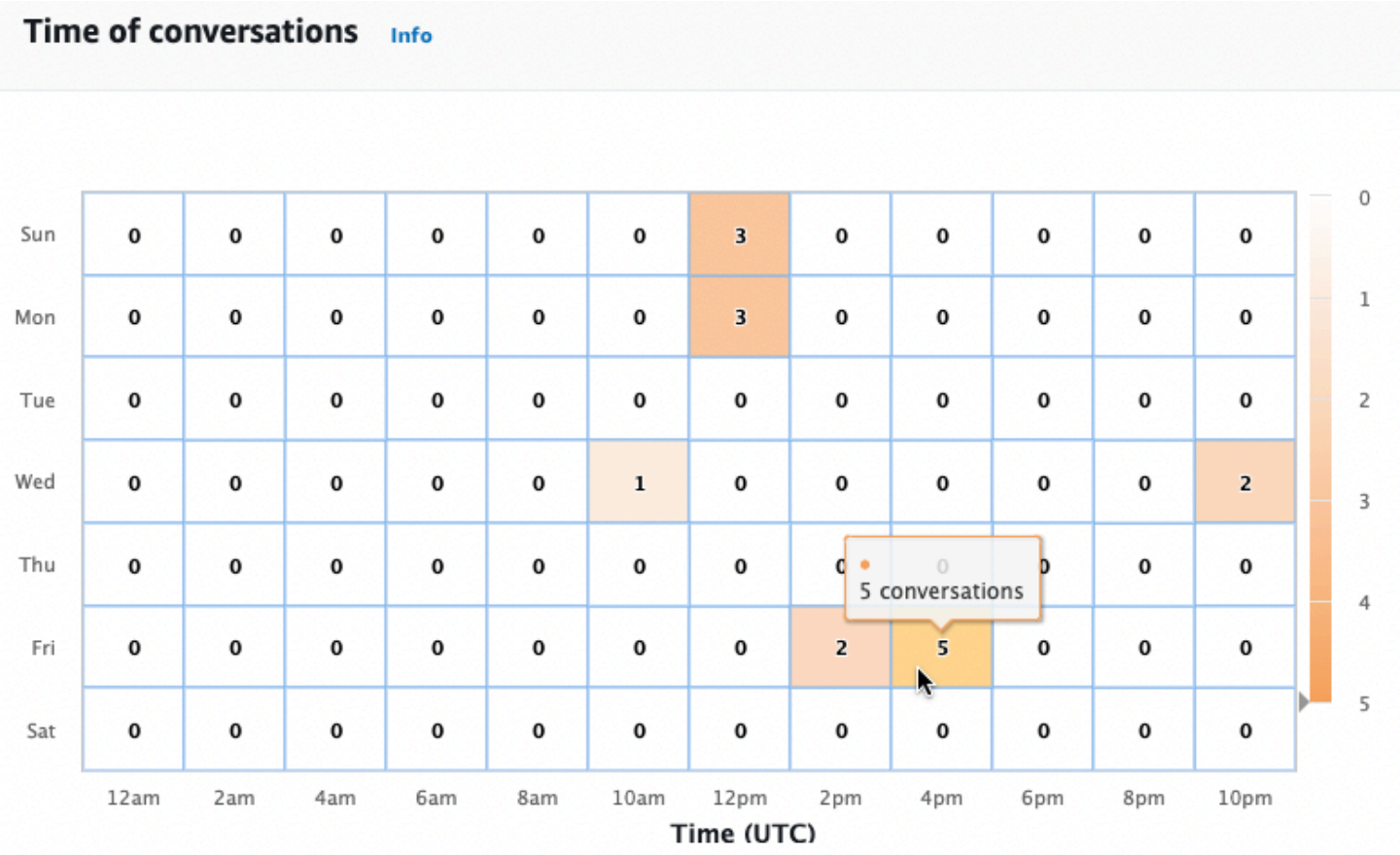
Bagian Hitungan percakapan dan Jumlah pesan masing-masing berisi grafik yang menunjukkan jumlah percakapan dan pesan, masing-masing, selama rentang waktu yang Anda tentukan dalam filter Anda. Arahkan kursor ke segmen waktu untuk melihat jumlah percakapan atau pesan di segmen tersebut. Ukuran segmen waktu tergantung pada rentang waktu yang Anda tentukan:

- Kurang dari 1 minggu — Hitungan ditampilkan untuk setiap jam.
- 1 minggu atau lebih — Hitungan ditampilkan untuk setiap hari.

Lihat contoh perilaku melayang pada gambar berikut.



Bagian Waktu percakapan menyajikan jumlah percakapan yang terjadi antara bot Anda dan pelanggan selama setiap interval dua jam pada setiap hari dalam seminggu, dalam rentang waktu yang Anda tentukan di filter. Sel yang lebih gelap menunjukkan waktu di mana lebih banyak percakapan terjadi. Arahkan kursor ke sel untuk menampilkan jumlah percakapan dalam 2 jam mulai dari slot waktu itu. Misalnya, tindakan pada gambar berikut menunjukkan jumlah percakapan yang terjadi antara pukul 16:00 dan 18:00 UTC.



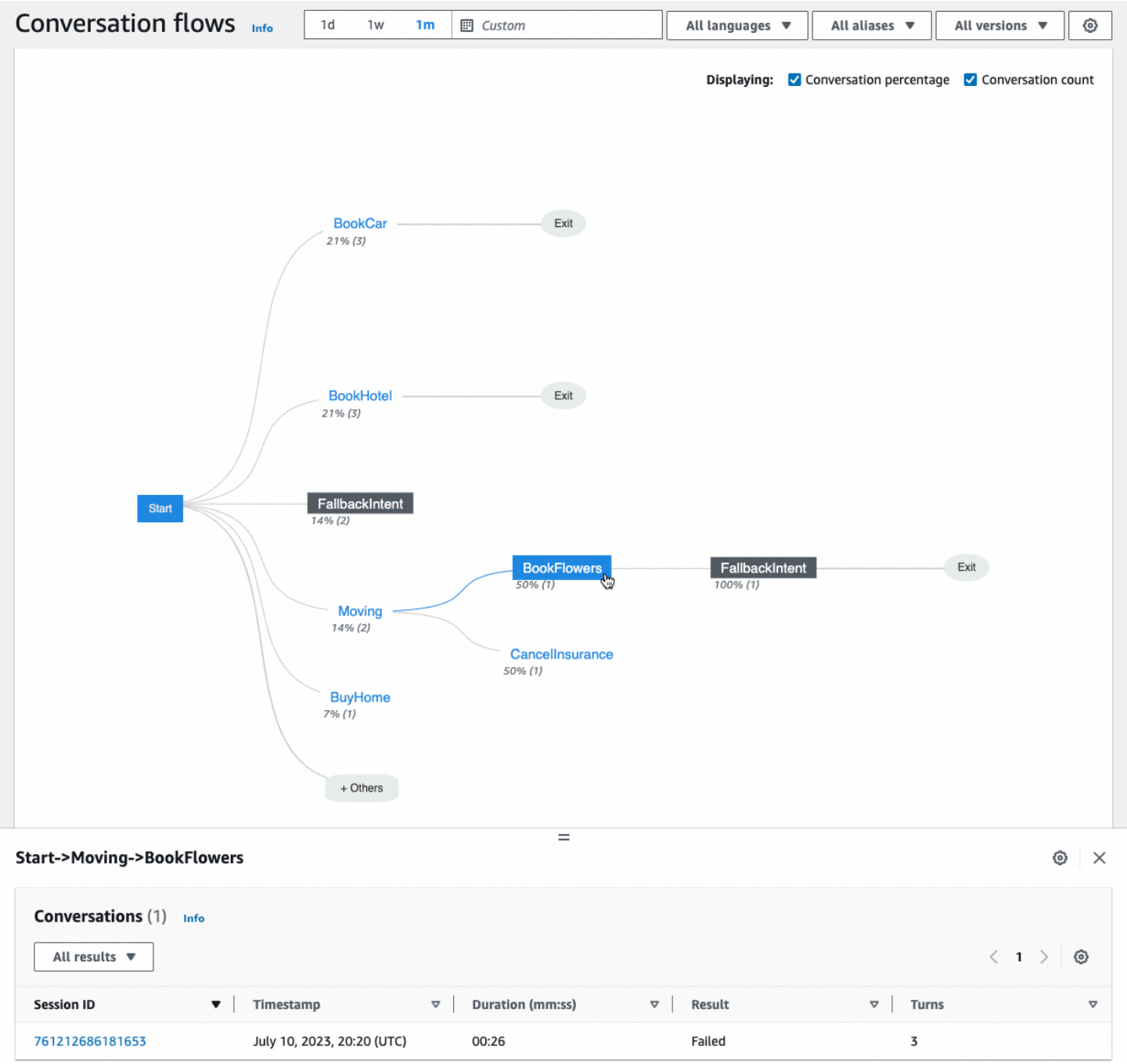
Dasbor Percakapan berisi dua alat, alur Percakapan dan Percakapan. Akses alat dengan memilihnya di dasbor Percakapan di panel navigasi kiri.

Alur percakapan

Gunakan alur Percakapan untuk memvisualisasikan urutan maksud yang diambil pelanggan dalam percakapan dengan bot Anda. Di bawah setiap maksud adalah persentase dan jumlah percakapan yang memanggil maksud itu pada saat itu dalam percakapan. Anda dapat mengubah persentase dan menghitung dengan memilih Persentase percakapan dan jumlah Percakapan di bagian atas. Secara default, lima maksud paling umum pada saat itu dalam percakapan ditampilkan dalam urutan frekuensi menurun. Pilih + Lainnya untuk menampilkan semua maksud.

Pilih maksud untuk memperluas ke kolom cabang baru yang menunjukkan daftar maksud yang diambil pada saat itu dalam percakapan, diurutkan dalam frekuensi menurun.

Saat Anda memilih node dalam alur percakapan, Anda dapat memperluas jendela di bawah ini untuk menampilkan daftar percakapan yang mengikuti urutan maksud tersebut. Pilih ID Sesi yang sesuai dengan percakapan untuk melihat detail tentang percakapan itu. Gambar berikut menunjukkan alur percakapan dan jendela Percakapan yang diperluas di bagian bawah.



Percakapan

Alat Percakapan menampilkan daftar percakapan untuk bot Anda. Anda dapat memilih kolom untuk mengurutkan menurut kolom itu dalam urutan naik atau turun.

Untuk memfilter percakapan berdasarkan hasil, pilih Semua hasil dan pilih Sukses, Gagal, atau Turun.

Untuk memfilter percakapan berdasarkan durasi

1. Pilih bilah pencarian bertanda Filter percakapan berdasarkan durasi
2. Tentukan filter dengan salah satu cara berikut:
   - Gunakan opsi yang telah ditentukan.
     a. Pilih Durasi.
     b. Pilih antara operator = (sama), > (lebih besar dari), dan < (kurang dari).
     c. Pilih jangka waktu.
   - Masukkan input dalam format “Durasi {operator} {angka} detik” Misalnya, untuk mencari semua percakapan yang berlangsung lebih dari 30 detik, masukkan**`Duration > 30 sec`**. Tentukan lamanya waktu dalam hitungan detik.

Untuk melihat informasi terperinci tentang sesi, termasuk metadata, penggunaan maksud, dan transkrip, pilih ID Sesi percakapan.

> **Note**
>
> Karena percakapan adalah kombinasi unik dari a `sessionId` dan`originatingRequestId`, hal yang sama `sessionId` mungkin muncul beberapa kali dalam tabel.

Bagian Detail berisi metadata berikut:

- Timestamp - Menentukan tanggal dan waktu mulai percakapan. Waktunya dalam format hh: mm: ss.
- Durasi - Menentukan berapa lama percakapan berlangsung dalam format mm: ss. Durasi tidak termasuk durasi batas waktu sesi (`idleSessionTTLInSeconds`).
- Hasil - Menentukan apakah percakapan dikategorikan sebagai sukses, gagal, atau gagal. Lihat Percakapan untuk detail lebih lanjut tentang hasil ini.
- Mode - Menentukan apakah percakapan itu`Speech`,`Text`, atau `DTMF` (menekan tombol nada sentuh). Percakapan yang terdiri dari beberapa mode adalah`Multimode`.
- Saluran - Menentukan saluran tempat percakapan berlangsung, jika berlaku. Lihat Mengintegrasikan bot Amazon Lex V2 dengan platform perpesanan.
- Bahasa - Menentukan bahasa bot.

Maksud yang ditimbulkan bot dalam percakapan ditunjukkan di bawah Detail. Pilih Pergi ke Maksud untuk pergi ke maksud itu di editor maksud. Pilih Jepret ke transkrip untuk secara otomatis menggulir Transkrip ke contoh pertama di mana bot memunculkan maksud.

Pilih panah kanan di sebelah nama maksud untuk melihat detail tentang slot yang ditimbulkan untuk maksud tersebut, termasuk nama slot, nilai yang diperoleh bot untuk setiap slot, dan berapa kali bot mencoba memperoleh setiap slot.

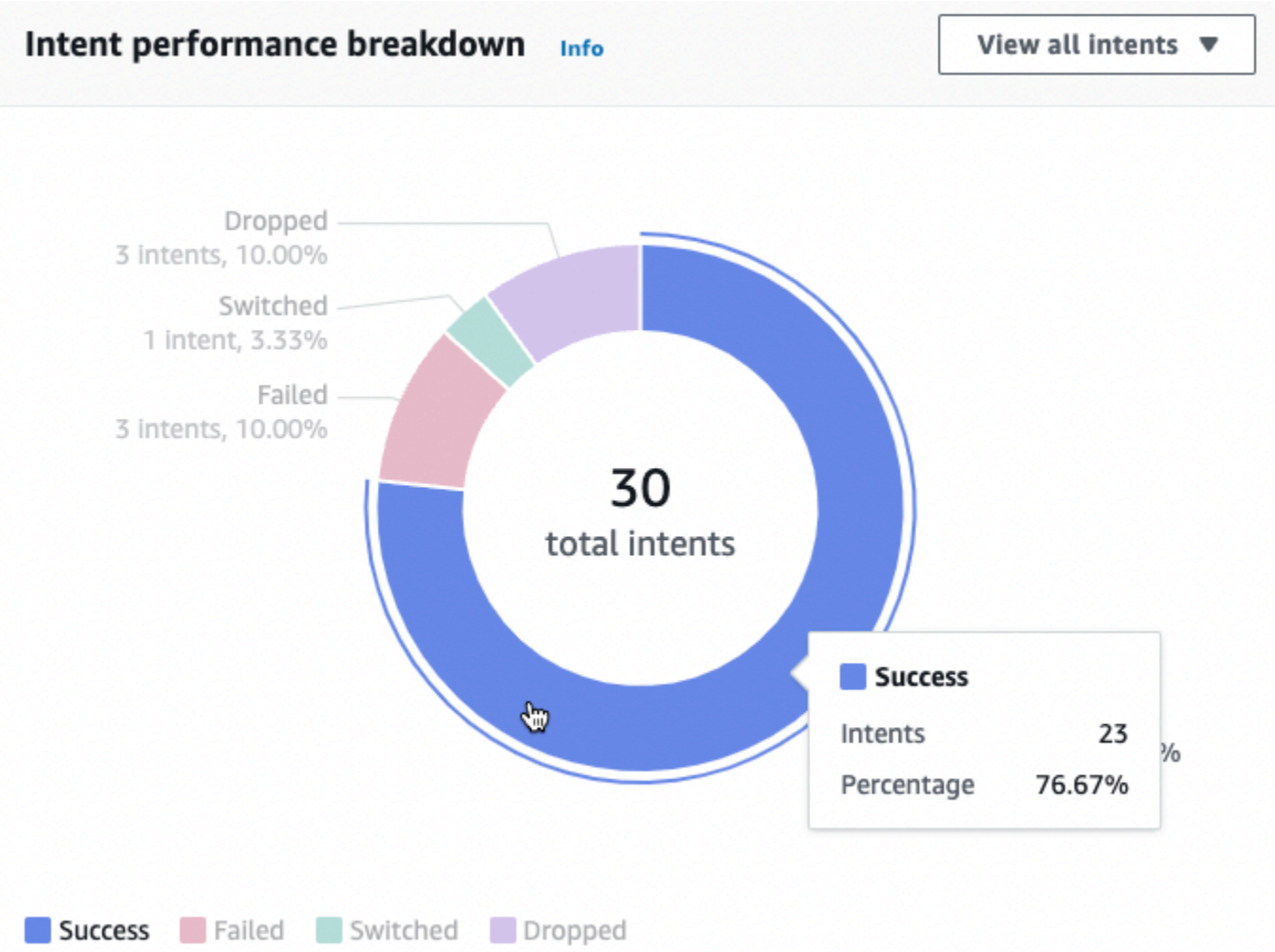
Transkrip memungkinkan Anda meninjau ucapan percakapan dan perilaku bot Anda dalam memunculkan maksud dan slot. Ucapan pengguna ditampilkan di sebelah kiri dan ucapan bot ditampilkan di sebelah kanan. Gunakan bilah pencarian bertanda Filter transkrip dalam sesi ini untuk menemukan teks dalam transkrip. Di samping Menampilkan: ada tiga informasi yang ditampilkan di bawah setiap giliran percakapan yang dapat Anda pilih untuk ditampilkan atau tidak:

- Timestamp - Menentukan waktu ucapan.
- Status maksud - Menentukan maksud yang ditimbulkan bot selama ucapan dan hasil dari maksud, jika berlaku. Status maksud berikut dimungkinkan:
  - Maksud yang dipanggil: *`intent name`* — Bot telah mengidentifikasi maksud yang diminta oleh pelanggan.
  - Niat beralih: *`intent name`* — Bot telah beralih ke maksud yang berbeda berdasarkan ucapan.
  - *`intent name`*: Sukses — Bot telah memenuhi niat.
- Status slot — Menentukan slot yang ditimbulkan bot selama ucapan, jika berlaku, dan nilai yang diberikan pelanggan.

Dasbor kinerja: ringkasan metrik maksud dan ucapan bot Anda

Di dasbor kinerja, Anda dapat melihat detail tentang kinerja pemenuhan maksud bot Anda dan pengenalan ucapan.

Bagian rincian kinerja Intent menampilkan jumlah total bot Anda memanggil intent dan memecah jumlah dan persentase kali intent dikategorikan sebagai sukses, gagal, dijatuhkan, dan diaktifkan. Lihat Intent penjelasan tentang definisi ini. Arahkan kursor ke segmen bagan untuk menampilkan kotak dengan jumlah dan persentase percakapan dengan hasil itu, seperti pada gambar berikut.



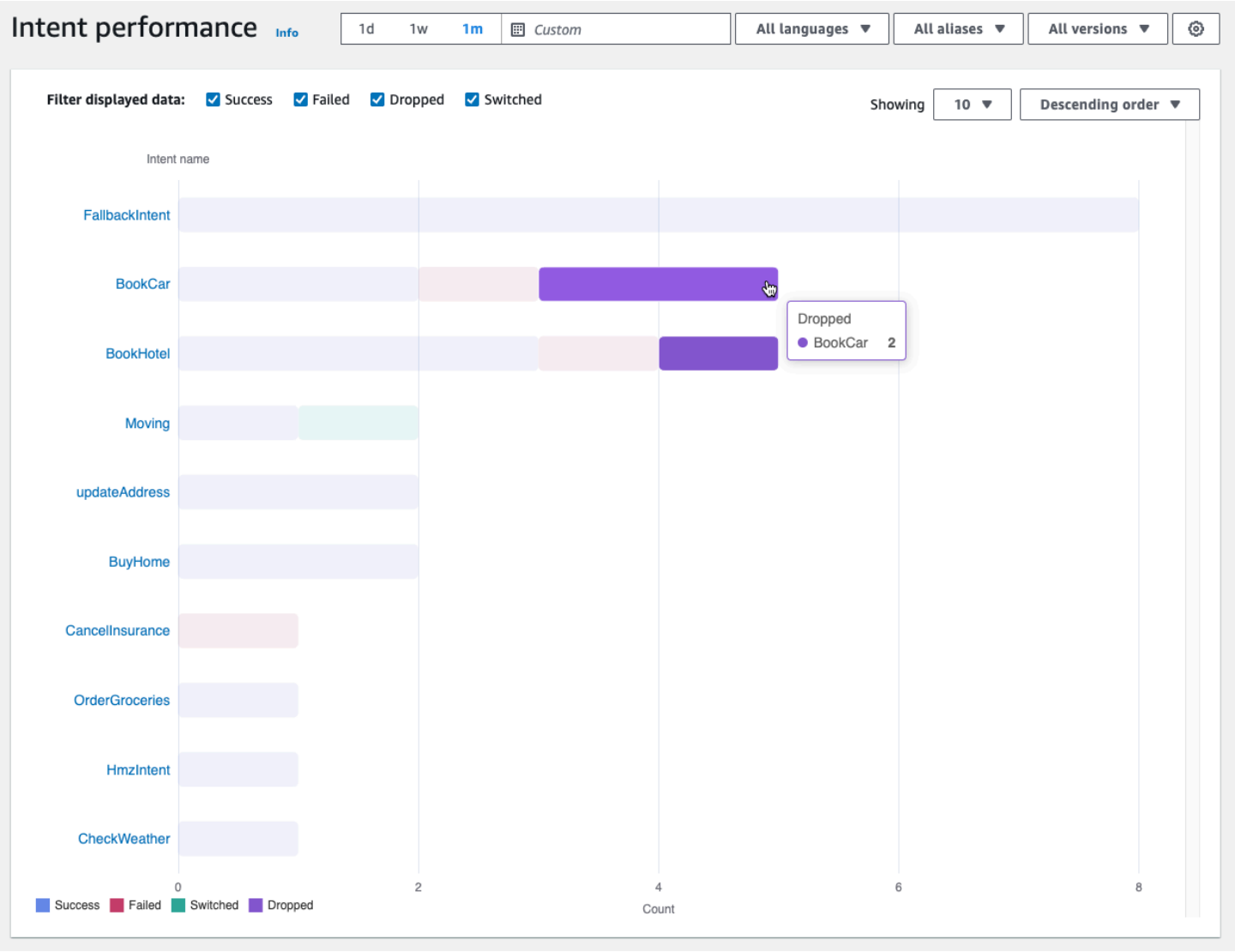
Pilih Lihat semua maksud untuk menampilkan menu tarik-turun, dari mana Anda dapat memilih untuk melihat daftar maksud yang ditimbulkan bot. Anda juga dapat memilih untuk melihat maksud dengan hasil tertentu (berhasil, gagal, jatuh, atau beralih). Tautan ini akan membawa Anda ke subbagian Kinerja Intent dari dasbor Kinerja. Untuk informasi selengkapnya, lihat Kinerja niat.

Bagian Pengenalan Ucapan merangkum jumlah ucapan yang terlewatkan dan terdeteksi. Pilih Lihat detail untuk menavigasi ke daftar ucapan untuk bot. Pilih nomor di bawah Ucapan tidak terjawab untuk melihat daftar ucapan yang tidak terjawab dan nomor di bawah Ucapan yang terdeteksi untuk melihat daftar ucapan yang terdeteksi untuk bot. Untuk informasi selengkapnya, lihat Pengakuan ucapan.

Pilih Performa maksud dan pengenalan Ucapan di bawah dasbor Kinerja di bilah sisi kiri untuk melihat detail tentang maksud dan ucapan di bot Anda.

Kinerja niat

Dasbor ini merangkum kinerja maksud yang digunakan dengan bot Anda dalam urutan frekuensi menurun. Bilah di sebelah setiap maksud memvisualisasikan berapa kali niat dikategorikan sebagai sukses, gagal, dijatuhkan, dan dialihkan. Lihat Intent penjelasan tentang definisi ini. Arahkan kursor ke segmen bilah untuk melihat jumlah percakapan menggunakan maksud tersebut dengan hasil tersebut, seperti pada gambar berikut:



> **Note**
>
> Dasbor menunjukkan 1.000 hasil teratas untuk satu set pengaturan filter. Untuk mendapatkan hasil yang lebih bertarget, konfigurasikan pengaturan filter granular.

Di bagian atas bagan, Anda dapat mengaktifkan status intent yang ingin Anda lihat dengan kotak centang Success, Failed, Dropped, dan Switched.

Pilih menu tarik-turun di sebelah kanan Menampilkan untuk menyesuaikan jumlah maksud yang akan ditampilkan dan apakah akan menampilkan maksud dalam urutan frekuensi naik atau turun.

Pilih nama maksud untuk menavigasi ke halaman yang menampilkan tiga bagan: Perincian kinerja intent, performa Slot, dan sakelar Intent.

Bagian rincian kinerja Intent menampilkan berapa kali bot menggunakan intent dan memecah jumlah dan persentase kali pemenuhan maksud dikategorikan sebagai sukses, gagal, dijatuhkan, dan diaktifkan. Lihat Intent penjelasan tentang definisi ini. Arahkan kursor ke segmen bagan untuk melihat hitungan dan persentase kali pemenuhan niat menghasilkan hasil itu.

Bagian kinerja Slot menampilkan metrik untuk slot yang termasuk dalam maksud saat ini. Untuk mengurutkan berdasarkan kolom, pilih kolom itu sekali untuk mengurutkan dalam urutan menaik dan dua kali untuk mengurutkannya dalam urutan menurun. Anda dapat menggunakan bilah pencarian untuk menemukan slot tertentu atau menggunakan tombol nomor halaman untuk menavigasi slot.

> **Note**
>
> Dasbor menunjukkan 1.000 hasil teratas untuk satu set pengaturan filter. Untuk mendapatkan hasil yang lebih bertarget, konfigurasikan pengaturan filter granular..

Bagian Intent switch mencantumkan instance di mana bot beralih dari intent saat ini ke yang lain dengan informasi berikut:

- Tahap — Tahap percakapan di mana bot mengalihkan maksud.
- Maksud beralih ke — Maksud bahwa bot mengalihkan maksud saat ini.
- Jumlah sesi — Jumlah sesi di mana Tahap dan Maksud beralih ke kombinasi terjadi.

> **Note**
>
> Dasbor menunjukkan 1.000 hasil teratas untuk satu set pengaturan filter. Untuk mendapatkan hasil yang lebih bertarget, konfigurasikan pengaturan filter granular..

Pengakuan ucapan

Halaman ini mencantumkan semua ucapan yang terlewatkan dan terdeteksi oleh bot Anda dan menyediakan alat bagi Anda untuk menambahkan contoh ucapan ke maksud untuk membantu melatih bot Anda. Lihat Ucapan penjelasan tentang definisi ini. Gunakan tab di bagian atas untuk beralih di antara daftar ucapan yang tidak terjawab dan ucapan Terdeteksi.

> **Note**
>
> Dasbor menunjukkan 1.000 hasil teratas untuk satu set pengaturan filter. Untuk mendapatkan hasil yang lebih bertarget, konfigurasikan pengaturan filter granular.

Untuk menambahkan ucapan ke intent:

1. Pilih kotak centang di samping ucapan yang ingin Anda tambahkan sebagai contoh ucapan untuk maksud.
2. Pilih Tambahkan ke maksud dan pilih maksud yang ingin Anda tambahkan ucapannya di menu tarik-turun di bawah Intent.
3. Pilih Tambahkan.

Menggunakan API untuk analitik

Bagian ini menjelaskan operasi API yang Anda gunakan untuk mengambil analitik untuk bot.

> **Note**
>
> Untuk menggunakan ListUtteranceMetricsdan ListUtteranceAnalyticsData, peran IAM Anda harus memiliki izin untuk melakukan ListAggregatedUtterancesoperasi, yang menyediakan akses ke analitik terkait ucapan. Lihat Melihat statistik ucapan dari percakapan Lex V2 untuk detail dan kebijakan IAM untuk diterapkan pada peran IAM.

- Operasi API berikut mengambil metrik ringkasan untuk bot:
  - ListSessionMetrics
  - ListIntentMetrics
  - ListIntentStageMetrics

  - ListUtteranceMetrics
- Operasi API berikut mengambil daftar metadata untuk sesi dan ucapan:
  - ListSessionAnalyticsData
  - ListUtteranceAnalyticsData
- ListIntentPathsOperasi mengambil metrik tentang urutan maksud yang diambil pelanggan dalam percakapan dengan bot.

Hasil penyaringan

Permintaan API Analytics mengharuskan Anda untuk menentukan `startTime` dan`endTime`. API menampilkan sesi, intent, tahapan maksud, atau ucapan yang dimulai setelah *`startTime`* dan berakhir sebelum. `endTime`

`filters`adalah bidang opsional dalam permintaan API Analytics. Ini memetakan ke daftar AnalyticsSessionFilter, AnalyticsIntentFilter, AnalyticsIntentStageFilter, atau AnalyticsUtteranceFilterobjek. Di setiap objek, gunakan bidang untuk membuat ekspresi untuk memfilter menurut. Misalnya, jika Anda menambahkan filter berikut ke daftar, bot akan mencari percakapan yang lebih dari 30 detik.

```
{
    "name": "Duration",
    "operator": "GT",
    "value": "30 sec",
}
```

Mengambil metrik untuk bot

Gunakan*`ListSessionMetrics`*, *`ListIntentMetrics`*`ListIntentStageMetrics`, dan *`ListUtteranceMetrics`* operasi untuk mengambil metrik ringkasan untuk sesi, maksud, tahapan maksud, dan ucapan.

Untuk operasi ini, isi kolom wajib berikut:

- Berikan `startTime` dan `endTime` untuk menentukan rentang waktu yang ingin Anda ambil hasilnya.
- Tentukan metrik yang ingin Anda hitung`metrics`, daftar AnalyticsSessionMetric,, AnalyticsIntentMetricAnalyticsIntentStageMetric, atau AnalyticsUtteranceMetricobjek. Di setiap

objek, gunakan `name` bidang untuk menentukan metrik untuk menghitung `statistic` bidang untuk menentukan apakah akan menghitung`Sum`,`Average`, atau `Max` angka, dan `order` bidang untuk menentukan apakah akan mengurutkan hasil dalam `Ascending` atau `Descending` urutan.

> **Note**
>
> Kedua objek `metrics` dan `binBy` objek mengandung `order` bidang. Anda dapat menentukan penyortiran hanya `order` dalam satu dari dua objek.

Bidang yang tersisa dalam permintaan adalah opsional. Anda dapat memfilter dan mengatur hasil dengan cara berikut:

- Hasil penyaringan — Gunakan `filters` bidang untuk memfilter hasil. Lihat Hasil penyaringan untuk detail selengkapnya.
- Mengelompokkan hasil berdasarkan kategori - Tentukan `groupBy` bidang, daftar yang berisi satu AnalyticsSessionResult,, AnalyticsIntentResultAnalyticsIntentStageResult, atau AnalyticsUtteranceResultobjek. Di objek, tentukan `name` bidang dengan kategori yang ingin Anda kelompokkan hasilnya.

  Jika Anda menentukan `groupBy` bidang dalam permintaan, `results` objek dalam respons berisi`groupByKeys`, daftar AnalyticsSessionGroupByKey,, AnalyticsIntentGroupByKeyAnalyticsIntentStageGroupByKey, atau AnalyticsUtteranceGroupByKeyobjek, masing-masing dengan `name` yang Anda tentukan dalam permintaan dan anggota kategori itu di `value` bidang tersebut.
- Hasil binning berdasarkan waktu - Tentukan `binBy` bidang, daftar yang berisi satu AnalyticsBinBySpecificationobjek. Dalam objek, tentukan `name` bidang dengan `ConversationStartTime` untuk membungkus hasil ketika percakapan dimulai atau `UtteranceTimestamp` untuk membuang hasil ketika ucapan berlangsung. Tentukan interval waktu yang Anda inginkan untuk memasukkan hasil di `interval` lapangan, dan apakah akan mengurutkan `Ascending` atau `Descending` urutan waktu di `order` lapangan.

  Jika Anda menentukan `binBy` bidang dalam permintaan, `results` objek dalam respons berisi`binKeys`, daftar AnalyticsBinKeyobjek, masing-masing dengan `name` yang Anda tentukan dalam permintaan dan interval waktu yang mendefinisikan bin itu di `value` bidang.

> **Note**
>
> Kedua objek `metrics` dan `binBy` objek mengandung `order` bidang. Anda dapat menentukan penyortiran hanya `order` dalam satu dari dua objek.

Gunakan bidang berikut untuk menangani tampilan respons:

- Tentukan angka antara 1 dan 1.000 di `maxResults` bidang untuk membatasi jumlah hasil yang akan dikembalikan dalam satu respons.
- Jika jumlah hasil lebih besar dari jumlah yang Anda tentukan di `maxResults` bidang, responsnya berisi a`nextToken`. Buat permintaan lagi, tetapi gunakan nilai ini di `nextToken` bidang untuk mengembalikan kumpulan hasil berikutnya.

Jika Anda menggunakan`ListUtteranceMetrics`, Anda dapat menentukan atribut untuk kembali di `attributes` bidang. Bidang ini memetakan ke daftar yang berisi satu AnalyticsUtteranceAttributeobjek. Tentukan `LastUsedIntent` di `name` bidang untuk mengembalikan maksud yang digunakan Amazon Lex V2 pada saat ucapan.

Sebagai tanggapan, `results` bidang memetakan ke daftar AnalyticsSessionResult,, AnalyticsIntentResultAnalyticsIntentStageResult, atau AnalyticsUtteranceResultobjek. Setiap objek berisi `metrics` bidang yang mengembalikan nilai statistik ringkasan untuk metrik yang Anda minta, selain bin atau grup apa pun yang dibuat dari metode yang Anda tentukan.

Mengambil metadata untuk sesi dan ucapan dalam bot

Gunakan ListSessionAnalyticsDatadan ListUtteranceAnalyticsDataoperasi untuk mengambil metadata tentang sesi dan ucapan individu.

Isi `endTime` kolom wajib `startTime` dan untuk menentukan rentang waktu yang ingin Anda ambil hasilnya.

Bidang yang tersisa dalam permintaan adalah opsional. Untuk memfilter dan mengurutkan hasil:

- Hasil penyaringan — Gunakan `filters` bidang untuk memfilter hasil. Lihat Hasil penyaringan untuk detail selengkapnya.

- Menyortir hasil — Urutkan hasil dengan `sortBy` bidang, yang berisi UtteranceDataSortByobjek SessionDataSortByatau. Tentukan nilai yang ingin Anda urutkan berdasarkan di `name` bidang dan apakah akan mengurutkan `Ascending` atau `Descending` mengurutkan di `order` bidang.

Gunakan bidang berikut untuk menangani tampilan respons:

- Tentukan angka antara 1 dan 1.000 di `maxResults` bidang untuk membatasi jumlah hasil yang akan dikembalikan dalam satu respons.
- Jika jumlah hasil lebih besar dari jumlah yang Anda tentukan di `maxResults` bidang, responsnya berisi a`nextToken`. Buat permintaan lagi, tetapi gunakan nilai ini di `nextToken` bidang untuk mengembalikan kumpulan hasil berikutnya.

Sebagai tanggapan, `sessions` atau `utterances` bidang memetakan ke daftar SessionSpecificationatau UtteranceSpecificationobjek. Setiap objek berisi metadata untuk satu sesi atau ucapan.

Mengambil data analitik jalur maksud

Gunakan ListIntentPathsoperasi untuk mengambil metrik tentang urutan maksud yang diambil pelanggan dalam percakapan dengan bot.

Untuk operasi ini, isi kolom wajib berikut:

- Berikan `startTime` dan `endTime` untuk menentukan rentang waktu yang ingin Anda ambil hasilnya.
- Berikan `intentPath` untuk menentukan urutan maksud yang ingin Anda ambil metriknya. Pisahkan maksud di jalan dengan garis miring ke depan. Misalnya, isi `intentPath` bidang dengan **`/BookCar/BookHotel`** untuk melihat detail tentang berapa kali pengguna memanggil `BookCar` dan `BookHotel` maksud dalam urutan itu.

Gunakan `filters` bidang opsional untuk memfilter hasil. Untuk detail selengkapnya, lihat Hasil penyaringan.

Melihat statistik ucapan dari percakapan Lex V2

Anda dapat menggunakan statistik ucapan untuk menentukan ucapan yang dikirim pengguna ke bot Anda. Anda dapat melihat kedua ucapan yang berhasil dideteksi Amazon Lex V2 dan ucapan yang tidak. Anda dapat menggunakan informasi ini untuk membantu menyetel bot Anda.

Misalnya, jika Anda menemukan bahwa pengguna Anda mengirim ucapan bahwa Amazon Lex V2 tidak ada, Anda dapat menambahkan ucapan ke intent. Versi Draft dari intent diperbarui dengan ucapan baru dan Anda dapat mengujinya sebelum menerapkannya ke bot Anda.

Ucapan terdeteksi saat Amazon Lex V2 mengenali ucapan tersebut sebagai upaya untuk memanggil maksud yang dikonfigurasi untuk bot. Ucapan terlewatkan ketika Amazon Lex V2 tidak mengenali ucapannya dan memanggilnya. `AMAZON.FallbackIntent`

Statistik ucapan dapat dilihat menggunakan `ListUtteranceMetrics` API dan API. `ListAggregatedUtterance`

Statistik ucapan tidak dibuat menggunakan `ListUtteranceMetrics` API dalam kondisi berikut:

- Pengaturan Undang-Undang Perlindungan Privasi Online Anak disetel ke Ya saat bot dibuat dengan konsol, atau `childDirected` bidang disetel ke true saat bot dibuat dengan `CreateBot` operasi.

`ListUtteranceMetrics`API menyediakan fitur tambahan termasuk:

- Informasi lebih lanjut tersedia, seperti maksud yang dipetakan untuk ucapan yang terdeteksi.
- Lebih banyak kemampuan penyaringan (termasuk saluran dan mode).
- Rentang tanggal retensi yang lebih lama (30 hari).
- Anda dapat menggunakan API bahkan jika Anda telah memilih keluar dari penyimpanan data. Fungsionalitas konsol untuk ucapan yang terlewat dan terdeteksi akan bergantung pada `ListUtteranceMetrics` API.

Statistik ucapan tidak dibuat menggunakan `ListAggregatedUtterance` API dalam kondisi berikut:

- Pengaturan Undang-Undang Perlindungan Privasi Online Anak disetel ke Ya saat bot dibuat dengan konsol, atau `childDirected` bidang disetel ke true saat bot dibuat dengan `CreateBot` operasi.
- Anda menggunakan slot obfuscation dengan satu atau lebih slot.
- Anda memilih untuk tidak berpartisipasi dalam meningkatkan Amazon Lex.

`ListAggregatedUtterance`API menyediakan fitur termasuk:

- Informasi yang kurang rinci tersedia (tidak ada maksud yang dipetakan untuk ucapan).
- Kemampuan penyaringan terbatas (tidak termasuk saluran dan mode).
- Rentang tanggal retensi pendek (15 hari).

Dengan menggunakan statistik ucapan, Anda dapat melihat apakah ucapan tertentu terdeteksi atau terlewatkan, di samping terakhir kali ucapan tersebut digunakan dalam interaksi bot.

Amazon Lex V2 menyimpan ucapan terus menerus saat pengguna berinteraksi dengan bot Anda. Anda dapat menanyakan statistik menggunakan konsol atau `ListAggregatedUtterances` operasi. Ini memiliki retensi data 15 hari dan tidak tersedia jika pengguna telah memilih keluar dari penyimpanan data. Anda dapat menghapus ucapan menggunakan `DeleteUtterances` operasi atau dengan memilih keluar dari penyimpanan data. Semua ucapan akan dihapus jika Anda menutup akun AWS Anda. Ucapan yang disimpan dienkripsi dengan kunci yang dikelola server.

Saat Anda menghapus versi bot, statistik ucapan tersedia untuk versi hingga 30 hari dengan`ListUtteranceMetrics`, dan 15 hari penggunaan. `ListAggregatedUtterances` Anda tidak dapat melihat statistik untuk versi yang dihapus di konsol Amazon Lex V2. Untuk melihat statistik untuk versi yang dihapus, Anda dapat menggunakan keduanya `ListAggregatedUtterances` dan `ListUtteranceMetrics` operasi.

Dengan `ListUtteranceMetrics` API `ListAggregatedUtterances` dan API, ucapan digabungkan dengan teks ucapan. Misalnya, semua contoh di mana pelanggan menggunakan frasa “Saya ingin memesan pizza” digabungkan ke dalam baris yang sama sebagai tanggapan. Saat Anda menggunakan RecognizeUtteranceoperasi, teks yang digunakan adalah transkrip input.

Untuk menggunakan `ListAggregatedUtterances` dan `ListUtteranceMetrics` API, terapkan kebijakan berikut ke peran.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ListAggregatedUtterancesPolicy",
            "Effect": "Allow",
            "Action": "lex:ListAggregatedUtterances",
```

```
            "Resource": "*"
        }
    ]
}
```

Mengelola izin akses untuk analitik

Untuk memberikan akses pengguna ke analitik, lampirkan kebijakan ke peran IAM yang mengizinkan peran tersebut memanggil operasi API untuk analitik. Anda dapat melampirkan Kebijakan terkelola AWS: AmazonLexFullAccess ke peran IAM untuk menyediakan akses penuh ke operasi Amazon Lex API, atau Anda dapat membuat kebijakan khusus yang hanya mengizinkan izin untuk analitik dan melampirkannya ke peran IAM.

Untuk membuat kebijakan kustom yang berisi izin untuk analitik

1. Jika Anda perlu terlebih dahulu membuat peran IAM, ikuti langkah-langkah di Membuat peran untuk mendelegasikan izin ke pengguna IAM.
2. Ikuti langkah-langkah di Membuat kebijakan IAM untuk membuat kebijakan menggunakan objek JSON berikut. Untuk mengaktifkan akses analitik ke bot tertentu untuk peran IAM, tambahkan ARN setiap bot ke bidang. `Resource` Ganti*`region`*,*`account-id`*, dan *`BOTID`* dengan nilai yang sesuai dengan bot. Anda juga dapat mengganti pengenal pernyataan*`AnalyticsActions`*,, dengan nama pilihan Anda.
3. Lampirkan kebijakan yang Anda buat ke peran yang ingin Anda berikan izin analitik dengan mengikuti langkah-langkah di Menambahkan dan menghapus izin identitas IAM.
4. Peran tersebut sekarang harus memiliki izin untuk melihat analitik untuk bot yang Anda tentukan.

Mengaktifkan log percakapan untuk bot Lex V2 Anda

Gunakan log percakapan untuk menyimpan percakapan pengguna dengan bot Anda. Tinjau log ini untuk mengidentifikasi masalah dengan interaksi bot Anda dengan pengguna dan mengubah perilaku bot Anda dengan wawasan ini. Bagian ini juga menjelaskan cara mengaburkan nilai slot untuk melindungi privasi pengguna.

Topik

- Mencatat percakapan dengan log percakapan di Lex V2
- Mengaburkan nilai slot dalam log percakapan dari Lex V2

- Pengambilan log percakapan selektif di Lex V2

Mencatat percakapan dengan log percakapan di Lex V2

Anda mengaktifkan log percakapan untuk menyimpan interaksi bot. Anda dapat menggunakan log ini untuk meninjau performa bot dan memecahkan masalah terkait percakapan. Anda dapat mencatat teks untuk RecognizeTextoperasi. Anda dapat mencatat teks dan audio untuk RecognizeUtteranceoperasi. Dengan mengaktifkan log percakapan, Anda mendapatkan tampilan terperinci tentang percakapan yang dimiliki pengguna dengan bot Anda.

Misalnya, sesi dengan bot Anda memiliki ID sesi. Anda dapat menggunakan ID ini untuk mendapatkan transkrip percakapan termasuk ucapan pengguna dan respons bot yang sesuai. Anda juga mendapatkan metadata seperti nama maksud dan nilai slot untuk ucapan.

> **Note**
>
> Anda tidak dapat menggunakan log percakapan dengan bot yang tunduk pada Undang-Undang Perlindungan Privasi Online Anak-anak (COPPA).

Log percakapan dikonfigurasi untuk alias. Setiap alias dapat memiliki pengaturan yang berbeda untuk teks dan log audio mereka. Anda dapat mengaktifkan log teks, log audio, atau keduanya untuk setiap alias. Log teks menyimpan input teks, transkrip input audio, dan metadata terkait di Log. CloudWatch Log audio menyimpan input audio di Amazon S3. Anda dapat mengaktifkan enkripsi log teks dan audio menggunakan CMK yang dikelola AWS KMS pelanggan.

Untuk mengkonfigurasi logging, gunakan konsol atau UpdateBotAliasoperasi CreateBotAliasatau. Setelah mengaktifkan log percakapan untuk alias, gunakan RecognizeUtteranceoperasi RecognizeTextatau untuk alias tersebut mencatat ucapan teks atau audio dalam grup log Log yang dikonfigurasi atau bucket CloudWatch S3.

Topik

- Kebijakan IAM untuk Log Percakapan
- Mengkonfigurasi log percakapan untuk bot Lex V2 Anda
- Melihat log teks di Amazon CloudWatch Logs dari Lex V2
- Mengakses log audio di Amazon S3
- Memantau status log percakapan dengan CloudWatch metrik

Kebijakan IAM untuk Log Percakapan

Bergantung pada jenis logging yang Anda pilih, Amazon Lex V2 memerlukan izin untuk menggunakan Amazon CloudWatch Logs dan Amazon Simple Storage Service (S3) bucket untuk menyimpan log Anda. Anda harus membuat peran dan izin AWS Identity and Access Management untuk mengaktifkan Amazon Lex V2 mengakses sumber daya ini.

Membuat Peran dan Kebijakan IAM untuk Log Percakapan

Untuk mengaktifkan log percakapan, Anda harus memberikan izin menulis untuk CloudWatch Log dan Amazon S3. Jika Anda mengaktifkan enkripsi objek untuk objek S3 Anda, Anda perlu memberikan izin akses ke AWS KMS kunci yang digunakan untuk mengenkripsi objek.

Anda dapat menggunakan konsol IAM, API IAM, atau AWS Command Line Interface untuk membuat peran dan kebijakan. Petunjuk ini menggunakan AWS CLI untuk membuat peran dan kebijakan.

> **Note**
>
> Kode berikut diformat untuk Linux dan macOS. Untuk Windows, ganti karakter kelanjutan baris Linux (\) dengan tanda sisipan (^).

Untuk membuat peran IAM untuk log percakapan

1. Buat dokumen di direktori saat ini yang disebut**`LexConversationLogsAssumeRolePolicyDocument.json`**, tambahkan kode berikut ke dalamnya, dan simpan. Dokumen kebijakan ini menambahkan Amazon Lex V2 sebagai entitas tepercaya ke peran tersebut. Hal ini memungkinkan Amazon Lex V2 untuk mengambil peran untuk mengirimkan log ke sumber daya yang dikonfigurasi untuk log percakapan.

   JSON

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

```
      "Principal": {
        "Service": "lexv2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

2. Di AWS CLI, jalankan perintah berikut untuk membuat peran IAM untuk log percakapan.

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

Selanjutnya, buat dan lampirkan kebijakan ke peran yang memungkinkan Amazon Lex V2 menulis ke CloudWatch Log.

Untuk membuat kebijakan IAM untuk mencatat teks percakapan ke CloudWatch Log

1. Buat dokumen di direktori saat ini yang disebut**LexConversationLogsCloudWatchLogsPolicy.json**, tambahkan kebijakan IAM berikut ke dalamnya, dan simpan.
2. Di AWS CLI, buat kebijakan IAM yang memberikan izin menulis ke grup log Log. CloudWatch

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

3. Lampirkan kebijakan ke peran IAM yang Anda buat untuk log percakapan.

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

Jika Anda mencatat audio ke bucket S3, buat kebijakan yang memungkinkan Amazon Lex V2 menulis ke bucket.

Untuk membuat kebijakan IAM untuk pencatatan audio ke bucket S3

1. Buat dokumen di direktori saat ini yang disebut**LexConversationLogsS3Policy.json**, tambahkan kebijakan berikut ke dalamnya, dan simpan.

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
      {
          "Effect": "Allow",
          "Action": [
              "s3:PutObject"
          ],
          "Resource": "arn:aws:s3:::bucket-name/*"
      }
  ]
}
```

2. Di AWS CLI, buat kebijakan IAM yang memberikan izin menulis ke bucket S3 Anda.

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

3. Lampirkan kebijakan ke peran yang Anda buat untuk log percakapan.

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

Memberikan Izin untuk Lulus Peran IAM

Saat Anda menggunakan konsol, AWS SDK AWS Command Line Interface, atau AWS untuk menentukan peran IAM yang akan digunakan untuk log percakapan, pengguna yang menentukan peran IAM log percakapan harus memiliki izin untuk meneruskan peran tersebut ke Amazon Lex V2. Untuk mengizinkan pengguna meneruskan peran ke Amazon Lex V2, Anda harus memberikan `PassRole` izin kepada pengguna, peran, atau grup IAM pengguna.

Kebijakan berikut menentukan izin untuk diberikan kepada pengguna, peran, atau grup. Anda dapat menggunakan tombol `iam:AssociatedResourceArn` dan `iam:PassedToService` kondisi untuk membatasi ruang lingkup izin. Untuk informasi selengkapnya, lihat Memberikan Izin Pengguna untuk Meneruskan Peran ke Layanan AWS dan IAM serta Kunci Konteks AWS STS Kondisi dalam Panduan Pengguna AWS Identity and Access Management.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::111122223333:role/role-name",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "lexv2.amazonaws.com"
                },
                "StringLike": {
                    "iam:AssociatedResourceARN":
 "arn:aws:lex:region:123456789012:bot:bot-name:bot-alias"
                }
            }
        }
    ]
}
```

Mengkonfigurasi log percakapan untuk bot Lex V2 Anda

Anda mengaktifkan dan menonaktifkan log percakapan menggunakan konsol atau `conversationLogSettings` bidang `UpdateBotAlias` operasi `CreateBotAlias` atau. Anda dapat mengaktifkan atau mematikan log audio, log teks, atau keduanya. Logging dimulai pada sesi bot baru. Perubahan pada setelan log tidak tercermin untuk sesi aktif.

Untuk menyimpan log teks, gunakan grup CloudWatch log Amazon Logs di akun AWS Anda. Anda dapat menggunakan grup log yang valid. Grup log harus berada di wilayah yang sama dengan bot Amazon Lex V2. Untuk informasi selengkapnya tentang membuat grup CloudWatch log Log, lihat Bekerja dengan Grup Log dan Aliran Log di Panduan Pengguna CloudWatch Log Amazon.

Untuk menyimpan log audio, gunakan bucket Amazon S3 di akun AWS Anda. Anda dapat menggunakan bucket S3 yang valid. Bucket harus berada di wilayah yang sama dengan bot Amazon Lex V2. Untuk informasi selengkapnya tentang membuat bucket S3, lihat Membuat bucket di Panduan Memulai Layanan Penyimpanan Sederhana Amazon.

Saat Anda mengelola log percakapan menggunakan konsol, konsol akan memperbarui peran layanan Anda sehingga memiliki akses ke grup log dan bucket S3.

Jika Anda tidak menggunakan konsol, Anda harus memberikan peran IAM dengan kebijakan yang memungkinkan Amazon Lex V2 untuk menulis ke grup log atau bucket yang dikonfigurasi. Jika Anda membuat peran terkait layanan menggunakan AWS Command Line Interface, Anda harus menambahkan akhiran kustom ke peran menggunakan `custom-suffix` opsi seperti pada contoh berikut. Untuk informasi selengkapnya, lihat Membuat Peran dan Kebijakan IAM untuk Log Percakapan.

```
aws iam create-service-linked-role \
    --aws-service-name lexv2.amazon.aws.com \
    --custom-suffix suffix
```

Peran IAM yang Anda gunakan untuk mengaktifkan log percakapan harus memiliki `iam:PassRole` izin. Kebijakan berikut harus dilampirkan pada peran:

JSON

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

Mengaktifkan log percakapan

Untuk mengaktifkan log menggunakan konsol

1. Buka konsol Amazon Lex V2 https://console.aws.amazon.com/lexv2.
2. Dari daftar, pilih bot.
3. Dari menu sebelah kiri, pilih Alias.
4. Dalam daftar alias, pilih alias yang ingin Anda konfigurasikan log percakapan.
5. Di bagian Log percakapan, pilih Kelola log percakapan.
6. Untuk log teks, pilih Aktifkan lalu masukkan nama grup CloudWatch log Amazon Logs.
7. Untuk log audio, pilih Aktifkan lalu masukkan informasi bucket S3.
8. Tidak wajib. Untuk mengenkripsi log audio, pilih AWS KMS kunci yang akan digunakan untuk enkripsi.
9. Pilih Simpan untuk mulai mencatat percakapan. Jika perlu, Amazon Lex V2 akan memperbarui peran layanan Anda dengan izin untuk mengakses grup CloudWatch log Log dan bucket S3 yang dipilih.

Menonaktifkan log percakapan di Lex V2

Untuk mematikan log menggunakan konsol

1. Buka konsol Amazon Lex V2 https://console.aws.amazon.com/lexv2.
2. Dari daftar, pilih bot.
3. Dari menu sebelah kiri, pilih Alias.
4. Dalam daftar alias, pilih alias yang ingin Anda konfigurasikan log percakapan.
5. Di bagian Log percakapan, pilih Kelola log percakapan.
6. Nonaktifkan pencatatan teks, pencatatan audio, atau keduanya untuk mematikan logging.
7. Pilih Simpan untuk menghentikan pencatatan percakapan.

Melihat log teks di Amazon CloudWatch Logs dari Lex V2

Amazon Lex V2 menyimpan log teks untuk percakapan Anda di Amazon CloudWatch Logs. Untuk melihat log, gunakan konsol CloudWatch Log atau API. Untuk informasi selengkapnya, lihat Cari Data

Log Menggunakan Pola Filter dan Sintaks Kueri Wawasan CloudWatch Log di Panduan Pengguna Amazon CloudWatch Logs.

Untuk melihat log menggunakan konsol Amazon Lex V2

1. Buka konsol Amazon Lex V2 https://console.aws.amazon.com/lexv2.
2. Dari daftar, pilih bot.
3. Dari menu kiri, pilih Analytics dan kemudian pilih CloudWatch metrik.
4. Lihat metrik untuk bot Anda di halaman CloudWatch metrik.

Anda juga dapat menggunakan CloudWatch konsol atau API untuk melihat entri log Anda. Untuk menemukan entri log, navigasikan ke grup log yang Anda konfigurasikan untuk alias. Anda dapat menemukan awalan aliran log untuk log Anda di konsol Amazon Lex V2 atau dengan menggunakan DescribeBotAliasoperasi.

Entri log untuk ucapan pengguna ditemukan di beberapa aliran log. Ucapan dalam percakapan memiliki entri di salah satu aliran log dengan awalan yang ditentukan. Entri dalam aliran log berisi informasi berikut:

versi pesan

Versi skema pesan.

bot

Detail tentang bot yang berinteraksi dengan pelanggan.

pesan

Tanggapan yang dikirim bot kembali ke pengguna.

UtteranceContext

Informasi tentang memproses ucapan ini.

- `runtimeHints`—konteks runtime digunakan untuk mentranskripsikan dan menafsirkan masukan pengguna. Untuk informasi selengkapnya, lihat Meningkatkan pengenalan nilai slot dengan petunjuk runtime dalam percakapan.
- `slotElicitationStyle`—Gaya elisitasi slot digunakan untuk menafsirkan input pengguna. Untuk informasi selengkapnya, lihat Menangkap nilai slot dengan gaya ejaan selama percakapan.

SessionState

Keadaan percakapan saat ini antara pengguna dan bot. Untuk informasi selengkapnya, lihat Memahami percakapan bot Amazon Lex V2.

interpretasi

Daftar maksud yang ditentukan Amazon Lex V2 dapat memuaskan ucapan pengguna. Menggunakan skor kepercayaan diri untuk meningkatkan akurasi percakapan.

InterpretasiSumber

Menunjukkan apakah slot diselesaikan oleh Amazon Lex V2 atau Amazon Bedrock. Nilai: Lex | Bedrock

SESSIONID

Pengidentifikasi sesi pengguna yang melakukan percakapan.

INPUTTRANSCRIPT

Transkripsi input dari pengguna.

- Untuk input teks, ini adalah teks yang diketik pengguna. Untuk input DTMF, ini adalah kunci yang dimasukkan pengguna.
- Untuk input ucapan, ini adalah teks yang digunakan Amazon Lex V2 untuk mengonversi ucapan pengguna untuk memanggil maksud atau mengisi slot.

mentah InputTranscript

Transkrip mentah dari input pengguna sebelum pemrosesan teks diterapkan. Catatan: Pemrosesan teks hanya untuk lokal en-US dan en-GB.

transkripsi

Daftar transkripsi potensial dari input pengguna. Untuk informasi selengkapnya, lihat Menggunakan skor kepercayaan transkripsi suara untuk meningkatkan percakapan dengan bot Lex V2 Anda.

RawTranskripsi

Menggunakan skor kepercayaan transkripsi suara. Untuk informasi selengkapnya, lihat Menggunakan skor kepercayaan transkripsi suara untuk meningkatkan percakapan dengan bot Lex V2 Anda.

MisseDutterance

Menunjukkan apakah Amazon Lex V2 dapat mengenali ucapan pengguna.

requestId

Amazon Lex V2 menghasilkan ID permintaan untuk input pengguna.

timestamp

Stempel waktu input pengguna.

DeveloperOverride

Menunjukkan apakah alur percakapan diperbarui menggunakan hook kode dialog. Untuk informasi selengkapnya tentang menggunakan hook kode dialog, lihatMengintegrasikan AWS Lambda fungsi ke bot Amazon Lex V2 Anda.

InputMode

Menunjukkan jenis input. Bisa audio, DTMF, atau teks.

requestAttributes

Atribut permintaan yang digunakan saat memproses input pengguna.

AudioProperties

Jika log percakapan audio diaktifkan dan input pengguna dalam format audio, termasuk total durasi input audio, durasi suara dan durasi keheningan dalam audio. Ini juga termasuk tautan ke file audio.

BargeIn

Menunjukkan apakah input pengguna mengganggu respons bot sebelumnya.

ResponseReason

Alasan respons dihasilkan. Bisa menjadi salah satu dari:

- `UtteranceResponse`— respon terhadap masukan pengguna
- `StartTimeout`— respons yang dihasilkan server saat pengguna tidak memberikan masukan
- `StillWaitingResponse`— respons yang dihasilkan server saat pengguna meminta bot menunggu
- `FulfillmentInitiated`— respons yang dihasilkan server bahwa pemenuhan akan dimulai
- `FulfillmentStartedResponse`— server menghasilkan respons bahwa pemenuhan telah dimulai
- `FulfillmentUpdateResponse`— respon yang dihasilkan server periodik saat pemenuhan sedang berlangsung

- `FulfillmentCompletedResponse`— respons yang dihasilkan server saat pemenuhan selesai.

operationName

API digunakan untuk berinteraksi dengan bot. Bisa menjadi salah satu`PutSession`,`RecognizeText`,`RecognizeUtterance`, atau`StartConversation`.

```
{
    "message-version": "2.0",
    "bot": {
        "id": "string",
        "name": "string",
        "aliasId": "string",
        "aliasName": "string",
        "localeId": "string",
        "version": "string"
    },
    "messages": [
        {
            "contentType": "PlainText | SSML | CustomPayload | ImageResponseCard",
            "content": "string",
            "imageResponseCard": {
                "title": "string",
                "subtitle": "string",
                "imageUrl": "string",
                "buttonsList": [
                    {
                        "text": "string",
                        "value": "string"
                    }
                ]
            }
        }
    ],
    "utteranceContext": {
        "activeRuntimeHints": {
            "slotHints": {
                "string": {
                    "string": {
                        "runtimeHintValues": [
                            {
```

```
                            "phrase": "string"
                        },
                        {
                            "phrase": "string"
                        }
                    ]
                }
            }
        }
    },
    "slotElicitationStyle": "string"
},
"sessionState": {
    "dialogAction": {
        "type": "Close | ConfirmIntent | Delegate | ElicitIntent | ElicitSlot",
        "slotToElicit": "string"
    },
    "intent": {
        "name": "string",
        "slots": {
            "string": {
                "value": {
                   "interpretedValue": "string",
                   "originalValue": "string",
                   "resolvedValues": [ "string" ]
                }
             },
            "string": {
                "shape": "List",
                "value": {
                    "originalValue": "string",
                    "interpretedValue": "string",
                    "resolvedValues": [ "string" ]
                },
                "values": [
                    {
                        "shape": "Scalar",
                        "value": {
                            "originalValue": "string",
                            "interpretedValue": "string",
                            "resolvedValues": [ "string" ]
                        }
                    },
                    {
```

```
                            "shape": "Scalar",
                            "value": {
                                "originalValue": "string",
                                "interpretedValue": "string",
                                "resolvedValues": [ "string" ]
                            }
                        }
                    ]
                }
            },
            "kendraResponse": {
                // Only present when intent is KendraSearchIntent. For details, see
                // https://docs.aws.amazon.com/kendra/latest/dg/
API_Query.html#API_Query_ResponseSyntax
                },
            "state": "InProgress | ReadyForFulfillment | Fulfilled | Failed",
            "confirmationState": "Confirmed | Denied | None"
        },
        "originatingRequestId": "string",
        "sessionAttributes": {
            "string": "string"
        },
        "runtimeHints": {
            "slotHints": {
                "string": {
                    "string": {
                        "runtimeHintValues": [
                            {
                                "phrase": "string"
                            },
                            {
                                "phrase": "string"
                            }
                        ]
                    }
                }
            }
        }
    },
   "dialogEventLogs": [
        {
   // only for conditional
     "conditionalEvaluationResult":[
      // all the branches until true
```

```
    {
    "conditionalBranchName": "string",
    "expressionString": "string",
    "evaluatedExpression": "string",
    "evaluationResult": "true | false"
    }
   ],
"dialogCodeHookInvocationLabel": "string",
"response": "string",
"nextStep": {
      "dialogAction": {
          "type": "Close | ConfirmIntent | Delegate | ElicitIntent | ElicitSlot",
          "slotToElicit": "string"
      },
     "intent": {
                        "name": "string",
         "slots": {
             }
      }
     }
  ]
  "interpretations": [
      {
          "interpretationSource": "Bedrock | Lex",
          "nluConfidence": "string",
          "intent": {
              "name": "string",
              "slots": {
                  "string": {
                      "value": {
                          "originalValue": "string",
                          "interpretedValue": "string",
                          "resolvedValues": [ "string" ]
                      }
                  },
                  "string": {
                      "shape": "List",
                      "value": {
                          "interpretedValue": "string",
                          "originalValue": "string",
                          "resolvedValues": [ "string" ]
                      },
                      "values": [
```

```
                            {
                                "shape": "Scalar",
                                "value": {
                                    "interpretedValue": "string",
                                    "originalValue": "string",
                                    "resolvedValues": [ "string" ]
                                }
                            },
                            {
                                "shape": "Scalar",
                                "value": {
                                    "interpretedValue": "string",
                                    "originalValue": "string",
                                    "resolvedValues": [ "string" ]
                                }

                            }
                        ]
                    }
                },
                "kendraResponse": {
                    // Only present when intent is KendraSearchIntent. For details, see

                    // https://docs.aws.amazon.com/kendra/latest/dg/
API_Query.html#API_Query_ResponseSyntax
                    },
                "state": "InProgress | ReadyForFulfillment | Fulfilled | Failed",
                "confirmationState": "Confirmed | Denied | None"
                },
            "sentimentResponse": {
                "sentiment": "string",
                "sentimentScore": {
                    "positive": "string",
                    "negative": "string",
                    "neutral": "string",
                    "mixed": "string"
                }
            }
        }
    ],
    "sessionId": "string",
    "inputTranscript": "string",
    "rawInputTranscript": "string",
    "transcriptions": [
```

```
        {
            "transcription": "string",
            "rawTranscription": "string",
            "transcriptionConfidence": "number",
            },
            "resolvedContext": {
                "intent": "string"
            },
            "resolvedSlots": {
                "string": {
                    "name": "slotName",
                    "shape": "List",
                    "value": {
                        "originalValue": "string",
                        "resolvedValues": [
                            "string"
                        ]
                    }
                }
            }
        }
    ],
    "missedUtterance": "bool",
    "requestId": "string",
    "timestamp": "string",
    "developerOverride": "bool",
    "inputMode": "DTMF | Speech | Text",
    "requestAttributes": {
        "string": "string"
    },
    "audioProperties": {
        "contentType": "string",
        "s3Path": "string",
        "duration": {
            "total": "integer",
            "voice": "integer",
            "silence": "integer"
        }
    },
    "bargeIn": "string",
    "responseReason": "string",
    "operationName": "string"
}
```

Isi entri log tergantung pada hasil transaksi dan konfigurasi bot dan permintaan.

- `slotToElicit`Bidang `intents`slots,, dan tidak muncul dalam entri jika `missedUtterance` bidang tersebut`true`.
- `s3PathForAudio`Bidang tidak muncul jika log audio dinonaktifkan atau jika `inputDialogMode` bidangnya`Text`.
- `responseCard`Bidang hanya muncul ketika Anda telah menentukan kartu respons untuk bot.
- `requestAttributes`Peta hanya muncul jika Anda telah menentukan atribut permintaan dalam permintaan.
- `kendraResponse`Bidang ini hanya ada ketika `AMAZON.KendraSearchIntent` membuat permintaan untuk mencari indeks Amazon Kendra.
- `developerOverride`Bidang ini benar ketika maksud alternatif ditentukan dalam fungsi Lambda bot.
- `sessionAttributes`Peta hanya muncul jika Anda telah menentukan atribut sesi dalam permintaan.
- `sentimentResponse`Peta hanya muncul jika Anda mengonfigurasi bot untuk mengembalikan nilai sentimen.

> **Note**
>
> Format input dapat berubah tanpa perubahan yang sesuai dalam format`messageVersion`. Kode Anda seharusnya tidak menimbulkan kesalahan jika ada bidang baru.

Mengakses log audio di Amazon S3

Amazon Lex V2 menyimpan log audio untuk percakapan Anda dalam ember S3.

Anda dapat menggunakan konsol Amazon S3 atau API untuk mengakses log audio. Anda dapat melihat awalan key objek S3 dari file audio di konsol Amazon Lex V2, atau di bidang `conversationLogSettings` dalam respons operasi`DescribeBotAlias`.

Memantau status log percakapan dengan CloudWatch metrik

Gunakan Amazon CloudWatch untuk memantau metrik pengiriman log percakapan Anda. Anda dapat mengatur alarm pada metrik sehingga Anda mengetahui masalah dengan pencatatan jika terjadi.

Amazon Lex V2 menyediakan empat metrik di `AWS/Lex` namespace untuk log percakapan:

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

Metrik keberhasilan menunjukkan bahwa Amazon Lex V2 telah berhasil menulis log audio atau teks Anda ke tujuan mereka.

Metrik kegagalan menunjukkan bahwa Amazon Lex V2 tidak dapat mengirimkan log audio atau teks ke tujuan yang ditentukan. Biasanya, ini adalah kesalahan konfigurasi. Jika metrik kegagalan Anda di atas nol, periksa hal berikut:

- Pastikan Amazon Lex V2 adalah entitas tepercaya untuk peran IAM.
- Untuk pencatatan teks, pastikan grup CloudWatch log Log ada. Untuk pencatatan audio, pastikan bucket S3 ada.
- Pastikan bahwa peran IAM yang digunakan Amazon Lex V2 untuk mengakses grup CloudWatch log Log atau bucket S3 memiliki izin menulis untuk grup log atau bucket.
- Pastikan bucket S3 ada di wilayah yang sama dengan bot Amazon Lex V2 dan milik akun Anda.

Mengaburkan nilai slot dalam log percakapan dari Lex V2

Amazon Lex V2 memungkinkan Anda untuk mengaburkan, atau menyembunyikan, isi slot sehingga konten tidak terlihat. Untuk melindungi data sensitif yang diambil sebagai nilai slot, Anda dapat mengaktifkan slot obfuscation untuk menutupi nilai-nilai tersebut untuk logging.

Ketika Anda memilih untuk mengaburkan nilai slot, Amazon Lex V2 menggantikan nilai slot dengan nama slot di log percakapan. Untuk slot yang disebut`full_name`, nilai slot akan dikaburkan sebagai berikut:

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

Jika ucapan berisi karakter braket ({}) Amazon Lex V2 lolos dari karakter braket dengan dua garis miring belakang (\\). Misalnya, teks `{John Stiles}` dikaburkan sebagai berikut:

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

Nilai slot dikaburkan dalam log percakapan. Nilai slot masih tersedia dalam respons dari `RecognizeText` dan `RecognizeUtterance` operasi, dan nilai slot tersedia untuk validasi dan pemenuhan fungsi Lambda Anda. Jika Anda menggunakan nilai slot dalam permintaan atau tanggapan Anda, nilai slot tersebut tidak dikaburkan dalam log percakapan.

Pada giliran pertama percakapan, Amazon Lex V2 mengaburkan nilai slot jika mengenali nilai slot dan slot dalam ucapan. Jika tidak ada nilai slot yang dikenali, Amazon Lex V2 tidak mengaburkan ucapan tersebut.

Pada putaran kedua dan selanjutnya, Amazon Lex V2 tahu slot yang akan diperoleh dan apakah nilai slot harus dikaburkan. Jika Amazon Lex V2 mengenali nilai slot, nilainya dikaburkan. Jika Amazon Lex V2 tidak mengenali nilai, seluruh ucapan dikaburkan. Nilai slot apa pun dalam ucapan yang terlewat tidak akan dikaburkan.

Amazon Lex V2 juga tidak mengaburkan nilai slot yang Anda simpan dalam atribut permintaan atau sesi. Jika Anda menyimpan nilai slot yang harus dikaburkan sebagai atribut, Anda harus mengenkripsi atau mengaburkan nilainya.

Amazon Lex V2 tidak mengaburkan nilai slot dalam audio. Itu mengaburkan nilai slot dalam transkripsi audio.

Anda dapat memilih slot mana yang akan dikaburkan dengan menggunakan konsol atau dengan menggunakan Amazon Lex V2 API. Di konsol, pilih Kekaburan slot di pengaturan untuk slot. Jika Anda menggunakan API, atur `obfuscationSetting` bidang slot ke `DEFAULT_OBFUSCATION` saat Anda memanggil CreateSlotatau UpdateSlotoperasi.

Pengambilan log percakapan selektif di Lex V2

Pengambilan log percakapan selektif memungkinkan pengguna untuk memilih bagaimana log percakapan ditangkap dengan data teks dan audio dari percakapan langsung.

Untuk mengaktifkan dan menangkap output dari fitur pengambilan log percakapan selektif, Anda harus mengaktifkan fitur di konsol Amazon Lex V2, dan mengaktifkan atribut sesi yang diperlukan dalam pengaturan API untuk menangkap output yang dipilih dari log.

Anda dapat memilih opsi berikut untuk pengambilan log percakapan selektif:

- teks saja
- hanya audio
- teks dan audio

Anda dapat menangkap bagian tertentu dari percakapan, dan memilih apakah audio, teks, atau keduanya ditangkap untuk log percakapan.

> **Note**
>
> Pengambilan log percakapan selektif hanya berfungsi untuk Amazon Lex V2.

Topik

- Kelola pengambilan log percakapan selektif
- Contoh pengambilan log percakapan selektif

Kelola pengambilan log percakapan selektif

Menggunakan konsol Lex, Anda dapat mengaktifkan pengaturan pengambilan log percakapan selektif dan memilih slot mana yang ingin Anda aktifkan pengambilan log percakapan selektif.

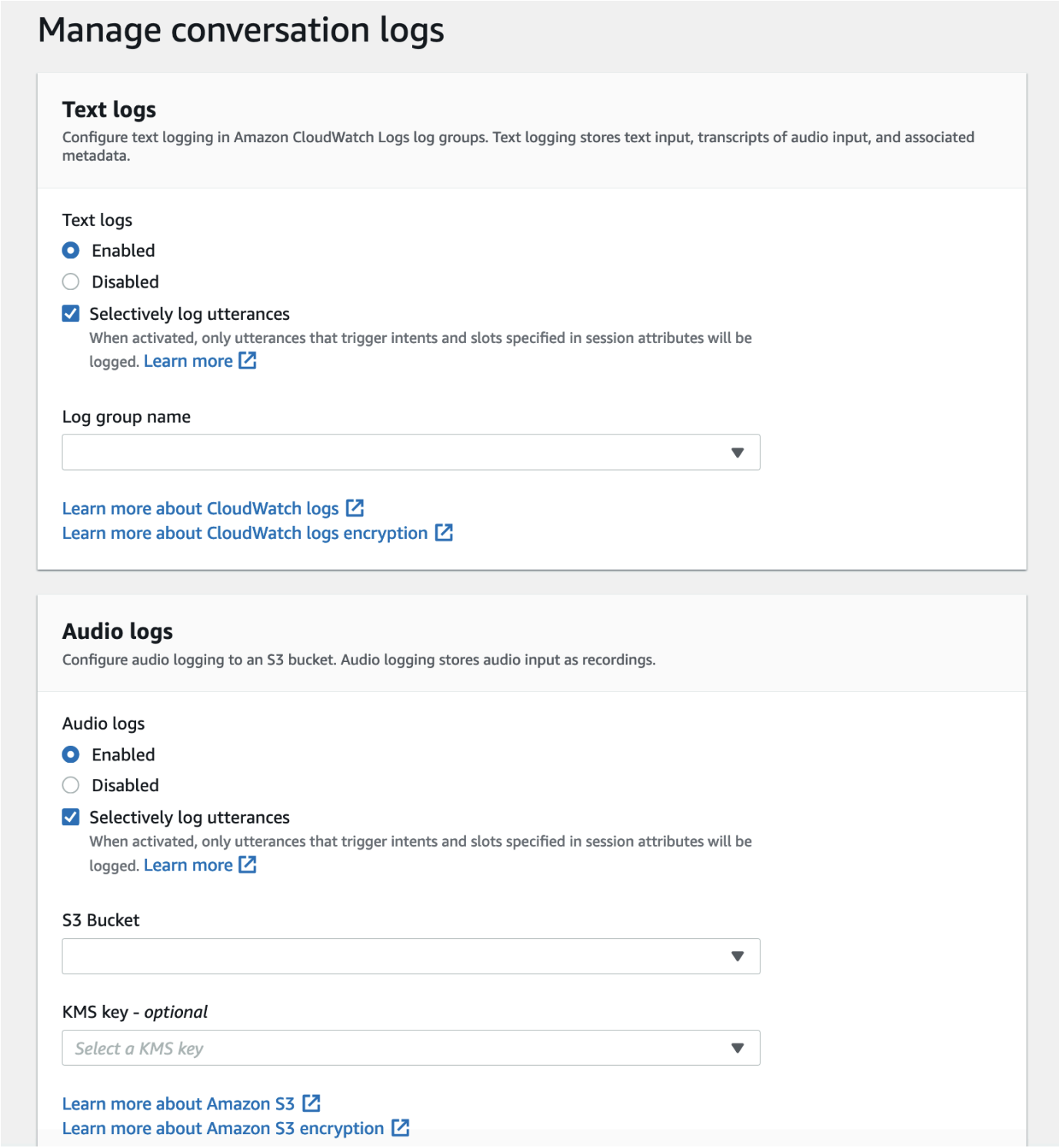
Aktifkan pengambilan log percakapan selektif di konsol Amazon Lex V2:

1. Masuk ke AWS Management Console dan buka konsol Amazon Lex V2 di https://console.aws.amazon.com/lexv2/home.
2. Pilih Bot dari panel sisi kiri dan pilih bot yang ingin Anda aktifkan pengambilan log percakapan selektif. Gunakan bot yang sudah ada atau buat yang baru.
3. Pilih Alias untuk bot yang Anda pilih di bawah bagian Deployment di panel sisi kiri.
4. Pilih Alias bot Anda, lalu pilih Kelola log percakapan.
5. Di panel Kelola log percakapan, untuk log Teks, pilih apakah log teks diaktifkan atau dinonaktifkan dengan memilih tombol radio. Jika Anda memilih Diaktifkan untuk log teks, maka

Anda harus memasukkan nama grup Log atau memilih nama grup log yang ada dari menu tarik-turun. Pilih kotak centang untuk Log ucapan secara selektif jika Anda secara selektif mencatat file teks.

> **Note**
>
> Aktifkan log and/or audio teks dengan memilih kotak centang Ucapan log selektif di pengaturan log percakapan (and/or audio teks) dalam pengaturan waktu pembuatan. BotAlias Anda harus mengonfigurasi grup CloudWatch log dan bucket Amazon S3 untuk memilih opsi ini.

6. Di bagian Log audio, pilih apakah log audio diaktifkan atau dinonaktifkan dengan memilih tombol radio. Jika Anda memilih Diaktifkan untuk log audio, Anda harus menentukan lokasi bucket Amazon S3 dan (opsional) kunci KMS untuk mengenkripsi data audio Anda. Pilih kotak centang untuk Ucapan log selektif jika Anda secara selektif mencatat file audio.

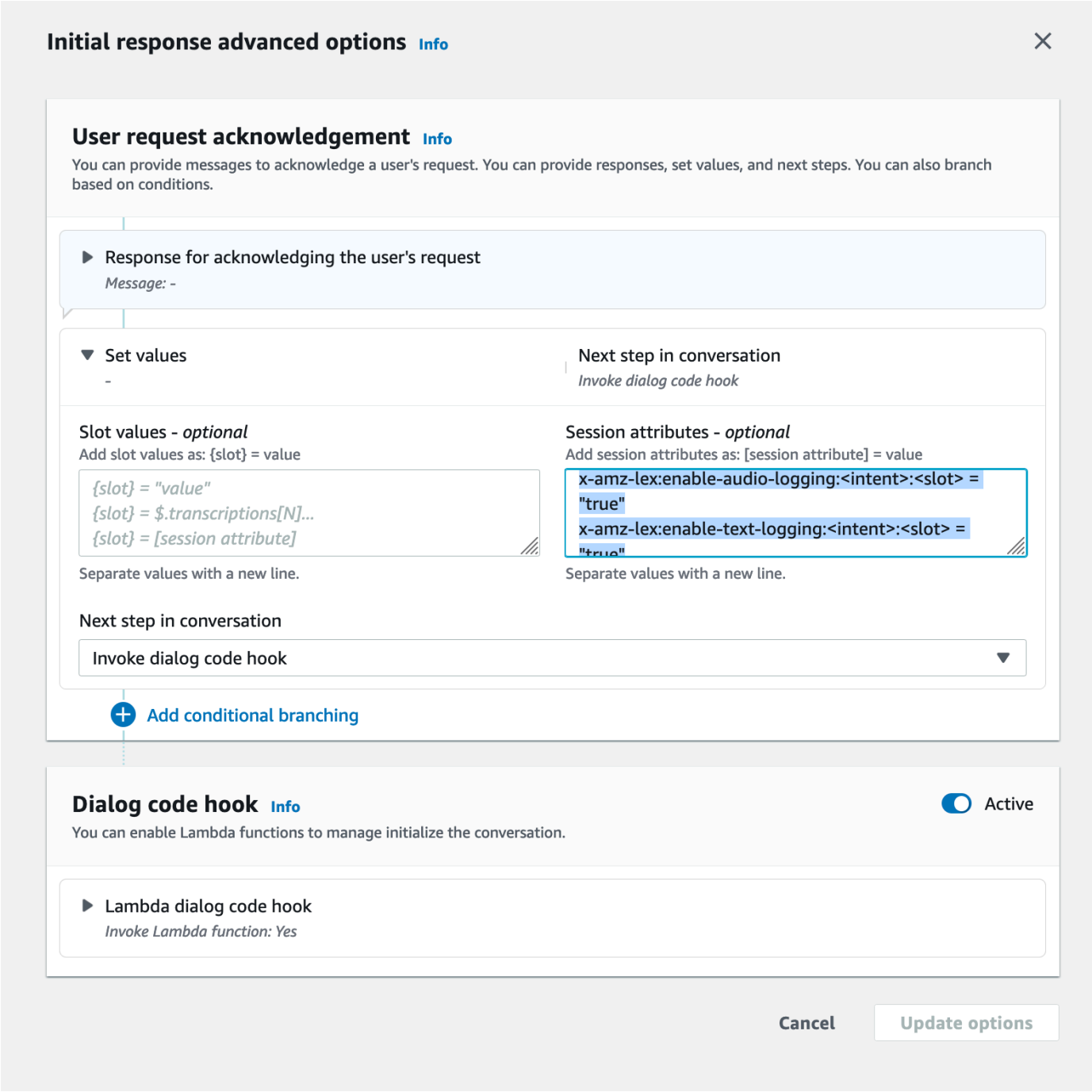


7. Pilih Simpan di sudut kanan bawah panel untuk menyimpan pengaturan pengambilan log percakapan selektif Anda.

Aktifkan pengambilan log percakapan selektif di konsol Amazon Lex V2:

1. Buka Intent dan pilih nama Intent, Initial Response, Advanced Settings, Set Values, Session Attributes.
2. Setel atribut berikut ke berdasarkan maksud dan slot yang ingin Anda aktifkan pengambilan log percakapan selektif:

   - `x-amz-lex:enable-audio-logging:intent:slot = "true"`
   - `x-amz-lex:enable-text-logging:intent:slot = "true"`



Note

Atur `x-amz-lex:enable-audio-logging:`*`intent`*`:`*`slot`*` = "true"` untuk menangkap ucapan yang hanya berisi slot tertentu dalam percakapan. Tindakan untuk mencatat ucapan tergantung pada penilaian*`intent`* : *`slot`* dalam ucapan,

dibandingkan dengan ekspresi atribut sesi, dan nilai flag yang sesuai. Untuk mencatat ucapan, setidaknya satu ekspresi dalam atribut session harus mengizinkannya, dengan flag enable logging disetel ke. `true` Nilai *`intent`* dan *`slot`* bisa `"*"` juga. Jika nilai and/or maksud slot adalah`"*"`, itu berarti bahwa setiap nilai and/or maksud slot `"*"` akan cocok dengannya. Mirip dengan`x-amz-lex:enable-audio-logging`, atribut sesi baru yang disebut `x-amz-lex:enable-text-logging` akan digunakan untuk mengontrol log teks.

3. Pilih opsi Perbarui dan buat bot untuk menyertakan pengaturan yang diperbarui.

> **Note**
>
> Peran IAM Anda harus memiliki izin akses untuk memungkinkan Anda menulis data ke bucket Amazon S3 dan menggunakan kunci KMS untuk mengenkripsi data. Lex akan memperbarui peran IAM Anda dengan izin Lex untuk mengakses grup CloudWatch log Log dan bucket Amazon S3 yang dipilih.

Pedoman untuk menggunakan penangkapan log percakapan selektif:

Anda hanya dapat mengaktifkan penangkapan log percakapan selektif untuk log and/or audio teks, jika Anda telah mengaktifkan log and/or audio teks di pengaturan log Percakapan. Dengan mengaktifkan pengambilan log percakapan selektif untuk log and/or audio teks, Anda menonaktifkan pencatatan untuk semua maksud dan slot dalam percakapan. Untuk menghasilkan log and/or audio teks untuk maksud dan slot tertentu, Anda harus mengatur atribut sesi pengambilan log percakapan selektif or/and audio teks untuk maksud dan slot tersebut ke “true”.

- Jika pengambilan log percakapan selektif diaktifkan, dan tidak ada atribut sesi dengan awalan x-amz-lex:enable-audio-logging yang ada, logging akan dinonaktifkan secara default untuk semua ucapan. Skenario ini juga berlaku mengenai x-amz-lex:enable-text-logging.
- Log ucapan akan disimpan secara eksklusif untuk segmen percakapan and/or audio teks jika setidaknya satu ekspresi dalam atribut sesi mengizinkannya.
- Konfigurasi untuk pengambilan log percakapan selektif and/or audio teks, seperti yang didefinisikan dalam atribut sesi, akan efektif hanya jika pengambilan log percakapan selektif untuk and/or audio teks diaktifkan di Pengaturan Log Percakapan dalam alias bot; jika tidak, atribut sesi akan diabaikan.

- Saat pengambilan log percakapan selektif diaktifkan, nilai slot apa pun di SessionState, Interpretasi, dan Transkripsi yang pendataannya tidak diaktifkan menggunakan atribut sesi akan dikaburkan dalam log teks yang dihasilkan.
- Keputusan untuk menghasilkan log and/or teks audio dievaluasi dengan mencocokkan slot yang ditimbulkan oleh bot dengan atribut sesi pengambilan log percakapan selektif, kecuali untuk giliran elisitasi maksud di mana pengguna dapat memberikan nilai slot bersama dengan elisitasi maksud. Pada giliran elisitasi maksud, slot yang diisi pada giliran saat ini dicocokkan dengan atribut sesi pengambilan log percakapan selektif.
- Slot yang dianggap terisi berasal dari keadaan sesi di akhir giliran. Oleh karena itu, setiap perubahan yang dilakukan oleh Lambda Codehook Dialog ke slot dalam status sesi akan memengaruhi perilaku pengambilan log percakapan selektif.
- Pada giliran elisitasi maksud, jika beberapa nilai slot diberikan oleh pengguna, log and/or audio teks hanya akan dihasilkan jika atribut text/audio sesi memungkinkan pencatatan untuk semua slot yang diisi pada giliran ini.
- Pendekatan operasional yang disarankan adalah mengatur atribut sesi pengambilan log percakapan selektif di awal sesi dan menahan diri untuk tidak memodifikasinya selama sesi.
- Jika ada slot yang berisi data sensitif, Anda harus selalu mengaktifkan pengaburan slot.

Contoh pengambilan log percakapan selektif

Berikut adalah contoh kasus penggunaan bisnis untuk pengambilan log percakapan selektif.

Kasus penggunaan:

Perusahaan fintech menggunakan bot Amazon Lex V2 untuk mendukung sistem IVR mereka, yang memungkinkan pengguna melakukan pembayaran tagihan. Untuk memenuhi persyaratan kepatuhan dan audit, mereka harus menyimpan rekaman audio dari persetujuan otorisasi yang diberikan pengguna. Namun, mengaktifkan log audio umum tidak layak karena akan membuatnya tidak sesuai, karena tidak mungkin untuk mengaburkan slot sensitif seperti, CVV CardNumber, dan informasi lainnya di log audio. Sebagai gantinya, mereka dapat mengaktifkan pengambilan log percakapan selektif untuk log audio dan mengatur atribut sesi untuk hanya menghasilkan log audio untuk ucapan yang memiliki persetujuan otorisasi.

BotAlias Pengaturan:

- Log Teks Diaktifkan: true

- Log Teks Logging Selektif Diaktifkan: false
- Log Audio Diaktifkan: benar
- Logging Selektif Log Audio Diaktifkan: true

Atribut Sesi:

```
x-amz-lex:enable-audio-logging:PayBill:AuthorizationConsent = "true"
```

Contoh Percakapan:

- Pengguna (Input Audio): “Saya ingin membayar tagihan saya dengan nomor tagihan 35XU68.”
- Bot: “Berapa jumlah jatuh tempo dalam dolar?”
- Pengguna (Input Audio): “235.”
- Bot: “Berapa nomor kartu kredit Anda?”
- Pengguna (Masukan Audio): “9239829722200348.”
- Bot: “Anda membayar 235 dolar menggunakan nomor kartu kredit Anda yang diakhiri dengan 0348. Tolong katakan 'Saya memberi wewenang untuk membayar 235 dolar. '”
- Pengguna (Input Audio): “Saya berwenang untuk membayar 235 dolar.”
- Bot: “Tagihan Anda telah dibayar.”

Keluaran Log Percakapan:

Dalam situasi ini, log teks akan diproduksi untuk semua belokan. Namun, log audio hanya akan direkam untuk giliran tertentu ketika AuthorizationConsentslot dalam PayBillmaksud diperoleh, dan tidak ada log audio yang akan diproduksi untuk giliran lainnya.

Kesalahan logging dengan log kesalahan di Lex V2

Anda mengaktifkan log kesalahan untuk menyimpan interaksi bot. Anda dapat menggunakan log kesalahan ini untuk meninjau kinerja bot Anda dan untuk memecahkan masalah kesalahan dengan percakapan.

Log kesalahan dikonfigurasi untuk versi. Setiap versi dapat memiliki pengaturan yang berbeda untuk log kesalahan mereka. Log teks menyimpan masukan teks di CloudWatch Log. Anda dapat mengaktifkan enkripsi log teks menggunakan CMK yang dikelola AWS KMS pelanggan.

Kebijakan IAM untuk Log Kesalahan

Bergantung pada jenis logging yang Anda pilih, Amazon Lex V2 memerlukan izin untuk menggunakan Amazon CloudWatch Logs dan Amazon Simple Storage Service (S3) bucket untuk menyimpan log Anda. Anda harus membuat peran dan izin AWS Identity and Access Management untuk mengaktifkan Amazon Lex V2 mengakses sumber daya ini.

Membuat Peran dan Kebijakan IAM untuk Log Kesalahan

Untuk mengaktifkan log percakapan, Anda harus memberikan izin menulis untuk CloudWatch Log dan Amazon S3. Jika Anda mengaktifkan enkripsi objek untuk objek S3 Anda, Anda perlu memberikan izin akses ke AWS KMS kunci yang digunakan untuk mengenkripsi objek.

Anda dapat menggunakan konsol IAM, API IAM, atau AWS Command Line Interface untuk membuat peran dan kebijakan. Petunjuk ini menggunakan AWS CLI untuk membuat peran dan kebijakan.

Untuk membuat peran IAM untuk log kesalahan

Peran IAM yang Anda gunakan untuk mengaktifkan log percakapan harus memiliki izin iam:PassRole . Kebijakan berikut harus dilampirkan pada peran:

JSON

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

Mengaktifkan Log Kesalahan di Lex V2

Untuk mengaktifkan log kesalahan menggunakan konsol Amazon Lex V2:

1. Masuk ke AWS Management Console dan buka konsol Amazon Lex di https://console.aws.amazon.com/lex/.

2. Dari daftar Bot, pilih bot yang ingin Anda aktifkan untuk log kesalahan.
3. Dari menu sebelah kiri, pilih Versi.
4. Dalam daftar Versi, pilih Versi yang ingin Anda konfigurasikan log kesalahan.
5. Di bagian Detail versi, pilih Aktifkan.
6. Pilih Simpan untuk mulai mencatat percakapan. Jika perlu, Amazon Lex V2 akan memperbarui peran layanan Anda dengan izin untuk mengakses grup CloudWatch log Log.

Menonaktifkan Log Kesalahan di Lex V2

Untuk mematikan log kesalahan menggunakan konsol Amazon Lex V2:

1. Masuk ke AWS Management Console dan buka konsol Amazon Lex di https://console.aws.amazon.com/lex/.
2. Dari daftar Bot, pilih bot yang ingin Anda aktifkan untuk log kesalahan.
3. Dari menu sebelah kiri, pilih Versi.
4. Dalam daftar Versi, pilih Versi yang ingin Anda konfigurasikan log kesalahan.
5. Di bagian Detail versi, pilih Nonaktifkan.
6. Pilih Simpan untuk menghentikan pencatatan percakapan.

Memantau metrik operasional di Lex V2

Amazon CloudWatch dan AWS CloudTrail merupakan dua layanan AWS yang terintegrasi dengan Amazon Lex V2 untuk membantu Anda memantau interaksi pengguna dengan bot Anda. Gunakan layanan ini untuk merekam tindakan, mengirim data mendekati waktu nyata, dan mengatur pemberitahuan dan tindakan otomatis saat kriteria terpenuhi.

Topik

- Mengukur metrik operasional dengan Amazon CloudWatch
- Melihat acara dengan AWS CloudTrail para Lex V2

Mengukur metrik operasional dengan Amazon CloudWatch

Anda dapat memantau Amazon Lex V2 menggunakan CloudWatch, yang mengumpulkan data mentah dan memprosesnya menjadi metrik yang dapat dibaca, mendekati waktu nyata. Statistik

ini disimpan untuk jangka waktu 15 bulan, sehingga Anda dapat mengakses informasi historis dan mendapatkan perspektif yang lebih baik tentang performa aplikasi atau layanan web Anda. Anda juga dapat mengatur alarm yang memperhatikan ambang batas tertentu dan mengirim notifikasi atau mengambil tindakan saat ambang batas tersebut terpenuhi. Untuk informasi selengkapnya, lihat Panduan CloudWatch Pengguna Amazon.

Layanan Amazon Lex V2 melaporkan metrik berikut di `AWS/Lex` namespace.

Metrik	Deskripsi
`AssistedSlotResolutionModelAccessDeniedErrorCount`	Berapa kali Amazon Lex V2 ditolak aksesnya ke Amazon Bedrock Dimensi yang valid untuk `RecognizeUtterance` dan `StartConversation` operasi: • BotId, BotAliasId, LocaleId, Operasi, InputMode, ModelType, Model • BotId, BotVersion, LocaleId, Operasi, InputMode, ModelType, Model Dimensi yang valid untuk`RecognizeText` : • BotId, BotAliasId, LocaleId, Operasi, ModelType, Model • BotId, BotVersion, LocaleId, Operasi, ModelType, Model Unit: Jumlah
`AssistedSlotResolutionModelInvocationCount`	Berapa kali Amazon Bedrock dipanggil. Dimensi yang valid untuk `RecognizeUtterance` dan `StartConversation` operasi: • BotId, BotAliasId, LocaleId, Operasi, InputMode, ModelType, Model • BotId, BotVersion, LocaleId, Operasi, InputMode, ModelType, Model Dimensi yang valid untuk`RecognizeText` : • BotId, BotAliasId, LocaleId, Operasi, ModelType, Model • BotId, BotVersion, LocaleId, Operasi, ModelType, Model

Metrik	Deskripsi
	Unit: Jumlah
AssistedSlotResolutionModelSystemErrorCount	Berapa kali 5xx terjadi saat menelepon Amazon Bedrock. Dimensi yang valid untuk RecognizeUtterance dan StartConversation operasi: • BotId, BotAliasId, LocaleId, Operasi, InputMode, ModelType, Model • BotId, BotVersion, LocaleId, Operasi, InputMode, ModelType, Model Dimensi yang valid untukRecognizeText : • BotId, BotAliasId, LocaleId, Operasi, ModelType, Model • BotId, BotVersion, LocaleId, Operasi, ModelType, Model Unit: Jumlah
AssistedSlotResolutionModelThrottlingErrorCount	Berapa kali Amazon Lex dibatasi oleh Amazon Bedrock. Dimensi yang valid untuk RecognizeUtterance dan StartConversation operasi: • BotId, BotAliasId, LocaleId, Operasi, InputMode, ModelType, Model • BotId, BotVersion, LocaleId, Operasi, InputMode, ModelType, Model Dimensi yang valid untukRecognizeText : • BotId, BotAliasId, LocaleId, Operasi, ModelType, Model • BotId, BotVersion, LocaleId, Operasi, ModelType, Model Unit: Jumlah

Metrik	Deskripsi
`AssistedSlotResolutionResolvedSlotCount`	Berapa kali Amazon Bedrock mengembalikan nilai slot. Dimensi yang valid untuk `RecognizeUtterance` dan `StartConversation` operasi: • BotId, BotAliasId, LocaleId, Operasi, InputMode, ModelType, Model • BotId, BotVersion, LocaleId, Operasi, InputMode, ModelType, Model Dimensi yang valid untuk`RecognizeText` : • BotId, BotAliasId, LocaleId, Operasi, ModelType, Model • BotId, BotVersion, LocaleId, Operasi, ModelType, Model Unit: Jumlah
`KendraIndexAccessError`	Berapa kali Amazon Lex V2 tidak dapat mengakses indeks Amazon Kendra Anda. • Operasi, BotId, BotAliasId, LocaleId Unit: Jumlah
`KendraLatency`	Jumlah waktu yang dibutuhkan Amazon Kendra untuk menanggapi permintaan dari. `AMAZON.KendraSearchIntent` Dimensi yang valid: • Operasi, BotId, BotVersion, LocaleId • Operasi, BotId, BotAliasId, LocaleId Satuan: Milidetik

Metrik	Deskripsi
KendraSuccess	Berapa kali Amazon Lex V2 tidak dapat mengakses indeks Amazon Kendra Anda. Dimensi yang valid: • Operasi, BotId, BotVersion, LocaleId • Operasi, BotId, BotAliasId, LocaleId Unit: Jumlah
KendraSystemErrors	Berapa kali Amazon Lex V2 tidak dapat menanyakan indeks Amazon Kendra. Dimensi yang valid: • Operasi, BotId, BotAliasId, InputMode, LocaleId Unit: Jumlah
KendraThrottledEvents	Berapa kali Amazon Kendra membatasi permintaan dari. AMAZON.KendraSearchIntent Dimensi yang valid: • Operasi, BotId, BotAliasId, InputMode. LocaleId Unit: Jumlah

Metrik	Deskripsi
RuntimeConcurrency	Jumlah koneksi bersamaan dalam periode waktu yang ditentukan. RuntimeConcurrency dilaporkan sebagai aStatisticSet . Dimensi yang valid untuk RecognizeUtterance atau StartConversation operasi: • Operasi, BotId, BotVersion, InputMode, LocaleId • Operasi, BotId, BotAliasId, InputMode, LocaleId Dimensi yang valid untuk operasi lain: • Operasi, BotId, BotVersion, LocaleId • Operasi, BotId, BotAliasId, LocaleId Unit: Jumlah
RuntimeInvalidLambdaResponses	Jumlah AWS Lambda tanggapan yang tidak valid dalam periode yang ditentukan. Dimensi yang valid: • Operasi, BotId, BotAliasId, InputMode, LocaleId Unit: Jumlah
RuntimeLambdaErrors	Jumlah kesalahan runtime Lambda dalam periode waktu yang ditentukan. Dimensi yang valid: • Operasi, BotId, BotAliasId, InputMode, LocaleId Unit: Jumlah

Metrik	Deskripsi
`RuntimePollyErrors`	Jumlah tanggapan Amazon Polly yang tidak valid dalam periode waktu yang ditentukan. Dimensi yang valid: • Operasi, BotId, BotAliasId, InputMode, LocaleId Unit: Jumlah
`RuntimeRequestCount`	Jumlah permintaan runtime dalam periode waktu yang ditentukan. Dimensi yang valid untuk `RecognizeUtterance` dan `StartConversation` operasi: • Operasi, BotId, BotVersion, InputMode, LocaleId • Operasi, BotId, BotAliasId, InputMode, LocaleId Dimensi yang valid untuk operasi lain: • Operasi, BotId, BotVersion, LocaleId • Operasi, BotId, BotAliasId, LocaleId Unit: Jumlah
`RuntimeRequestLength`	Total panjang percakapan dengan bot Amazon Lex V2. Hanya berlaku untuk StartConversationoperasi. Dimensi yang valid: • BotAliasId, BotId, LocaleId, Operasi • BotId, BotAliasId, LocaleId, Operasi Satuan: milidetik

Metrik	Deskripsi
`RuntimeSuccessfulRequestLatency` **Important** Metrik ini adalah `RuntimeSuccessfulRequestLatency` dan tidak `RuntimeSuccessfulRequestLatency`.	Latensi untuk permintaan yang berhasil antara waktu permintaan dibuat dan respons diteruskan kembali. Dimensi yang valid untuk `RecognizeUtterance` dan `StartConversation` operasi: • Operasi, BotId, BotVersion, InputMode, LocaleId • Operasi, BotId, BotAliasId, InputMode, LocaleId Dimensi yang valid untuk operasi lain: • Operasi, BotId, BotVersion, LocaleId • Operasi, BotId, BotAliasId, LocaleId Satuan: milidetik
`RuntimeSystemErrors`	Jumlah kesalahan sistem dalam periode yang ditentukan. Rentang kode respons untuk kesalahan sistem adalah 500 hingga 599. Dimensi yang valid untuk `RecognizeUtterance` dan `StartConversation` operasi: • Operasi, BotId, BotVersion, InputMode, LocaleId • Operasi, BotId, BotAliasId, InputMode, LocaleId Dimensi yang valid untuk operasi lain: • Operasi, BotId, BotVersion, LocaleId • Operasi, BotId, BotAliasId, LocaleId Unit: Jumlah

Metrik	Deskripsi
`RuntimeThrottledEvents`	Jumlah peristiwa yang dibatasi. Amazon Lex V2 membatasi peristiwa ketika menerima lebih banyak permintaan daripada batas transaksi per detik yang ditetapkan untuk akun Anda. Jika batas yang ditetapkan untuk akun Anda sering terlampaui, Anda dapat meminta penambahan kuota. Untuk meminta peningkatan, lihat batas layanan AWS. Dimensi yang valid untuk `RecognizeUtterance` dan `StartConversation` operasi: • Operasi, BotId, BotVersion, InputMode, LocaleId • Operasi, BotId, BotAliasId, InputMode, LocaleId Dimensi yang valid untuk operasi lain: • Operasi, BotId, BotVersion, LocaleId • Operasi, BotId, BotAliasId, LocaleId Unit: Jumlah
`RuntimeUserErrors`	Jumlah kesalahan pengguna dalam periode yang ditentukan. Kisaran kode respons untuk kesalahan pengguna adalah 400 hingga 499. Dimensi yang valid untuk `RecognizeUtterance` dan `StartConversation` operasi: • Operasi, BotId, BotVersion, InputMode, LocaleId • Operasi, BotId, BotAliasId, InputMode, LocaleId Dimensi yang valid untuk operasi lain: • Operasi, BotId, BotVersion, LocaleId • Operasi, BotId, BotAliasId, LocaleId Unit: Jumlah

Dimensi berikut didukung untuk metrik Amazon Lex V2.

Dimensi	Deskripsi
`Operation`	Nama operasi Amazon Lex V2 —`RecognizeText` ,`RecognizeUtterance` ,`StartConversation` , `GetSession` `PutSession` , `DeleteSession` — yang menghasilkan entri.
`BotId`	Pengidentifikasi unik alfanumerik untuk bot.
`BotAliasId`	Pengidentifikasi unik alfanumerik untuk alias bot.
`BotVersion`	Versi numerik bot.
`InputMode`	Jenis input ke bot — ucapan, teks, atau DTMF.
`LocaleId`	Pengidentifikasi lokal bot, seperti en-US atau fr-CA.
`Model`	Menunjukkan id model model bahasa besar Amazon Bedrock.
`ModelType`	Menunjukkan jenis model bahasa besar yang dipanggil dari Amazon Bedrock.

Melihat acara dengan AWS CloudTrail para Lex V2

Amazon Lex V2 terintegrasi dengan AWS CloudTrail, layanan yang menyediakan catatan tindakan yang diambil oleh pengguna, peran, atau layanan AWS di Amazon Lex V2. CloudTrail menangkap panggilan API untuk Amazon Lex V2 sebagai acara. Panggilan yang diambil termasuk panggilan dari konsol Amazon Lex V2 dan panggilan kode ke operasi Amazon Lex V2 API. Jika Anda membuat jejak, Anda dapat mengaktifkan pengiriman CloudTrail acara secara terus menerus ke bucket Amazon S3, termasuk acara untuk Amazon Lex V2. Jika Anda tidak membuat konfigurasi jejak, Anda masih dapat melihat kejadian terbaru dalam konsol CloudTrail di Riwayat peristiwa. Dengan menggunakan informasi yang dikumpulkan oleh CloudTrail, Anda dapat menentukan permintaan yang dibuat untuk Amazon Lex V2, alamat IP dari mana permintaan itu dibuat, siapa yang membuat permintaan, kapan dibuat, dan detail tambahan.

Untuk mempelajari selengkapnya CloudTrail, lihat Panduan AWS CloudTrail Pengguna.

Informasi Amazon Lex V2 di CloudTrail

CloudTrail diaktifkan di akun AWS Anda saat Anda membuat akun. Saat aktivitas terjadi di Amazon Lex V2, aktivitas tersebut direkam dalam suatu CloudTrail peristiwa bersama dengan peristiwa layanan AWS lainnya dalam riwayat Acara. Anda dapat melihat, mencari, dan mengunduh kejadian terbaru di akun AWS Anda. Untuk informasi selengkapnya, lihat Melihat peristiwa dengan Riwayat CloudTrail acara.

Untuk catatan peristiwa yang sedang berlangsung di akun AWS Anda, termasuk peristiwa untuk Amazon Lex V2, buat jejak. Jejak memungkinkan CloudTrail untuk mengirim berkas log ke bucket Amazon S3. Secara default, ketika Anda membuat jejak di konsol tersebut, jejak tersebut diterapkan ke semua Wilayah AWS. Jejak mencatat kejadian dari semua Wilayah di partisi AWS dan mengirimkan berkas log ke bucket Amazon S3 yang Anda tentukan. Selain itu, Anda dapat mengonfigurasi layanan AWS lainnya untuk menganalisis lebih lanjut dan menindaklanjuti data kejadian yang dikumpulkan di log CloudTrail . Untuk informasi selengkapnya, lihat berikut:

- Gambaran umum untuk membuat jejak
- CloudTrail layanan dan integrasi yang didukung
- Mengonfigurasi notifikasi Amazon SNS untuk CloudTrail
- Menerima file CloudTrail log dari beberapa wilayah dan Menerima file CloudTrail log dari beberapa akun

Amazon Lex V2 mendukung pencatatan untuk semua tindakan yang tercantum dalam Model Building API V2.

Setiap entri peristiwa atau log berisi informasi tentang entitas yang membuat permintaan tersebut. Informasi identitas membantu Anda menentukan hal berikut ini:

- Apakah permintaan dibuat dengan root atau kredensil pengguna AWS Identity and Access Management IAM.
- Permintaan tersebut dibuat dengan kredensial keamanan sementara untuk peran atau pengguna gabungan atau tidak.
- Jika permintaan tersebut dibuat oleh layanan AWS lainnya.

Untuk informasi selengkapnya, lihat Elemen userIdentity CloudTrail .

Memahami entri file log Amazon Lex V2

Trail adalah konfigurasi yang memungkinkan pengiriman peristiwa sebagai file log ke bucket Amazon S3 yang Anda tentukan. CloudTrail file log berisi satu atau lebih entri log. Peristiwa mewakili permintaan tunggal dari sumber mana pun dan mencakup informasi tentang tindakan yang diminta, tanggal dan waktu tindakan, parameter permintaan, dan sebagainya. CloudTrail file log bukanlah jejak tumpukan yang diurutkan dari panggilan API publik, jadi file tersebut tidak muncul dalam urutan tertentu.

Contoh berikut menunjukkan entri CloudTrail log yang menunjukkan CreateBotAliastindakan.

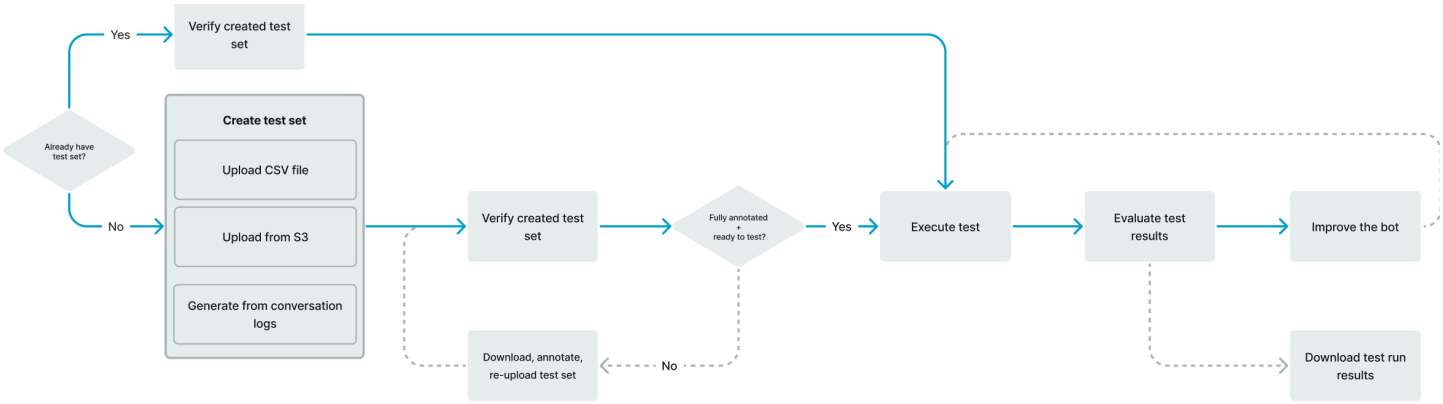
```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "ID of caller:temporary credentials",
        "arn": "arn:aws:sts::111122223333:assumed-role/role name/role ARN",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "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"
}
```

Mengevaluasi kinerja bot Lex V2 dengan Test Workbench

Untuk meningkatkan kinerja bot, Anda dapat mengevaluasi kinerja bot Anda dalam skala besar. Hasil untuk evaluasi pengujian Anda ditampilkan dalam tabel dan bagan sederhana.

Anda dapat menggunakan Meja Kerja Uji untuk membuat set pengujian referensi yang menggunakan data transkripsi yang ada. Anda dapat menguji bot untuk mengevaluasi kinerja sebelum penerapan, dan melihat kerusakan hasil pengujian dalam skala besar.



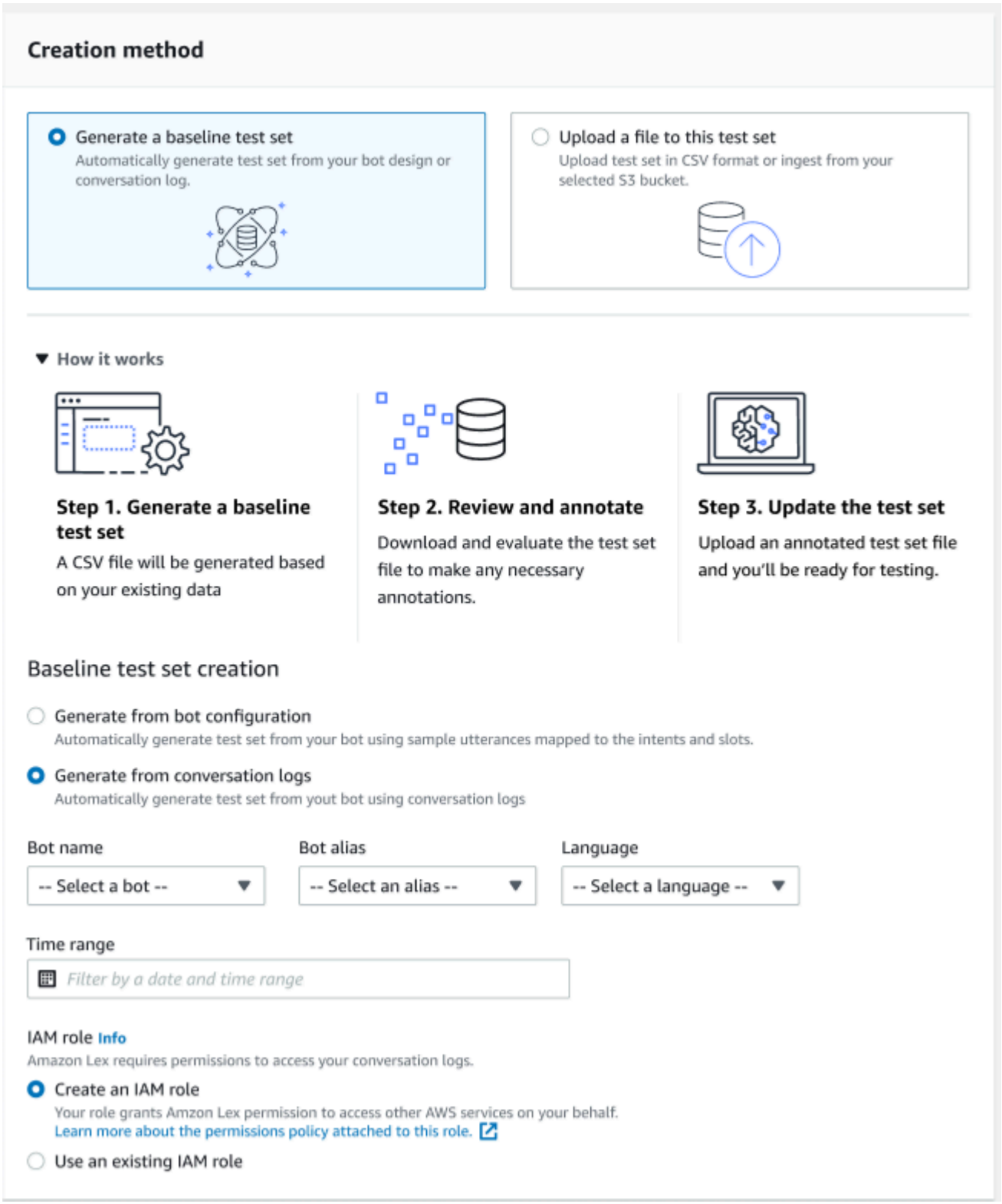
Pengguna dapat menggunakan Test Workbench untuk menetapkan kinerja dasar untuk bot. Ini mencakup kinerja maksud dan slot untuk ucapan yang dalam bentuk input tunggal atau percakapan. Setelah set pengujian berhasil dimuat, Anda dapat menjalankannya terhadap bot pra-produksi atau produksi yang ada. Meja Kerja Uji membantu Anda mengidentifikasi peluang untuk pengisian slot yang lebih baik dan klasifikasi maksud.

Topik

- Menghasilkan set tes untuk Test Workbench
- Kelola set tes
- Jalankan tes
- Cakupan set uji di Meja Kerja Uji
- Lihat hasil tes
- Rincian hasil tes di Test Workbench

Menghasilkan set tes untuk Test Workbench

Anda dapat membuat set pengujian untuk mengevaluasi kinerja bot Anda. Buat set pengujian dengan mengunggah set pengujian yang dalam format file CSV atau dengan membuat set pengujian dari log percakapan. Set tes dapat berisi input audio atau teks.



Jika set pengujian membuat kesalahan validasi, hapus set pengujian dan ganti dengan daftar data set pengujian lainnya, atau edit data dalam file CSV dengan menggunakan program pengeditan spreadsheet.

Untuk membuat set pengujian:

1. Masuk ke AWS Management Console dan buka konsol Amazon Lex di https://console.aws.amazon.com/lex/.
2. Pilih Test workbench dari panel sisi kiri.
3. Pilih Set uji dari opsi di bawah meja kerja Uji.
4. Pilih tombol Create test set di konsol.
5. Dalam Detail, masukkan nama set pengujian dan deskripsi opsional.
6. Pilih Hasilkan set uji dasar.
7. Pilih Hasilkan dari log percakapan.
8. Pilih nama Bot, Alias Bot, dan Bahasa dari menu drop-down.
9. Jika Anda membuat tes dasar dari log percakapan, pilih Rentang waktu dan peran IAM, jika diperlukan. Anda dapat membuat peran dengan izin dasar Amazon Lex V2 atau menggunakan peran yang ada.
10. Pilih modalitas Audio atau Teks untuk set pengujian yang Anda buat. CATATAN: Meja Kerja Uji dapat mengimpor file teks hingga 50k, dan audio hingga 5 jam.
11. Pilih lokasi Amazon S3 untuk menyimpan hasil pengujian, dan tambahkan kunci KMS opsional untuk mengenkripsi transkrip keluaran.
12. Pilih Buat.

Untuk mengunggah set pengujian yang ada dalam format file CSV, atau memperbarui set pengujian:

1. Pilih Test workbench dari panel sisi kiri.
2. Pilih Set uji dari opsi di bawah meja kerja Uji.
3. Pilih Unggah file ke set pengujian ini di konsol.
4. Pilih Unggah dari bucket Amazon S3 Amazon atau Unggah dari komputer Anda. CATATAN: Anda dapat mengunggah file CSV yang dibuat dari templat. Klik template CSV untuk mengunduh file zip yang berisi templat.
5. Pilih Buat peran dengan izin Amazon Lex dasar atau Gunakan peran yang ada untuk Peran ARN.
6. Pilih modalitas Audio atau Teks untuk set pengujian yang Anda buat. CATATAN: Meja Kerja Uji dapat mengimpor file teks hingga 50k, dan audio hingga 5 jam.

7. Pilih lokasi Amazon S3 untuk menyimpan hasil pengujian, dan tambahkan kunci KMS opsional untuk mengenkripsi transkrip keluaran.
8. Pilih Buat.

Jika operasi berhasil, pesan konfirmasi akan menunjukkan bahwa set tes siap untuk diuji, dan status akan ditampilkan Siap untuk pengujian.

Kiat untuk membuat set tes yang sukses

- Anda dapat membuat peran IAM untuk Test Workbench di konsol, atau Anda dapat mengonfigurasi peran IAM Anda selangkah demi selangkah. Untuk informasi selengkapnya, lihat Membuat peran IAM untuk Meja Kerja Uji.
- Sebelum menjalankan pengujian, validasi set pengujian dan definisi bot untuk setiap inkonsistensi menggunakan tombol Validasi perbedaan. Jika konvensi penamaan maksud dan slot yang digunakan dalam set pengujian konsisten dengan bot, lanjutkan untuk menjalankan pengujian. Jika ada anomali yang diidentifikasi, revisi set pengujian, perbarui set pengujian, dan pilih Validasi perbedaan. Ulangi urutan ini lagi sampai tidak ada inkonsistensi yang dicatat, lalu jalankan tes.
- Meja Kerja Uji dapat menguji dengan format nilai slot yang berbeda di kolom Slot Output yang Diharapkan. Untuk slot bawaan apa pun, Anda dapat memilih nilai yang diberikan dalam input pengguna (misalnya, Tanggal = besok), atau memberikan nilai penyelesaian absolutnya (misalnya, Tanggal = 2023-03-21). Untuk informasi lebih lanjut seputar slot bawaan dan nilai absolutnya, lihat Built-inslot.
- Untuk konsistensi dan keterbacaan di kolom Slot Output yang Diharapkan, ikuti konvensi "SlotName = SlotValue" (misalnya, AppointmentType = pembersihan) dengan spasi sebelum dan sesudah tanda sama dengan.
- Jika bot menyertakan slot komposit, di Slot Output yang Diharapkan tentukan subslot ke nama slot, dipisahkan oleh titik (misalnya, “Car.Color”). Tidak ada sintaks dan tanda baca lain yang akan berfungsi.
- Jika bot menyertakan slot multi-nilai, di Slot Output yang Diharapkan memberikan beberapa nilai slot, dipisahkan dengan koma (“FlowerType = mawar, bunga lili”). Tidak ada sintaks dan tanda baca lain yang akan berfungsi.
- Pastikan set pengujian dibuat dari log percakapan yang valid.
- Slot:slot nilai akan berada di kolom yang sama setelah kolom maksud dalam format CSV.
- Input DTMF dari giliran Pengguna ditafsirkan sebagai transkripsi yang diharapkan dan tidak mencantumkan lokasi Amazon S3.

Membuat kasus uji dalam set pengujian menggunakan Test Workbench

Hasil Test Workbench tergantung pada definisi bot dan set pengujian yang sesuai. Anda dapat membuat set pengujian dengan informasi dari definisi bot untuk menentukan area yang perlu ditingkatkan. Buat kumpulan data pengujian dengan contoh yang Anda curigai (atau ketahui) akan menantang bot untuk menafsirkan dengan benar mengingat desain bot saat ini dan pengetahuan Anda tentang percakapan pelanggan Anda.

Tinjau maksud Anda berdasarkan pembelajaran dari bot produksi Anda secara teratur. Terus tambahkan dan sesuaikan contoh ucapan bot dan nilai slot. Pertimbangkan untuk meningkatkan resolusi slot dengan menggunakan opsi yang tersedia, seperti petunjuk runtime. Desain dan pengembangan bot Anda adalah proses berulang yang merupakan siklus berkelanjutan.

Berikut adalah beberapa tips lain untuk mengoptimalkan set tes Anda:

- Pilih kasus penggunaan yang paling umum dengan maksud dan slot yang sering digunakan di set pengujian.
- Jelajahi berbagai cara pelanggan dapat merujuk pada maksud dan slot Anda. Ini dapat mencakup input pengguna dalam bentuk pernyataan, pertanyaan, dan perintah yang panjangnya bervariasi dari minimal hingga diperpanjang.
- Sertakan input pengguna dengan jumlah slot yang bervariasi.
- Sertakan sinonim atau singkatan dari nilai slot khusus yang umum digunakan yang didukung oleh bot Anda (misalnya, “saluran akar”, “kanal”, atau “RC”).
- Sertakan variasi nilai slot bawaan (misalnya, “besok”, “secepatnya”, atau “hari berikutnya”).
- Periksa ketahanan bot untuk modalitas lisan dengan mengumpulkan input pengguna yang dapat disalahartikan (misalnya, “tinta”, “pergelangan kaki”, atau “jangkar”).

Membuat set pengujian dari file CSV untuk Test Workbench

Anda dapat membuat set pengujian dari template file CSV yang disediakan di konsol Amazon Lex V2 dengan memasukkan nilai secara langsung menggunakan editor spreadsheet CSV. Set pengujian adalah file nilai dipisahkan koma (CSV) yang terdiri dari ucapan pengguna tunggal dan percakapan multi-putaran yang direkam dalam kolom berikut:

- Baris # - kolom ini adalah penghitung tambahan yang melacak total baris yang diisi untuk diuji.

- Percakapan # — kolom ini melacak jumlah putaran dalam percakapan. Untuk input tunggal, kolom ini dapat dibiarkan kosong, diisi dengan “-” atau "N/A”. Untuk percakapan, setiap giliran dalam percakapan akan diberi nomor percakapan yang sama.
- Sumber - kolom ini diatur ke “Pengguna” atau “Agen”. Untuk input tunggal, itu akan selalu diatur ke “Pengguna”.
- Input - kolom ini mencakup ucapan pengguna atau permintaan bot.
- Intent Output yang Diharapkan - kolom ini menangkap maksud yang terpenuhi dalam input.
- Intent Expected Output Slot 1 — kolom ini menangkap slot pertama yang ditimbulkan dalam input pengguna. Set tes harus menyertakan kolom yang disebut Expected Output Slot X untuk setiap slot di input pengguna.

Contoh set uji dengan input tunggal:

Baris #	Percakapan #	Sumber	Input	Maksud Output yang Diharapkan	Slot Output yang Diharapkan 1	Slot Output yang Diharapkan 2
1		Pengguna	pesan janji pembersihan besok	MakeAppointment	AppointmentType = membersihkan	Tanggal = besok
2	N/A	Pengguna	memesan janji pembersihan pada 15 April	MakeAppointment	AppointmentType = membersihkan	Tanggal = 4/15 /23
3	N/A	Pengguna	janji temu untuk bulan Desember pertama	MakeAppointment	Tanggal = Desember pertama	

Baris #	Percakapan #	Sumber	Input	Maksud Output yang Diharapkan	Slot Output yang Diharapkan 1	Slot Output yang Diharapkan 2
4	N/A	Pengguna	memesan janji pembersihan	MakeAppointment	AppointmentType = membersihkan	
1		Pengguna	Bisakah Anda membantu saya membuat janji?	MakeAppointment		

Contoh set tes dengan percakapan

Baris #	Percakapan #	Sumber	Input	Maksud Output yang Diharapkan	Slot Output yang Diharapkan 1	Slot Output yang Diharapkan 2	Slot Output yang Diharapkan 3
1	1	Pengguna	memesan janji	MakeAppointment			
2	1	Agen	Jenis janji apa yang ingin Anda jadwalkan?	MakeAppointment			

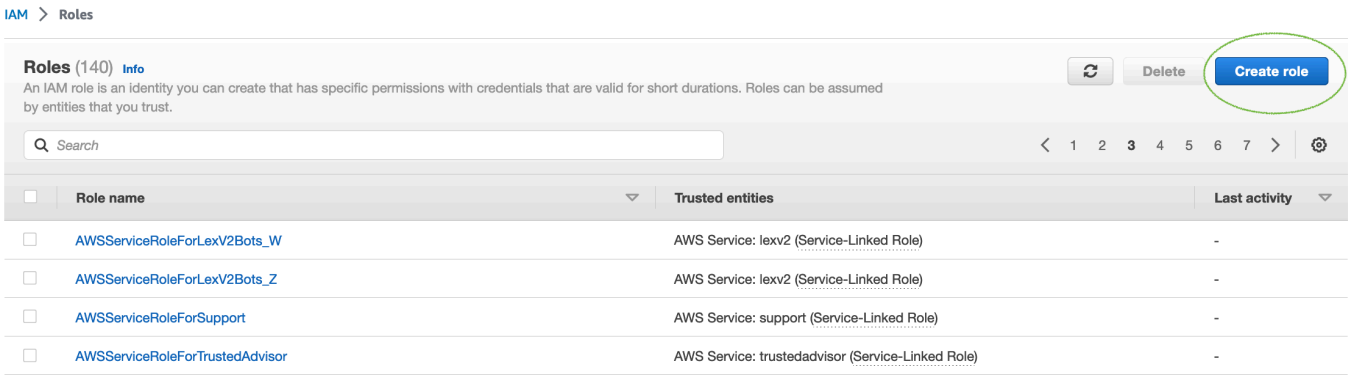
Baris #	Percakapan #	Sumber	Input	Maksud Output yang Diharapkan	Slot Output yang Diharapkan 1	Slot Output yang Diharapkan 2	Slot Output yang Diharapkan 3
3	1	Pengguna	pembersihan	MakeAppointment	AppointmentType = membersihkan		
4	1	Agen	Kapan saya harus menjadwalkan janji temu Anda?	MakeAppointment			
5	1	Pengguna	besok	MakeAppointment		Tanggal = besok	
6	2	Pengguna	pesan janji saluran akar hari ini	MakeAppointment	AppointmentType = saluran akar	Tanggal = hari ini	
7	2	Agen	Pada jam berapa saya harus menjadwalkan janji temu Anda?	MakeAppointment			

Baris #	Percakapan #	Sumber	Input	Maksud Output yang Diharapkan	Slot Output yang Diharapkan 1	Slot Output yang Diharapkan 2	Slot Output yang Diharapkan 3
8	2	Pengguna	sebelas pagi	MakeAppointment			Waktu = sebelas pagi

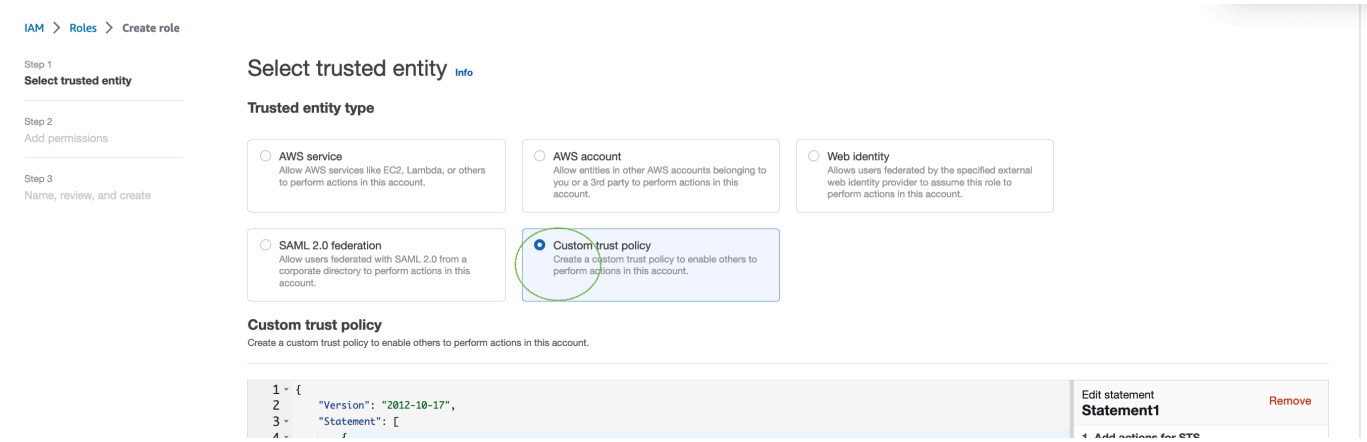
Buat peran IAM untuk Test Workbench

Untuk membuat peran IAM untuk Test Workbench

1. Ikuti langkah-langkah di Buat pengguna IAM untuk membuat pengguna IAM yang dapat digunakan untuk mengakses konsol meja kerja uji.
2. Pilih tombol Buat peran.



3. Pilih opsi untuk Kebijakan kepercayaan khusus.



4. Masukkan kebijakan kepercayaan di bawah ini dan klik Berikutnya.

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "sid4",
      "Effect": "Allow",
      "Principal": {
        "Service": "lexv2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

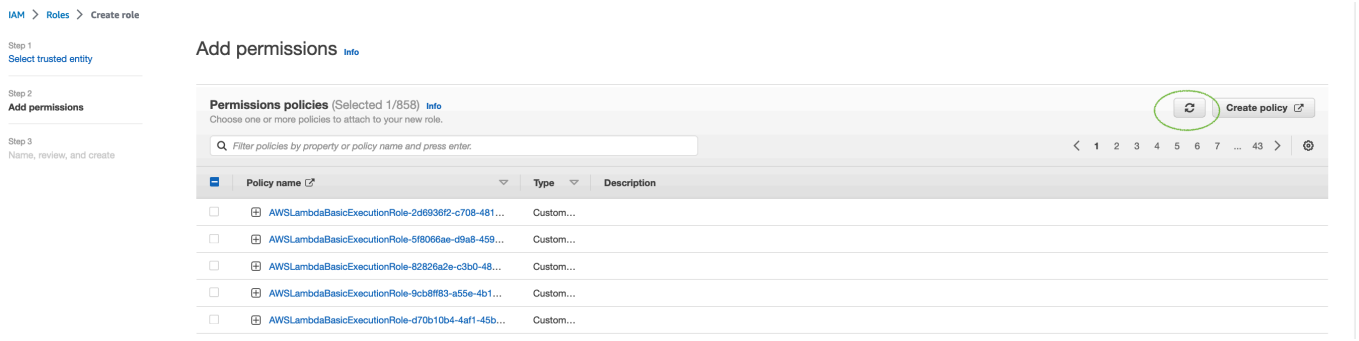
5. Pilih tombol Buat kebijakan.
6. Tab baru akan terbuka di browser Anda di mana Anda dapat memasukkan kebijakan di bawah ini dan klik tombol Berikutnya: Tag.

   JSON

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

```
            "Action": [
                "s3:*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "logs:FilterLogEvents"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "lex:*"
            ],
            "Resource": "*"
        }
    ]
}
```

7. Masukkan nama kebijakan, misalnya 'LexTestWorkbenchPolicy' dan kemudian klik pada Create Policy.
8. Kembali ke tab sebelumnya di browser Anda dan Refresh daftar kebijakan dengan mengklik tombol Refresh seperti yang ditunjukkan di bawah ini.



9. Cari di daftar kebijakan dengan memasukkan nama kebijakan yang Anda gunakan pada langkah ke-6 dan pilih kebijakan.
10. Pilih tombol Next.
11. Masukkan nama peran dan kemudian klik tombol Buat Peran.
12. Pilih peran IAM baru Anda saat diminta di konsol Amazon Lex V2 untuk Test Workbench.

Buat peran IAM untuk Meja Kerja Uji - Fitur Lanjutan

Pengaturan izin untuk peran IAM meja kerja Uji

Bagian ini menunjukkan beberapa contoh kebijakan berbasis identitas AWS Identity and Access Management (IAM) untuk menerapkan kontrol akses hak istimewa paling rendah untuk izin Test Workbench.

1. Kebijakan untuk Test Workbench untuk membaca file audio di S3 — Kebijakan ini memungkinkan Test Workbench untuk membaca file audio yang digunakan dalam set pengujian. Kebijakan di bawah ini harus dimodifikasi sesuai *S3Path* untuk memperbarui *S3BucketName* dan mengarahkannya ke lokasi Amazon S3 dari file audio dalam set pengujian.

   JSON

   ```
   {
     "Version":"2012-10-17",
     "Statement": [
       {
         "Sid": "TestWorkbenchS3AudioFilesReadOnly",
         "Effect": "Allow",
         "Action": [
           "s3:GetObject",
           "s3:GetObjectVersion"
         ],
         "Resource": [
           "arn:aws:s3:::S3BucketName/S3Path/*"
         ]
       }
     ]
   }
   ```

2. Kebijakan untuk Test Workbench untuk membaca dan menulis set pengujian dan hasil ke dalam bucket Amazon S3 — Kebijakan ini memungkinkan Test Workbench menyimpan input dan hasil set pengujian. Kebijakan di bawah ini harus dimodifikasi untuk memperbarui *S3BucketName* ke Amazon S3 Bucket tempat data set uji akan disimpan. Test Workbench menyimpan data ini secara eksklusif di bucket Amazon S3 Anda dan bukan di infrastruktur Lex Service. Oleh karena itu Untuk alasan ini, Test Workbench memerlukan akses ke bucket Amazon S3 Anda agar berfungsi dengan baik.

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. Kebijakan untuk Meja Kerja Uji untuk membaca CloudWatch Log — Kebijakan ini memungkinkan Meja Kerja Uji menghasilkan set uji dari Log Teks Percakapan Lex yang disimpan di Log Amazon. CloudWatch Kebijakan di bawah ini harus dimodifikasi untuk memperbarui*Region*,*AwsAccountId*,*LogGroupName*.
4. Kebijakan untuk Test Workbench untuk memanggil Lex Runtime — Kebijakan ini memungkinkan Test Workbench untuk menjalankan set pengujian terhadap bot Lex. Kebijakan di bawah ini harus dimodifikasi untuk memperbarui*Region*,*AwsAccountId*,*BotId*. Karena Test Workbench dapat menguji bot apa pun di lingkungan Lex Anda, Anda dapat mengganti sumber daya dengan “arn:aws:lex: *Region* ::bot-alias/*” *AwsAccountId* untuk memungkinkan akses Test Workbench ke semua bot Amazon Lex V2 di akun.
5. (Opsional) Kebijakan untuk Test Workbench untuk mengenkripsi dan mendekripsi data set pengujian — Jika Test Workbench dikonfigurasi untuk menyimpan input set tes dan menghasilkan bucket Amazon S3 menggunakan kunci KMS yang dikelola pelanggan, Test Workbench akan memerlukan izin enkripsi dan dekripsi ke kunci KMS. Kebijakan di bawah ini harus dimodifikasi untuk memperbarui *RegionAwsAccountId*, dan *KmsKeyId* di *KmsKeyId* mana ID kunci KMS yang dikelola pelanggan.
6. (Opsional) Kebijakan untuk Test Workbench untuk mendekripsi file audio — Jika file Audio disimpan dalam bucket S3 menggunakan kunci KMS yang dikelola pelanggan, Test Workbench akan memerlukan izin dekripsi ke kunci KMS. Kebijakan di bawah ini harus dimodifikasi untuk memperbarui *RegionAwsAccountId*, dan *KmsKeyId* di *KmsKeyId* mana ID kunci KMS yang dikelola pelanggan digunakan untuk mengenkripsi file audio.

Kelola set tes

Anda dapat mengunduh, memperbarui, dan menghapus set pengujian dari jendela set pengujian. Atau Anda dapat menggunakan daftar set pengujian yang tersedia untuk mengedit atau membubuhi keterangan file set pengujian Anda secara manual. Kemudian, unggah lagi untuk mencoba validasi lagi, karena kesalahan atau masalah input lainnya.

Untuk mengunduh file set pengujian dari catatan set uji:

1. Pilih nama set tes dari daftar set uji.
2. Di jendela test set record, pilih tombol Unduh di sisi kanan layar di bagian Test Inputs.
3. jika ada detail kesalahan validasi di bagian atas jendela mengenai set pengujian, pilih tombol Unduh. File akan disimpan ke folder Unduhan Anda. Anda dapat memperbaiki kesalahan validasi dalam set pengujian dari pesan kesalahan dalam file CSV set pengujian. Temukan kesalahan yang diidentifikasi dalam langkah validasi, perbaiki baris atau hapus, dan unggah file untuk mencoba lagi langkah validasi.
4. jika Anda berhasil mengunduh set tes, pesan spanduk hijau akan muncul.

Untuk mengunduh set pengujian dari daftar set pengujian:

1. Dari daftar set tes, pilih tombol radio di sebelah item set tes yang ingin Anda unduh.
2. Dari menu Tindakan di kanan atas, pilih Unduh.
3. Pesan spanduk hijau akan menunjukkan jika Anda berhasil mengunduh set tes. File akan disimpan ke folder Unduhan Anda.

Kolom set uji didukung oleh Test Workbench

Di bawah ini adalah daftar lengkap kolom set pengujian yang didukung oleh Test Workbench dan instruksi tentang cara menggunakannya dengan Amazon Lex V2.

Nama kolom	Jenis set uji	Jenis Nilai	Beberapa Kolom Support	Deskripsi
Nomor Baris	Teks dan Audio	Bilangan	Tidak	Ini adalah kolom pengguna yang diabaikan oleh Amazon Lex V2. Hal ini dimaksudkan untuk membantu penulis set tes untuk mengurutkan dan memfilter

Nama kolom	Jenis set uji	Jenis Nilai	Beberapa Kolom Support	Deskripsi
				baris test-set. “Baris #” dapat digunakan sebagai nama kolom alternatif.
Nomor Percakapan	Teks dan Audio	Bilangan	Tidak	Kolom ini memungkinkan Anda untuk menempatkan baris dalam percakapan bersama. “Percakapan #” dapat digunakan sebagai nama kolom alternatif.
Sumber	Teks dan Audio	Enum (“Pengguna”, “Agen”)	Tidak	Nilai dalam kolom ini menunjukkan apakah baris adalah untuk pengguna atau agen. “Peserta Percakapan” dapat digunakan sebagai nama kolom alternatif.

Nama kolom	Jenis set uji	Jenis Nilai	Beberapa Kolom Support	Deskripsi
Input	Teks	String	Tidak	Kolom ini digunakan untuk menambahkan transkrip untuk set tes teks. Input teks harus digunakan dalam baris Pengguna. Prompt Agen harus digunakan dalam baris Agen.
Transkripsi yang Diharapkan	Audio	String	Tidak	Kolom ini digunakan untuk menambahkan transkrip untuk set uji audio. Transkripsi yang diharapkan dari file audio harus digunakan di baris Pengguna dengan input audio. Input DTMF dapat digunakan dalam baris Pengguna dengan input DTMF. Prompt Agen harus digunakan dalam baris Agen.

Nama kolom	Jenis set uji	Jenis Nilai	Beberapa Kolom Support	Deskripsi
Lokasi Audio S3	Audio	String	Tidak	Kolom ini digunakan untuk menambahkan lokasi file audio dan hanya berlaku untuk set uji audio. Jalur S3 harus digunakan di baris Pengguna dengan input audio. Bidang ini harus dibiarkan kosong di baris Pengguna dengan input DTMF. Bidang ini harus dibiarkan kosong di baris Agen.

Nama kolom	Jenis set uji	Jenis Nilai	Beberapa Kolom Support	Deskripsi
Tag Konteks Masukan	Teks dan Audio	Tali	Ya	Kolom ini digunakan untuk memberikan nama konteks masukan yang akan digunakan dalam masukan ke Lex saat mengeksekusi baris dalam set tes. Ini mengacu pada konteks masukan dalam Mengatur konteks maksud untuk bot Lex V2 Anda. Perhatikan bahwa Test Workbench hanya mendukung nama konteks. Itu tidak mendukung parameter dalam konteks. Beberapa kolom bernama seperti 'Input Context Tag 1', 'Input Context Tag 2', dan seterusnya, dapat digunakan.

Nama kolom	Jenis set uji	Jenis Nilai	Beberapa Kolom Support	Deskripsi
Permintaan Atribut	Teks dan Audio	Tali	Ya	Kolom ini digunakan untuk menyediakan atribut permintaan yang akan digunakan dalam input untuk Lex saat mengeksekusi baris dalam set tes. Nilai dalam kolom harus disediakan dalam format `<request-attribute-name>=<request-attribute-value>`. Spasi dapat ditambahkan di sekitar '=' untuk keterbacaan. Misalnya: request-attribute-foo = this is a dummy response request-attribute-foo = 'this is a “dummy response" ' request-attribute-foo = “this is a 'dummy response' ”. Beberapa

Nama kolom	Jenis set uji	Jenis Nilai	Beberapa Kolom Support	Deskripsi
				kolom bernama seperti 'Request Attribute 1', 'Request Attribute 2', dan seterusnya, dapat digunakan.

Nama kolom	Jenis set uji	Jenis Nilai	Beberapa Kolom Support	Deskripsi
Atribut Sesi	Teks dan Audio	Tali	Ya	Kolom ini digunakan untuk menyediakan atribut sesi yang akan digunakan dalam masukan untuk Lex saat mengeksekusi baris dalam set tes. • Nilai dalam kolom harus disediakan dalam format `<session-attribute-name>=<session-attribute-value>`. Spasi dapat ditambahkan di sekitar '=' untuk keterbacaan. • Contoh: • session-attribute-foo = ini adalah respons tiruan

Nama kolom	Jenis set uji	Jenis Nilai	Beberapa Kolom Support	Deskripsi
				• session-attribute-foo = 'ini adalah “respons dummy"' • session-attribute-foo = “ini adalah 'respons dummy'” • Beberapa kolom bernama seperti: 'Atribut Sesi 1', 'Atribut Sesi 2', dan seterusnya, dapat digunakan.

Nama kolom	Jenis set uji	Jenis Nilai	Beberapa Kolom Support	Deskripsi
RunTime Petunjuk	Teks dan Audio	Tali	Ya	Kolom ini digunakan untuk memberikan Petunjuk Runtime untuk slot dalam maksud yang akan digunakan dalam input ke Lex saat mengeksekusi baris dalam set pengujian. Di bawah ini adalah contoh: • Nilai dalam kolom harus disediakan dalam format `<intent-name>. <slot - name>< slot-value>=`. Spasi dapat ditambahkan di sekitar '=' untuk keterbacaan. • Contoh: • IntentNameFoo.slot

Nama kolom	Jenis set uji	Jenis Nilai	Beberapa Kolom Support	Deskripsi
				NameFoo = nilai dummy • IntentNameFoo.slotNameFoo = 'a “nilai dummy"' • IntentNameFoo.slotNameFoo = “sebuah 'nilai dummy'” • Meja kerja uji tidak mendukung slot komposit untuk petunjuk runtime. • Beberapa kolom bernama seperti 'RunTime Petunjuk 1',' RunTime Petunjuk 2', dan seterusnya, dapat digunakan.

Nama kolom	Jenis set uji	Jenis Nilai	Beberapa Kolom Support	Deskripsi
Tongkang Masuk	Audio	Boolean	Tidak	Kolom ini digunakan untuk menentukan apakah Meja Kerja Uji harus masuk saat mengirim file audio ke Lex Runtime untuk baris dalam set pengujian. • Hanya berlaku untuk set pengujian audio untuk API streaming. • Kolom ini diabaikan saat menjalankan set pengujian dalam mode API non-strea ming.
Maksud Output yang Diharapkan	Teks dan Audio	String	Tidak	Kolom ini digunakan menentukan nama maksud yang diharapka n dalam output dari Lex untuk baris dalam set pengujian.

Nama kolom	Jenis set uji	Jenis Nilai	Beberapa Kolom Support	Deskripsi

Slot Output yang Diharapkan	Teks dan Audio	Tali	Ya	Kolom ini digunakan untuk memberikan nilai slot yang diharapkan dalam output dari Lex saat mengeksekusi baris dalam set tes.
				• < slot-value>Nilai dalam kolom harus disediakan dalam format `<slot-name>=`. Spasi dapat ditambahkan di sekitar '=' untuk keterbacaan.
				• Contoh untuk slot yang bukan slot multi-nilai atau slot komposit:
				  • slot NameFoo = nilai dummy
				  • slot NameFoo

Nama kolom	Jenis set uji	Jenis Nilai	Beberapa Kolom Support	Deskripsi
				= 'a “nilai dummy"' • slot NameFoo = “sebuah 'nilai dummy'” • Contoh untuk slot multi-nilai: • slot NameFoo = nilai1, nilai2 • slot NameFoo = nilai1, “Item Foo” • slot NameFoo = nilai1, 'nilai2' • Contoh untuk slot komposit di mana nama slot adalah 'Mobil' dan nama sub slot adalah 'Make': • Car.Make = Toyota • Car.Make = “Toyota” • Car.Make = 'Toyota'

Nama kolom	Jenis set uji	Jenis Nilai	Beberapa Kolom Support	Deskripsi
				• Beberapa kolom bernama seperti: 'Slot Output yang Diharapkan 1', 'Slot Output yang Diharapkan 2', dll., Dapat digunakan.

Nama kolom	Jenis set uji	Jenis Nilai	Beberapa Kolom Support	Deskripsi
Tag Konteks Output yang Diharapkan	Teks dan Audio	Tali	Ya	Kolom ini digunakan untuk menentukan nama konteks keluaran yang diharapkan dalam output dari Lex untuk baris dalam set pengujian. • Ini mengacu pada konteks keluaran dalam Mengatur konteks maksud untuk bot Lex V2 Anda. • Perhatikan bahwa Test Workbench hanya mendukung nama konteks dan belum mendukung parameter di dalam konteks. • Beberapa kolom

Nama kolom	Jenis set uji	Jenis Nilai	Beberapa Kolom Support	Deskripsi
				bernama seperti: 'Tag Konteks Keluaran yang Diharapkan 1', 'Tag Konteks Keluaran yang Diharapkan 2', dll., Dapat digunakan.

Lihat kesalahan validasi pengujian di meja kerja pengujian

Anda dapat memperbaiki set pengujian yang melaporkan kesalahan validasi. Kesalahan validasi ini dihasilkan ketika set pengujian tidak siap untuk diuji. Meja Kerja Uji dapat menunjukkan kolom yang diperlukan dalam file CSV input set pengujian yang tidak memiliki nilai dalam format yang diharapkan.

Untuk melihat kesalahan validasi pengujian:

1. Dari daftar set pengujian, pilih nama set pengujian yang melaporkan Status Kesalahan Validasi yang ingin Anda lihat. Nama-nama set tes adalah tautan aktif yang membawa Anda ke detail mengenai set tes.
2. Rekaman set uji menampilkan detail kesalahan validasi di bagian atas layar. Pilih Lihat Detail untuk melihat laporan tentang Kesalahan Validasi.
3. Dari jendela laporan kesalahan, tinjau Baris # dan Jenis Kesalahan untuk melihat di mana kesalahan terjadi. Untuk daftar kesalahan yang panjang, Anda dapat memilih untuk Mengunduh laporan kesalahan.
4. Bandingkan kesalahan yang tercantum dalam file CSV input set pengujian Anda dengan file pengujian asli Anda untuk memperbaiki masalah apa pun dan mengunggah set pengujian lagi.

Tabel berikut mencantumkan pesan kesalahan validasi CSV masukan dengan skenario.

Skenario	Pesan kesalahan	Catatan
Set Uji Ukuran File Melebihi	Ukuran file Test Set lebih besar dari 200 MB. Berikan file yang lebih kecil dan coba permintaan Anda lagi.	
Set tes melebihi catatan maks	File input memiliki catatan lebih dari jumlah maksimum yang didukung 200.000.	
Unggah set Uji Kosong	Set tes yang diimpor kosong. Berikan set pengujian yang tidak kosong dan coba permintaan Anda lagi.	
Nama header kolom kosong	Kolom Header Baris: menemukan nama kolom kosong di kolom nomor 5.	
Nama header kolom yang tidak dikenal	Kolom Header Baris: tidak dapat mengenali nama kolom 'dummy' di kolom nomor 2.	
Nama header kolom duplikat	Baris Header Kolom: menemukan beberapa kolom 'tautan audio S3' dan 'tautan audio S3' yang sama atau setara. Hapus atau ganti nama salah satu kolom tersebut.	
Nama kolom multi nilai melebihi batas	Kolom Header Baris: jumlah kolom untuk 'Slot Output Diharapkan' melebihi jumlah maksimum yang didukung: 6. Hapus beberapa kolom untuk 'Expected Output Slot' dan coba lagi.	Jumlah maksimum kolom yang didukung untuk kolom multi nilai adalah 6.

Skenario	Pesan kesalahan	Catatan
Header kolom terkait teks atau Audio tidak ada	Tidak dapat menemukan kolom untuk percakapan teks atau audio. Untuk percakapan teks, gunakan kolom {'Input teks'}. Untuk percakapan audio, gunakan kolom {'S3 audio link', 'Transkripsi yang diharapkan'}.	Kolom Wajib Audio: {'Tautan audio S3', 'Transkripsi yang diharapkan'} Kolom Wajib Teks: {'Input teks'}
Header kolom terkait Teks dan Audio ada	Kolom yang ditemukan untuk percakapan teks dan audio. Anda dapat menggunakan kolom {'Input teks'} untuk percakapan teks, atau kolom {'S3 audio link', 'Transkripsi yang diharapkan'} untuk percakapan audio.	Kolom Wajib Audio: {'Tautan audio S3', 'Transkripsi yang diharapkan'} Kolom Wajib Teks: {'Input teks'}
Kolom wajib tidak ada	Tidak dapat menemukan kolom wajib ["Intent Output yang Diharapkan"].	Kolom Wajib: {"Baris #", “Sumber”, “Maksud Keluaran yang Diharapkan"}
Menemukan data di kolom tanpa header	Data yang ditemukan di kolom nomor 8 untuk nomor baris 6, tetapi kolom yang sesuai tidak memiliki header kolom.	
Data tidak ditemukan untuk kolom wajib	Baris=12: tidak ada nilai yang ditemukan untuk kolom wajib: {"Sumber", “Maksud Keluaran yang Diharapkan"}	

Skenario	Pesan kesalahan	Catatan
Duplikat id percakapan ditemukan	nomor percakapan '19' terlihat untuk percakapan sebelumnya di baris nomor 39.” Pastikan nomor percakapan yang sama belum disediakan untuk dua percakapan, Anda dapat melakukannya dengan memastikan bahwa semua baris untuk nomor percakapan dikelompokkan bersama.	
Id percakapan tidak valid yang diberikan	Menemukan nilai 'test_conversation' yang tidak valid di kolom 'Percakapan #'. Nilai untuk kolom ini harus berupa numerik atau N/A (yaitu Tidak Berlaku) untuk baris pengguna.	
Nilai non numerik disediakan untuk nomor baris	Ditemukan nilai non-numerik 'test_line' di kolom 'Baris #'. Nilainya harus numerik.	
Id percakapan tidak ditemukan di baris agen	Tidak ada nilai yang ditemukan untuk kolom 'Percakapan #'. Itu harus disediakan untuk baris agen.	
Id percakapan non numerik ditemukan di baris agen	Menemukan nilai non-numerik 'test_conversation' di kolom 'Percakapan #'. Nilainya harus numerik untuk baris agen.	

Skenario	Pesan kesalahan	Catatan
Lokasi S3 tidak valid	Nilai 'bucket/folder' tidak valid diberikan. <bucketName><keyName>Format yang valid adalah S3:///.	
Nama bucket S3 tidak valid	Nama bucket s3 'test_bucket' tidak valid telah disediakan. Periksa nama ember.	
Lokasi audio S3 adalah folder	Lokasi audio yang disediakan 'S3://bucket/folder'tidak valid. Ini menunjuk ke folder S3.	
Nama maksud tidak valid	Karakter yang tidak valid hadir di intent 'intent @name '. Periksa nama intent.	Pemeriksaan Regex: ^ ([0-9a-zA-Z] [_-]?) + $
Nama slot tidak valid	Karakter tidak valid hadir di slot 'Slot @Name '. Periksa nama slotnya.	Regex: ^ ([0-9a-zA-Z] [_-]?) +$Seharusnya tidak dimulai atau diakhiri dengan dot (.)
Nilai slot disediakan untuk slot induk	Nilai slot disediakan untuk subslot 'Address.City' serta slot induk 'Alamat'. Nilai harus hanya disediakan untuk subslot.	Slot induk di CST seharusnya tidak memiliki nilai slot
Karakter tidak valid dalam nama konteks	Karakter tidak valid hadir dalam nama konteks 'context @1 '. Periksa nama konteksnya.	Regex: ^ ([A-Za-z] _?) + $

Skenario	Pesan kesalahan	Catatan
Gaya ejaan slot tidak valid	Nilai 'test' tidak valid diberikan . Pastikan semuanya huruf besar. Nilai yang valid adalah ["Default", "SpellByLetter","SpellByWord"]	Nilai yang didukung ["Default", "SpellByLetter","SpellByWord"
Peserta atau sumber harus agen atau Pengguna	Nilai 'bot' tidak valid disediaka n. Nilai yang valid adalah ["Agen", "Pengguna"].	Enum yang Didukung: "Agen", "Pengguna"
Nomor Baris tidak boleh desimal	Nilai '10.1' tidak valid diberikan . Itu harus menjadi angka yang valid tanpa pecahan apa pun.	
Nomor Percakapan tidak boleh desimal	Nilai '10.1' tidak valid diberikan . Itu harus menjadi angka yang valid tanpa pecahan apa pun.	
Nomor baris harus dalam jangkauan	Nilai '92233720368547758 071' tidak valid disediaka n. Ini harus lebih besar dari atau sama dengan 1 dan kurang dari atau sama dengan 9223372036854775807.	
Barge-in kolom hanya menerima nilai boolean	Nilai 'test' tidak valid diberikan. Itu harus berupa nilai boolean yang valid seperti 'true' atau 'false'. Atau 'ya' dan 'tidak' dapat digunakan.	Nilai yang Mungkin: "Benar", "benar", "T", "Ya", "Ya", "Y", "1", "1.0", "Salah", "salah", "F", "Tidak", "tidak", "N", "0", "0.0"
Slot yang diharapkan, Atribut Sesi, Atribut Permintaan harus dipisahkan dengan sama dengan (=)	Nilai 'slotName:slotValue' tidak memiliki '='. <value>Ni lai tersebut harus diberikan sebagai pasangan kunci-nilai dalam format '<key>='.	Misalnya: slotName = slotType

Skenario	Pesan kesalahan	Catatan
Slot yang diharapkan, Atribut Sesi, Atribut Permintaan harus memiliki pasangan nilai kunci	'=SlotValue' tidak memiliki kunci sebelum '='. <value>Nilai tersebut harus diberikan sebagai pasangan kunci-nilai dalam format '<key>='.	Misalnya: slotName = slotType
Kutipan tidak valid di akhir	Menemukan kutipan yang salah dalam 'item Foo'". Dimulai dengan karakter kutipan `"`tetapi tidak berakhir dengan karakter kutipan yang sama.	Misalnya: `“Item Foo”, KFC`
Kutipan tidak valid di tengah	Menemukan kutipan yang salah di `"Foo's” Burger, dll.`. Ini berisi karakter kutipan `"`di dalam kontennya. Nilai yang mengandung tanda kutip tunggal harus dibungkus dalam tanda kutip ganda dan sebaliknya.	Benar Misalnya: `“Item Foo”, KFC`
Kutipan yang diperlukan	`key = item Foo` berisi tanda kutip tunggal atau tanda kutip ganda tetapi belum dibungkus dalam tanda kutip. Nilai yang mengandung tanda kutip tunggal harus dibungkus dalam tanda kutip ganda dan sebaliknya.	
Kunci Duplikat diulang di kolom	Kunci `key1` diulang dalam dua kolom: `Atribut Sesi 3` dan `Atribut Sesi 1`.	

Skenario	Pesan kesalahan	Catatan
Format tidak valid dalam petunjuk Runtime	Kunci `tidak valid. BookFligh t.Car “`disediakan untuk Petunjuk Runtime. <intentNa me>Untuk Petunjuk Runtime, kunci harus dalam format. <slotName>.	Jika '.' harus ada di tengah kunci, nama maksud dan nama slot tidak dapat diekstraksi dari kunci tersebut. contoh pemformatan yang salah seperti: "“BookFlight,”. BookFlight.Car“,"BookFlight .Car.”
Nama Intent tidak valid di kunci petunjuk runtime	Menemukan intent `intent @name `yang tidak valid untuk Petunjuk Runtime. Periksa nama maksud.	Pemeriksaan Regex: ^ ([0-9a-zA-Z] [_-]?) + $
Nama Slot tidak valid di kunci petunjuk runtime	Menemukan nama slot yang tidak valid di `Slot @Name `untuk Petunjuk Runtime. Periksa nama slot.	Regex: ^ ([0-9a-zA-Z] [_-]?) +$Seharusnya tidak dimulai atau diakhiri dengan dot (.)

Hapus set pengujian di Test Workbench

Anda dapat dengan mudah menghapus set pengujian dari daftar set pengujian Anda.

Untuk menghapus set pengujian:

1. Pergi ke daftar Set Tes dari menu sisi kiri untuk melihat daftar set tes.
2. Dari daftar set pengujian, pilih set pengujian yang ingin Anda hapus.
3. Buka menu tarik-turun Tindakan di kanan atas, dan pilih Hapus.
4. Sebuah pesan mengonfirmasi bahwa set pengujian dihapus.

Edit detail set tes

Anda dapat mengedit nama Test Set dan rincian dalam daftar set tes. Nama atau detail dapat ditambahkan atau diperbarui nanti. Namun, Anda harus memperbarui set pengujian Anda sebelum menjalankan pengujian dengan bot atau data transkripsi Anda.

Untuk mengedit detail set pengujian:

1. Pergi ke daftar set tes dari menu sisi kiri untuk melihat daftar set tes.
2. Dari daftar set pengujian, pilih kotak centang untuk set tes yang ingin Anda edit.
3. Buka menu tarik-turun Tindakan di kanan atas, dan pilih Edit Detail.
4. Sebuah pesan mengonfirmasi bahwa set pengujian berhasil diedit.

Perbarui set tes

Anda dapat memperbarui, memperbaiki, memodifikasi, atau menghapus item dari set pengujian untuk mengoptimalkan hasil dasar Anda, atau untuk memperbaiki kesalahan lain yang mungkin terjadi di set pengujian

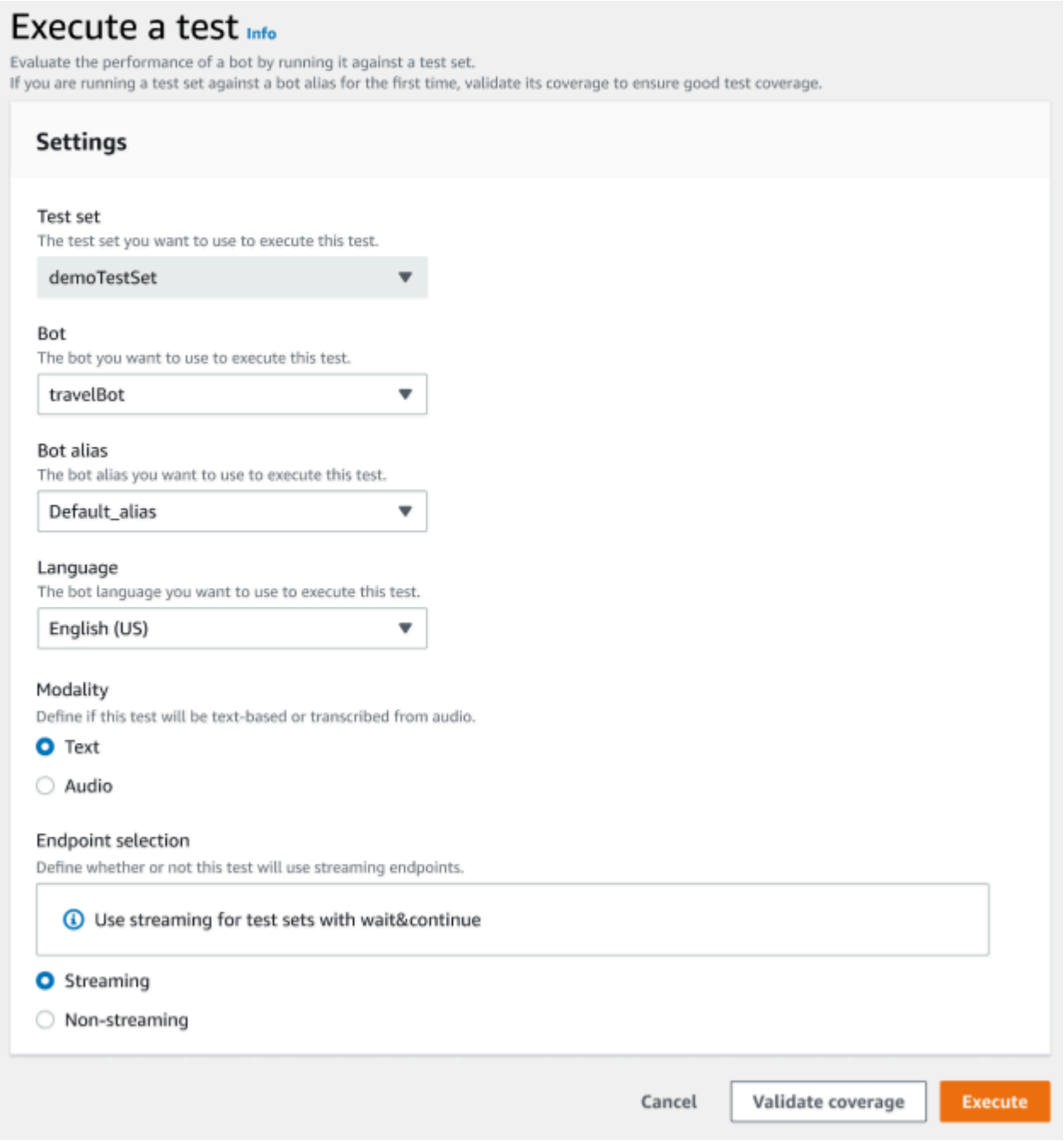
Anda dapat mengunduh set pengujian dan memperbaiki kesalahan validasi sebelum mengunggah set pengujian yang diperbaiki. Lihat Lihat kesalahan validasi pengujian.

Untuk memperbarui set pengujian:

1. Dari catatan set tes, pilih tombol Update Test Set di kanan atas.
2. Pilih file untuk diunggah dari akun Amazon S3 Anda atau unggah file uji CSV dari komputer Anda. CATATAN: Memperbarui set pengujian akan menimpa data yang ada.
3. Pilih tombol Perbarui.
4. Sebuah pesan mengonfirmasi bahwa set pengujian berhasil diperbarui. CATATAN: Operasi ini dapat memakan waktu beberapa menit, tergantung pada kompleksitas dan ukuran set tes.
5. Sebuah pesan mengonfirmasi bahwa set pengujian berhasil diperbarui dan Status menampilkan Siap untuk Pengujian.

Jalankan tes

Untuk menjalankan set pengujian, Anda harus memilih bot yang sesuai untuk menjalankan tes terhadap set pengujian. Anda dapat memilih bot dari akun AWS Anda dari menu tarik-turun di bawah Test Set. Operasi ini akan menguji bot yang Anda pilih terhadap data pengujian yang divalidasi untuk melaporkan metrik kinerja terhadap data dasar dari set pengujian.



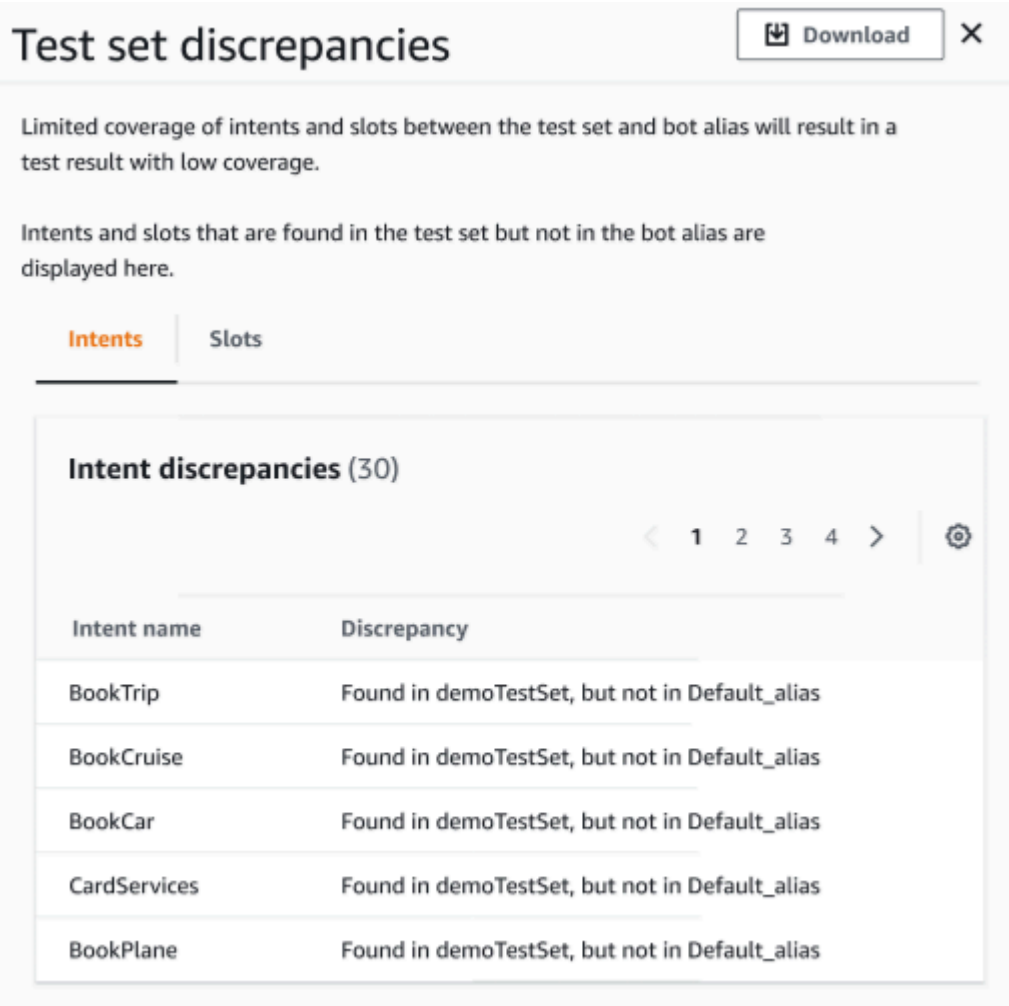
Untuk menjalankan tes di Test Workbench

1. Di halaman catatan set tes, pilih Execute Test.
2. Pilih set tes yang ingin Anda gunakan dalam tes.
3. Pilih nama bot yang akan digunakan dalam pengujian dari menu tarik-turun Bot.
4. Pilih alias bot, jika ada, dari menu drop-down alias Bot.
5. Dari pilihan Bahasa, pilih versi bahasa Inggris.

6. Pilih Teks atau Audio untuk jenis Modalitas.
7. Pilih lokasi Amazon S3 Anda. (hanya audio)
8. Pilih pilihan Endpoint Anda untuk bot Anda. (hanya streaming)
9. Pilih tombol Validasi cakupan untuk mengonfirmasi pengujian Anda siap dijalankan. Jika ada kesalahan yang ada dalam langkah validasi, tinjau parameter sebelumnya dan lakukan koreksi.
10. Pilih Jalankan untuk menjalankan tes.
11. Sebuah pesan menegaskan bahwa tes berhasil dijalankan.

Cakupan set uji di Meja Kerja Uji

Cakupan maksud dan slot yang terbatas antara set pengujian dan bot dapat menghasilkan ukuran kinerja yang diharapkan. Kami menyarankan Anda meninjau cakupan set pengujian sebelum menjalankan tes.



Untuk meninjau cakupan validasi

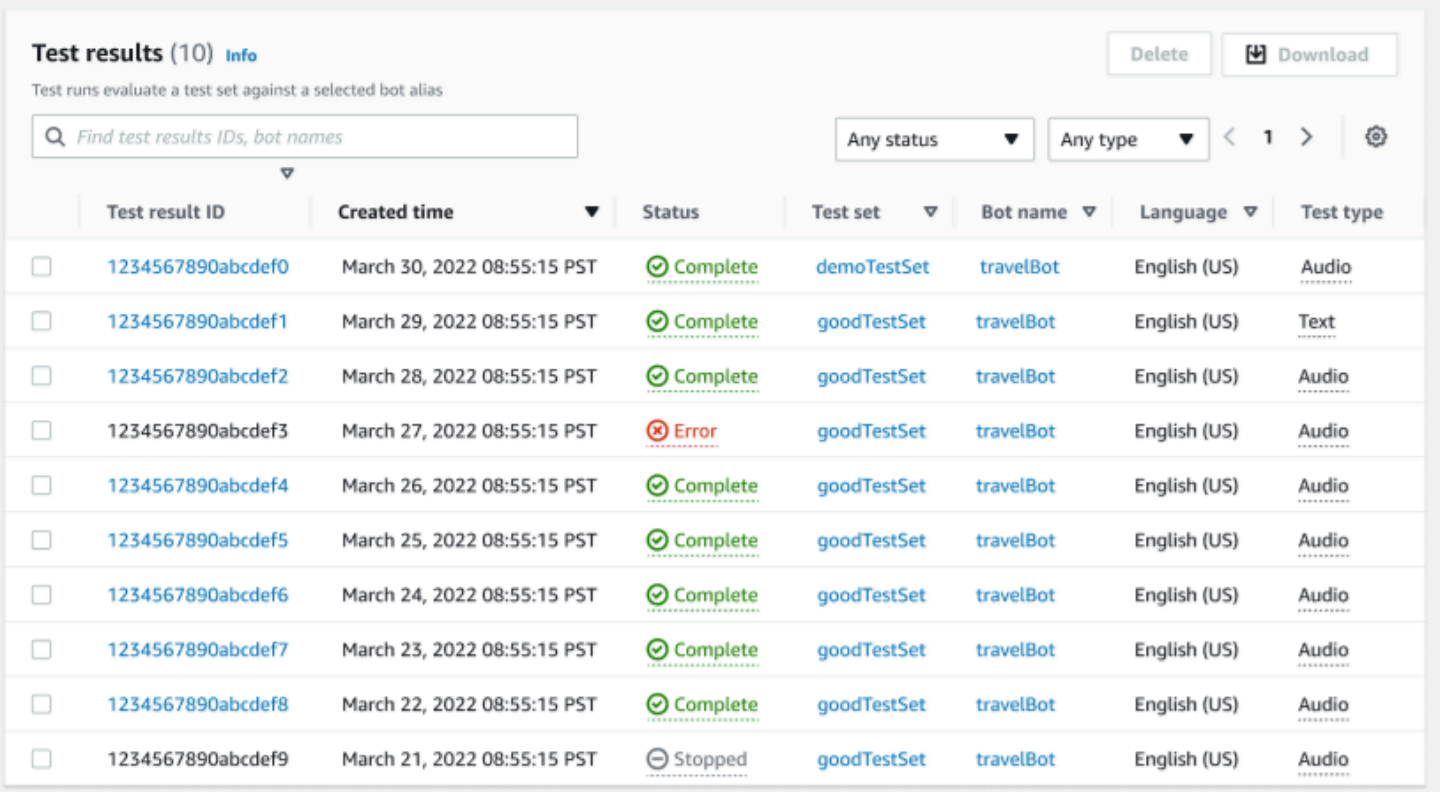
1. Dalam catatan set pengujian, pilih tombol Validasi cakupan.
2. Pesan menunjukkan itu memvalidasi cakupan antara set pengujian dan bot yang dipilih.
3. Setelah operasi selesai, pesan menunjukkan validasi Cakupan berhasil.
4. Pilih tombol Lihat Detail di bagian bawah jendela.
5. Lihat perbedaan set tes untuk maksud dan slot dengan memilih tab untuk masing-masing. Anda dapat mengunduh data ini ke dalam format CSV dengan memilih tombol Unduh.
6. Tinjau hasil validasi untuk data set pengujian, maksud bot, dan slot Anda. Identifikasi masalah dan buat perubahan dalam arsitektur set pengujian bot Anda untuk meningkatkan hasil. Unggah set pengujian yang diedit dan bot untuk menjalankan pengujian setelah Anda membuat perubahan pada file CSV. CATATAN: Cakupan validasi berjalan terhadap set pengujian dan bukan terhadap bot. Maksud dalam bot tetapi tidak ada dalam set pengujian tidak akan tercakup.

Lihat hasil tes

Menafsirkan hasil pengujian dari Meja Kerja Uji untuk menentukan di mana percakapan antara bot Anda dan pelanggan mungkin gagal, atau mengharuskan pelanggan untuk melakukan beberapa upaya untuk memenuhi maksud tersebut.

Dengan menemukan masalah ini di hasil pengujian Anda, Anda dapat mengoptimalkan kinerja bot Anda dengan meningkatkan kinerja maksud menggunakan data pelatihan berbeda atau ucapan yang lebih konsisten dengan nilai transkripsi bot waktu nyata.

Anda bisa mendapatkan tampilan rinci tentang maksud dan slot yang memiliki perbedaan kinerja. Setelah Anda mengidentifikasi maksud atau slot yang memiliki perbedaan, Anda dapat menelusuri lebih lanjut dan meninjau ucapan dan alur percakapan.



Untuk meninjau hasil tes:

1. Pergi ke daftar set tes dari menu sisi kiri untuk memilih opsi Hasil tes di bawah meja kerja Uji. CATATAN: Hasil tes menunjukkan Status lengkap jika berhasil.
2. Pilih ID Hasil Tes untuk hasil tes yang ingin Anda tinjau.

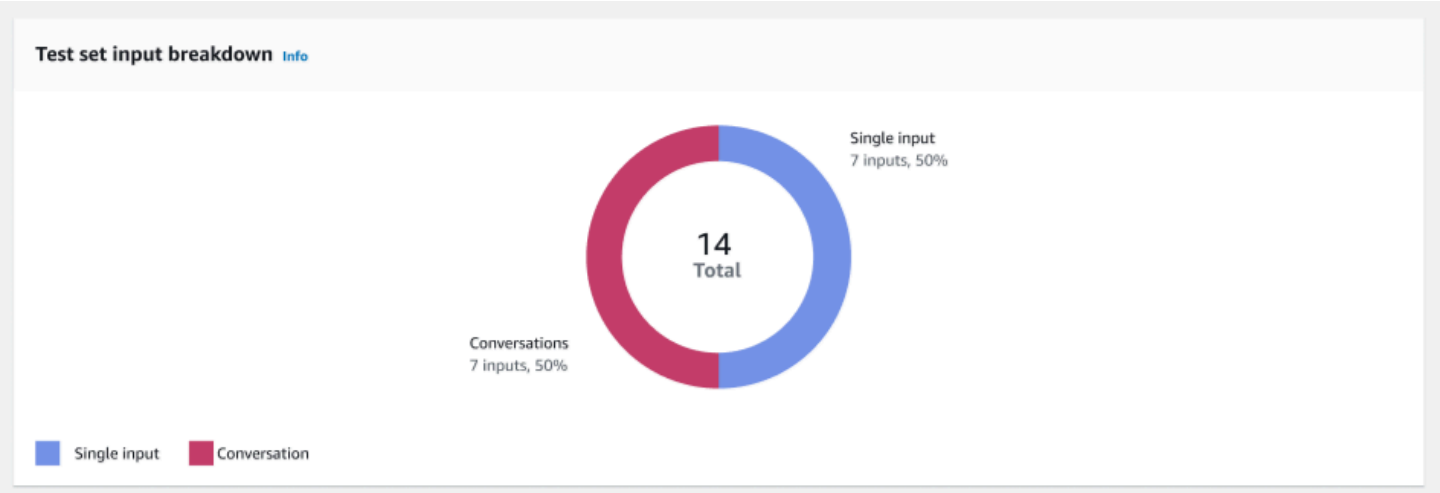
Rincian hasil tes di Test Workbench

Hasil tes menunjukkan rincian set tes, maksud yang digunakan, dan slot yang digunakan. Ini juga menyediakan rincian input set tes keseluruhan mencakup hasil keseluruhan, hasil percakapan, maksud, dan hasil slot.

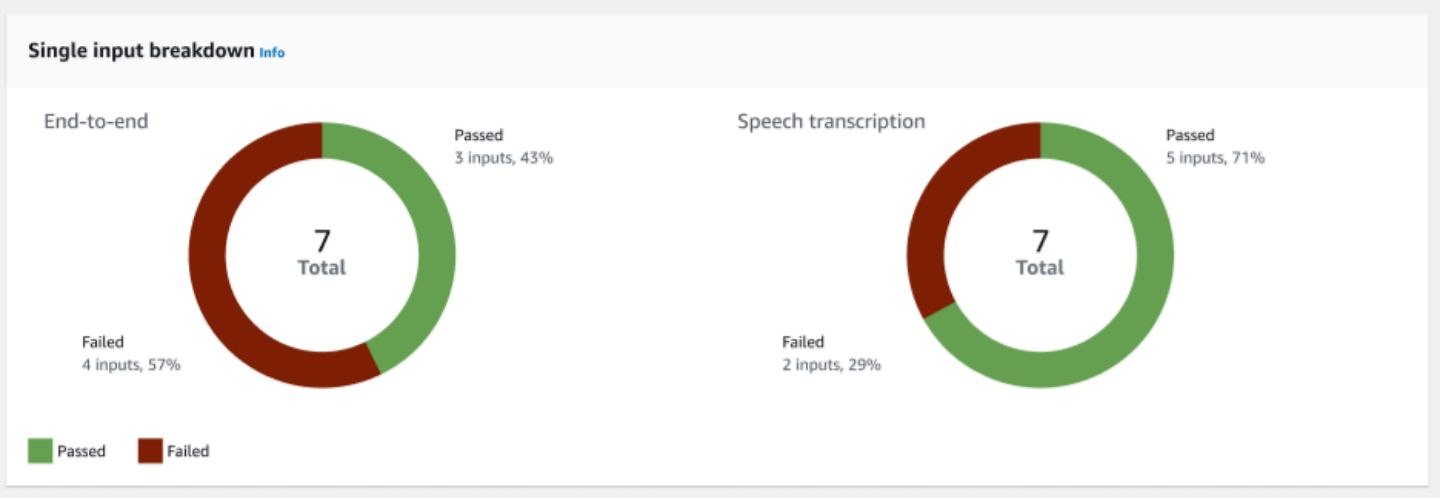
Hasil tes terdiri dari semua informasi terkait pengujian seperti:

- Metadata detail uji
- Hasil keseluruhan
- Hasil percakapan
- Hasil maksud dan slot
- Hasil terperinci

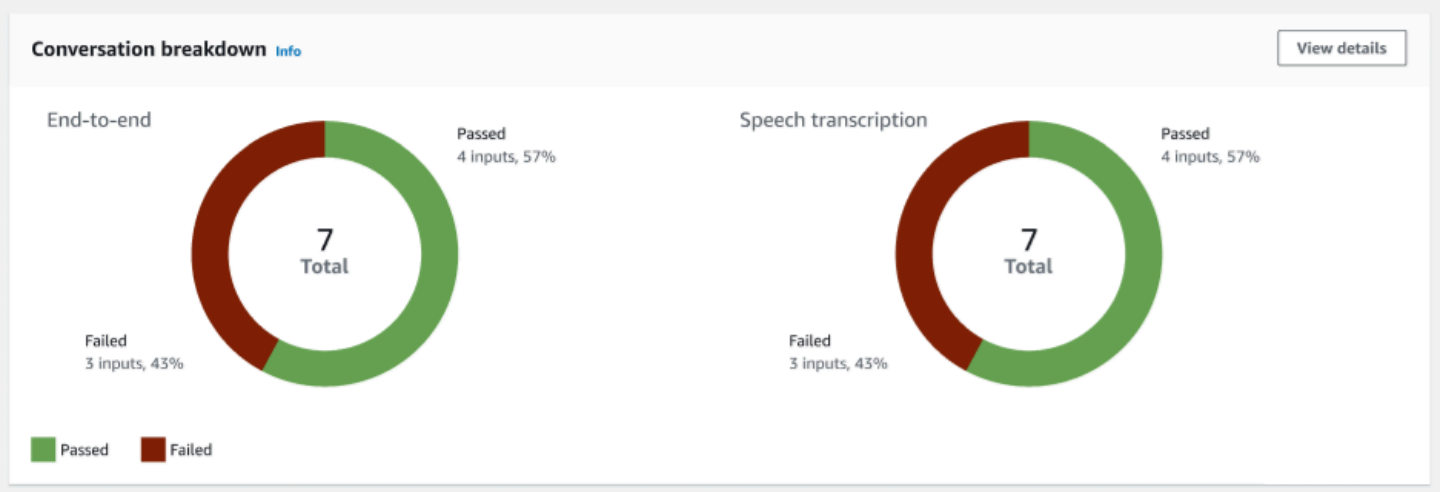
Tab hasil keseluruhan:



Rincian input set tes - Bagan ini menunjukkan rincian jumlah percakapan dan ucapan input tunggal dalam set pengujian.

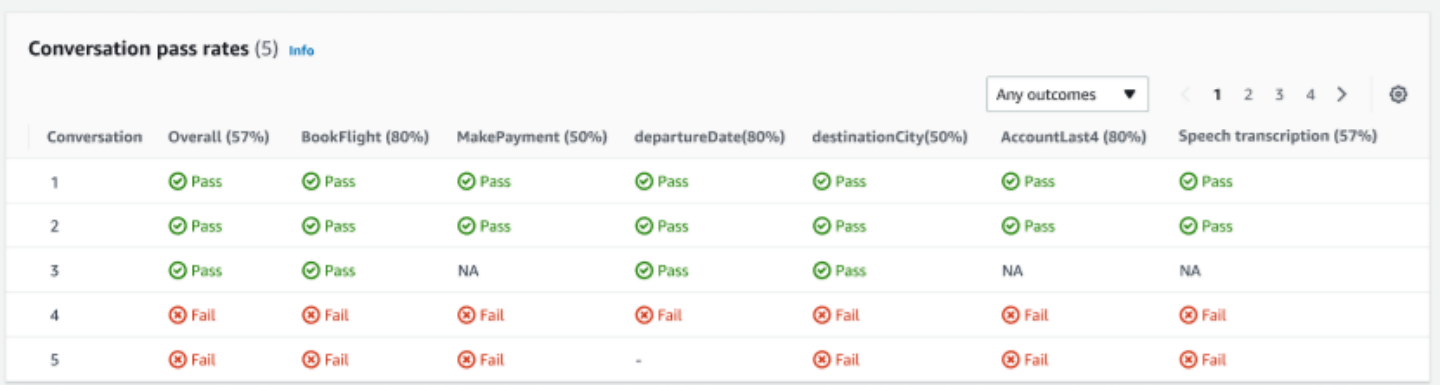


Perincian input tunggal — Menampilkan dua bagan yang mencakup percakapan ujung ke ujung dan transkripsi ucapan. Jumlah input yang lulus dan gagal ditunjukkan pada setiap bagan. Catatan: Bagan transkripsi ucapan hanya akan terlihat untuk set tes audio.



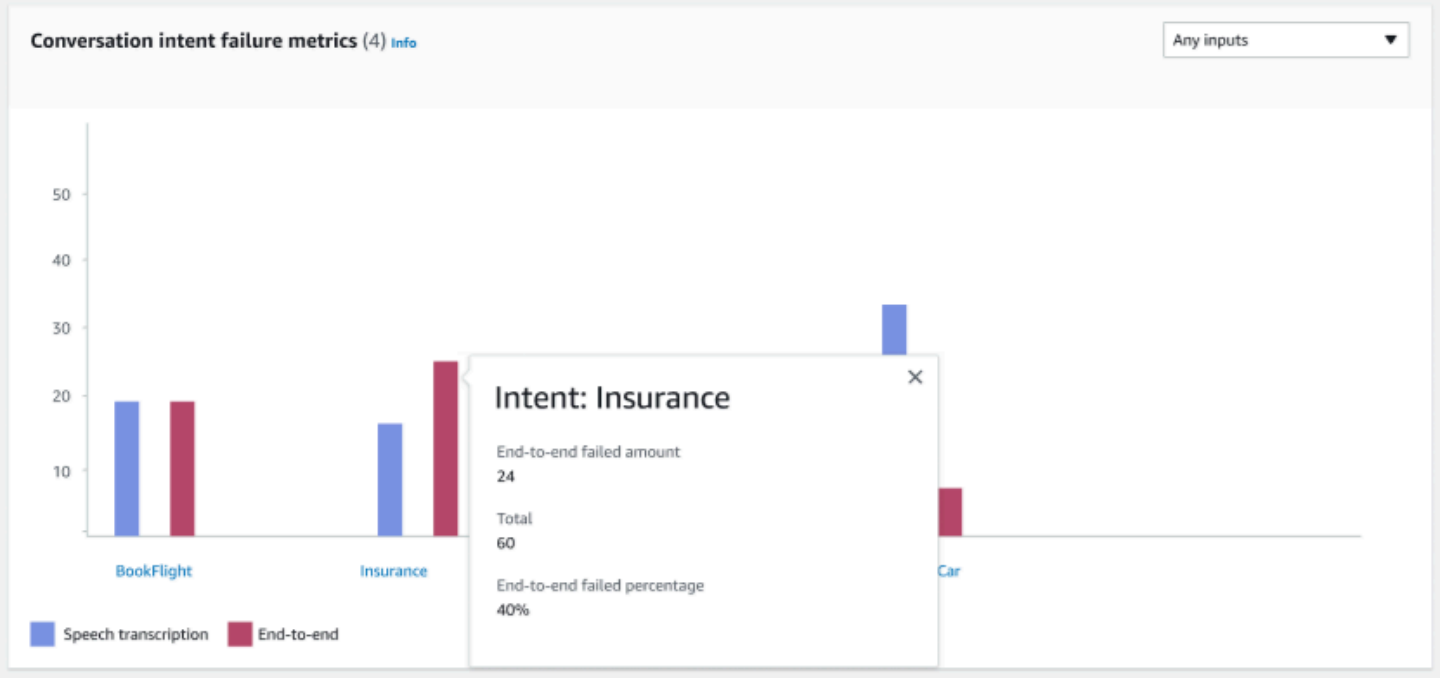
Perincian percakapan - Menampilkan dua bagan yang mencakup percakapan ujung ke ujung dan transkripsi ucapan. Jumlah input yang lulus dan gagal ditunjukkan pada setiap bagan. Catatan: Bagan transkripsi ucapan hanya akan terlihat untuk set tes audio.

Tab hasil percakapan:



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

Tingkat kelulusan percakapan — Tabel tarif kelulusan percakapan digunakan untuk melihat maksud dan slot mana yang digunakan dalam setiap percakapan di set tes. Anda dapat memvisualisasikan di mana percakapan gagal dengan meninjau maksud atau slot mana yang gagal, bersama dengan persentase kelulusan dari setiap maksud dan slot.

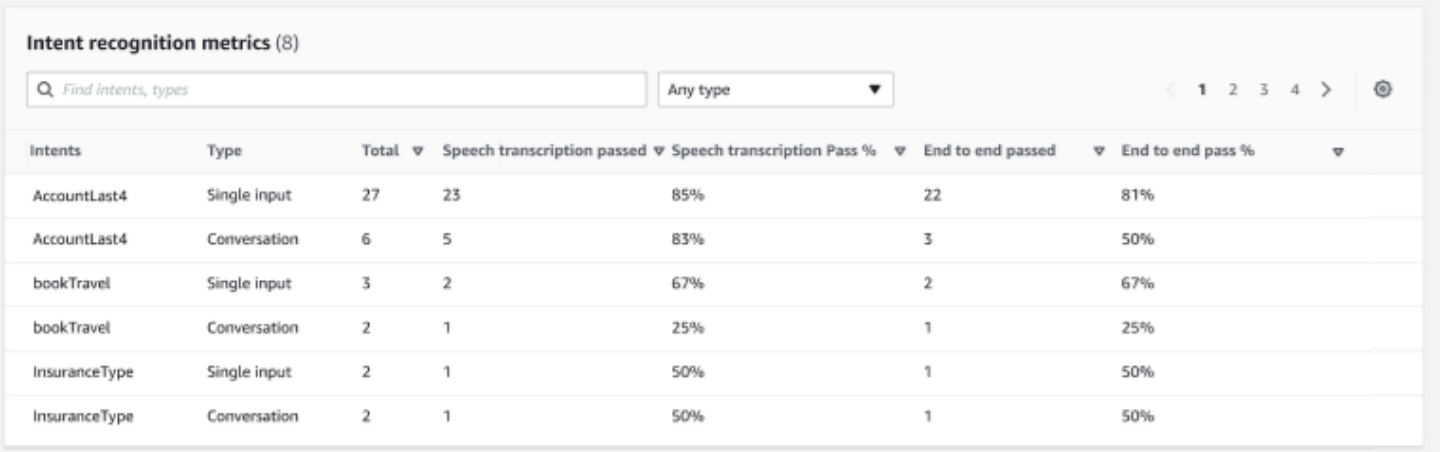


Metrik kegagalan maksud percakapan — Metrik ini menunjukkan 5 intent berkinerja terburuk teratas dalam set pengujian. Panel ini menunjukkan bagan berapa persen atau jumlah maksud yang berhasil atau gagal berdasarkan log percakapan atau transkripsi bot. Niat sukses tidak berarti bahwa seluruh percakapan berhasil. Metrik ini hanya berlaku untuk nilai intent, terlepas dari maksud mana yang datang sebelum atau sesudahnya.



Metrik kegagalan slot percakapan — Metrik ini menunjukkan 5 slot berkinerja terburuk teratas dalam set pengujian. Menunjukkan tingkat keberhasilan untuk setiap slot dalam maksud. Grafik batang menunjukkan transkripsi ucapan dan percakapan ujung ke ujung untuk setiap slot dalam maksud.

Tab hasil intent dan slot:

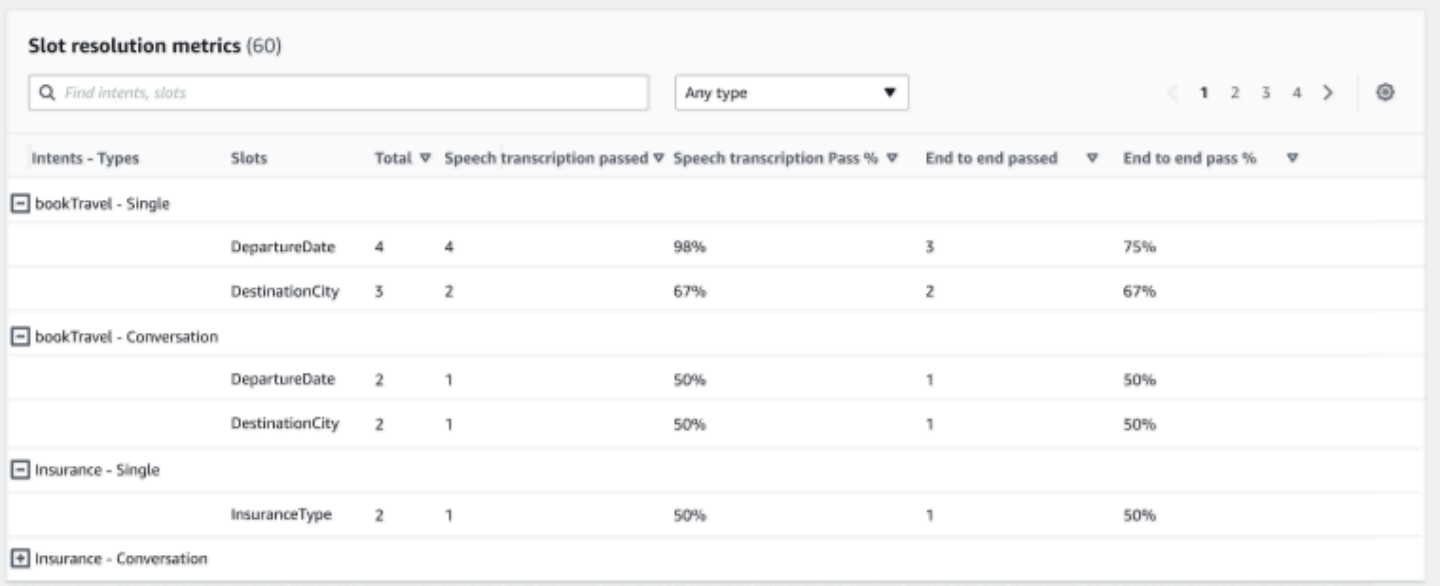


Intent recognition metrics (8)

Find intents, types | Any type | 1 2 3 4

Intents	Type	Total	Speech transcription passed	Speech transcription Pass %	End to end passed	End to end pass %
AccountLast4	Single input	27	23	85%	22	81%
AccountLast4	Conversation	6	5	83%	3	50%
bookTravel	Single input	3	2	67%	2	67%
bookTravel	Conversation	2	1	25%	1	25%
InsuranceType	Single input	2	1	50%	1	50%
InsuranceType	Conversation	2	1	50%	1	50%

Metrik pengenalan maksud — Menampilkan tabel berapa banyak intent yang berhasil dikenali. Menampilkan tingkat kelulusan transkripsi ucapan dan percakapan ujung ke ujung.



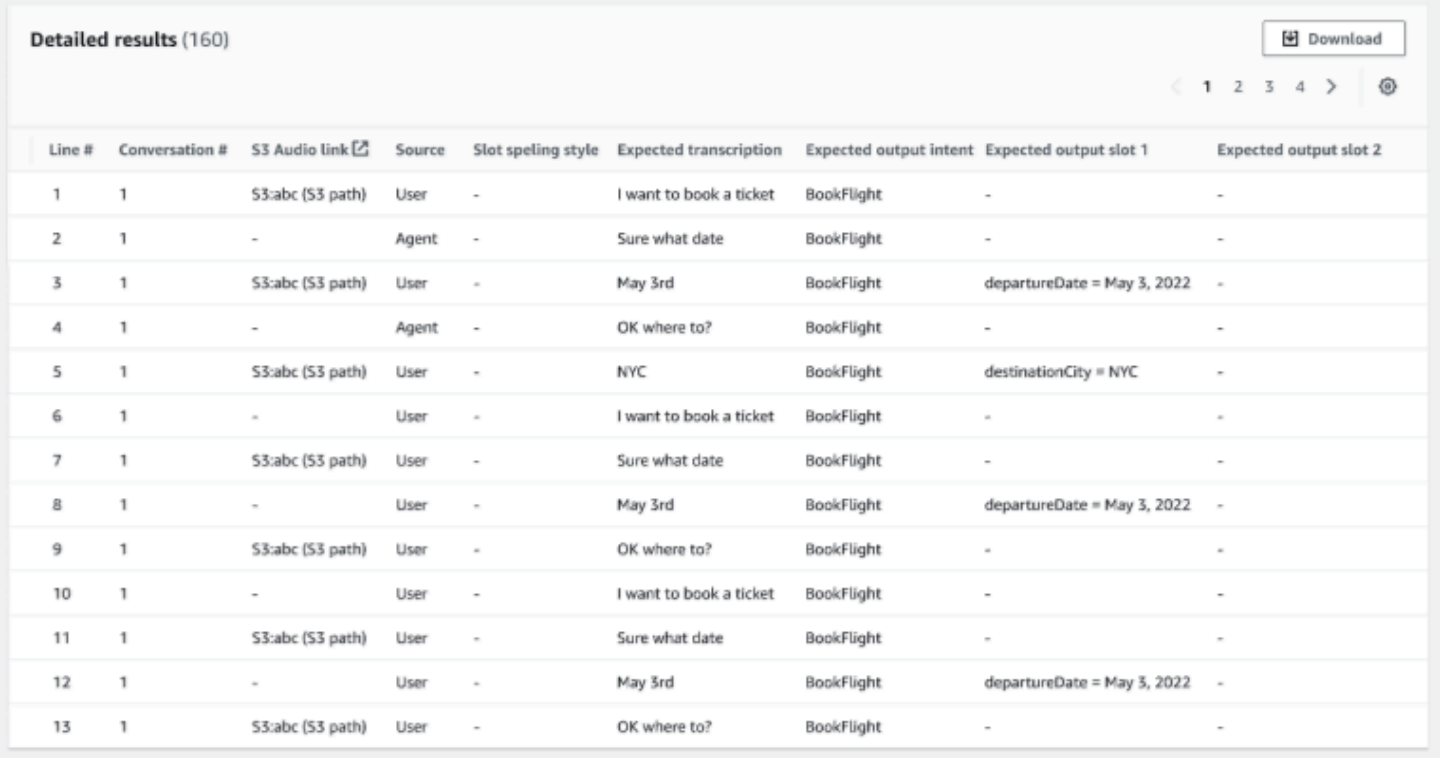
Slot resolution metrics (60)

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						

Metrik resolusi slot — Menunjukkan maksud dan slot secara terpisah, dan tingkat keberhasilan dan kegagalan setiap slot untuk setiap maksud yang digunakan dalam percakapan atau input tunggal. Menampilkan tingkat kelulusan transkripsi ucapan dan percakapan ujung ke ujung.

Tab hasil terperinci:



Detailed results (160)

Download

1 2 3 4

Line #	Conversation #	S3 Audio link	Source	Slot speling style	Expected transcription	Expected output intent	Expected output slot 1	Expected output slot 2
1	1	S3:abc (S3 path)	User	-	I want to book a ticket	BookFlight	-	-
2	1	-	Agent	-	Sure what date	BookFlight	-	-
3	1	S3:abc (S3 path)	User	-	May 3rd	BookFlight	departureDate = May 3, 2022	-
4	1	-	Agent	-	OK where to?	BookFlight	-	-
5	1	S3:abc (S3 path)	User	-	NYC	BookFlight	destinationCity = NYC	-
6	1	-	User	-	I want to book a ticket	BookFlight	-	-
7	1	S3:abc (S3 path)	User	-	Sure what date	BookFlight	-	-
8	1	-	User	-	May 3rd	BookFlight	departureDate = May 3, 2022	-
9	1	S3:abc (S3 path)	User	-	OK where to?	BookFlight	-	-
10	1	-	User	-	I want to book a ticket	BookFlight	-	-
11	1	S3:abc (S3 path)	User	-	Sure what date	BookFlight	-	-
12	1	-	User	-	May 3rd	BookFlight	departureDate = May 3, 2022	-
13	1	S3:abc (S3 path)	User	-	OK where to?	BookFlight	-	-

Hasil terperinci — Menampilkan tabel terperinci pada log percakapan dengan ucapan Pengguna dan Agen serta output yang diharapkan dan transkripsi yang diharapkan untuk setiap slot. Anda dapat mengunduh laporan ini dengan memilih tombol Unduh.

Tabel berikut mencantumkan pesan kesalahan kegagalan hasil dengan skenario.

Skenario	Pesan kesalahan	Tindakan
Ketidakcocokan Niat	BookFlight Niat yang diharapkan tetapi itu BookHotel niat.	Lewati belokan lain dalam percakapan
Ketidakcocokan Elisitasi Slot	Slot DepartureDate yang diharapkan akan diperoleh tetapi itu CabinType.	Lewati belokan lain dalam percakapan
Ketidakcocokan nilai slot	Ketidakcocokan antara nilai slot yang diharapkan dan aktual.	Lanjutkan dengan belokan lain dalam percakapan

Skenario	Pesan kesalahan	Tindakan
Back-to-back prompt agen tidak ada	Bot yang diharapkan untuk mengembalikan prompt agen pada giliran ini tetapi tidak diterima.	Lewati belokan lain dalam percakapan
Ketidakcocokan Transkripsi	Transkripsi yang diharapkan tidak cocok dengan transkripsi yang sebenarnya.	Lanjutkan dengan belokan lain dalam percakapan
Slot opsional tidak ditimbulkan	Diharapkan untuk mendapatk an slot CabinType di giliran berikutnya, namun niat saat ini terpenuhi sebelum itu.	Lewati belokan lain dalam percakapan
Slot tidak dikenali	Slot DepartureDate yang diharapkan tidak dikenali pada giliran ini.	Lewati belokan lain dalam percakapan
Prompt agen back-to-back ekstra	Diharapkan giliran pengguna tetapi itu adalah prompt agen	Lewati belokan lain dalam percakapan

Streaming percakapan ke bot Amazon Lex V2

Anda dapat menggunakan API streaming Amazon Lex V2 untuk memulai aliran dua arah antara bot Amazon Lex V2 dan aplikasi Anda. Memulai aliran memungkinkan bot untuk mengelola percakapan antara bot dan pengguna. Bot merespons input pengguna tanpa Anda menulis kode untuk menangani tanggapan dari pengguna. Bot dapat:

- Menangani interupsi dari pengguna saat sedang memutar prompt. Untuk informasi selengkapnya, lihat Mengaktifkan bot Amazon Lex V2 Anda terganggu oleh pengguna.
- Tunggu pengguna untuk memberikan masukan. Misalnya, bot dapat menunggu pengguna mengumpulkan informasi kartu kredit. Untuk informasi selengkapnya, lihat Mengaktifkan bot Amazon Lex V2 untuk menunggu pengguna memberikan informasi lebih lanjut selama jeda.
- Ambil dual-tone multiple-frequency (DTMF) dan input audio dalam aliran yang sama.
- Menangani jeda dalam input pengguna lebih baik daripada jika Anda mengelola percakapan dari aplikasi Anda.

Bot Amazon Lex V2 tidak hanya merespons data yang dikirim dari aplikasi Anda, tetapi juga mengirimkan informasi tentang keadaan percakapan ke aplikasi Anda. Anda dapat menggunakan informasi ini untuk mengubah cara aplikasi Anda merespons pelanggan.

Bot Amazon Lex V2 juga memantau koneksi antara bot dan aplikasi Anda. Ini dapat menentukan apakah koneksi telah habis waktunya.

Untuk menggunakan API untuk memulai streaming ke bot Amazon Lex V2, lihatMemulai aliran percakapan ke bot Amazon Lex V2.

Saat Anda mulai streaming ke bot Amazon Lex V2 dari aplikasi Anda, Anda dapat mengonfigurasi bot untuk menerima input audio atau input teks dari pengguna. Anda juga dapat memilih apakah pengguna menerima audio atau teks sebagai tanggapan atas input mereka.

Jika Anda telah mengonfigurasi bot Amazon Lex V2 untuk menerima input audio dari pengguna, bot tersebut tidak dapat mengambil input teks. Jika Anda telah mengonfigurasi bot untuk menerima input teks, pengguna hanya dapat menggunakan teks tertulis untuk berkomunikasi dengannya.

Ketika bot Amazon Lex V2 mengambil input audio streaming, bot menentukan kapan pengguna mulai berbicara dan kapan mereka berhenti berbicara. Ini menangani jeda atau gangguan apa pun dari pengguna. Ini juga dapat mengambil input DTMF (dual-tone multi-frequency) dan input ucapan dalam aliran yang sama. Ini membantu pengguna berinteraksi dengan bot secara lebih alami. Anda dapat

menyajikan pesan dan permintaan selamat datang kepada pengguna. Anda juga dapat mengaktifkan pengguna untuk mengganggu pesan dan permintaan tersebut.

Saat Anda memulai aliran dua arah, Amazon Lex V2 menggunakan protokol. HTTP/2 Aplikasi Anda dan bot bertukar data dalam satu aliran sebagai serangkaian acara. Sebuah acara dapat berupa salah satu dari yang berikut:

- Teks, audio, atau input DTMF dari pengguna.
- Sinyal dari aplikasi ke bot Amazon Lex V2. Ini termasuk indikasi bahwa pemutaran audio pesan telah selesai, atau bahwa pengguna telah terputus dari sesi.

Untuk informasi selengkapnya tentang peristiwa, lihat Memulai aliran percakapan ke bot Amazon Lex V2. Untuk informasi tentang cara menyandikan peristiwa, lihatPengkodean aliran acara.

Topik

- Memulai aliran percakapan ke bot Amazon Lex V2
- Pengkodean aliran acara
- Mengaktifkan bot Amazon Lex V2 Anda terganggu oleh pengguna
- Mengaktifkan bot Amazon Lex V2 untuk menunggu pengguna memberikan informasi lebih lanjut selama jeda
- Mengkonfigurasi pembaruan kemajuan pemenuhan untuk bot Lex V2 Anda
- Mengkonfigurasi batas waktu untuk menangkap input pengguna dengan bot Lex V2

Memulai aliran percakapan ke bot Amazon Lex V2

Anda menggunakan StartConversationoperasi untuk memulai streaming antara pengguna dan bot Amazon Lex V2 di aplikasi Anda. `POST`Permintaan dari aplikasi membuat koneksi antara aplikasi Anda dan bot Amazon Lex V2. Ini memungkinkan aplikasi Anda dan bot untuk mulai bertukar informasi satu sama lain melalui acara.

> **Note**
>
> Saat menggunakan StartConversation, jika durasi audio ucapan melebihi nilai yang dikonfigurasi untuk`max-length-ms`, Amazon Lex V2 memotong audio pada durasi yang ditentukan.

`StartConversation`Operasi hanya didukung dalam SDK berikut:

- AWS SDK for C++
- AWS SDK for Java V2
- AWS SDK untuk v3 JavaScript
- AWS SDK for Ruby V3

Acara pertama yang harus dikirim aplikasi Anda ke bot Amazon Lex V2 adalah ConfigurationEvent. Acara ini mencakup informasi seperti format tipe respons. Berikut ini adalah parameter yang dapat Anda gunakan dalam acara konfigurasi:

- respons ContentType - Menentukan apakah bot merespons input pengguna dengan teks atau ucapan.
- SessionState — Informasi yang berkaitan dengan sesi streaming dengan bot seperti maksud yang telah ditentukan atau status dialog.
- WelcomeMessages - Menentukan pesan selamat datang yang bermain untuk pengguna di awal percakapan mereka dengan bot. Pesan-pesan ini diputar sebelum pengguna memberikan masukan apa pun. Untuk mengaktifkan pesan selamat datang, Anda juga harus menentukan nilai untuk `sessionState` dan `dialogAction` parameter.
- DisablePlayback — Menentukan apakah bot harus menunggu isyarat dari klien sebelum mulai mendengarkan input pemanggil. Secara default, pemutaran diaktifkan, jadi nilai bidang ini adalah`false`.
- RequestAttributes - Menyediakan informasi tambahan untuk permintaan.

Untuk informasi tentang cara menentukan nilai untuk parameter sebelumnya, lihat tipe ConfigurationEventdata operasi. StartConversation

Setiap aliran antara bot dan aplikasi Anda hanya dapat memiliki satu peristiwa konfigurasi. Setelah aplikasi Anda mengirim acara konfigurasi, bot dapat mengambil komunikasi tambahan dari aplikasi Anda.

Jika Anda telah menentukan bahwa pengguna Anda menggunakan audio untuk berkomunikasi dengan bot Amazon Lex V2, aplikasi Anda dapat mengirim peristiwa berikut ke bot selama percakapan itu:

- AudioInputEvent- Berisi potongan audio yang memiliki ukuran maksimum 320 byte. Aplikasi Anda harus menggunakan beberapa peristiwa input audio untuk mengirim pesan dari server ke bot. Setiap peristiwa input audio dalam aliran harus memiliki format audio yang sama.
- DTMFInputEvent— Mengirim input DTMF ke bot. Setiap penekanan tombol DTMF sesuai dengan satu peristiwa.
- PlaybackCompletionEvent— Menginformasikan server bahwa respons dari input pengguna telah diputar kembali kepada mereka. Anda harus menggunakan acara penyelesaian pemutaran jika Anda mengirim respons audio ke pengguna. Jika `disablePlayback` peristiwa konfigurasi Anda`true`, Anda tidak dapat menggunakan fitur ini.
- DisconnectionEvent— Menginformasikan bot bahwa pengguna telah terputus dari percakapan.

Jika Anda telah menentukan bahwa pengguna menggunakan teks untuk berkomunikasi dengan bot, aplikasi Anda dapat mengirim peristiwa berikut ke bot selama percakapan itu:

- TextInputEvent— Teks yang dikirim dari aplikasi Anda ke bot. Anda dapat memiliki hingga 512 karakter dalam acara input teks.
- PlaybackCompletionEvent— Menginformasikan server bahwa respons dari input pengguna telah diputar kembali kepada mereka. Anda harus menggunakan acara ini jika Anda memutar audio kembali ke pengguna. Jika `disablePlayback` peristiwa konfigurasi Anda`true`, Anda tidak dapat menggunakan fitur ini.
- DisconnectionEvent— Menginformasikan bot bahwa pengguna telah terputus dari percakapan.

Anda harus menyandikan setiap acara yang Anda kirim ke bot Amazon Lex V2 dalam format yang benar. Untuk informasi selengkapnya, lihat Pengkodean aliran acara.

Setiap acara memiliki ID acara. Untuk membantu memecahkan masalah apa pun yang mungkin terjadi di aliran, tetapkan ID peristiwa unik untuk setiap peristiwa masukan. Anda kemudian dapat memecahkan masalah kegagalan pemrosesan apa pun dengan bot.

Amazon Lex V2 juga menggunakan stempel waktu untuk setiap acara. Anda dapat menggunakan stempel waktu ini selain ID acara untuk membantu memecahkan masalah transmisi jaringan apa pun.

Selama percakapan antara pengguna dan bot Amazon Lex V2, bot dapat mengirim peristiwa keluar berikut sebagai tanggapan terhadap pengguna:

- IntentResultEvent— Berisi maksud bahwa Amazon Lex V2 ditentukan dari ucapan pengguna. Setiap peristiwa hasil internal meliputi:

  - InputMode — Jenis ucapan pengguna. Nilai yang valid adalah `Speech`, `DTMF`, atau `Text`.
  - interpretasi — Interpretasi yang ditentukan Amazon Lex V2 dari ucapan pengguna.
  - requestAttributes - Jika Anda belum memodifikasi atribut permintaan dengan menggunakan fungsi lambda, ini adalah atribut yang sama yang diteruskan pada awal percakapan.
  - SessionID — Pengenal sesi yang digunakan untuk percakapan.
  - SessionState - Keadaan sesi pengguna dengan Amazon Lex V2.
- TranscriptEvent— Jika pengguna memberikan masukan ke aplikasi Anda, acara ini berisi transkrip ucapan pengguna ke bot. Aplikasi Anda tidak menerima `TranscriptEvent` jika tidak ada input pengguna.

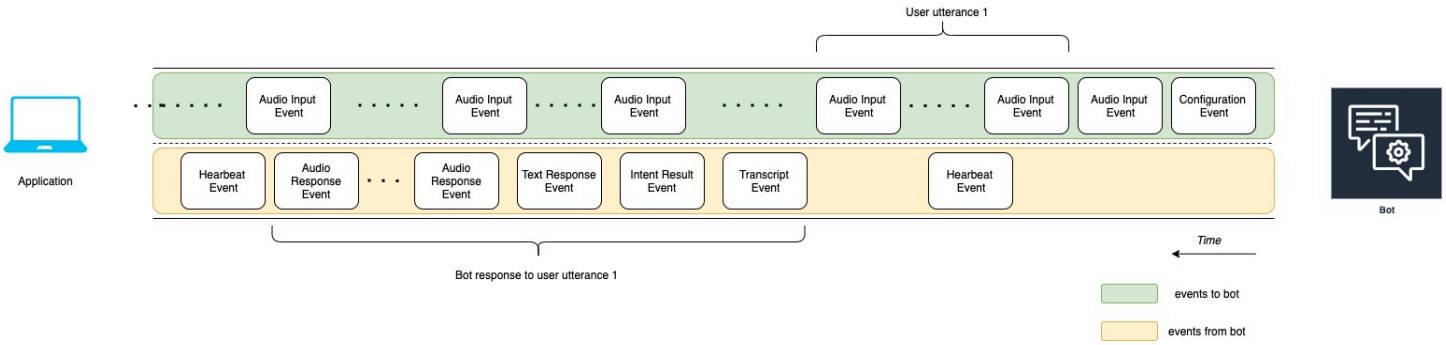
  Nilai peristiwa transkrip yang dikirim ke aplikasi Anda tergantung pada apakah Anda telah menentukan audio (ucapan dan DMTF) atau teks sebagai mode percakapan:

  - Transkrip input ucapan — Jika pengguna berbicara dengan bot, peristiwa transkrip adalah transkripsi audio pengguna. Ini adalah transkrip dari semua pidato dari saat pengguna mulai berbicara hingga saat mereka berhenti berbicara.
  - Transkrip input DTMF - Jika pengguna mengetik pada keypad, peristiwa transkrip berisi semua digit yang ditekan pengguna dalam input mereka.
  - Transkrip input teks - Jika pengguna memberikan input teks, peristiwa transkrip berisi semua teks dalam input pengguna.
- TextResponseEvent— Berisi respons bot dalam format teks. Respons teks dikembalikan secara default. Jika Anda telah mengonfigurasi Amazon Lex V2 untuk mengembalikan respons audio, teks ini digunakan untuk menghasilkan respons audio. Setiap peristiwa respons teks berisi array objek pesan yang bot kembali ke pengguna.
- AudioResponseEvent— Berisi respons audio yang disintesis dari teks yang dihasilkan di. `TextResponseEvent` Untuk menerima peristiwa respons audio, Anda harus mengonfigurasi Amazon Lex V2 untuk memberikan respons audio. Semua acara respons audio memiliki format audio yang sama. Setiap acara berisi potongan audio tidak lebih dari 100 byte. Amazon Lex V2 mengirimkan potongan audio kosong dengan `bytes` bidang yang disetel `null` untuk menunjukkan bahwa akhir peristiwa respons audio ke aplikasi Anda.
- PlaybackInterruptionEvent— Ketika pengguna mengganggu respons yang dikirim bot ke aplikasi Anda, Amazon Lex V2 memicu acara ini untuk menghentikan pemutaran respons.
- HeartbeatEvent— Amazon Lex V2 mengirimkan acara ini kembali secara berkala untuk menjaga koneksi antara aplikasi Anda dan bot dari waktu habis.

Urutan waktu acara untuk percakapan audio saat menggunakan bot Amazon Lex V2

Diagram berikut menunjukkan percakapan audio streaming antara pengguna dan bot Amazon Lex V2. Aplikasi terus mengalirkan audio ke bot, dan bot mencari input pengguna dari audio. Dalam contoh ini, pengguna dan bot menggunakan ucapan untuk berkomunikasi. Setiap diagram sesuai dengan ucapan pengguna dan respons bot terhadap ucapan itu.

Diagram berikut menunjukkan awal percakapan antara aplikasi dan bot. Aliran dimulai pada waktu nol (t0).

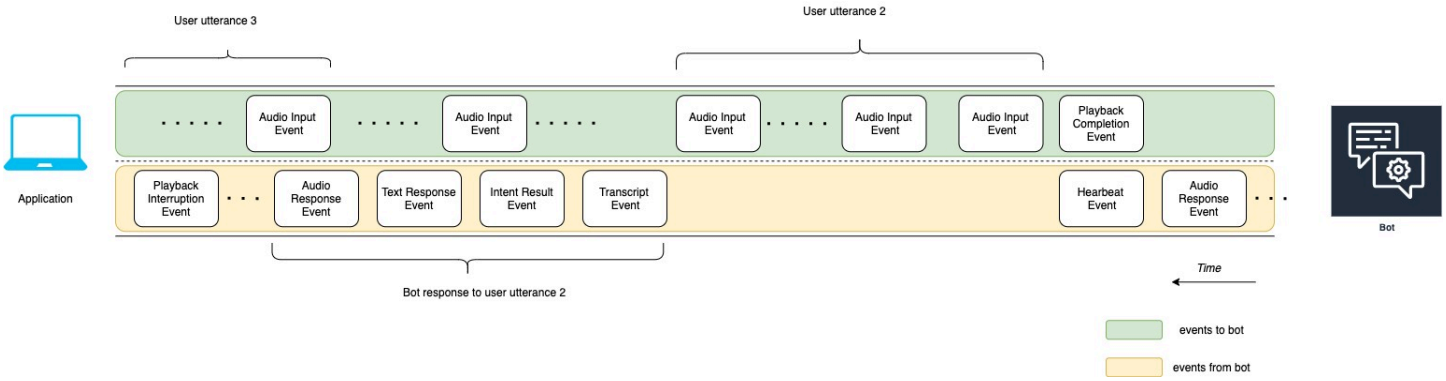


Daftar berikut menjelaskan peristiwa diagram sebelumnya.

- t0: Aplikasi mengirimkan peristiwa konfigurasi ke bot untuk memulai streaming.
- t1: Aplikasi mengalirkan data audio. Data ini dipecah menjadi serangkaian peristiwa masukan dari aplikasi.
- t2: Untuk ucapan pengguna 1, bot mendeteksi peristiwa input audio saat pengguna mulai berbicara.
- t2: Saat pengguna berbicara, bot mengirimkan acara detak jantung untuk mempertahankan koneksi. Ini mengirimkan peristiwa ini sebentar-sebentar untuk memastikan koneksi tidak habis waktu.
- t3: Bot mendeteksi akhir ucapan pengguna.
- t4: Bot mengirimkan kembali peristiwa transkrip yang berisi transkrip pidato pengguna ke aplikasi. Ini adalah awal dari respons Bot terhadap ucapan pengguna 1.
- t5: Bot mengirimkan peristiwa hasil maksud untuk menunjukkan tindakan yang ingin dilakukan pengguna.
- t6: Bot mulai memberikan responsnya sebagai teks dalam acara respons teks.
- t7: Bot mengirimkan serangkaian peristiwa respons audio ke aplikasi untuk diputar bagi pengguna.

- t8: Bot mengirimkan acara detak jantung lain untuk menjaga koneksi sebentar-sebentar.

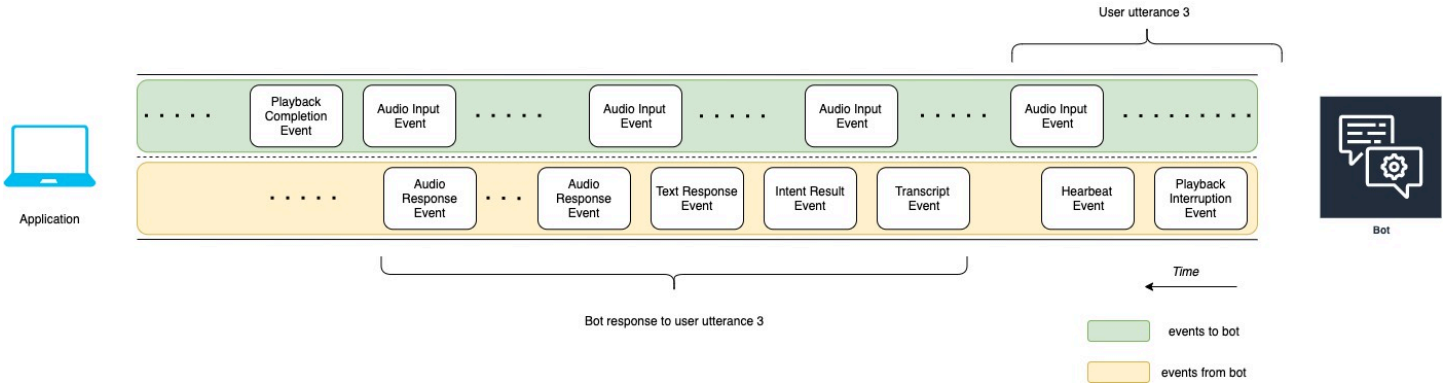
Diagram berikut merupakan kelanjutan dari diagram sebelumnya. Ini menunjukkan aplikasi mengirim acara penyelesaian pemutaran ke bot untuk menunjukkan bahwa ia telah berhenti memutar respons audio untuk pengguna. Aplikasi ini memutar kembali respons Bot terhadap ucapan pengguna 1 kepada pengguna. Pengguna merespons respons Bot terhadap ucapan pengguna 1 dengan ucapan Pengguna 2.



Daftar berikut menjelaskan peristiwa diagram sebelumnya:

- t10: Aplikasi mengirimkan acara penyelesaian pemutaran untuk menunjukkan bahwa ia telah selesai memutar pesan bot kepada pengguna.
- t11: Aplikasi mengirimkan respons pengguna kembali ke bot sebagai ucapan Pengguna 2.
- t12: Untuk respons Bot terhadap ucapan pengguna 2, bot menunggu pengguna berhenti berbicara dan kemudian mulai memberikan respons audio.
- t13: Sementara bot mengirimkan respons Bot ke ucapan pengguna 2 ke aplikasi, bot mendeteksi awal ucapan Pengguna 3. Bot menghentikan respons Bot terhadap ucapan pengguna 2 dan mengirimkan peristiwa gangguan pemutaran.
- t14: Bot mengirimkan peristiwa gangguan pemutaran ke aplikasi untuk memberi sinyal bahwa pengguna telah mengganggu prompt.

Diagram berikut menunjukkan respons Bot terhadap ucapan pengguna 3, dan percakapan berlanjut setelah bot merespons ucapan pengguna.



Menggunakan API untuk memulai percakapan streaming

Saat Anda memulai streaming ke bot Amazon Lex V2, Anda menyelesaikan tugas-tugas berikut:

1. Buat koneksi awal ke server.
2. Konfigurasikan kredenal keamanan dan detail bot. Detail bot mencakup apakah bot mengambil DTMF dan input audio, atau input teks.
3. Kirim acara ke server. Peristiwa ini adalah data teks atau data audio dari pengguna.
4. Proses peristiwa yang dikirim dari server. Pada langkah ini, Anda menentukan apakah output bot disajikan kepada pengguna sebagai teks atau ucapan.

Contoh kode berikut menginisialisasi percakapan streaming dengan bot Amazon Lex V2 dan mesin lokal Anda. Anda dapat memodifikasi kode untuk memenuhi kebutuhan Anda.

Kode berikut adalah contoh permintaan menggunakan AWS SDK untuk Java untuk memulai koneksi ke bot dan mengkonfigurasi rincian bot dan kredensialnya.

```
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();
        }
    }
}
```

Kode berikut adalah contoh permintaan menggunakan AWS SDK untuk Java untuk mengirim acara ke bot. Kode dalam contoh ini menggunakan mikrofon di komputer Anda untuk mengirim acara audio.

```
package com.lex.streaming.sample;
```

```
import org.reactivestreams.Publisher;
import org.reactivestreams.Subscriber;
import
 software.amazon.awssdk.services.lexruntimev2.model.StartConversationRequestEventStream;

/**
 * You use the Events publisher to send events to the Amazon Lex bot. When you
 establish a connection, the bot uses the
 * subscribe() method and enables the events publisher starts sending events to
 * your computer. The bot uses the "request" method of the subscription to make more
 requests. For more information on the request method, see https://github.com/reactive-
streams/reactive-streams-jvm.
 */
public class EventsPublisher implements Publisher<StartConversationRequestEventStream>
 {

    private AudioEventsSubscription audioEventsSubscription;

    @Override
    public void subscribe(Subscriber<? super StartConversationRequestEventStream>
 subscriber) {
        if (audioEventsSubscription == null) {

            audioEventsSubscription = new AudioEventsSubscription(subscriber);
            subscriber.onSubscribe(audioEventsSubscription);

        } else {
            throw new IllegalStateException("received unexpected subscription
 request");
        }
    }

    public void disconnect() {
        if (audioEventsSubscription != null) {
            audioEventsSubscription.disconnect();
        }
    }

    public void stop() {
        if (audioEventsSubscription != null) {
            audioEventsSubscription.stop();
        }
    }
```

```
    public void playbackFinished() {
        if (audioEventsSubscription != null) {
            audioEventsSubscription.playbackFinished();
        }
    }
}
```

Kode berikut adalah contoh permintaan menggunakan AWS SDK untuk Java untuk menangani tanggapan dari bot. Kode dalam contoh ini mengonfigurasi Amazon Lex V2 untuk memutar respons audio kembali kepada Anda.

```
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;
    }

}
```

Untuk mengonfigurasi bot untuk merespons peristiwa input dengan audio, Anda harus terlebih dahulu berlangganan acara audio dari Amazon Lex V2 dan kemudian mengonfigurasi bot untuk memberikan respons audio ke peristiwa input dari pengguna.

Kode berikut adalah AWS SDK untuk Java contoh untuk berlangganan acara audio dari 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);
            }
        }
    }
}
```

AWS SDK untuk Java Contoh berikut mengonfigurasi bot Amazon Lex V2 untuk memberikan respons audio ke peristiwa input.

```
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();
    }
}
```

Pengkodean aliran acara

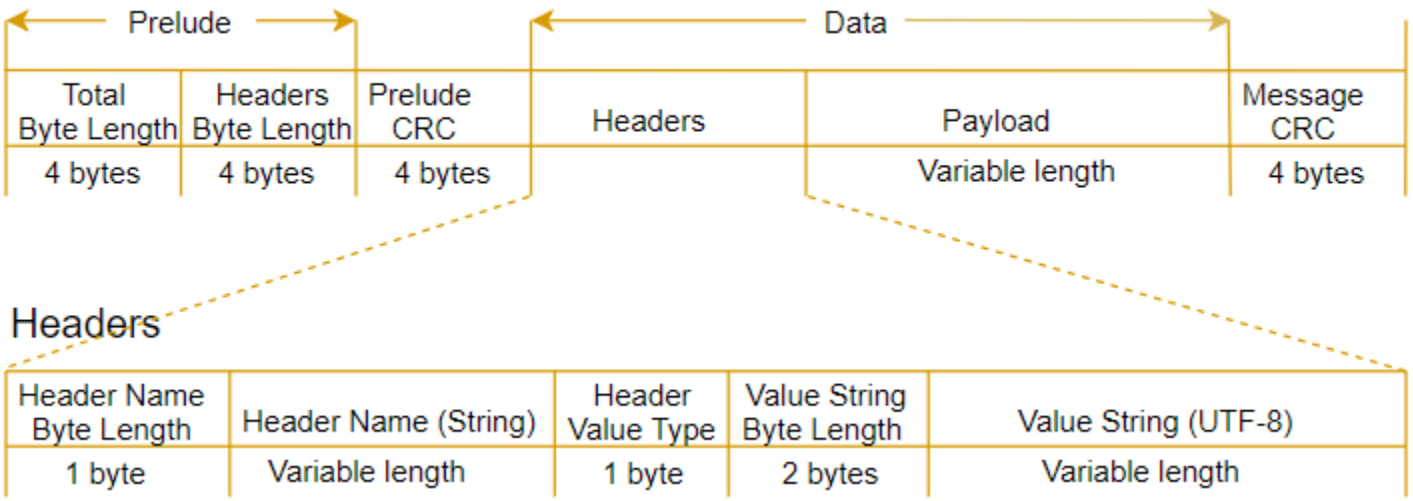
Event stream encoding menyediakan komunikasi dua arah menggunakan pesan antara klien dan server. Bingkai data yang dikirim ke layanan streaming Amazon Lex V2 dikodekan dalam format ini. Tanggapan dari Amazon Lex V2 juga menggunakan pengkodean ini.

Setiap pesan terdiri dari dua bagian: pendahuluan dan data. Bagian pendahuluan berisi panjang byte total pesan dan panjang byte gabungan dari semua header. Bagian data berisi header dan payload.

Setiap bagian diakhiri dengan checksum CRC integer besar-endian 4-byte. Pesan CRC checksum mencakup bagian pendahuluan dan bagian data. Amazon Lex V2 menggunakan CRC32 (sering disebut sebagai GZIP CRC32) untuk menghitung kedua CRC. Untuk informasi selengkapnya tentang CRC32, lihat spesifikasi format file GZIP versi 4.3.

Total overhead pesan, termasuk pendahuluan dan kedua checksum, adalah 16 byte.

Diagram berikut menunjukkan komponen yang membentuk pesan dan header. Ada beberapa header per pesan.



Setiap pesan berisi komponen-komponen berikut:

- Pendahuluan: Selalu ukuran tetap 8 byte, dua bidang masing-masing 4 byte.
  - 4 byte pertama: Total byte-length. Ini adalah panjang byte integer endian besar dari seluruh pesan, termasuk bidang panjang 4-byte itu sendiri.
  - Kedua 4 byte: Header byte-length. Ini adalah panjang byte integer endian besar dari bagian header pesan, tidak termasuk bidang panjang header itu sendiri.
- Prelude CRC: Checksum CRC 4-byte untuk bagian awal pesan, tidak termasuk CRC itu sendiri. Pendahuluan memiliki CRC terpisah dari pesan CRC untuk memastikan bahwa Amazon Lex V2 dapat segera mendeteksi informasi panjang byte yang rusak tanpa menyebabkan kesalahan seperti buffer overruns.
- Header: Metadata yang menganotasi pesan, seperti jenis pesan, jenis konten, dan sebagainya. Pesan memiliki beberapa header. Header adalah pasangan kunci-nilai di mana kuncinya adalah string. UTF-8 Header dapat muncul dalam urutan apapun di bagian header pesan dan setiap header yang diberikan hanya dapat muncul sekali. Untuk jenis header yang diperlukan, lihat bagian berikut.
- Payload: Konten audio atau teks yang dikirim ke Amazon Lex.
- Pesan CRC: Checksum CRC 4-byte dari awal pesan hingga awal checksum. Itu termasuk semua yang ada dalam pesan kecuali CRC itu sendiri.

Setiap header berisi komponen-komponen berikut. Ada beberapa header per frame.

- Nama header byte-length: Panjang byte dari nama header.

- Nama header: Nama header yang menunjukkan jenis header. Untuk nilai yang valid, lihat deskripsi bingkai berikut.
- Jenis nilai header: Pencacahan yang menunjukkan jenis nilai header.
- Nilai panjang string byte: Panjang byte dari string nilai header.
- Nilai header: Nilai string header. Nilai yang valid untuk bidang ini tergantung pada jenis header. Untuk nilai yang valid, lihat deskripsi bingkai berikut.

Mengaktifkan bot Amazon Lex V2 Anda terganggu oleh pengguna

Saat Anda memulai aliran audio dua arah antara bot Amazon Lex V2 dan aplikasi Anda, Anda dapat mengonfigurasi bot untuk mendengarkan input pengguna saat mengirim kembali prompt. Masukan ini dapat diatur sebagai interupsi. Dengan ini, pengguna dapat mengganggu prompt sebelum bot selesai memutarnya kembali. Anda dapat menggunakan konfigurasi ini untuk situasi di mana pengguna mungkin sudah mengetahui jawaban atas pertanyaan, seperti ketika mereka diminta untuk memberikan kode CVV.

Bot tahu kapan pengguna menginterupsi prompt saat mendeteksi input pengguna sebelum aplikasi Anda dapat mengirim kembali suatu peristiwa. `PlaybackCompletion` Ketika pengguna menyela bot, bot mengirimkan file. `PlaybackInterruptionEvent`

Secara default, pengguna dapat mengganggu setiap prompt yang bot streaming ke aplikasi Anda. Anda dapat mengubah pengaturan ini di konsol Amazon Lex V2.

Anda dapat mengubah cara pengguna merespons prompt dengan mengedit slot. Slot adalah bagian dari niat, dan itu adalah sarana yang digunakan pengguna untuk memberi Anda informasi yang Anda inginkan. Setiap slot memiliki prompt bagi pengguna untuk memberi Anda informasi itu. Untuk mempelajari lebih lanjut tentang slot, lihatKonsep inti Amazon Lex V2.

Untuk mengubah apakah pengguna dapat mengganggu prompt (konsol)

1. Masuk ke AWS Management Console dan buka konsol Amazon Lex V2 di konsol Amazon Lex V2.
2. Di bawah Bots, pilih bot.
3. Di bawah Bahasa, pilih bahasa bot.
4. Pilih Lihat maksud.
5. Pilih maksudnya.
6. Untuk Slots, pilih slot.

7. Di bawah Opsi lanjutan, pilih Petunjuk Slot.
8. Pilih Opsi prompt lainnya.
9. Pilih atau batal pilihan Pengguna dapat mengganggu prompt saat sedang dibaca.

Anda dapat menguji fungsi ini dengan membuat bot dengan dua slot dan menentukan bahwa pengguna tidak dapat mengganggu prompt untuk satu slot. Jika Anda mengganggu prompt yang dapat diinterupsi, bot akan mengirimkan acara interupsi pemutaran. Jika Anda menyela yang tidak terputus, prompt terus diputar.

Mengaktifkan bot Amazon Lex V2 untuk menunggu pengguna memberikan informasi lebih lanjut selama jeda

Saat Anda memulai aliran dua arah dari bot Amazon Lex V2 ke aplikasi Anda, Anda dapat mengonfigurasi bot untuk menunggu pengguna memberikan informasi tambahan. Ada keadaan ketika pengguna mungkin tidak siap untuk menanggapi prompt. Misalnya, pengguna mungkin tidak siap untuk memberikan informasi kartu kredit mereka karena dompet mereka ada di ruangan lain.

Dengan menggunakan perilaku Tunggu dan lanjutkan bot Amazon Lex V2, pengguna dapat mengucapkan frasa seperti “tahan sebentar” untuk membuat bot menunggu mereka menemukan informasi dan memberikannya. Saat Anda mengaktifkan perilaku ini, bot mengirimkan pengingat berkala kepada pengguna untuk memberikan informasi. Itu tidak mengirim kembali peristiwa transkrip karena tidak ada ucapan pengguna untuk ditranskripsikan.

Bot Amazon Lex V2 secara otomatis mengelola percakapan streaming. Anda tidak perlu menulis kode tambahan apa pun untuk mengaktifkan fungsi ini. Ketika bot diminta untuk menunggu oleh pengguna, `Intent` adalah `Waiting` dan `type` dari `DialogAction` is`ElicitSlot`. `state` Anda dapat menggunakan informasi ini untuk membantu menyesuaikan aplikasi Anda untuk kebutuhan Anda. Misalnya, Anda dapat mengonfigurasi aplikasi Anda untuk memutar musik saat pengguna mencari kartu kredit mereka.

Anda mengaktifkan perilaku menunggu dan melanjutkan untuk slot individu. Untuk mempelajari lebih lanjut tentang slot, lihatKonsep inti Amazon Lex V2.

Untuk mengaktifkan tunggu dan lanjutkan

1. Masuk ke AWS Management Console dan buka konsol Amazon Lex V2 di konsol Amazon Lex V2.

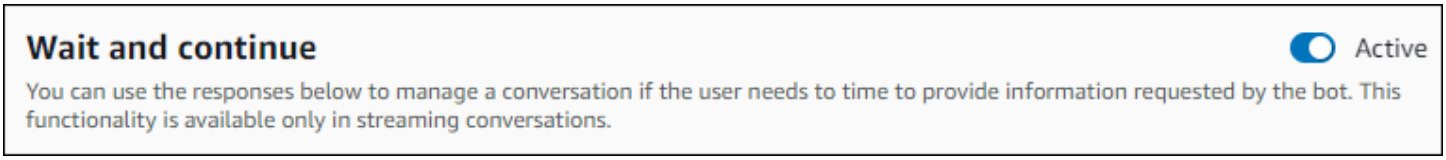
2. Di bawah Bots, pilih bot.
3. Di bawah Bahasa, pilih bahasa bot.
4. Pilih Lihat maksud.
5. Pilih maksudnya.
6. Di bawah Slots, pilih slot.
7. Di bawah Opsi lanjutan, pilih Tunggu dan lanjutkan.
8. Di bawah Tunggu dan lanjutkan tentukan bidang berikut:

   - Respons ketika pengguna ingin bot menunggu — Ini adalah bagaimana bot merespons ketika pengguna memintanya untuk menunggu informasi tambahan.
   - Respons jika pengguna membutuhkan bot untuk terus menunggu — Ini adalah respons yang dikirim bot untuk mengingatkan pengguna bahwa masih menunggu informasi. Anda dapat mengubah seberapa sering bot mengingatkan pengguna.
   - Respons ketika pengguna ingin melanjutkan — Ini adalah respons bot ketika pengguna memiliki informasi yang diminta.

Untuk setiap respons bot, Anda dapat memberikan beberapa variasi respons, dan satu disajikan kepada pengguna secara acak. Anda juga dapat memilih apakah tanggapan ini dapat terganggu oleh pengguna.

Untuk menguji fungsi tunggu dan lanjutkan, konfigurasikan bot Anda untuk menunggu input pengguna dan mulai streaming ke bot Amazon Lex V2. Untuk informasi tentang streaming ke bot, lihatMenggunakan API untuk memulai percakapan streaming.

Anda mungkin perlu mematikan tunggu dan melanjutkan tanggapan. Gunakan sakelar Aktif untuk mengatur apakah respons tunggu dan lanjutkan digunakan atau tidak.



> **Note**
>
> Tunggu dan lanjutkan tersedia di lokasi berikut: Ca-es, De-at, De-de, en-AU, en-GB, en-in, en-US, en-ZA, es-419, es-ES, es-US, fr-Ca, fr-Fr, IT-it, Ja-jp, Ko-kr, Pt-BR, Pt-PT, Zh-CN, Zh-hk.

Mengkonfigurasi pembaruan kemajuan pemenuhan untuk bot Lex V2 Anda

Ketika fungsi Lambda pemenuhan untuk intent dipanggil, bot tidak mengirim respons sampai fungsi selesai. Jika fungsi Lambda membutuhkan waktu lebih dari beberapa detik untuk diselesaikan, pengguna mungkin berpikir bahwa bot tidak responsif. Untuk mengatasi ini, Anda dapat mengonfigurasi bot Anda untuk mengirim pembaruan kepada pengguna saat fungsi Lambda pemenuhan berjalan sehingga pengguna tahu bahwa bot masih berfungsi berdasarkan permintaan mereka.

Saat Anda menambahkan pembaruan pemenuhan ke intent, bot merespons pada awal pemenuhan dan secara berkala saat pemenuhan sedang berlangsung. Saat Anda mengonfigurasi respons awal, Anda dapat menentukan penundaan sebelum bot mengirimkan respons. Dengan ini, Anda dapat mendukung kasus di mana pemenuhan tidak selesai dengan relatif cepat. Saat Anda mengonfigurasi respons pembaruan, Anda menentukan frekuensi yang Anda inginkan pembaruan dikirim. Anda juga mengonfigurasi batas waktu untuk membatasi waktu fungsi pemenuhan harus dijalankan.

Anda juga dapat menambahkan tanggapan pasca-pemenuhan ke bot. Ini memungkinkan bot untuk mengirim respons yang berbeda tergantung pada apakah pemenuhan berhasil, gagal, atau habis waktu.

Pembaruan pemenuhan hanya digunakan saat berinteraksi dengan bot menggunakan operasi. StartConversation Anda dapat menggunakan pembaruan pasca-pemenuhan saat berinteraksi dengan bot menggunakan StartConversation,, dan operasi RecognizeTextRecognizeUtterance

Pembaruan pemenuhan

Pembaruan pemenuhan dikirim saat fungsi Lambda Anda memenuhi maksud. Saat Anda mengaktifkan pembaruan pemenuhan, Anda memberikan respons awal yang dikirim pada awal pemenuhan dan respons pembaruan yang dikirim secara berkala saat pemenuhan sedang berlangsung.

Saat menentukan respons pembaruan, Anda juga menentukan batas waktu yang menentukan berapa lama fungsi pemenuhan dapat berjalan. Anda dapat menentukan panjang batas waktu hingga 15 menit (900 detik).

Jika Anda menonaktifkan pembaruan pemenuhan dengan menyetel `active` ke false di konsol atau menggunakan UpdateIntentoperasi CreateIntentatau, batas waktu yang ditentukan untuk pembaruan pemenuhan tidak digunakan dan batas waktu default 30 detik digunakan sebagai gantinya.

Jika fungsi pemenuhan habis waktu, Amazon Lex V2 melakukan salah satu dari tiga hal:

- Respons pasca-pemenuhan dikonfigurasi dan aktif — mengembalikan respons batas waktu.
- Respons pasca-pemenuhan dikonfigurasi dan tidak aktif — mengembalikan pengecualian.
- Respons pasca-pemenuhan tidak dikonfigurasi — mengembalikan pengecualian.

Mulai respon

Amazon Lex V2 mengembalikan respons awal saat fungsi pemenuhan Lambda dipanggil selama percakapan streaming. Ini biasanya memberitahu pengguna bahwa memenuhi maksud membutuhkan waktu dan bahwa mereka harus menunggu. Respons awal tidak dikembalikan saat Anda menggunakan `RecognizeUtterance` operasi `RecognizeText` atau.

Anda dapat menentukan hingga lima pesan respons. Amazon Lex V2 memilih salah satu pesan untuk dimainkan kepada pengguna.

Anda dapat mengonfigurasi penundaan antara saat fungsi Lambda dipanggil dan saat respons mulai dikembalikan. Respons awal tidak dikembalikan jika fungsi Lambda menyelesaikan pekerjaannya sebelum penundaan selesai.

Anda dapat menggunakan `active` sakelar di konsol atau FulfillmentUpdatesSpecificationstruktur untuk mengaktifkan dan menonaktifkan respons awal. `active`Kapan salah, respons awal tidak dimainkan.

Perbarui tanggapan

Amazon Lex mengembalikan respons pembaruan secara berkala selama percakapan streaming saat fungsi pemenuhan Lambda sedang berjalan. Respons pembaruan tidak diputar saat Anda menggunakan `RecognizeText` atau `RecognizeUtterance` operasi. Anda dapat mengonfigurasi seberapa sering respons pembaruan diputar. Misalnya, Anda dapat memutar respons pembaruan setiap 30 detik saat fungsi pemenuhan berjalan untuk memberi tahu pengguna bahwa proses sedang berjalan dan bahwa mereka harus terus menunggu.

Anda dapat menentukan hingga lima pesan pembaruan. Amazon Lex V2 memilih pesan untuk diputar ke pengguna. Menggunakan beberapa pesan membuat pembaruan tidak berulang.

Jika pengguna memberikan input melalui suara, DTMF, atau teks saat fungsi Lambda pemenuhan berjalan, Amazon Lex V2 mengembalikan respons pembaruan kepada pengguna.

Jika fungsi Lambda menyelesaikan pekerjaannya sebelum periode pembaruan pertama berakhir, respons pembaruan tidak akan dikembalikan.

Anda dapat menggunakan `active` sakelar di konsol atau FulfillmentUpdatesSpecificationstruktur untuk mengaktifkan dan menonaktifkan respons pembaruan. `active`Kapan salah, respons pembaruan tidak dikembalikan.

Respon pasca-pemenuhan

Amazon Lex V2 mengembalikan respons pasca-pemenuhan saat fungsi pemenuhan berakhir. Respons pasca-pemenuhan dapat digunakan saat memenuhi maksud apa pun, tidak hanya saat streaming percakapan. Respons pasca-pemenuhan memungkinkan pengguna mengetahui bahwa fungsinya sudah lengkap dan hasilnya.

Anda dapat menggunakan `active` sakelar di konsol atau PostFulfillmentStatusSpecificationstruktur untuk mengaktifkan dan menonaktifkan respons pasca-pemenuhan. Kapan `active` salah, responsnya tidak dimainkan.

Ada tiga jenis tanggapan pasca-pemenuhan:

- Sukses — kembali ketika pemenuhan fungsi Lambda menyelesaikan pekerjaannya dengan sukses. Jika tanggapan pasca-pemenuhan tidak aktif. Amazon Lex V2 mengambil tindakan yang dikonfigurasi berikutnya.
- Timeout — dikembalikan jika fungsi Lambda tidak menyelesaikan pekerjaannya sebelum periode batas waktu yang dikonfigurasi berlalu. Jika tanggapan pasca-pemenuhan tidak aktif, Amazon Lex V2 mengembalikan pengecualian.
- Kegagalan — dikembalikan saat fungsi Lambda mengembalikan status `Failed` dalam respons atau saat Amazon Lex V2 mengalami kesalahan saat memenuhi maksud. Jika tanggapan pasca-pemenuhan tidak aktif, Amazon Lex V2 mengembalikan pengecualian.

Anda dapat menentukan hingga lima pesan untuk setiap jenis. Amazon Lex V2 memilih salah satu pesan untuk dimainkan kepada pengguna.

Tidak seperti respons pemenuhan awal dan pemenuhan pembaruan, respons pasca-pemenuhan diputar ulang untuk percakapan streaming dan non-streaming.

Anda juga memiliki opsi untuk mengganti pesan ini dengan mengonfigurasi fungsi Lambda untuk mengembalikan pesan pasca-pemenuhan.

> **Note**
>
> Jika maksud memiliki respons penutupan, itu dikembalikan setelah respons pasca-pemenuhan.

Contoh pasca-pemenuhan untuk Lex V2

Untuk lebih memahami respons pasca-pemenuhan, mari kita ambil, sebagai contoh, *BookTrip* bot, dibuat untuk membantu merencanakan perjalanan, dengan *BookFlight* maksud, dikonfigurasi dengan fungsi Lambda pemenuhan yang memesan penerbangan pelanggan dengan maskapai penerbangan. Setelah slot untuk diperoleh*BookFlight*, Amazon Lex V2 memanggil fungsi Lambda pemenuhan. Selama proses pemenuhan ini salah satu dari tiga hasil berikut dapat terjadi:

- Sukses — Penerbangan berhasil dipesan.
- Timeout — Proses pemesanan memakan waktu lebih lama dari pemenuhan yang dikonfigurasi waktu eksekusi Lambda (misalnya, jika maskapai tidak dapat dihubungi dalam waktu yang ditentukan).
- Kegagalan — Pemesanan gagal karena alasan lain.

Anda dapat memanfaatkan respons pasca-pemenuhan untuk memberikan respons yang lebih bermakna kepada pelanggan Anda dalam setiap situasi ini. Contoh untuk setiap situasi adalah sebagai berikut:

- Tanggapan sukses — “Kami berhasil memesan tiket Anda dan telah mengirimi Anda email konfirmasi. Jangan ragu untuk menghubungi kami menggunakan informasi kontak yang disediakan dalam email itu jika Anda memiliki pertanyaan.”
- Tanggapan batas waktu — “Karena lalu lintas yang padat di sistem kami, pemesanan tiket Anda memakan waktu lebih lama dari yang diharapkan. Kami memiliki permintaan Anda dalam antrian kami dan telah mengirimi Anda email dengan nomor referensi yang sesuai dengan permintaan ini. Setelah kami memesan tiket, kami akan mengirimkan konfirmasi reservasi kepada Anda. Jangan ragu untuk menghubungi kami menggunakan informasi kontak yang disediakan dalam email itu jika Anda memiliki pertanyaan.”

> **Note**
>
> Jika Anda tidak mengonfigurasi pesan batas waktu, Amazon Lex V2 memunculkan kesalahan 4XX yang sesuai dengan kasus penggunaan.

- Tanggapan kegagalan — “Sayangnya, kami tidak dapat memesan tiket Anda. Kami telah mengirim email dengan rincian mengenai masalah yang kami temui saat memesan reservasi Anda.”

Mengkonfigurasi batas waktu untuk menangkap input pengguna dengan bot Lex V2

API streaming Amazon Lex V2 memungkinkan bot mendeteksi ucapan secara otomatis dalam input pengguna. Saat membuat intent atau slot, Anda dapat mengonfigurasi aspek ucapan, seperti durasi maksimum ucapan, batas waktu saat menunggu input pengguna, atau karakter akhir untuk input DTMF. Anda dapat menyesuaikan perilaku bot untuk kasus penggunaan Anda. Misalnya, Anda dapat membatasi jumlah digit untuk nomor kartu kredit menjadi 16.

Anda juga dapat mengonfigurasi batas waktu melalui atribut sesi saat memulai percakapan dengan bot, dan menimpa mereka di fungsi Lambda Anda jika perlu.

Kunci konfigurasi untuk atribut menggunakan sintaks berikut:

```
x-amz-lex:<InputType>:<BehaviorName>:<IntentName>:<SlotName>
```

`InputType`bisa**`audio`**,**`dtmf`**, atau**`text`**.

Anda dapat mengonfigurasi pengaturan default untuk semua maksud atau slot di bot dengan menentukan **`*`** nama maksud atau slot. Pengaturan khusus maksud atau slot apa pun lebih diutamakan daripada pengaturan default.

Amazon Lex V2 menyediakan atribut sesi yang telah ditentukan untuk mengelola cara StartConversationoperasi bekerja dengan teks, suara, atau input DTMF ke bot Anda. Semua atribut yang telah ditentukan ada di `x-amz-lex` namespace.

Anda dapat mengonfigurasi pengaturan default untuk semua maksud, slot, atau subslot di bot dengan menentukan nama `*` maksud atau slot. Setiap maksud atau pengaturan khusus slot lebih diutamakan daripada setelan default. Gunakan pola ini untuk semua batas waktu di bawah ini.

Untuk subslot slot komposit Anda dapat memisahkannya. . Contoh:

```
<slotName>.<subSlotName>
```

```
x-amz-lex:allow-interrupt:<intentName>:<slotName>.<subSlotName>
```

Ekspresi	Skenario
Niat:Slot. SubSlot	Berlaku hanya untuk sub slot bernama 'SubSlot' di dalam slot komposit bernama 'Slot'
Niat:Slot. *	Berlaku untuk setiap sub slot di dalam slot komposit bernama 'Slot'
Maksud: *. SubSlot	Berlaku hanya untuk sub slot bernama 'SubSlot' di dalam slot komposit apa pun
Maksud: * . *	Berlaku untuk setiap sub slot di dalam slot komposit

Cara kerja perilaku interupsi di bot Lex V2

Anda dapat mengatur perilaku interupsi untuk bot. Atribut didefinisikan oleh Amazon Lex V2.

Izinkan interupsi

```
x-amz-lex:allow-interrupt:<intentName>:<slotName>
```

Mendefinisikan apakah pengguna dapat mengganggu prompt yang dimainkan oleh bot Amazon Lex V2. Anda dapat mematikannya secara selektif.

Default: BETUL

Mengatur batas waktu untuk input suara

Anda dapat mengatur nilai batas waktu untuk interaksi suara dengan bot Anda menggunakan atribut sesi. Atribut ditentukan oleh Amazon Lex V2. Atribut ini memungkinkan Anda menentukan berapa lama Amazon Lex V2 menunggu pelanggan selesai berbicara sebelum mengumpulkan pidato input.

Semua atribut ini ada di `x-amz-lex:audio` namespace.

Panjang ucapan maksimum

```
x-amz-lex:audio:max-length-ms:<intentName>:<slotName>
```

Mendefinisikan berapa lama Amazon Lex V2 menunggu sebelum input ucapan terpotong dan pidato dikembalikan ke aplikasi Anda. Anda dapat menambah panjang input ketika Anda mengharapkan tanggapan panjang, atau jika Anda ingin memberi pelanggan lebih banyak waktu untuk memberikan informasi.

Default: 55.000 milidetik (55 detik). Nilai maksimumnya adalah 55.000 milidetik (55 detik)

Jika Anda menyetel `max-length-ms` atribut ke lebih dari 55.000 milidetik, nilainya akan default menjadi 55.000 milidetik.

Batas waktu suara

```
x-amz-lex:audio:start-timeout-ms:<intentName>:<slotName>
```

Berapa lama bot menunggu sebelum berasumsi bahwa pelanggan tidak akan berbicara. Anda dapat meningkatkan waktu dalam situasi di mana pelanggan mungkin memerlukan lebih banyak waktu untuk menemukan atau mengingat informasi sebelum berbicara. Misalnya, Anda mungkin ingin memberi pelanggan waktu untuk mengeluarkan kartu kredit mereka sehingga mereka dapat memasukkan nomornya.

Default: 4.000 milidetik (4 detik)

Batas waktu diam

```
x-amz-lex:audio:end-timeout-ms:<intentName>:<slotName>
```

Berapa lama bot menunggu setelah pelanggan berhenti berbicara untuk menganggap ucapan selesai. Anda dapat meningkatkan waktu dalam situasi di mana periode keheningan diharapkan sambil memberikan masukan.

Default: 600 milidetik (0,6 detik)

Izinkan input audio

```
x-amz-lex:allow-audio-input:<intentName>:<slotName>
```

Anda dapat mengaktifkan atribut ini sehingga bot menerima input pengguna hanya melalui modalitas audio. Bot tidak akan menerima input audio jika flag ini disetel ke false. Nilai diatur ke true secara default.

Default: BETUL

Batas waktu untuk masukan teks

Gunakan atribut sesi berikut untuk menentukan bagaimana bot Anda berperilaku dengan mode percakapan teks.

Atribut ini ada di `x-amz-lex:text` namespace.

Ambang batas waktu mulai

```
x-amz-lex:text:start-timeout-ms:<intentName>:<slotName>
```

Berapa lama bot menunggu sebelum meminta kembali pelanggan untuk input teks. Anda dapat meningkatkan waktu dalam situasi di mana Anda ingin memberi pelanggan lebih banyak waktu untuk menemukan atau mengingat informasi sebelum memberikan input teks. Misalnya, Anda mungkin ingin memberi pelanggan lebih banyak waktu untuk menemukan detail pesanan mereka. Atau, Anda dapat mengurangi ambang batas untuk meminta pelanggan lebih awal.

Default: 30.000 milidetik (30 detik)

Tetapkan konfigurasi untuk input DTMF

Gunakan atribut sesi berikut untuk menentukan bagaimana bot Amazon Lex V2 Anda merespons input DTMF saat menggunakan percakapan audio.

Semua atribut ini ada di `x-amz-lex:dtmf` namespace.

Karakter penghapusan

```
x-amz-lex:dtmf:deletion-character:<intentName>:<slotName>
```

Karakter DTMF yang menghapus digit DTMF yang terakumulasi dan segera mengakhiri input.

Default: *

Karakter akhir

```
x-amz-lex:dtmf:end-character:<intentName>:<slotName>
```

Karakter DTMF yang segera mengakhiri input. Jika pengguna tidak menekan karakter ini, input berakhir setelah batas waktu akhir.

Default: #

Batas waktu akhir

```
x-amz-lex:dtmf:end-timeout-ms:<intentName>:<slotName>
```

Berapa lama bot harus menunggu dari input karakter DTMF terakhir sebelum mengasumsikan bahwa input telah selesai.

Default: 5000 milidetik (5 detik)

Jumlah maksimum digit DTMF per ucapan

```
x-amz-lex:dtmf:max-length:<intentName>:<slotName>
```

Jumlah maksimum digit DTMF yang diizinkan dalam suatu ucapan. Misalnya, Anda dapat menetapkan nilai ini ke 16 untuk membatasi jumlah karakter yang dapat dimasukkan untuk nomor kartu kredit. Nilai ini tidak dapat ditingkatkan.

Default: 1024 karakter

Izinkan masukan DTMF

Anda dapat mengatur jenis input yang dapat diterima bot menggunakan atribut sesi. Atribut ditentukan oleh Amazon Lex V2.

```
x-amz-lex:allow-dtmf-input:<intentName>:<slotName>
```

Anda dapat mengaktifkan atribut ini sehingga bot menerima input pengguna melalui modalitas DTMF. Bot tidak akan menerima input DTMF jika flag ini disetel ke false. Nilai diatur ke true secara default.

Default: BETUL

Mengimpor dan mengekspor bot di Lex V2

Anda dapat mengekspor definisi bot, lokal bot, atau kosakata khusus dan kemudian mengimpornya kembali untuk membuat sumber daya baru atau menimpa sumber daya yang ada di akun AWS. Misalnya, Anda dapat mengekspor bot dari akun pengujian dan kemudian membuat salinan bot di akun produksi Anda. Anda juga dapat menyalin bot dari satu Wilayah AWS ke Wilayah lain.

Anda dapat mengubah sumber daya dari sumber daya yang diekspor sebelum mengimpornya. Misalnya, Anda dapat mengekspor bot dan kemudian mengedit file JSON untuk slot untuk menambah atau menghapus ucapan elisitasi nilai slot dari slot tertentu. Setelah Anda selesai mengedit definisi, Anda dapat mengimpor file yang dimodifikasi.

Topik

- Mengekspor bot dari Lex V2
- Mengimpor bot di Lex V2
- Menggunakan kata sandi saat mengimpor atau mengekspor
- Format JSON untuk mengimpor dan mengekspor bot di Lex V2

Mengekspor bot dari Lex V2

Anda mengekspor bot, lokal bot, atau kosakata khusus menggunakan konsol atau operasi. `CreatExport` Anda menentukan sumber daya yang akan diekspor, dan Anda dapat memberikan kata sandi opsional untuk membantu melindungi file.zip saat memulai ekspor. Setelah Anda mengunduh file.zip, Anda harus menggunakan kata sandi untuk mengakses file sebelum Anda dapat menggunakannya. Untuk informasi selengkapnya, lihat Menggunakan kata sandi saat mengimpor atau mengekspor.

Mengekspor adalah operasi asinkron. Setelah Anda memulai ekspor, Anda dapat menggunakan konsol atau `DescribeExport` operasi untuk memantau kemajuan ekspor. Setelah ekspor selesai, konsol atau `DescribeExport` operasi menunjukkan status`COMPLETED`, dan konsol mengunduh file ekspor.zip ke browser Anda. Jika Anda menggunakan `DescribeExport` operasi ini, Amazon Lex V2 menyediakan URL Amazon S3 yang telah ditandatangani sebelumnya tempat Anda dapat mengunduh hasil ekspor. URL unduhan hanya tersedia selama lima menit, tetapi Anda bisa mendapatkan URL baru dengan memanggil `DescribeExport` operasi lagi.

Anda dapat melihat riwayat ekspor untuk sumber daya dengan konsol atau dengan `ListExports` operasi. Hasilnya menunjukkan ekspor beserta statusnya saat ini. Ekspor tersedia dalam sejarah selama tujuh hari.

Saat Anda mengekspor `Draft` versi bot atau lokal bot, definisi dalam file JSON mungkin dalam keadaan tidak konsisten karena `Draft` versi lokal bot atau bot dapat diubah saat ekspor sedang berlangsung. Jika `Draft` versi diubah saat sedang diekspor, perubahan mungkin tidak termasuk dalam file ekspor.

Saat Anda mengekspor lokal bot, Amazon Lex mengekspor semua informasi yang mendefinisikan lokal, termasuk lokal, kosakata khusus, maksud, jenis slot, dan slot.

Saat Anda mengekspor bot, Amazon Lex mengekspor semua lokal yang ditentukan untuk bot, termasuk maksud, jenis slot, dan slot. Item berikut tidak diekspor dengan bot:

- Alias bot
- Peran ARN terkait dengan bot
- Tag yang terkait dengan bot dan alias bot
- Kait kode Lambda yang terkait dengan alias bot

Peran ARN dan tag dimasukkan sebagai parameter permintaan saat Anda mengimpor bot. Anda perlu membuat alias bot dan menetapkan kait kode Lambda setelah mengimpor, jika perlu.

Anda dapat menghapus ekspor dan file.zip terkait menggunakan konsol atau `DeleteExport` operasi.

Untuk contoh mengekspor bot menggunakan konsol, lihatMengekspor bot Lex V2 (konsol).

Izin IAM diperlukan untuk mengekspor bot di Lex V2

Untuk mengekspor bot, lokal bot, dan kosakata khusus, pengguna yang menjalankan ekspor harus memiliki izin IAM berikut.

API	• Tindakan IAM yang diperlukan	Sumber daya
CreateExport	• CreateExport	Bot
UpdateExport	• UpdateExport	Bot

API	• Tindakan IAM yang diperlukan	Sumber daya
DescribeExport	• DescribeExport • DescribeBot • DescribeCustomVocabulary • DescribeLocale • DescribeIntent • DescribeSlot • DescribeSlotType • ListLocale • ListIntent • ListSlot • ListSlotType	Bot
DescribeExportuntuk kosakata khusus	• DescribeExport • DescribeCustomVocabulary	bot
DeleteExport	• DeleteExport	Bot
ListExports	• ListExports	*

Untuk contoh kebijakan IAM, lihat Izinkan pengguna untuk mengekspor bot dan lokal bot .

Mengekspor bot Lex V2 (konsol)

Anda dapat mengekspor bot dari daftar bot, dari daftar versi, atau dari halaman detail versi. Saat Anda memilih versi, Amazon Lex V2 mengekspor versi itu. Instruksi berikut mengasumsikan bahwa Anda mulai mengekspor bot dari daftar bot, tetapi ketika Anda memulai dengan versi langkah-langkahnya sama.

Untuk mengekspor bot menggunakan konsol

1. Masuk ke AWS Management Console dan buka konsol Amazon Lex V2 di https://console.aws.amazon.com/lexv2/rumah.
2. Dari daftar bot, pilih bot yang akan diekspor.

3. Dari Tindakan, pilih Ekspor.
4. Pilih versi bot, platform, dan format ekspor.
5. (Opsional) Masukkan kata sandi untuk file.zip. Memberikan kata sandi membantu melindungi arsip keluaran.
6. Pilih Ekspor.

Setelah Anda memulai ekspor, Anda kembali ke daftar bot. Untuk memantau kemajuan ekspor, gunakan daftar riwayat impor/ekspor. Ketika status ekspor Selesai, konsol secara otomatis mengunduh file.zip ke komputer Anda.

Untuk mengunduh ekspor lagi, pada import/export daftar, pilih ekspor, lalu pilih Unduh. Anda dapat memberikan kata sandi untuk file.zip yang diunduh.

Untuk mengekspor bahasa bot

1. Masuk ke AWS Management Console dan buka konsol Amazon Lex V2 di https://console.aws.amazon.com/lexv2/rumah.
2. Dari daftar bot, pilih bot yang bahasanya ingin Anda ekspor.
3. Dari Tambah bahasa, pilih Lihat bahasa.
4. Dari daftar Semua bahasa, pilih bahasa yang akan diekspor.
5. Dari Tindakan, pilih Ekspor.
6. Pilih versi bot, platform, dan format.
7. (Opsional) Masukkan kata sandi untuk file.zip. Memberikan kata sandi membantu melindungi arsip keluaran.
8. Pilih Ekspor.

Setelah Anda memulai ekspor, Anda kembali ke daftar bahasa. Untuk memantau kemajuan ekspor, gunakan daftar riwayat impor/ekspor. Ketika status ekspor Selesai, konsol secara otomatis mengunduh file.zip ke komputer Anda.

Untuk mengunduh ekspor lagi, pada import/export daftar, pilih ekspor, lalu pilih Unduh. Anda dapat memberikan kata sandi untuk file.zip yang diunduh.

Mengimpor bot di Lex V2

Untuk menggunakan konsol untuk mengimpor bot, lokal bot, atau kosakata khusus yang diekspor sebelumnya, Anda memberikan lokasi file di komputer lokal Anda dan kata sandi opsional untuk membuka kunci file. Sebagai contoh, lihat Mengimpor bot Lex V2 (konsol).

Saat Anda menggunakan API, mengimpor sumber daya adalah proses tiga langkah:

1. Buat URL unggahan menggunakan `CreateUploadUrl` operasi. Anda tidak perlu membuat URL unggahan saat menggunakan konsol.
2. Unggah file.zip yang berisi definisi sumber daya.
3. Mulai impor dengan `StartImport` operasi.

URL unggahan adalah URL Amazon S3 yang telah ditandatangani sebelumnya dengan izin menulis. URL tersedia selama lima menit setelah dibuat. Jika kata sandi Anda melindungi file.zip, Anda harus memberikan kata sandi saat memulai impor. Untuk informasi selengkapnya, lihat Menggunakan kata sandi saat mengimpor atau mengekspor.

Impor adalah proses asinkron. Anda dapat memantau kemajuan impor menggunakan konsol atau `DescribeImport` operasi.

Saat Anda mengimpor bot atau bot lokal, mungkin ada konflik antara nama sumber daya dalam file impor dan nama sumber daya yang ada di Amazon Lex V2. Amazon Lex V2 dapat menangani konflik dengan tiga cara:

- Gagal saat konflik - Impor berhenti dan tidak ada sumber daya yang diimpor dari file impor.zip.
- Menimpa - Amazon Lex V2 mengimpor semua sumber daya dari file import .zip dan mengganti sumber daya yang ada dengan definisi dari file impor.
- Append - Amazon Lex V2 mengimpor semua sumber daya dari file import .zip dan menambahkan ke sumber daya yang ada dengan definisi dari file impor. Ini hanya tersedia untuk lokal bot.

Anda dapat melihat daftar impor ke sumber daya menggunakan konsol atau `ListImports` operasi. Impor tetap dalam daftar selama tujuh hari. Anda dapat menggunakan konsol atau `DescribeImport` operasi untuk melihat detail tentang impor tertentu.

Anda juga dapat menghapus impor dan file.zip terkait menggunakan konsol atau `DeleteImport` operasi.

Untuk contoh mengimpor bot menggunakan konsol, lihatMengimpor bot Lex V2 (konsol).

Izin IAM diperlukan untuk mengimpor

Untuk mengimpor bot, lokal bot, dan kosakata khusus, pengguna yang menjalankan impor harus memiliki izin IAM berikut.

API	Tindakan IAM yang diperlukan	Sumber daya
CreateUploadUrl	• CreateUploadUrl	*
StartImportuntuk bot dan bot lokal	• StartImport • saya: PassRole • CreateBot • CreateCustomVocabulary • CreateLocale • CreateIntent • CreateSlot • CreateSlotType • UpdateBot • UpdateCustomVocabulary • UpdateLocale • UpdateIntent • UpdateSlot • UpdateSlotType • DeleteBot • DeleteCustomVocabulary • DeleteLocale • DeleteIntent • DeleteSlot • DeleteSlotType	1. Untuk mengimpor bot baru: bot, bot alias. 2. Untuk menimpa bot yang ada: bot. 3. Untuk mengimpor lokal baru: bot.
StartImportuntuk kosakata khusus	• StartImport • CreateCustomVocabulary	bot

API	Tindakan IAM yang diperlukan	Sumber daya
	• DeleteCustomVocabulary • UpdateCustomVocabulary	
DescribeImport	• DescribeImport	Bot
DeleteImport	• DeleteImport	Bot
ListImports	• ListImports	*

Untuk contoh kebijakan IAM, lihat Izinkan pengguna untuk mengimpor bot dan lokal bot .

Mengimpor bot Lex V2 (konsol)

Untuk mengimpor bot menggunakan konsol

1. Masuk ke AWS Management Console dan buka konsol Amazon Lex V2 di https://console.aws.amazon.com/lexv2/rumah.
2. Dari Action, pilih Impor.
3. Dalam file Input, beri nama bot dan kemudian pilih file.zip yang berisi file JSON yang menentukan bot.
4. Jika file.zip dilindungi kata sandi, masukkan kata sandi untuk file.zip. Kata sandi yang melindungi arsip adalah opsional, tetapi membantu melindungi konten.
5. Buat atau masukkan peran IAM yang menentukan izin untuk bot Anda.
6. Tunjukkan apakah bot Anda tunduk pada Undang-Undang Perlindungan Privasi Online Anak-anak (COPPA).
7. Berikan pengaturan batas waktu idle untuk bot Anda. Jika Anda tidak memberikan nilai, nilai dari file zip digunakan. Jika file.zip tidak berisi pengaturan batas waktu, Amazon Lex V2 menggunakan default 300 detik (lima menit).
8. (Opsional) Tambahkan tag untuk bot Anda.
9. Pilih apakah akan memperingatkan tentang menimpa bot yang ada dengan nama yang sama. Jika Anda mengaktifkan peringatan, jika bot yang Anda impor akan menimpa bot yang ada, Anda menerima peringatan dan bot tidak diimpor. Jika Anda menonaktifkan peringatan, bot yang diimpor menggantikan bot yang ada dengan nama yang sama.
10. Pilih Impor.

Setelah Anda memulai impor, Anda kembali ke daftar bot. Untuk memantau kemajuan impor, gunakan daftar riwayat Impor/ekspor. Ketika status impor Selesai, Anda dapat memilih bot dari daftar bot untuk memodifikasi atau membangun bot.

Untuk mengimpor bahasa bot

1. Masuk ke AWS Management Console dan buka konsol Amazon Lex V2 di https://console.aws.amazon.com/lexv2/rumah.
2. Dari daftar bot, pilih bot yang ingin Anda impor bahasa.
3. Dari Tambah bahasa, pilih Lihat bahasa.
4. Dari Tindakan, pilih impor.
5. Di file Input, pilih file yang berisi bahasa yang akan diimpor. Jika Anda melindungi file.zip, berikan kata sandi di Kata Sandi.
6. Dalam Bahasa, pilih bahasa yang akan diimpor sebagai. Bahasa tidak harus cocok dengan bahasa dalam file impor. Anda dapat menyalin maksud dari satu bahasa ke bahasa lain.
7. Di Voice, pilih suara Amazon Polly yang akan digunakan untuk interaksi suara, atau pilih Tidak Ada untuk bot teks saja.
8. Di ambang skor Keyakinan, masukkan ambang batas tempat Amazon Lex V2 menyisipkan`AMAZON.FallbackIntent`, the`AMAZON.KendraSearchIntent`, atau keduanya saat mengembalikan maksud alternatif.
9. Pilih apakah akan memperingatkan tentang menimpa bahasa yang ada. Jika Anda mengaktifkan peringatan, jika bahasa yang Anda impor akan menimpa bahasa yang ada, Anda menerima peringatan dan bahasa tersebut tidak diimpor. Jika Anda menonaktifkan peringatan, bahasa yang diimpor menggantikan bahasa yang ada.
10. Pilih Impor untuk mulai mengimpor bahasa.

Setelah Anda memulai impor, Anda kembali ke daftar bahasa. Untuk memantau kemajuan impor, gunakan daftar riwayat Impor/ekspor. Ketika status impor Selesai, Anda dapat memilih bahasa dari daftar bot untuk memodifikasi atau membangun bot.

Menggunakan kata sandi saat mengimpor atau mengekspor

Amazon Lex V2 dapat melindungi arsip ekspor Anda dengan kata sandi atau membaca arsip impor Anda yang dilindungi menggunakan kompresi file.zip standar. Anda harus selalu melindungi arsip impor dan ekspor Anda dengan kata sandi.

Amazon Lex V2 mengirimkan arsip ekspor Anda ke bucket S3, dan tersedia untuk Anda dengan URL S3 yang telah ditandatangani sebelumnya. URL hanya tersedia selama lima menit. Arsip tersedia untuk siapa saja yang memiliki akses ke URL unduhan. Untuk membantu melindungi data dalam arsip, berikan kata sandi saat Anda mengekspor sumber daya. Jika Anda perlu mendapatkan arsip setelah URL kedaluwarsa, Anda dapat menggunakan konsol atau `DescribeExport` operasi untuk mendapatkan URL baru.

Jika Anda kehilangan kata sandi untuk arsip ekspor, Anda dapat membuat kata sandi baru untuk file yang ada dengan memilih Unduh dari tabel import/export riwayat atau dengan menggunakan `UpdateExport` operasi. Jika Anda memilih Unduh dalam tabel riwayat untuk ekspor dan Anda tidak memberikan kata sandi, Amazon Lex V2 mengunduh file zip yang tidak dilindungi.

Format JSON untuk mengimpor dan mengekspor bot di Lex V2

Anda mengimpor dan mengekspor bot, lokal bot, atau kosakata khusus dari Amazon Lex V2 menggunakan file.zip yang berisi struktur JSON yang menjelaskan bagian-bagian sumber daya. Saat Anda mengekspor sumber daya, Amazon Lex V2 membuat file.zip dan membuatnya tersedia untuk Anda menggunakan URL Amazon S3 yang telah ditandatangani sebelumnya. Saat Anda mengimpor sumber daya, Anda harus membuat file.zip yang berisi struktur JSON dan mengunggahnya ke URL S3 yang telah ditandatangani sebelumnya.

Amazon Lex membuat struktur direktori berikut dalam file.zip saat Anda mengekspor bot. Saat Anda mengekspor lokal bot, hanya struktur di bawah lokal yang diekspor. Saat Anda mengekspor kosakata khusus, hanya struktur di bawah kosakata khusus yang diekspor.

```
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 file manifes

File manifes berisi metadata untuk file ekspor.

```
{
    "metadata": {
        "schemaVersion": "1.0",
        "fileFormat": "LexJson",
        "resourceType": "Bot | BotLocale | CustomVocabulary"
    }
}
```

Struktur file bot

File bot berisi informasi konfigurasi untuk bot.

```
{
    "name": "BotName",
    "identifier": "identifier",
    "version": "number",
    "description": "description",
    "dataPrivacy": {
        "childDirected": true | false
    },
    "idleSessionTTLInSeconds": seconds
}
```

Struktur file lokal bot

File bot lokal berisi deskripsi lokal atau bahasa bot. Saat Anda mengekspor bot, mungkin ada lebih dari satu file lokal bot di file.zip. Saat Anda mengekspor lokal bot, hanya ada satu lokal di file zip.

```
{
    "name": "locale name",
    "identifier": "locale ID",
    "version": "number",
    "description": "description",
    "voiceSettings": {
        "voiceId": "voice",
        "engine": "standard | neural
    },
    "nluConfidenceThreshold": number
}
```

Struktur file maksud

File maksud berisi informasi konfigurasi untuk maksud. Ada satu file maksud dalam file.zip untuk setiap maksud di lokal tertentu.

Berikut ini adalah contoh struktur JSON untuk BookCar maksud dalam bot sampel BookTrip . Untuk contoh lengkap struktur JSON untuk maksud, lihat operasinya. CreateIntent

```
{
    "name": "BookCar",
    "identifier": "891RWHHICO",
    "description": "Intent to book a car.",
    "parentIntentSignature": null,
    "sampleUtterances": [
        {
            "utterance": "Book a car"
        },
        {
            "utterance": "Reserve a car"
        },
        {
            "utterance": "Make a car reservation"
        }
    ],
    "intentConfirmationSetting": {
```

```
        "confirmationPrompt": {
            "messageGroupList": [
                {
                    "message": {
                        "plainTextMessage": {
                            "value": "OK, I have you down for a {CarType} hire in
{PickUpCity} from {PickUpDate} to {ReturnDate}.  Should I book the reservation?"
                        },
                        "ssmlMessage": null,
                        "customPayload": null,
                        "imageResponseCard": null
                    },
                    "variations": null
                }
            ],
            "maxRetries": 2
        },
        "declinationResponse": {
            "messageGroupList": [
                {
                    "message": {
                        "plainTextMessage": {
                            "value": "OK, I have cancelled your reservation in
progress."
                        },
                        "ssmlMessage": null,
                        "customPayload": null,
                        "imageResponseCard": null
                    },
                    "variations": null
                }
            ]
        }
    },
    "intentClosingSetting": null,
    "inputContexts": null,
    "outputContexts": null,
    "kendraConfiguration": null,
    "dialogCodeHook": null,
    "fulfillmentCodeHook": null,
    "slotPriorities": [
        {
            "slotName": "DriverAge",
            "priority": 4
```

```
        },
        {
            "slotName": "PickUpDate",
            "priority": 2
        },
        {
            "slotName": "ReturnDate",
            "priority": 3
        },
        {
            "slotName": "PickUpCity",
            "priority": 1
        },
        {
            "slotName": "CarType",
            "priority": 5
        }
    ]
}
```

Struktur file slot

File slot berisi informasi konfigurasi untuk slot dalam maksud. Ada satu file slot di file.zip untuk setiap slot yang ditentukan untuk maksud di lokal tertentu.

Contoh berikut adalah struktur JSON dari slot yang memungkinkan pelanggan untuk memilih jenis mobil yang ingin mereka sewa sesuai BookCar maksud di bot BookTrip contoh. Untuk contoh lengkap struktur JSON untuk slot, lihat CreateSlotoperasinya.

```
{
    "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,
    }
}
```

Contoh berikut menunjukkan struktur JSON dari slot komposit.

```
{
  "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)"
  }
}
```

Struktur file jenis slot

File jenis slot berisi informasi konfigurasi untuk jenis slot khusus yang digunakan dalam bahasa atau lokal. Ada satu file jenis slot di file.zip untuk setiap jenis slot khusus di lokal tertentu.

Berikut ini adalah struktur JSON untuk jenis slot yang mencantumkan jenis mobil yang tersedia di bot BookTrip contoh. Untuk contoh lengkap struktur JSON untuk jenis slot, lihat CreateSlotTypeoperasinya.

```
{
    "name": "CarTypeValues",
    "identifier": "T1YUHGD9ZR",
    "description": "Enumeration representing possible types of cars available for
 hire",
    "slotTypeValues": [{
        "synonyms": null,
        "sampleValue": {
            "value": "economy"
        }
    }, {
        "synonyms": null,
        "sampleValue": {
            "value": "standard"
        }
    }, {
        "synonyms": null,
        "sampleValue": {
            "value": "midsize"
        }
    }, {
        "synonyms": null,
        "sampleValue": {
            "value": "full size"
        }
    }, {
        "synonyms": null,
        "sampleValue": {
            "value": "luxury"
        }
    }, {
        "synonyms": null,
        "sampleValue": {
            "value": "minivan"
        }
    }],
    "parentSlotTypeSignature": null,
    "valueSelectionSetting": {
        "resolutionStrategy": "TOP_RESOLUTION",
```

```
        "advancedRecognitionSetting": {
            "audioRecognitionStrategy": "UseSlotValuesAsCustomVocabulary"
        },
        "regexFilter": null
    }
}
```

Contoh berikut menunjukkan struktur JSON untuk jenis slot komposit.

```
{
  "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
      }
    ]
  }
}
```

Berikut ini adalah jenis slot yang menggunakan tata bahasa khusus untuk memahami ucapan pelanggan. Untuk informasi selengkapnya, lihat Jenis slot tata bahasa.

```
{
  "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 file kosakata khusus

File kosakata khusus berisi entri dalam kosakata khusus untuk satu bahasa atau lokal. Ada satu file kosakata khusus di file.zip untuk setiap lokal yang memiliki kosakata khusus.

Berikut ini adalah file kosakata khusus untuk bot yang menerima pesanan restoran. Ada satu file per lokal di 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"
        }
    ]
}
```

Menandai sumber daya di Lex V2

Untuk membantu mengelola bot Amazon Lex V2 dan alias bot, Anda dapat menetapkan metadata ke setiap sumber daya sebagai tag. Tag adalah sebuah label yang Anda tetapkan ke sebuah sumber AWS. Setiap tag terdiri dari kunci dan nilai.

Tag memungkinkan Anda untuk mengkategorikan sumber daya AWS Anda dengan berbagai cara, misalnya, berdasarkan tujuan, pemilik, atau aplikasi. Tanda membantu Anda untuk:

- Mengidentifikasi dan mengelola sumber daya AWS Anda. Banyak sumber daya AWS mendukung penandaan, sehingga Anda dapat menetapkan tag yang sama ke sumber daya di layanan yang berbeda untuk menunjukkan bahwa sumber dayanya sama. Misalnya, Anda dapat menandai bot dan fungsi Lambda yang digunakannya dengan tag yang sama.
- Alokasikan biaya. Anda mengaktifkan tag di AWS Manajemen Penagihan dan Biaya dasbor. AWS menggunakan tag untuk mengategorikan biaya Anda dan mengirimkan laporan alokasi biaya bulanan kepada Anda. Untuk Amazon Lex V2, Anda dapat mengalokasikan biaya untuk setiap alias menggunakan tag khusus untuk alias. Untuk informasi selengkapnya, lihat Menggunakan tag alokasi biaya di Panduan AWS Manajemen Penagihan dan Biaya Pengguna.
- Mengendalikan akses ke sumber daya Anda. Anda dapat menggunakan tag dengan Amazon Lex V2 untuk membuat kebijakan untuk mengontrol akses ke sumber daya Amazon Lex V2 Kebijakan ini dapat dilampirkan ke peran IAM atau pengguna untuk mengaktifkan kontrol akses berbasis tag.

Anda dapat bekerja dengan tag menggunakan AWS Management Console, API Amazon Lex V2AWS Command Line Interface, atau Amazon Lex V2.

Menandai sumber daya Anda dengan konsol atau API

Jika Anda menggunakan konsol Amazon Lex V2, Anda dapat menandai sumber daya saat membuatnya, atau Anda dapat menambahkan tag nanti. Anda juga dapat menggunakan konsol untuk memperbarui atau menghapus tag yang ada.

Jika Anda menggunakan AWS CLI atau Amazon Lex V2 API, Anda menggunakan operasi berikut untuk mengelola tag untuk sumber daya Anda:

- CreateBotdan CreateBotAlias— terapkan tag saat Anda membuat bot atau alias bot.
- ListTagsForResource— lihat tag yang terkait dengan sumber daya.

- TagResource— menambahkan dan memodifikasi tag pada sumber daya yang ada.
- UntagResource— hapus tag dari sumber daya.

Sumber daya berikut di Amazon Lex V2 mendukung penandaan:

- Bot — gunakan Amazon Resource Name (ARN) seperti berikut:
  - `arn:aws:lex:${Region}:${account}:bot/${bot-id}`
- Alias bot — gunakan ARN seperti berikut ini:
  - `arn:aws:lex:${Region}:${account}:bot-alias/${bot-id}/${bot-alias-id}`

`bot-alias-id`Nilai `bot-id` dan dikapitalisasi string alfanumerik sepanjang 10 karakter.

Batasan tag saat menggunakan Lex V2

Pembatasan dasar berikut berlaku untuk tag di sumber daya Amazon Lex V2:

- Jumlah maksimum kunci — 50 menggunakan konsol, 200 menggunakan API
- Panjang kunci maksimum – 128 karakter
- Panjang nilai maksimum – 256 karakter
- Karakter yang valid untuk kunci dan nilai - a-z, A-Z, 0-9, spasi, dan karakter berikut: _.: /=+- dan @
- Kunci dan nilai peka huruf besar/kecil
- Jangan gunakan `aws:` sebagai awalan untuk kunci, itu dicadangkan untuk penggunaan AWS

Sumber daya penandaan (konsol)

Anda dapat menggunakan konsol untuk mengelola tag pada bot atau bot alias. Anda dapat menambahkan tag saat membuat sumber daya, atau Anda dapat menambahkan, memodifikasi, atau menghapus tag dari sumber daya yang ada.

Untuk menambahkan tag saat Anda membuat bot

1. Masuk ke AWS Management Console dan buka konsol Amazon Lex di https://console.aws.amazon.com/lex/.
2. Pilih Buat bot.

3. Di bagian Pengaturan lanjutan dari Konfigurasi pengaturan bot, pilih Tambahkan tag baru. Anda dapat menambahkan tag ke bot dan `TestBotAlias` alias.
4. Pilih Berikutnya untuk terus membuat bot Anda.

Untuk menambahkan tag saat Anda membuat alias bot

1. Masuk ke AWS Management Console dan buka konsol Amazon Lex di https://console.aws.amazon.com/lex/.
2. Pilih bot yang ingin Anda tambahkan alias bot.
3. Dari menu sebelah kiri, pilih Alias dan kemudian pilih Buat alias.
4. Dalam Info umum, pilih Tambahkan tag baru dari Tag.
5. Pilih Buat.

Untuk menambah, menghapus, atau memodifikasi tag pada bot yang ada

1. Masuk ke AWS Management Console dan buka konsol Amazon Lex di https://console.aws.amazon.com/lex/.
2. Pilih bot yang ingin Anda modifikasi.
3. Dari menu sebelah kiri, pilih Pengaturan, lalu pilih Edit.
4. Di Tag, buat perubahan Anda.
5. Pilih Simpan untuk menyimpan perubahan Anda ke bot.

Untuk menambah, menghapus, atau memodifikasi tag pada alias yang ada

1. Masuk ke AWS Management Console dan buka konsol Amazon Lex di https://console.aws.amazon.com/lex/.
2. Pilih bot yang ingin Anda modifikasi.
3. Dari menu kiri, pilih Alias dan kemudian dari daftar alias, pilih alias untuk dimodifikasi.
4. Dari detail Alias, di Tag, pilih Ubah tag.
5. Di Kelola tag, buat perubahan Anda.
6. Pilih Simpan untuk menyimpan perubahan Anda ke alias.

Keamanan di Amazon Lex V2

Keamanan cloud di AWS merupakan prioritas tertinggi. Sebagai pelanggan AWS, Anda mendapatkan keuntungan dari pusat data dan arsitektur jaringan yang dibangun untuk memenuhi persyaratan organisasi yang paling sensitif terhadap keamanan.

Keamanan adalah tanggung jawab bersama antara AWS dan Anda. Model tanggung jawab bersama menggambarkan ini sebagai keamanan dari cloud dan keamanan di dalam cloud:

- Keamanan cloud – AWS bertanggung jawab melindungi infrastruktur yang menjalankan layanan AWS dalam AWS Cloud. AWS juga menyediakan layanan yang dapat Anda gunakan dengan aman. Third-party auditor secara teratur menguji dan memverifikasi efektivitas keamanan kami sebagai bagian dari Program Kepatuhan AWS AWS Program Kepatuhan . Untuk mempelajari tentang program kepatuhan yang berlaku untuk Amazon Lex V2, lihat AWS Services in Scope by Compliance Program .
- Keamanan dalam cloud – Tanggung jawab Anda ditentukan oleh layanan AWS yang Anda gunakan. Anda juga bertanggung jawab atas faktor lain, yang mencakup sensitivitas data Anda, persyaratan perusahaan Anda, serta undang-undang dan peraturan yang berlaku.

Dokumentasi ini membantu Anda memahami cara menerapkan model tanggung jawab bersama saat menggunakan Amazon Lex V2. Topik berikut menunjukkan cara mengonfigurasi Amazon Lex V2 untuk memenuhi tujuan keamanan dan kepatuhan Anda. Anda juga mempelajari cara menggunakan layanan AWS lain yang membantu Anda memantau dan mengamankan sumber daya Amazon Lex V2 Anda.

Topik

- Perlindungan data di Amazon Lex V2
- Manajemen identitas dan akses untuk Amazon Lex V2
- Pencatatan dan pemantauan di Amazon Lex V2
- Validasi kepatuhan untuk Amazon Lex V2
- Ketahanan di Amazon Lex V2
- Keamanan infrastruktur di Amazon Lex V2
- Amazon Lex V2 dan titik akhir VPC antarmuka (AWS PrivateLink)

Perlindungan data di Amazon Lex V2

Amazon Lex V2 sesuai dengan model tanggung jawab bersama model tanggung jawab , yang mencakup peraturan dan pedoman untuk perlindungan data. AWS bertanggung jawab untuk melindungi infrastruktur global yang menjalankan semua layanan AWS. AWS mempertahankan kontrol atas data yang dihosting di infrastruktur ini, termasuk kontrol konfigurasi keamanan untuk menangani konten pelanggan dan data pribadi. Pelanggan AWS dan mitra APN, yang bertindak sebagai pengontrol data atau pemroses data, bertanggung jawab atas data pribadi apa pun yang mereka masukkan ke AWS Cloud.

Untuk tujuan perlindungan data, kami menyarankan agar Anda melindungi kredensial akun AWS dan mengatur akun pengguna individu dengan AWS Identity and Access Management (IAM), sehingga setiap pengguna hanya diberi izin yang diperlukan untuk memenuhi tugas pekerjaan mereka. Kami juga menyarankan supaya Anda mengamankan data dengan cara-cara berikut:

- Gunakan autentikasi multi-faktor (MFA) pada setiap akun.
- Gunakan SSL/TLS untuk berkomunikasi dengan sumber daya AWS.

- Siapkan API dan logging aktivitas pengguna dengan AWS CloudTrail.
- Gunakan solusi enkripsi AWS, bersama semua kendali keamanan default dalam layanan AWS.
- Gunakan layanan keamanan terkelola lanjutan seperti Amazon Macie, yang membantu menemukan dan mengamankan data pribadi yang disimpan di Amazon S3.

Sebaiknya jangan pernah memasukkan informasi identitas yang sensitif, seperti nomor rekening pelanggan Anda, ke dalam bidang isian bebas seperti bidang Nama. Ini termasuk saat Anda bekerja dengan Amazon Lex V2 atau layanan AWS lainnya menggunakan konsol, API, AWS CLI, atau AWS SDK. Data apa pun yang Anda masukkan ke Amazon Lex V2 atau layanan lain mungkin diambil untuk dimasukkan dalam log diagnostik. Saat Anda memberikan URL ke server eksternal, jangan sertakan informasi kredensial di URL untuk memvalidasi permintaan Anda ke server tersebut.

Untuk informasi selengkapnya tentang perlindungan data, lihat AWS Shared Responsibility Model dan postingan blog GDPR di AWS Security Blog.

Enkripsi saat diam

Amazon Lex V2 mengenkripsi ucapan pengguna dan informasi lain yang disimpannya.

Topik

- Sampel utterance
- Atribut sesi
- Atribut permintaan

Sampel utterance

Saat Anda mengembangkan bot, Anda dapat memberikan contoh ucapan untuk setiap maksud dan slot. Anda juga dapat memberikan nilai khusus dan sinonim untuk slot. Informasi ini dienkripsi saat istirahat, dan hanya digunakan untuk membangun bot dan menciptakan pengalaman pelanggan.

Atribut sesi

Atribut sesi berisi informasi khusus aplikasi yang diteruskan antara Amazon Lex V2 dan aplikasi klien. Amazon Lex V2 meneruskan atribut sesi ke semua AWS Lambda fungsi yang dikonfigurasi untuk bot. Jika fungsi Lambda menambahkan atau memperbarui atribut sesi, Amazon Lex V2 meneruskan informasi baru kembali ke aplikasi klien.

Atribut sesi bertahan di penyimpanan terenkripsi selama sesi berlangsung. Anda dapat mengonfigurasi sesi agar tetap aktif selama minimal 1 menit dan hingga 24 jam setelah ucapan pengguna terakhir. Durasi sesi default adalah 5 menit.

Atribut permintaan

Atribut permintaan berisi informasi khusus permintaan dan hanya berlaku untuk permintaan saat ini. Aplikasi klien menggunakan atribut permintaan untuk mengirim informasi ke Amazon Lex V2 saat runtime.

Anda menggunakan atribut permintaan untuk meneruskan informasi yang tidak perlu bertahan selama seluruh sesi. Karena atribut permintaan tidak bertahan di seluruh permintaan, atribut tersebut tidak disimpan.

Enkripsi saat bergerak

Amazon Lex V2 menggunakan protokol HTTPS untuk berkomunikasi dengan aplikasi klien Anda. Ini menggunakan tanda tangan HTTPS dan AWS untuk berkomunikasi dengan layanan lain, seperti Amazon Polly AWS Lambda dan atas nama aplikasi Anda.

Manajemen identitas dan akses untuk Amazon Lex V2

AWS Identity and Access Management (IAM) adalah Layanan AWS perangkat yang membantu administrator mengontrol akses ke sumber daya AWS dengan aman. Administrator IAM mengontrol siapa yang dapat diautentikasi (masuk) dan diotorisasi (memiliki izin) untuk menggunakan sumber daya Amazon Lex V2. IAM adalah Layanan AWS yang dapat Anda gunakan tanpa biaya tambahan.

Topik

- Audiens
- Mengautentikasi dengan identitas
- Mengelola akses menggunakan kebijakan
- Bagaimana Amazon Lex V2 bekerja dengan IAM
- Identity-based contoh kebijakan untuk Amazon Lex V2
- Resource-based contoh kebijakan untuk Amazon Lex V2
- Kebijakan terkelola AWS untuk Amazon Lex V2
- Menggunakan peran terkait layanan untuk Amazon Lex V2
- Menggunakan peran layanan untuk Amazon Lex V2
- Migrasi ke peran layanan untuk Amazon Lex V2
- Memecahkan masalah identitas dan akses Amazon Lex V2

Audiens

Cara Anda menggunakan AWS Identity and Access Management (IAM) berbeda berdasarkan peran Anda:

- Pengguna layanan - minta izin dari administrator Anda jika Anda tidak dapat mengakses fitur (lihat Memecahkan masalah identitas dan akses Amazon Lex V2)
- Administrator layanan - tentukan akses pengguna dan mengirimkan permintaan izin (lihat Bagaimana Amazon Lex V2 bekerja dengan IAM)
- Administrator IAM - tulis kebijakan untuk mengelola akses (lihat Identity-based contoh kebijakan untuk Amazon Lex V2)

Mengautentikasi dengan identitas

Autentikasi adalah cara Anda masuk ke AWS menggunakan kredensial identitas Anda. Anda harus diautentikasi sebagai Pengguna root akun AWS, pengguna IAM, atau dengan mengasumsikan peran IAM.

Anda dapat masuk sebagai identitas federasi menggunakan kredensil dari sumber identitas seperti AWS IAM Identity Center (Pusat Identitas IAM), autentikasi masuk tunggal, atau kredensional. Google/Facebook Untuk informasi selengkapnya tentang cara masuk, lihat Cara masuk ke Akun AWS Anda dalam Panduan Pengguna AWS Sign-In .

Untuk akses terprogram, AWS menyediakan SDK dan CLI untuk menandatangani permintaan secara kriptografis. Untuk informasi selengkapnya, lihat AWS Signature Version 4 untuk permintaan API di Panduan Pengguna IAM.

Akun AWS pengguna root

Saat Anda membuat Akun AWS, Anda mulai dengan satu identitas masuk yang disebut pengguna Akun AWS root yang memiliki akses lengkap ke semua Layanan AWS dan sumber daya. Kami sangat menyarankan agar Anda tidak menggunakan pengguna root untuk tugas sehari-hari. Untuk tugas yang memerlukan kredensial pengguna root, lihat Tugas yang memerlukan kredensial pengguna root dalam Panduan Pengguna IAM.

Identitas terfederasi

Sebagai praktik terbaik, mewajibkan pengguna manusia untuk menggunakan federasi dengan penyedia identitas untuk mengakses Layanan AWS menggunakan kredensyal sementara.

Identitas federasi adalah pengguna dari direktori perusahaan Anda, penyedia identitas web, atau Directory Service yang mengakses Layanan AWS menggunakan kredensil dari sumber identitas. Identitas terfederasi mengambil peran yang memberikan kredensial sementara.

Untuk manajemen akses terpusat, kami menyarankan AWS IAM Identity Center. Untuk informasi selengkapnya, lihat Apa itu Pusat Identitas IAM? dalam Panduan Pengguna AWS IAM Identity Center .

Pengguna dan grup IAM

Pengguna IAM adalah identitas dengan izin khusus untuk satu orang atau aplikasi. Sebaiknya gunakan kredensial sementara alih-alih pengguna IAM dengan kredensial jangka panjang. Untuk

informasi selengkapnya, lihat Mewajibkan pengguna manusia untuk menggunakan federasi dengan penyedia identitas untuk mengakses AWS menggunakan kredensil sementara di Panduan Pengguna IAM.

Grup IAM menentukan kumpulan pengguna IAM dan mempermudah pengelolaan izin untuk pengguna dalam jumlah besar. Untuk mempelajari selengkapnya, lihat Kasus penggunaan untuk pengguna IAM dalam Panduan Pengguna IAM.

Peran IAM

Peran IAM adalah identitas dengan izin khusus yang menyediakan kredensial sementara. Anda dapat mengambil peran dengan beralih dari pengguna ke peran IAM (konsol) atau dengan memanggil operasi AWS CLI atau AWS API. Untuk informasi selengkapnya, lihat Metode untuk mengambil peran dalam Panduan Pengguna IAM.

Peran IAM berguna untuk akses pengguna terfederasi, izin pengguna IAM sementara, akses lintas akun, akses lintas layanan, dan aplikasi yang berjalan di Amazon EC2. Untuk informasi selengkapnya, lihat Akses sumber daya lintas akun di IAM dalam Panduan Pengguna IAM.

Mengelola akses menggunakan kebijakan

Anda mengontrol akses di AWS dengan membuat kebijakan dan melampirkannya ke identitas atau sumber daya AWS. Kebijakan menentukan izin saat dikaitkan dengan identitas atau sumber daya. AWS mengevaluasi kebijakan ini saat prinsipal membuat permintaan. Sebagian besar kebijakan disimpan dalam dokumen AWS as JSON. Untuk informasi selengkapnya tentang dokumen kebijakan JSON, lihat Gambaran umum kebijakan JSON dalam Panduan Pengguna IAM.

Menggunakan kebijakan, administrator menentukan siapa yang memiliki akses ke apa dengan mendefinisikan principal mana yang dapat melakukan tindakan pada sumber daya apa, dan dalam kondisi apa.

Secara default, pengguna dan peran tidak memiliki izin. Administrator IAM membuat kebijakan IAM dan menambahkannya ke peran, yang kemudian dapat diambil oleh pengguna. Kebijakan IAM mendefinisikan izin terlepas dari metode yang Anda gunakan untuk melakukan operasinya.

Identity-based kebijakan

Identity-based kebijakan adalah dokumen kebijakan izin JSON yang Anda lampirkan ke identitas (pengguna, grup, atau peran). Kebijakan ini mengontrol tindakan apa yang bisa dilakukan oleh

identitas tersebut, terhadap sumber daya yang mana, dan dalam kondisi apa. Untuk mempelajari cara membuat kebijakan berbasis identitas, lihat Tentukan izin IAM kustom dengan kebijakan yang dikelola pelanggan dalam Panduan Pengguna IAM.

Identity-based kebijakan dapat berupa kebijakan inline (disematkan langsung ke dalam satu identitas) atau kebijakan terkelola (kebijakan mandiri yang dilampirkan pada beberapa identitas). Untuk mempelajari cara memilih antara kebijakan terkelola dan kebijakan inline, lihat Pilih antara kebijakan terkelola dan kebijakan inline dalam Panduan Pengguna IAM.

Resource-based kebijakan

Resource-based kebijakan adalah dokumen kebijakan JSON yang Anda lampirkan ke sumber daya. Contohnya termasuk kebijakan kepercayaan peran IAM dan kebijakan bucket Amazon S3. Dalam layanan yang mendukung kebijakan berbasis sumber daya, administrator layanan dapat menggunakannya untuk mengontrol akses ke sumber daya tertentu. Anda harus menentukan principal dalam kebijakan berbasis sumber daya.

Resource-based kebijakan adalah kebijakan inline yang terletak di layanan tersebut. Anda tidak dapat menggunakan kebijakan yang dikelola AWS dari IAM dalam kebijakan berbasis sumber daya.

Jenis-jenis kebijakan lain

AWS mendukung jenis kebijakan tambahan yang dapat menetapkan izin maksimum yang diberikan oleh jenis kebijakan yang lebih umum:

- Batasan izin – Menetapkan izin maksimum yang dapat diberikan oleh kebijakan berbasis identitas kepada entitas IAM. Untuk informasi selengkapnya, lihat Batasan izin untuk entitas IAM dalam Panduan Pengguna IAM.
- Kebijakan kontrol layanan (SCP) – Menentukan izin maksimum untuk organisasi atau unit organisasi di AWS Organizations. Untuk informasi selengkapnya, lihat Kebijakan kontrol layanan dalam Panduan Pengguna AWS Organizations .
- Kebijakan kontrol sumber daya (RCP) – Menetapkan izin maksimum yang tersedia untuk sumber daya di akun Anda. Untuk informasi selengkapnya, lihat Kebijakan kontrol sumber daya (RCP) dalam Panduan Pengguna AWS Organizations .
- Kebijakan sesi – Kebijakan lanjutan yang diteruskan sebagai parameter saat membuat sesi sementara untuk peran atau pengguna terfederasi. Untuk informasi selengkapnya, lihat Kebijakan sesi dalam Panduan Pengguna IAM.

Berbagai jenis kebijakan

Ketika beberapa jenis kebijakan berlaku untuk permintaan, izin yang dihasilkan lebih rumit untuk dipahami. Untuk mempelajari bagaimana AWS menentukan apakah mengizinkan permintaan jika beberapa jenis kebijakan dilibatkan, lihat Logika evaluasi kebijakan dalam Panduan Pengguna IAM.

Bagaimana Amazon Lex V2 bekerja dengan IAM

Sebelum Anda menggunakan IAM untuk mengelola akses ke Amazon Lex V2, pelajari fitur IAM apa yang tersedia untuk digunakan dengan Amazon Lex V2.

Fitur IAM yang dapat Anda gunakan dengan Amazon Lex V2

Fitur IAM	Dukungan Amazon Lex V2
Identity-based kebijakan	Ya
Resource-based kebijakan	Ya
Tindakan kebijakan	Ya
Sumber daya kebijakan	Ya
Kunci kondisi kebijakan	Tidak
ACL	Tidak
ABAC (tanda dalam kebijakan)	Ya
Kredensial sementara	Tidak
Izin principal	Ya
Peran layanan	Ya
Service-linked peran	Sebagian

Untuk mendapatkan tampilan tingkat tinggi tentang cara kerja Amazon Lex V2 dan layanan AWS lainnya dengan sebagian besar fitur IAM, lihat layanan AWS yang bekerja dengan IAM di Panduan Pengguna IAM.

Identity-based kebijakan untuk Amazon Lex V2

Mendukung kebijakan berbasis identitas: Ya

Identity-based kebijakan adalah dokumen kebijakan izin JSON yang dapat Anda lampirkan ke identitas, seperti pengguna IAM, grup pengguna, atau peran. Kebijakan ini mengontrol jenis tindakan yang dapat dilakukan oleh pengguna dan peran, di sumber daya mana, dan berdasarkan kondisi seperti apa. Untuk mempelajari cara membuat kebijakan berbasis identitas, lihat Tentukan izin IAM kustom dengan kebijakan terkelola pelanggan dalam Panduan Pengguna IAM.

Dengan kebijakan berbasis identitas IAM, Anda dapat menentukan secara spesifik apakah tindakan dan sumber daya diizinkan atau ditolak, serta kondisi yang menjadi dasar dikabulkan atau ditolaknya tindakan tersebut. Untuk mempelajari semua elemen yang dapat Anda gunakan dalam kebijakan JSON, lihat Referensi elemen kebijakan JSON IAM dalam Panduan Pengguna IAM.

Identity-based contoh kebijakan untuk Amazon Lex V2

Untuk melihat contoh kebijakan berbasis identitas Amazon Lex V2, lihat. Identity-based contoh kebijakan untuk Amazon Lex V2

Resource-based kebijakan dalam Amazon Lex V2

Mendukung kebijakan berbasis sumber daya: Ya

Resource-based kebijakan adalah dokumen kebijakan JSON yang Anda lampirkan ke sumber daya. Contoh kebijakan berbasis sumber daya adalah kebijakan kepercayaan peran IAM dan kebijakan bucket Amazon S3. Dalam layanan yang mendukung kebijakan berbasis sumber daya, administrator layanan dapat menggunakannya untuk mengontrol akses ke sumber daya tertentu. Untuk sumber daya tempat kebijakan dilampirkan, kebijakan menentukan tindakan apa yang dapat dilakukan oleh principal tertentu pada sumber daya tersebut dan dalam kondisi apa. Anda harus menentukan principal dalam kebijakan berbasis sumber daya. Prinsipal dapat mencakup pengguna, peran, pengguna federasi, atau layanan AWS.

Anda tidak dapat menggunakan kebijakan lintas akun atau lintas wilayah dengan Amazon Lex V2. Jika Anda membuat kebijakan untuk sumber daya dengan ARN lintas akun atau lintas wilayah, Amazon Lex V2 menampilkan kesalahan.

Layanan Amazon Lex V2 mendukung kebijakan berbasis sumber daya yang disebut kebijakan bot dan kebijakan alias bot, yang dilampirkan ke bot atau alias bot. Kebijakan ini menentukan prinsipal mana yang dapat melakukan tindakan pada bot atau bot alias.

Tindakan hanya dapat digunakan pada sumber daya tertentu. Misalnya, `UpdateBot` tindakan hanya dapat digunakan pada sumber daya bot, `UpdateBotAlias` tindakan hanya dapat digunakan pada sumber daya alias bot. Jika Anda menentukan tindakan dalam kebijakan yang tidak dapat digunakan pada sumber daya yang ditentukan dalam kebijakan, Amazon Lex V2 akan menampilkan kesalahan. Untuk daftar tindakan dan sumber daya yang dapat digunakan, lihat tabel berikut.

Tindakan	Mendukung kebijakan berbasis sumber daya	Sumber daya
BuildBotLocale	Didukung	BotId
CreateBot	Tidak	
CreateBotAlias	Tidak	
CreateBotChannel [hanya izin]	Didukung	BotId
CreateBotLocale	Didukung	BotId
CreateBotVersion	Didukung	BotId
CreateExport	Didukung	BotId
CreateIntent	Didukung	BotId
CreateResourcePolicy	Didukung	BotId, BotAliasId
CreateSlot	Didukung	BotId
CreateSlotType	Didukung	BotId
CreateUploadUrl	Tidak	
DeleteBot	Didukung	BotId, BotAliasId
DeleteBotAlias	Didukung	BotAliasId
DeleteBotChannel [hanya izin]	Didukung	BotId
DeleteBotLocale	Didukung	BotId

Tindakan	Mendukung kebijakan berbasis sumber daya	Sumber daya
DeleteBotVersion	Didukung	BotId
DeleteExport	Didukung	BotId
DeleteImport	Didukung	BotId
DeleteIntent	Didukung	BotId
DeleteResourcePolicy	Didukung	BotId, BotAliasId
DeleteSession	Didukung	BotAliasId
DeleteSlot	Didukung	BotId
DeleteSlotType	Didukung	BotId
DescribeBot	Didukung	BotId
DescribeBotAlias	Didukung	BotAliasId
DescribeBotChannel [hanya izin]	Didukung	BotId
DescribeBotLocale	Didukung	BotId
DescribeBotVersion	Didukung	BotId
DescribeExport	Didukung	BotId
DescribeImport	Didukung	BotId
DescribeIntent	Didukung	BotId
DescribeResourcePolicy	Didukung	BotId, BotAliasId
DescribeSlot	Didukung	BotId
DescribeSlotType	Didukung	BotId

Tindakan	Mendukung kebijakan berbasis sumber daya	Sumber daya
GetSession	Didukung	BotAliasId
ListBotAliases	Didukung	BotId
ListBotChannels [hanya izin]	Didukung	BotId
ListBotLocales	Didukung	BotId
ListBots	Tidak	
ListBotVersions	Didukung	BotId
ListBuiltInIntents	Tidak	
ListBuiltIntSlotTypes	Tidak	
ListExports	Tidak	
ListImports	Tidak	
ListIntents	Didukung	BotId
ListSlots	Didukung	BotId
ListSlotTypes	Didukung	BotId
PutSession	Didukung	BotAliasId
RecognizeText	Didukung	BotAliasId
RecognizeUtterance	Didukung	BotAliasId
StartConversation	Didukung	BotAliasId
StartImport	Didukung	BotId, BotAliasId
TagResource	Tidak	
UpdateBot	Didukung	BotId

Tindakan	Mendukung kebijakan berbasis sumber daya	Sumber daya
UpdateBotAlias	Didukung	BotAliasId
UpdateBotLocale	Didukung	BotId
UpdateBotVersion	Didukung	BotId
UpdateExport	Didukung	BotId
UpdateIntent	Didukung	BotId
UpdateResourcePolicy	Didukung	BotId, BotAliasId
UpdateSlot	Didukung	BotId
UpdateSlotType	Didukung	BotId
UntagResource	Tidak	

Untuk mempelajari cara melampirkan kebijakan berbasis sumber daya ke bot atau alias bot, lihat. Resource-based contoh kebijakan untuk Amazon Lex V2

Resource-based contoh kebijakan dalam Amazon Lex V2

Untuk melihat contoh kebijakan berbasis sumber daya Amazon Lex V2, lihat. Resource-based contoh kebijakan untuk Amazon Lex V2

Tindakan kebijakan untuk Amazon Lex V2

Mendukung tindakan kebijakan: Ya

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke hal apa. Yaitu, di mana utama dapat melakukan tindakan pada sumber daya, dan dalam kondisi apa.

Elemen `Action` dari kebijakan JSON menjelaskan tindakan yang dapat Anda gunakan untuk mengizinkan atau menolak akses dalam sebuah kebijakan. Sertakan tindakan dalam kebijakan untuk memberikan izin untuk melakukan operasi terkait.

Untuk melihat daftar tindakan Amazon Lex V2, lihat Tindakan yang ditentukan oleh Amazon Lex V2 di Referensi Otorisasi Layanan.

Tindakan kebijakan di Amazon Lex V2 menggunakan awalan berikut sebelum tindakan:

```
lex
```

Untuk menetapkan secara spesifik beberapa tindakan dalam satu pernyataan, pisahkan tindakan tersebut dengan koma.

```
"Action": [
      "lex:action1",
      "lex:action2"
         ]
```

Untuk melihat contoh kebijakan berbasis identitas Amazon Lex V2, lihat. Identity-based contoh kebijakan untuk Amazon Lex V2

Sumber daya kebijakan untuk Amazon Lex V2

Mendukung sumber daya kebijakan: Ya

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke hal apa. Yaitu, di mana utama dapat melakukan tindakan pada sumber daya, dan dalam kondisi apa.

Elemen kebijakan JSON `Resource` menentukan objek yang menjadi target penerapan tindakan. Praktik terbaiknya, tentukan sumber daya menggunakan Amazon Resource Name (ARN). Untuk tindakan yang tidak mendukung izin di tingkat sumber daya, gunakan wildcard (*) untuk menunjukkan bahwa pernyataan tersebut berlaku untuk semua sumber daya.

```
"Resource": "*"
```

Untuk melihat daftar jenis sumber daya Amazon Lex V2 dan ARNnya, lihat Sumber daya yang ditentukan oleh Amazon Lex V2 di Referensi Otorisasi Layanan. Untuk mempelajari tindakan mana yang dapat Anda tentukan ARN dari setiap sumber daya, lihat Tindakan yang ditentukan oleh Amazon Lex V2.

Untuk melihat contoh kebijakan berbasis identitas Amazon Lex V2, lihat. Identity-based contoh kebijakan untuk Amazon Lex V2

Kunci kondisi kebijakan untuk Amazon Lex V2

Mendukung kunci kondisi kebijakan khusus layanan: Tidak

Administrator dapat menggunakan kebijakan AWS JSON untuk menentukan siapa yang memiliki akses ke hal apa. Yaitu, prinsipal mana yang dapat melakukan tindakan di sumber daya apa, dan dalam kondisi apa.

Elemen `Condition` menentukan ketika pernyataan dieksekusi berdasarkan kriteria yang ditetapkan. Anda dapat membuat ekspresi bersyarat yang menggunakan operator kondisi, misalnya sama dengan atau kurang dari, untuk mencocokkan kondisi dalam kebijakan dengan nilai-nilai yang diminta. Untuk melihat semua kunci ketentuan global AWS, lihat Kunci konteks ketentuan global AWS dalam Panduan Pengguna IAM.

Untuk melihat daftar kunci kondisi Amazon Lex V2, lihat Kunci kondisi untuk Amazon Lex V2 di Referensi Otorisasi Layanan. Untuk mempelajari tindakan dan sumber daya yang dapat Anda gunakan kunci kondisi, lihat Tindakan yang ditentukan oleh Amazon Lex V2.

Untuk melihat contoh kebijakan berbasis identitas Amazon Lex V2, lihat. Identity-based contoh kebijakan untuk Amazon Lex V2

Daftar kontrol akses (ACL) di Amazon Lex V2

Mendukung ACL: Tidak

Daftar kontrol akses (ACL) mengendalikan principal mana (anggota akun, pengguna, atau peran) yang memiliki izin untuk mengakses sumber daya. ACL serupa dengan kebijakan berbasis sumber daya, meskipun kebijakan tersebut tidak menggunakan format dokumen kebijakan JSON.

Attribute-based kontrol akses (ABAC) dengan Amazon Lex V2

Mendukung ABAC (tag dalam kebijakan): Ya

Attribute-based Access Control (ABAC) adalah strategi otorisasi yang mendefinisikan izin berdasarkan atribut yang disebut tag. Anda dapat melampirkan tag ke entitas IAM dan sumber daya

AWS, lalu merancang kebijakan ABAC untuk mengizinkan operasi saat tag prinsipal cocok dengan tag pada sumber daya.

Untuk mengendalikan akses berdasarkan tanda, berikan informasi tentang tanda di elemen kondisi dari kebijakan menggunakan kunci kondisi `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, atau `aws:TagKeys`.

Jika sebuah layanan mendukung ketiga kunci kondisi untuk setiap jenis sumber daya, nilainya adalah Ya untuk layanan tersebut. Jika suatu layanan mendukung ketiga kunci kondisi untuk hanya beberapa jenis sumber daya, nilainya adalah Parsial.

Untuk informasi selengkapnya tentang ABAC, lihat Tentukan izin dengan otorisasi ABAC dalam Panduan Pengguna IAM. Untuk melihat tutorial yang menguraikan langkah-langkah pengaturan ABAC, lihat Menggunakan kontrol akses berbasis atribut (ABAC) dalam Panduan Pengguna IAM.

Menggunakan kredensyal Sementara dengan Amazon Lex V2

Mendukung kredensi sementara: Tidak

Kredensyal sementara menyediakan akses jangka pendek ke sumber daya AWS dan dibuat secara otomatis saat Anda menggunakan federasi atau beralih peran. AWS menyarankan agar Anda menghasilkan kredensyal sementara secara dinamis alih-alih menggunakan kunci akses jangka panjang. Untuk informasi selengkapnya, lihat Kredensial keamanan sementara di IAM dan Layanan AWS yang berfungsi dengan IAM dalam Panduan Pengguna IAM.

Cross-service izin utama untuk Amazon Lex V2

Mendukung sesi akses maju (FAS): Ya

Sesi akses teruskan (FAS) menggunakan izin dari pemanggilan utama Layanan AWS, dikombinasikan dengan permintaan Layanan AWS untuk membuat permintaan ke layanan hilir. Untuk detail kebijakan ketika mengajukan permintaan FAS, lihat Sesi akses terusan.

Peran layanan untuk Amazon Lex V2

Mendukung peran layanan: Ya

Peran layanan adalah peran IAM yang diambil oleh sebuah layanan untuk melakukan tindakan atas nama Anda. Administrator IAM dapat membuat, mengubah, dan menghapus peran layanan dari dalam IAM. Untuk informasi selengkapnya, lihat Buat sebuah peran untuk mendelegasikan izin ke Layanan AWS dalam Panduan pengguna IAM.

> **Warning**
>
> Mengubah izin untuk peran layanan dapat merusak fungsionalitas Amazon Lex V2. Edit peran layanan hanya jika Amazon Lex V2 memberikan panduan untuk melakukannya.

Service-linked peran untuk Amazon Lex V2

Mendukung peran terkait layanan: Sebagian

Peran terkait layanan adalah jenis peran layanan yang ditautkan ke. Layanan AWS Layanan dapat mengambil peran untuk melakukan tindakan atas nama Anda. Service-linked peran muncul di Anda Akun AWS dan dimiliki oleh layanan. Administrator IAM dapat melihat, tetapi tidak dapat mengedit izin untuk peran terkait layanan.

Untuk detail tentang membuat atau mengelola peran terkait layanan, lihat layanan AWS yang bekerja dengan IAM. Temukan layanan dalam tabel yang `Yes` menyertakan kolom Service-linked peran. Pilih tautan Ya untuk melihat dokumentasi peran terkait layanan untuk layanan tersebut.

Identity-based contoh kebijakan untuk Amazon Lex V2

Secara default, pengguna dan peran tidak memiliki izin untuk membuat atau memodifikasi sumber daya Amazon Lex V2. Untuk memberikan izin kepada pengguna untuk melakukan tindakan di sumber daya yang mereka perlukan, administrator IAM dapat membuat kebijakan IAM.

Untuk mempelajari cara membuat kebijakan berbasis identitas IAM dengan menggunakan contoh dokumen kebijakan JSON ini, lihat Membuat kebijakan IAM (konsol) di Panduan Pengguna IAM.

Untuk detail tentang tindakan dan jenis sumber daya yang ditentukan oleh Amazon Lex V2, termasuk format ARN untuk setiap jenis sumber daya, lihat Tindakan, sumber daya, dan kunci kondisi untuk Amazon Lex V2 di Referensi Otorisasi Layanan.

Topik

- Praktik terbaik kebijakan
- Menggunakan konsol Amazon Lex V2
- Izinkan pengguna untuk menambahkan fungsi ke bot
- Memungkinkan pengguna untuk menambahkan saluran ke bot
- Izinkan pengguna membuat dan memperbarui bot

- Memungkinkan pengguna untuk menggunakan Desainer Chatbot Otomatis
- Memungkinkan pengguna untuk menggunakan AWS KMS kunci untuk mengenkripsi dan mendekripsi file
- Izinkan pengguna untuk menghapus bot
- Memungkinkan pengguna untuk melakukan percakapan dengan bot
- Izinkan pengguna tertentu untuk mengelola kebijakan berbasis sumber daya
- Izinkan pengguna untuk mengekspor bot dan lokal bot
- Izinkan pengguna untuk mengekspor kosakata khusus
- Izinkan pengguna untuk mengimpor bot dan lokal bot
- Izinkan pengguna untuk mengimpor kosakata khusus
- Izinkan pengguna untuk memigrasikan bot dari Amazon Lex ke Amazon Lex V2
- Mengizinkan pengguna melihat izin mereka sendiri
- Izinkan pengguna menggambar alur percakapan dengan pembuat percakapan visual di Amazon Lex V2
- Izinkan pengguna membuat dan melihat replika bot, tetapi tidak menghapusnya

Praktik terbaik kebijakan

Identity-based kebijakan menentukan apakah seseorang dapat membuat, mengakses, atau menghapus sumber daya Amazon Lex V2 di akun Anda. Tindakan ini membuat Akun AWS Anda dikenai biaya. Ketika Anda membuat atau mengedit kebijakan berbasis identitas, ikuti panduan dan rekomendasi ini:

- Mulailah dengan kebijakan terkelola AWS dan lanjutkan ke izin hak istimewa paling sedikit — Untuk mulai memberikan izin kepada pengguna dan beban kerja Anda, gunakan kebijakan terkelola AWS yang memberikan izin untuk banyak kasus penggunaan umum. Mereka tersedia di Anda Akun AWS. Kami menyarankan Anda mengurangi izin lebih lanjut dengan menentukan kebijakan yang dikelola pelanggan AWS yang khusus untuk kasus penggunaan Anda. Untuk informasi selengkapnya, lihat kebijakan terkelola AWS atau kebijakan terkelola AWS untuk fungsi pekerjaan di Panduan Pengguna IAM.
- Menerapkan izin dengan hak akses paling rendah – Ketika Anda menetapkan izin dengan kebijakan IAM, hanya berikan izin yang diperlukan untuk melakukan tugas. Anda melakukannya dengan mendefinisikan tindakan yang dapat diambil pada sumber daya tertentu dalam kondisi

tertentu, yang juga dikenal sebagai izin dengan hak akses paling rendah. Untuk informasi selengkapnya tentang cara menggunakan IAM untuk mengajukan izin, lihat Kebijakan dan izin dalam IAM dalam Panduan Pengguna IAM.

- Gunakan kondisi dalam kebijakan IAM untuk membatasi akses lebih lanjut – Anda dapat menambahkan suatu kondisi ke kebijakan Anda untuk membatasi akses ke tindakan dan sumber daya. Sebagai contoh, Anda dapat menulis kondisi kebijakan untuk menentukan bahwa semua permintaan harus dikirim menggunakan SSL. Anda juga dapat menggunakan ketentuan untuk memberikan akses ke tindakan layanan jika digunakan melalui yang spesifik Layanan AWS, seperti CloudFormation. Untuk informasi selengkapnya, lihat Elemen kebijakan JSON IAM: Kondisi dalam Panduan Pengguna IAM.
- Gunakan IAM Access Analyzer untuk memvalidasi kebijakan IAM Anda untuk memastikan izin yang aman dan fungsional – IAM Access Analyzer memvalidasi kebijakan baru dan yang sudah ada sehingga kebijakan tersebut mematuhi bahasa kebijakan IAM (JSON) dan praktik terbaik IAM. IAM Access Analyzer menyediakan lebih dari 100 pemeriksaan kebijakan dan rekomendasi yang dapat ditindaklanjuti untuk membantu Anda membuat kebijakan yang aman dan fungsional. Untuk informasi selengkapnya, lihat Validasi kebijakan dengan IAM Access Analyzer dalam Panduan Pengguna IAM.
- Memerlukan otentikasi multi-faktor (MFA) - Jika Anda memiliki skenario yang mengharuskan pengguna IAM atau pengguna root di Anda, Akun AWS aktifkan MFA untuk keamanan tambahan. Untuk meminta MFA ketika operasi API dipanggil, tambahkan kondisi MFA pada kebijakan Anda. Untuk informasi selengkapnya, lihat Amankan akses API dengan MFA dalam Panduan Pengguna IAM.

Untuk informasi selengkapnya tentang praktik terbaik dalam IAM, lihat Praktik terbaik keamanan di IAM dalam Panduan Pengguna IAM.

Menggunakan konsol Amazon Lex V2

Untuk mengakses konsol Amazon Lex V2, Anda harus memiliki set izin minimum. Izin ini harus memungkinkan Anda untuk membuat daftar dan melihat detail tentang sumber daya Amazon Lex V2 di Anda Akun AWS. Jika Anda membuat kebijakan berbasis identitas yang lebih ketat daripada izin minimum yang diperlukan, konsol tidak akan berfungsi sebagaimana mestinya untuk entitas (pengguna atau peran) dengan kebijakan tersebut.

Anda tidak perlu mengizinkan konsol minimum untuk pengguna yang melakukan panggilan ke AWS CLI atau API AWS saja. Sebagai gantinya, izinkan akses hanya ke tindakan yang sesuai dengan operasi API yang coba mereka lakukan.

Untuk memastikan bahwa pengguna dan peran masih dapat menggunakan konsol Amazon Lex V2, pengguna harus memiliki akses Konsol. Untuk informasi selengkapnya tentang membuat pengguna dengan akses Konsol, lihat Membuat pengguna IAM di akun AWS Anda di Panduan Pengguna IAM.

Izinkan pengguna untuk menambahkan fungsi ke bot

Contoh ini menunjukkan kebijakan yang memungkinkan pengguna IAM menambahkan Amazon Comprehend, analisis sentimen, dan izin kueri Amazon Kendra ke bot Amazon Lex V2.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "Id1",
            "Effect": "Allow",
            "Action": "iam:PutRolePolicy",
            "Resource": "arn:aws:iam::*:role/aws-service-role/
lexv2.amazonaws.com/AWSServiceRoleForLexV2Bots*"
        },
        {
            "Sid": "Id2",
            "Effect": "Allow",
            "Action": "iam:GetRolePolicy",
            "Resource": "arn:aws:iam::*:role/aws-service-role/
lexv2.amazonaws.com/AWSServiceRoleForLexV2Bots*"
        }
    ]
}
```

Memungkinkan pengguna untuk menambahkan saluran ke bot

Contoh ini adalah kebijakan yang memungkinkan pengguna IAM untuk menambahkan saluran pesan ke bot. Pengguna harus memiliki kebijakan ini sebelum mereka dapat menerapkan bot di platform perpesanan.

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*"
        }
    ]
}
```

Izinkan pengguna membuat dan memperbarui bot

Contoh ini menunjukkan contoh kebijakan yang memungkinkan pengguna IAM untuk membuat dan memperbarui bot apa pun. Kebijakan ini mencakup izin untuk menyelesaikan tindakan ini di konsol atau menggunakan AWS CLI atau AWS API.

Memungkinkan pengguna untuk menggunakan Desainer Chatbot Otomatis

Contoh ini menunjukkan contoh kebijakan yang memungkinkan pengguna IAM menjalankan Desainer Chatbot Otomatis.

Memungkinkan pengguna untuk menggunakan AWS KMS kunci untuk mengenkripsi dan mendekripsi file

Contoh ini menunjukkan contoh kebijakan yang memungkinkan pengguna IAM menggunakan kunci yang dikelola AWS KMS pelanggan untuk mengenkripsi dan mendekripsi data.

Izinkan pengguna untuk menghapus bot

Contoh ini menunjukkan contoh kebijakan yang memungkinkan pengguna IAM untuk menghapus bot apa pun. Kebijakan ini mencakup izin untuk menyelesaikan tindakan ini di konsol atau menggunakan AWS CLI atau AWS 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:us-east-1:123412341234:bot/*",
                "arn:aws:lex:us-east-1:123412341234:bot-alias/*"
            ]
        }
    ]
}
```

Memungkinkan pengguna untuk melakukan percakapan dengan bot

Contoh ini menunjukkan contoh kebijakan yang memungkinkan pengguna IAM melakukan percakapan dengan bot apa pun. Kebijakan ini mencakup izin untuk menyelesaikan tindakan ini di konsol atau menggunakan AWS CLI atau AWS 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:us-east-1:123412341234:bot-alias/*"
        }
    ]
}
```

Izinkan pengguna tertentu untuk mengelola kebijakan berbasis sumber daya

Contoh berikut memberikan izin kepada pengguna tertentu untuk mengelola kebijakan berbasis sumber daya. Ini memungkinkan akses konsol dan API ke kebijakan yang terkait dengan bot dan alias bot.

Izinkan pengguna untuk mengekspor bot dan lokal bot

Kebijakan izin IAM berikut memungkinkan pengguna untuk membuat, memperbarui, dan mendapatkan ekspor untuk bot atau bot lokal.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "lex:CreateExport",
                "lex:UpdateExport",
                "lex:DescribeExport",
                "lex:DescribeBot",
                "lex:DescribeBotLocale",
                "lex:ListBotLocales",
                "lex:DescribeIntent",
                "lex:ListIntents",
                "lex:DescribeSlotType",
                "lex:ListSlotTypes",
                "lex:DescribeSlot",
                "lex:ListSlots",
                "lex:DescribeCustomVocabulary"
            ],
```

```
            "Effect": "Allow",
            "Resource": [
                "arn:aws:lex:us-east-1:123456789012:bot/*"
            ]
        }
    ]
}
```

Izinkan pengguna untuk mengekspor kosakata khusus

Kebijakan izin IAM berikut memungkinkan pengguna untuk mengekspor kosakata khusus dari lokal bot.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "lex:CreateExport",
                "lex:UpdateExport",
                "lex:DescribeExport",
                "lex:DescribeCustomVocabulary"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:lex:us-east-1:123456789012:bot/*"
            ]
        }
    ]
}
```

Izinkan pengguna untuk mengimpor bot dan lokal bot

Kebijakan izin IAM berikut memungkinkan pengguna untuk mengimpor bot atau bot lokal dan memeriksa status impor.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "lex:CreateUploadUrl",
                "lex:StartImport",
                "lex:DescribeImport",
                "lex:CreateBot",
                "lex:UpdateBot",
                "lex:DeleteBot",
                "lex:CreateBotLocale",
                "lex:UpdateBotLocale",
                "lex:DeleteBotLocale",
                "lex:CreateIntent",
                "lex:UpdateIntent",
                "lex:DeleteIntent",
                "lex:CreateSlotType",
                "lex:UpdateSlotType",
                "lex:DeleteSlotType",
                "lex:CreateSlot",
                "lex:UpdateSlot",
                "lex:DeleteSlot",
                "lex:CreateCustomVocabulary",
                "lex:UpdateCustomVocabulary",
                "lex:DeleteCustomVocabulary",
                "iam:PassRole"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:lex:us-east-1:123456789012:bot/*",
                "arn:aws:lex:us-east-1:123456789012:bot-alias/*"
            ]
        }
    ]
}
```

Izinkan pengguna untuk mengimpor kosakata khusus

Kebijakan izin IAM berikut memungkinkan pengguna untuk mengimpor kosakata khusus ke lokal bot.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "lex:CreateUploadUrl",
                "lex:StartImport",
                "lex:DescribeImport",
                "lex:CreateCustomVocabulary",
                "lex:UpdateCustomVocabulary",
                "lex:DeleteCustomVocabulary"
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:lex:us-east-1:123456789012:bot/*"
            ]
        }
    ]
}
```

Izinkan pengguna untuk memigrasikan bot dari Amazon Lex ke Amazon Lex V2

Kebijakan izin IAM berikut memungkinkan pengguna untuk mulai memigrasikan bot dari Amazon Lex ke Amazon Lex V2.

Mengizinkan pengguna melihat izin mereka sendiri

Contoh ini menunjukkan bagaimana Anda dapat membuat kebijakan yang memungkinkan pengguna IAM melihat kebijakan inline dan kebijakan terkelola yang terlampir pada identitas pengguna mereka. Kebijakan ini mencakup izin untuk menyelesaikan tindakan ini pada konsol atau secara terprogram menggunakan AWS CLI atau AWS API.

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

```
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Izinkan pengguna menggambar alur percakapan dengan pembuat percakapan visual di Amazon Lex V2

Kebijakan izin IAM berikut memungkinkan pengguna untuk menggambar alur percakapan dengan pembuat percakapan visual di Amazon Lex V2.

Izinkan pengguna membuat dan melihat replika bot, tetapi tidak menghapusnya

Anda dapat melampirkan izin berikut ke peran IAM untuk memungkinkannya hanya membuat dan melihat replika bot. Dengan menghilangkan`lex:DeleteBotReplica`, Anda mencegah peran menghapus replika bot. Untuk informasi selengkapnya, lihat Izin untuk mereplikasi bot dan mengelola replika bot di Lex V2.

Resource-based contoh kebijakan untuk Amazon Lex V2

Kebijakan berbasis sumber daya dilampirkan ke sumber daya, seperti bot atau alias bot. Dengan kebijakan berbasis sumber daya, Anda dapat menentukan siapa yang memiliki akses ke sumber daya dan tindakan yang dapat mereka lakukan di dalamnya. Misalnya, Anda dapat menambahkan kebijakan berbasis sumber daya yang memungkinkan pengguna memodifikasi bot tertentu, atau mengizinkan pengguna menggunakan operasi runtime pada alias bot tertentu.

Saat Anda menggunakan kebijakan berbasis sumber daya, Anda dapat mengizinkan layanan AWS lainnya mengakses sumber daya di akun Anda. Misalnya, Anda dapat mengizinkan Connect Customer mengakses bot Amazon Lex V2.

Untuk mempelajari cara membuat bot atau bot alias, lihatBekerja dengan bot Amazon Lex V2.

Topik

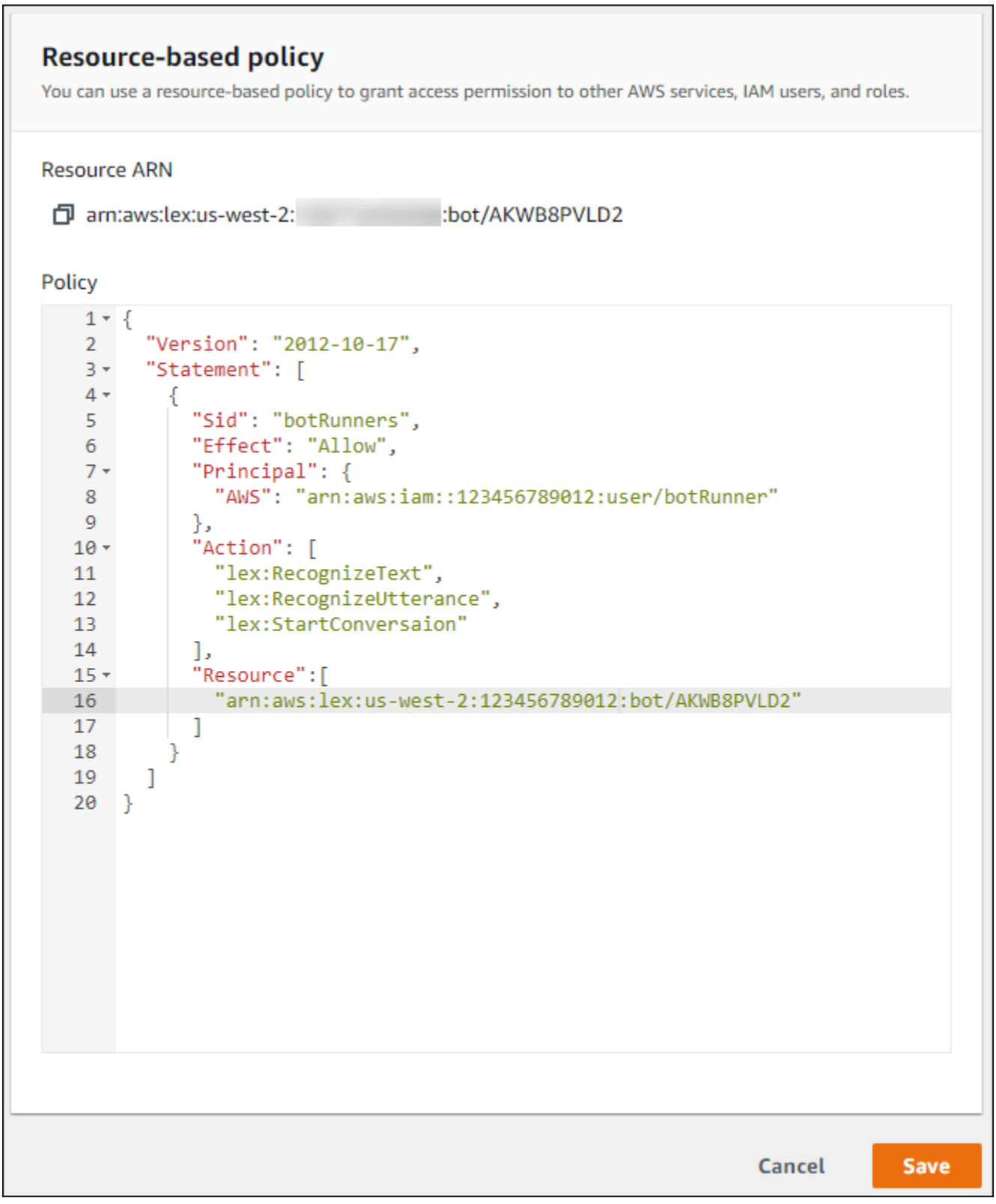
- Menggunakan konsol untuk menentukan kebijakan berbasis sumber daya
- Menggunakan API untuk menentukan kebijakan berbasis sumber daya
- Izinkan peran IAM untuk memperbarui bot dan daftar alias bot
- Memungkinkan pengguna untuk melakukan percakapan dengan bot
- Izinkan layanan AWS menggunakan bot Amazon Lex V2 tertentu

Menggunakan konsol untuk menentukan kebijakan berbasis sumber daya

Anda dapat menggunakan konsol Amazon Lex V2 untuk mengelola kebijakan berbasis sumber daya untuk bot dan alias bot Anda. Anda memasukkan struktur JSON kebijakan dan konsol mengaitkannya dengan sumber daya. Jika ada kebijakan yang telah dikaitkan dengan sumber daya, Anda dapat menggunakan konsol untuk melihat dan mengubah kebijakan.

Saat Anda menyimpan kebijakan dengan editor kebijakan, konsol akan memeriksa sintaks kebijakan tersebut. Jika kebijakan berisi kesalahan, seperti pengguna yang tidak ada atau tindakan yang tidak didukung oleh sumber daya, kebijakan akan menampilkan kesalahan dan tidak menyimpan kebijakan.

Berikut ini menunjukkan editor kebijakan berbasis sumber daya untuk bot di konsol. Editor kebijakan untuk alias bot serupa.


Resource-based policy
You can use a resource-based policy to grant access permission to other AWS services, IAM users, and roles.
Resource ARN
arn:aws:lex:us-west-2: :bot/AKWB8PVLD2
Policy
{
"Version": "2012-10-17",
"Statement": [
{
"Sid": "botRunners",
"Effect": "Allow",
"Principal": {
"AWS": "arn:aws:iam::123456789012:user/botRunner"
},
"Action": [
"lex:RecognizeText",
"lex:RecognizeUtterance",
"lex:StartConversaion"
],
"Resource":[
"arn:aws:lex:us-west-2:123456789012:bot/AKWB8PVLD2"
]
}
]
}
Cancel
Save

Untuk membuka editor kebijakan untuk bot

1. Masuk ke AWS Management Console dan buka konsol Amazon Lex di https://console.aws.amazon.com/lex/.
2. Dari daftar Bots, pilih bot yang kebijakannya ingin Anda edit.
3. Di bagian Resource-based kebijakan, pilih Edit.

Untuk membuka editor kebijakan untuk alias bot

1. Masuk ke AWS Management Console dan buka konsol Amazon Lex di https://console.aws.amazon.com/lex/.
2. Dari daftar Bots, pilih bot yang berisi alias yang ingin Anda edit.
3. Dari menu sebelah kiri, pilih Alias, lalu pilih alias yang akan diedit.
4. Di bagian Resource-based kebijakan, pilih Edit.

Menggunakan API untuk menentukan kebijakan berbasis sumber daya

Anda dapat menggunakan operasi API untuk mengelola kebijakan berbasis sumber daya untuk bot dan alias bot Anda. Ada operasi untuk membuat, memperbarui, dan menghapus kebijakan.

CreateResourcePolicy

Menambahkan kebijakan sumber daya baru dengan pernyataan kebijakan yang ditentukan ke bot atau alias bot.

CreateResourcePolicyStatement

Menambahkan pernyataan kebijakan sumber daya baru ke bot atau bot alias.

DeleteResourcePolicy

Menghapus kebijakan sumber daya dari bot atau bot alias.

DeleteResourcePolicyStatement

Menghapus pernyataan kebijakan sumber daya dari bot atau bot alias.

DescribeResourcePolicy

Mendapat kebijakan sumber daya dan revisi kebijakan.

UpdateResourcePolicy

Mengganti kebijakan sumber daya yang ada untuk bot atau bot alias dengan yang baru.

Contoh

Java

Contoh berikut menunjukkan cara menggunakan operasi kebijakan berbasis sumber daya untuk mengelola kebijakan berbasis sumber daya.

```
        /*
         * Create a new policy for the specified bot alias
         * that allows a role to invoke lex:UpdateBotAlias on it.
         * The created policy will have revision id 1.
         */

        CreateResourcePolicyRequest createPolicyRequest =
                CreateResourcePolicyRequest.builder()
                        .resourceArn("arn:aws:lex:Region:123456789012:bot-
alias/MYBOTALIAS/TSTALIASID")
                        .policy("{\"Version\": \"2012-10-17\",\"Statement
\": [{\"Sid\": \"BotAliasEditor\",\"Effect\": \"Allow\",\"Principal\":
 {\"AWS\": \"arn:aws:iam::123456789012:role/BotAliasEditor\"},\"Action\":
 [\"lex:UpdateBotAlias\"],\"Resource\":[\"arn:aws:lex:Region:123456789012:bot-
alias/MYBOTALIAS/TSTALIASID\"]]}")

        lexmodelsv2Client.createResourcePolicy(createPolicyRequest);

        /*
         * Overwrite the policy for the specified bot alias with a new policy.
         * Since no expectedRevisionId is provided, this request overwrites the
 current revision.
         * After this update, the revision id for the policy is 2.
         */
        UpdateResourcePolicyRequest updatePolicyRequest =
        UpdateResourcePolicyRequest.builder()
                        .resourceArn("arn:aws:lex:Region:123456789012:bot-
alias/MYBOTALIAS/TSTALIASID")
                        .policy("{\"Version\": \"2012-10-17\",\"Statement
\": [{\"Sid\": \"BotAliasEditor\",\"Effect\": \"Deny\",\"Principal\":
 {\"AWS\": \"arn:aws:iam::123456789012:role/BotAliasEditor\"},\"Action\":
```

```
 [\"lex:UpdateBotAlias\"],\"Resource\":[\"arn:aws:lex:Region:123456789012:bot-
alias/MYBOTALIAS/TSTALIASID\"]]}")

        lexmodelsv2Client.updateResourcePolicy(updatePolicyRequest);

        /*
         * Creates a statement in an existing policy for the specified bot alias
         * that allows a role to invoke lex:RecognizeText on it.
         * This request expects to update revision 2 of the policy. The request will
 fail
         * if the current revision of the policy is no longer revision 2.
         * After this request, the revision id for this policy will be 3.
         */

        CreateResourcePolicyStatementRequest createStatementRequest =
                CreateResourcePolicyStatementRequest.builder()
                        .resourceArn("arn:aws:lex:Region:123456789012:bot-
alias/MYBOTALIAS/TSTALIASID")
                        .effect("Allow")

 .principal(Principal.builder().arn("arn:aws:iam::123456789012:role/
BotRunner").build())
                        .action("lex:RecognizeText")
                        .statementId("BotRunnerStatement")
                        .expectedRevisionId(2)
                        .build();

        lexmodelsv2Client.createResourcePolicyStatement(createStatementRequest);

        /*
         * Deletes a statement from an existing policy for the specified bot alias
 by statementId.
         * Since no expectedRevisionId is supplied, the request will remove the
 statement from
         * the current revision of the policy for the bot alias.
         * After this request, the revision id for this policy will be 4.
         */
        DeleteResourcePolicyRequest deleteStatementRequest =
                DeleteResourcePolicyRequest.builder()
                        .resourceArn("arn:aws:lex:Region:123456789012:bot-
alias/MYBOTALIAS/TSTALIASID")
                        .statementId("BotRunnerStatement")
                        .build();
```

```
        lexmodelsv2Client.deleteResourcePolicy(deleteStatementRequest);

        /*
         * Describe the current policy for the specified bot alias
         * It always returns the current revision.
         */
        DescribeResourcePolicyRequest describePolicyRequest =
                DescribeResourcePolicyRequest.builder()
                        .resourceArn("arn:aws:lex:Region:123456789012:bot-
alias/MYBOTALIAS/TSTALIASID")
                        .build();

        lexmodelsv2Client.describeResourcePolicy(describePolicyRequest);


        /*
         * Delete the current policy for the specified bot alias
         * This request expects to delete revision 3 of the policy. Since the
 revision id for
         * this policy is already at 4, this request will fail.
         */
        DeleteResourcePolicyRequest deletePolicyRequest =
                DeleteResourcePolicyRequest.builder()
                        .resourceArn("arn:aws:lex:Region:123456789012:bot-
alias/MYBOTALIAS/TSTALIASID")
                        .expectedRevisionId(3);
                        .build();

        lexmodelsv2Client.deleteResourcePolicy(deletePolicyRequest);
```

Izinkan peran IAM untuk memperbarui bot dan daftar alias bot

Contoh berikut memberikan izin untuk peran IAM tertentu untuk memanggil operasi API pembuatan model Amazon Lex V2 untuk memodifikasi bot yang ada. Pengguna dapat membuat daftar alias untuk bot dan memperbarui bot, tetapi tidak dapat menghapus alias bot atau bot.

JSON

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

```
        {
            "Sid": "botBuilders",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:role/BotBuilder"
            },
            "Action": [
                "lex:ListBotAliases",
                "lex:UpdateBot"
            ],
            "Resource": [
                "arn:aws:lex:us-east-1:123456789012:bot/MYBOT"
            ]
        }
    ]
}
```

Memungkinkan pengguna untuk melakukan percakapan dengan bot

Contoh berikut memberikan izin bagi pengguna tertentu untuk memanggil operasi API runtime Amazon Lex V2 pada satu alias bot.

Pengguna secara khusus ditolak izin untuk memperbarui atau menghapus alias bot.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "botRunners",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:user/botRunner"
            },
            "Action": [
                "lex:RecognizeText",
                "lex:RecognizeUtterance",
                "lex:StartConversation",
                "lex:DeleteSession",
                "lex:GetSession",
                "lex:PutSession"
```

```
            ],
            "Resource": [
                "arn:aws:lex:us-east-1:123456789012:bot-alias/MYBOT/MYBOTALIAS"
            ]
        },
        {
            "Sid": "botRunners",
            "Effect": "Deny",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:user/botRunner"
            },
            "Action": [
                "lex:UpdateBotAlias",
                "lex:DeleteBotAlias"
            ],
            "Resource": [
                "arn:aws:lex:us-east-1:123456789012:bot-alias/MYBOT/MYBOTALIAS"
            ]
        }
    ]
}
```

Izinkan layanan AWS menggunakan bot Amazon Lex V2 tertentu

Contoh berikut memberikan izin untuk AWS Lambda dan Connect Customer untuk memanggil operasi API runtime Amazon Lex V2.

Blok kondisi diperlukan untuk prinsipal layanan, dan harus menggunakan kunci konteks global dan. `AWS:SourceAccount` `AWS:SourceArn`

`AWS:SourceAccount`Ini adalah ID akun yang memanggil bot Amazon Lex V2.

`AWS:SourceArn`Ini adalah ARN sumber daya dari instans layanan Connect Customer atau fungsi Lambda tempat panggilan ke alias bot Amazon Lex V2 berasal.

Kebijakan terkelola AWS untuk Amazon Lex V2

Kebijakan terkelola AWS adalah kebijakan mandiri yang dibuat dan dikelola oleh AWS. Kebijakan terkelola AWS dirancang untuk memberikan izin bagi banyak kasus penggunaan umum sehingga Anda dapat mulai menetapkan izin kepada pengguna, grup, dan peran.

Perlu diingat bahwa kebijakan yang dikelola AWS mungkin tidak memberikan izin hak istimewa paling sedikit untuk kasus penggunaan spesifik Anda karena tersedia untuk digunakan oleh semua pelanggan AWS. Kami menyarankan Anda untuk mengurangi izin lebih lanjut dengan menentukan kebijakan yang dikelola pelanggan yang khusus untuk kasus penggunaan Anda.

Anda tidak dapat mengubah izin yang ditentukan dalam kebijakan terkelola AWS. Jika AWS memperbarui izin yang ditentukan dalam kebijakan terkelola AWS, pembaruan memengaruhi semua identitas utama (pengguna, grup, dan peran) yang dilampirkan kebijakan tersebut. AWS kemungkinan besar akan memperbarui kebijakan terkelola AWS saat kebijakan baru Layanan AWS diluncurkan atau operasi API baru tersedia untuk layanan yang ada.

Untuk informasi selengkapnya, lihat kebijakan terkelola AWS di Panduan Pengguna IAM.

Kebijakan terkelola AWS: AmazonLexReadOnly

Anda dapat melampirkan kebijakan `AmazonLexReadOnly` ke identitas IAM Anda.

Kebijakan ini memberikan izin hanya-baca yang memungkinkan pengguna melihat semua tindakan di Amazon Lex V2 dan layanan pembuatan model Amazon Lex.

Detail izin

Kebijakan ini mencakup izin berikut:

- `lex`— Read-only akses ke sumber daya Amazon Lex V2 dan Amazon Lex dalam layanan pembuatan model.

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": "*"
        }
    ]
}
```

Kebijakan terkelola AWS: AmazonLexRunBotsOnly

Anda dapat melampirkan kebijakan `AmazonLexRunBotsOnly` ke identitas IAM Anda.

Kebijakan ini memberikan izin hanya-baca yang memungkinkan akses untuk menjalankan bot percakapan Amazon Lex V2 dan Amazon Lex.

Detail izin

Kebijakan ini mencakup izin berikut:

- `lex`— Read-only akses ke semua tindakan di runtime Amazon Lex V2 dan Amazon Lex.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "lex:PostContent",
                "lex:PostText",
                "lex:PutSession",
                "lex:GetSession",
                "lex:DeleteSession",
                "lex:RecognizeText",
                "lex:RecognizeUtterance",
                "lex:StartConversation"
            ],
            "Resource": "*"
        }
```

```
    ]
}
```

Kebijakan terkelola AWS: AmazonLexFullAccess

Anda dapat melampirkan kebijakan `AmazonLexFullAccess` ke identitas IAM Anda.

Kebijakan ini memberikan izin administratif yang memungkinkan pengguna untuk membuat, membaca, memperbarui, dan menghapus sumber daya Amazon Lex V2 dan Amazon Lex; dan menjalankan bot percakapan Amazon Lex V2 dan Amazon Lex.

Detail izin

Kebijakan ini mencakup izin berikut:

- `lex`— Memungkinkan kepala sekolah membaca dan menulis akses ke semua tindakan di Amazon Lex V2 dan Amazon Lex model building dan layanan runtime.
- `cloudwatch`— Memungkinkan kepala sekolah untuk melihat CloudWatch metrik dan alarm Amazon.
- `iam`— Memungkinkan prinsipal untuk membuat dan menghapus peran terkait layanan, meneruskan peran, dan melampirkan serta melepaskan kebijakan ke peran. Izin dibatasi untuk “lex.amazonaws.com” untuk operasi Amazon Lex dan “lexv2.amazonaws.com” untuk operasi Amazon Lex V2.
- `kendra`— Memungkinkan kepala sekolah untuk membuat daftar indeks Amazon Kendra.
- `kms`— Memungkinkan kepala sekolah untuk mendeskripsikan AWS KMS kunci dan alias.
- `lambda`— Memungkinkan kepala sekolah untuk membuat daftar AWS Lambda fungsi dan mengelola izin yang dilampirkan ke fungsi Lambda apa pun.
- `polly`— Memungkinkan kepala sekolah untuk mendeskripsikan suara Amazon Polly dan mensintesis ucapan.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AmazonLexFullAccessStatement1",
```

```
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:DescribeAlarms",
                "cloudwatch:DescribeAlarmsForMetric",
                "kms:DescribeKey",
                "kms:ListAliases",
                "lambda:GetPolicy",
                "lambda:ListFunctions",
                "lambda:ListAliases",
                "lambda:ListVersionsByFunction",
                "lex:*",
                "polly:DescribeVoices",
                "polly:SynthesizeSpeech",
                "kendra:ListIndices",
                "iam:ListRoles",
                "s3:ListAllMyBuckets",
                "logs:DescribeLogGroups",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Sid": "AmazonLexFullAccessStatement2",
            "Effect": "Allow",
            "Action": [
                "bedrock:ListFoundationModels"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "bedrock:InvokeModel"
            ],
            "Resource": "arn:aws:bedrock:*::foundation-model/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "lambda:AddPermission",
                "lambda:RemovePermission"
```

```
            ],
            "Resource": "arn:aws:lambda:*:*:function:AmazonLex*",
            "Condition": {
                "StringEquals": {
                    "lambda:Principal": "lex.amazonaws.com"
                }
            }
        },
        {
            "Sid": "AmazonLexFullAccessStatement3",
            "Effect": "Allow",
            "Action": [
                "iam:GetRole",
                "iam:GetRolePolicy"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/lex.amazonaws.com/
AWSServiceRoleForLexBots",
                "arn:aws:iam::*:role/aws-service-role/channels.lex.amazonaws.com/
AWSServiceRoleForLexChannels",
                "arn:aws:iam::*:role/aws-service-role/lexv2.amazonaws.com/
AWSServiceRoleForLexV2Bots*",
                "arn:aws:iam::*:role/aws-service-role/
channels.lexv2.amazonaws.com/AWSServiceRoleForLexV2Channels*",
                "arn:aws:iam::*:role/aws-service-role/
replication.lexv2.amazonaws.com/AWSServiceRoleForLexV2Replication*"
            ]
        },
        {
            "Sid": "AmazonLexFullAccessStatement4",
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/lex.amazonaws.com/
AWSServiceRoleForLexBots"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "lex.amazonaws.com"
                }
            }
        },
```

```
        {
            "Sid": "AmazonLexFullAccessStatement5",
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/channels.lex.amazonaws.com/
AWSServiceRoleForLexChannels"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "channels.lex.amazonaws.com"
                }
            }
        },
        {
            "Sid": "AmazonLexFullAccessStatement6",
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/lexv2.amazonaws.com/
AWSServiceRoleForLexV2Bots*"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "lexv2.amazonaws.com"
                }
            }
        },
        {
            "Sid": "AmazonLexFullAccessStatement7",
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/
channels.lexv2.amazonaws.com/AWSServiceRoleForLexV2Channels*"
            ],
            "Condition": {
                "StringEquals": {
```

```
                    "iam:AWSServiceName": "channels.lexv2.amazonaws.com"
                }
            }
        },
        {
            "Sid": "AmazonLexFullAccessStatement8",
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/
replication.lexv2.amazonaws.com/AWSServiceRoleForLexV2Replication*"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "replication.lexv2.amazonaws.com"
                }
            }
        },
        {
            "Sid": "AmazonLexFullAccessStatement9",
            "Effect": "Allow",
            "Action": [
                "iam:DeleteServiceLinkedRole",
                "iam:GetServiceLinkedRoleDeletionStatus"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/lex.amazonaws.com/
AWSServiceRoleForLexBots",
                "arn:aws:iam::*:role/aws-service-role/channels.lex.amazonaws.com/
AWSServiceRoleForLexChannels",
                "arn:aws:iam::*:role/aws-service-role/lexv2.amazonaws.com/
AWSServiceRoleForLexV2Bots*",
                "arn:aws:iam::*:role/aws-service-role/
channels.lexv2.amazonaws.com/AWSServiceRoleForLexV2Channels*",
                "arn:aws:iam::*:role/aws-service-role/
replication.lexv2.amazonaws.com/AWSServiceRoleForLexV2Replication*"
            ]
        },
        {
            "Sid": "AmazonLexFullAccessStatement10",
            "Effect": "Allow",
            "Action": [
```

```
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/lex.amazonaws.com/
AWSServiceRoleForLexBots"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "lex.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Sid": "AmazonLexFullAccessStatement11",
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/lexv2.amazonaws.com/
AWSServiceRoleForLexV2Bots*"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "lexv2.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Sid": "AmazonLexFullAccessStatement12",
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/
channels.lexv2.amazonaws.com/AWSServiceRoleForLexV2Channels*"
            ],
            "Condition": {
                "StringEquals": {
```

```
                    "iam:PassedToService": [
                        "channels.lexv2.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Sid": "AmazonLexFullAccessStatement13",
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/
replication.lexv2.amazonaws.com/AWSServiceRoleForLexV2Replication*"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "lexv2.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

Kebijakan terkelola AWS: AmazonLexReplicationPolicy

Anda tidak dapat melampirkan `AmazonLexReplicationPolicy` ke entitas IAM Anda. Kebijakan ini dilampirkan pada peran terkait layanan yang memungkinkan Amazon Lex V2 melakukan tindakan atas nama Anda. Untuk informasi selengkapnya, lihat Menggunakan peran terkait layanan untuk Amazon Lex V2.

Kebijakan ini memberikan izin administratif yang memungkinkan Amazon Lex V2 mereplikasi sumber daya AWS di seluruh Wilayah atas nama Anda. Anda dapat melampirkan kebijakan ini untuk mengizinkan peran mereplikasi sumber daya dengan mudah, termasuk bot, lokal, versi, alias, maksud, jenis slot, slot, dan kosakata khusus.

Detail izin

Kebijakan ini mencakup izin berikut.

- `lex`— Memungkinkan kepala sekolah untuk mereplikasi sumber daya di Wilayah lain.
- `iam`— Memungkinkan kepala sekolah untuk lulus peran dari IAM. Ini diperlukan agar Amazon Lex V2 memiliki izin untuk mereplikasi sumber daya di Wilayah lain.

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
 {
  "Sid": "ReplicationPolicyStatement1",
  "Effect": "Allow",
  "Action": [
   "lex:BuildBotLocale",
   "lex:ListBotLocales",
   "lex:CreateBotAlias",
   "lex:UpdateBotAlias",
   "lex:DeleteBotAlias",
   "lex:DescribeBotAlias",
   "lex:CreateBotVersion",
   "lex:DeleteBotVersion",
   "lex:DescribeBotVersion",
   "lex:CreateExport",
   "lex:DescribeBot",
   "lex:UpdateExport",
   "lex:DescribeExport",
   "lex:DescribeBotLocale",
   "lex:DescribeIntent",
   "lex:ListIntents",
   "lex:DescribeSlotType",
   "lex:ListSlotTypes",
   "lex:DescribeSlot",
   "lex:ListSlots",
   "lex:DescribeCustomVocabulary",
   "lex:StartImport",
   "lex:DescribeImport",
   "lex:CreateBot",
```

```
    "lex:UpdateBot",
    "lex:DeleteBot",
    "lex:CreateBotLocale",
    "lex:UpdateBotLocale",
    "lex:DeleteBotLocale",
    "lex:CreateIntent",
    "lex:UpdateIntent",
    "lex:DeleteIntent",
    "lex:CreateSlotType",
    "lex:UpdateSlotType",
    "lex:DeleteSlotType",
    "lex:CreateSlot",
    "lex:UpdateSlot",
    "lex:DeleteSlot",
    "lex:CreateCustomVocabulary",
    "lex:UpdateCustomVocabulary",
    "lex:DeleteCustomVocabulary",
    "lex:DeleteBotChannel",
    "lex:ListTagsForResource",
    "lex:TagResource",
    "lex:UntagResource",
    "lex:CreateResourcePolicy",
    "lex:DeleteResourcePolicy",
    "lex:DescribeResourcePolicy",
    "lex:UpdateResourcePolicy"
   ],
   "Resource": [
    "arn:aws:lex:*:*:bot/*",
    "arn:aws:lex:*:*:bot-alias/*"
   ]
  },
  {
   "Sid": "ReplicationPolicyStatement2",
   "Effect": "Allow",
   "Action": [
    "lex:CreateUploadUrl",
    "lex:ListBots"
   ],
   "Resource": "*"
  },
  {
   "Sid": "ReplicationPolicyStatement3",
   "Effect": "Allow",
   "Action": [
```

```
    "iam:PassRole"
   ],
   "Resource": "*",
   "Condition": {
    "StringEquals": {
     "iam:PassedToService": "lexv2.amazonaws.com"
    }
   }
  }
     ]
}
```

Kebijakan terkelola AWS: AmazonLexV2BedrockAgentPolicy

Kebijakan Amazon Lex V2 untuk agen Amazon Bedrock

Respons

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Sid": "LexV2TrustPolicy",
      "Principal": {
        "Service": "lexv2.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "{accountId}"
        }
      }
    }
  ]
}
```

Kebijakan terkelola AWS: AmazonLexV2BedrockKnowledgeBasePolicy

Kebijakan Amazon Lex V2 untuk basis pengetahuan Amazon Bedrock

Respons

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Sid": "LexV2TrustPolicy",
      "Principal": {
        "Service": "lexv2.amazonaws.com"
      },
      "Action": "sts:AssumeRole",
     "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "{accountId}"
        }
      }
    }
  ]
}
```

Kebijakan terkelola AWS: AmazonLexV2BedrockAgentPolicyInternal

Kebijakan internal Amazon Lex V2 untuk agen Amazon Bedrock

Respons

JSON

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

```
      "Sid": "LexV2InternalTrustPolicy",
      "Principal": {
        "Service": "lexv2.aws.internal"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "{accountId}"
        }
      }
    }
  ]
}
```

Kebijakan terkelola AWS: AmazonLexV2BedrockKnowledgeBasePolicyInternal

Kebijakan internal Amazon Lex V2 untuk basis pengetahuan Amazon Bedrock

Respons

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "LexV2InternalTrustPolicy",
      "Effect": "Allow",
      "Principal": {
        "Service": "lexv2.aws.internal"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "aws:SourceAccount": "{accountId}"
        }
      }
    }
  ]
}
```

Pembaruan Amazon Lex V2 ke kebijakan terkelola AWS

Lihat detail tentang pembaruan kebijakan terkelola AWS untuk Amazon Lex V2 sejak layanan ini mulai melacak perubahan ini. Untuk peringatan otomatis tentang perubahan pada halaman ini, berlangganan umpan RSS di halaman Amazon Lex V2Riwayat dokumen untuk Amazon Lex V2.

Ubah	Deskripsi	Date
AmazonLexReplicationPolicy— Kebijakan yang diperbarui	Amazon Lex V2 memperbarui kebijakan untuk mengizinkan replikasi tag dan ResourceBasedPolicy.	Juni 24, 2025
AmazonLexV2BedrockKnowledgeBasePolicyInternal – Kebijakan baru	Amazon Lex V2 menambahkan kebijakan baru untuk memungkinkan replikasi sumber daya basis pengetahuan Amazon Bedrock.	Agustus 30, 2024
AmazonLexV2BedrockAgentPolicyInternal – Kebijakan baru	Amazon Lex V2 menambahkan kebijakan baru untuk mengizinkan replikasi sumber daya agen Amazon Bedrock.	Agustus 30, 2024
AmazonLexV2BedrockKnowledgeBasePolicy – Kebijakan baru	Amazon Lex V2 menambahkan kebijakan baru untuk memungkinkan replikasi sumber daya basis pengetahuan Amazon Bedrock.	Agustus 30, 2024
AmazonLexV2BedrockAgentPolicy – Kebijakan baru	Amazon Lex V2 menambahkan kebijakan baru untuk	Agustus 30, 2024

Ubah	Deskripsi	Date
	mengizinkan replikasi sumber daya agen Amazon Bedrock.	
AmazonLexReadOnly – Pembaruan ke kebijakan yang ada	Amazon Lex V2 menambahkan izin baru untuk mengizinkan replika akses read-only dari sumber daya bot.	10 Mei 2024
AmazonLexFullAccess – Pembaruan ke kebijakan yang ada	Amazon Lex V2 menambahkan izin baru untuk memungkinkan replikasi sumber daya bot ke wilayah lain.	16 April 2024
AmazonLexFullAccess – Pembaruan ke kebijakan yang ada	Amazon Lex V2 menambahkan izin baru untuk memungkinkan replikasi sumber daya bot ke wilayah lain.	Januari 31, 2024
AmazonLexReplicationPolicy – Kebijakan baru	Amazon Lex V2 menambahkan kebijakan baru untuk memungkinkan replikasi sumber daya bot ke wilayah lain.	Januari 31, 2024
AmazonLexReadOnly – Pembaruan ke kebijakan yang ada	Amazon Lex V2 menambahkan izin baru untuk memungkinkan akses hanya-baca untuk mencantumkan item kosakata khusus.	29 November 2022
AmazonLexFullAccess – Pembaruan ke kebijakan yang ada	Amazon Lex V2 menambahkan izin baru untuk mengizinkan akses hanya-baca ke operasi layanan pembuatan model Amazon Lex V2.	18 Agustus 2021

Ubah	Deskripsi	Date
AmazonLexReadOnly – Pembaruan ke kebijakan yang ada	Amazon Lex V2 menambahkan izin baru untuk memungkinkan akses hanya-baca ke operasi Amazon Lex V2 Automated Chatbot Designer.	1 Desember 2021
AmazonLexFullAccess – Pembaruan ke kebijakan yang ada	Amazon Lex V2 menambahkan izin baru untuk mengizinkan akses hanya-baca ke operasi layanan pembuatan model Amazon Lex V2.	18 Agustus 2021
AmazonLexReadOnly – Pembaruan ke kebijakan yang ada	Amazon Lex V2 menambahkan izin baru untuk mengizinkan akses hanya-baca ke operasi layanan pembuatan model Amazon Lex V2.	18 Agustus 2021
AmazonLexRunBotsOnly – Pembaruan ke kebijakan yang ada	Amazon Lex V2 menambahkan izin baru untuk memungkinkan akses hanya-baca ke operasi layanan runtime Amazon Lex V2.	18 Agustus 2021
Amazon Lex V2 mulai melacak perubahan	Amazon Lex V2 mulai melacak perubahan untuk kebijakan yang dikelola AWS.	18 Agustus 2021

Menggunakan peran terkait layanan untuk Amazon Lex V2

Amazon Lex V2 menggunakan peran terkait layanan AWS Identity and Access Management (IAM). Peran terkait layanan adalah jenis peran IAM unik yang ditautkan langsung ke Amazon Lex V2. Service-linked peran telah ditentukan sebelumnya oleh Amazon Lex V2 dan menyertakan semua izin yang diperlukan layanan untuk memanggil layanan AWS lain atas nama Anda.

Peran terkait layanan membuat pengaturan Amazon Lex V2 lebih mudah karena Anda tidak perlu menambahkan izin yang diperlukan secara manual. Amazon Lex V2 mendefinisikan izin peran terkait layanannya, dan kecuali ditentukan lain, hanya Amazon Lex V2 yang dapat mengambil perannya. Izin yang ditentukan mencakup kebijakan kepercayaan dan kebijakan izin, serta bahwa kebijakan izin tidak dapat dilampirkan ke entitas IAM lainnya.

Untuk informasi tentang layanan lain yang mendukung peran terkait layanan, lihat AWS Services That Work with IAM dan cari layanan yang memiliki Ya di kolom PeranService-Linked . Pilih Ya dengan sebuah tautan untuk melihat dokumentasi peran terkait layanan untuk layanan tersebut.

Anda harus mengonfigurasi izin agar entitas IAM (seperti pengguna, grup, atau peran) dapat membuat, mengedit, atau menghapus peran terkait layanan. Untuk informasi selengkapnya, lihat Izin Service-Linked Peran di Panduan Pengguna IAM.

Anda dapat menghapus peran terkait layanan hanya setelah menghapus sumber daya terkait terlebih dahulu. Ini melindungi sumber daya Amazon Lex V2 Anda karena Anda tidak dapat secara tidak sengaja menghapus izin untuk mengakses sumber daya.

Topik

- Membuat peran terkait layanan untuk Amazon Lex V2
- Mengedit peran terkait layanan untuk Amazon Lex V2
- Menghapus peran terkait layanan untuk Amazon Lex V2
- Service-linked izin peran untuk Amazon Lex V2
- Wilayah yang didukung untuk peran terkait layanan Amazon Lex V2

Membuat peran terkait layanan untuk Amazon Lex V2

Anda tidak perlu membuat peran terkait layanan secara manual, karena Amazon Lex V2 membuat peran terkait layanan untuk Anda saat Anda melakukan tindakan yang relevan (lihat Service-linked izin peran untuk Amazon Lex V2 untuk informasi lebih lanjut) di AWS Management Console, AWS CLI, atau AWS API.

Jika Anda menghapus peran terkait layanan ini, dan kemudian perlu membuatnya lagi, Anda dapat menggunakan proses yang sama untuk membuat peran baru di akun Anda.

Mengedit peran terkait layanan untuk Amazon Lex V2

Amazon Lex V2 tidak mengizinkan Anda mengedit peran terkait layanan. Setelah membuat peran terkait layanan, Anda tidak dapat mengubah nama peran karena berbagai entitas mungkin mereferensikan peran tersebut. Namun, Anda dapat mengedit deskripsi peran menggunakan IAM. Untuk informasi selengkapnya, lihat Mengedit Service-Linked Peran di Panduan Pengguna IAM.

Menghapus peran terkait layanan untuk Amazon Lex V2

Jika Anda tidak perlu lagi menggunakan fitur atau layanan yang memerlukan peran terkait layanan, sebaiknya hapus peran tersebut. Dengan begitu, Anda tidak memiliki entitas yang tidak digunakan yang tidak dipantau atau dipelihara secara aktif. Tetapi, Anda harus membersihkan sumber daya peran yang terhubung dengan layanan sebelum menghapusnya secara manual.

> **Note**
>
> Jika layanan Amazon Lex V2 menggunakan peran saat Anda mencoba menghapus sumber daya, maka penghapusan mungkin gagal. Jika hal itu terjadi, tunggu beberapa menit dan coba mengoperasikannya lagi.

Untuk melihat langkah-langkah menghapus sumber daya untuk peran terkait layanan tertentu di Amazon Lex V2, lihat bagian khusus untuk peran tersebut. Service-linked izin peran untuk Amazon Lex V2

Untuk menghapus peran terkait layanan secara manual menggunakan IAM

Setelah menghapus sumber daya yang terkait dengan peran terkait layanan, gunakan konsol IAM, AWS CLI, atau AWS API untuk menghapus peran tersebut. Untuk informasi selengkapnya, lihat Menghapus Service-Linked Peran di Panduan Pengguna IAM.

Service-linked izin peran untuk Amazon Lex V2

Amazon Lex V2 menggunakan peran terkait layanan dengan awalan berikut.

Topik

- AWSServiceRoleForLexV2Bots_
- AWSServiceRoleForLexV2Channels_
- AWSServiceRoleForLexV2Replication

AWSServiceRoleForLexV2Bots_

Peran AWSServiceRoleForLexV2Bots _ memberikan izin untuk menghubungkan bot Anda ke layanan lain yang diperlukan. Peran ini mencakup kebijakan kepercayaan untuk mengizinkan layanan lexv2.amazonaws.com mengambil peran dan menyertakan izin untuk melakukan tindakan berikut.

- Gunakan Amazon Polly untuk mensintesis ucapan di semua sumber daya Amazon Lex V2 yang didukung tindakan tersebut.
- Jika bot dikonfigurasi untuk menggunakan analisis sentimen Amazon Comprehend, deteksi sentimen pada semua sumber daya Amazon Lex V2 yang didukung tindakan tersebut.
- Jika bot dikonfigurasi untuk menyimpan log audio di bucket S3, masukkan objek ke dalam bucket tertentu.
- Jika bot dikonfigurasi untuk menyimpan log audio dan teks, buat aliran log dan masukkan log ke grup log tertentu.
- Jika bot dikonfigurasi untuk menggunakan AWS KMS kunci untuk mengenkripsi data, buat kunci data tertentu.
- Jika bot dikonfigurasi untuk menggunakan `KendraSearchIntent` intent, kueri akses ke indeks Amazon Kendra yang ditentukan.

Untuk membuat peran

Amazon Lex V2 membuat peran AWSServiceRoleForLexV2Bots _ baru dengan akhiran acak di akun Anda setiap kali Anda membuat bot. Amazon Lex V2 memodifikasi peran saat Anda menambahkan kemampuan tambahan ke bot. Misalnya, jika Anda menambahkan analisis sentimen Amazon Comprehend ke bot, Amazon Lex V2 menambahkan izin `lex:DetectSentiment` untuk tindakan ke peran layanan.

Untuk menghapus peran

1. Masuk ke AWS Management Console dan buka konsol Amazon Lex di https://console.aws.amazon.com/lex/.
2. Dari panel navigasi kiri, pilih Bots dan pilih bot yang peran terkait layanannya ingin Anda hapus.
3. Pilih versi bot apa pun.
4. Peran runtime izin IAM ada di detail Versi.
5. Kembali ke halaman Bots dan pilih tombol radio di sebelah bot untuk dihapus.
6. Pilih Tindakan dan kemudian pilih Hapus.

7. Ikuti langkah-langkah di Menghapus peran terkait layanan untuk menghapus peran IAM.

AWSServiceRoleForLexV2Channels_

Peran AWSServiceRoleForLexV2Channels _ memberikan izin untuk membuat daftar bot di akun dan memanggil API percakapan untuk bot. Peran ini mencakup kebijakan kepercayaan untuk memungkinkan layanan channels.lexv2.amazonaws.com untuk mengambil peran tersebut. Jika bot dikonfigurasi untuk menggunakan saluran untuk berkomunikasi dengan layanan pesan, kebijakan izin peran AWSServiceRoleForLexV2Channels _ memungkinkan Amazon Lex V2 menyelesaikan tindakan berikut.

- Buat daftar izin pada semua bot di akun.
- Mengenali teks, mendapatkan sesi dan menempatkan izin sesi pada alias bot tertentu.

Untuk membuat peran

Saat Anda membuat integrasi saluran untuk menyebarkan bot di platform perpesanan, Amazon Lex V2 membuat peran terkait layanan baru di akun Anda untuk setiap saluran dengan akhiran acak.

Untuk menghapus peran

1. Masuk ke AWS Management Console dan buka konsol Amazon Lex di https://console.aws.amazon.com/lex/.
2. Dari panel navigasi kiri, pilih Bots.
3. Pilih bot.
4. Dari panel navigasi kiri, pilih Integrasi saluran di bawah Deployment.
5. Pilih saluran yang peran terkait layanannya ingin Anda hapus.
6. Peran runtime izin IAM ada di konfigurasi Umum
7. Pilih Hapus, lalu pilih Hapus lagi untuk menghapus saluran.
8. Ikuti langkah-langkah di Menghapus peran terkait layanan untuk menghapus peran IAM.

AWSServiceRoleForLexV2Replication

AWSServiceRoleForLexV2Replication Peran memberikan izin untuk mereplikasi bot di wilayah kedua. Peran ini mencakup kebijakan kepercayaan untuk memungkinkan layanan

replication.lexv2.amazonaws.com mengambil peran dan juga menyertakan kebijakan terkelola AmazonLexReplicationPolicyAWS, yang memungkinkan izin untuk tindakan berikut.

- Lulus peran bot IAM ke bot replika untuk menggandakan kembali izin yang sesuai untuk bot replika.
- Buat dan kelola bot dan sumber daya bot (versi, alias, maksud, slot, kosakata khusus, dan sebagainya) di Wilayah lain.

Untuk membuat peran

Saat Anda mengaktifkan Ketahanan Global untuk bot, Amazon Lex V2 akan membuat peran AWSServiceRoleForLexV2Replication terkait layanan di akun Anda. Pastikan Anda memiliki izin yang benar untuk memberikan izin layanan Amazon Lex V2 untuk membuat peran terkait layanan.

Untuk menghapus sumber daya Amazon Lex V2 yang digunakan oleh AWSServiceRoleForLexV2Replication sehingga Anda dapat menghapus peran

1. Masuk ke AWS Management Console dan buka konsol Amazon Lex di https://console.aws.amazon.com/lex/.
2. Pilih bot yang Ketahanan Globalnya diaktifkan.
3. Pilih Ketahanan Global di bawah Deployment.
4. Pilih Nonaktifkan Ketahanan Global.
5. Ulangi proses untuk semua bot yang mengaktifkan Ketahanan Global.
6. Ikuti langkah-langkah di Menghapus peran terkait layanan untuk menghapus peran IAM.

Wilayah yang didukung untuk peran terkait layanan Amazon Lex V2

Amazon Lex V2 mendukung penggunaan peran terkait layanan di semua wilayah tempat layanan tersedia. Untuk informasi lebih lanjut, lihat Wilayah dan Titik Akhir AWS .

Menggunakan peran layanan untuk Amazon Lex V2

Amazon Lex V2 menggunakan peran layanan IAM untuk mengakses layanan AWS lain atas nama Anda saat runtime. Peran layanan memberikan izin yang diperlukan untuk membangun dan menjalankan bot Anda. Peran layanan adalah peran IAM standar yang dikelola pelanggan di akun Anda. Ini menghormati kebijakan kontrol layanan (SCP) yang Anda konfigurasikan di AWS Organizations dan memberi Anda kontrol penuh atas izin yang digunakan Amazon Lex V2.

AmazonLexServiceRole-

Peran AmazonLexServiceRole - memberikan izin untuk menghubungkan bot Anda ke layanan lain yang diperlukan. Peran ini mencakup kebijakan kepercayaan untuk mengizinkan layanan lexv2.amazonaws.com mengambil peran dan menyertakan izin untuk melakukan tindakan berikut.

- Jika bot dikonfigurasi untuk menggunakan text-to-speech, gunakan Amazon Polly untuk mensintesis ucapan di semua resource Amazon Lex V2 yang didukung tindakan tersebut.
- Jika bot dikonfigurasi untuk menggunakan analisis sentimen Amazon Comprehend, deteksi sentimen pada semua sumber daya Amazon Lex V2 yang didukung tindakan tersebut.
- Jika bot dikonfigurasi untuk menyimpan log audio di bucket S3, masukkan objek ke dalam bucket tertentu.
- Jika bot dikonfigurasi untuk menyimpan log audio dan teks, buat aliran log dan masukkan log ke grup log tertentu.
- Jika bot dikonfigurasi untuk menggunakan AWS KMS kunci untuk mengenkripsi data, buat kunci data tertentu.
- Jika bot dikonfigurasi untuk menggunakan fitur AI generatif Amazon Bedrock, panggil model Amazon Bedrock yang ditentukan, profil inferensi, dan basis pengetahuan.

Menciptakan peran

Amazon Lex V2 membuat peran baru AmazonLexServiceRole dengan akhiran acak di akun Anda saat Anda membuat bot dan memilih untuk membuat peran baru. Amazon Lex V2 memodifikasi peran saat Anda menambahkan kemampuan tambahan ke bot. Misalnya, jika Anda menambahkan analisis sentimen Amazon Comprehend ke bot, Amazon Lex V2 menambahkan izin `comprehend:DetectSentiment` untuk tindakan ke peran layanan.

Anda juga dapat memilih peran IAM yang sudah ada yang telah Anda buat dan konfigurasikan sendiri, selama itu mempercayai prinsipal layanan Amazon Lex V2 ()`lexv2.amazonaws.com`.

Mengedit atau menghapus peran

Karena peran layanan adalah peran IAM standar yang dikelola pelanggan, Anda mengelolanya dengan cara yang sama seperti peran IAM lainnya. Gunakan konsol IAM, AWS CLI, atau IAM API. Anda dapat meninjau kebijakan kepercayaan peran dan kebijakan izin terlampir kapan saja di konsol IAM.

> **Important**
>
> Sebelum Anda menghapus peran layanan, pastikan tidak ada bot yang masih menggunakannya. Menghapus peran yang bergantung pada bot menyebabkan bot kehilangan izin yang diperlukan untuk menjalankannya.

Wilayah yang Didukung untuk peran layanan Amazon Lex V2

Amazon Lex V2 mendukung penggunaan peran layanan di semua Wilayah tempat layanan tersedia. Untuk informasi lebih lanjut, lihat Wilayah dan Titik Akhir AWS .

Migrasi ke peran layanan untuk Amazon Lex V2

Amazon Lex V2 sedang beralih dari peran AWS-managed terkait layanan (SLR) ke peran layanan yang dikelola pelanggan untuk izin runtime bot. Topik ini menjelaskan mengapa Amazon Lex V2 membuat perubahan ini, cara memigrasikan bot yang ada, dan garis waktu transisi.

Mengapa Amazon Lex V2 pindah ke peran layanan

Peran terkait layanan adalah peran IAM yang ditautkan langsung ke Amazon Lex V2 dan dikelola oleh AWS. Service-linked peran tidak menghormati kebijakan kontrol layanan (SCP) yang Anda konfigurasikan di AWS Organizations.

Peran layanan adalah peran IAM yang Anda miliki dan kelola di akun Anda sendiri. Karena peran layanan adalah peran standar yang dikelola pelanggan, ia menghormati kebijakan kontrol layanan (SCP) yang Anda konfigurasikan di organisasi Anda. Ini memberi Anda kontrol penuh atas izin yang digunakan Amazon Lex V2 atas nama Anda. Pindah ke peran layanan menyelaraskan Amazon Lex V2 dengan praktik terbaik IAM dan meningkatkan postur keamanan dan kepatuhan Anda.

Migrasikan bot yang ada ke peran layanan

Anda dapat memigrasikan bot yang ada yang menggunakan peran terkait layanan ke peran layanan yang dikelola pelanggan langsung dari konsol Amazon Lex V2.

> **Note**
>
> Amazon Lex V2 berencana untuk membuat migrasi bot tunggal menggunakan tombol Migrasi ke Peran Layanan tersedia sebelum 1 September 2026. Setelah tombol tersedia, selesaikan langkah-langkah berikut untuk memigrasikan bot.

1. Buka konsol Amazon Lex V2 di https://console.aws.amazon.com/lexv2/home.
2. Di panel navigasi kiri, pilih Bot, lalu pilih bot yang ingin Anda migrasi.
3. Pada halaman versi Draf bot, jika bot menggunakan peran terkait layanan, sebuah spanduk muncul yang menunjukkan bahwa bot menggunakan peran terkait layanan. Pilih Migrasi ke Peran Layanan.
4. Amazon Lex V2 menciptakan peran layanan baru yang dikelola pelanggan, melampirkan izin yang dibutuhkan bot Anda, dan memperbarui bot untuk menggunakan peran baru.
5. Saat migrasi selesai, spanduk peran terkait layanan dan tombol Migrasi ke Peran Layanan tidak lagi muncul untuk bot, mengonfirmasi bahwa bot sekarang menggunakan peran layanan.

> **Note**
>
> Tombol Migrasi ke Peran Layanan hanya muncul untuk bot yang menggunakan peran terkait layanan. Migrasi tidak mengubah atau menghapus peran asli terkait layanan Anda. Jika migrasi tidak selesai, bot Anda terus menggunakan peran terkait layanan yang ada dan tetap berfungsi penuh.

Garis waktu migrasi

Transisi dari peran terkait layanan ke peran layanan mengikuti garis waktu yang dijelaskan dalam daftar berikut.

- Bot baru — Bot baru yang dibuat di konsol sekarang default ke peran layanan. Kami merekomendasikan membuat bot baru dengan peran layanan.
- Migrasi satu bot — Tombol Migrasi ke Peran Layanan sekali klik hanya muncul untuk bot yang menggunakan peran terkait layanan. Dengan satu klik, Anda dapat memigrasikan bot ke peran layanan. Amazon Lex V2 berencana untuk menyediakan ini sebelum 1 September 2026.

- Migrasi beberapa bot — Wizard mencantumkan bot Anda sehingga Anda dapat memilih beberapa bot peran terkait layanan dan memigrasikannya ke peran layanan sekaligus. Amazon Lex V2 berencana untuk membuat ini tersedia pada 15 September 2026.
- Akhir dukungan untuk peran terkait layanan — Dukungan untuk peran terkait layanan berakhir pada akhir 2026.

Memecahkan masalah identitas dan akses Amazon Lex V2

Gunakan informasi berikut untuk membantu Anda mendiagnosis dan memperbaiki masalah umum yang mungkin Anda temui saat bekerja dengan Amazon Lex V2 dan IAM.

Topik

- Saya tidak berwenang untuk melakukan tindakan di Amazon Lex V2
- Saya tidak berwenang untuk melakukan iam: PassRole
- Saya seorang administrator dan ingin mengizinkan orang lain mengakses Amazon Lex V2
- Berikan akses terprogram ke pengguna
- Saya ingin mengizinkan orang di luar akun AWS saya untuk mengakses sumber daya Amazon Lex V2 saya

Saya tidak berwenang untuk melakukan tindakan di Amazon Lex V2

Jika Konsol Manajemen AWS memberi tahu bahwa Anda tidak terotorisasi untuk mengambil tindakan, Anda harus menghubungi administrator untuk mendapatkan bantuan. Administrator Anda adalah orang yang memberi Anda kredensial masuk.

Contoh kesalahan berikut terjadi ketika pengguna IAM `mateojackson` mencoba menggunakan konsol untuk melihat detail tentang suatu sumber daya fiktif *`my-example-widget`*, tetapi tidak memiliki izin fiktif `lex:`*`GetWidget`*.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 lex:GetWidget on resource: my-example-widget
```

Dalam hal ini, Mateo meminta administratornya untuk memperbarui kebijakannya agar dia dapat mengakses *`my-example-widget`* menggunakan `lex:`*`GetWidget`* tindakan.

Saya tidak berwenang untuk melakukan iam: PassRole

Jika Anda menerima kesalahan bahwa Anda tidak berwenang untuk melakukan `iam:PassRole` tindakan, kebijakan Anda harus diperbarui agar Anda dapat meneruskan peran ke Amazon Lex V2.

Beberapa Layanan AWS memungkinkan Anda untuk meneruskan peran yang ada ke layanan tersebut alih-alih membuat peran layanan baru atau peran terkait layanan. Untuk melakukannya, Anda harus memiliki izin untuk meneruskan peran ke layanan.

Contoh kesalahan berikut terjadi ketika pengguna IAM bernama `marymajor` mencoba menggunakan konsol untuk melakukan tindakan di Amazon Lex V2. Namun, tindakan tersebut memerlukan layanan untuk mendapatkan izin yang diberikan oleh peran layanan. Mary tidak memiliki izin untuk meneruskan peran tersebut pada layanan.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

Dalam kasus ini, kebijakan Mary harus diperbarui agar dia mendapatkan izin untuk melakukan tindakan `iam:PassRole` tersebut.

Jika Anda memerlukan bantuan, hubungi administrator AWS Anda. Administrator Anda adalah orang yang memberi Anda kredensial masuk.

Saya seorang administrator dan ingin mengizinkan orang lain mengakses Amazon Lex V2

Untuk mengizinkan orang lain mengakses Amazon Lex V2, Anda harus memberikan izin kepada orang atau aplikasi yang memerlukan akses. Jika Anda menggunakan AWS IAM Identity Center untuk mengelola orang dan aplikasi, Anda menetapkan set izin kepada pengguna atau grup untuk menentukan tingkat akses mereka. Set izin secara otomatis membuat dan menetapkan kebijakan IAM ke peran IAM yang terkait dengan orang atau aplikasi. Untuk informasi selengkapnya, lihat Set izin di Panduan AWS IAM Identity Center Pengguna.

Jika Anda tidak menggunakan IAM Identity Center, Anda harus membuat entitas IAM (pengguna atau peran) untuk orang atau aplikasi yang membutuhkan akses. Anda kemudian harus melampirkan kebijakan ke entitas yang memberi mereka izin yang benar di Amazon Lex V2. Setelah izin diberikan, berikan kredensialnya kepada pengguna atau pengembang aplikasi. Mereka akan menggunakan kredensyal tersebut untuk mengakses AWS. Untuk mempelajari selengkapnya tentang membuat

pengguna, grup, kebijakan, dan izin IAM, lihat Identitas dan Kebijakan IAM dan izin di IAM di Panduan Pengguna IAM.

Berikan akses terprogram ke pengguna

Untuk informasi tentang cara mendapatkan ID kunci akses dan kunci akses rahasia, lihat Memahami dan mendapatkan kredensi AWS Anda di AWS General Reference.

Saya ingin mengizinkan orang di luar akun AWS saya untuk mengakses sumber daya Amazon Lex V2 saya

Anda dapat membuat peran yang dapat digunakan pengguna di akun lain atau orang-orang di luar organisasi Anda untuk mengakses sumber daya Anda. Anda dapat menentukan siapa saja yang dipercaya untuk mengambil peran tersebut. Untuk layanan yang mendukung kebijakan berbasis sumber daya atau daftar kontrol akses (ACL), Anda dapat menggunakan kebijakan tersebut untuk memberi orang akses ke sumber daya Anda.

Untuk mempelajari selengkapnya, periksa referensi berikut:

- Untuk mengetahui apakah Amazon Lex V2 mendukung fitur-fitur ini, lihatBagaimana Amazon Lex V2 bekerja dengan IAM.
- Untuk mempelajari cara menyediakan akses ke sumber daya Anda di seluruh sumber daya Akun AWS yang Anda miliki, lihat Menyediakan akses ke pengguna IAM di pengguna lain Akun AWS yang Anda miliki di Panduan Pengguna IAM.
- Untuk mempelajari cara menyediakan akses ke sumber daya Anda kepada pihak ketiga Akun AWS, lihat Menyediakan akses yang Akun AWS dimiliki oleh pihak ketiga dalam Panduan Pengguna IAM.
- Untuk mempelajari cara memberikan akses melalui federasi identitas, lihat Menyediakan akses ke pengguna terautentikasi eksternal (federasi identitas) dalam Panduan Pengguna IAM.
- Untuk mempelajari perbedaan antara menggunakan peran dan kebijakan berbasis sumber daya untuk akses lintas akun, lihat Akses sumber daya lintas akun di IAM di Panduan Pengguna IAM.

Pencatatan dan pemantauan di Amazon Lex V2

Pemantauan adalah bagian penting dalam menjaga keandalan, ketersediaan, dan kinerja Amazon Lex V2 dan solusi AWS Anda yang lain. AWS menyediakan alat pemantauan berikut untuk menonton

Amazon Lex V2, melaporkan ketika ada sesuatu yang salah, dan mengambil tindakan otomatis bila perlu:

- Amazon CloudWatch memantau sumber daya AWS Anda dan aplikasi yang Anda jalankan di AWS secara real time. Anda dapat mengumpulkan dan melacak metrik, membuat dasbor yang disesuaikan, dan mengatur alarm yang memberi tahu Anda atau mengambil tindakan saat metrik tertentu mencapai ambang batas yang ditentukan. Misalnya, Anda dapat CloudWatch melacak penggunaan CPU atau metrik lain dari instans Amazon EC2 Anda dan secara otomatis meluncurkan instans baru bila diperlukan. Untuk informasi selengkapnya, lihat Panduan CloudWatch Pengguna Amazon.
- AWS CloudTrailmenangkap panggilan API dan peristiwa terkait yang dibuat oleh atau atas nama akun AWS Anda dan mengirimkan file log ke bucket Amazon S3 yang Anda tentukan. Anda dapat mengidentifikasi pengguna dan akun mana yang memanggil AWS, alamat IP sumber yang melakukan panggilan, dan kapan panggilan tersebut terjadi. Untuk informasi selengkapnya, silakan lihat Panduan Pengguna AWS CloudTrail.

Validasi kepatuhan untuk Amazon Lex V2

Third-party auditor menilai keamanan dan kepatuhan Amazon Lex V2 sebagai bagian dari beberapa program kepatuhan AWS. Amazon Lex V2 adalah layanan yang memenuhi syarat HIPAA. Ini sesuai dengan PCI, SOC, dan ISO.

Pengalaman pendaftaran AWS baru kami tidak dirancang untuk beban kerja yang diatur. Jika Anda menggunakan pengalaman pendaftaran AWS baru kami, tetapi Anda ingin menggunakan AWS untuk beban kerja yang diatur, Anda dapat mendaftar ke AWS (lanjutan) atau mengaktifkan fitur-fitur lanjutan untuk lingkungan AWS Anda.

Untuk mempelajari apakah an Layanan AWS berada dalam lingkup program kepatuhan tertentu, lihat Layanan AWS di Lingkup oleh Program Kepatuhan Layanan AWS dan pilih program kepatuhan yang Anda minati. Untuk informasi umum, lihat AWS Compliance Programs .

Anda dapat mengunduh laporan audit pihak ketiga menggunakan AWS Artifact. Untuk informasi selengkapnya, lihat Mengunduh Laporan di AWS Artifact .

Tanggung jawab kepatuhan Anda saat menggunakan Layanan AWS ditentukan oleh sensitivitas data Anda, tujuan kepatuhan perusahaan Anda, dan hukum dan peraturan yang berlaku. Untuk informasi selengkapnya tentang tanggung jawab kepatuhan Anda saat menggunakan Layanan AWS, lihat AWS Security Documentation.

Ketahanan di Amazon Lex V2

Infrastruktur global AWS dibangun di sekitar Wilayah AWS dan Availability Zone. Wilayah AWS menyediakan beberapa Availability Zone yang terpisah dan terisolasi secara fisik, yang terhubung dengan jaringan berlatensi rendah, throughput yang tinggi, dan sangat redundan. Dengan Zona Ketersediaan, Anda dapat merancang serta mengoperasikan aplikasi dan basis data yang secara otomatis mengalami kegagalan di antara zona tanpa gangguan. Zona Ketersediaan lebih tersedia, toleran terhadap kegagalan, dan dapat diskalakan dibandingkan infrastruktur pusat data tunggal atau ganda tradisional.

Untuk informasi lebih lanjut tentang Wilayah AWS dan Zona Ketersediaan, lihat Infrastruktur Global AWS.

Selain infrastruktur global AWS, Amazon Lex V2 menawarkan beberapa fitur untuk membantu mendukung ketahanan data dan kebutuhan pencadangan Anda.

> **Note**
>
> Untuk informasi selengkapnya tentang Ketahanan Global di Amazon Lex V2, yang memungkinkan Anda membuat bot yang direplikasi di wilayah kedua dalam pasangan yang telah ditentukan sebelumnya, lihat Ketahanan Global.

Keamanan infrastruktur di Amazon Lex V2

Sebagai layanan terkelola, Amazon Lex V2 dilindungi oleh prosedur keamanan jaringan global AWS yang dijelaskan dalam whitepaper Amazon Web Services: Tinjauan Proses Keamanan.

Anda menggunakan panggilan API AWS yang diterbitkan untuk mengakses Amazon Lex V2 melalui jaringan. Klien harus mendukung Keamanan Lapisan Pengangkutan (TLS) 1.0 atau versi yang lebih baru. Kami merekomendasikan TLS 1.2 atau versi yang lebih baru. Klien juga harus mendukung cipher suite dengan perfect forward secrecy (PFS) seperti Ephemeral (DHE) atau Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). Diffie-Hellman Sebagian besar sistem modern seperti Java 7 dan versi lebih baru mendukung mode-mode ini.

Selain itu, permintaan harus ditandatangani menggunakan ID kunci akses dan kunci akses rahasia yang terkait dengan principal IAM. Atau Anda dapat menggunakan AWS Security Token Service (AWS STS) untuk menghasilkan kredensial keamanan sementara untuk menandatangani permintaan.

Amazon Lex V2 dan titik akhir VPC antarmuka (AWS PrivateLink)

Anda dapat membuat koneksi pribadi antara VPC Anda dan Amazon Lex V2 dengan membuat titik akhir VPC antarmuka. Titik akhir antarmuka didukung oleh AWS PrivateLink, teknologi yang memungkinkan Anda mengakses API Amazon Lex V2 secara pribadi tanpa gateway internet, perangkat NAT, koneksi VPN, atau koneksi AWS Direct Connect. Instans di VPC Anda tidak memerlukan alamat IP publik untuk berkomunikasi dengan Amazon Lex V2 API. Lalu lintas antara VPC Anda dan Amazon Lex V2 tidak meninggalkan jaringan Amazon.

Setiap titik akhir antarmuka diwakili oleh satu atau beberapa Antarmuka Jaringan Elastis di subnet Anda.

Untuk informasi selengkapnya, lihat Titik akhir VPC Antarmuka (AWS PrivateLink) di Panduan Pengguna Amazon VPC.

Pertimbangan untuk titik akhir VPC Amazon Lex V2

Sebelum Anda menyiapkan titik akhir VPC antarmuka untuk Amazon Lex V2, pastikan Anda meninjau properti dan batasan titik akhir Antarmuka di Panduan Pengguna Amazon VPC.

Amazon Lex V2 mendukung panggilan ke semua tindakan API-nya dari VPC Anda.

Membuat titik akhir VPC antarmuka untuk Amazon Lex V2

Anda dapat membuat titik akhir VPC untuk layanan Amazon Lex V2 menggunakan konsol Amazon VPC atau (AWS AWS Command Line Interface CLI). Untuk informasi selengkapnya, lihat Membuat titik akhir antarmuka dalam Panduan Pengguna Amazon VPC.

Buat titik akhir VPC untuk Amazon Lex V2 menggunakan nama layanan berikut:

- com.amazonaws. *region*.model-v2-lex
- com.amazonaws. *region*.runtime-v2-lex

Jika Anda mengaktifkan DNS pribadi untuk titik akhir, Anda dapat membuat permintaan API ke Amazon Lex V2 menggunakan nama DNS default untuk Wilayah, misalnya,. `runtime-v2-lex.us-east-1.amazonaws.com`

Untuk informasi selengkapnya, lihat Mengakses layanan melalui titik akhir antarmuka dalam Panduan Pengguna Amazon VPC.

Membuat kebijakan titik akhir VPC untuk Amazon Lex V2

Anda dapat melampirkan kebijakan titik akhir ke titik akhir VPC Anda yang mengontrol akses ke Amazon Lex V2. Kebijakan titik akhir menentukan informasi berikut:

- Prinsipal yang dapat melakukan tindakan.
- Tindakan yang dapat dilakukan.
- Sumber daya yang menjadi target tindakan.

Untuk informasi selengkapnya, lihat Mengontrol Akses ke Layanan dengan titik akhir VPC dalam Panduan Pengguna Amazon VPC.

Contoh: Kebijakan titik akhir VPC untuk tindakan Amazon Lex V2

Berikut ini adalah contoh kebijakan endpoint untuk Amazon Lex V2. Saat dilampirkan ke titik akhir, kebijakan ini memberikan akses ke tindakan Amazon Lex V2 yang terdaftar untuk semua prinsipal di semua sumber daya.

```
{
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
            "lex:RecognizeText",
            "lex:RecognizeUtterance",
            "lex:StartConversation",
            "lex:DeleteSession",
            "lex:GetSession",
            "lex:DeleteSession"
         ],
         "Resource":"*"
      }
   ]
}
```

Contoh kode untuk Amazon Lex V2 menggunakan AWS SDK

Contoh kode berikut menunjukkan cara menggunakan Amazon Lex V2 dengan kit pengembangan AWS perangkat lunak (SDK).

Skenario adalah contoh kode yang menunjukkan kepada Anda bagaimana menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan atau dikombinasikan dengan yang lain Layanan AWS.

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan Amazon Lex V2 dengan AWS SDK. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Contoh kode

- Skenario untuk Amazon Lex V2 menggunakan AWS SDK
  - Buat chatbot Amazon Lex V2 untuk melibatkan pengunjung situs web Anda

Skenario untuk Amazon Lex V2 menggunakan AWS SDK

Contoh kode berikut menunjukkan cara menerapkan skenario umum di Amazon Lex V2 dengan AWS SDK. Skenario ini menunjukkan kepada Anda cara menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam Amazon Lex V2 atau digabungkan dengan yang lain Layanan AWS. Setiap skenario menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode.

Skenario menargetkan tingkat pengalaman menengah untuk membantu Anda memahami tindakan layanan dalam konteks.

Contoh

- Buat chatbot Amazon Lex V2 untuk melibatkan pengunjung situs web Anda

Buat chatbot Amazon Lex V2 untuk melibatkan pengunjung situs web Anda

Contoh kode berikut menunjukkan cara membuat chatbot untuk melibatkan pengunjung situs web Anda.

Java

SDK untuk Java 2.x

Menunjukkan cara menggunakan Amazon Lex V2 API untuk membuat Chatbot dalam aplikasi web untuk melibatkan pengunjung situs web Anda.

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di GitHub.

Layanan yang digunakan dalam contoh ini

- Amazon Comprehend
- Amazon Lex V2
- Amazon Translate

JavaScript

SDK untuk JavaScript (v3)

Menunjukkan cara menggunakan Amazon Lex V2 API untuk membuat Chatbot dalam aplikasi web untuk melibatkan pengunjung situs web Anda.

Untuk kode sumber lengkap dan petunjuk tentang cara mengatur dan menjalankan, lihat contoh lengkap Membangun chatbot Amazon Lex V2 di panduan AWS SDK untuk JavaScript pengembang.

Layanan yang digunakan dalam contoh ini

- Amazon Comprehend
- Amazon Lex V2
- Amazon Translate

Untuk daftar lengkap panduan pengembang AWS SDK dan contoh kode, lihatMenggunakan Amazon Lex V2 dengan AWS SDK. Topik ini juga mencakup informasi tentang memulai dan detail tentang versi SDK sebelumnya.

Panduan dan praktik terbaik

Lihat panduan dan praktik terbaik berikut untuk mengoptimalkan perilaku dan interaksi bot Anda dengan pelanggan.

Permintaan penandatanganan

Semua permintaan pembuatan model dan runtime Amazon Lex V2 di Referensi API menggunakan tanda tangan V4 untuk mengautentikasi permintaan. Untuk informasi selengkapnya tentang mengautentikasi permintaan, lihat proses penandatanganan Versi Tanda Tangan 4 di. Referensi Umum AWS

Melindungi informasi rahasia

Operasi API runtime RecognizeTextdan RecognizeUtterancemengambil ID sesi sebagai parameter wajib. Pengembang dapat menyetel ini ke nilai apa pun yang memenuhi batasan yang dijelaskan dalam API. Kami menyarankan Anda untuk tidak menggunakan parameter ini untuk mengirim informasi rahasia apa pun, seperti login pengguna, email, atau nomor jaminan sosial. ID ini terutama digunakan untuk mengidentifikasi percakapan dengan bot secara unik.

Menangkap nilai slot dari ucapan pengguna

Amazon Lex V2 menggunakan nilai enumerasi yang Anda berikan dalam definisi tipe slot untuk melatih model pembelajaran mesinnya. Misalkan Anda mendefinisikan maksud yang dipanggil `GetPredictionIntent` dengan contoh ucapan berikut:

```
"Tell me the prediction for {sign}"
```

di mana {sign} adalah slot dengan tipe khusus `ZodiacSign` yang memiliki 12 nilai enumerasi: melalui. `Aries` `Pisces` Sekarang anggaplah pengguna mengatakan “Katakan padaku prediksi untuk bumi”:

- Amazon Lex V2 menyimpulkan bahwa “bumi” adalah ZodiacSign nilai jika Anda melakukan salah satu tindakan berikut:
  - Atur `valueSelectionStrategy` bidang untuk `ORIGINAL_VALUE` menggunakan CreateSlotTypeoperasi
  - Pilih Perluas nilai di konsol

- Amazon Lex V2 tidak mengenali nilai “bumi” jika Anda membatasi pengenalan pada nilai yang Anda tetapkan untuk jenis slot dengan melakukan salah satu tindakan berikut:
  - Atur `valueSelectionStrategy` bidang untuk `TOP_RESOLUTION` menggunakan `CreateSlotType` operasi
  - Pilih Batasi nilai slot dan sinonim di konsol

Ketika Anda mendefinisikan sinonim untuk nilai slot, mereka diakui sama dengan nilai slot. Namun, nilai slot dikembalikan bukan sinonim.

Karena Amazon Lex V2 meneruskan nilai ini ke aplikasi klien Anda atau ke fungsi Lambda, Anda harus memeriksa apakah nilai slot adalah nilai yang valid sebelum menggunakannya dalam aktivitas pemenuhan Anda.

Ketika Amazon Lex V2 memanggil fungsi Lambda atau mengembalikan hasil interaksi ucapan dengan klien Anda, kasus nilai slot tidak dijamin. Dalam interaksi teks, kasus nilai slot cocok dengan teks yang dimasukkan atau nilai slot, tergantung pada nilai `valueResolutionStrategy` bidang.

Akronim dalam nilai slot

Saat menentukan nilai slot yang berisi akronim, gunakan pola berikut:

- Huruf kapital dipisahkan oleh periode (D.V.D.)
- Huruf kapital dipisahkan oleh spasi (D V D)

Slot bawaan untuk tanggal dan waktu

Jenis slot AMAZON.Date dan AMAZON.Time built-in menangkap tanggal dan waktu (baik absolut maupun relatif). Tanggal dan waktu relatif diselesaikan pada waktu dan tanggal Amazon Lex V2 menerima permintaan dan di wilayah tempat permintaan tersebut diproses.

Untuk jenis slot `AMAZON.Time` bawaan, jika pengguna tidak menentukan bahwa waktu sebelum atau sesudah tengah hari, waktunya ambigu. Dalam hal ini, Amazon Lex V2 akan meminta pengguna lagi. Kami merekomendasikan petunjuk yang menghasilkan waktu absolut. Misalnya, gunakan prompt seperti “Kapan Anda ingin pizza Anda dikirim? Anda bisa mengatakan jam 6 sore atau 6 sore.”

Menghindari ambiguitas dalam data pelatihan untuk bot Anda

Menyediakan data pelatihan yang membingungkan di bot Anda mengurangi kemampuan Amazon Lex V2 untuk memahami masukan pengguna. Misalkan Anda memiliki dua maksud

(`OrderPizza`dan`OrderDrink`) di bot Anda, dan Anda menyertakan “Saya ingin memesan” sebagai contoh ucapan. Saat Anda membangun bot Anda, Amazon Lex V2 tidak dapat memetakan ucapan ini ke maksud tertentu. Akibatnya, ketika pengguna memasukkan ucapan ini saat runtime, Amazon Lex V2 tidak dapat memilih maksud dengan tingkat kepercayaan yang tinggi.

Jika Anda memiliki dua maksud dengan contoh ucapan yang sama, gunakan konteks input untuk membantu Amazon Lex V2 membedakan antara dua maksud saat runtime. Untuk informasi selengkapnya, lihat Menyetel konteks maksud.

Menggunakan alias TSTALIASID

- Alias TSTALIASID bot Anda menunjuk ke versi Draft dan hanya boleh digunakan untuk pengujian manual. Amazon Lex membatasi jumlah permintaan runtime yang dapat Anda buat untuk alias TSTALIASID bot.
- Saat Anda memperbarui versi Draft bot, Amazon Lex menutup percakapan apa pun yang sedang berlangsung untuk aplikasi klien apa pun menggunakan alias TSTALIASID bot. Umumnya, Anda tidak boleh menggunakan alias TSTALIASID bot dalam produksi karena versi Draft dapat diperbarui. Anda harus mempublikasikan versi dan alias dan menggunakannya sebagai gantinya.
- Saat Anda memperbarui alias, Amazon Lex membutuhkan beberapa menit untuk mengambil perubahan. Saat Anda memodifikasi versi Draft bot, perubahan akan segera diambil oleh alias TSTALIASID.

Kuota

Kuota layanan, juga disebut sebagai batas, adalah jumlah maksimum sumber daya layanan yang diizinkan untuk akun AWS Anda. Untuk informasi selengkapnya, lihat kuota layanan AWS di referensi umum AWS.

Beberapa kuota layanan dapat disesuaikan atau ditingkatkan. Lihat kolom Adjustable dalam tabel berikut untuk melihat apakah kuota dapat disesuaikan dan ke Self-servicekolom untuk melihat apakah Anda dapat meminta penyesuaian kuota melalui konsol kuota Layanan. Kontak Dukungan untuk menambah kuota yang dapat disesuaikan, tetapi tidak melalui swalayan. Diperlukan waktu beberapa hari untuk meningkatkan kuota layanan. Jika Anda meningkatkan kuota sebagai bagian dari proyek yang lebih besar, pastikan untuk menambahkan waktu ini ke rencana Anda.

> **Note**
>
> Batas karakter dihitung sebagai jumlah unit kode Unicode. Dalam kebanyakan kasus, satu karakter Unicode setara dengan satu unit kode Unicode. Beberapa karakter khusus mungkin lebih besar dari satu unit dan jumlah mungkin berbeda untuk pengkodean yang berbeda. Untuk informasi selengkapnya tentang menghitung panjang string, lihat dokumentasi ini.

Build-time kuota

Kuota maksimum berikut diberlakukan saat Anda membuat bot.

Deskripsi	Default	Dapat disesuaikan	Self-service
Bot per akun AWS	100	Ya	Ya
Asosiasi saluran bot per akun AWS	5.000	Tidak	N/A
Pembuatan lokal paralel per akun AWS	5	Ya	Tidak
Bot per jaringan bot	5	Tidak	N/A
Jaringan bot per bot	25	Tidak	N/A

Deskripsi	Default	Dapat disesuaikan	Self-service
Versi per bot	100	Tidak	N/A
Maksud per lokal di setiap bot	• 1.000 di en-AU, en-GB, dan en-US • 250 di semua lokal lainnya	Ya	Tidak
Slot per lokal di setiap bot	• 4.000 di en-AU, en-GB, dan en-US • 2.000 di semua lokal lainnya	Tidak	N/A
Jenis slot khusus per bot lokal	• 250 di en-AU, en-GB, dan en-US • 100 di semua lokal lainnya	Tidak	N/A
Nilai jenis slot khusus dan sinonim per lokal di setiap bot	50.000	Tidak	N/A
Total karakter dalam contoh ucapan per lokal di setiap bot	• 2.000.000 di en-AU, en-GB, dan en-US • 200.000 di semua lokal lainnya	Tidak	N/A
Asosiasi saluran per bot alias	10	Tidak	N/A
Slot per niat	100	Tidak	N/A
Contoh ucapan per maksud	1.500	Ya	Ya
Karakter per contoh ucapan	500	Tidak	N/A

Deskripsi	Default	Dapat disesuaikan	Self-service
Panjang respons teks	4.000	Tidak	N/A
Contoh ucapan per slot	10	Ya	Ya
Karakter per contoh ucapan slot	500	Tidak	N/A
Prompt per slot	30	Tidak	N/A
Nilai dan sinonim per jenis slot khusus	10.000	Tidak	N/A
Karakter per nilai jenis slot khusus	500	Tidak	N/A
Karakter dalam nama asosiasi saluran	100	Tidak	N/A
Jumlah pekerjaan analisis Desainer Chatbot Otomatis bersamaan di semua bot di akun Anda per Wilayah	10	Tidak	N/A
Ukuran file XMLX tipe slot tata bahasa kustom	100 KB	Tidak	N/A

Kuota runtime

Kuota maksimum berikut diberlakukan saat runtime.

Deskripsi	Default	Dapat disesuaikan	Self-service
Masukan ukuran teks untuk Recognize Textdan Recognize Utterance	1024 karakter	Tidak	N/A
Panjang input ucapan untuk `Recognize Utterance` operasi	55 detik	Ya	Tidak
Ukuran `Recognize Utterance` header	16 KB	Tidak	N/A
Ukuran permintaan gabungan dan header sesi untuk `Recognize Utterance`	12 KB	Tidak	N/A
Jumlah maksimum percakapan mode teks bersamaan untuk`Recognize Text` ,`Recognize Utterance` , atau untuk `StartConversation` TestBotAlias	2	Tidak	N/A
Jumlah maksimum percakapan mode teks bersamaan untuk`Recognize Text` ,`Recognize Utterance` ,	50	Ya	Tidak

Deskripsi	Default	Dapat disesuaikan	Self-service
atau `StartConversation` untuk alias lainnya			
Jumlah maksimum percakapan mode suara bersamaan untuk `RecognizeUtterance` TestBotAlias	2	Tidak	N/A
Jumlah maksimum percakapan mode suara bersamaan untuk alias lain `RecognizeUtterance`	125	Ya	Tidak
Jumlah maksimum percakapan mode suara bersamaan untuk `StartConversation` TestBotAlias	2	Tidak	N/A
Jumlah maksimum percakapan mode suara bersamaan untuk alias lain `StartConversation`	400	Ya	Tidak

Deskripsi	Default	Dapat disesuaikan	Self-service
Jumlah maksimum operasi manajemen sesi bersamaan (`PutSession` ,`GetSession` , atau`DeleteSession`) saat menggunakan TestBotAlias	2	Tidak	N/A
Jumlah maksimum operasi manajemen sesi bersamaan (`PutSession` ,`GetSession` , atau`DeleteSession`) saat menggunakan alias lain	50	Ya	Tidak
Ukuran input maksimum ke fungsi Lambda	12 KB	Tidak	N/A
Ukuran output maksimum dari fungsi Lambda	50 KB	Tidak	N/A
Ukuran maksimum atribut sesi dalam output fungsi Lambda (setelah pengkodean basis-64)	12 KB	Tidak	N/A

Deskripsi	Default	Dapat disesuaikan	Self-service
Batas waktu maksimum fungsi Lambda	30 detik	Ya	Tidak
Durasi maksimum untuk satu percakapan	15 menit	Tidak	Tidak
Panjang audio maksimum	55 detik	Tidak	Tidak

Panduan migrasi Amazon Lex V1 ke V2

Konsol Amazon Lex V2 dan APIs membuatnya lebih mudah untuk membangun dan mengelola bot. Gunakan panduan ini untuk mempelajari tentang peningkatan di Amazon Lex V2 API saat Anda memigrasikan bot.

Anda memigrasikan bot menggunakan konsol Amazon Lex atau API. Untuk informasi selengkapnya, lihat Memigrasi bot di panduan pengembang Amazon Lex.

Ikhtisar Amazon Lex V2

Beberapa bahasa dapat ditambahkan ke bot sehingga Anda dapat mengelolanya sebagai sumber daya tunggal. Arsitektur informasi yang disederhanakan memungkinkan Anda mengelola versi bot Anda secara efisien. Kemampuan seperti 'alur percakapan', penghematan sebagian konfigurasi bot, dan pengunggahan ucapan massal memberi Anda lebih banyak fleksibilitas.

Beberapa bahasa dalam bot

Anda dapat menambahkan beberapa bahasa dengan Amazon Lex V2 API. Anda menambahkan, memodifikasi, dan membangun setiap bahasa secara mandiri. Sumber daya seperti jenis slot dicakup pada tingkat bahasa. Anda dapat dengan cepat berpindah antar bahasa yang berbeda untuk membandingkan dan menyempurnakan percakapan. Anda dapat menggunakan satu dasbor di konsol untuk meninjau ucapan untuk semua bahasa untuk analisis dan iterasi yang lebih cepat. Operator bot dapat mengelola izin dan operasi logging untuk semua bahasa dengan satu konfigurasi bot. Anda harus menyediakan bahasa sebagai parameter runtime untuk berkomunikasi dengan bot Amazon Lex V2. Untuk informasi selengkapnya, lihat Bahasa dan lokal yang didukung oleh Amazon Lex V2.

Arsitektur informasi yang disederhanakan

Amazon Lex V2 API mengikuti arsitektur informasi yang disederhanakan (IA) dengan tipe maksud dan slot yang dicakup ke bahasa. Versi Anda di tingkat bot sehingga sumber daya seperti maksud dan jenis slot tidak diversi satu per satu. Secara default, bot dibuat dengan versi Draft yang bisa berubah dan digunakan untuk menguji perubahan. Anda dapat membuat snapshot bernomor dari versi draf. Anda memilih bahasa yang akan disertakan dalam versi. Semua sumber daya dalam bot (bahasa, maksud, jenis slot) diarsipkan sebagai bagian dari pembuatan versi bot. Untuk informasi selengkapnya, lihat Versi.

Peningkatan produktivitas pembangun

Anda memiliki alat dan kemampuan produktivitas pembangun tambahan yang memberi Anda lebih banyak fleksibilitas dan kontrol atas proses desain bot Anda.

Simpan konfigurasi sebagian

Amazon Lex V2 API memungkinkan Anda menyimpan sebagian perubahan selama pengembangan. Misalnya, Anda dapat menyimpan slot yang mereferensikan jenis slot yang dihapus. Fleksibilitas ini memungkinkan Anda untuk menyimpan pekerjaan Anda dan kembali ke sana nanti. Anda dapat menyelesaikan perubahan ini sebelum membuat bot. Di Amazon Lex V2, penyimpanan sebagian dapat diterapkan ke slot, versi, dan alias.

Mengganti nama sumber daya

Dengan Amazon Lex V2 Anda dapat mengganti nama sumber daya setelah dibuat. Gunakan nama sumber daya untuk mengaitkan metadata yang mudah digunakan dengan setiap sumber daya. Amazon Lex V2 API menetapkan setiap sumber daya ID sumber daya 10 karakter yang unik. Semua sumber daya memiliki nama sumber daya. Anda dapat mengganti nama sumber daya berikut:

- Bot
- Niat
- Jenis slot
- Slot
- Alias

Anda dapat menggunakan sumber daya IDs untuk membaca dan memodifikasi sumber daya Anda. Jika Anda menggunakan AWS Command Line Interface atau Amazon Lex V2 API untuk bekerja dengan Amazon Lex V2, sumber daya IDs diperlukan untuk perintah tertentu.

Manajemen fungsi Lambda yang disederhanakan

Di Amazon Lex V2 API, Anda menentukan satu fungsi Lambda per bahasa, bukan fungsi untuk setiap intent. Fungsi Lambda dikonfigurasi dalam alias untuk bahasa dan digunakan untuk dialog dan hook kode pemenuhan. Anda masih dapat memilih untuk mengaktifkan atau menonaktifkan dialog dan hook kode pemenuhan secara independen untuk setiap maksud. Untuk informasi selengkapnya, lihat Mengintegrasikan AWS Lambda fungsi ke bot Amazon Lex V2 Anda.

Pengaturan granular

Amazon Lex V2 API memindahkan ambang batas skor keyakinan klasifikasi suara dan maksud dari bot ke cakupan bahasa. Bendera analisis sentimen bergerak dari cakupan bot ke lingkup alias. Waktu sesi habis dan pengaturan privasi di cakupan bot, dan log percakapan di lingkup alias, tetap tidak berubah.

Maksud fallback default

Amazon Lex V2 API menambahkan intent fallback default saat Anda membuat bahasa. Gunakan untuk mengonfigurasi penanganan kesalahan untuk bot Anda alih-alih permintaan penanganan kesalahan tertentu.

Pembaruan variabel sesi yang dioptimalkan

Dengan Amazon Lex V2 API Anda dapat memperbarui status sesi secara langsung dengan RecognizeTextdan RecognizeUtteranceoperasi tanpa ketergantungan pada sesi APIs.

Membuat sumber daya Amazon Lex V2 dengan AWS CloudFormation

Amazon Lex V2 terintegrasi dengan AWS CloudFormation, layanan yang membantu Anda memodelkan dan menyiapkan sumber daya AWS sehingga Anda dapat menghabiskan lebih sedikit waktu untuk membuat dan mengelola sumber daya dan infrastruktur Anda. Anda membuat templat yang menjelaskan semua sumber daya AWS yang Anda inginkan (seperti chatbots Amazon Lex V2), serta menyediakan serta CloudFormation mengonfigurasi sumber daya tersebut untuk Anda.

Bila Anda menggunakan CloudFormation, Anda dapat menggunakan kembali template Anda untuk mengatur sumber daya Amazon Lex V2 Anda secara konsisten dan berulang kali. Jelaskan sumber daya Anda sekali, lalu sediakan sumber daya yang sama berulang-ulang di beberapa Akun AWS dan Wilayah.

Amazon Lex V2 dan CloudFormation template

Untuk menyediakan dan mengonfigurasi sumber daya untuk Amazon Lex V2 dan layanan terkait, Anda harus memahami CloudFormation templat. Templat adalah file teks dengan format JSON atau YAML. Templat ini menjelaskan sumber daya yang ingin Anda sediakan di tumpukan CloudFormation . Jika Anda tidak terbiasa dengan JSON atau YAMAL, Anda dapat menggunakan CloudFormation Designer untuk membantu Anda memulai dengan template. CloudFormation Untuk informasi lebih lanjut, lihat Apa itu CloudFormation Desainer? dalam AWS CloudFormation User Guide.

Amazon Lex V2 mendukung pembuatan sumber daya berikut di CloudFormation:

- AWS::Lex::Bot
- AWS::Lex::BotAlias
- AWS::Lex::BotVersion
- AWS::Lex::ResourcePolicy

Untuk informasi selengkapnya, termasuk contoh templat JSON dan YAMAL untuk sumber daya ini, lihat referensi jenis sumber daya Amazon Lex V2 di AWS CloudFormation Panduan Pengguna.

Pelajari lebih lanjut tentang CloudFormation

Untuk mempelajari selengkapnya CloudFormation, lihat sumber daya berikut:

- AWS CloudFormation
- AWS CloudFormation panduan pengguna
- CloudFormation Referensi API
- AWS CloudFormation Panduan pengguna antarmuka baris perintah

Riwayat dokumen untuk Amazon Lex V2

- Pembaruan dokumentasi terbaru: 27 Oktober 2025

Tabel berikut menjelaskan perubahan penting dalam setiap rilis Amazon Lex V2. Untuk notifikasi tentang pembaruan dokumentasi ini, Anda dapat berlangganan ke umpan RSS.

Perubahan	Deskripsi	Tanggal
Pembaruan fitur	Tunggu dan lanjutkan sekarang tersedia di lokal tambahan: Ca-es, De-at, De-de, es-419, es-ES, es-US, fr-Ca, fr-FR, IT-it, Ja-jp, Ko-kr, Pt-BR, Pt-pt, Zh-CN, Zh-HK. Untuk informasi selengkapnya, lihat Mengaktifkan bot Amazon Lex V2 untuk menunggu pengguna memberikan informasi selengkapnya selama jeda.	Oktober 27, 2025
Pembaruan fitur	Amazon.currency dan Amazon.Confirmation jenis slot bawaan sekarang mendukung semua lokal. Untuk informasi selengkapnya, lihat Jenis slot bawaan.	Agustus 18, 2025
Pembaruan dokumentasi	Dokumentasi yang diperbarui untuk fitur NLU Berbantuan dengan contoh dan klarifikasi yang ditingkatkan. Untuk informasi lebih lanjut, lihat NLU Bantuan.	Agustus 4, 2025

Fitur baru	Anda dapat menggunakan Amazon. BedrockAgentIntent niat bawaan untuk menghubungkan bot Amazon Lex V2 Anda dengan Agen Bedrock Amazon dan Pangkalan Pengetahuan. Untuk informasi selengkapnya, lihat AMAZON.BedrockAgentIntent niat bawaan.	Juli 3, 2025
Memperbarui ke kebijakan terkelola AWS	Amazon Lex V2 memperbarui kebijakan untuk mengizinkan replikasi tag dan ResourceBasedPolicy saat menggunakan ketahanan global lihat. AmazonLexReplicationPolicy	Juni 24, 2025
Fitur baru	Anda sekarang dapat menggunakan Assisted NLU untuk meningkatkan klasifikasi maksud dan resolusi slot di Amazon Lex V2. Fitur ini menggunakan Large Language Models (LLMs) untuk meningkatkan akurasi sambil tetap berada dalam intent dan slot yang dikonfigurasi bot Anda. Untuk informasi lebih lanjut, lihat NLU Bantuan.	Juni 11, 2025
Perbarui ke fitur (2025-06-03)	Anda dapat menggunakan 17 bahasa baru dengan kosakata kustom Anda di Amazon Lex V2. Untuk informasi selengkapnya, lihat Kosakata khusus.	Juni 3, 2025

Fitur baru (2024-12-01)	Anda dapat menggunakan intent QinConnect bawaan untuk menghubungkan bot Amazon Lex V2 Anda dengan Amazon Connect. Untuk informasi selengkapnya, lihat maksud QinConnect bawaan.	Desember 1, 2024
Fitur baru (2024-10-01)	Anda dapat menggunakan Amazon.currency built-in slot untuk lokal AS dan Spanyol di Amazon Lex V2. Untuk informasi selengkapnya, lihat Amazon.currency.	Oktober 1, 2024
Perbarui ke fitur (2024-07-26)	Maksud bawaan QnA untuk AI generatif sekarang mendukung Basis Pengetahuan Batuan Dasar dan Pagar Pembatas di Amazon Lex V2. Untuk informasi lebih lanjut, lihat AIntentAmazon.qn.	Juli 26, 2024
Perbarui ke fitur (2024-07-10)	Maksud bawaan QnA untuk AI generatif sekarang mendukung model Anthropic Claude 3 Haiku dan Anthropic Claude 3 Sonnet di Amazon Lex V2. Untuk informasi selengkapnya, lihat Optimalkan pembuatan dan kinerja bot dengan AI generatif.	Juli 10, 2024

Pembaruan ke kebijakan terkelola AWS (2024-05-10)	Amazon Lex V2 menambahkan izin baru ke kebijakan AmazonLexReadOnlyterkelola untuk memungkinkan akses baca ke sumber daya bot yang telah direplikasi di wilayah lain.	10 Mei 2024
Pembaruan ke kebijakan terkelola AWS (2024-04-15)	Amazon Lex V2 menambahkan izin baru ke kebijakan AmazonLexFullAccesterkelola untuk memungkinkan pembaruan sumber daya bot yang direplikasi ke wilayah lain.	April 15, 2024
Perluasan wilayah	Amazon Lex V2 sekarang tersedia di AWS GovCloud (AS-Barat) (us-gov-west-1).	Maret 22, 2024
Pembaruan ke kebijakan terkelola AWS (2024-03-07)	Amazon Lex V2 menambahkan izin baru ke kebijakan AmazonLexReplicationPolicyterkelola untuk memungkinkan pembaruan sumber daya bot yang direplikasi ke wilayah lain.	Maret 7, 2024
Fungsi baru	Anda dapat menggunakan fn.length () fungsi untuk menentukan panjang nilai dari nilai string di Amazon Lex V2. Untuk informasi selengkapnya, lihat Percabangan bersyarat - Fungsi.	Maret 4, 2024

Perbarui ke fitur (2024-02-28)	Slot built-in QnA untuk kemampuan AI generatif di Amazon Lex V2 sekarang GA. Untuk informasi selengkapnya, lihat Optimalkan pembuatan dan kinerja bot dengan AI generatif.	Februari 28, 2024
Pembaruan ke kebijakan terkelola AWS (2024-02-08)	Amazon Lex V2 menambahkan izin baru ke kebijakan AmazonLexFullAccesterkelola untuk memungkinkan replikasi sumber daya bot ke wilayah lain.	Februari 8, 2024
Kebijakan terkelola baru	Amazon Lex V2 menambahkan kebijakan terkelola yang menyediakan izin untuk mereplikasi sumber daya bot di wilayah lain. Untuk informasi selengkapnya, lihat AmazonLexReplicationPolicy.	Februari 8, 2024
Fitur baru (2024-02-08)	Anda dapat menggunakan ketahanan Global untuk mereplikasi bot Anda di wilayah AWS kedua di Amazon Lex V2. Untuk informasi lebih lanjut, lihat Ketahanan global.	Februari 8, 2024

Fitur baru (2023-11-29)	Anda sekarang dapat memanfaatkan kemampuan AI generatif di Amazon Lex V2. Untuk informasi selengkapnya, lihat Optimalkan pembuatan dan kinerja bot dengan AI generatif.	November 29, 2023
Fitur baru (2023-11-08)	Amazon Lex V2 sekarang dapat menggunakan logging selektif untuk menangkap and/or audio teks pada level intent atau slot. Untuk informasi selengkapnya, lihat Pencatatan selektif.	8 November 2023
Fitur baru (2023-08-17)	Amazon Lex V2 sekarang dapat menggunakan slot bawaan untuk menentukan tanggapan yang Yes/No/Maybe/Don tidak tahu menggunakan`AMAZON.Confirmation` . Untuk informasi selengkapnya, lihat Jenis Slot Bawaan.	17 Agustus 2023
Fitur baru (2023-07-18)	Anda dapat melihat metrik kinerja maksud dan slot, selain metrik percakapan lainnya menggunakan dasbor analitik. Untuk informasi selengkapnya, lihat Analytics.	Juli 18, 2023

Fitur baru (2023-06-06)	Anda dapat meningkatkan akurasi dan keberhasilan pemenuhan bot Anda dengan Test Workbench. Untuk informasi selengkapnya, lihat Meja Kerja Uji.	6 Juni 2023
Fitur baru (2023-02-23)	Anda sekarang dapat membuat bot dari template untuk beberapa vertikal bisnis populer. Untuk informasi selengkapnya, lihat Templat bot.	23 Februari 2023
Fitur baru (2023-02-09)	Anda sekarang dapat menggabungkan beberapa bot ke dalam jaringan bot untuk menciptakan pengalaman pelanggan yang terintegrasi. Untuk informasi selengkapnya, lihat Jaringan bot.	9 Februari 2023
Fitur baru (2022-12-06)	Amazon Lex V2 sekarang mendukung bahasa Arab Teluk (Uni Emirat Arab), Kanton (Hong Kong), Finlandia (Finlandia), Norwegia (Norwegia), Polandia (Polandia), dan Swedia (Swedia). Untuk informasi selengkapnya, lihat Bahasa dan lokal yang didukung oleh Amazon Lex V2.	6 Desember 2022

Diperbarui ke kebijakan terkelola AWS	Amazon Lex V2 menambahkan izin baru ke kebijakan AmazonLexReadOnlyterkelola untuk memungkinkan tampilan item kosakata khusus.	29 November 2022
Fitur baru (2022-11-07)	Amazon Lex V2 dapat menampilkan representasi alternatif untuk frasa atau kata dengan menggunakan konsol atau APIs untuk menyesuaikan ucapan ke output teks. Untuk informasi selengkapnya, lihat Membuat kosakata khusus untuk pengenalan suara.	7 November 2022
Fitur baru (2022-10-28)	Amazon Lex V2 dapat menambahkan atribut bobot ke elemen item yang akan mewakili sejauh mana frasa ditingkatkan selama pengenalan suara. Untuk informasi lebih lanjut, lihat Bobot Tata Bahasa.	28 Oktober 2022
Fitur baru (2022-10-19) (2022-10-19) - 2	Amazon Lex V2 dapat menampilkan representasi alternatif untuk frasa atau kata dalam pidato akhir untuk output teks. Untuk informasi selengkapnya, lihat Membuat kosakata khusus untuk pengenalan suara.	Oktober 19, 2022

Fitur baru (2022-10-19) (2022-10-19)	Amazon Lex V2 dapat digunakan untuk menangkap input formulir gratis dari pengguna akhir yang terdiri dari kata-kata atau karakter yang digunakan`AMAZON.FreeFormInput` . Untuk informasi selengkapnya, lihat Jenis Slot Bawaan.	Oktober 19, 2022
Fitur baru (2022-10-14)	Amazon Lex V2 sekarang mendukung lokal Hindi (India) dan Belanda (Belanda). Untuk informasi selengkapnya, lihat Bahasa dan lokal yang didukung oleh Amazon Lex V2.	14 Oktober 2022
Fitur baru (2022-09-22)	Ada pembaruan cara Amazon Lex V2 mengelola input pengguna. Sekarang Anda dapat memilih apakah Amazon Lex V2 menerima input teks, audio, atau DTMF kapan saja dalam alur percakapan. Untuk informasi selengkapnya, lihat Atribut yang Dapat Dikonfigurasi.	September 22, 2022

Fitur baru (2022-09-14)	Ada pembaruan cara Amazon Lex V2 mengelola alur percakapan. Pembuat percakapan visual adalah pembuat percakapan seret dan lepas untuk merancang dan memvisualisasikan jalur percakapan dengan mudah. Untuk informasi selengkapnya, lihat Pembuat Percakapan Visual.	14 September 2022
Fitur baru (2022-09-09)	Ada pembaruan cara Amazon Lex V2 membangun slot yang kompleks. Anda sekarang dapat membuat subslot kompleks dalam slot untuk mengelola maksud dalam desain percakapan yang kompleks. Untuk informasi lebih lanjut, lihat Membangun slot komposit.	9 September 2022
Fitur baru (2022-08-17)	Ada pembaruan cara Amazon Lex V2 mengelola aliran jalur percakapan dengan pengguna Anda. Anda sekarang dapat membuat jalur percakapan yang kompleks dengan memesan langkah berikutnya dalam percakapan. Untuk informasi selengkapnya, lihat Membuat jalur percakapan.	17 Agustus 2022

Fitur baru (2022-07-05)	Ada pembaruan cara Amazon Lex V2 mengelola aliran percakapan dengan pengguna Anda. Anda sekarang dapat membuat percakapan yang kompleks dengan memesan petunjuknya. Untuk informasi selengkapnya, lihat Mengonfigurasi prompt.	Juli 5, 2022
Fitur baru (2022-05-03)	Ada pembaruan cara Amazon Lex V2 mengelola aliran percakapan dengan pengguna Anda. Anda sekarang dapat membuat percakapan yang kompleks menggunakan kondisi. Untuk informasi selengkapnya, lihat Memahami alur percakapan baru.	Mei 3, 2022
Fitur baru (2022-03-22)	Ditambahkan contoh tata bahasa industri untuk tipe slot tata bahasa bawaan. Untuk informasi lebih lanjut, lihat Tata bahasa industri.	22 Maret 2022
Fitur baru (2022-03-18)	Menambahkan dokumentasi tentang mengintegrasikan Amazon Lex V2 dengan Amazon Chime SDK. Untuk informasi selengkapnya, lihat Amazon Chime SDK.	18 Maret 2022

Fitur baru (2022-01-27)	Amazon Lex V2 sekarang memberikan skor kepercaya an untuk transkripsi suara. Gunakan skor untuk membantu menentukan respons yang benar dari pengguna. Untuk informasi lebih lanjut, lihat Menggunak an skor kepercayaan transkrip si suara.	27 Januari 2022
Fitur baru (2022-01-13)	Anda sekarang dapat menambahkan petunjuk kontekstual dan dinamis ke slot untuk meningkatkan akurasi bot Anda. Untuk informasi selengkapnya, lihat Menggunakan petunjuk untuk meningkatkan akurasi.	Januari 13, 2022
Fitur baru (2022-01-12)	Amazon Lex V2 menambahk an dukungan untuk kosakata khusus untuk meningkat kan pengenalan suara untuk input audio. Untuk informasi selengkapnya, lihat Membuat kosakata khusus untuk meningkatkan pengenalan suara.	12 Januari 2022
Fitur baru (2022-01-07)	Amazon Lex V2 sekarang mendukungAWS PrivateLink. Untuk informasi selengkapnya, lihat titik akhir VPC (AWS). PrivateLink	7 Januari 2022

Fitur baru (2022-01-03)	Amazon Lex V2 sekarang mendukung lokal Catalan (Spanyol). Untuk informasi selengkapnya, lihat Bahasa dan lokal yang didukung oleh Amazon Lex V2.	Januari 3, 2022
Fitur baru (2021-12-20) (2021-12-20) - 2	AWS CloudFormationsekarang mendukung Amazon Lex V2. Untuk informasi selengkapnya, lihat AWS CloudFormationsumber daya.	Desember 20, 2021
Fitur baru (2021-12-20) (2021-12-20)	Anda sekarang dapat membuat jenis slot menggunakan tata bahasa khusus Anda sendiri. Untuk informasi selengkapnya, lihat Menggunakan jenis slot tata bahasa khusus.	Desember 20, 2021
Fitur baru (2021-12-16)	Amazon Lex V2 sekarang mendukung bahasa Portugis (Brasil), Portugis (Portugal), dan Mandarin (RRC). Untuk informasi selengkapnya, lihat Bahasa dan lokal yang didukung oleh Amazon Lex V2.	Desember 16, 2021

Fitur baru (2021-12-01)	Amazon Lex V2 sekarang menyediakan pratinjau Desainer Chatbot Otomatis untuk membantu Anda mulai membuat chatbot dari transkrip pusat kontak. Untuk informasi selengkapnya, lihat Menggunakan Perancang Chatbot Otomatis (Pratinjau).	1 Desember 2021
Fitur baru (2021-11-19) (2021-11-19) - 2	Anda sekarang dapat menggunakan suara Amazon Polly neural text to speech (NTTS) untuk percakapan audio dengan pengguna Anda. Untuk informasi selengkapnya, lihat Suara di Amazon Polly.	November 19, 2021
Fitur baru (2021-11-19) (2021-11-19)	Anda sekarang dapat menggunakan spell-by-letter dan spell-by-word gaya untuk memasukkan nilai slot yang sulit dipahami oleh Amazon Lex V2. Untuk informasi selengkapnya, lihat Menggunakan gaya ejaan untuk menangkap nilai slot.	November 19, 2021
Fitur baru (2021-11-09)	Amazon Lex V2 sekarang mendukung bahasa Inggris (Afrika Selatan) lokal. Untuk informasi selengkapnya, lihat Bahasa dan lokal yang didukung oleh Amazon Lex V2.	November 9, 2021

Fitur baru (2021-11-05)	Amazon Lex V2 sekarang mendukung lokal Jerman (Austria). Untuk informasi selengkapnya, lihat Bahasa dan lokal yang didukung oleh Amazon Lex V2.	November 5, 2021
Fitur baru (2021-10-07)	Anda sekarang dapat memberi pengguna pesan pembaruan yang diputar di awal fungsi pemenuhan dan secara berkala saat fungsi berjalan. Anda juga dapat membuat pesan yang memberi tahu pengguna tentang status pemenuhan saat fungsi selesai. Untuk informasi selengkapnya, lihat Mengonfigurasi pembaruan kemajuan pemenuhan.	7 Oktober 2021
Perluasan wilayah (2021-09-22)	Amazon Lex V2 sekarang tersedia di Afrika (Cape Town) (af-south-1) dan Asia Pasifik (Seoul) (ap-northeast-2).	22 September 2021
Fitur baru (2021-09-22)	Anda sekarang dapat melihat statistik untuk ucapan yang dikirim pengguna Anda ke bot Anda. Untuk informasi selengkapnya, lihat Melihat statistik ucapan.	22 September 2021

Fitur baru (2021-09-09)	Amazon Lex V2 sekarang mendukung lokal Korea (Korea). Untuk informasi selengkapnya, lihat Bahasa dan lokal yang didukung oleh Amazon Lex V2.	9 September 2021
Fitur baru (2021-07-27)	Amazon Lex V2 sekarang menyediakan tipe slot built-in untuk kode pos Inggris. Untuk informasi selengkapnya, lihat AMAZON. UKPostalKode.	27 Juli 2021
Fitur baru (2021-07-15)	Amazon Lex V2 sekarang mendukung lokal Inggris (India). Untuk informasi selengkapnya, lihat Bahasa dan lokal yang didukung oleh Amazon Lex V2.	15 Juli 2021
Fitur baru (2021-07-13)	Amazon Lex V2 sekarang menyediakan alat untuk memigrasikan bot dari Amazon Lex V1 ke Amazon Lex V2 API. Untuk informasi selengkapnya, lihat Memigrasi bot di Panduan Pengembang Amazon Lex.	13 Juli 2021

Fitur baru (2021-06-26)	Anda sekarang dapat mengonfigurasi bot Amazon Lex V2 Anda untuk mendapatkan kembali slot yang telah diisi dengan mengatur nilai slot ke null dan mengonfigurasi alur percakapan untuk memutar kembali. Fitur ini memungkinkan percakapan yang lebih dinamis berdasarkan tanggapan pelanggan. Untuk informasi lebih lanjut, lihat Memunculkan kembali slot.	Juni 26, 2021
Dokumentasi baru	Amazon Lex V2 sekarang menyediakan panduan komprehensif dan praktik terbaik untuk pengembangan bot untuk membantu Anda menciptakan pengalaman percakapan yang lebih efektif. Untuk informasi selengkapnya, lihat Pedoman pengembangan bot Amazon Lex V2.	Juni 26, 2021
Fitur baru (2021-06-15)	Amazon Lex V2 sekarang memungkinkan Anda untuk menerima beberapa nilai untuk satu slot dalam bahasa Inggris (AS). Untuk informasi selengkapnya, lihat Menggunakan beberapa nilai dalam slot.	15 Juni 2021

Fitur baru (2021-06-11)	Anda sekarang dapat membangun bot yang lebih besar untuk bahasa Inggris. Untuk informasi lebih lanjut, lihat Kuota .	11 Juni 2021
Fitur baru (2021-05-20)	Gunakan kebijakan berbasis sumber daya Amazon Lex V2 untuk mengelola akses ke bot dan alias bot Anda. Untuk informasi selengkapnya, lihat Kebijakan berbasis sumber daya dalam Amazon Lex V2.	20 Mei 2021
Fitur baru (2021-05-18)	Amazon Lex V2 sekarang memungkinkan Anda untuk mengimpor dan mengekspor bot dan lokal bot. Anda dapat menggunakan fitur ini untuk menyalin bot dan lokal bot antara akun dan Wilayah AWS. Untuk informasi selengkapnya, lihat Mengimpor dan mengekspor.	18 Mei 2021
Perluasan wilayah (2021-05-17)	Amazon Lex V2 sekarang tersedia di Kanada (Tengah) (ca-central-1).	17 Mei 2021
Fitur baru (2021-04-01)	Amazon Lex V2 sekarang mendukung lokal Jepang (Jepang). Untuk informasi selengkapnya, lihat Bahasa dan lokal yang didukung oleh Amazon Lex V2.	1 April 2021

Fitur baru (2021-03-12)	Amazon Lex V2 sekarang mendukung tiga jenis slot bawaan baru:`AMAZON.City` ,`AMAZON.Country` , dan`AMAZON.State` .	12 Maret 2021
Panduan baru	Ini adalah rilis pertama dari panduan pengguna Amazon Lex V2.	21 Januari 2021

Referensi API

Referensi API sekarang menjadi dokumen terpisah.

AWS Glosarium

Untuk AWS terminologi terbaru, lihat AWS glosarium di Referensi.Glosarium AWS

Terjemahan disediakan oleh mesin penerjemah. Jika konten terjemahan yang diberikan bertentangan dengan versi bahasa Inggris aslinya, utamakan versi bahasa Inggris.