aws

Panduan Developer

Amazon Lex V1

Amazon Lex V1: Panduan Developer

Table of Contents

Jika Anda menggunakan Amazon Lex V2, lihat panduan Amazon Lex V2 sebagai gantinya.

Jika Anda menggunakan Amazon Lex V1, kami sarankan untuk meningkatkan bot Anda ke Amazon Lex V2. Kami tidak lagi menambahkan fitur baru ke V1 dan sangat menyarankan menggunakan V2 untuk semua bot baru.

Terjemahan disediakan oleh mesin penerjemah. Jika konten terjemahan yang diberikan bertentangan dengan versi bahasa Inggris aslinya, utamakan versi bahasa Inggris.

Apa itu Amazon Lex?

Amazon Lex adalah layanan AWS untuk membangun antarmuka percakapan untuk aplikasi menggunakan suara dan teks. Dengan Amazon Lex, mesin percakapan yang sama dengan kekuatan Amazon Alexa sekarang tersedia untuk pengembang mana pun, memungkinkan Anda untuk membangun chatbots bahasa alami yang canggih ke dalam aplikasi baru dan yang sudah ada. Amazon Lex 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 mirip manusia hidup, dan menciptakan kategori produk baru.

Amazon Lex memungkinkan pengembang untuk membangun chatbots percakapan dengan cepat. Dengan Amazon Lex, tidak diperlukan keahlian deep learning — untuk membuat bot, Anda cukup menentukan alur percakapan dasar di konsol Amazon Lex. Amazon Lex 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 bot pada perangkat seluler, aplikasi web, dan platform obrolan (misalnya, Facebook Messenger).

Amazon Lex menyediakan integrasi pra-bangun denganAWS Lambda, dan Anda dapat dengan mudah berintegrasi dengan banyak layanan lain di platform AWS, termasuk Amazon Cognito,AWS Mobile Hub, Amazon CloudWatch, dan Amazon DynamoDB. Integrasi dengan Lambda menyediakan akses bot ke konektor perusahaan tanpa server yang telah dibangun sebelumnya untuk menautkan ke data dalam aplikasi SaaS, seperti Salesforce, HubSpot, atau Marketo.

Beberapa manfaat menggunakan Amazon Lex meliputi:

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

- Teknologi pembelajaran mendalam yang didemokratisasi— Didukung oleh teknologi yang sama dengan Alexa, Amazon Lex menyediakan teknologi ASR dan NLU untuk menciptakan sistem

Speech Language Understanding (SLU). Melalui SLU, Amazon Lex mengambil ucapan bahasa alami dan input teks, memahami maksud di balik input, dan memenuhi maksud pengguna dengan menerapkan fungsi bisnis yang sesuai.

Pengenalan ucapan dan pemahaman bahasa alami adalah beberapa masalah paling menantang untuk dipecahkan dalam ilmu komputer, membutuhkan algoritma pembelajaran mendalam yang canggih untuk dilatih pada sejumlah besar data dan infrastruktur. Amazon Lex menempatkan teknologi deep learning dalam jangkauan semua pengembang, didukung oleh teknologi yang sama dengan Alexa. Chatbots Amazon Lex mengkonversi pidato masuk ke teks dan memahami maksud pengguna untuk menghasilkan respons cerdas, sehingga Anda dapat fokus pada membangun bot Anda dengan nilai tambah yang berbeda untuk pelanggan Anda, untuk menentukan kategori produk yang sama sekali baru yang dimungkinkan melalui antarmuka percakapan.

- Penyebaran dan penskalaan yang mulus— Dengan Amazon Lex, Anda dapat membangun, menguji, dan menyebar chatbots Anda langsung dari Amazon Lex konsol. Amazon Lex memungkinkan Anda untuk dengan mudah mempublikasikan chatbots suara atau teks untuk digunakan pada perangkat seluler, aplikasi web, dan layanan obrolan (misalnya, Facebook Messenger). Amazon Lex menskalakan secara otomatis sehingga Anda tidak perlu khawatir tentang penyediaan perangkat keras dan mengelola infrastruktur untuk meningkatkan pengalaman bot Anda.

- Integrasi bawaan dengan platform AWS— Amazon Lex memiliki interoperabilitas asli dengan layanan AWS lainnya, seperti Amazon Cognito,AWS Lambda, Amazon CloudWatch, danAWS Mobile Hub. Anda dapat memanfaatkan kekuatan platform AWS untuk keamanan, pemantauan, otentikasi pengguna, logika bisnis, penyimpanan, dan pengembangan aplikasi seluler.

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

Apakah Anda Pengguna Amazon Lex?

Jika Anda baru pertama kali menggunakan Amazon Lex, kami menyarankan agar Anda membaca bagian-bagian berikut secara berurutan:

1. Memulai dengan Amazon Lex— Pada bagian ini, Anda mengatur akun Anda dan menguji Amazon Lex.
2. Referensi API — Bagian ini menyediakan contoh tambahan yang dapat Anda gunakan untuk menjelajahi Amazon Lex.

Amazon Lex: Cara Kerjanya

Amazon Lex memungkinkan Anda membuat aplikasi menggunakan antarmuka ucapan atau teks yang didukung oleh teknologi yang sama yang mendukung Amazon Alexa. Berikut ini adalah langkah-langkah khas yang Anda lakukan saat bekerja dengan Amazon Lex:

1. Buat bot dan konfigurasikan dengan satu atau lebih maksud yang ingin Anda dukung. Konfigurasikan bot agar dapat memahami tujuan pengguna (intent), terlibat dalam percakapan dengan pengguna untuk mendapatkan informasi, dan memenuhi maksud pengguna.
2. Uji bot. Anda dapat menggunakan klien jendela pengujian yang disediakan oleh konsol Amazon Lex.
3. Terbitkan versi dan membuat alias.
4. Terapkan bot yang terdapat bot. Anda dapat menggunakan bot pada platform seperti aplikasi seluler atau platform perpesanan seperti Facebook Messenger.

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

- Bot - Bot melakukan tugas otomatis seperti memesan pizza, memesan hotel, memesan bunga, dan sebagainya. Bot Amazon Lex didukung oleh kemampuan Automatic Speech Recognition (ASR) dan Natural Language Understanding (NLU). Setiap bot harus memiliki nama unik dalam akun Anda.

  Bot Amazon Lex dapat memahami input pengguna yang disediakan dengan teks atau ucapan dan berkomunikasi dalam bahasa alami. Anda dapat membuat fungsi Lambda dan menambahkannya sebagai pengait kode dalam konfigurasi maksud Anda untuk melakukan validasi data pengguna dan tugas pemenuhan.

- Maksud - Maksud mewakili tindakan yang ingin dilakukan pengguna. Anda membuat bot untuk mendukung satu atau lebih maksud terkait. Misalnya, Anda dapat membuat bot yang memesan pizza dan minuman. Untuk setiap maksud, Anda memberikan informasi berikut:

- Nama maksud - Nama deskriptif untuk maksud. Sebagai contoh, **`OrderPizza`**. Nama maksud harus unik dalam akun Anda.
- Contoh ucapan - Bagaimana pengguna dapat menyampaikan maksud. 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 (misalnya, memesan dengan toko pizza lokal). Kami merekomendasikan Anda membuat fungsi Lambda untuk memenuhi maksud.

  Anda opsional dapat mengkonfigurasi maksud sehingga Amazon Lex hanya mengembalikan informasi kembali ke aplikasi klien untuk melakukan pemenuhan yang diperlukan.

  Selain maksud khusus seperti memesan pizza, Amazon Lex juga menyediakan maksud bawaan untuk mengatur bot Anda dengan cepat. Untuk informasi selengkapnya, lihat Intent dan Jenis Slot Bawaan.

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

  Misalnya,`OrderPizza` maksud membutuhkan slot seperti ukuran pizza, jenis kerak, dan jumlah pizza. Dalam konfigurasi maksud, Anda menambahkan slot ini. Untuk setiap slot, Anda memberikan jenis slot dan prompt untuk Amazon Lex untuk mengirim ke klien untuk mendapatkan data dari pengguna. Seorang pengguna dapat membalas dengan nilai slot yang mencakup kata-kata tambahan, seperti “pizza besar silakan” atau “mari kita tetap dengan kecil.” Amazon Lex masih bisa memahami nilai slot yang dimaksud.

- Jenis slot - Setiap slot memiliki tipe. Anda dapat membuat jenis slot khusus Anda atau menggunakan jenis slot bawaan. Setiap jenis slot harus memiliki nama unik dalam akun Anda.

Misalnya, Anda dapat membuat dan menggunakan jenis slot berikut untuk`OrderPizza` maksud tersebut:

- Ukuran - Dengan nilai pencacahan`Small`,`Medium`, dan`Large`.
- Kerak - Dengan nilai pencacahan`Thick` dan`Thin`.

Amazon Lex juga menyediakan built-in jenis slot. Misalnya,`AMAZON.NUMBER` adalah tipe slot built-in yang dapat Anda gunakan untuk jumlah pizza yang dipesan. Untuk informasi selengkapnya, lihat Intent dan Jenis Slot Bawaan.

Untuk daftar Wilayah AWS tempat Amazon Lex terdapat Wilayah dan Titik Akhir AWS dalam Referensi Umum Amazon Web Services.

Topik berikut memberikan informasi tambahan. Kami menyarankan Anda memeriksanya secara berurutan dan kemudian menjelajahiMemulai dengan Amazon Lex latihannya.

Topik

- Bahasa yang Didukung di Amazon Lex
- Model Pemrograman
- Mengelola Pesan
- Mengelola Konteks Percakapan
- Menggunakan Skor Keyakinan
- Log Percakapan
- Mengelola Sesi Dengan Amazon Lex API
- Opsi Deployment bot Opsi Deployment Bot
- Intent dan Jenis Slot Bawaan
- Jenis Slot Kustom Slot Jenis Slot Khusus
- Obfuscation Slot
- Analisis Sentimen
- Menandai Sumber Daya Amazon Lex

Bahasa yang Didukung di Amazon Lex

Amazon Lex V1 mendukung berbagai bahasa dan lokal. Bahasa yang didukung dan fitur yang mendukungnya tercantum dalam tabel berikut.

Amazon Lex V2 mendukung bahasa tambahan, lihat Bahasa yang Didukung di Amazon Lex V2

Bahasa dan Lokal yang Didukung

Amazon Lex V1 mendukung bahasa dan lokal berikut.

Code	Bahasa dan lokal
De-de	Jerman (Jerman)
id-ID	Inggris (Australia)
ID-ID	Bahasa Inggris (UK)
ID-in	Inggris (India)
id-ID	Bahasa Inggris (AS)
es-419	Spanyol (Amerika Latin)
ES	Spanyol (Spanyol)
ES-AS	Spanyol (AS)
FR-CA	Bahasa Prancis (Kanada)
FR-FR	Perancis (Perancis)
itu-itu	Italia (Italia)
J-JP	Jepang (Japan)
Ko	Korea (Korea)

Bahasa dan Lokal yang Didukung oleh Fitur Amazon Lex

Semua fitur Amazon Lex didukung dalam semua bahasa dan lokal kecuali seperti yang tercantum dalam tabel ini.

Fitur	Bahasa dan lokal yang didukung
Konteks Maksud Konteks Maksud Maksud	Bahasa Inggris (ID) (id-ID)

Model Pemrograman

Bot adalah jenis sumber daya utama di Amazon Lex. Jenis sumber daya lain di Amazon Lex adalah maksud, jenis slot, alias, dan asosiasi saluran bot.

Anda membuat bot menggunakan konsol Amazon Lex atau API pembuatan model. Konsol menyediakan antarmuka pengguna grafis yang Anda gunakan untuk membuat bot siap produksi untuk aplikasi Anda. Jika mau, Anda dapat menggunakan API pembuatan model melaluiAWS CLI atau program khusus Anda sendiri untuk membuat bot.

Setelah Anda membuat bot, Anda menerapkannya di salah satu platform yang didukung atau mengintegrasikannya ke dalam aplikasi Anda sendiri. Saat pengguna berinteraksi dengan bot, aplikasi klien mengirimkan permintaan ke bot menggunakan API runtime Amazon Lex. Misalnya, ketika pengguna mengatakan “Saya ingin memesan pizza,” klien Anda mengirimkan masukan ini ke Amazon Lex menggunakan salah satu operasi API runtime. Pengguna dapat memberikan masukan sebagai ucapan atau teks.

Anda juga dapat membuat fungsi Lambda dan menggunakannya dalam intent. Gunakan kait kode fungsi Lambda ini untuk melakukan aktivitas runtime seperti inisialisasi, validasi input pengguna, dan pemenuhan maksud. Bagian berikut memberikan informasi tambahan.

Topik

- Operasi API Membangun Model Membangun Operasi API Bangunan Model
- Operasi API Waktu Aktif Waktu Aktif Ter
- Fungsi Lambda Sebagai Pengait Kode

Operasi API Membangun Model Membangun Operasi API Bangunan Model

Untuk membuat bot, intent, dan tipe slot secara terprogram, gunakan operasi API pembuatan model. Anda juga dapat menggunakan API pembuatan model untuk mengelola, memperbarui, dan menghapus sumber daya untuk bot Anda. Operasi API pembuatan model meliputi:

- PutBot,PutBotAlias,PutIntent, danPutSlotType untuk membuat dan memperbarui bot, alias bot, maksud, dan jenis slot, masing-masing.
- CreateBotVersion,CreateIntentVersion, danCreateSlotTypeVersion untuk membuat dan mempublikasikan versi bot, maksud, dan jenis slot Anda, masing-masing.
- GetBotdanGetBots untuk mendapatkan bot tertentu atau daftar bot yang telah Anda buat, masing-masing.
- GetIntentdanGetIntents untuk mendapatkan maksud tertentu atau daftar maksud yang telah Anda buat, masing-masing.
- GetSlotTypedanGetSlotTypes untuk mendapatkan jenis slot tertentu atau daftar jenis slot yang telah Anda buat, masing-masing.
- GetBuiltinIntent,GetBuiltinIntents, danGetBuiltinSlotTypes untuk mendapatkan Amazon Lex built-in niat, daftar Amazon Lex built-in maksud, atau daftar built-in jenis slot yang dapat Anda gunakan dalam bot Anda, masing-masing.
- GetBotChannelAssociationdanGetBotChannelAssociations untuk mendapatkan hubungan antara bot Anda dan platform perpesanan atau daftar asosiasi antara bot dan platform perpesanan Anda.
- DeleteBot,DeleteBotAlias,DeleteBotChannelAssociation,DeleteIntent, danDeleteSlotType untuk menghapus sumber daya yang tidak dibutuhkan di akun Anda.

Anda dapat menggunakan API pembuatan model untuk membuat alat khusus untuk mengelola sumber daya Amazon Lex Anda. Misalnya, ada batas 100 versi masing-masing untuk bot, intent, dan jenis slot. Anda dapat menggunakan API pembuatan model untuk membuat alat yang secara otomatis menghapus versi lama saat bot Anda mendekati batas.

Untuk memastikan bahwa hanya satu operasi yang memperbarui sumber daya pada satu waktu, Amazon Lex menggunakan checksum. Saat Anda menggunakan operasi`Put` API —PutBot,PutBotAliasPutIntent, atauPutSlotType —untuk memperbarui sumber daya, Anda harus meneruskan checksum sumber daya saat ini dalam permintaan. Jika dua alat mencoba memperbarui sumber daya secara bersamaan, keduanya menyediakan checksum saat ini yang sama. Permintaan pertama untuk mencapai Amazon Lex cocok dengan checksum

sumber daya saat ini. Pada saat permintaan kedua tiba, checksum berbeda. Alat kedua menerimaPreconditionFailedException pengecualian dan pembaruan berakhir.

GetOperasi—GetBot,GetIntent, danGetSlotType - akhirnya konsisten. Jika Anda menggunakanGet operasi segera setelah Anda membuat atau memodifikasi sumber daya dengan salah satuPut operasi, perubahan mungkin tidak dikembalikan. SetelahGet operasi mengembalikan pembaruan terbaru, selalu mengembalikan sumber daya yang diperbarui sampai sumber daya diubah lagi. Anda dapat menentukan apakah sumber daya yang diperbarui telah dikembalikan dengan melihat checksum.

Operasi API Waktu Aktif Waktu Aktif Ter

Aplikasi klien menggunakan operasi API runtime berikut untuk berkomunikasi dengan Amazon Lex:

- PostContent- Membawa ucapan atau input teks dan mengembalikan informasi maksud dan teks atau pesan ucapan untuk disampaikan kepada pengguna. Saat ini, Amazon Lex mendukung format audio berikut:

  Format audio masukan — LPCM dan Opus

  Format audio keluaran - MPEG, OGG, dan PCM

  PostContentOperasi ini mendukung input audio pada 8 kHz dan 16 kHz. Aplikasi di mana pengguna akhir berbicara dengan Amazon Lex melalui telepon, seperti pusat panggilan otomatis, dapat melewati audio 8 kHz secara langsung.

- PostText- Membawa teks sebagai input dan mengembalikan informasi maksud dan pesan teks untuk disampaikan kepada pengguna.

Aplikasi klien Anda menggunakan API waktu proses untuk memanggil bot Amazon Lex tertentu untuk memproses ucapan — teks pengguna atau input suara. Misalnya, misalkan pengguna mengatakan “Saya ingin pizza.” Klien mengirimkan input pengguna ini ke bot menggunakan salah satu operasi API runtime Amazon Lex. Dari input pengguna, Amazon Lex mengakui bahwa permintaan pengguna adalah untukOrderPizza maksud yang ditentukan dalam bot. Amazon

Lex melibatkan pengguna dalam percakapan untuk mengumpulkan informasi yang diperlukan, atau data slot, seperti ukuran pizza, topping, dan jumlah pizza. Setelah pengguna menyediakan semua data slot yang diperlukan, Amazon Lex akan memanggil hook kode fungsi Lambda untuk memenuhi maksud, atau mengembalikan data maksud ke klien, tergantung pada bagaimana maksud dikonfigurasi.

GunakanPostContent operasi saat bot Anda menggunakan input ucapan. Misalnya, aplikasi call center otomatis dapat mengirim pidato ke bot Amazon Lex, bukan agen untuk menjawab pertanyaan pelanggan. Anda dapat menggunakan format audio 8 kHz untuk mengirim audio langsung dari telepon ke Amazon Lex.

Jendela pengujian di konsol Amazon Lex menggunakanPostContent API untuk mengirim permintaan teks dan ucapan ke Amazon Lex. Anda menggunakan jendela tes ini dalamMemulai dengan Amazon Lex latihan.

Fungsi Lambda Sebagai Pengait Kode

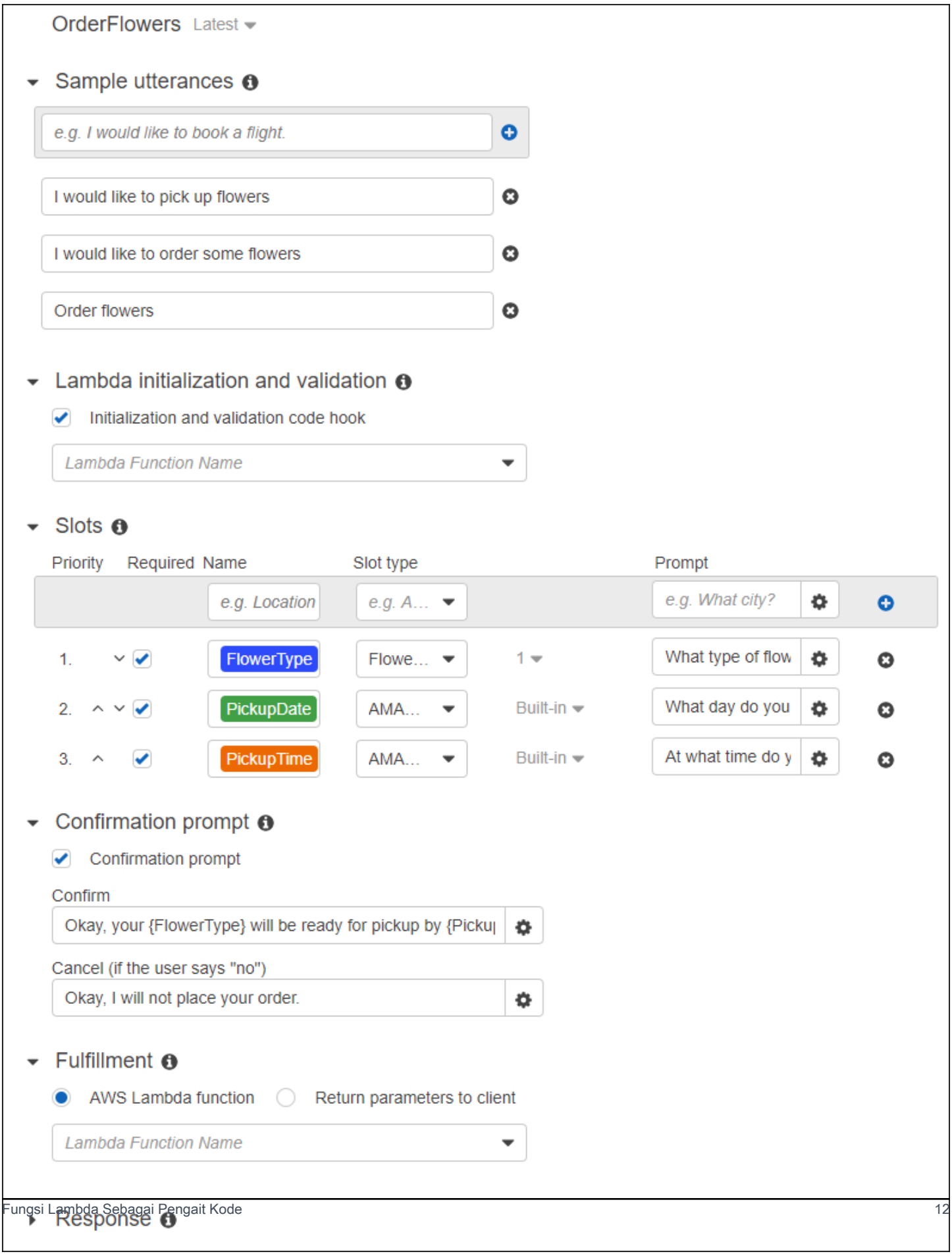
Anda dapat mengonfigurasi bot Amazon Lex Anda untuk menjalankan fungsi Lambda sebagai pengait kode. Kode hook dapat melayani beberapa tujuan:

- Menyesuaikan interaksi pengguna—misalnya, ketika Joe meminta topping pizza yang tersedia, Anda dapat menggunakan pengetahuan sebelumnya tentang pilihan Joe untuk menampilkan subset topping.
- Memvalidasi input pengguna—Misalkan Jen ingin mengambil bunga setelah jam kerja. Anda dapat memvalidasi waktu yang Jen masukan dan mengirim respon yang sesuai.
- Memenuhi maksud pengguna—Setelah Joe menyediakan semua informasi untuk pesanan pizza-nya, Amazon Lex dapat memanggil fungsi Lambda untuk memesan dengan restoran pizza lokal.

Saat mengonfigurasi maksud, Anda menentukan fungsi Lambda sebagai kait kode di tempat berikut:

- Pengait kode dialog untuk inisialisasi dan validasi — Fungsi Lambda ini dipanggil pada setiap input pengguna, dengan asumsi Amazon Lex memahami maksud pengguna.
- Fulfillment code hook—Fungsi Lambda ini dipanggil setelah pengguna menyediakan semua data slot yang diperlukan untuk memenuhi maksud.

Anda memilih maksud dan mengatur kait kode di konsol Amazon Lex, seperti yang ditunjukkan pada tangkapan layar berikut:


OrderFlowers Latest
Sample utterances
e.g. I would like to book a flight.
I would like to pick up flowers
I would like to order some flowers
Order flowers
Lambda initialization and validation
Initialization and validation code hook
Lambda Function Name
Slots
Priority Required Name Slot type Prompt
e.g. Location
e.g. A…
e.g. What city?
1. FlowerType Flowe… 1 What type of flow
2. PickupDate AMA… Built-in What day do you
3. PickupTime AMA… Built-in At what time do y
Confirmation prompt
Confirmation prompt
Confirm
Okay, your {FlowerType} will be ready for pickup by {Picku
Cancel (if the user says "no")
Okay, I will not place your order.
Fulfillment
AWS Lambda function
Return parameters to client
Lambda Function Name
Response

Anda juga dapat mengatur kait kode menggunakan`dialogCodeHook` dan`fulfillmentActivity` bidang dalamPutIntent operasi.

Satu fungsi Lambda dapat melakukan inisialisasi, validasi, dan pemenuhan. Data peristiwa yang diterima fungsi Lambda memiliki bidang yang mengidentifikasi pemanggil sebagai dialog atau pengait kode pemenuhan. Anda dapat menggunakan informasi ini untuk menjalankan bagian kode Anda yang sesuai.

Anda dapat menggunakan fungsi Lambda untuk membuat bot yang dapat menavigasi dialog yang kompleks. Anda menggunakan`dialogAction` bidang dalam respons fungsi Lambda untuk mengarahkan Amazon Lex untuk mengambil tindakan tertentu. Misalnya, Anda dapat menggunakan tindakan`ElicitSlot` dialog untuk memberi tahu Amazon Lex agar meminta nilai slot yang tidak diperlukan kepada pengguna. Jika Anda memiliki prompt klarifikasi yang ditentukan, Anda dapat menggunakan tindakan`ElicitIntent` dialog untuk mendapatkan maksud baru saat pengguna selesai dengan yang sebelumnya.

Untuk informasi selengkapnya, lihat Menggunakan Fungsi Lambda.

Mengelola Pesan

Topik

- Jenis Pesan
- Konteks untuk Mengkonfigurasi Pesan
- Format Pesan yang Didukung
- Grup Pesan
- Kartu Respon

Saat Anda membuat bot, Anda dapat mengonfigurasi pesan klarifikasi atau informasi yang ingin Anda kirimkan ke klien. Pertimbangkan contoh berikut:

- Anda dapat mengonfigurasi bot Anda dengan prompt klarifikasi berikut:

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

  Amazon Lex mengirimkan pesan ini ke klien jika tidak memahami maksud pengguna.

- Misalkan Anda membuat bot untuk mendukung maksud yang dipanggil`OrderPizza`. Untuk pesanan pizza, Anda ingin pengguna memberikan informasi seperti ukuran pizza, topping, dan jenis kerak. Anda dapat mengkonfigurasi perintah berikut:

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

  Setelah Amazon Lex menentukan maksud pengguna untuk memesan pizza, ia mengirimkan pesan ini ke klien untuk mendapatkan informasi dari pengguna.

Bagian ini menjelaskan merancang interaksi pengguna dalam konfigurasi bot Anda.

Jenis Pesan

Sebuah pesan dapat berupa prompt atau pernyataan.

- Prompt biasanya merupakan pertanyaan dan mengharapkan respons pengguna.
- Pernyataan bersifat informasi. Itu tidak mengharapkan respons.

Pesan dapat menyertakan referensi ke slot, atribut sesi, dan atribut permintaan. Pada waktu proses, Amazon Lex mengganti referensi ini dengan nilai aktual.

Untuk merujuk nilai slot yang telah ditetapkan, gunakan sintaksis berikut:

```
{SlotName}
```

Untuk merujuk atribut sesi, gunakan sintaksis berikut:

```
[SessionAttributeName]
```

Untuk merujuk atribut permintaan, gunakan sintaksis berikut:

```
((RequestAttributeName))
```

Pesan dapat mencakup nilai slot, atribut sesi, dan atribut permintaan.

Misalnya, anggap Anda mengkonfigurasi pesan berikut dalam OrderPizza maksud bot Anda:

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

Pesan ini mengacu pada kedua slot (`PizzaTopping`) dan atribut sesi (`FirstNamedanDeliveryTime`). Pada waktu proses, Amazon Lex menggantikan placeholder ini dengan nilai dan mengembalikan pesan berikut ke klien:

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

Untuk menyertakan tanda kurung ([]) atau tanda kurung ({}) dalam pesan, gunakan karakter escape garis miring terbalik (\). Misalnya, pesan berikut menyertakan kurung kurawal dan tanda kurung persegi:

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

Teks yang dikembalikan ke aplikasi klien terlihat seperti ini:

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

Untuk informasi tentang atribut sesi, lihat operasi API runtimePostText danPostContent. Sebagai contoh, lihat Buku Perjalanan.

Fungsi Lambda juga dapat menghasilkan pesan dan mengembalikannya ke Amazon Lex untuk dikirim ke pengguna. Jika Anda menambahkan fungsi Lambda saat mengonfigurasi maksud, Anda dapat membuat pesan secara dinamis. Dengan memberikan pesan saat mengonfigurasi bot Anda, Anda dapat menghilangkan kebutuhan untuk membuat prompt dalam fungsi Lambda Anda.

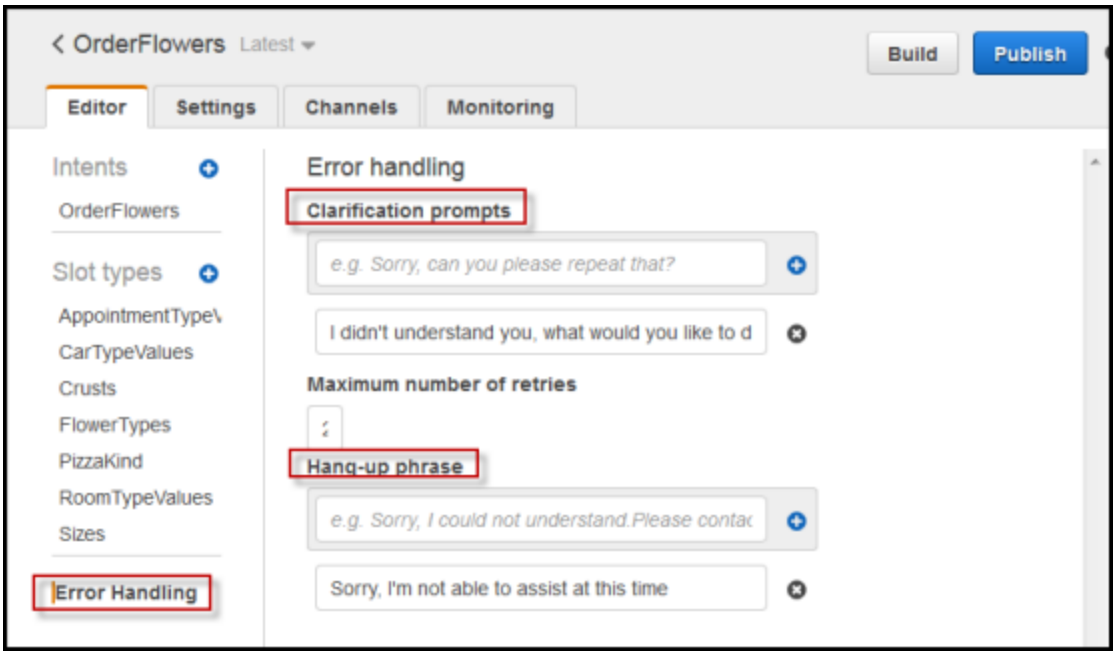
Konteks untuk Mengkonfigurasi Pesan

Saat Anda membuat bot, Anda dapat membuat pesan dalam konteks yang berbeda, seperti petunjuk klarifikasi dalam bot, petunjuk untuk nilai slot, dan pesan dari maksud. Amazon Lex memilih pesan yang sesuai dalam setiap konteks untuk kembali ke pengguna Anda. Anda dapat menyediakan grup pesan untuk setiap konteks. Jika Anda melakukannya, Amazon Lex secara acak memilih satu pesan dari grup. Anda juga dapat menentukan format pesan atau mengelompokkan pesan secara bersamaan. Untuk informasi selengkapnya, lihat Format Pesan yang Didukung.

Jika Anda memiliki fungsi Lambda yang terkait dengan maksud, Anda dapat mengganti pesan apa pun yang Anda konfigurasikan pada waktu pembuatan. Namun, fungsi Lambda tidak diperlukan untuk menggunakan pesan ini.

Pesan Bot

Anda dapat mengonfigurasi bot Anda dengan petunjuk klarifikasi dan pesan akhir sesi. Saat runtime, Amazon Lex menggunakan prompt klarifikasi jika tidak memahami maksud pengguna. Anda dapat mengonfigurasi berapa kali Amazon Lex meminta klarifikasi sebelum mengirim pesan akhir sesi. Anda mengonfigurasi pesan tingkat bot di bagian Penanganan Kesalahan pada konsol Amazon Lex, seperti pada gambar berikut:



Dengan API, Anda mengkonfigurasi pesan dengan mengatur`clarificationPrompt` dan`abortStatement` bidang dalamPutBot operasi.

Jika Anda menggunakan fungsi Lambda dengan maksud, fungsi Lambda mungkin mengembalikan respons yang mengarahkan Amazon Lex untuk menanyakan maksud pengguna. Jika fungsi Lambda tidak menyediakan pesan seperti itu, Amazon Lex menggunakan prompt klarifikasi.

Perintah Slot Perintah slot

Anda harus menentukan setidaknya satu pesan prompt untuk setiap slot yang diperlukan dalam intent. Pada saat runtime, Amazon Lex menggunakan salah satu pesan ini untuk meminta pengguna memberikan nilai untuk slot. Misalnya, untuk`cityName` slot, berikut ini adalah prompt yang valid:

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

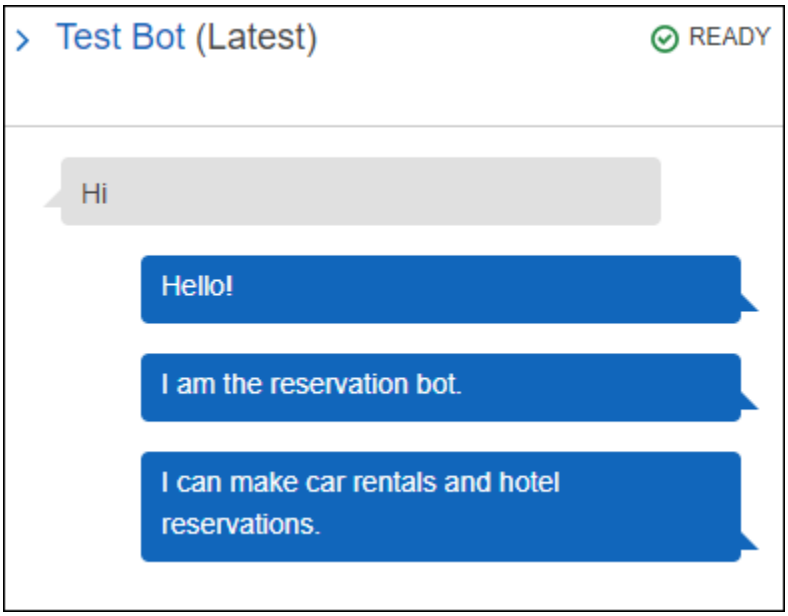
Anda dapat mengatur satu atau lebih petunjuk untuk setiap slot menggunakan konsol. Anda juga dapat membuat grup petunjuk menggunakanPutIntent operasi. Untuk informasi selengkapnya, lihat Grup Pesan.

Respons

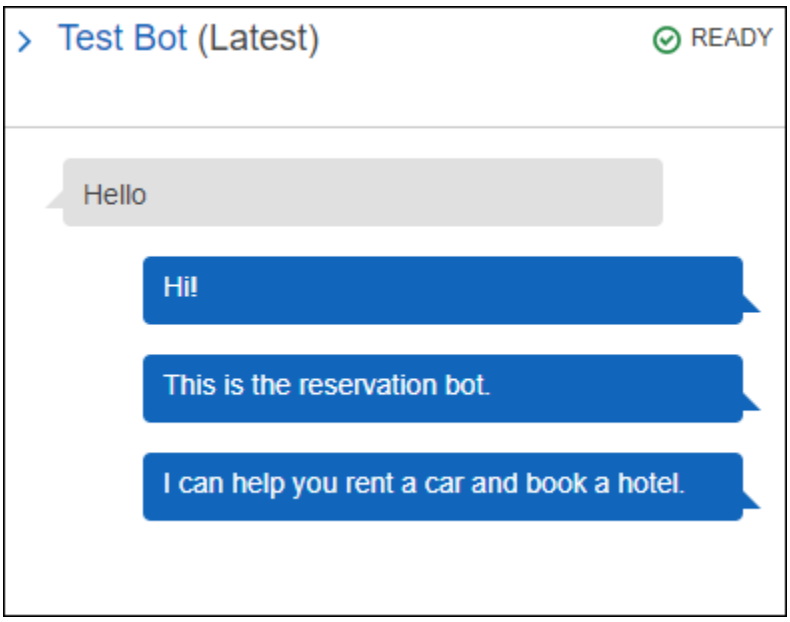
Di konsol, gunakan bagian Respons untuk membangun percakapan yang dinamis dan menarik untuk bot Anda. Anda dapat membuat satu atau beberapa grup pesan untuk respons. Pada waktu proses, Amazon Lex membuat respons dengan memilih satu pesan dari setiap grup pesan. Untuk informasi selengkapnya tentang grup pesan, lihatGrup Pesan.

Misalnya, grup pesan pertama Anda dapat berisi salam yang berbeda: “Halo,” “Hai,” dan “Salam.” Grup pesan kedua dapat berisi berbagai bentuk pengenalan: “Saya bot reservasi” dan “Ini adalah bot reservasi.” Kelompok pesan ketiga dapat mengkomunikasikan kemampuan bot: “Saya dapat membantu dengan penyewaan mobil dan pemesanan hotel,” “Anda dapat membuat penyewaan mobil dan pemesanan hotel,” dan “Saya dapat membantu Anda menyewa mobil dan memesan hotel.”

Lex menggunakan pesan dari masing-masing kelompok pesan untuk secara dinamis membangun tanggapan dalam percakapan. Misalnya, satu interaksi bisa menjadi berikut:

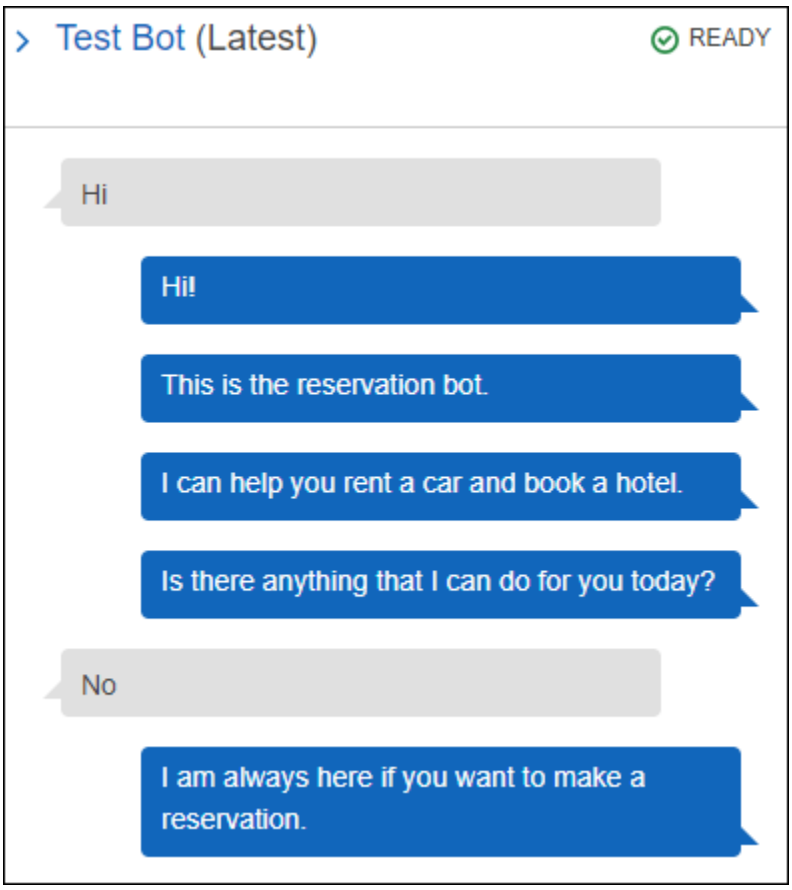


Satu lagi bisa sebagai berikut:

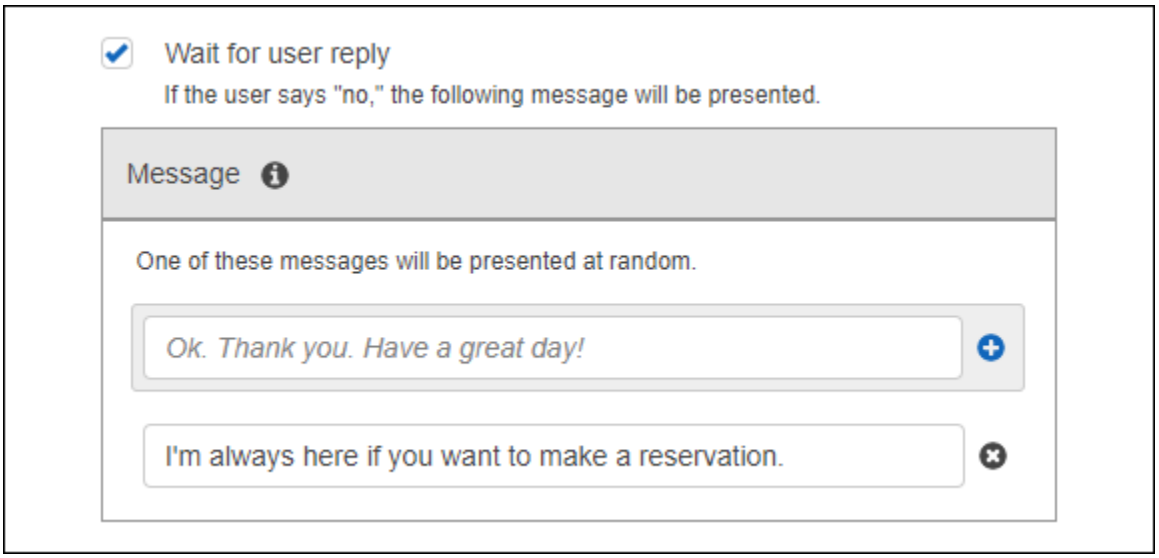


Dalam kedua kasus tersebut, pengguna dapat merespons dengan maksud baru, sepertiBookHotel intentBookCar atau.

Anda dapat mengatur bot untuk mengajukan pertanyaan tindak lanjut dalam respons. Misalnya, untuk interaksi sebelumnya, Anda dapat membuat grup pesan keempat dengan pertanyaan-pertanyaan berikut: “Dapatkah saya membantu dengan mobil atau hotel?” , “Apakah Anda ingin membuat reservasi sekarang?” , dan “Apakah ada sesuatu yang bisa saya lakukan untuk Anda?”. Untuk pesan yang menyertakan “Tidak” sebagai respons, Anda dapat membuat prompt tindak lanjut. Gambar berikut memberikan contoh:



Untuk membuat perintah tindak lanjut, pilih Tunggu balasan pengguna. Kemudian ketik pesan atau pesan yang ingin Anda kirim ketika pengguna mengatakan “Tidak.” Saat Anda membuat respons untuk digunakan sebagai prompt tindak lanjut, Anda juga harus menentukan pernyataan yang sesuai ketika jawaban atas pernyataan tersebut adalah “Tidak.” Lihat gambar berikut untuk contoh:



Untuk menambahkan respons ke intent dengan API, gunakan`PutIntent` operasi. Untuk menentukan respons, atur`conclusionStatement` bidang dalam`PutIntent` permintaan. Untuk mengatur prompt tindak lanjut, mengatur`followUpPrompt` bidang dan termasuk pernyataan untuk mengirim ketika pengguna mengatakan “Tidak.” Anda tidak dapat mengatur`conclusionStatement` bidang dan`followUpPrompt` bidang pada maksud yang sama.

Format Pesan yang Didukung

Saat Anda menggunakanPostText operasi, atau saat Anda menggunakanPostContent operasi dengan`Accept` header yang disetel ke`text/plain;charset=utf8`, Amazon Lex mendukung pesan dalam format berikut:

- `PlainText`—Pesan berisi teks UTF-8 biasa.
- `SSML`—Pesan berisi teks yang diformat untuk output suara.
- `CustomPayload`—Pesan berisi format kustom yang telah Anda buat untuk klien Anda. Anda dapat menetapkan payload untuk memenuhi kebutuhan aplikasi Anda.
- `Composite`—Pesan adalah kumpulan pesan, satu dari setiap grup pesan. Untuk informasi selengkapnya tentang grup pesan, lihatGrup Pesan.

Secara default, Amazon Lex mengembalikan salah satu pesan yang ditentukan untuk prompt tertentu. Misalnya, jika Anda menentukan lima pesan untuk mendapatkan nilai slot, Amazon Lex memilih salah satu pesan secara acak dan mengembalikannya ke klien.

Jika Anda ingin Amazon Lex mengembalikan jenis pesan tertentu ke klien dalam permintaan run-time, tetapkan parameter`x-amzn-lex:accept-content-types` permintaan. Respons terbatas pada jenis atau jenis yang diminta. Jika ada lebih dari satu pesan dari jenis yang ditentukan, Amazon Lex mengembalikan satu pesan secara acak. Untuk informasi selengkapnya tentang`x-amz-lex:accept-content-types` header, lihatMengatur Jenis Respons.

Grup Pesan

Grup pesan adalah sekumpulan respons yang sesuai untuk prompt tertentu. Gunakan grup pesan ketika Anda ingin bot Anda membangun respons secara dinamis dalam percakapan. Ketika Amazon Lex mengembalikan respons ke aplikasi klien, Amazon Lex akan memilih satu pesan secara acak dari setiap grup. Anda dapat membuat maksimal lima grup pesan untuk setiap respons. Setiap grup dapat berisi maksimal lima pesan. Untuk contoh membuat grup pesan di konsol, lihatRespons.

Untuk membuat grup pesan, Anda dapat menggunakan konsol atau Anda dapat menggunakanPutBot,PutIntent, atauPutSlotType operasi untuk menetapkan nomor grup ke pesan. Jika Anda tidak membuat grup pesan, atau jika Anda hanya membuat satu grup pesan, Amazon Lex mengirimkan satu pesan di`Message` bidang tersebut. Aplikasi klien mendapatkan beberapa pesan dalam respons hanya jika Anda telah membuat lebih dari satu grup pesan di konsol, atau ketika Anda membuat lebih dari satu grup pesan saat Anda membuat atau memperbarui maksud dengan`PutIntent` operasi.

Saat Amazon Lex mengirim pesan dari grup,`Message` kolom respons berisi objek JSON yang diloloskan yang berisi pesan. Contoh berikut ini menunjukkan konten`Message` bidang ketika berisi beberapa pesan.

> **Note**
>
> Contoh ini diformat agar dapat dibaca. Respons tidak mengandung carriage returns (CR).

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

Anda dapat mengatur format pesan. Formatnya dapat menjadi salah satu dari yang berikut:

- PlainText—Pesannya ada dalam teks UTF-8 biasa.
- SSML—pesannya adalah Speech Synthesis Markup Language (SSKL).
- CustomPayload—Pesan dalam format kustom yang Anda tentukan.

Untuk mengontrol format pesan yang dikembalikan`PostContent` dan`PostText` operasi di`Message` lapangan, atur atributx-amz-lex:accept-content-types permintaan. Misalnya, jika Anda menyetel header ke yang berikut, Anda hanya menerima pesan teks biasa dan SSMP dalam respons:

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

Jika Anda meminta format pesan tertentu dan grup pesan tidak berisi pesan dengan format tersebut, Anda akan mendapatkan`NoUsableMessageException` pengecualian. Bila Anda menggunakan

grup pesan untuk mengelompokkan pesan berdasarkan jenis, jangan gunakan`x-amz-lex:accept-content-types` header.

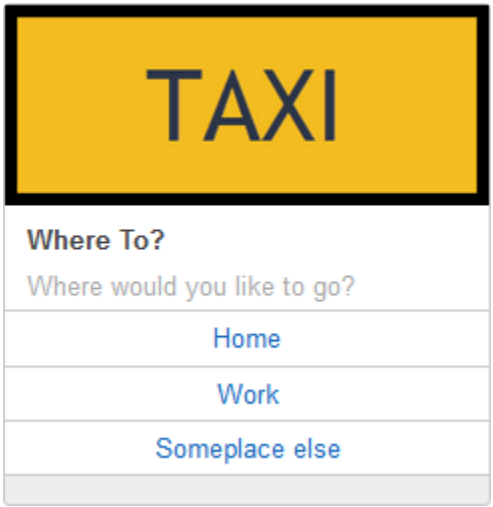
Untuk informasi selengkapnya tentang`x-amz-lex:accept-content-types` header, lihatMengatur Jenis Respons.

Kartu Respon

> **Note**
>
> Kartu respons tidak berfungsi dengan obrolan Amazon Connect. Namun, lihat Menambahkan pesan interaktif ke obrolan untuk fungsionalitas serupa.

Kartu respons berisi serangkaian tanggapan yang sesuai untuk prompt. Gunakan kartu respons untuk menyederhanakan interaksi bagi pengguna Anda dan meningkatkan akurasi bot Anda dengan mengurangi kesalahan ketik dalam interaksi teks. Anda dapat mengirim kartu respons untuk setiap prompt yang dikirim Amazon Lex ke aplikasi klien Anda. Anda dapat menggunakan kartu respons dengan Facebook Messenger, Slack, Twilio, dan aplikasi klien Anda sendiri.

Misalnya, dalam aplikasi taksi, Anda dapat mengkonfigurasi opsi di kartu respons untuk “Home” dan menetapkan nilai ke alamat rumah pengguna. Saat pengguna memilih opsi ini, Amazon Lex menerima seluruh alamat sebagai teks masukan. Lihat citra berikut:



Anda dapat menentukan kartu respons untuk petunjuk berikut:

- Pernyataan kesimpulan kesimpulan

- Prompt konfirmasi
- Tindak lanjut prompt
- Pernyataan penolakan
- Ucapan jenis slot

Anda hanya dapat menentukan satu kartu respons untuk setiap prompt.

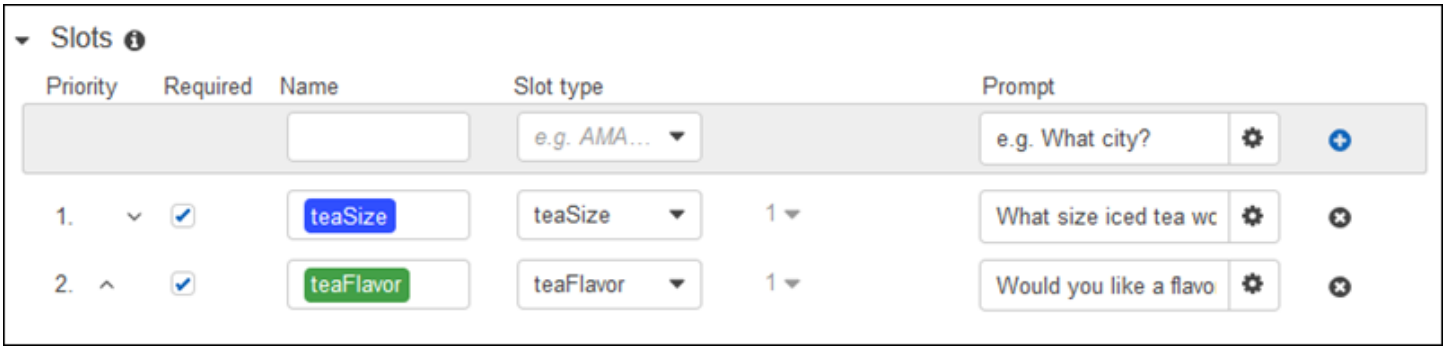
Anda mengonfigurasi kartu respons saat membuat maksud. Anda dapat menentukan kartu respons statis pada waktu pembuatan menggunakan konsol atauPutIntent operasi. Atau Anda dapat menentukan kartu respons dinamis saat runtime dalam fungsi Lambda. Jika Anda menentukan kartu respons statis dan dinamis, kartu respons dinamis akan diutamakan.

Amazon Lex mengirimkan kartu respons dalam format yang dipahami klien. Ini mengubah kartu respons untuk Facebook Messenger, Slack, dan Twilio. Untuk klien lain, Amazon Lex mengirimkan struktur JSON dalamPostText respons. Misalnya, jika klien adalah Facebook Messenger, Amazon Lex mengubah kartu respons menjadi template generik. Untuk informasi lebih lanjut tentang template generik Facebook Messenger, lihat Template Generik di situs web Facebook. Untuk contoh struktur JSON, lihatMenghasilkan Kartu Respon Secara Dinamis.

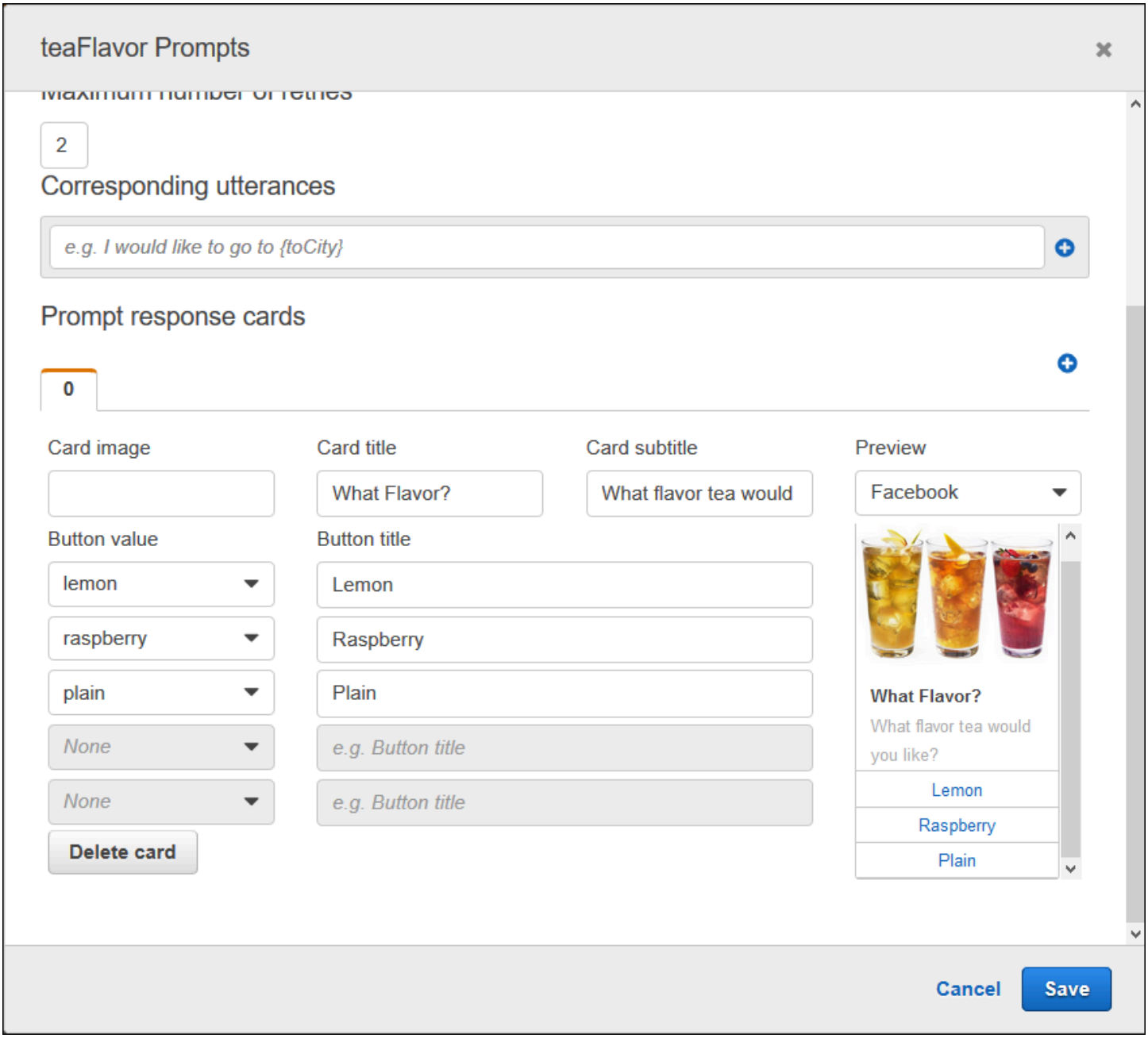
Anda dapat menggunakan kartu respons hanya denganPostText operasi. Anda tidak dapat menggunakan kartu respons denganPostContent operasi.

Mendefinisikan Kartu Respon Statis

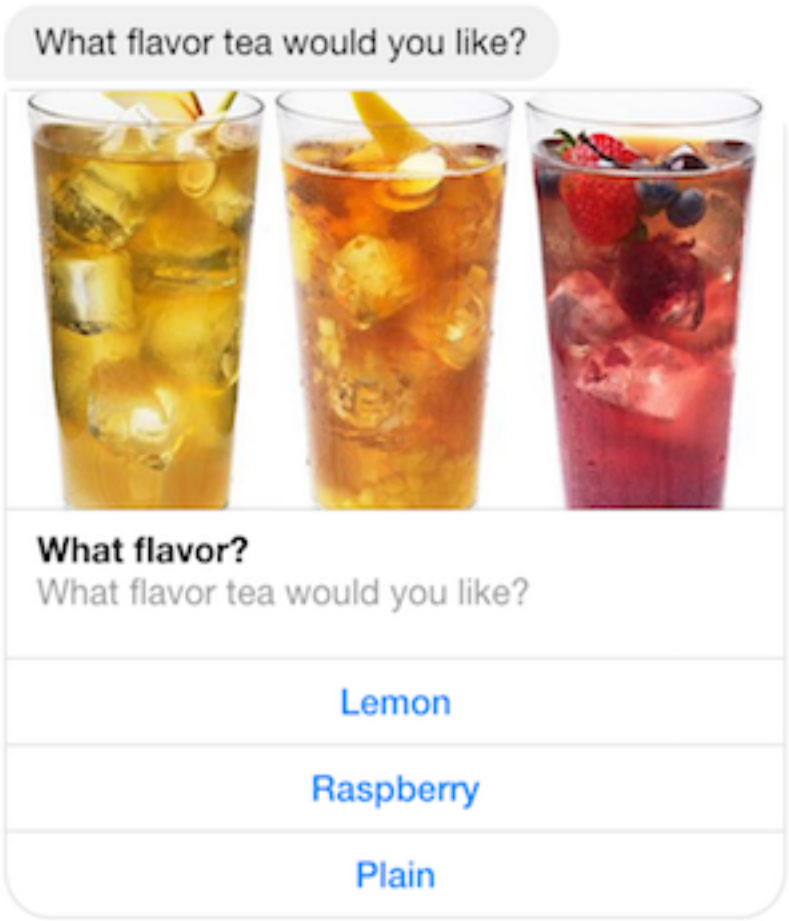
Tentukan kartu respons statis denganPutBot operasi atau konsol Amazon Lex saat Anda membuat maksud. Kartu respons statis didefinisikan bersamaan dengan intent. Gunakan kartu respons statis saat tanggapan diperbaiki. Misalkan Anda membuat bot dengan maksud yang memiliki slot untuk rasa. Saat mendefinisikan slot rasa, Anda menetapkan perintah, seperti yang ditunjukkan pada gambar konsol berikut:



Saat menentukan prompt, Anda dapat mengaitkan kartu respons secara opsional dan menentukan detail dengan[PutBot](#) operasi, atau, di konsol Amazon Lex, seperti yang ditunjukkan pada contoh berikut:



Sekarang anggaplah Anda telah mengintegrasikan bot Anda dengan Facebook Messenger. Pengguna dapat mengklik tombol untuk memilih ragam, seperti yang ditunjukkan pada ilustrasi berikut:



Untuk menyesuaikan konten kartu respons, Anda dapat merujuk ke atribut sesi. Pada waktu proses, Amazon Lex mengganti referensi ini dengan nilai yang sesuai dari atribut sesi. Untuk informasi selengkapnya, lihat Mengatur Atribut Sesi. Sebagai contoh, lihat Menggunakan Kartu Respons.

Menghasilkan Kartu Respon Secara Dinamis

Untuk menghasilkan kartu respons secara dinamis saat runtime, gunakan fungsi Lambda inisialisasi dan validasi untuk intent tersebut. Gunakan kartu respons dinamis saat respons ditentukan pada waktu proses dalam fungsi Lambda. Menanggapi input pengguna, fungsi Lambda menghasilkan kartu respons dan mengembalikannya di`dialogAction` bagian respons. Untuk informasi selengkapnya, lihat Format Respons.

Berikut ini adalah respons parsi dari fungsi Lambda yang menunjukkan`responseCard` elemen. Nama ini menghasilkan pengalaman pengguna yang mirip dengan yang ditunjukkan pada bagian sebelumnya.

```
responseCard: {
```

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

Sebagai contoh, lihat Jadwalkan Janji.

Mengelola Konteks Percakapan

Konteks percakapan adalah informasi yang disediakan pengguna, aplikasi Anda, atau fungsi Lambda ke bot Amazon Lex untuk memenuhi maksud. Konteks percakapan mencakup data slot yang disediakan pengguna, meminta atribut yang ditetapkan oleh aplikasi klien, dan atribut sesi yang dibuat oleh aplikasi klien dan fungsi Lambda.

Topik

- Konteks Maksud Konteks Maksud Maksud
- Menggunakan Nilai Slot Default
- Mengatur Atribut Sesi
- Mengatur Atribut Permintaan

- Mengatur Timeout Sesi
- Berbagi Informasi Antar Maksud
- Mengatur Atribut Kompleks Kompleks

Konteks Maksud Konteks Maksud Maksud

Anda dapat memiliki maksud pemicu Amazon Lex 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 menggunakanPutIntent operasi. Anda hanya dapat menggunakan konteks dalam bahasa Inggris (AS) (en-US) lokal, dan hanya jika Anda mengaturenableModelImprovements parametertrue ketika Anda membuat bot denganPutBot operasi.

Ada dua jenis hubungan untuk konteks, konteks output dan konteks masukan. Konteks keluaran menjadi aktif saat maksud terkait terpenuhi. Konteks keluaran dikembalikan ke aplikasi Anda dalam respons dariPostContent operasiPostText atau, dan diatur untuk sesi saat ini. Setelah konteks diaktifkan, itu tetap aktif untuk jumlah putaran atau batas waktu yang dikonfigurasi ketika konteks didefinisikan.

Konteks masukan menentukan kondisi di mana intent dapat dikenali. Maksud hanya dapat dikenali selama percakapan ketika semua konteks inputnya aktif. Maksud tanpa konteks masukan selalu memenuhi syarat untuk pengakuan.

Amazon Lex secara otomatis mengelola siklus hidup konteks yang diaktifkan dengan memenuhi maksud dengan konteks keluaran. Anda juga dapat mengatur konteks aktif dalam panggilan kePostContent atauPostText operasi.

Anda juga dapat menetapkan konteks suatu percakapan menggunakan fungsi Lambda untuk maksud tersebut. Konteks keluaran dari Amazon Lex dikirim ke acara input fungsi Lambda. Fungsi Lambda dapat mengirim konteks dalam responsnya. Untuk informasi selengkapnya, lihat Lambda Fungsi Input Event dan Response Format.

Misalnya, Anda memiliki maksud untuk memesan mobil sewaan yang dikonfigurasi untuk mengembalikan konteks keluaran yang disebut “book_car_fulfilled”. Saat maksud terpenuhi, Amazon Lex menyetel variabel konteks keluaran “book_car_fulfilled”. Karena “book_car_fulfilled” adalah konteks aktif, maksud dengan konteks “book_car_fulfilled” ditetapkan sebagai konteks masukan sekarang dipertimbangkan untuk pengenalan, selama ucapan pengguna diakui sebagai upaya untuk

mendapatkan maksud itu. Anda dapat menggunakan ini untuk maksud yang hanya masuk akal setelah memesan mobil, seperti mengirim email tanda terima atau memodifikasi reservasi.

konteks Keluaran Konteks Keluaran

Amazon Lex 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. Parameternya adalah nilai slot untuk maksud yang terpenuhi. Anda dapat menggunakan parameter ini untuk pra-mengisi nilai slot untuk maksud lainnya. Untuk informasi selengkapnya, lihatMenggunakan Nilai Slot Default.

Anda mengonfigurasi konteks keluaran saat membuat maksud dengan konsol atau denganPutIntent operasi. Anda dapat mengonfigurasi maksud dengan lebih dari satu konteks keluaran. Ketika maksud terpenuhi, semua konteks keluaran diaktifkan dan dikembalikan dalamPostText atauPostContent respons.

Berikut ini menunjukkan menetapkan konteks keluaran ke maksud menggunakan konsol.

Ketika Anda menentukan konteks keluaran, Anda juga menentukan waktunya untuk hidup, lamanya waktu atau jumlah putaran konteks disertakan dalam respons dari Amazon Lex. Giliran adalah salah satu permintaan dari aplikasi Anda ke Amazon Lex. Setelah jumlah belokan atau waktu telah berakhir, konteksnya tidak lagi aktif.

Aplikasi Anda dapat menggunakan konteks keluaran sesuai kebutuhan. Misalnya, aplikasi Anda dapat menggunakan konteks keluaran untuk:

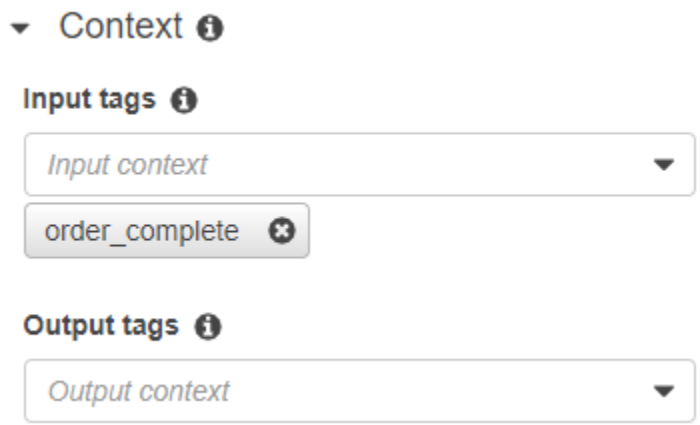
- Mengubah perilaku aplikasi berdasarkan konteks. Misalnya, aplikasi perjalanan dapat memiliki tindakan yang berbeda untuk konteks “book_car_fulfilled” dari “rental_hotel_fulfilled.”

- Kembalikan konteks keluaran ke Amazon Lex sebagai konteks masukan untuk ucapan berikutnya. Jika Amazon Lex mengenali ucapan sebagai upaya untuk mendapatkan maksud, ia menggunakan konteks untuk membatasi maksud yang dapat dikembalikan ke intent dengan konteks yang ditentukan.

konteks masukan Konteks Masukan

Anda menetapkan konteks masukan untuk membatasi titik dalam percakapan tempat maksud dikenali. Maksud tanpa konteks masukan selalu memenuhi syarat untuk dikenali.

Anda menyetel konteks masukan yang ditanggapi intent menggunakan konsol atau`PutIntent` operasi. Maksud dapat memiliki lebih dari satu konteks input. Berikut ini menunjukkan menetapkan konteks masukan ke maksud menggunakan konsol.



Untuk maksud dengan lebih dari satu konteks masukan, semua konteks harus aktif untuk memicu maksud. Anda dapat mengatur konteks masukan ketika Anda memanggilPostText,PostContent, atauPutSession operasi.

Anda dapat mengonfigurasi slot dalam intent untuk mengambil nilai default dari konteks aktif saat ini. Nilai default digunakan saat Amazon Lex mengenali maksud baru tetapi tidak menerima nilai slot. Anda menentukan nama konteks dan nama slot dalam formulir`#context-name.parameter-name` ketika Anda menentukan slot. Untuk informasi selengkapnya, lihat Menggunakan Nilai Slot Default.

Menggunakan Nilai Slot Default

Bila 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 atribut dialog, permintaan, atau sesi sebelumnya, atau nilai tetap yang Anda tetapkan pada waktu pembuatan.

Anda dapat menggunakan berikut ini sebagai sumber nilai default Anda.

- Dialog sebelumnya (konteks) - #context -name.parameter-name
- Atribut sesi - [nama atribut]
- Atribut permintaan - <attribute-name>
- Nilai tetap - Nilai apa pun yang tidak cocok dengan sebelumnya

Bila Anda menggunakanPutIntent operasi untuk menambahkan slot ke intent, Anda dapat menambahkan daftar nilai default. Nilai default digunakan dalam daftar. Misalnya, anggap Anda memiliki maksud dengan slot dengan definisi berikut:

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

Ketika maksud dikenali, slot bernama "reservation-start-date" memiliki nilainya ditetapkan ke salah satu dari berikut ini.

1. Jika konteksbook-car-fulfilled "" aktif, nilai parameter “startDate” digunakan sebagai nilai default.
2. Jika konteksbook-car-fulfilled "" tidak aktif, atau jika parameter “startDate” tidak diatur, nilai atribut sesireservationStartDate "" 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 akan memperoleh nilai seperti biasa.

Jika nilai default digunakan untuk slot, slot tidak memunculkan bahkan jika diperlukan.

Mengatur Atribut Sesi

Atribut sesi berisi informasi khusus aplikasi yang diteruskan antara bot dan aplikasi klien selama sesi berlangsung. Amazon Lex meneruskan atribut sesi ke semua fungsi Lambda yang dikonfigurasi untuk bot. Jika fungsi Lambda menambahkan atau memperbarui atribut sesi, Amazon Lex meneruskan informasi baru kembali ke aplikasi klien. Misalnya:

- DalamLatihan 1: Membuat Bot Amazon Lex Menggunakan Cetak Biru (Konsol), contoh bot menggunakan atributprice sesi untuk mempertahankan harga bunga. Fungsi Lambda menetapkan atribut ini berdasarkan jenis bunga yang dipesan. Untuk informasi selengkapnya, lihat Langkah 5 (Opsional): Tinjau Rincian Aliran Informasi (Konsol).
- DiBuku Perjalanan, bot contoh menggunakan atributcurrentReservation sesi untuk menyimpan salinan data jenis slot selama percakapan untuk memesan hotel atau memesan mobil sewaan. Untuk informasi selengkapnya, lihat Rincian Alur Informasi.

Gunakan atribut sesi dalam fungsi Lambda Anda untuk menginisialisasi bot dan menyesuaikan prompt dan kartu respons. Misalnya:

- Inisialisasi - Dalam bot pemesanan pizza, aplikasi klien melewati lokasi pengguna sebagai atribut sesi dalam panggilan pertama kePostText operasiPostContent atau. Sebagai contoh, `"Location": "111 Maple Street"`. Fungsi Lambda menggunakan informasi ini untuk menemukan restoran pizza terdekat untuk melakukan pemesanan.
- Personalize 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 sesi (`{"FirstName": "Jo"}`), Amazon Lex mengganti nama untuk placeholder. Kemudian mengirimkan prompt pribadi untuk pengguna, “Hei Jo, yang topping akan Anda inginkan?”

Atribut sesi bertahan selama sesi berlangsung. Amazon Lex menyimpannya di penyimpanan data terenkripsi hingga sesi berakhir. Klien dapat membuat atribut sesi dalam permintaan dengan memanggil baikPostContent atauPostText operasi dengansessionAttributes bidang diatur ke nilai. Fungsi Lambda dapat membuat atribut sesi dalam respons. Setelah klien atau fungsi Lambda membuat atribut sesi, nilai atribut yang disimpan digunakan kapan saja aplikasi klien tidak menyertakansessionAttribute bidang dalam permintaan ke Amazon Lex.

Misalnya, anggap Anda memiliki dua atribut sesi,`{"x": "1", "y": "2"}`. Jika klien memanggilPostContent atauPostText operasi tanpa menentukansessionAttributes bidang,

Amazon Lex memanggil fungsi Lambda dengan atribut sesi yang disimpan (`{"x": 1, "y": 2}`). Jika fungsi Lambda tidak mengembalikan atribut sesi, Amazon Lex mengembalikan atribut sesi yang disimpan ke aplikasi klien.

Jika aplikasi klien atau fungsi Lambda melewati atribut sesi, Amazon Lex memperbarui atribut sesi yang disimpan. Melewati nilai yang ada, seperti `{"x": 2}`, memperbarui nilai yang disimpan. Jika Anda melewati serangkaian 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, Anda membuat string-to-string peta atribut. Berikut ini menunjukkan cara memetakan atribut sesi:

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

Untuk`PostText` operasi, Anda memasukkan peta ke dalam badan permintaan menggunakan`sessionAttributes` bidang, sebagai berikut:

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

Untuk`PostContent` operasi, Anda base64 menyandikan peta, dan kemudian mengirimkannya sebagai`x-amz-lex-session-attributes` header.

Jika Anda mengirim data biner atau terstruktur dalam atribut sesi, Anda harus terlebih dahulu mengubah data ke string sederhana. Untuk informasi selengkapnya, lihat Mengatur Atribut Kompleks Kompleks.

Mengatur Atribut Permintaan

Atribut permintaan berisi informasi khusus permintaan dan hanya berlaku untuk permintaan saat ini. Aplikasi klien mengirimkan informasi ini ke Amazon Lex. Gunakan atribut permintaan untuk meneruskan informasi yang tidak perlu bertahan untuk seluruh sesi. Anda dapat membuat atribut permintaan Anda sendiri atau Anda dapat menggunakan atribut standar. Untuk mengirim atribut permintaan, gunakan`x-amz-lex-request-attributes` header dalamthe section called

"PostContent" ataurequestAttributes bidang dalamthe section called "PostText" permintaan. Karena atribut permintaan tidak bertahan di seluruh permintaan seperti atribut sesi, atribut tersebut tidak ditampilkanPostContent atauPostText ditanggapi.

> **Note**
>
> Untuk mengirim informasi yang tetap ada di seluruh permintaan, gunakan atribut sesi.

Namespacex-amz-lex: dicadangkan untuk atribut permintaan yang telah ditetapkan. Jangan membuat atribut permintaan dengan awalanx-amz-lex:.

Mengatur Atribut Permintaan yang Telah Ditetapkan

Amazon Lex 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 ditetapkan dalam setiap permintaan. Semua atribut yang telah ditetapkan berada dix-amz-lex: namespace.

Selain atribut yang telah ditentukan sebelumnya berikut, Amazon Lex menyediakan atribut yang telah ditentukan sebelumnya untuk platform perpesanan. Untuk daftar atribut tersebut, lihatMenyebarkan Amazon Lex Bot pada Platform Pesan.

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 untuk teks biasa, tetapi memungkinkan klien seluler untuk menggunakan teks biasa dan Speech Synthesis Markup Language (SSMP). Untuk mengatur format pesan yang dikembalikan olehPostContent danPostText operasi, gunakan atributx-amz-lex:accept-content-types" request.

Anda dapat menetapkan atribut ke kombinasi dari jenis pesan berikut:

- PlainText—Pesan berisi teks UTF-8 biasa.
- SSML—Pesan berisi teks yang diformat untuk output suara.
- CustomPayload—Pesan berisi format kustom yang telah Anda buat untuk klien Anda. Anda dapat menetapkan payload untuk memenuhi kebutuhan aplikasi Anda.

Amazon Lex hanya mengembalikan pesan dengan jenis yang ditentukan di`Message` bidang 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.

> **Note**
>
> Atribut`x-amz-lex:accept-content-types` request tidak berpengaruh pada isi tubuh HTML. Isi respons`PostText` operasi selalu teks UTF-8 biasa. Tubuh respons`PostContent` operasi berisi data dalam format yang ditetapkan dalam`Accept` header dalam permintaan.

Mengatur Zona Waktu Pilihan Zona Waktu Pilihan

Untuk mengatur zona waktu yang digunakan untuk menyelesaikan tanggal sehingga relatif terhadap zona waktu pengguna, gunakan atribut`x-amz-lex:time-zone` request. Jika Anda tidak menentukan zona waktu dalam`x-amz-lex:time-zone` atribut, default tergantung pada wilayah yang Anda gunakan untuk bot Anda.

Wilayah	Zona waktu default
US East (N. Virginia)	`America/New_York`
US West (Oregon)	`America/Los_Angeles`
Asia Pasifik (Singapura)	`Asia/Singapore`
Asia Pacific (Sydney)	`Australia/Sydney`
Asia Pacific (Tokyo)	`Asia/Tokyo`
Europe (Frankfurt)	`Europe/Berlin`
Europe (Ireland)	`Europe/Dublin`
Europe (London)	`Europe/London`

Misalnya, jika pengguna merespons sebagai`tomorrow` tanggapan atas prompt “Hari apa Anda ingin paket Anda dikirimkan?” tanggal aktual paket dikirimkan tergantung pada zona waktu pengguna. Misalnya, ketika pukul 01:00 16 September di New York, sekarang pukul 22:00 15 September di Los Angeles. Jika layanan Anda berjalan di Wilayah AS Timur (Virginia Utara) dan seseorang di Los Angeles memesan paket yang akan dikirim “besok” menggunakan zona waktu default, paket akan dikirimkan pada tanggal 17, bukan tanggal 16. Namun, jika Anda menetapkan atribut`x-amz-lex:time-zone` request ke`America/Los_Angeles`, paket akan dikirimkan pada tanggal 16.

Anda dapat mengatur atribut ke salah satu nama zona waktu Internet Assigned Number Authority (IANA). Untuk daftar nama zona waktu, lihat Daftar zona waktu database tz di Wikipedia.

Mengatur Atribut Permintaan yang Ditetapkan Pengguna

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`PostContent` permintaan atau di`requestAttributes` bidang`PostText` permintaan.

Untuk mengirim atribut permintaan ke Amazon Lex, Anda membuat string-to-string peta 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`x-amz-lex-request-attributes` header.

Jika Anda mengirim data biner atau terstruktur dalam atribut request, Anda harus terlebih dahulu mengubah data ke string sederhana. Untuk informasi selengkapnya, lihat Mengatur Atribut Kompleks Kompleks.

Mengatur Timeout Sesi

Amazon Lex mempertahankan 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 intent seperti`OrderShoes` dan`GetOrderStatus`. Ketika Amazon Lex mendeteksi bahwa maksud pengguna adalah untuk 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 berakhir, ia dapat memberikan data slot yang tersisa, dan menyelesaikan pembelian.

Di konsol Amazon Lex, Anda mengatur batas waktu sesi saat membuat bot. Dengan antarmuka baris perintah AWS (AWS CLI) atau API, Anda menetapkan batas waktu saat membuat atau memperbarui bot dengan[PutBot](#) operasi dengan mengaturInSeconds bidang [IdleSessionTTL](#).

Berbagi Informasi Antar Maksud

Amazon Lex mendukung berbagi informasi antar maksud. Untuk berbagi antara maksud, gunakan atribut sesi.

Misalnya, penggunaan`ShoeOrdering` bot mulai dengan memesan sepatu. Bot terlibat dalam percakapan dengan pengguna, mengumpulkan data slot, seperti ukuran sepatu, warna, dan merek. Ketika pengguna melakukan pemesanan, fungsi Lambda yang memenuhi urutan menetapkan atribut`orderNumber` sesi, yang berisi nomor pesanan. 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, ia mengembalikan status pesanan.

Jika Anda berpikir bahwa pengguna Anda mungkin beralih intent selama sesi yang sama, Anda dapat merancang bot Anda untuk mengembalikan status pesanan terbaru. Alih-alih meminta informasi pesanan kepada pengguna lagi, Anda menggunakan atribut`orderNumber` sesi untuk berbagi informasi di seluruh maksud dan memenuhi`GetOrderStatus` maksud. Bot melakukan ini dengan mengembalikan status urutan terakhir yang ditempatkan pengguna.

Untuk contoh berbagi informasi cross-intent, lihat[Buku Perjalanan](#).

Mengatur Atribut Kompleks Kompleks

Sesi dan permintaan atribut adalah string-to-string peta atribut dan nilai-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 string-to-string peta. Dalam kasus seperti itu, Anda dapat mengubah objek menjadi string sederhana sehingga Anda dapat mengirimkannya ke bot Anda dengan PostContent dan PostText operasi.

Misalnya, jika Anda menggunakan JavaScript, Anda dapat menggunakan `JSON.stringify` operasi untuk mengonversi objek ke JSON, dan `JSON.parse` operasi untuk mengonversi teks JSON ke 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 sesi dengan`PostContent` operasi, Anda harus base64 menyandikan atribut sebelum Anda menambahkannya ke header permintaan, seperti yang ditunjukkan dalam JavaScript kode berikut:

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

Anda dapat mengirim data biner ke`PostContent` dan`PostText` operasi dengan terlebih dahulu mengkonversi data ke string base64 dikodekan, dan kemudian mengirim string sebagai nilai dalam atribut sesi:

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

Menggunakan Skor Keyakinan

Ketika pengguna mengucapkan ucapan, Amazon Lex menggunakan pemahaman bahasa alami (NLU) untuk memahami permintaan pengguna dan mengembalikan maksud yang tepat. Secara default, Amazon Lex mengembalikan maksud yang paling mungkin didefinisikan oleh bot Anda.

Dalam beberapa kasus mungkin sulit bagi Amazon Lex 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 denganskor kepercayaandari daftar maksud alternatif. Skor kepercayaan adalah peringkat yang Amazon Lex menyediakan yang menunjukkan betapa percaya diri bahwa maksud adalah maksud yang benar.

Untuk menentukan perbedaan antara dua maksud alternatif, Anda dapat membandingkan skor kepercayaan diri mereka. Misalnya, jika satu maksud memiliki skor kepercayaan 0,95 dan yang lainnya memiliki skor 0,65, maksud pertama mungkin benar. Namun, jika satu maksud memiliki skor

0.75 dan yang lainnya memiliki skor 0.72, ada ambiguitas antara dua maksud yang mungkin dapat Anda diskriminasi menggunakan pengetahuan domain dalam aplikasi Anda.

Anda juga dapat menggunakan skor kepercayaan diri 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 perbarui intent dengan ucapan baru. Anda kemudian dapat memeriksa skor kepercayaan diri untuk melihat apakah ada perbaikan.

Skor kepercayaan yang dihasilkan Amazon Lex adalah nilai komparatif. Anda tidak harus mengandalkan mereka sebagai skor absolut. Nilai dapat berubah berdasarkan perbaikan Amazon Lex.

Bila Anda menggunakan skor kepercayaan diri, Amazon Lex mengembalikan maksud yang paling mungkin dan hingga 4 maksud alternatif dengan skor terkait mereka dalam setiap respons. Jika semua skor kepercayaan kurang dari ambang batas, Amazon Lex menyertakan`AMAZON.FallbackIntent`, yang`AMAZON.KendraSearchIntent`, atau keduanya, jika Anda memilikinya dikonfigurasi. Anda dapat menggunakan ambang batas default atau Anda dapat mengatur ambang batas Anda sendiri.

Kode JSON berikut menunjukkan`alternativeIntents`bidang dalam respon dariPostTextoperasi.

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

Mengatur ambang batas saat membuat atau memperbarui bot. Anda dapat menggunakan API atau konsol Amazon Lex. Untuk wilayah yang tercantum di bawah ini, Anda perlu ikut serta untuk mengaktifkan peningkatan akurasi dan skor kepercayaan diri. Di konsol, pilih skor kepercayaan diri diOpsi LanjutanBagian. Menggunakan API, atur`enableModelImprovements`parameter saat Anda memanggilPutBotoperasi. :

- US East (N. Virginia) (us-east-1)

- US West (Oregon) (us-west-2)
- Asia Pacific (Sydney) (ap-southeast-2)
- Europe (Ireland) (eu-west-1)

Di semua wilayah lain, peningkatan akurasi dan dukungan skor kepercayaan tersedia secara default.

Untuk mengubah ambang kepercayaan, atur di konsol atau gunakanPutBotoperasi. Ambang batas harus berupa angka antara 1.00 dan 0.00.

Untuk menggunakan konsol, atur ambang kepercayaan saat Anda membuat atau memperbarui bot Anda.

Untuk mengatur ambang kepercayaan saat membuat bot (Console)

- PadaMembuat bot Anda, masukkan nilai diAmbang batas kepercayaanBidang.

Untuk memperbarui ambang kepercayaan (Console)

1. Dari daftar bot Anda, pilih bot untuk diperbarui.
2. Pilih tab Pengaturan.
3. Di navigasi kiri, pilihUmum.
4. Memperbarui nilai diAmbang batas kepercayaanBidang.

Untuk mengatur atau memperbarui ambang kepercayaan (SDK)

- Mengaturnlu`IntentConfidenceThreshold`parameterPutBotoperasi. Kode JSON berikut menunjukkan parameter yang ditetapkan.

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

Manajemen Sesi

Untuk mengubah maksud yang digunakan Amazon Lex dalam percakapan dengan pengguna, Anda dapat menggunakan respons dari fungsi Lambda hook kode dialog, atau Anda dapat menggunakan API manajemen sesi di aplikasi kustom Anda.

Menggunakan fungsi Lambda

Saat Anda menggunakan fungsi Lambda, Amazon Lex menyebutnya dengan struktur JSON yang berisi input ke fungsi. Struktur JSON berisi bidang yang disebut`currentIntent`yang berisi maksud yang Amazon Lex telah diidentifikasi sebagai maksud yang paling mungkin untuk ucapan pengguna. Struktur JSON juga mencakup`alternativeIntents`bidang yang berisi hingga empat maksud tambahan yang dapat memenuhi maksud pengguna. Setiap maksud mencakup bidang yang disebut`nluIntentConfidenceScore`yang berisi skor kepercayaan bahwa Amazon Lex ditugaskan untuk maksud.

Untuk menggunakan maksud alternatif, Anda menentukannya di`ConfirmIntent`atau`ElicitSlot`tindakan dialog dalam fungsi Lambda Anda.

Untuk informasi selengkapnya, lihat Menggunakan Fungsi Lambda.

Menggunakan API Manajemen Sesi

Untuk menggunakan intent berbeda dari intent saat ini, gunakanPutSessionoperasi. Misalnya, jika Anda memutuskan bahwa alternatif pertama lebih baik dari maksud yang Amazon Lex pilih, Anda dapat menggunakan`PutSession`operasi untuk mengubah maksud sehingga maksud berikutnya yang berinteraksi pengguna adalah yang Anda pilih.

Untuk informasi selengkapnya, lihat Mengelola Sesi Dengan Amazon Lex API.

Log Percakapan

Anda mengaktifkan log percakapan untuk menyimpan interaksi bot. Anda dapat menggunakan log ini untuk meninjau kinerja bot Anda dan memecahkan masalah dengan percakapan. Anda dapat log teks untukPostText operasi. Anda dapat mencatat teks dan audio untukPostContent operasi. 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 Children's Online Privacy Protection Act (COPPA).

Log percakapan dikonfigurasi untuk alias. Setiap alias dapat memiliki pengaturan yang berbeda untuk log teks dan 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 CloudWatch Log. Log audio menyimpan input audio di Amazon S3. Anda dapat mengaktifkan enkripsi log teks dan audio menggunakan CMK yang dikelolaAWS KMS pelanggan.

Untuk mengkonfigurasi logging, gunakan konsol atauPutBotAlias operasi. Anda tidak dapat mencatat percakapan untuk`$LATEST` alias bot Anda atau untuk bot uji yang tersedia di konsol Amazon Lex. Setelah mengaktifkan log percakapan untuk alias,PostContent atauPostText operasi untuk alias tersebut mencatat ucapan teks atau audio dalam grup CloudWatch log Log yang dikonfigurasi atau bucket S3.

Topik

- Kebijakan IAM untuk Log Percakapan
- MengMengonfigurasi Log
- Mengenkripsi Log
- Melihat Log Teks di Amazon CloudWatch Logs
- Mengakses Log Audio di Amazon S3
- Memantau Status Log Percakapan dengan CloudWatch Metrik

Kebijakan IAM untuk Log Percakapan

Bergantung pada jenis pencatatan yang Anda pilih, Amazon Lex memerlukan izin untuk menggunakan bucket Amazon CloudWatch Logs dan Amazon Simple Storage Service (S3) untuk menyimpan log Anda. Anda harus membuatAWS Identity and Access Management peran dan izin untuk mengaktifkan Amazon Lex mengakses sumber daya ini.

Membuat Kebijakan IAM Role dan Kebijakan 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 keAWS KMS kunci yang digunakan untuk mengenkripsi objek.

Anda dapat menggunakan IAMAWS Management Console, IAM API, atauAWS Command Line Interface untuk membuat peran dan kebijakan. Petunjuk ini menggunakanAWS CLI untuk membuat peran dan kebijakan. Untuk informasi tentang membuat kebijakan dengan konsol, lihat Membuat kebijakan pada tab JSON di Panduan Pengguna AWS Identity and Access Management.

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

Untuk membuat IAM role untuk log percakapan

1. Buat dokumen di direktori saat ini disebut**`LexConversationLogsAssumeRolePolicyDocument.json`**, tambahkan kode berikut ke dalamnya, dan menyimpannya. Dokumen kebijakan ini menambahkan Amazon Lex sebagai entitas tepercaya ke peran tersebut. Hal ini memungkinkan Lex untuk mengambil peran untuk mengirimkan log ke sumber daya yang dikonfigurasi untuk log percakapan.

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

2. DiAWS CLI, jalankan perintah berikut ini untuk membuat IAM role 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 menulis ke CloudWatch Log.

Untuk membuat kebijakan IAM untuk logging teks percakapan ke CloudWatch Log

1. Buat dokumen di direktori saat ini yang disebut**LexConversationLogsCloudWatchLogsPolicy.json**, tambahkan kebijakan IAM berikut untuk itu, dan simpan.

```
{
  "Version": "2012-10-17",
  "Statement": [
      {
          "Effect": "Allow",
          "Action": [
              "logs:CreateLogStream",
              "logs:PutLogEvents"
          ],
          "Resource": "arn:aws:logs:region:account-id:log-group:log-group-name:*"
      }
  ]
}
```

2. DalamAWS CLI, buat kebijakan IAM yang memberikan izin menulis ke grup CloudWatch log log.

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

3. Melampirkan kebijakan ke IAM role 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 menulis ke bucket.

Untuk membuat kebijakan IAM untuk logging audio ke bucket S3

1. Buat dokumen di direktori saat ini yang disebut**`LexConversationLogsS3Policy.json`**, tambahkan kebijakan berikut ke dalamnya, dan simpan.

```
{
  "Version": "2012-10-17",
  "Statement": [
      {
          "Effect": "Allow",
          "Action": [
              "s3:PutObject"
          ],
          "Resource": "arn:aws:s3:::bucket-name/*"
      }
  ]
}
```

2. Di bagianAWS CLI, buat kebijakan IAM yang memberikan izin tulis ke bucket S3 Anda.

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

3. Melampirkan kebijakan pada 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
```

Pemberian Izin untuk Melewati Peran IAM

Saat Anda menggunakan konsolAWS Command Line Interface, atauAWS SDK untuk menentukan peran IAM yang akan digunakan untuk log percakapan, pengguna yang menentukan log percakapan peran IAM harus memiliki izin untuk meneruskan peran tersebut ke Amazon Lex. Agar pengguna dapatmeneruskan peran tersebut ke Amazon Lex, Anda harus memberikan`PassRole` izin kepada pengguna, atau kelompok dari Amazon Lex.

Kebijakan berikut menentukan izin untuk memberikan kepada pengguna, peran, atau grup. Anda dapat menggunakan`iam:AssociatedResourceArn` dan kunci`iam:PassedToService` kondisi untuk membatasi ruang lingkup izin. Untuk informasi selengkapnya, lihat Memberikan Izin Pengguna untuk Meneruskan Peran keAWS Layanan dan IAM serta Kunci KonteksAWS STS Kondisi di PanduanAWS Identity and Access Management Pengguna.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::account-id:role/role-name",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "lex.amazonaws.com"
                },
                "StringLike": {
                    "iam:AssociatedResourceARN": "arn:aws:lex:region:account-
id:bot:bot-name:bot-alias"
                }
            }
        }
    ]
}
```

MengMengonfigurasi Log

Anda mengaktifkan dan menonaktifkan log percakapan menggunakan konsol atau`conversationLogs` bidang`PutBotAlias` operasi. Anda dapat mengaktifkan atau menonaktifkan log audio, log teks, atau keduanya. Logging dimulai pada sesi bot baru. Perubahan pada pengaturan log tidak tercermin untuk sesi aktif.

Untuk menyimpan log teks, gunakan grup CloudWatch log Amazon Logs diAWS akun Anda. Anda dapat menggunakan grup log yang valid. Grup log harus berada di wilayah yang sama dengan bot Amazon Lex. Untuk informasi selengkapnya tentang membuat grup CloudWatch Log, lihat Bekerja dengan Grup Log dan Pengaliran Log di Panduan Pengguna Amazon CloudWatch Logs.

Untuk menyimpan log audio, gunakan bucket Amazon S3 diAWS akun Anda. Anda dapat menggunakan bucket S3 yang valid. Bucket harus berada di wilayah yang sama dengan bot Amazon

Lex. Untuk informasi selengkapnya tentang membuat bucket S3, lihat Membuat Bucket di Panduan Memulai Amazon Simple Storage Service.

Anda harus menyediakan peran IAM dengan kebijakan yang memungkinkan Amazon Lex menulis ke grup log atau bucket yang dikonfigurasi. Untuk informasi selengkapnya, lihat Membuat Kebijakan IAM Role dan Kebijakan untuk Log Percakapan.

Jika Anda membuat peran terkait layanan menggunakanAWS Command Line Interface, Anda harus menambahkan akhiran kustom ke peran menggunakan`custom-suffix` opsi sebagai berikut:

```
aws iam create-service-linked-role \
    --aws-service-name lex.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 tersebut.

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

Mengaktifkan Log

Untuk mengaktifkan log menggunakan konsol

1. Buka konsol Amazon Lex https://console.aws.amazon.com/lex.
2. Dari daftar, pilih bot.
3. Pilih tab Pengaturan, dan kemudian dari menu kiri pilih Log percakapan.
4. Di daftar alias, pilih ikon pengaturan untuk alias yang ingin Anda konfigurasikan log percakapan.
5. Pilih apakah akan mencatat teks, audio, atau keduanya.
6. Untuk pencatatan teks, masukkan nama grup CloudWatch log log Amazon.

7. Untuk pencatatan audio, masukkan informasi bucket S3.
8. Tidak wajib. Untuk mengenkripsi log audio, pilihAWS KMS kunci yang akan digunakan untuk enkripsi.
9. Pilih IAM role dengan izin yang diperlukan.
10. Pilih Simpan untuk memulai pencatatan percakapan.

Mengaktifkan log teks menggunakan API

1. PanggilPutBotAlias operasi dengan entri di`logSettings` anggota`conversationLogs` lapangan
   - Atur`destination` anggota ke`CLOUDWATCH_LOGS`
   - Atur`logType` anggota ke`TEXT`
   - Mengatur`resourceArn` anggota ke Amazon Resource Name (ARN) dari grup CloudWatch Log yang merupakan tujuan untuk log
2. Tetapkan`iamRoleArn` anggota`conversationLogs` bidang ke Amazon Resource Name (ARN) dari IAM role yang memiliki izin yang diperlukan untuk mengaktifkan log percakapan pada sumber daya yang ditentukan.

Mengaktifkan log audio menggunakan API

1. PanggilPutBotAlias operasi dengan entri di`logSettings` anggota`conversationLogs` lapangan
   - Atur`destination` anggota ke`S3`
   - Atur`logType` anggota ke`AUDIO`
   - Mengatur`resourceArn` anggota ke ARN dari bucket Amazon S3 tempat log audio disimpan
   - Tidak wajib. Untuk mengenkripsi log audio denganAWS KMS kunci tertentu, atur`kmsKeyArn` anggota ARN kunci yang digunakan untuk enkripsi.
2. Tetapkan`iamRoleArn` anggota`conversationLogs` bidang ke Amazon Resource Name (ARN) dari IAM role yang memiliki izin yang diperlukan untuk mengaktifkan log percakapan pada sumber daya yang ditentukan.

Menonaktifkan Log Percakapan

Untuk mematikan log menggunakan konsol

1. Buka konsol Amazon Lex https://console.aws.amazon.com/lex.
2. Dari daftar, pilih bot.
3. Pilih tab Pengaturan, dan kemudian dari menu kiri pilih Log percakapan.
4. Di daftar alias, pilih ikon pengaturan untuk alias yang ingin Anda konfigurasikan log percakapan.
5. Kosongkan cek dari teks, audio, atau keduanya untuk menonaktifkan pencatatan.
6. Pilih Simpan untuk menghentikan pencatatan percakapan.

Untuk menonaktifkan log menggunakan API

- PanggilPutBotAlias operasi tanpaconversationLogs lapangan.

Untuk menonaktifkan log teks menggunakan API

- • Jika Anda mencatat audio
  - PanggilPutBotAlias operasi denganlogSettings entri hanya untukAUDIO.
  - Panggilan kePutBotAlias operasi tidak boleh memilikilogSettings entri untukTEXT.
  - Jika Anda tidak mencatat audio
    - PanggilPutBotAlias operasi tanpaconversationLogs lapangan.

Menonaktifkan log audio menggunakan API

- • Jika Anda mencatat teks
  - PanggilPutBotAlias operasi denganlogSettings entri hanya untukTEXT.
  - Panggilan kePutBotAlias operasi tidak boleh memilikilogSettings entri untukAUDIO.
  - Jika Anda tidak mencatat teks
    - PanggilPutBotAlias operasi tanpaconversationLogs lapangan.

Mengenkripsi Log

Anda dapat menggunakan enkripsi untuk membantu melindungi konten log percakapan Anda. Untuk log teks dan audio, Anda dapat menggunakan CMK yang dikelolaAWS KMS pelanggan untuk mengenkripsi data di grup CloudWatch log Log dan bucket S3 Anda.

> **Note**
>
> Amazon Lex hanya mendukung CMK simetris. Jangan gunakan CMK asimetris untuk mengenkripsi data Anda.

Anda mengaktifkan enkripsi menggunakanAWS KMS kunci pada grup CloudWatch log Log yang digunakan Amazon Lex untuk log teks. Anda tidak dapat memberikanAWS KMS kunci dalam pengaturan log untuk mengaktifkanAWS KMS enkripsi grup log Anda. Untuk informasi selengkapnya, lihat Mengenkripsi Data Log di CloudWatch Log MenggunakanAWS KMS di Panduan Pengguna Amazon CloudWatch Logs.

Untuk log audio, Anda menggunakan enkripsi default pada bucket S3 atau menentukanAWS KMS kunci untuk mengenkripsi objek audio Anda. Bahkan jika bucket S3 Anda menggunakan enkripsi default, Anda masih dapat menentukanAWS KMS kunci yang berbeda untuk mengenkripsi objek audio Anda. Untuk informasi selengkapnya, lihat Enskripsi Default Amazon S3 S3 Default untuk S3 Buckets di Panduan Developer Amazon Simple Storage Service.

Amazon Lex memerlukanAWS KMS izin jika Anda memilih untuk mengenkripsi log audio Anda. Anda perlu melampirkan kebijakan tambahan ke peran IAM yang digunakan untuk log percakapan. Jika Anda menggunakan enkripsi default pada bucket S3, kebijakan Anda harus memberikan akses keAWS KMS kunci yang dikonfigurasi untuk bucket tersebut. Jika Anda menentukanAWS KMS kunci dalam pengaturan log audio Anda, Anda harus memberikan akses ke kunci itu.

Jika Anda belum membuat peran untuk log percakapan, lihatKebijakan IAM untuk Log Percakapan.

Untuk membuat kebijakan IAM untuk menggunakanAWS KMS kunci untuk mengenkripsi log audio

1. Buat dokumen di direktori saat ini yang disebut**`LexConversationLogsKMSPolicy.json`**, tambahkan kebijakan berikut ke dalamnya, dan simpan.

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

```
        {
            "Effect": "Allow",
            "Action": [
                "kms:GenerateDataKey"
            ],
            "Resource": "kms-key-arn"
        }
    ]
}
```

2. DalamAWS CLI, buat kebijakan IAM yang memberikan izin untuk menggunakanAWS KMS kunci untuk mengenkripsi log audio.

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

3. Melampirkan kebijakan pada peran yang Anda buat untuk log percakapan.

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

Melihat Log Teks di Amazon CloudWatch Logs

Amazon Lex menyimpan log teks untuk percakapan Anda di Amazon CloudWatch Logs. Untuk melihat log, Anda dapat menggunakan konsol CloudWatch Log atau API. Untuk informasi selengkapnya, lihat Cari Data Log Menggunakan Pola Filter dan CloudWatch Sintaks Kueri Wawasan CloudWatch Log di Panduan Pengguna Amazon Logs.

Untuk melihat log menggunakan konsol Amazon Lex

1. Buka konsol Amazon Lex https://console.aws.amazon.com/lex.
2. Dari daftar, pilih bot.
3. Pilih tab Pengaturan, lalu dari menu kiri pilih Log percakapan.
4. Pilih tautan di bawah Log teks untuk melihat log alias di CloudWatch konsol.

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 menemukan awalan aliran log untuk log Anda di konsol Amazon Lex atau dengan menggunakanGetBotAlias operasi.

Entri log untuk ucapan pengguna ada 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.

```
{
   "messageVersion": "1.0",
   "botName": "bot name",
   "botAlias": "bot alias",
   "botVersion": "bot version",
   "inputTranscript": "text used to process the request",
   "botResponse": "response from the bot",
   "intent": "matched intent",
   "nluIntentConfidence": "number",
   "slots": {
       "slot name": "slot value",
       "slot name": null,
       "slot name": "slot value"
       ...
   },
   "alternativeIntents": [
       {
           "name": "intent name",
           "nluIntentConfidence": "number",
           "slots": {
               "slot name": slot value,
               "slot name": null,
               "slot name": slot value
               ...
           }
       },
       {
           "name": "intent name",
           "nluIntentConfidence": number,
           "slots": {}
       }
   ],
   "developerOverride": "true" | "false",
   "missedUtterance": true | false,
   "inputDialogMode": "Text" | "Speech",
   "requestId": "request ID",
```

```
  "s3PathForAudio": "S3 path to audio file",
  "userId": "user ID",
  "sessionId": "session ID",
  "sentimentResponse": {
      "sentimentScore": "{Positive: number, Negative: number, Neutral: number,
Mixed: number}",
      "sentimentLabel": "Positive" | "Negative" | "Neutral" | "Mixed"
  },
  "slotToElicit": "slot name",
  "dialogState": "ElicitIntent" | "ConfirmIntent" | "ElicitSlot" | "Fulfilled" |
"ReadyForFulfillment" | "Failed",
  "responseCard": {
      "genericAttachments": [
          ...
      ],
      "contentType": "application/vnd.amazonaws.card.generic",
      "version": 1
   },
  "locale": "locale",
  "timestamp": "ISO 8601 UTC timestamp",
  "kendraResponse": {
     "totalNumberOfResults": number,
     "resultItems": [
         {
             "id": "query ID",
             "type": "DOCUMENT" | "QUESTION_ANSWER" | "ANSWER",
             "additionalAttributes": [
                 {
                    ...
                 }
             ],
             "documentId": "document ID",
             "documentTitle": {
                 "text": "title",
                 "highlights": null
             },
             "documentExcerpt": {
                 "text": "text",
                 "highlights": [
                     {
                         "beginOffset": number,
                         "endOffset": number,
                         "topAnswer": true | false
                     }
```

```
                    ]
                },
                "documentURI": "URI",
                "documentAttributes": []
            }
        ],
        "facetResults": [],
        "sdkResponseMetadata": {
            "requestId": "request ID"
        },
        "sdkHttpMetadata": {
            "httpHeaders": {
                "Content-Length": "number",
                "Content-Type": "application/x-amz-json-1.1",
                "Date": "date and time",
                "x-amzn-RequestId": "request ID"
            },
            "httpStatusCode": 200
        },
        "queryId": "query ID"
    },
    "sessionAttributes": {
        "attribute name": "attribute value"
        ...
     },
    "requestAttributes": {
        "attribute name": "attribute value"
        ...
     }
}
```

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 hadir ketika`AMAZON.KendraSearchIntent` membuat permintaan untuk mencari indeks Amazon Kendra.
- `developerOverride`Bidang true 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`messageVersion`. Kode Anda seharusnya tidak melempar kesalahan jika bidang baru ada.

Anda harus memiliki peran dan kebijakan yang ditetapkan untuk mengaktifkan Amazon Lex menulis ke CloudWatch Log. Untuk informasi selengkapnya, lihat Kebijakan IAM untuk Log Percakapan.

Mengakses Log Audio di Amazon S3

Amazon Lex menyimpan log audio untuk percakapan Anda dalam bucket S3.

Untuk mengakses log audio menggunakan konsol

1. Buka konsol Amazon Lex https://console.aws.amazon.com/lex.
2. Dari daftar, pilih bot.
3. Pilih tab Pengaturan, lalu dari menu kiri pilih Log percakapan.
4. Pilih tautan di bawah Log audio untuk mengakses log alias di konsol Amazon S3.

Anda juga dapat menggunakan konsol Amazon S3 atau API untuk mengakses log audio. Anda dapat melihat key prefix objek S3 dari file audio di konsol Amazon Lex, atau di`resourcePrefix` bidang dalam respons`GetBotAlias` operasi.

Memantau Status Log Percakapan dengan CloudWatch Metrik

Gunakan Amazon CloudWatch untuk memantau metrik pengiriman log percakapan Anda. Anda dapat menyetel alarm pada metrik sehingga Anda mengetahui masalah dengan pencatatan log jika harus terjadi.

Amazon Lex menyediakan empat metrik dalamAWS/Lex namespace untuk log percakapan:

- ConversationLogsAudioDeliverySuccess
- ConversationLogsAudioDeliveryFailure
- ConversationLogsTextDeliverySuccess
- ConversationLogsTextDeliveryFailure

Untuk informasi selengkapnya, lihat CloudWatch Metrik untuk Log Percakapan.

Metrik keberhasilan menunjukkan bahwa Amazon Lex telah berhasil menulis log audio atau teks Anda ke tujuan mereka.

Metrik kegagalan menunjukkan bahwa Amazon Lex tidak dapat mengirimkan log audio atau teks ke tujuan yang ditentukan. Biasanya, ini adalah kesalahan konfigurasi. Ketika metrik kegagalan Anda berada di atas nol, periksa hal berikut:

- Pastikan Amazon Lex adalah entitas tepercaya untuk peran IAM.
- Untuk pencatatan teks, pastikan bahwa grup CloudWatch log log ada. Untuk pencatatan audio, pastikan bucket S3 ada.
- Pastikan peran IAM yang digunakan Amazon Lex 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 dan milik akun Anda.
- Jika Anda menggunakanAWS KMS kunci untuk enkripsi S3, pastikan tidak ada kebijakan yang mencegah Amazon Lex menggunakan kunci Anda dan pastikan bahwa peran IAM yang Anda berikan memilikiAWS KMS izin yang diperlukan. Untuk informasi selengkapnya, lihat Kebijakan IAM untuk Log Percakapan.

Mengelola Sesi Dengan Amazon Lex API

Ketika pengguna memulai percakapan dengan bot Anda, Amazon Lex membuatsesi. Informasi yang dipertukarkan antara aplikasi Anda dan Amazon Lex membentuk status sesi untuk percakapan. Ketika Anda membuat permintaan, sesi diidentifikasi dengan kombinasi nama bot dan pengenal pengguna yang Anda tentukan. Untuk informasi selengkapnya tentang pengenal pengguna, lihatuserIdbidang diPostContentatauPostTextOperasi.

Respon dari operasi sesi mencakup pengenal sesi unik yang mengidentifikasi sesi tertentu dengan pengguna. Anda dapat menggunakan pengenal ini selama pengujian atau untuk membantu memecahkan masalah bot Anda.

Anda dapat memodifikasi status sesi yang dikirim antara aplikasi dan bot Anda. Misalnya, Anda dapat membuat dan memodifikasi atribut sesi yang berisi informasi khusus tentang sesi, dan Anda dapat mengubah alur percakapan dengan mengatur konteks dialog untuk menafsirkan ucapan berikutnya.

Ada dua cara untuk memperbarui status sesi. Yang pertama adalah menggunakan fungsi Lambda dengan`PostContent`atau`PostText`operasi yang disebut setelah setiap pergantian percakapan. Untuk informasi selengkapnya, lihat Menggunakan Fungsi Lambda. Yang lainnya adalah menggunakan API runtime Amazon Lex dalam aplikasi Anda untuk membuat perubahan pada status sesi.

API runtime Amazon Lex menyediakan operasi yang memungkinkan Anda mengelola informasi sesi untuk percakapan dengan bot Anda. Operasi adalahPutSessionOperasi,GetSessionoperasi, danDeleteSessionOperasi. Anda menggunakan operasi ini untuk mendapatkan informasi tentang status sesi pengguna Anda dengan bot Anda, dan memiliki kontrol halus atas status.

Menggunakan`GetSession`operasi ketika Anda ingin mendapatkan keadaan saat sesi. Operasi mengembalikan status sesi saat ini, termasuk status dialog dengan pengguna Anda, atribut sesi apa pun yang telah ditetapkan dan nilai slot untuk tiga intent terakhir yang berinteraksi pengguna.

Parameter`PutSession`operasi memungkinkan Anda untuk langsung memanipulasi keadaan sesi saat ini. Anda dapat mengatur jenis tindakan dialog yang akan dilakukan bot berikutnya. Ini memberi Anda kontrol atas aliran percakapan dengan bot. Mengatur tindakan dialog`type`bidang untuk`Delegate`untuk memiliki Amazon Lex menentukan tindakan berikutnya untuk bot.

Anda dapat menggunakan`PutSession`operasi untuk membuat sesi baru dengan bot dan mengatur maksud bahwa bot harus mulai dengan. Anda juga dapat menggunakan`PutSession`operasi untuk mengubah dari satu maksud ke yang lain. Ketika Anda membuat sesi atau mengubah maksud Anda juga dapat mengatur status sesi, seperti nilai slot dan atribut sesi. Setelah intent baru selesai, Anda memiliki opsi untuk memulai ulang maksud sebelumnya. Anda dapat menggunakan`GetSession`operasi untuk mendapatkan status dialog dari intent sebelumnya dari Amazon Lex dan menggunakan informasi untuk mengatur status dialog maksud.

Respons dari`PutSession`operasi berisi informasi yang sama dengan`PostContent`Operasi. Anda dapat menggunakan informasi ini untuk meminta pengguna untuk informasi berikutnya, seperti yang Anda lakukan dengan respon dari`PostContent`Operasi.

MenggunakanDeleteSessionoperasi untuk menghapus sesi yang ada dan memulai dari awal dengan sesi baru. Misalnya, ketika Anda menguji bot Anda, Anda dapat menggunakanDeleteSessionoperasi untuk menghapus sesi tes dari bot Anda.

Operasi sesi bekerja dengan fungsi Lambda pemenuhan Anda. Misalnya, jika fungsi Lambda Anda kembaliFailedsebagai negara pemenuhan Anda dapat menggunakanPutSessionoperasi untuk mengatur jenis tindakan dialog keclosedanfulfillmentStatekepadaReadyForFulfillmentuntuk mencoba lagi langkah pemenuhan.

Berikut adalah beberapa hal yang bisa Anda lakukan dengan operasi sesi:

- Minta bot memulai percakapan alih-alih menunggu pengguna.
- Beralih maksud selama percakapan.
- Kembali ke maksud sebelumnya.
- Memulai atau memulai ulang percakapan di tengah interaksi.
- Validasi nilai slot dan minta bot re-prompt untuk nilai yang tidak valid.

Masing-masing dijelaskan lebih lanjut di bawah ini.

Mengalihkan Maksud

Anda dapat menggunakanPutSessionoperasi untuk beralih dari satu maksud ke yang lain. Anda juga dapat menggunakannya untuk beralih kembali ke maksud sebelumnya. Anda dapat menggunakanPutSessionoperasi untuk mengatur atribut sesi atau nilai slot untuk maksud baru.

- MemanggilPutSessionOperasi. Setel nama intent ke nama intent baru dan atur tindakan dialogDelegate. Anda juga dapat mengatur nilai slot atau atribut sesi yang diperlukan untuk maksud baru.
- Amazon Lex akan memulai percakapan dengan pengguna menggunakan maksud baru.

Melanjutkan Intent Sebelumnya

Untuk melanjutkan maksud sebelumnya, Anda menggunakanGetSessionoperasi untuk mendapatkan ringkasan maksud, dan kemudian menggunakanPutSessionoperasi untuk mengatur maksud ke status dialog sebelumnya.

- MemanggilGetSessionOperasi. Respons dari operasi mencakup ringkasan status dialog dari tiga intent terakhir yang berinteraksi pengguna.
- Menggunakan informasi dari ringkasan intent, panggilPutSessionOperasi. 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, 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 serupa dengan ini:

- Aplikasi Anda menentukan bahwa pengguna perlu berbicara dengan perwakilan layanan pelanggan.
- MenggunakanGetSessionoperasi untuk mendapatkan keadaan dialog maksud saat ini.
- Perwakilan layanan pelanggan berbicara kepada pengguna dan menyelesaikan masalah.
- MenggunakanPutSessionoperasi untuk mengatur keadaan dialog maksud. Ini mungkin termasuk pengaturan nilai slot, pengaturan atribut sesi, atau mengubah maksud.
- Bot melanjutkan percakapan dengan pengguna.

Anda dapat menggunakanPutSessionoperasicheckpointLabelparameter untuk memberi label maksud sehingga Anda dapat menemukannya nanti. Misalnya, bot yang meminta pelanggan untuk informasi mungkin masuk keWaitingniat sementara pelanggan mengumpulkan informasi. Bot membuat label pos pemeriksaan untuk maksud saat ini dan kemudian memulaiWaitingniat. Ketika pelanggan mengembalikan bot dapat menemukan maksud sebelumnya menggunakan label pos pemeriksaan dan beralih kembali.

Maksud harus ada direcentIntentSummaryViewstruktur dikembalikan olehGetSessionOperasi. Jika Anda menentukan label pos pemeriksaan diGetSessionpermintaan operasi, itu akan mengembalikan maksimal tiga maksud dengan label pos pemeriksaan itu.

- MenggunakanGetSessionOperasi untuk mendapatkan tahapan sesi saat ini.
- MenggunakanPutSessionoperasi untuk menambahkan label pos pemeriksaan ke intent terakhir. Jika perlu Anda dapat menggunakan iniPutSessionpanggilan untuk beralih ke maksud yang berbeda.

- Ketika saatnya untuk beralih kembali ke intent berlabel, hubungi`GetSession`operasi untuk mengembalikan daftar intent terbaru. Anda dapat menggunakan`checkpointLabelFilter`parameter sehingga Amazon Lex hanya mengembalikan maksud dengan label pos pemeriksaan yang ditentukan.

Memulai Sesi Baru

Jika Anda ingin memiliki 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 dapat mengatakan 'Pesan minuman' atau 'Pesan pizza. '”
- Memanggil`PutSession`Operasi. Atur nama maksud ke nama maksud selamat datang Anda dan atur tindakan dialog`Delegate`.
- Amazon Lex akan merespons dengan prompt dari maksud selamat datang Anda untuk memulai percakapan dengan pengguna Anda.

Memvalidasi Nilai Slot

Anda dapat memvalidasi tanggapan bot Anda menggunakan aplikasi klien Anda. Jika respons tidak valid, Anda dapat menggunakan`PutSession`operasi untuk mendapatkan respon baru dari pengguna Anda. Misalnya, anggaplah bahwa bot pemesanan bunga Anda hanya bisa menjual tulip, mawar, dan bunga lili. Jika perintah pengguna anyelir, aplikasi Anda dapat melakukan hal berikut:

- Periksa nilai slot yang dikembalikan dari`PostText`atau`PostContent`tanggapan.
- Jika nilai slot tidak valid, hubungi`PutSession`Operasi. Aplikasi Anda harus menghapus nilai slot, mengatur`slotToElicit`lapangan, dan mengatur`dialogAction.type`nilai untuk`elicitSlot`. Opsional, Anda dapat mengatur`message`dan`messageFormat`bidang jika Anda ingin mengubah pesan yang Amazon Lex gunakan untuk mendapatkan nilai slot.

Opsi Deployment bot Opsi Deployment Bot

Saat ini, Amazon Lex menyediakan opsi penyebaran bot berikut:

- AWS Mobile SDK — Anda dapat membuat aplikasi seluler yang berkomunikasi dengan Amazon Lex menggunakan AWS Mobile SDK.
- Facebook Messenger - Anda dapat mengintegrasikan halaman Facebook Messenger Anda dengan bot Amazon Lex Anda sehingga pengguna akhir di Facebook dapat berkomunikasi dengan bot. Dalam implementasi saat ini, integrasi ini hanya mendukung pesan input teks.
- Slack - Anda dapat mengintegrasikan bot Amazon Lex Anda dengan aplikasi perpesanan Slack.
- Twilio - Anda dapat mengintegrasikan bot Amazon Lex Anda dengan Twilio Simple Messaging Service (SMS).

Sebagai contoh, lihat Menerapkan Bot Amazon Lex.

Intent dan Jenis Slot Bawaan

Untuk membuatnya lebih mudah untuk membuat bot, Amazon Lex memungkinkan Anda untuk menggunakan maksud bawaan standar dan jenis slot.

Topik

- Maksud bawaan
- Jenis Slot Bawaan

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 panel navigasi, pilih plus (+) di sebelah Maksud.
4. Untuk Tambah maksud, pilih Cari maksud yang ada.
5. Di kotak Search intents, ketikkan nama intent bawaan untuk ditambahkan ke bot Anda.
6. Untuk Salin maksud bawaan, beri nama maksud, lalu pilih Tambah.
7. Konfigurasikan intent seperti yang diperlukan untuk bot Anda.

Topik

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

> **Note**
>
> Untuk lokal bahasa Inggris (AS) (en-AS), Amazon Lex mendukung maksud dari intent bawaan standar Alexa. Untuk daftar intent bawaan, lihat Intent Bawaan Standar di Alexa Skills Kit.
> Amazon Lex tidak mendukung maksud berikut:
>
> - `AMAZON.YesIntent`
> - `AMAZON.NoIntent`
> - Maksud di Built-in Intent Library di Alexa Skills Kit

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 untuk memanggil intent fallback. Misalnya, jika input pengguna “Saya ingin memesan permen” tidak cocok dengan maksud di `OrderFlowers` bot Anda, Amazon Lex memanggil maksud fallback untuk menangani respons.

Anda menambahkan intent fallback dengan menambahkan tipe intent bawaan ke `AMAZON.FallbackIntent` bot Anda. Anda dapat menentukan intent menggunakan PutBot operasi atau dengan memilih intent dari daftar intent bawaan di konsol.

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. Misalnya, jika jumlah maksimum upaya untuk menentukan maksud adalah 2, bot mengembalikan prompt klarifikasi bot dua kali sebelum menjalankan maksud fallback.

Amazon Lex 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 ambigu dan Amazon Lex tidak dapat menentukan maksud mana yang akan dipanggil.

Maksud fallback dipanggil saat:

- Bot tidak mengenali input pengguna sebagai maksud setelah jumlah percobaan yang dikonfigurasi untuk klarifikasi saat percakapan dimulai.
- 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 dapat menggunakan yang berikut ini dengan maksud fallback:

- Fungsi Lambda pemenuhan
- Pernyataan kesimpulan
- Prompt tindak lanjut

Anda tidak dapat menambahkan yang berikut ini ke intent fallback:

- Ucapan
- Slot
- Fungsi Lambda inisialisasi dan validasi
- Prompt konfirmasi

Jika Anda telah mengonfigurasi pernyataan pembatalan dan maksud fallback untuk bot, Amazon Lex menggunakan maksud fallback. Jika Anda membutuhkan bot Anda untuk memiliki pernyataan pembatalan, Anda dapat menggunakan fungsi pemenuhan untuk maksud fallback untuk memberikan perilaku yang sama seperti pernyataan pembatalan. Untuk informasi lebih lanjut, lihat `abortStatement` parameter PutBot operasi.

Menggunakan Prompt Klarifikasi

Jika Anda memberikan bot Anda dengan prompt klarifikasi, prompt digunakan untuk meminta maksud yang valid dari pengguna. Prompt klarifikasi akan diulang berapa kali yang Anda konfigurasikan. Setelah itu maksud fallback akan dipanggil.

Jika Anda tidak menetapkan prompt klarifikasi saat membuat bot dan pengguna tidak memulai percakapan dengan maksud yang valid, Amazon Lex segera memanggil maksud fallback Anda.

Saat Anda menggunakan intent fallback tanpa prompt klarifikasi, Amazon Lex tidak memanggil fallback dalam keadaan berikut:

- Saat pengguna merespons prompt tindak lanjut tetapi tidak memberikan maksud. Misalnya, sebagai tanggapan atas prompt tindak lanjut yang mengatakan “Apakah Anda ingin hal lain hari ini?” , pengguna mengatakan “Ya.” Amazon Lex mengembalikan pengecualian 400 Permintaan

Buruk karena tidak memiliki prompt klarifikasi untuk dikirim ke pengguna untuk mendapatkan maksud.

- Saat menggunakan AWS Lambda fungsi, Anda mengembalikan tipe `ElicitIntent` dialog. Karena Amazon Lex tidak memiliki prompt klarifikasi untuk mendapatkan maksud dari pengguna, Amazon Lex mengembalikan pengecualian 400 Permintaan Buruk.
- Saat menggunakan `PutSession` operasi, Anda mengirim jenis `ElicitIntent` dialog. Karena Amazon Lex tidak memiliki prompt klarifikasi untuk mendapatkan maksud dari pengguna, Amazon Lex mengembalikan pengecualian 400 Permintaan Buruk.

Menggunakan Fungsi Lambda dengan Maksud Fallback

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

- Mengembalikan informasi maksud ke aplikasi klien.
- Memanggil fungsi Lambda pemenuhan. Ini memanggil fungsi dengan variabel sesi yang diatur untuk sesi.

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

Jika Anda menggunakan fungsi Lambda pemenuhan dalam 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.

Anda dapat melakukan tindakan apa pun dalam fungsi Lambda intent fallback yang dapat Anda lakukan dalam fungsi pemenuhan untuk maksud lainnya. Untuk informasi selengkapnya tentang membuat fungsi pemenuhan menggunakanAWS Lambda, lihatMenggunakan Fungsi Lambda.

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 tidak dapat menyimpulkan maksud pengguna setelah jumlah percobaan yang dikonfigurasi, Amazon Lex 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.

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:

- help
- 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 tidak dapat menentukan tindakan selanjutnya dalam percakapan dengan pengguna, Amazon Lex 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 akan memanggil intent kapan pun intent tidak dapat menentukan ucapan pengguna untuk slot atau intent. Misalnya, jika bot Anda memunculkan respons untuk jenis slot yang disebut “pizza topping” dan pengguna mengatakan “Apa itu pizza? ,” Amazon Lex memanggil `AMAZON.KendraSearchIntent` untuk menangani pertanyaan itu. Jika tidak ada tanggapan dari Amazon Kendra, percakapan berlanjut seperti yang dikonfigurasi di bot.

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

1. Amazon Lex memanggil`AMAZON.KendraSearchIntent`. Maksudnya menyebut operasi Amazon `Query` Kendra.
2. Jika Amazon Kendra mengembalikan respons, Amazon Lex menampilkan hasilnya kepada pengguna.
3. Jika tidak ada tanggapan dari Amazon Kendra, Amazon Lex meminta kembali pengguna. Tindakan selanjutnya tergantung pada respons dari pengguna.
   - Jika respons dari pengguna berisi ucapan yang dikenali Amazon Lex, seperti mengisi nilai slot atau mengonfirmasi maksud, percakapan dengan pengguna dilanjutkan seperti yang dikonfigurasi untuk bot.
   - Jika respons dari pengguna tidak mengandung ucapan yang dikenali Amazon Lex, Amazon Lex membuat panggilan lain ke operasi. `Query`
4. Jika tidak ada respons setelah jumlah percobaan ulang yang dikonfigurasi, Amazon Lex 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 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 menggunakan filter dalam permintaan kueri.
- Tambahkan parameter kueri tambahan ke permintaan untuk mempersempit hasil pencarian menggunakan fungsi Lambda dialog 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 dialog Lambda. Anda dapat membuat permintaan kueri Amazon Kendra lengkap yang dikirimkan Amazon Lex. 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 ke Amazon Kendra, Anda dapat menentukan kueri di `kendraQueryRequestPayload` bidang di dialog Anda fungsi Lambda. Jika kueri tidak valid, Amazon Lex 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 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, 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 atau menggunakan peran khusus untuk memanggil operasi Amazon Kendra. `Query` Bila 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 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 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.

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

Menentukan Peran

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

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 AWS Identity and Access Management Pengguna.

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

Peran runtime yang perlu digunakan Amazon Lex 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, API IAM, 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

   ```
   {
       "Version": "2012-10-17",
       "Statement": [
           {
               "Effect": "Allow",
               "Action": [
                   "kendra:Query"
               ],
               "Resource": [
   ```

```
                "arn:aws:kendra:region:account:index/index ID"
            ]
        }
    ]
}
```

2. DalamAWS CLI, jalankan perintah berikut untuk membuat kebijakan IAM untuk menjalankan operasi Amazon `Query` Kendra.

```
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 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 agar bot Anda dapat 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 informasi selengkapnya tentang atribut sesi, lihatMengatur Atribut Sesi.

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`Filter 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 dalam pernyataan maksud. `conclusion` Maksud harus memiliki `conclusion` pernyataan kecuali fungsi Lambda pemenuhan menghasilkan pesan kesimpulan.

Amazon Kendra memiliki empat jenis tanggapan.

- `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.
- `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.

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

Menggunakan Hook Kode Pemenuhan untuk Respons

Setelah Amazon Lex mengirim 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 menggunakan `conclusion` pernyataan tersebut untuk tanggapan.

Contoh: Membuat Bot FAQ untuk Indeks Amazon Kendra

Contoh ini membuat bot Amazon Lex 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. Untuk membuat bot, Anda:

1. Buat bot yang akan berinteraksi dengan pelanggan Anda untuk mendapatkan jawaban dari bot Anda.
2. Buat maksud khusus. Bot Anda membutuhkan setidaknya satu maksud dengan setidaknya satu ucapan. Maksud ini memungkinkan bot Anda untuk membangun, tetapi tidak digunakan sebaliknya.
3. Tambahkan `KendraSearchIntent` intent ke bot Anda dan konfigurasikan agar berfungsi dengan indeks Amazon Kendra Anda.
4. Uji bot dengan mengajukan pertanyaan yang dijawab oleh dokumen yang disimpan dalam indeks Amazon Kendra Anda.

Sebelum Anda dapat menggunakan contoh ini, Anda perlu membuat indeks Amazon Kendra. Untuk informasi selengkapnya, lihat Memulai bucket (konsol) S3 di Panduan Pengembang Amazon Kendra.

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 Create (Buat).
4. Pilih Bot khusus. Konfigurasikan bot sebagai berikut:
   - Nama bot — Berikan bot nama yang menunjukkan tujuannya, seperti**`KendraTestBot`**.
   - Suara keluaran - Pilih Tidak Ada.
   - Waktu tunggu sesi — Masuk**`5`**.
   - Analisis sentimen - Pilih No.
   - COPPA — Pilih No.
   - Penyimpanan ucapan pengguna — Pilih Jangan simpan.
5. Pilih Create (Buat).

Agar berhasil membangun bot, Anda harus membuat setidaknya satu maksud dengan setidaknya satu contoh ucapan. Maksud ini diperlukan untuk membangun bot Amazon Lex Anda, tetapi tidak digunakan untuk respons FAQ. Ucapan untuk maksud tersebut tidak boleh berlaku untuk pertanyaan apa pun yang diajukan pelanggan Anda.

Untuk membuat maksud 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, beri maksud nama, seperti. **`RequiredIntent`**
4. Untuk ucapan Sampel, ketikkan ucapan, seperti. **`Required utterance`**
5. Pilih Simpan maksud.

Sekarang, buat maksud untuk mencari indeks Amazon Kendra dan pesan respons yang harus dikembalikan.

Untuk membuat AMAZON. KendraSearchIntent pesan maksud dan respons

1. Di panel navigasi, pilih plus (+) di sebelah Maksud.
2. Untuk Tambah maksud, pilih Cari maksud yang ada.
3. Di kotak Maksud pencarian, masukkan**`AMAZON.KendraSearchIntent`**, lalu pilih dari daftar.
4. Untuk Salin maksud bawaan, beri maksud nama, seperti**`KendraSearchIntent`**, lalu pilih Tambah.
5. Di editor maksud, pilih kueri Amazon Kendra untuk membuka opsi kueri.
6. Dari menu indeks Amazon Kendra, pilih indeks yang ingin Anda cari.
7. Di bagian Respons, tambahkan tiga pesan berikut:

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

8. Pilih Simpan maksud, lalu pilih Build untuk membangun bot.

Terakhir, gunakan jendela pengujian konsol untuk menguji respons dari bot Anda. Pertanyaan Anda harus berada di domain yang didukung indeks Anda.

Untuk menguji bot FAQ Anda

1. Di jendela pengujian konsol, ketikkan pertanyaan untuk indeks Anda.
2. Verifikasi jawabannya di bagian respons jendela pengujian.
3. Untuk mengatur ulang jendela pengujian untuk pertanyaan lain, pilih Hapus riwayat obrolan.

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 GetSession operasi untuk mengambil data maksud saat melanjutkan maksud saat ini.

Ucapan umum:

- jeda
- jeda itu

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 GetSession operasi 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
- mati
- diam

Jenis Slot Bawaan

Amazon Lex mendukung tipe slot built-in 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. Jenis slot bawaan tidak memiliki versi.

Jenis Slot	Deskripsi Singkat	Lokal yang Didukung	
Amazon.Bandara	Mengakui kata-kata yang mewakili bandara.	Semua lokal	
AMAZON.AlphaNumeric	Mengenali kata-kata yang terdiri dari huruf dan angka.	Semua lokal kecuali Korea (Ko-kr)	
Amazon.kota	Mengakui kata-kata yang mewakili kota.	Semua lokal	

Jenis Slot	Deskripsi Singkat	Lokal yang Didukung
Amazon.negara	Mengakui kata-kata yang mewakili suatu negara.	Semua lokal
AMAZON.DATE	Mengenali kata-kata yang mewakili tanggal dan mengubahn ya menjadi format standar.	Semua lokal
AMAZON.DURASI	Mengenali kata-kata yang mewakili durasi dan mengubahn ya menjadi format standar.	Semua lokal
AMAZON. EmailAddr ess	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

Jenis Slot	Deskripsi Singkat	Lokal yang Didukung
Amazon.persentase	Mengenali kata-kata yang mewakili persentase dan mengubahnya menjadi angka dan tanda persen (%).	Semua lokal
AMAZON. PhoneNumber	Mengenali kata-kata yang mewakili nomor telepon dan mengubahn ya menjadi string numerik.	Semua lokal
AMAZON. SpeedUnit	Mengenali kata-kata yang mewakili unit kecepatan dan mengubahnya menjadi singkatan standar.	Inggris (AS) (en-US)
Amazon.state	Mengakui kata-kata yang mewakili suatu negara.	Semua lokal
AMAZON. StreetNam e	Mengenali kata-kata yang mewakili nama jalan.	Semua lokal kecuali Inggris (AS) (en-US)
AMAZON.TIME	Mengenali kata-kata yang menunjukk an waktu dan mengubahnya menjadi format waktu.	Semua lokal

Jenis Slot	Deskripsi Singkat	Lokal yang Didukung	
AMAZON. WeightUnit	Mengenali kata-kata yang mewakili satuan berat dan mengubahnya menjadi singkatan standar	Inggris (AS) (en-US)	

> **Note**
>
> Untuk lokal bahasa Inggris (AS) (en-AS), Amazon Lex mendukung jenis slot dari Alexa Skill Kit. Untuk daftar jenis slot bawaan yang tersedia, lihat Referensi Jenis Slot dalam dokumentasi Alexa Skills Kit.
>
> - Amazon Lex tidak mendukung `AMAZON.LITERAL` atau jenis slot `AMAZON.SearchQuery` bawaan.

Amazon.Bandara

Menyediakan daftar bandara. Contohnya termasuk:

- Bandara Internasional John F. Kennedy
- Bandara Melbourne

AMAZON. AlphaNumeric

Mengenali string yang terdiri dari huruf dan angka, seperti. **`APQ123`**

Jenis slot ini tidak tersedia di lokal Korea (KO-KR).

Anda dapat menggunakan jenis `AMAZON.AlphaNumeric` slot untuk string yang berisi:

- Karakter abjad, seperti **`ABC`**
- Karakter numerik, seperti **`123`**
- Kombinasi karakter alfanumerik, seperti **`ABC123`**

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 Inggris Raya atau Kanada
- Nomor SIM
- Nomor identifikasi kendaraan

Gunakan ekspresi reguler standar. Amazon Lex mendukung karakter berikut dalam ekspresi reguler:

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

Amazon Lex 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 di AMAZON. AlphaNumeric Jenis 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 PutSlotType operasi. Ekspresi reguler divalidasi saat Anda menyimpan jenis slot. Jika ekspresi tidak valid, Amazon Lex mengembalikan pesan kesalahan.

Saat Anda menggunakan ekspresi reguler dalam jenis slot, Amazon Lex memeriksa input ke slot jenis itu terhadap ekspresi reguler. Jika input cocok dengan ekspresi, nilai diterima untuk slot. Jika input tidak cocok, Amazon Lex meminta pengguna untuk mengulangi input.

Amazon.kota

Menyediakan daftar kota lokal dan dunia. Jenis slot mengenali variasi umum nama kota. Amazon Lex tidak mengonversi dari variasi ke nama resmi.

Contoh:

- New York
- Reykjavik
- Tokyo
- Versailles

Amazon.negara

Nama-nama negara di seluruh dunia. Contoh:

- Australia
- Germany
- Jepang
- Amerika Serikat
- Uruguay

AMAZON.DATE

Mengonversi kata-kata yang mewakili tanggal menjadi format tanggal.

Tanggal diberikan sesuai maksud Anda dalam format tanggal ISO-8601. 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 minggu tertentu, seperti “minggu ini,” atau “minggu depan,” dikonversi ke tanggal hari pertama dalam seminggu. Dalam format ISO-8601, minggu dimulai pada hari Senin dan berakhir pada hari Minggu. Misalnya, jika hari ini 2020-11-25, “minggu depan” dikonversi menjadi. `2020-11-30`
- Ucapan yang dipetakan menjadi satu bulan, 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`
- Ucapan yang dipetakan menjadi satu tahun, 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`

AMAZON.DURASI

Mengkonversi kata-kata yang menunjukkan durasi menjadi durasi numerik.

Durasi diselesaikan ke format berdasarkan format durasi ISO-8601,. `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 (+).

AMAZON. FirstName

Nama depan yang umum digunakan. Jenis slot ini mengenali nama formal dan nama panggilan informal. Nama yang dikirim ke intent Anda adalah nilai yang dikirim oleh pengguna. Amazon Lex tidak mengonversi dari nama panggilan ke nama resmi.

Untuk nama depan yang terdengar sama tetapi dieja berbeda, Amazon Lex mengirimkan maksud Anda satu bentuk umum.

Dalam bahasa Inggris (AS) (en-US) lokal, gunakan nama slot Amazon.us_first_name.

Contoh:

- Emily
- John
- Sophie

AMAZON. LastName

Nama belakang yang umum digunakan. Untuk nama yang terdengar sama yang dieja berbeda, Amazon Lex mengirimkan maksud Anda satu bentuk umum.

Dalam bahasa Inggris (AS) (en-US) lokal, gunakan nama slot Amazon.us_last_name.

Contoh:

- Brosky
- Dasher
- Evers
- Parres
- Welt

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	Response
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

Amazon.persentase

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	Response
50 persen	50%
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	Nomor 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	7 digit nomor telepon tanpa kode internasi onal atau kode area	555-1212	`5551212`

AMAZON. SpeedUnit

Mengubah kata-kata yang mewakili satuan kecepatan menjadi singkatan yang sesuai. Misalnya, “mil per jam” dikonversi menjadi`mph`.

Jenis slot ini hanya tersedia di bahasa Inggris (AS) (en-AS) lokal.

Contoh berikut menunjukkan bagaimana jenis `AMAZON.SpeedUnit` slot menangkap unit kecepatan.

Satuan kecepatan	Singkatan
mil per jam, mph, MPH, m/jam	mph
kilometer per jam, km per jam, kmph, KMPH, km/jam	kmph
meter per detik, mps, MPS, m/s	anggota parlemen
mil laut per jam, simpul, simpul	simpul

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.

Jenis slot ini tidak tersedia di lokal Inggris (AS) (en-AS).

Contoh:

- Jalan Canberra
- Jalan Depan
- Jalan Pasar

AMAZON.TIME

Mengubah kata-kata yang mewakili waktu menjadi nilai waktu. Termasuk resolusi untuk waktu yang ambigu. Saat pengguna memasukkan waktu yang ambigu, Amazon Lex menggunakan `slotDetails` atribut peristiwa Lambda untuk meneruskan resolusi untuk waktu 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 dalam `slots` peta adalah`null`, dan `slotDetails` entitas berisi dua kemungkinan resolusi saat itu. Amazon Lex memasukkan yang berikut ini ke dalam fungsi Lambda:

```
"slots": {
   "deliveryTime": null
```

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

Ketika pengguna merespons dengan waktu yang tidak ambigu, Amazon Lex mengirimkan waktu ke fungsi Lambda Anda dalam atribut `slots` peristiwa Lambda dan atribut kosong. `slotDetails` Misalnya, jika pengguna Anda merespons prompt untuk waktu pengiriman dengan "10:00 PM," Amazon Lex memasukkan hal berikut ke dalam fungsi Lambda:

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

Untuk informasi selengkapnya tentang data yang dikirim dari Amazon Lex ke fungsi Lambda, lihat. Format Peristiwa Masukan

AMAZON. WeightUnit

Mengubah kata-kata yang mewakili satuan berat menjadi singkatan yang sesuai. Misalnya, “kilogram” dikonversi menjadi`kg`.

Jenis slot ini hanya tersedia di bahasa Inggris (AS) (en-AS) lokal.

Contoh berikut menunjukkan bagaimana jenis `AMAZON.WeightUnit` slot menangkap satuan berat:

Satuan berat	Singkatan
kilogram, kilo, kg, KGS	kg
gram, gms, gm, GMS, g	g

Satuan berat	Singkatan
miligram, mg, mg	mg
pound, lbs, lbs	lbs
ons, ons, OZ	ons
ton, ton, t	t
kiloton, kt	kt

Jenis Slot Kustom Slot Jenis Slot Khusus

Untuk setiap intent, Anda dapat menentukan parameter yang menunjukkan informasi yang dibutuhkan maksud untuk memenuhi permintaan pengguna. Parameter ini, atau slot, memiliki tipe. Jenis slot adalah daftar nilai yang digunakan Amazon Lex untuk melatih model pembelajaran mesin untuk mengenali nilai slot. Misalnya, Anda dapat menentukan jenis slot yang disebut "`Genres.`" Setiap nilai dalam jenis slot adalah nama genre, “komedi,” “petualangan,” “dokumenter,” dll. Anda dapat menentukan sinonim untuk nilai jenis slot. Misalnya, Anda dapat mendefinisikan sinonim “lucu” dan “lucu” untuk nilai “komedi.”

Anda dapat mengkonfigurasi jenis slot untuk membatasi resolusi ke nilai slot. Nilai slot akan digunakan sebagai pencacahan dan nilai yang dimasukkan oleh pengguna akan diselesaikan ke nilai slot hanya jika sama dengan salah satu nilai slot atau sinonim. Sebuah sinonim diselesaikan dengan nilai slot yang sesuai. Misalnya, jika pengguna memasukkan “lucu” itu akan menyelesaikan dengan nilai slot “komedi.”

Bergantian, Anda dapat mengkonfigurasi jenis slot untuk memperluas nilai. Nilai slot akan digunakan sebagai data pelatihan dan slot diselesaikan dengan nilai yang diberikan oleh pengguna jika mirip dengan nilai slot dan sinonim. Ini adalah perilaku default.

Amazon Lex mempertahankan daftar resolusi yang mungkin untuk celah. Setiap entri dalam daftar memberikan nilai resolusi yang diakui Amazon Lex sebagai kemungkinan tambahan untuk slot. Nilai resolusi adalah upaya terbaik untuk mencocokkan nilai slot. Daftar ini berisi hingga lima nilai.

Ketika nilai yang dimasukkan oleh pengguna adalah sinonim, entri pertama dalam daftar nilai resolusi adalah nilai jenis slot. Misalnya, jika pengguna memasukkan “lucu,”`slots` bidang berisi “lucu” dan entri pertama di`slotDetails` lapangan adalah “komedi.” Anda dapat

mengkonfigurasi`valueSelectionStrategy` ketika Anda membuat atau memperbarui jenis slot dengan[PutSlotType](#) operasi sehingga nilai slot diisi dengan nilai pertama dalam daftar resolusi.

Jika Anda menggunakan fungsi Lambda, acara input ke fungsi tersebut mencakup daftar resolusi yang disebut`slotDetails`. Contoh berikut menunjukkan slot dan slot rincian bagian input ke fungsi Lambda:

```
"slots": {
   "MovieGenre": "funny";
},
"slotDetails": {
   "Movie": {
      "resolutions": [
         "value": "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. Jika Anda melebihi batas ini, Anda akan mendapatkan`LimitExceededException` saat Anda menelepon[PutBot](#) operasi.

Obfuscation Slot

Amazon Lex memungkinkan Anda untuk mengaburkan, atau menyembunyikan, isi slot sehingga konten tidak terlihat. Untuk melindungi data sensitif yang ditangkap sebagai nilai slot, Anda dapat mengaktifkan slot obfuscation untuk menutupi nilai-nilai tersebut dalam log percakapan.

Bila Anda memilih untuk mengaburkan nilai Slot, Amazon Lex menggantikan nilai slot dengan nama slot di log percakapan. Untuk slot yang disebut`full_name`, nilai slot akan dikaburkan sebagai berikut:

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

Jika ucapan berisi karakter braket ({}) Amazon Lex lolos dari karakter braket dengan dua garis miring belakang (\\). Misalnya, teks`{John Stiles}`dikaburkan sebagai berikut:

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

Nilai Slot dikaburkan dalam log percakapan. Nilai slot masih tersedia dalam respon dari`PostContent`dan`PostText`operasi, dan nilai-nilai slot yang tersedia untuk validasi dan pemenuhan fungsi Lambda Anda. Jika Anda menggunakan nilai slot dalam petunjuk atau tanggapan Anda, nilai-nilai slot tersebut tidak dikaburkan dalam log percakapan.

Pada giliran pertama dari percakapan, Amazon Lex mengaburkan nilai Slot jika mengakui slot dan slot nilai dalam ucapan tersebut. Jika tidak ada nilai slot yang diakui, Amazon Lex tidak mengaburkan ucapan tersebut.

Pada putaran kedua dan kemudian, Amazon Lex tahu slot untuk mendapatkan dan jika nilai slot harus dikaburkan. Jika Amazon Lex mengenali nilai slot, nilainya dikaburkan. Jika Amazon Lex tidak mengenali nilai, seluruh ucapan dikaburkan. Nilai slot apa pun dalam ucapan yang tidak terjawab tidak akan dikaburkan.

Amazon Lex 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 nilai.

Amazon Lex tidak mengaburkan nilai slot dalam audio. Itu mengaburkan nilai slot dalam transkripsi audio.

Anda tidak perlu mengaburkan semua slot pada bot. Anda dapat memilih slot mana yang mengaburkan menggunakan konsol atau dengan menggunakan API Amazon Lex. Di konsol, pilih`Slot`dalam pengaturan untuk slot. Jika Anda menggunakan API, atur`obfuscationSetting`bidang slot untuk`DEFAULT_OBFUSCATION`ketika Anda menelepon`PutIntent`operasi.

Analisis Sentimen

Anda dapat menggunakan analisis sentimen untuk menentukan sentimen yang dinyatakan dalam ucapan pengguna. Dengan informasi sentimen Anda dapat mengelola alur percakapan atau

melakukan analisis pasca-panggilan. Misalnya, jika sentimen pengguna negatif, Anda dapat membuat alur untuk menyerahkan percakapan kepada 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 menunjukkan kemungkinan bahwa sentimen terdeteksi dengan benar.

Anda mengaktifkan analisis sentimen untuk bot menggunakan konsol atau dengan menggunakan API Amazon Lex. Di konsol Amazon Lex, pilih tab Pengaturan untuk bot Anda, lalu atur opsi Analisis Sentimen ke Ya. Jika Anda menggunakan API, panggilPutBot operasi dengan`detectSentiment` bidang yang diatur ke`true`.

Ketika analisis sentimen diaktifkan, respons dariPostContent danPostText operasi mengembalikan bidang yang disebut`sentimentResponse` dalam respons bot dengan metadata lain. `sentimentResponse`Bidang ini memiliki dua bidang`SentimentScore`,`SentimentLabel` dan, yang berisi hasil analisis sentimen. Jika Anda menggunakan fungsi Lambda,`sentimentResponse` kolom tersebut disertakan dalam data peristiwa yang dikirim ke fungsi Anda.

Berikut ini adalah contoh dari`sentimentResponse` bidang dikembalikan sebagai bagian dari`PostText` atau`PostContent` respon. `SentimentScore`Bidang adalah string yang berisi skor untuk respon.

```
{
    "SentimentScore":
        "{
        Mixed: 0.030585512690246105,
        Positive: 0.94992071056365967,
        Neutral: 0.0141543131828308,
        Negative: 0.00893945890665054
        }",
    "SentimentLabel": "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 layanan dan perjanjian 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 dalam Panduan Pengembang Amazon Comprehend.

Menandai Sumber Daya Amazon Lex

Untuk membantu Anda mengelola bot Amazon Lex, alias bot, dan saluran bot, Anda dapat menetapkan metadata ke setiap sumber daya sebagaitag. Tanda merupakan sebuah label yang Anda tetapkan ke sebuah sumber daya AWS. Setiap tanda terdiri atas kunci dan nilai.

Tag memungkinkan Anda untuk mengategorikan sumber daya AWS Anda dengan berbagai cara, misalnya, berdasarkan tujuan, pemilik, atau aplikasi. Tanda membantu Anda untuk:

- Mengidentifikasi dan mengorganisir sumber daya AWS Anda. BanyakAWSsumber daya mendukung penandaan, sehingga Anda dapat menetapkan tag yang sama ke sumber daya dalam layanan yang berbeda untuk menunjukkan bahwa sumber daya tersebut terkait. Misalnya, Anda dapat menandai bot dan fungsi Lambda yang digunakan dengan tag yang sama.
- Alokasikan biaya. Anda mengaktifkan tag padaAWS Billing and Cost Managementdasbor.AWSmenggunakan tag untuk mengategorikan biaya Anda dan mengirimkan laporan alokasi biaya bulanan kepada Anda. Untuk Amazon Lex, Anda dapat mengalokasikan biaya untuk setiap alias menggunakan tag khusus untuk alias, kecuali untuk`$LATEST`alias. Anda mengalokasikan biaya untuk`$LATEST`alias menggunakan tag untuk bot Amazon Lex Anda. Untuk informasi selengkapnya, lihatMenggunakan Tag Alokasi Biayadi dalamAWS Billing and Cost ManagementPanduan Pengguna.
- Kontrol akses ke sumber daya Anda. Anda dapat menggunakan tag ke Amazon Lex untuk membuat kebijakan untuk mengontrol akses ke sumber daya Amazon Lex. Kebijakan ini dapat dilampirkan ke IAM role atau pengguna untuk mengaktifkan kontrol akses berbasis tag. Untuk informasi selengkapnya, lihat ABAC dengan Amazon Lex. Untuk melihat contoh kebijakan berbasis identitas untuk membatasi akses ke sumber daya berdasarkan tag pada sumber daya tersebut, lihat Menggunakan Tag untuk Mengakses Sumber Daya.

Anda dapat bekerja dengan tag dengan menggunakanAWS Management Console, yangAWS Command Line Interface, atau API Amazon Lex.

Menandai Sumber Daya Anda

Jika Anda menggunakan Amazon Lex console, 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 menggunakanAWS CLIatau Amazon Lex API, Anda menggunakan operasi berikut untuk mengelola tag untuk sumber daya Anda:

- ListTagsForResource— lihat tag yang terkait dengan sumber daya.
- PutBotdanPutBotAlias— Terapkan tag saat Anda membuat bot atau alias bot.
- TagResource- Menambahkan dan memodifikasi tag pada sumber daya yang ada.
- UntagResource— menghapus tag dari sumber daya.

Sumber daya berikut di Amazon Lex mendukung:

- Bots - gunakan Amazon Resource Name (ARN) seperti berikut:
  - `arn:${partition}:lex:${region}:${account}:bot:${bot-name}`
- alias bot - menggunakan ARN seperti berikut:
  - `arn:${partition}:lex:${region}:${account}:bot:${bot-name}:${bot-alias}`
- saluran Bot - menggunakan ARN seperti berikut:
  - `arn:${partition}:lex:${region}:${account}:bot-channel:${bot-name}:${bot-alias}:${channel-name}`

Pembatasan Tag

Pembatasan dasar berikut berlaku untuk tag pada sumber daya Amazon Lex:

- Jumlah maksimum tag - 50
- Panjang kunci maksimum – 128 karakter
- Panjang nilai maksimum – 256 karakter
- Karakter yang valid untuk kunci dan nilai — a-z, A-Z, spasi, dan karakter berikut: _.:/= + - dan @
- Kunci dan nilai peka huruf besar dan 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, alias bot, atau sumber daya bot. 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 keAWS Management Consoledan buka konsol Amazon Lex dihttps://console.aws.amazon.com/lex/.
2. MemiilihBuatuntuk membuat bot baru.
3. Di bagian bawahMembuat bot Andahalaman, pilihTag.
4. MemiilihTambahkan tagdan tambahkan satu tag atau lebih ke bot. Anda dapat menambahkan hingga 50 tanda.

Untuk menambahkan tag saat Anda membuat alias bot

1. Masuk keAWS Management Consoledan buka konsol Amazon Lex dihttps://console.aws.amazon.com/lex/.
2. Pilih bot yang ingin Anda tambahkan alias bot.
3. Pilih Pengaturan.
4. Tambahkan nama alias, pilih versi bot, dan kemudian pilihTambahkan tag.
5. MemiilihTambahkan tagdan tambahkan satu tag atau lebih ke alias bot. Anda dapat menambahkan hingga 50 tanda.

Untuk menambahkan tag saat Anda membuat saluran bot

1. Masuk keAWS Management Consoledan buka konsol Amazon Lex dihttps://console.aws.amazon.com/lex/.
2. Pilih bot yang ingin Anda tambahkan saluran bot.
3. MemiilihChannellalu pilih saluran yang ingin Anda tambahkan.
4. Tambahkan detail untuk saluran bot, lalu pilihTag.
5. MemiilihTambahkan tagdan tambahkan satu tag atau lebih ke saluran bot. Anda dapat menambahkan hingga 50 tanda.

Untuk menambahkan tag saat Anda mengimpor bot

1. Masuk keAWS Management Consoledan buka konsol Amazon Lex dihttps://console.aws.amazon.com/lex/.
2. MemiilihTindakandan kemudian pilihImpor.
3. Pilih file zip untuk mengimpor bot.
4. MemiilihTag, lalu pilihTambahkan taguntuk menambahkan satu tag atau lebih ke bot. Anda dapat menambahkan hingga 50 tanda.

Untuk menambahkan, menghapus, atau memodifikasi tag pada bot yang ada

1. Masuk keAWS Management Consoledan buka konsol Amazon Lex dihttps://console.aws.amazon.com/lex/.
2. Dari menu sebelah kiri, pilihBotkemudian pilih bot yang ingin Anda ubah.
3. MemiilihPengaturandan kemudian dari menu kiri pilihUmum.
4. MemiilihTaglalu tambahkan, memodifikasi, atau menghapus tag untuk bot.

Untuk menambahkan, menghapus, atau memodifikasi tag pada alias bot

1. Masuk keAWS Management Consoledan buka konsol Amazon Lex dihttps://console.aws.amazon.com/lex/.
2. Dari menu sebelah kiri, pilihBotkemudian pilih bot yang ingin Anda ubah.
3. MemiilihPengaturandan kemudian dari menu kiri pilihAlias.
4. MemiilihKelola taguntuk alias yang ingin Anda ubah, lalu tambahkan, memodifikasi, atau menghapus tag untuk alias bot.

Untuk menambahkan, menghapus, atau memodifikasi tag pada saluran bot yang ada

1. Masuk keAWS Management Consoledan buka konsol Amazon Lex dihttps://console.aws.amazon.com/lex/.
2. Dari menu sebelah kiri, pilihBotkemudian pilih bot yang ingin Anda ubah.
3. MemiilihChannel.
4. MemiilihTaglalu tambahkan, memodifikasi, atau menghapus tag untuk saluran bot.

Pemberian Tag pada Sumber Daya (AWS CLI)

Anda dapat menggunakanAWS CLIuntuk mengelola tag pada bot, alias bot, atau sumber daya saluran bot. Anda dapat menambahkan tag saat membuat bot atau alias bot, atau Anda dapat menambahkan, memodifikasi, atau menghapus tag dari bot, alias bot, atau saluran bot.

Semua contoh diformat untuk Linux dan macOS. Untuk menggunakan perintah di Windows, ganti karakter kelanjutan Linux (\) dengan tanda (^).

Untuk menambahkan tag saat Anda membuat bot

- Berikut disingkat`put-bot` AWS CLIperintah menunjukkan parameter yang harus Anda gunakan untuk menambahkan tag saat Anda membuat bot. Untuk benar-benar membuat bot, Anda harus menyediakan parameter lain. Untuk informasi selengkapnya, lihat Langkah 4: Memulai (AWS CLI).

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

Untuk menambahkan tag saat Anda membuat alias bot

- Berikut disingkat`put-bot-alias` AWS CLIperintah menunjukkan parameter yang harus Anda gunakan untuk menambahkan tag saat Anda membuat alias bot. Untuk benar-benar membuat alias bot, Anda harus menyediakan parameter lain. Untuk informasi selengkapnya, lihat Latihan 5: Membuat Alias (AWS CLI).

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

Untuk membuat daftar tanda pada sumber daya

- Menggunakan`list-tags-for-resource` AWS CLIperintah untuk menunjukkan sumber daya yang terkait dengan bot, bot alias, saluran bot.

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

Untuk menambah atau memodifikasi tag pada sumber daya

- Menggunakan`tag-resource` AWS CLIperintah untuk menambah atau memodifikasi bot, bot alias, atau saluran bot.

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

Untuk menghapus tag dari sumber daya

- Menggunakan`untag-resource` AWS CLIperintah untuk menghapus tag dari bot, bot alias, atau saluran bot.

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

Memulai dengan Amazon Lex

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

- AWS SDK — Saat menggunakan SDK, permintaan Anda ke Amazon Lex ditandatangani dan diautentikasi secara otomatis menggunakan kredensi yang Anda berikan. Ini adalah pilihan yang direkomendasikan untuk membangun aplikasi Anda.
- AWS CLI— Anda dapat menggunakanAWS CLI untuk mengakses Amazon Lex tanpa harus menulis kode apa pun.
- Konsol AWS — Konsol adalah cara termudah untuk memulai pengujian dan menggunakan Amazon Lex

Jika Anda baru menggunakan Amazon Lex, kami sarankan agar AndaAmazon Lex: Cara Kerjanya membaca. pertama.

Topik

- Langkah 1: Siapkan Akun AWS dan Buat Pengguna Administrator
- Langkah 2: Siapkan AWS Command Line Interface
- Langkah 3: Memulai (Konsol)
- Langkah 4: Memulai (AWS CLI)

Langkah 1: Siapkan Akun AWS dan Buat Pengguna Administrator

Sebelum Anda menggunakan Amazon Lex untuk pertama kali, selesaikan tugas berikut:

1. Mendaftar AWS
2. Membuat pengguna

Mendaftar AWS

Jika sudah memilikiAWS akun, lewati tugas ini.

Saat Anda mendaftar untuk Amazon Web Services (AWS),AWS akun Anda secara otomatis terdaftar untuk semua layanan diAWS, termasuk Amazon Lex. Anda hanya dikenakan biaya untuk layanan yang Anda gunakan.

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

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

Untuk membuat akun AWS

1. Buka https://portal.aws.amazon.com/billing/signup.
2. Ikuti petunjuk online.

   Bagian dari prosedur pendaftaran melibatkan menerima panggilan telepon dan memasukkan kode verifikasi di keypad telepon.

   Ketika Anda mendaftar untuk sebuahAkun AWS, sebuah Pengguna root akun AWSdibuat. Pengguna root memiliki akses ke semuaLayanan AWS dan sumber daya di akun. Sebagai praktik terbaik keamanan, tetapkan akses administratif ke pengguna administratif, dan gunakan hanya pengguna root untuk melakukan tugas yang memerlukan akses pengguna root.

Catat IDAWS akun Anda karena Anda akan membutuhkannya untuk tugas baru.

Membuat pengguna

Layanan diAWS, seperti Amazon Lex, mengharuskan Anda memberikan kredensia ketika Anda mengaksesnya agar layanan dapat menentukan apakah Anda memiliki izin untuk mengakses sumber daya yang dimiliki layanan. Konsol tersebut memerlukan kata sandi Anda. Namun, sebaiknya jangan akses AWS menggunakan kredensial untuk akun AWS Anda. Sebagai gantinya, kami merekomendasikan Anda:

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

Anda kemudian dapat mengaksesAWS menggunakan URL khusus dan kredensia pengguna.

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` diAWS akun Anda. Untuk melihat instruksi, lihat Membuat Grup Pengguna dan Administrator IAM di Panduan Pengguna IAM.
2. Sebagai pengguna, Anda dapat masuk keAWS Management Console menggunakan URL khusus. Untuk informasi selengkapnya, Cara Pengguna Masuk ke Akun Anda dalam Panduan Pengguna IAM.

Untuk informasi selengkapnya tentang IAM, lihat hal berikut:

- AWS Identity and Access Management (IAM)
- Memulai
- Panduan Pengguna IAM

Langkah Selanjutnya

Langkah 2: Siapkan AWS Command Line Interface

Langkah 2: Siapkan AWS Command Line Interface

Jika Anda lebih suka menggunakan Amazon Lex denganAWS Command Line Interface (AWS CLI), unduh dan konfigurasikan.

> **Important**
>
> Anda tidak memerlukanAWS CLI untuk melakukan langkah-langkah dalam latihan Memulai. Namun, beberapa latihan kemudian dalam panduan ini menggunakanAWS CLI. Jika Anda lebih suka memulai dengan menggunakan konsol, lewati langkah ini dan lanjutkan keLangkah 3: Memulai (Konsol). Kemudian, ketika Anda membutuhkanAWS CLI, kembali ke sini untuk mengaturnya.

Untuk mengatur AWS CLI

1. Unduh dan konfigurasi AWS CLI. Untuk melakukannya, lihat topik berikut di Panduan Pengguna AWS Command Line Interface:

   - Mendapatkan Set Up denganAWS Command Line Interface
   - MengkonfigurasiAWS Command Line Interface

2. Tambahkan profil bernama untuk pengguna administrator ke akhir fileAWS CLI config. Anda menggunakan profil ini saat menjalankanAWS CLI perintah. Untuk informasi selengkapnya tentang profil yang diberi nama, lihat Profil yang Diberi Nama dalam Panduan Pengguna AWS Command Line Interface.

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

   Untuk daftarAWS Wilayah yang tersedia, lihat Wilayah dan Titik Akhir di Referensi Umum Amazon Web Services.

3. Verifikasikan penyiapan dengan mengetikkan perintah Bantuan pada command prompt:

   ```
   aws help
   ```

Langkah 3: Memulai (Konsol)

Langkah 3: Memulai (Konsol)

Cara termudah untuk mempelajari cara menggunakan Amazon Lex adalah dengan menggunakan konsol. Untuk membantu Anda memulai, kami membuat latihan berikut, yang semuanya menggunakan konsol:

- Latihan 1 — Buat bot Amazon Lex menggunakan cetak biru, bot yang telah ditentukan sebelumnya yang menyediakan semua konfigurasi bot yang diperlukan. Anda hanya melakukan pekerjaan minimum untuk menguji end-to-end pengaturan.

Selain itu, Anda menggunakan cetak biru fungsi Lambda, yang disediakan olehAWS Lambda, untuk membuat fungsi Lambda. Fungsinya adalah hook kode yang menggunakan kode standar yang kompatibel dengan bot Anda.

- Latihan 2 - Buat bot khusus dengan membuat dan mengonfigurasi bot secara manual. Anda juga membuat fungsi Lambda sebagai pengait kode. Kode sampel disediakan.
- Latihan 3 - Publikasikan bot, lalu buat versi barunya. Sebagai bagian dari latihan ini, Anda membuat alias yang menunjuk ke versi bot.

Topik

Latihan 1: Membuat Bot Amazon Lex Menggunakan Cetak Biru (Konsol)

Dalam latihan ini, Anda melakukan hal berikut:

- Buat bot Amazon Lex pertama Anda, dan uji di konsol Amazon Lex.

  Untuk latihan ini, Anda menggunakan OrderFlowerscetak biru. Untuk informasi tentang cetak biru, lihatAmazon Lex danAWS LambdaCetak Biru.

- BuatAWS Lambda fungsi dan uji di konsol Lambda Saat memproses permintaan, bot Anda memanggil fungsi Lambda ini. Untuk latihan ini, Anda menggunakan cetak biru Lambda (lex-order-flowers-python) yang disediakan diAWS Lambda konsol untuk membuat fungsi Lambda Anda. Kode cetak biru menggambarkan bagaimana Anda dapat menggunakan fungsi Lambda yang sama untuk melakukan inisialisasi dan validasi, dan untuk memenuhi`OrderFlowers` maksud.

- Perbarui bot untuk menambahkan fungsi Lambda sebagai pengait kode untuk memenuhi maksud. Uji end-to-end pengalamannya.

Bagian berikut menjelaskan apa yang dilakukan cetak biru.

Amazon Lex Bot: Ikhtisar Cetak Biru

Anda menggunakan OrderFlowerscetak biru untuk membuat Amazon Lex Bot.Untuk informasi selengkapnya tentang struktur bot, lihatAmazon Lex: Cara Kerjanya. Bot telah dikonfigurasi sebelumnya sebagai berikut:

- Niat - OrderFlowers
- jenis Slot - Salah satu jenis slot kustom disebut`FlowerTypes` dengan nilai pencacahan:`roses`,`lilies`, dan`tulips`.
- Slot - Tujuannya membutuhkan informasi berikut (yaitu, slot) sebelum bot dapat memenuhi maksud.
  - `PickupTime`(AMAZON.TIME tipe bawaan)
  - `FlowerType`(tipeFlowerTypes khusus)
  - `PickupDate`(AMAZON.DATE tipe bawaan)
- Ucapan - Contoh ucapan berikut menunjukkan maksud pengguna:
  - “Saya ingin mengambil bunga.”
  - “Saya ingin memesan beberapa bunga.”
- Petunjuk - Setelah bot mengidentifikasi maksud, bot menggunakan petunjuk 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 oke?”

AWS LambdaFungsi: Ringkasan Cetak Biru

Fungsi Lambda dalam latihan ini melakukan tugas inisialisasi dan validasi dan pemenuhan. Oleh karena itu, setelah membuat fungsi Lambda, Anda memperbarui konfigurasi maksud dengan menentukan fungsi Lambda yang sama sebagai hook kode untuk menangani tugas inisialisasi dan validasi dan pemenuhan.

- Sebagai pengait kode inisialisasi dan validasi, fungsi Lambda melakukan validasi dasar. Misalnya, jika pengguna menyediakan waktu untuk pengambilan yang berada di luar jam kerja normal, fungsi Lambda mengarahkan Amazon Lex untuk meminta ulang pengguna untuk waktu tersebut.
- Sebagai bagian dari hook kode pemenuhan, fungsi Lambda mengembalikan pesan ringkasan yang menunjukkan bahwa urutan bunga telah ditempatkan (yaitu, maksud terpenuhi).

Langkah Selanjutnya

Langkah 1: Buat Bot Amazon Lex (Konsol)

Langkah 1: Buat Bot Amazon Lex (Konsol)

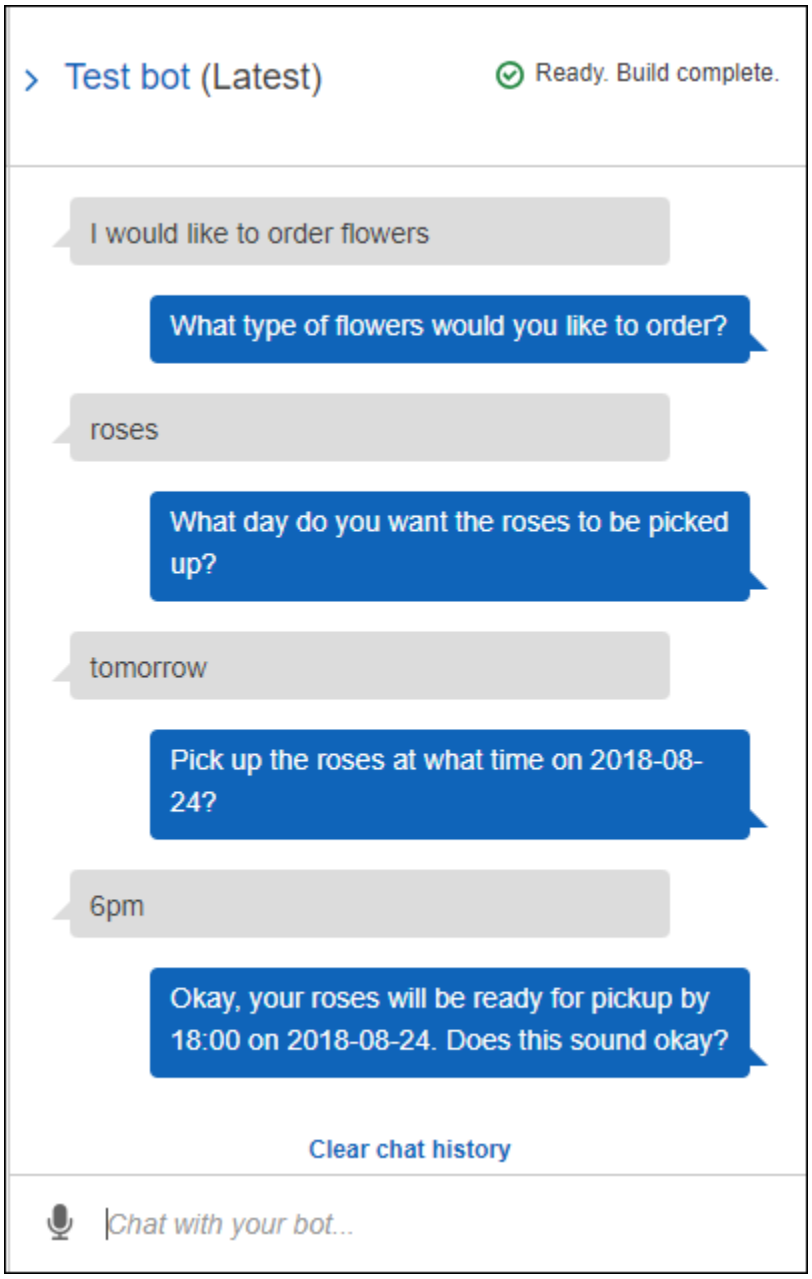
Untuk latihan ini, buat bot untuk memesan bunga, yang disebut OrderFlowersBot.

Untuk membuat bot Amazon Lex (konsol)

1. Masuk keAWS Management Console dan buka konsol Amazon Lex di https://console.aws.amazon.com/lex/.
2. Jika ini adalah bot pertama Anda, pilih Memulai; jika tidak, pada halaman Bot, pilih Buat.
3. Pada halaman Create your Lex bot, berikan informasi berikut, lalu pilih Create.
   - Pilih OrderFlowerscetak biru.
   - Tinggalkan nama bot default (OrderFlowers).
   - Untuk COPPA, pilih**No**.
   - Untuk penyimpanan ucapan Pengguna, pilih respons yang sesuai.
4. Pilih Create (Buat). Konsol membuat permintaan yang diperlukan ke Amazon Lex untuk menyimpan konfigurasi. Konsol kemudian menampilkan jendela editor bot.
5. Tunggu konfirmasi bahwa bot Anda dibangun.
6. Uji bot.

> **Note**
>
> Anda dapat menguji bot dengan mengetikkan teks ke jendela pengujian, atau, untuk browser yang kompatibel, dengan memilih tombol mikrofon di jendela pengujian dan berbicara.

Gunakan contoh teks berikut untuk terlibat dalam percakapan dengan bot untuk memesan bunga:



Dari input ini, bot menyimpulkan`OrderFlowers` maksud dan meminta data slot. Ketika Anda menyediakan semua data slot yang diperlukan, bot memenuhi maksud (`OrderFlowers`) dengan mengembalikan semua informasi ke aplikasi klien (dalam hal ini, konsol). Konsol menampilkan informasi di jendela pengujian.

Secara khusus:

- Dalam pernyataan “Hari apa Anda ingin mawar dijemput? , "istilah “mawar” muncul karena prompt untuk`pickupDate` slot dikonfigurasi menggunakan substitusi,`{FlowerType}`. Verifikasi ini di konsol.
- Pernyataan “Oke, mawar Anda akan siap...” adalah konfirmasi konfirmasi yang Anda konfigurasikan.
- Pernyataan terakhir (”`FlowerType:roses...` “) hanyalah data slot yang dikembalikan ke klien, dalam hal ini, di jendela pengujian. Pada latihan berikutnya, Anda menggunakan fungsi Lambda untuk memenuhi maksud, dalam hal ini Anda mendapatkan pesan yang menunjukkan bahwa pesanan terpenuhi.

Langkah Selanjutnya

Langkah 2 (Opsional): Tinjau Rincian Aliran Informasi (Konsol)

Langkah 2 (Opsional): Tinjau Rincian Aliran Informasi (Konsol)

Bagian ini menjelaskan alur informasi antara klien dan Amazon Lex untuk setiap input pengguna dalam percakapan contoh kami.

Contoh menggunakan jendela uji konsol untuk percakapan dengan bot.

Untuk membuka jendela pengujian Amazon Lex

1. Masuk keAWS Management Console dan buka konsol Amazon Lex di https://console.aws.amazon.com/lex/.
2. Pilih bot untuk diuji.
3. Dari sisi kanan konsol, pilih Uji chatbot.

Untuk melihat alur informasi untuk konten lisan atau diketik, pilih topik yang sesuai.

Topik

- Langkah 2a (Opsional): Tinjau Rincian Aliran Informasi Lisan (Konsol)
- Langkah 2b (Opsional): Tinjau Rincian Aliran Informasi yang Diketik (Konsol)

Langkah 2a (Opsional): Tinjau Rincian Aliran Informasi Lisan (Konsol)

Bagian ini menjelaskan alur informasi antara klien dan Amazon Lex saat klien menggunakan pidato untuk mengirim permintaan. Untuk informasi selengkapnya, lihat PostContent.

1. Pengguna mengatakan: Saya ingin memesan beberapa bunga.

   a. Klien (konsol) mengirimkanPostContent permintaan berikut ke Amazon Lex:

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


Request body
input stream
```

   URI permintaan dan badan memberikan informasi kepada Amazon Lex:

   - Permintaan URI - Menyediakan nama bot (`OrderFlowers`), bot alias (`$LATEST`), dan nama pengguna (string acak yang mengidentifikasi pengguna). `content`menunjukkan bahwa ini adalah permintaan`PostContent` API (bukan`PostText` permintaan).
   - Minta header
     - `x-amz-lex-session-attributes`- Nilai base64 yang dikodekan mewakili “{}”. Ketika klien membuat permintaan pertama, tidak ada atribut sesi.
     - `Content-Type`- Mencerminkan format audio.
   - Permintaan tubuh - Aliran audio input pengguna (“Saya ingin memesan beberapa bunga.”).

   > **Note**
   >
   > Jika pengguna memilih untuk mengirim teks (“Saya ingin memesan beberapa bunga”) ke`PostContent` API alih-alih berbicara, badan permintaan adalah input pengguna. `Content-Type`Header diatur sesuai:

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

Request body
input stream
```

b. Dari input stream, Amazon Lex mendeteksi intent (`OrderFlowers`). Kemudian memilih salah satu slot intent (dalam hal ini,`FlowerType`) dan salah satu petunjuk elikitasi nilainya, dan kemudian mengirimkan respons dengan header berikut:

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

Nilai header memberikan informasi berikut:

- `x-amz-lex-input-transcript`- Menyediakan transkrip audio (input pengguna) dari permintaan
- `x-amz-lex-message`- Menyediakan transkrip audio Amazon Lex yang dikembalikan dalam respons
- `x-amz-lex-slots`- Versi slot dan nilai yang dikodekan base64:

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

- `x-amz-lex-session-attributes`- Versi berkode base64 dari atribut sesi ({})

Klien memutar audio di bodi respons.

2. Pengguna mengatakan: mawar

   a. Klien (konsol) mengirimkanPostContent permintaan berikut ke Amazon Lex:

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


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

Badan permintaan adalah input pengguna aliran audio (mawar). `sessionAttributes`Sisa-sisa kosong.

b. Amazon Lex menafsirkan aliran input dalam konteks maksud saat ini (ia ingat bahwa ia telah meminta pengguna ini untuk informasi yang berkaitan dengan`FlowerType` slot). Amazon Lex pertama-tama memperbarui nilai slot untuk maksud saat ini. Kemudian memilih slot lain (`PickupDate`), bersama dengan salah satu pesan prompt (Kapan Anda ingin mengambil mawar?) , dan mengembalikan respon dengan header berikut:

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

Nilai header memberikan informasi berikut:

- `x-amz-lex-slots`- Versi berkode base64 dari slot dan nilai:

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

- `x-amz-lex-session-attributes`- Versi berkode base64 dari atribut sesi ({})

Klien memutar audio di bodi respons.

3. Pengguna mengatakan: besok

a. Klien (konsol) mengirimkanPostContent permintaan berikut ke Amazon Lex:

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


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

Permintaan tubuh adalah input pengguna aliran audio (“besok”).`sessionAttributes` Sisa-sisa kosong.

b. Amazon Lex menafsirkan aliran input dalam konteks maksud saat ini (ia ingat bahwa ia telah meminta pengguna ini untuk informasi yang berkaitan dengan`PickupDate` slot). Amazon Lex memperbarui nilai slot (`PickupDate`) untuk maksud saat ini. Kemudian memilih slot lain untuk mendapatkan nilai untuk (`PickupTime`) dan salah satu petunjuk nilai elicitation (Kapan Anda ingin mengambil mawar pada 2017-03-18?) , dan mengembalikan respon dengan header berikut:

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

Nilai header memberikan informasi berikut:

- `x-amz-lex-slots`- Versi berkode base64 dari slot dan nilai:

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

- `x-amz-lex-session-attributes`- Versi berkode base64 dari atribut sesi ({})

Klien memutar audio di bodi respons.

4. Pengguna mengatakan: 6 sore

   a. Klien (konsol) mengirimkanPostContent permintaan berikut ke Amazon Lex:

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


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

   Badan permintaan adalah aliran audio input pengguna (“6 sore”). `sessionAttributes`Sisa-sisa kosong.

   b. Amazon Lex menafsirkan aliran input dalam konteks maksud saat ini (ia ingat bahwa ia telah meminta pengguna ini untuk informasi yang berkaitan dengan`PickupTime` slot). Ini pertama kali memperbarui nilai slot untuk maksud saat ini.

   Sekarang Amazon Lex mendeteksi bahwa ia memiliki informasi untuk semua slot. Namun,`OrderFlowers` maksud dikonfigurasi dengan pesan konfirmasi. Oleh karena itu, Amazon Lex memerlukan konfirmasi eksplisit dari pengguna sebelum dapat melanjutkan untuk memenuhi maksud. Ini mengirimkan respons dengan header berikut meminta konfirmasi sebelum memesan bunga:

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

   Nilai header memberikan informasi berikut:

- `x-amz-lex-slots`- Versi berkode base64 dari slot dan nilai:

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

- `x-amz-lex-session-attributes`- Versi berkode base64 dari atribut sesi ({})

Klien memutar audio di bodi respons.

5. Pengguna mengatakan: Ya

   a. Klien (konsol) mengirimkanPostContent permintaan berikut ke Amazon Lex:

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


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

   Badan permintaan adalah aliran audio input pengguna (“Ya”). `sessionAttributes`Sisa-sisa kosong.

   b. Amazon Lex menafsirkan aliran input dan memahami bahwa pengguna ingin melanjutkan pesanan. `OrderFlowers`Maksud dikonfigurasi`ReturnIntent` sebagai aktivitas pemenuhan. Ini mengarahkan Amazon Lex untuk mengembalikan semua data maksud ke klien. Amazon Lex mengembalikan respons dengan mengikuti:

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

   Header`x-amz-lex-dialog-state` respon diatur ke`ReadyForFulfillment`. Klien kemudian dapat memenuhi maksud.

6. Sekarang, uji ulang botnya. Untuk membuat konteks (pengguna) baru, pilih tautan Hapus di konsol. Berikan data untuk`OrderFlowers` maksud, dan sertakan beberapa data yang tidak valid. Misalnya:

   - Melati sebagai jenis bunga (bukan salah satu jenis bunga yang didukung)
   - Kemarin sebagai hari ketika Anda ingin mengambil bunga

   Perhatikan bahwa bot menerima nilai-nilai ini karena Anda tidak memiliki kode untuk menginisialisasi dan memvalidasi data pengguna. Di bagian selanjutnya, Anda menambahkan fungsi Lambda untuk melakukan ini. Perhatikan hal berikut tentang fungsi Lambda:

   - Ini memvalidasi data slot setelah setiap input pengguna. Ini memenuhi maksud di akhir. Artinya, bot memproses urutan bunga dan mengembalikan pesan kepada pengguna alih-alih hanya mengembalikan data slot ke klien. Untuk informasi selengkapnya, lihat Menggunakan Fungsi Lambda.
   - Hal ini juga menetapkan atribut sesi. Untuk informasi selengkapnya tentang atribut sesi, lihatPostText.

     Setelah Anda menyelesaikan bagian Memulai, Anda dapat melakukan latihan tambahan (Contoh Tambahan: Membuat Amazon Lex Bots). Buku Perjalananmenggunakan atribut sesi untuk berbagi informasi lintas-maksud untuk terlibat dalam percakapan dinamis dengan pengguna.

Langkah Selanjutnya

Langkah 3: Buat Fungsi Lambda (Konsol)

Langkah 2b (Opsional): Tinjau Rincian Aliran Informasi yang Diketik (Konsol)

Bagian ini menjelaskan alur informasi antara klien dan Amazon Lex di mana klien menggunakan`PostText` API untuk mengirim permintaan. Untuk informasi selengkapnya, lihat PostText.

1. Jenis pengguna: Saya ingin memesan beberapa bunga

   a. Klien (konsol) mengirimkanPostText permintaan berikut ke Amazon Lex:

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

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

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

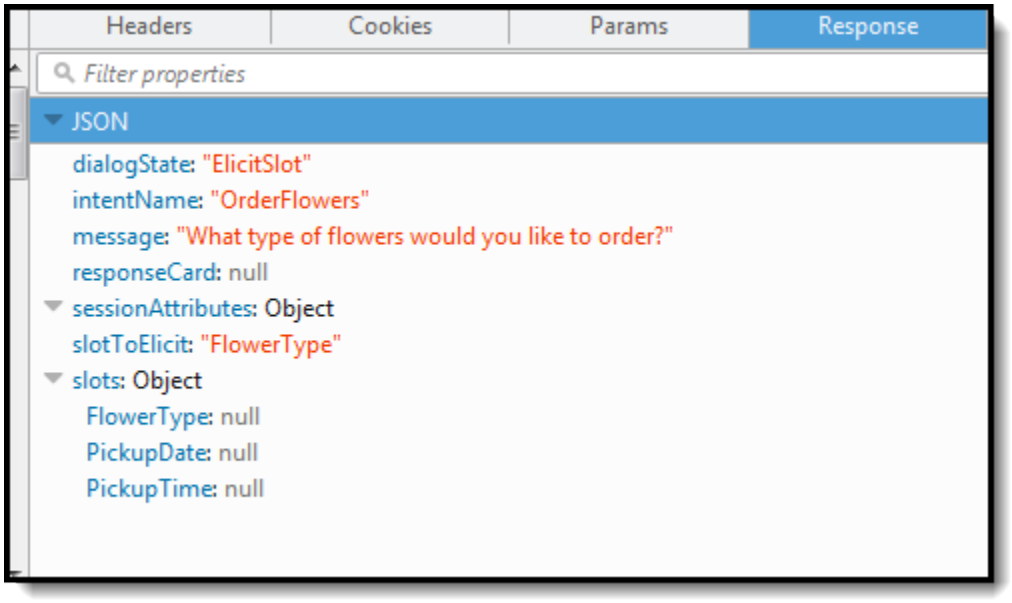
URI permintaan dan badan memberikan informasi kepada Amazon Lex:

- Permintaan URI - Menyediakan nama bot (`OrderFlowers`), bot alias (`$LATEST`), dan nama pengguna (string acak yang mengidentifikasi pengguna). Trailing`text` menunjukkan bahwa itu adalah permintaan`PostText` API (dan bukan`PostContent`).

- Permintaan tubuh - Termasuk input pengguna (`inputText`) dan kosong`sessionAttributes`. Ketika klien membuat permintaan pertama, tidak ada atribut sesi. Fungsi Lambda memulai mereka nanti.

b. Dari`inputText`, Amazon Lex mendeteksi maksud (`OrderFlowers`). Maksud ini tidak memiliki kait kode (yaitu, fungsi Lambda) untuk inisialisasi dan validasi input atau pemenuhan pengguna.

   Amazon Lex memilih salah satu slot (`FlowerType`) maksud untuk mendapatkan nilai. Ini juga memilih salah satu petunjuk nilai elicitation untuk slot (semua bagian dari konfigurasi maksud), dan kemudian mengirimkan respons berikut kembali ke klien. Konsol menampilkan pesan dalam respons terhadap pengguna.



Klien menampilkan pesan dalam respons.

2. Jenis pengguna: mawar

   a. Klien (konsol) mengirimkanPostText permintaan berikut ke Amazon Lex:

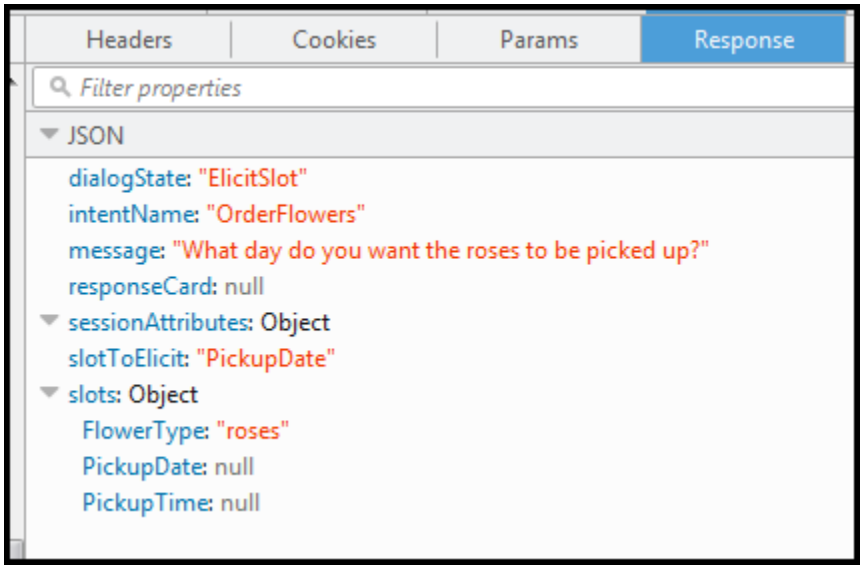
```
POST /bot/OrderFlowers/alias/$LATEST/user/4o9wwdhx6nlheferh6a73fujd3118f5w/text
"Content-Type":"application/json"
"Content-Encoding":"amz-1.0"

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

   `inputText`Dalam badan permintaan menyediakan masukan pengguna. `sessionAttributes`Sisa-sisa kosong.

   b. Amazon Lex pertama menafsirkan`inputText` dalam konteks tujuan saat ini — layanan ingat bahwa ia telah meminta pengguna tertentu untuk informasi tentang`FlowerType` slot. Amazon Lex pertama-tama memperbarui nilai slot untuk maksud saat ini dan memilih slot lain (`PickupDate`) bersama dengan salah satu pesan cepat-hari apa Anda ingin mawar dijemput? — untuk slot.

   Kemudian, Amazon Lex mengembalikan respons berikut:



   Klien menampilkan pesan dalam respons.

3. Jenis pengguna: besok

   a. Klien (konsol) mengirimkanPostText permintaan berikut ke Amazon Lex:

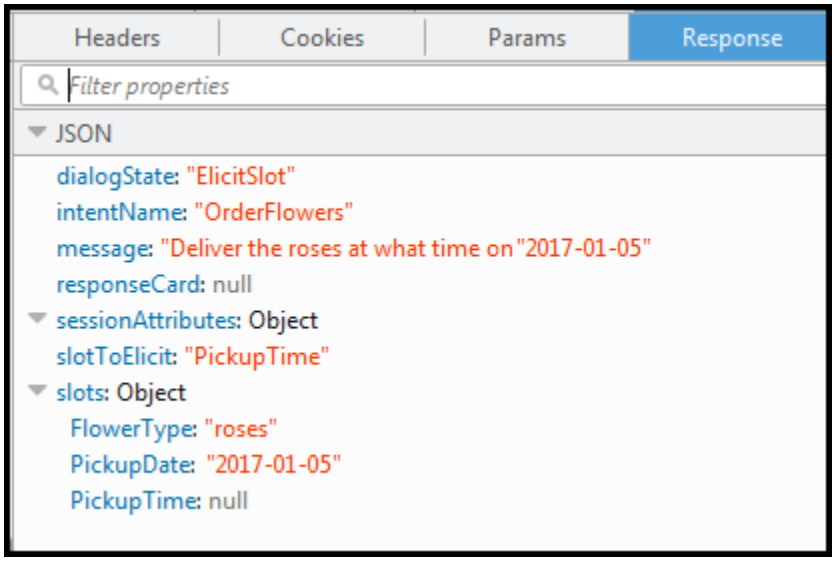
```
POST /bot/OrderFlowers/alias/$LATEST/user/4o9wwdhx6nlheferh6a73fujd3118f5w/text
"Content-Type":"application/json"
"Content-Encoding":"amz-1.0"

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

   `inputText`Dalam badan permintaan menyediakan masukan pengguna. `sessionAttributes`Sisa-sisa kosong.

   b. Amazon Lex pertama menafsirkan`inputText` dalam konteks tujuan saat ini — layanan ingat bahwa ia telah meminta pengguna tertentu untuk informasi tentang`PickupDate` slot. Amazon Lex memperbarui nilai slot (`PickupDate`) untuk maksud saat ini. Ia memilih slot lain untuk mendapatkan nilai untuk (`PickupTime`). Ia mengembalikan salah satu petunjuk nilai-elicitation — memberikan mawar pada jam berapa 2017-01-05? — ke klien.

   Amazon Lex kemudian mengembalikan respons berikut:



   Klien menampilkan pesan dalam respons.

4. Jenis pengguna: 6 sore

   a. Klien (konsol) mengirimkanPostText permintaan berikut ke Amazon Lex:

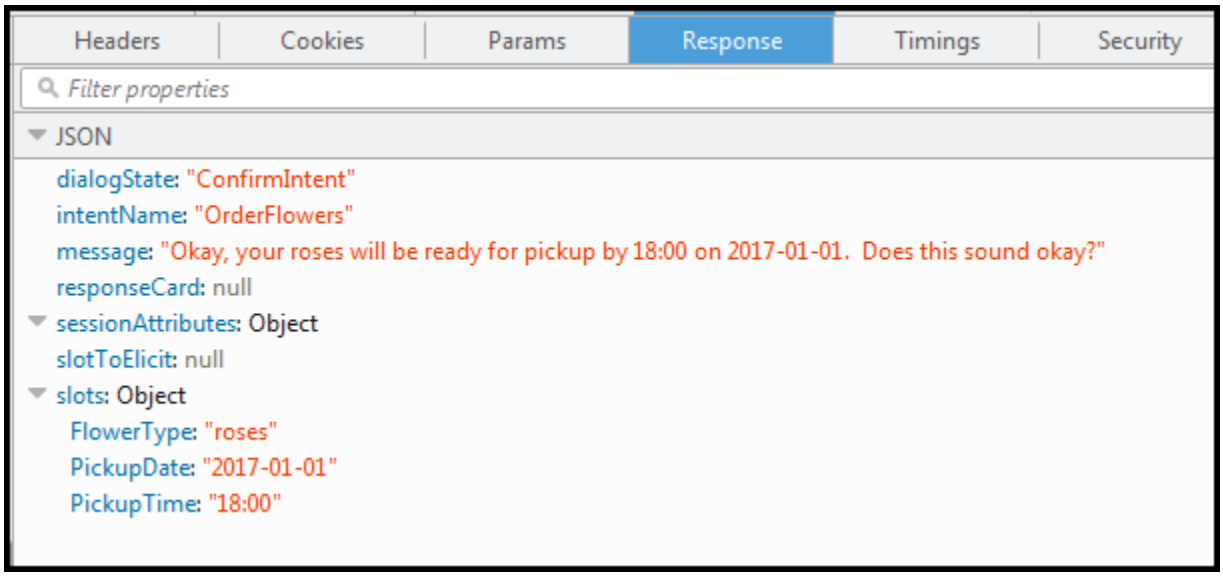
```
POST /bot/OrderFlowers/alias/$LATEST/user/4o9wwdhx6nlheferh6a73fujd3118f5w/text
"Content-Type":"application/json"
"Content-Encoding":"amz-1.0"

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

`inputText`Dalam badan permintaan menyediakan masukan pengguna. `sessionAttributes`Sisa-sisa kosong.

b. Amazon Lex pertama menafsirkan`inputText` dalam konteks tujuan saat ini — layanan ingat bahwa ia telah meminta pengguna tertentu untuk informasi tentang`PickupTime` slot. Amazon Lex pertama-tama memperbarui nilai slot untuk maksud saat ini. Sekarang Amazon Lex mendeteksi bahwa ia memiliki informasi untuk semua slot.

`OrderFlowers`Maksud dikonfigurasi dengan pesan konfirmasi. Oleh karena itu, Amazon Lex memerlukan konfirmasi eksplisit dari pengguna sebelum dapat melanjutkan untuk memenuhi maksud. Amazon Lex mengirimkan pesan berikut ke klien yang meminta konfirmasi sebelum memesan bunga:



Klien menampilkan pesan dalam respons.

5. Jenis pengguna: Ya

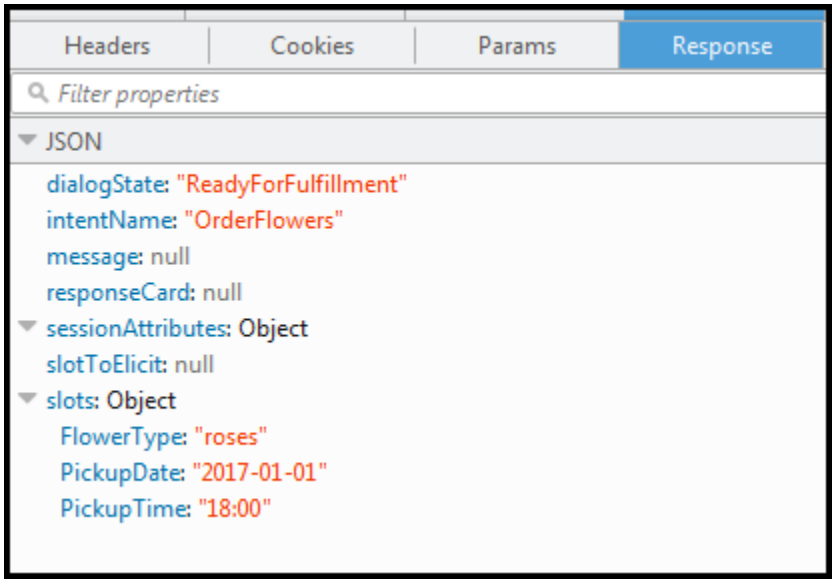
a. Klien (konsol) mengirimkan[PostText](#) permintaan berikut ke Amazon Lex:

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

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

`inputText`Dalam badan permintaan menyediakan masukan pengguna. `sessionAttributes`Sisa-sisa kosong.

b. Amazon Lex menafsirkan`inputText` dalam konteks mengonfirmasi maksud saat ini. Ini memahami bahwa pengguna ingin melanjutkan pesanan. `OrderFlowers`Maksud dikonfigurasi`ReturnIntent` sebagai aktivitas pemenuhan (tidak ada fungsi Lambda untuk memenuhi maksud). Oleh karena itu, Amazon Lex mengembalikan data slot berikut ke klien.



Amazon Lex mengaturdialogState` ke`ReadyForFulfillment`. Klien kemudian dapat memenuhi maksud.

6. Sekarang uji bot lagi. Untuk melakukan itu, Anda harus memilih tautan Hapus di konsol untuk membuat konteks (pengguna) baru. Sekarang saat Anda memberikan data untuk maksud bunga pesanan, cobalah untuk memberikan data yang tidak valid. Misalnya:

   - Melati sebagai jenis bunga (bukan salah satu jenis bunga yang didukung).
   - Kemarin sebagai hari ketika Anda ingin mengambil bunga.

Perhatikan bahwa bot menerima nilai-nilai ini karena Anda tidak memiliki kode untuk menginisialisasi/memvalidasi data pengguna. Di bagian selanjutnya, Anda menambahkan fungsi Lambda untuk melakukan ini. Perhatikan hal berikut tentang fungsi Lambda:

- Fungsi Lambda memvalidasi data slot setelah setiap input pengguna. Ini memenuhi maksud di akhir. Artinya, bot memproses pesanan bunga dan mengembalikan pesan kepada pengguna alih-alih hanya mengembalikan data slot ke klien. Untuk informasi selengkapnya, lihat Menggunakan Fungsi Lambda.
- Fungsi Lambda juga menetapkan atribut sesi. Untuk informasi selengkapnya tentang atribut sesi, lihatPostText.

  Setelah Anda menyelesaikan bagian Memulai, Anda dapat melakukan latihan tambahan (Contoh Tambahan: Membuat Amazon Lex Bots). Buku Perjalananmenggunakan atribut sesi untuk berbagi informasi lintas-maksud untuk terlibat dalam percakapan dinamis dengan pengguna.

Langkah Selanjutnya

Langkah 3: Buat Fungsi Lambda (Konsol)

Langkah 3: Buat Fungsi Lambda (Konsol)

Buat fungsi Lambda (menggunakan lex-order-flowers-pythoncetak biru) dan lakukan pemanggilan pengujian menggunakan contoh data peristiwa diAWS Lambda konsol.

Anda kembali ke konsol Amazon Lex dan menambahkan fungsi Lambda sebagai pengait kode untuk memenuhi`OrderFlowers` maksud`OrderFlowersBot` yang Anda buat di bagian sebelumnya.

Untuk membuat fungsi Lambda (konsol)

1. Masuk ke AWS Management Console dan buka konsol AWS Lambda di https://console.aws.amazon.com/lambda/.
2. Pilih Buat fungsi.
3. Pada halaman Buat fungsi, pilih Gunakan cetak biru. Ketik**`lex-`** kotak teks filter dan kemudian tekan`Enter` untuk menemukan cetak biru, pilih`lex-order-flowers-python` cetak biru.

   Cetak biru fungsi Lambda disediakan di Node.js dan Python. Untuk latihan ini, gunakan cetak biru berbasis Python.

4. Di halaman Informasi Basic, lakukan hal berikut.
   - Ketik nama fungsi Lambda (`OrderFlowersCodeHook`).
   - Untuk peran eksekusi, pilih Membuat peran baru dengan izin Lambda dasar.
   - Biarkan nilai default lainnya.
5. Pilih Buat fungsi.
6. Jika Anda menggunakan lokal selain bahasa Inggris (AS) (id), perbarui nama maksud seperti yang dijelaskan diMemperbarui Cetak Biru untuk Lokal Tertentu.
7. Uji fungsi Lambda.
   a. Pilih Pilih acara pengujian, Konfigurasikan peristiwa pengujian.
   b. Pilih Amazon Lex Order Flowers dari daftar template Event. Contoh kejadian ini cocok dengan model permintaan/respons Amazon Lex (lihatMenggunakan Fungsi Lambda). Berikan acara uji nama (`LexOrderFlowersTest`).
   c. Pilih Create (Buat).
   d. Pilih Test untuk menguji hook kode.
   e. Verifikasikan bahwa fungsi Lambda berhasil berjalan. Tanggapan dalam hal ini cocok dengan model respons Amazon Lex.

Langkah Selanjutnya

Langkah 4: Tambahkan Fungsi Lambda sebagai Kode Hook (Konsol)

Langkah 4: Tambahkan Fungsi Lambda sebagai Kode Hook (Konsol)

Di bagian ini, Anda memperbarui konfigurasi OrderFlowers maksud untuk menggunakan fungsi Lambda sebagai berikut:

- Pertama gunakan fungsi Lambda sebagai pengait kode untuk melakukan pemenuhan`OrderFlowers` maksud. Anda menguji bot dan memverifikasi bahwa Anda menerima pesan pemenuhan dari fungsi Lambda. Amazon Lex memanggil fungsi Lambda hanya setelah Anda memberikan data untuk semua slot yang diperlukan untuk memesan bunga.
- Konfigurasikan fungsi Lambda yang sama dengan hook kode untuk melakukan inisialisasi dan validasi. Anda menguji dan memverifikasi bahwa fungsi Lambda melakukan validasi (saat Anda menyediakan data slot).

Untuk menambahkan fungsi Lambda sebagai pengait kode (konsol)

1. Di konsol Amazon Lex, pilih OrderFlowersbot. Konsol menunjukkan OrderFlowersintent. Pastikan bahwa versi intent diatur ke`$LATEST` karena ini adalah satu-satunya versi yang dapat kita modifikasi.
2. Tambahkan fungsi Lambda sebagai pengait kode pemenuhan dan uji.

   a. Di Editor, pilih AWS Lambdafungsi sebagai Fulfillment, dan pilih fungsi Lambda yang Anda buat di langkah sebelumnya (`OrderFlowersCodeHook`). Pilih OK untuk memberikan izin Amazon Lex untuk memanggil fungsi Lambda.

      Anda mengonfigurasi fungsi Lambda ini sebagai pengait kode untuk memenuhi maksud. Amazon Lex memanggil fungsi ini hanya setelah memiliki semua data slot yang diperlukan dari pengguna untuk memenuhi maksud.

   b. Tentukan pesan Selamat tinggal.

   c. Pilih Build.

   d. Uji bot menggunakan percakapan sebelumnya.

   Pernyataan terakhir “Terima kasih, pesanan Anda untuk mawar...” adalah respons dari fungsi Lambda yang Anda konfigurasikan sebagai pengait kode. Pada bagian sebelumnya, tidak ada fungsi Lambda. Sekarang Anda menggunakan fungsi Lambda untuk benar-benar memenuhi`OrderFlowers` maksud.
3. Tambahkan fungsi Lambda sebagai pengait kode inisialisasi dan validasi, dan uji.

   Contoh kode fungsi Lambda yang Anda gunakan dapat melakukan validasi dan pemenuhan input pengguna. Peristiwa masukan yang diterima fungsi Lambda memiliki field (`invocationSource`) yang digunakan kode untuk menentukan bagian kode mana yang akan dijalankan. Untuk informasi selengkapnya, lihat Lambda Fungsi Input Event dan Response Format.

   a. Pilih versi $LATEST dari`OrderFlowers` maksud. Itu adalah satu-satunya versi yang dapat Anda perbarui.

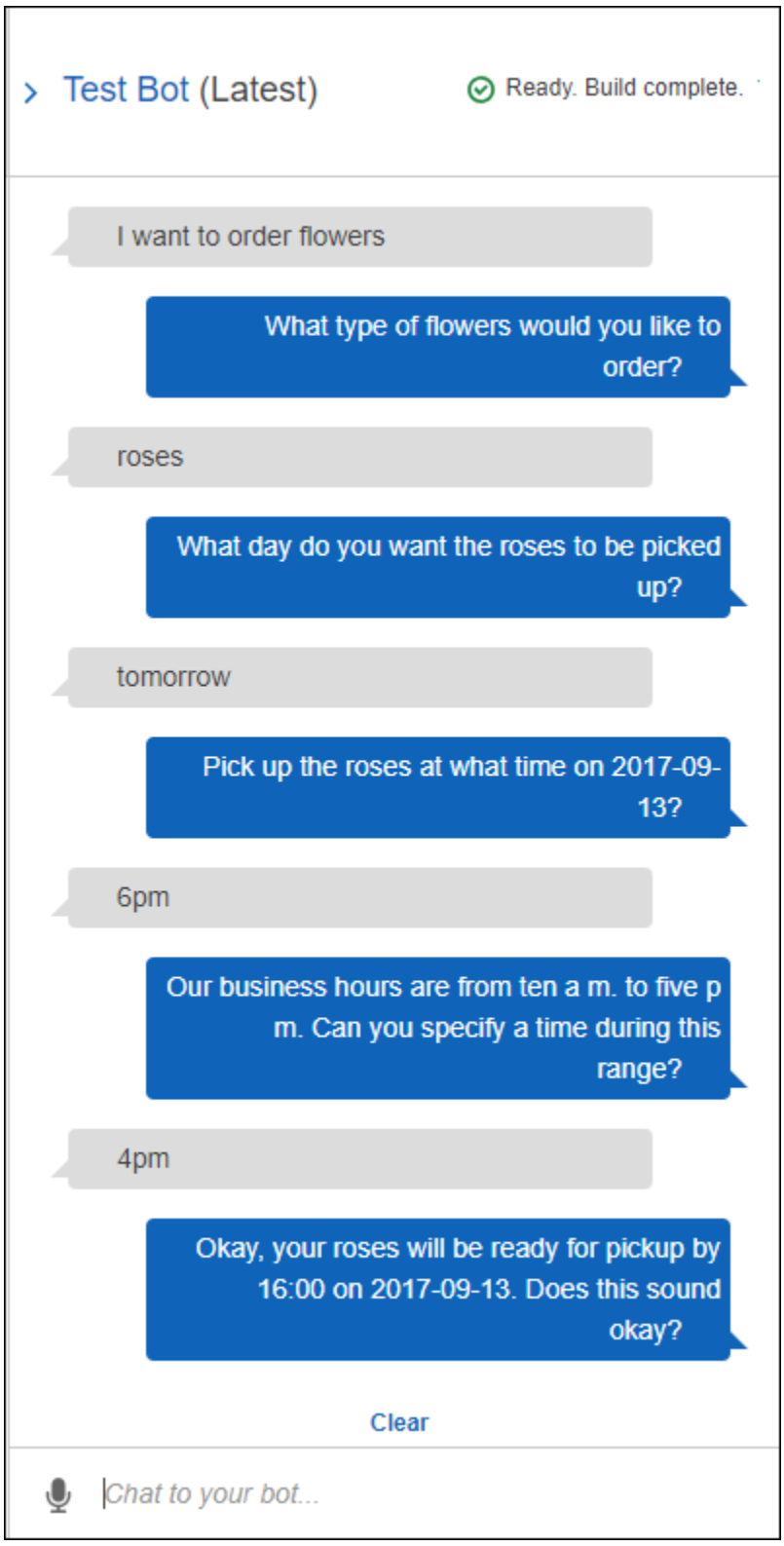
   b. Di Editor, pilih Inisialisasi dan validasi di Opsi.

   c. Sekali lagi, pilih fungsi Lambda yang sama.

   d. Pilih Build.

e. Uji bot.

Anda sekarang siap untuk berkomunikasi dengan Amazon Lex seperti pada gambar berikut. Untuk menguji bagian validasi, pilih waktu 6 sore, dan fungsi Lambda Anda mengembalikan respons (“Jam kerja kami adalah dari jam 10 pagi hingga 5 sore.”), dan meminta Anda lagi. Setelah Anda memberikan semua data slot yang valid, fungsi Lambda memenuhi pesanan.



Langkah Selanjutnya

Langkah 5 (Opsional): Tinjau Rincian Aliran Informasi (Konsol)

Langkah 5 (Opsional): Tinjau Rincian Aliran Informasi (Konsol)

Bagian ini menjelaskan alur informasi antara klien dan Amazon Lex untuk setiap input pengguna, termasuk integrasi fungsi Lambda.

> **Note**
>
> Bagian ini mengasumsikan bahwa klien mengirimkan permintaan ke Amazon Lex menggunakan API`PostText` waktu proses dan menampilkan detail permintaan dan respons yang sesuai. Untuk contoh alur informasi antara klien dan Amazon Lex di mana klien menggunakan`PostContent` API, lihatLangkah 2a (Opsional): Tinjau Rincian Aliran Informasi Lisan (Konsol) .

Untuk informasi selengkapnya tentang`PostText` API waktu proses dan detail tambahan tentang permintaan dan tanggapan yang ditunjukkan pada langkah-langkah berikut, lihatPostText.

1. Pengguna: Saya ingin memesan beberapa bunga.

   a. Klien (konsol) mengirimkanPostText permintaan berikut ke Amazon Lex:

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

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

   URI permintaan dan badan memberikan informasi kepada Amazon Lex:

   - Permintaan URI - Menyediakan nama bot (`OrderFlowers`), bot alias (`$LATEST`), dan nama pengguna (string acak yang mengidentifikasi pengguna). Trailing`text` menunjukkan bahwa itu adalah permintaan`PostText` API (dan bukan`PostContent`).
   - Permintaan tubuh - Termasuk input pengguna (`inputText`) dan kosong`sessionAttributes`. Ketika klien membuat permintaan pertama, tidak ada atribut sesi. Fungsi Lambda memulai mereka nanti.

b. Dari`inputText`, Amazon Lex mendeteksi maksud (`OrderFlowers`). Maksud ini dikonfigurasi dengan fungsi Lambda sebagai pengait kode untuk inisialisasi dan validasi data pengguna. Oleh karena itu, Amazon Lex memanggil fungsi Lambda tersebut dengan meneruskan informasi berikut sebagai data peristiwa:

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

Untuk informasi selengkapnya, lihat Format Peristiwa Masukan.

Selain informasi yang dikirim klien, Amazon Lex juga menyertakan data tambahan berikut:

- `messageVersion`- Saat ini Amazon Lex hanya mendukung versi 1.0.
- `invocationSource`- Menunjukkan tujuan pemanggilan fungsi Lambda. Dalam hal ini, itu adalah untuk melakukan inisialisasi data pengguna dan validasi. Pada saat ini, Amazon Lex tahu bahwa pengguna belum menyediakan semua data slot untuk memenuhi maksud.
- `currentIntent`informasi dengan semua nilai slot diatur ke null.

c. Pada saat ini, semua nilai slot yang nol. Tidak ada fungsi Lambda untuk memvalidasi. Fungsi Lambda mengembalikan respons berikut ke Amazon Lex:

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

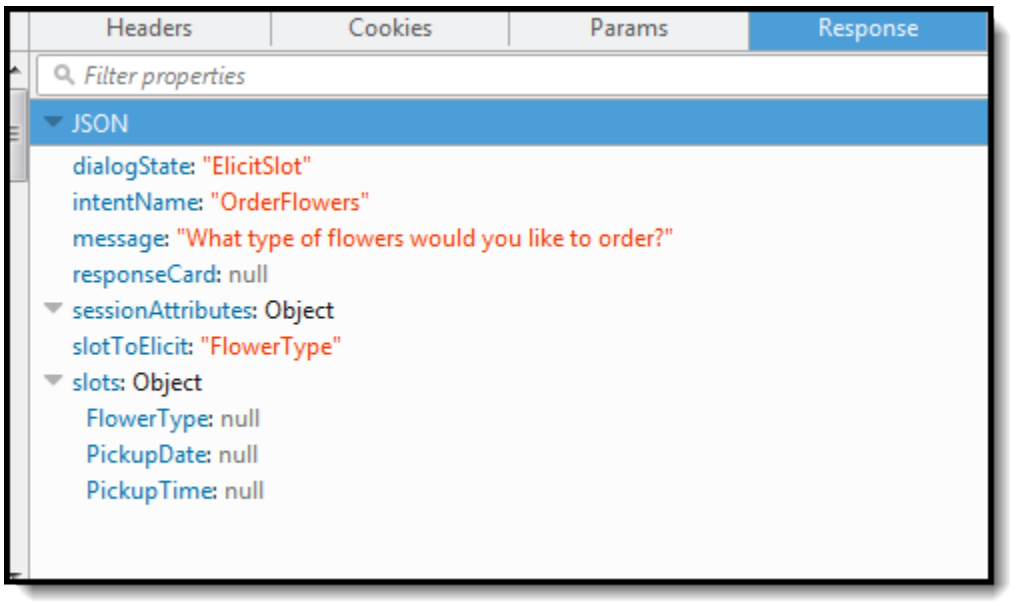
Untuk informasi tentang format respons, lihatFormat Respons.

Perhatikan hal berikut:

- `dialogAction.type`- Dengan menetapkan nilai ini`Delegate`, fungsi Lambda mendelegasikan tanggung jawab untuk memutuskan tindakan berikutnya ke Amazon Lex.

> **Note**
>
> Jika fungsi Lambda mendeteksi sesuatu dalam validasi data pengguna, itu menginstruksikan Amazon Lex apa yang harus dilakukan selanjutnya, seperti yang ditunjukkan dalam beberapa langkah berikutnya.

d. Menurut`dialogAction.type`, Amazon Lex memutuskan tindakan berikutnya. Karena tidak ada slot yang diisi, ia memutuskan untuk mendapatkan nilai untuk`FlowerType` slot. Ini memilih salah satu petunjuk nilai elikitasi (“Jenis bunga apa yang ingin Anda pesan?”) untuk slot ini dan mengirimkan respons berikut kembali ke klien:



Klien menampilkan pesan dalam respons.

2. Pengguna: mawar

   a. Klien mengirimkan[PostText](#) permintaan berikut ke Amazon Lex:

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

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

   Dalam badan permintaan,`inputText` menyediakan masukan pengguna. `sessionAttributes`Sisa-sisa kosong.

   b. Amazon Lex pertama menafsirkan`inputText` dalam konteks maksud saat ini. Layanan ini ingat bahwa ia telah meminta pengguna tertentu untuk informasi tentang`FlowerType` slot. Ini memperbarui nilai slot dalam maksud saat ini dan memanggil fungsi Lambda dengan data peristiwa berikut:

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

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

Perhatikan hal berikut:

- `invocationSource`- terus menjadi`DialogCodeHook` (kami hanya memvalidasi data pengguna).
- `currentIntent.slots`- Amazon Lex telah memperbarui`FlowerType` slot untuk mawar.

c. Menurut`invocationSource` nilai`DialogCodeHook`, fungsi Lambda melakukan validasi data pengguna. Ini mengakui`roses` sebagai nilai slot yang valid (dan ditetapkan`Price` sebagai atribut sesi) dan mengembalikan respons berikut ke Amazon Lex.

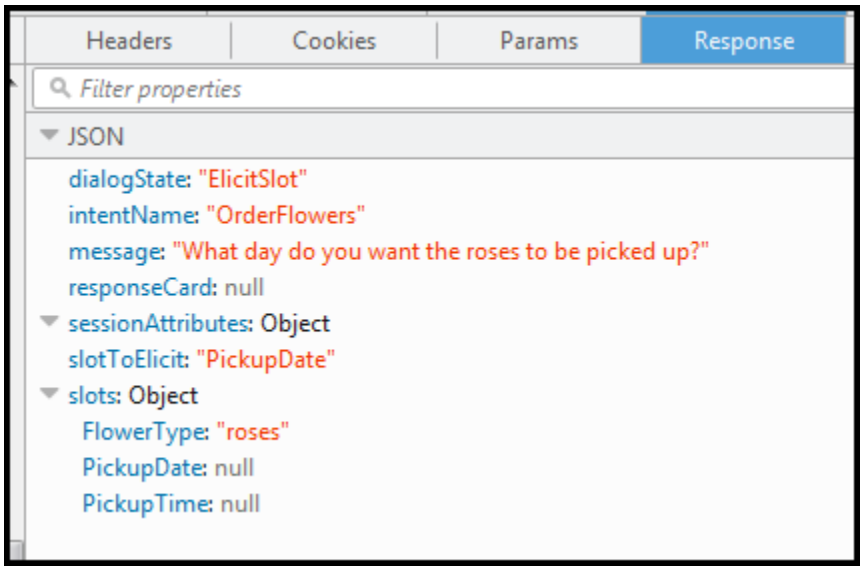
```
{
    "sessionAttributes": {
        "Price": 25
    },
    "dialogAction": {
        "type": "Delegate",
        "slots": {
            "PickupTime": null,
            "FlowerType": "roses",
            "PickupDate": null
        }
    }
```

```
}
```

Perhatikan hal berikut:

- `sessionAttributes`- Fungsi Lambda telah menambahkan`Price` (mawar) sebagai atribut sesi.
- `dialogAction.type`- diatur ke`Delegate`. Data pengguna valid sehingga fungsi Lambda mengarahkan Amazon Lex untuk memilih tindakan berikutnya.

d. Menurut`dialogAction.type`, Amazon Lex memilih tindakan berikutnya. Amazon Lex tahu itu membutuhkan lebih banyak data slot sehingga mengambil slot yang tidak terisi berikutnya (`PickupDate`) dengan prioritas tertinggi sesuai dengan konfigurasi maksud. Amazon Lex memilih salah satu pesan prompt nilai-elicitation - “Hari apa Anda ingin mawar dijemput?” —untuk slot ini sesuai dengan konfigurasi maksud, dan kemudian mengirimkan respons berikut kembali ke klien:



Klien hanya menampilkan pesan dalam respons - “Hari apa Anda ingin mawar dijemput?.”

3. Pengguna: besok

a. Klien mengirimkan[PostText](#) permintaan berikut ke Amazon Lex:

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

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

Dalam badan permintaan,`inputText` menyediakan input pengguna dan klien melewati atribut sesi kembali ke layanan.

b. Amazon Lex ingat konteks-bahwa itu memunculkan data untuk`PickupDate` slot. Dalam konteks ini, ia tahu`inputText` nilai adalah untuk`PickupDate` slot. Amazon Lex kemudian memanggil fungsi Lambda dengan mengirimkan peristiwa berikut:

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

Amazon Lex telah memperbarui`currentIntent.slots` dengan menetapkan`PickupDate` nilai. Perhatikan juga bahwa layanan melewati fungsi Lambda.`sessionAttributes`

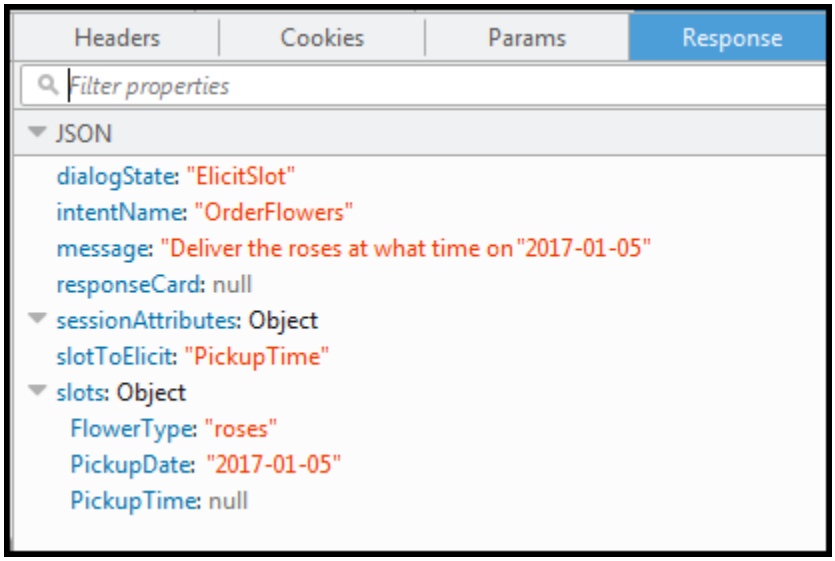
c. Sesuai`invocationSource` nilai`DialogCodeHook`, fungsi Lambda melakukan validasi data pengguna. Ia mengakui nilai`PickupDate` slot yang valid dan mengembalikan respon berikut ke Amazon Lex:

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

Perhatikan hal berikut:

- `sessionAttributes`Tidak ada perubahan.
- `dialogAction.type`- diatur ke`Delegate`. Data pengguna valid, dan fungsi Lambda mengarahkan Amazon Lex untuk memilih tindakan berikutnya.

d. Menurut`dialogAction.type`, Amazon Lex memilih tindakan berikutnya. Amazon Lex tahu itu membutuhkan lebih banyak data slot sehingga mengambil slot yang tidak terisi berikutnya (`PickupTime`) dengan prioritas tertinggi sesuai dengan konfigurasi maksud. Amazon Lex memilih salah satu pesan prompt (“Memberikan mawar pada jam berapa pada 2017-01-05?”) untuk slot ini sesuai dengan konfigurasi maksud dan mengirimkan respons berikut kembali ke klien:



Klien menampilkan pesan dalam tanggapan - “Memberikan mawar pada jam berapa pada 2017-01-05?”

4. Pengguna: 4 sore

   a. Klien mengirimkanPostText permintaan berikut ke Amazon Lex:

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

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

   Dalam badan permintaan,`inputText` menyediakan input pengguna. Klien melewati`sessionAttributes` permintaan.

   b. Amazon Lex memahami konteksnya. Ia memahami bahwa itu memunculkan data untuk`PickupTime` slot. Dalam konteks ini, ia tahu bahwa`inputText` nilai adalah untuk`PickupTime` slot. Amazon Lex kemudian memanggil fungsi Lambda dengan mengirimkan peristiwa berikut:

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

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

Amazon Lex telah memperbarui`currentIntent.slots` dengan menetapkan`PickupTime` nilai.

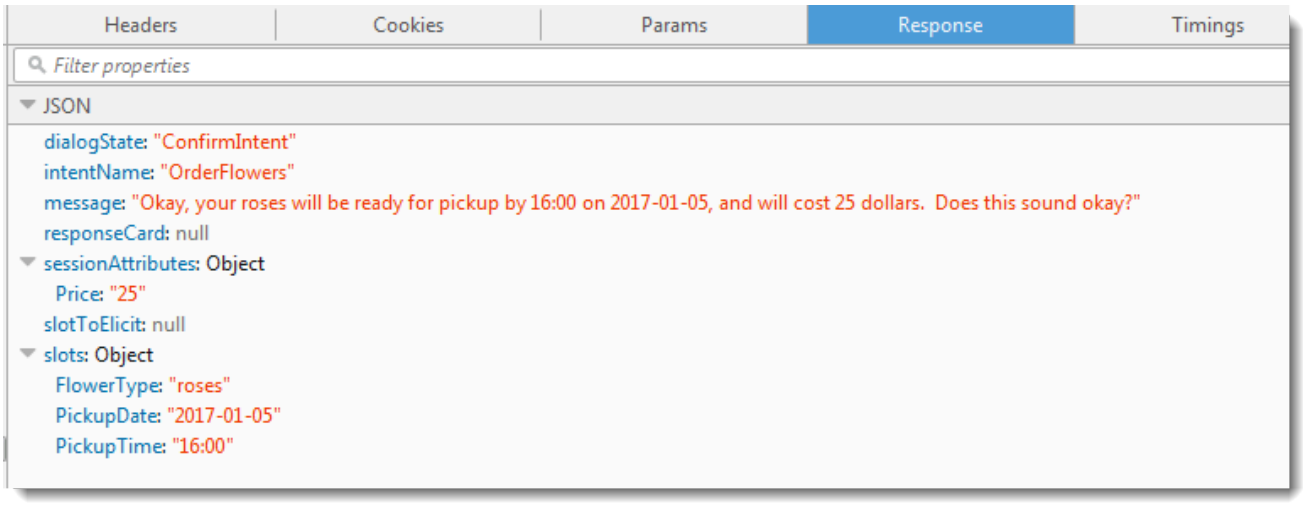
c. Menurut`invocationSource` nilai`DialogCodeHook`, fungsi Lambda melakukan validasi data pengguna. Ia mengakui nilai`PickupDate` slot yang valid dan mengembalikan respon berikut untuk Amazon Lex.

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

Perhatikan hal berikut:

- `sessionAttributes`- Tidak ada perubahan atribut sesi.
- `dialogAction.type`- diatur ke`Delegate`. Data pengguna valid sehingga fungsi Lambda mengarahkan Amazon Lex untuk memilih tindakan berikutnya.

d. Pada saat ini Amazon Lex tahu ia memiliki semua data Slot. Maksud ini dikonfigurasi dengan prompt konfirmasi. Oleh karena itu, Amazon Lex mengirimkan respons berikut kepada pengguna yang meminta konfirmasi sebelum memenuhi maksud:



Klien hanya menampilkan pesan dalam respons dan menunggu respons pengguna.

5. Pengguna: Ya

a. Klien mengirimkanPostText permintaan berikut ke Amazon Lex:

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

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

b. Amazon Lex menafsirkan`inputText` dalam konteks mengonfirmasi maksud saat ini. Amazon Lex memahami bahwa pengguna ingin melanjutkan pesanan. Kali ini Amazon Lex

memanggil fungsi Lambda untuk memenuhi maksud dengan mengirimkan peristiwa berikut, yang menetapkan`invocationSource` ke`FulfillmentCodeHook` jika ia mengirim ke fungsi Lambda. Amazon Lex juga menetapkan`confirmationStatus` untuk`Confirmed`.

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

Perhatikan hal berikut:

- `invocationSource`— Kali ini Amazon Lex menetapkan nilai ini`FulfillmentCodeHook`, mengarahkan fungsi Lambda untuk memenuhi maksud.
- `confirmationStatus`- diatur ke`Confirmed`.

c. Kali ini, fungsi Lambda memenuhi`OrderFlowers` maksud, dan mengembalikan respons berikut:

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

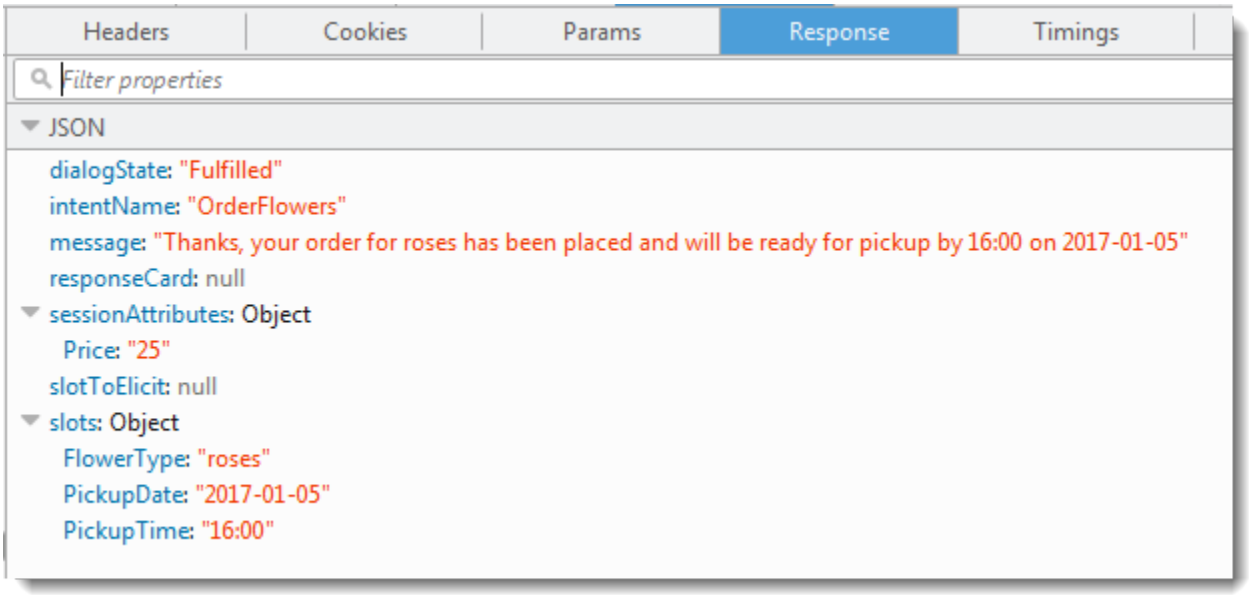
```
        "type": "Close",
        "fulfillmentState": "Fulfilled",
        "message": {
            "contentType": "PlainText",
            "content": "Thanks, your order for roses has been placed and will
 be ready for pickup by 16:00 on 2017-01-05"
        }
    }
}
```

Perhatikan hal berikut:

- Menetapkan`dialogAction.type` - Fungsi Lambda menetapkan nilai ini ke`Close`, mengarahkan Amazon Lex untuk tidak mengharapkan respons pengguna.
- `dialogAction.fulfillmentState`- diatur ke Terpenuhi dan termasuk yang sesuai`message` untuk disampaikan kepada pengguna.

d. Amazon Lex mengulas`fulfillmentState` dan mengirimkan respons berikut kembali ke klien.

Amazon Lex kemudian mengembalikan hal berikut ke klien:



Perhatikan bahwa:

- `dialogState`- Amazon Lex menetapkan nilai ini ke`fulfilled`.
- `message`- adalah pesan yang sama dengan fungsi Lambda yang disediakan.

Klien menampilkan pesan.

6. Sekarang uji bot lagi. Untuk membuat konteks (pengguna) baru, pilih tautan Hapus di jendela pengujian. Sekarang berikan data slot yang tidak valid untuk`OrderFlowers` maksud tersebut. Kali ini fungsi Lambda melakukan validasi data, menyetel ulang nilai data slot yang tidak valid ke null, dan meminta Amazon Lex untuk meminta pengguna untuk data yang valid. Misalnya, coba hal berikut ini:

   - Melati sebagai jenis bunga (bukan salah satu jenis bunga yang didukung).
   - Kemarin sebagai hari ketika Anda ingin mengambil bunga.
   - Setelah melakukan pemesanan, masukkan jenis bunga lain alih-alih menjawab “ya” untuk mengonfirmasi pesanan. Sebagai tanggapan, fungsi Lambda memperbarui atribut`Price` in the session, menjaga total pesanan bunga yang berjalan.

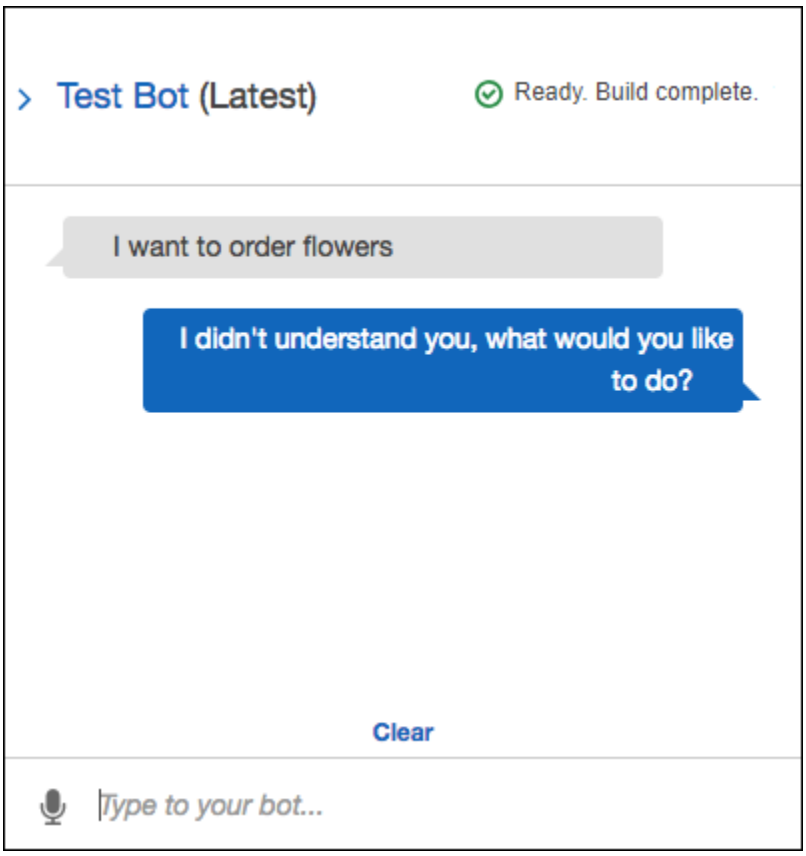
   Fungsi Lambda juga melakukan aktivitas pemenuhan.

Langkah Selanjutnya

Langkah 6: Perbarui konfigurasi maksud untuk menambahkan ucapan (konsol)

Langkah 6: Perbarui konfigurasi maksud untuk menambahkan ucapan (konsol)

`OrderFlowers`Bot dikonfigurasi hanya dengan dua ucapan. Ini memberikan informasi terbatas bagi Amazon Lex untuk membangun model pembelajaran mesin yang mengenali dan merespons maksud pengguna. Coba ketik “Saya ingin memesan bunga”, seperti pada jendela tes berikut. Amazon Lex tidak mengenali teks, dan menjawab dengan “Saya tidak mengerti Anda, apa yang ingin Anda lakukan?” Anda dapat meningkatkan model pembelajaran mesin dengan menambahkan lebih banyak ucapan.



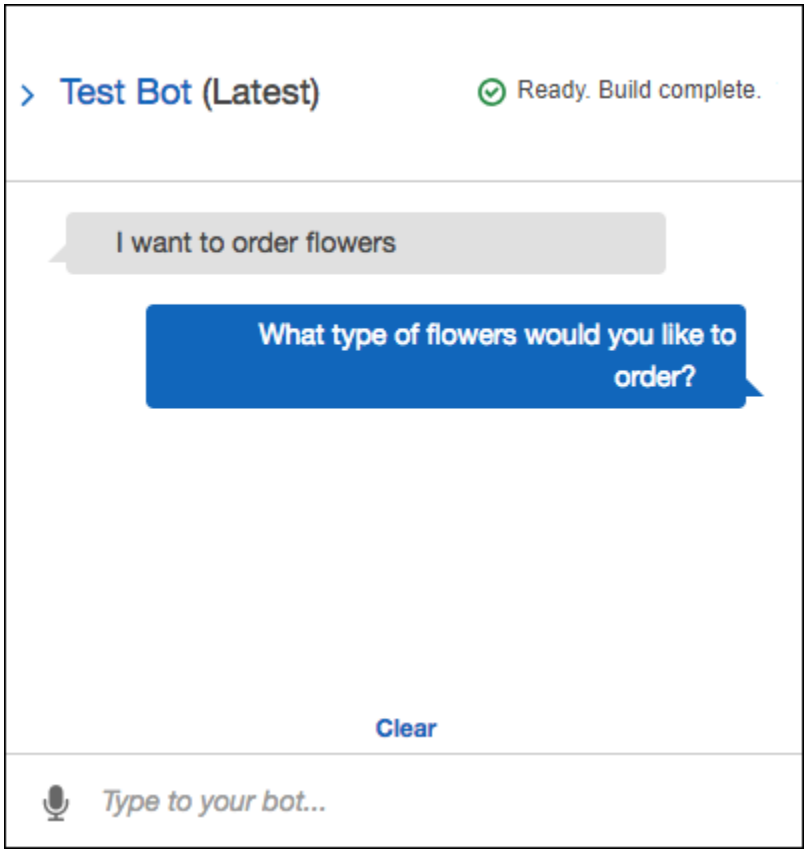
Setiap ucapan yang Anda tambahkan memberi Amazon Lex informasi lebih lanjut tentang cara merespons pengguna Anda. Anda tidak perlu menambahkan ucapan yang tepat, Amazon Lex menggeneralisasi dari sampel yang Anda berikan untuk mengenali kecocokan persis dan input serupa.

Untuk menambahkan ucapan (konsol)

1. Tambahkan ucapan “Saya ingin bunga” ke maksud dengan mengetiknya di bagian Sampel ucapan editor maksud, seperti pada gambar berikut, dan kemudian mengklik ikon plus di sebelah ucapan baru.

2. Bangun bot Anda untuk mengambil perubahan. Pilih Bangun, lalu pilih Bangun lagi.

3. Uji bot Anda untuk mengonfirmasi bahwa ia mengenali ucapan baru. Di jendela uji, seperti pada gambar berikut, ketik “Saya ingin memesan bunga.” Amazon Lex mengenali frasa tersebut dan merespons dengan “Jenis bunga apa yang ingin Anda pesan?”.



Langkah Selanjutnya

[Langkah 7 (Opsional): Bersihkan (Konsol)]

Langkah 7 (Opsional): Bersihkan (Konsol)

Sekarang, hapus sumber daya yang Anda buat dan bersihkan akun Anda.

Anda hanya dapat menghapus sumber daya yang tidak digunakan. Secara umum, Anda harus menghapus sumber daya dalam urutan berikut:

- Hapus bot untuk mengosongkan sumber daya maksud.
- Hapus maksud untuk membebaskan sumber daya jenis slot.
- Hapus jenis slot yang lalu.

Untuk membersihkan akun Anda (konsol)

1. Masuk keAWS Management Console dan buka konsol Amazon Lex di https://console.aws.amazon.com/lex/.
2. Dari daftar bot, pilih kotak di samping OrderFlowers.
3. Untuk menghapus bot, pilih Hapus, lalu pilih Lanjutkan di kotak dialog konfirmasi.
4. Di panel sebelah kiri, pilih Maksud.
5. Dalam daftar intent, pilih OrderFlowersIntent.
6. Untuk menghapus maksud, pilih Hapus, lalu pilih Lanjutkan di kotak dialog konfirmasi.
7. Di panel sebelah kiri, pilih Jenis slot.
8. Dalam daftar jenis slot, pilih Bunga.
9. Untuk menghapus jenis slot, pilih Hapus, lalu pilih Lanjutkan di kotak dialog konfirmasi.

Anda telah menghapus semua sumber daya Amazon Lex yang Anda buat dan dibersihkan akun Anda. Jika diinginkan, Anda dapat menggunakan konsol Lambda untuk menghapus fungsi Lambda yang digunakan dalam latihan ini.

Latihan 2: Membuat Bot Amazon Lex Kustom

Dalam latihan ini, Anda menggunakan konsol Amazon Lex untuk membuat bot khusus yang memesan pizza (`OrderPizzaBot`). Anda mengonfigurasi bot dengan menambahkan intent (`OrderPizza`) khusus, menentukan jenis slot khusus, dan menentukan slot yang diperlukan untuk memenuhi pesanan pizza (kerak pizza, ukuran, dan sebagainya). Untuk informasi lebih lanjut tentang jenis slot dan slot, melihatAmazon Lex: Cara Kerjanya.

Topik

- Langkah 1: Membuat Fungsi Lambda
- Langkah 2: Buat Bot
- Langkah 3: Bangun dan Uji Bot
- Langkah 4 (Opsional): (Opsional) Bersihkan

Langkah 1: Membuat Fungsi Lambda

Pertama, buat fungsi Lambda yang memenuhi pesanan pizza. Anda perlu menyebutkan fungsi ini di bot Amazon Lex, yang Anda buat di bagian berikutnya.

Untuk membuat fungsi Lambda

1. Masuk ke AWS Management Console dan buka konsol AWS Lambda di https://console.aws.amazon.com/lambda/.
2. Pilih Buat fungsi.
3. Pilih halaman Buat fungsi, pilih Penulis dari scratch.

   Karena Anda menggunakan kode kustom yang diberikan kepada Anda dalam latihan ini untuk membuat fungsi Lambda, Anda memilih penulis fungsi dari awal.

   Lakukan hal berikut:

   a. Ketik nama (`PizzaOrderProcessor`).
   b. Untuk Runtime, pilih versi terbaru dari Node.js.
   c. Untuk Peran, pilih Buat peran baru dari templat.
   d. Masukkan nama peran baru (`PizzaOrderProcessorRole`).
   e. Pilih Buat fungsi.
4. Di halaman fungsi, lakukan hal berikut:

   Di bagian Kode fungsi, pilih Edit kode sebaris, lalu salin kode fungsi Node.js berikut dan tempelkan di jendela.

```
'use strict';

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

// --------------- Events -----------------------
```

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

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

}

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

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

5. Pilih Save (Simpan).

Uji Fungsi Lambda Menggunakan Data Peristiwa Sampel

Di konsol, uji fungsi Lambda dengan menggunakan data kejadian sampel untuk memanggilnya secara manual.

Untuk menguji fungsi Lambda:

1. Masuk ke AWS Management Console dan buka konsol AWS Lambda di https://console.aws.amazon.com/lambda/.
2. Pada halaman fungsi Lambda, pilih fungsi Lambda (`PizzaOrderProcessor`).

3. Pada halaman fungsi, dalam daftar peristiwa pengujian, pilih Konfigurasikan peristiwa pengujian.
4. Pada halaman Konfigurasikan peristiwa pengujian, lakukan hal berikut:

   a. Pilih Buat peristiwa pengujian baru.

   b. Di bidang Nama peristiwa, masukkan nama untuk acara (`PizzaOrderProcessorTest`).

   c. Salin acara Amazon Lex berikut ke jendela.

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

5. Pilih Create (Buat).

AWS Lambdamenciptakan tes dan Anda kembali ke halaman fungsi. Pilih Test dan Lambda menjalankan fungsi Lambda Anda.

Di kotak hasil, pilih Detail. Konsol menampilkan output berikut di panel Hasil eksekusi.

```
{
  "sessionAttributes": {},
  "dialogAction": {
    "type": "Close",
    "fulfillmentState": "Fulfilled",
```

```
    "message": {
      "contentType": "PlainText",
      "content": "Okay, I have ordered your large meat pizza on thin crust."
    }
}
```

Langkah Selanjutnya

Langkah 2: Buat Bot

Langkah 2: Buat Bot

Pada langkah ini, Anda membuat bot untuk menangani pesanan pizza.

Topik

- Buat Bot
- Membuat Intent
- Buat Jenis Slot
- Mengkonfigurasi Intent
- Konfigurasikan Bot

Buat Bot

Buat`PizzaOrderingBot` bot dengan informasi minimum yang dibutuhkan. Anda menambahkan intent, tindakan yang ingin dilakukan pengguna, untuk bot nanti.

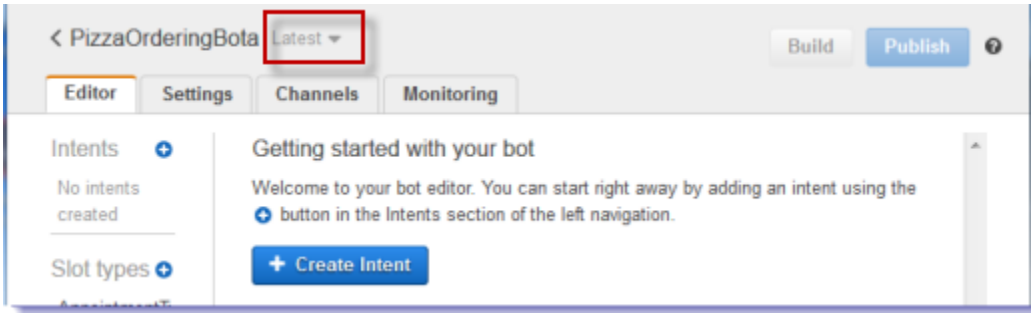
Untuk membuat bot

1. Masuk keAWS Management Console dan buka konsol Amazon Lex di https://console.aws.amazon.com/lex/.
2. Buat bot.
   a. Jika Anda membuat bot pertama Anda, pilih Memulai. Jika tidak, pilih Bot, lalu pilih Buat.
   b. Pada halaman Create your Lex bot, pilih Custom bot dan berikan informasi berikut:
      - Nama bot: PizzaOrderingBot
      - Bahasa: Pilih bahasa dan lokal untuk bot Anda.
      - Suara keluaran: Salli

- Batas waktu sesi: 5 menit.
- COPPA: Pilih respons yang sesuai.
- Penyimpanan ucapan pengguna: Pilih respons yang sesuai.

c. Pilih Create (Buat).

Konsol Amazon Lex untuk membuat bot baru. Amazon Lex menetapkan versi bot ke`$LATEST`. Setelah membuat bot, Amazon Lex menampilkan tab Editor bot, seperti pada gambar berikut:



- Versi bot, Terbaru, muncul di sebelah nama bot di konsol. Sumber daya Amazon Lex baru memiliki`$LATEST` versi. Untuk informasi selengkapnya, lihat Pembuatan Versi dan Alias.
- Karena Anda belum membuat maksud atau jenis slot, tidak ada yang terdaftar.
- Bangun dan Publikasikan adalah aktivitas tingkat bot. Setelah Anda mengkonfigurasi seluruh bot, Anda akan mempelajari lebih lanjut tentang aktivitas ini.

Langkah Selanjutnya

Membuat Intent

Membuat Intent

Sekarang, buat`OrderPizza` intent, tindakan yang ingin dilakukan pengguna, dengan informasi minimum yang diperlukan. Anda menambahkan jenis slot untuk maksud dan kemudian mengkonfigurasi intent nanti.

Membuat intent

1. Di konsol Amazon Lex, pilih tanda tambah (+) di sebelah Intent, lalu pilih Buat maksud baru.
2. Di kotak dialog Create intent, ketik nama intent (`OrderPizza`), lalu pilih Add.

Konsol mengirimkan permintaan ke Amazon Lex untuk membuat`OrderPizza` intent. Dalam contoh ini Anda membuat slot untuk maksud setelah Anda membuat jenis slot.

Langkah Selanjutnya

[Buat Jenis Slot]

Buat Jenis Slot

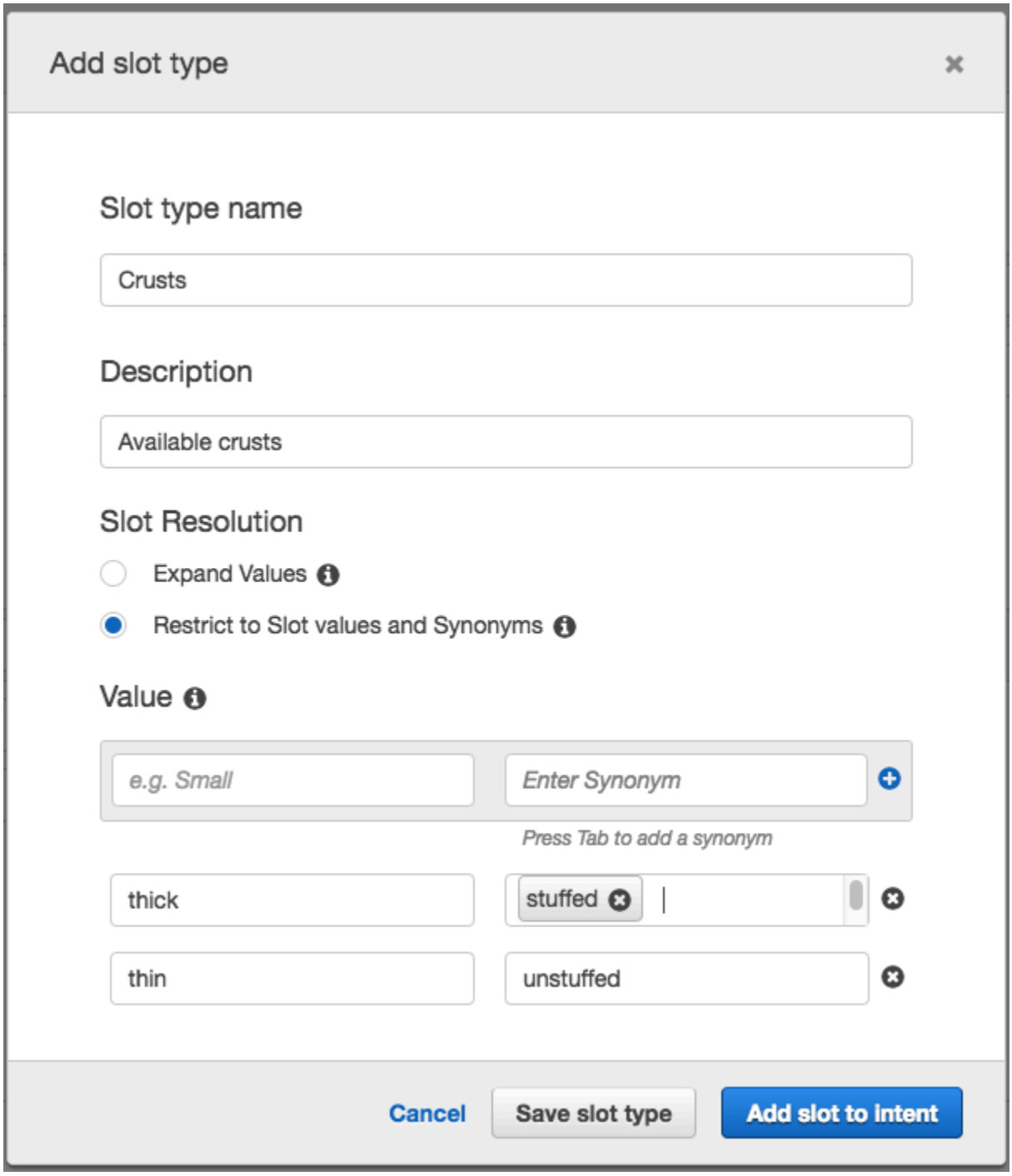
Buat jenis slot, atau nilai parameter, yang digunakan`OrderPizza` maksud.

Untuk membuat jenis slot

1. Di menu sebelah kiri, pilih tanda tambah (+) di sebelah jenis Slot.
2. Dalam Tambahkan jenis slot kotak dialog, tambahkan yang berikut ini:

   - Nama jenis Slot - Kerak
   - Deskripsi - Kerak yang tersedia
   - Pilih Batasi nilai Slot dan Sinonim
   - Nilai - Jenis**`thick`**. Tekan tab dan dalam jenis bidang sinonim**`stuffed`**. Pilih tanda tambah (+). Ketik**`thin`** dan pilih tanda tambah (+) lagi.

   Dialog akan terlihat seperti gambar berikut:



3. Pilih Tambahkan slot ke maksud.
4. Pada halaman Intent, pilih Diperlukan. Ubah nama slot dari**`slotOne`** ke**`crust`**. Ubah prompt ke**`What kind of crust would you like?`**
5. UlangiStep 1 melaluiStep 4 menggunakan nilai-nilai dalam tabel berikut:

Nama	Penjelasan	Nilai	Nama slot	Prompt
Ukuran	Ukuran yang tersedia	kecil, sedang, besar	size	Berapa ukuran pizza?
PizzaKind	Pizza yang tersedia	sayuran, keju	PizzaKind	Apakah Anda ingin pizza sayuran atau keju?

Langkah Selanjutnya

Mengkonfigurasi Intent

Mengkonfigurasi Intent

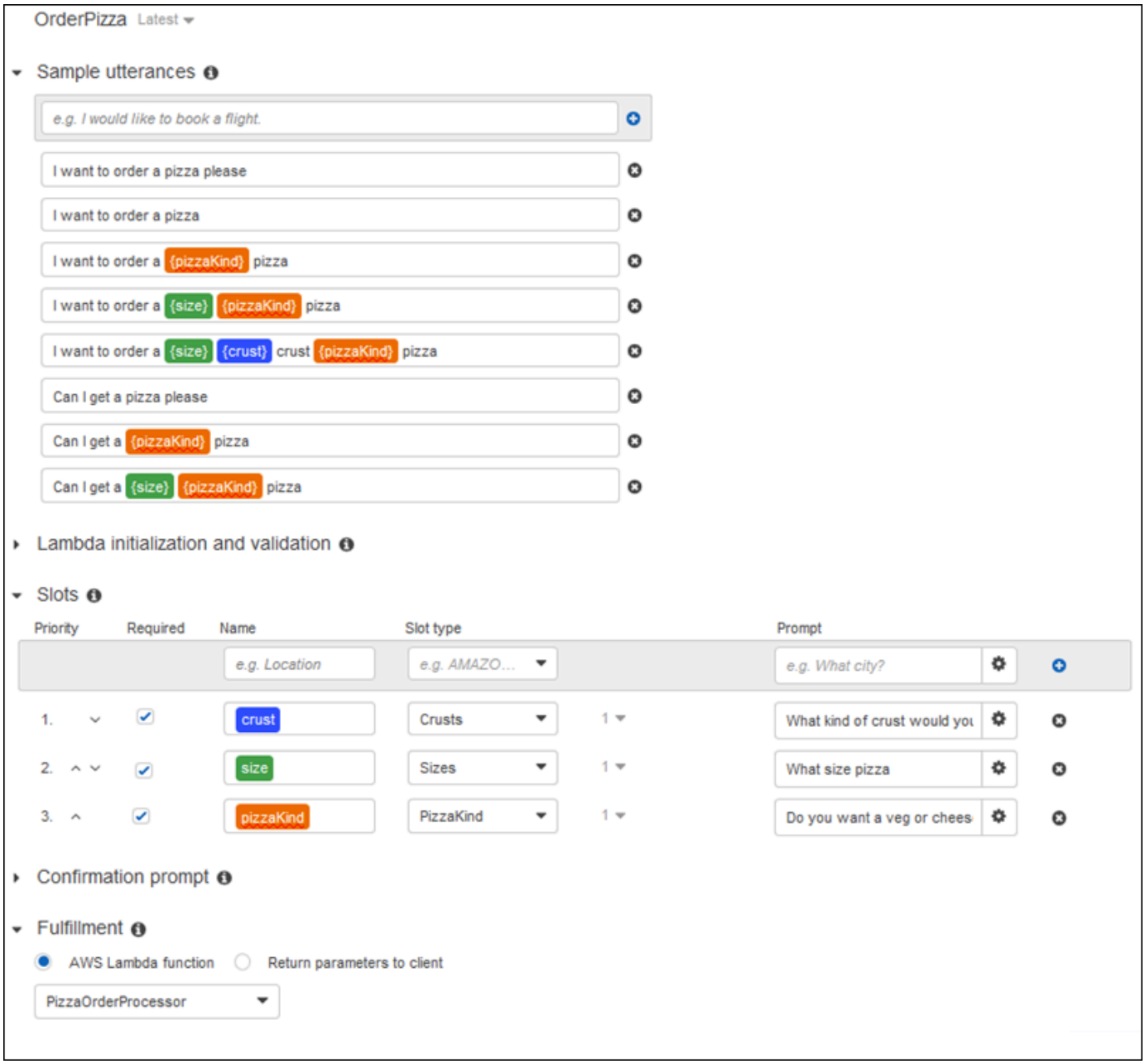
Konfigurasikan`OrderPizza` intent untuk memenuhi permintaan pengguna agar memesan pizza.

Mengonfigurasi maksud

- Pada halaman OrderPizzakonfigurasi, konfigurasikan maksud sebagai berikut:
  - Sampel ucapan - Ketik string berikut. Kurung kurawal {} melampirkan nama slot.
    - Saya ingin memesan pizza silakan
    - Saya ingin memesan pizza
    - Saya ingin memesan pizza {PizzaKind}
    - Saya ingin memesan pizza {size} {PizzaKind}
    - Saya ingin {size} {crust} crust {PizzaKind} pizza
    - Bisakah saya mendapatkan pizza
    - Bisakah saya mendapatkan pizza {PizzaKind}
    - Bisakah saya mendapatkan {size} {PizzaKind} pizza
  - Inisialisasi dan validasi Lambda - Tinggalkan pengaturan default.
  - Prompt konfirmasi - Tinggalkan pengaturan default.
  - Pemenuhan - Lakukan tugas-tugas berikut:
    - Pilih AWS Lambdafungsi.

- Pilih **`PizzaOrderProcessor`**.
- Jika kotak dialog Tambahkan izin ke fungsi Lambda ditampilkan, pilih OK untuk memberikan izin`OrderPizza` maksud untuk memanggil fungsi`PizzaOrderProcessor` Lambda.
- Tinggalkan None dipilih.

Maksud akan terlihat seperti berikut ini:



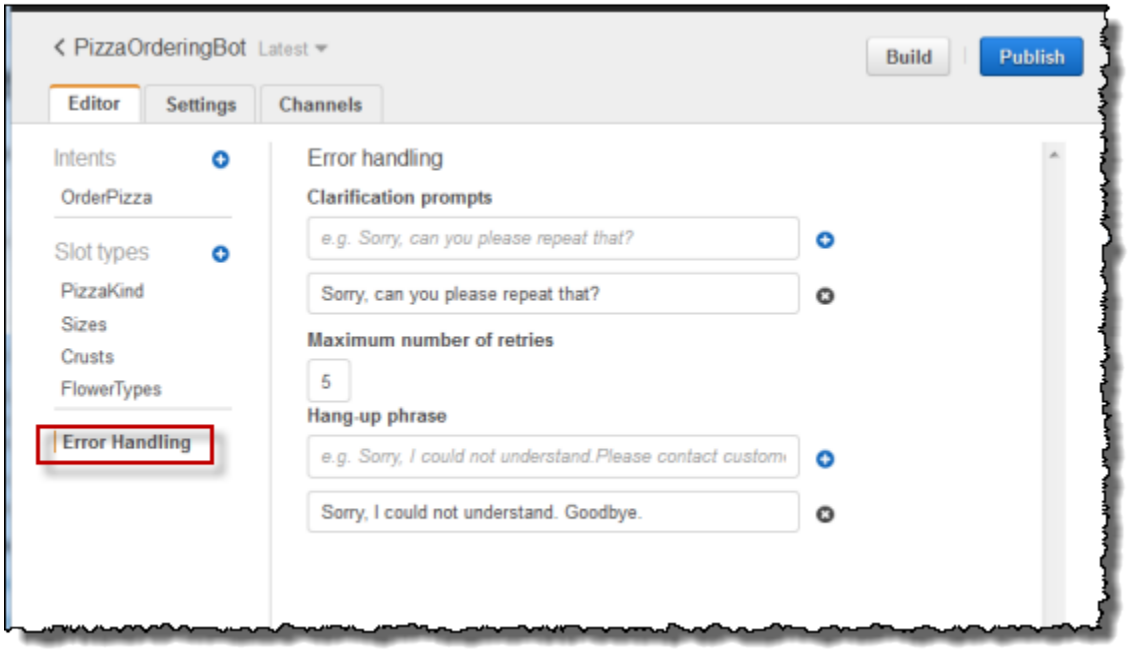
Langkah Selanjutnya

[Konfigurasikan Bot]

Konfigurasikan Bot

Konfigurasikan penanganan kesalahan untuk`PizzaOrderingBot` bot.

1. Arahkan ke`PizzaOrderingBot` bot. Pilih Editor. dan kemudian pilih Penanganan Kesalahan, seperti pada gambar berikut:



2. Gunakan tab Editor untuk mengonfigurasi penanganan kesalahan bot.

   - Informasi yang Anda berikan di Prompt Klarifikasi dipetakan ke konfigurasi clarificationPrompt bot.

     Jika Amazon Lex tidak dapat menentukan maksud pengguna, layanan akan menampilkan respons dengan pesan ini.

   - Informasi yang Anda berikan di peta frase Hang-up ke konfigurasi AbortStatement bot.

     Jika layanan tidak dapat menentukan maksud pengguna setelah sejumlah permintaan berturut-turut yang ditetapkan, Amazon Lex mengembalikan respons dengan pesan ini.

   Biarkan default.

Langkah Selanjutnya

Langkah 3: Bangun dan Uji Bot

Langkah 3: Bangun dan Uji Bot

Pastikan bot bekerja, dengan membangun dan mengujinya.

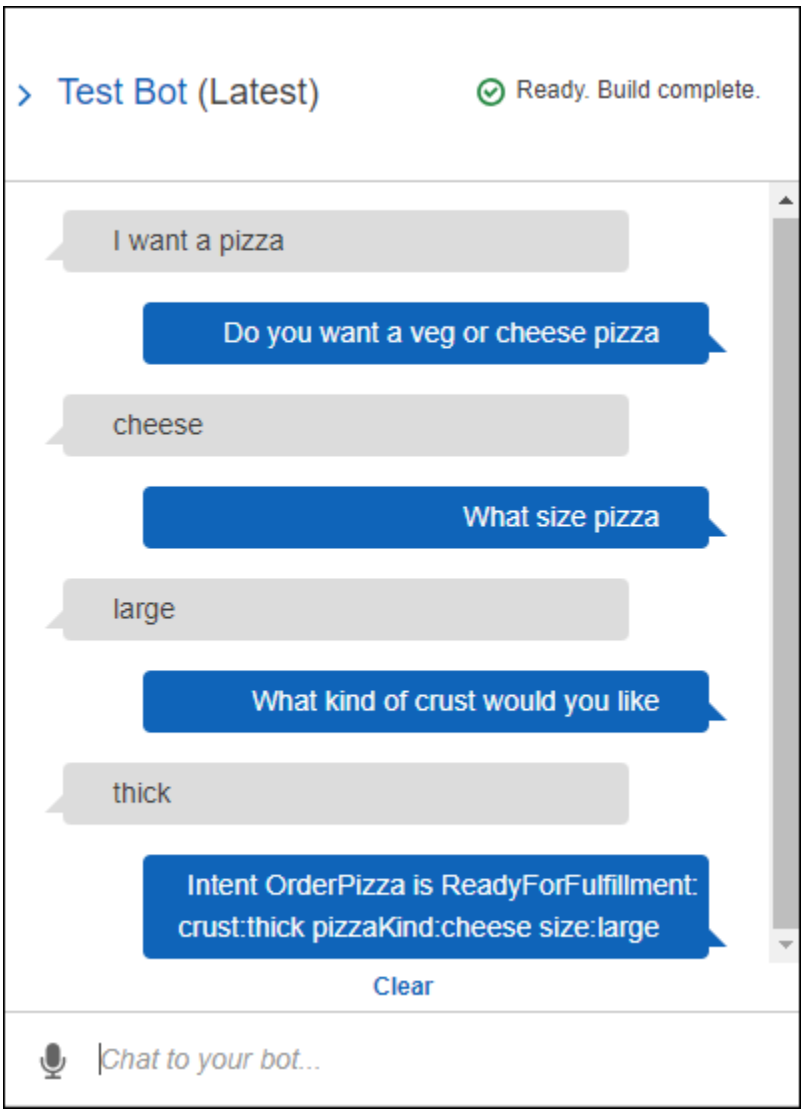
Untuk membangun dan menguji bot

1. Untuk membangun`PizzaOrderingBot` bot, pilih Build.

   Amazon Lex membuat model pembelajaran mesin untuk bot. Saat Anda menguji bot, konsol menggunakan API waktu proses untuk mengirim input pengguna kembali ke Amazon Lex. Amazon Lex kemudian menggunakan model machine learning untuk menafsirkan input pengguna.

   Dibutuhkan beberapa waktu untuk menyelesaikan bangunan.
2. Untuk menguji bot, di jendela Test Bot, mulailah berkomunikasi dengan bot Amazon Lex Anda.

   - Misalnya, Anda dapat mengatakan atau mengetikkan hal berikut:



- Gunakan contoh ucapan yang Anda konfigurasikan dalam`OrderPizza` intent untuk menguji bot. Misalnya, berikut ini adalah salah satu contoh ucapan yang Anda konfigurasikan untuk`PizzaOrder` intent:

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

  Untuk mengujinya, ketik yang berikut ini:

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

Ketika Anda mengetik “Saya ingin memesan pizza,” Amazon Lex mendeteksi maksud (`OrderPizza`). Kemudian, Amazon Lex meminta informasi Slot.

Setelah Anda memberikan semua informasi slot, Amazon Lex memanggil fungsi Lambda yang Anda dikonfigurasi untuk maksud.

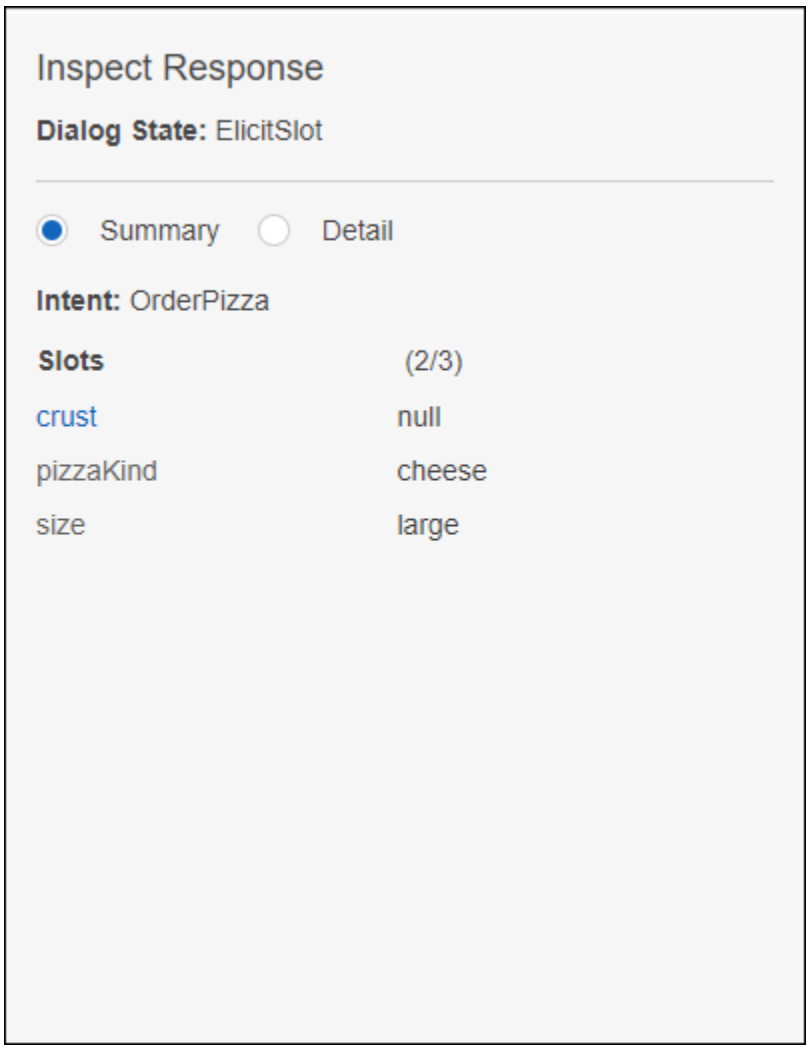
Fungsi Lambda mengembalikan pesan (“Oke, saya telah memesan Anda...”) ke Amazon Lex, yang Amazon Lex kembali kepada Anda..

Memeriksa Respon

Di bawah jendela obrolan terdapat panel yang memungkinkan Anda memeriksa respons dari Amazon Lex. Panel memberikan informasi komprehensif tentang status bot Anda yang berubah saat Anda berinteraksi dengan bot Anda. Isi panel menunjukkan keadaan operasi saat ini.

- Status Dialog — Status percakapan saat ini dengan pengguna. Hal ini dapat`ElicitIntent`,`ElicitSlot`,`ConfirmIntent` atau`Fulfilled`.

- Ringkasan - Menampilkan tampilan dialog yang disederhanakan yang menunjukkan nilai slot untuk maksud yang dipenuhi sehingga Anda dapat melacak alur informasi. Ini menunjukkan nama maksud, jumlah slot dan jumlah slot diisi, dan daftar semua slot dan nilai-nilai yang terkait. Lihat citra berikut:



- Detail - Menampilkan respons JSON mentah dari chatbot untuk memberi Anda tampilan yang lebih dalam tentang interaksi bot dan status dialog saat ini saat Anda menguji dan men-debug chatbot Anda. Jika Anda mengetik di jendela obrolan, panel inspeksi menampilkan respons JSON dariPostText operasi. Jika Anda berbicara dengan jendela obrolan, panel inspeksi menampilkan header respons dariPostContent operasi. Lihat citra berikut:

Langkah Selanjutnya

[Langkah 4 (Opsional): (Opsional) Bersihkan](#)

Langkah 4 (Opsional): (Opsional) Bersihkan

Hapus sumber daya yang Anda buat dan bersihkan akun agar tidak menimbulkan biaya lebih untuk sumber daya yang Anda buat.

Anda hanya dapat menghapus sumber daya yang tidak digunakan. Misalnya, Anda tidak dapat menghapus jenis slot yang direferensikan oleh maksud. Anda tidak dapat menghapus maksud yang direferensikan oleh bot.

Hapus sumber daya dalam urutan berikut:

- Hapus bot untuk mengosongkan sumber daya maksud.

- Hapus maksud untuk membebaskan sumber daya jenis slot.
- Hapus jenis slot yang lalu.

Membersihkan akun Anda

1. Masuk keAWS Management Console dan buka konsol Amazon Lex di https://console.aws.amazon.com/lex/.
2. Dari daftar bot, pilih PizzaOrderingBot.
3. Untuk menghapus bot, pilih Hapus, lalu pilih Lanjutkan.
4. Di panel sebelah kiri, pilih Maksud.
5. Dalam daftar intent, pilih OrderPizza.
6. Untuk menghapus maksud, pilih Hapus, lalu pilih Lanjutkan.
7. Di menu sebelah kiri, pilih jenis Slot.
8. Dalam daftar jenis slot, pilih Kerak.
9. Untuk menghapus jenis slot, pilih Hapus, lalu pilih Lanjutkan.
10. UlangiStep 8 danStep 9 untuk`Sizes` dan jenis`PizzaKind` slot.

Anda telah menghapus semua sumber daya yang Anda buat dan bersihkan akun Anda.

Langkah Selanjutnya

- Publikasikan Versi dan Buat Alias
- Buat bot Amazon Lex denganAWS Command Line Interface

Latihan 3: Publikasikan Versi dan Buat Alias

Dalam Latihan Memulai 1 dan 2, Anda membuat bot dan mengujinya. Dalam latihan ini, Anda melakukan hal berikut:

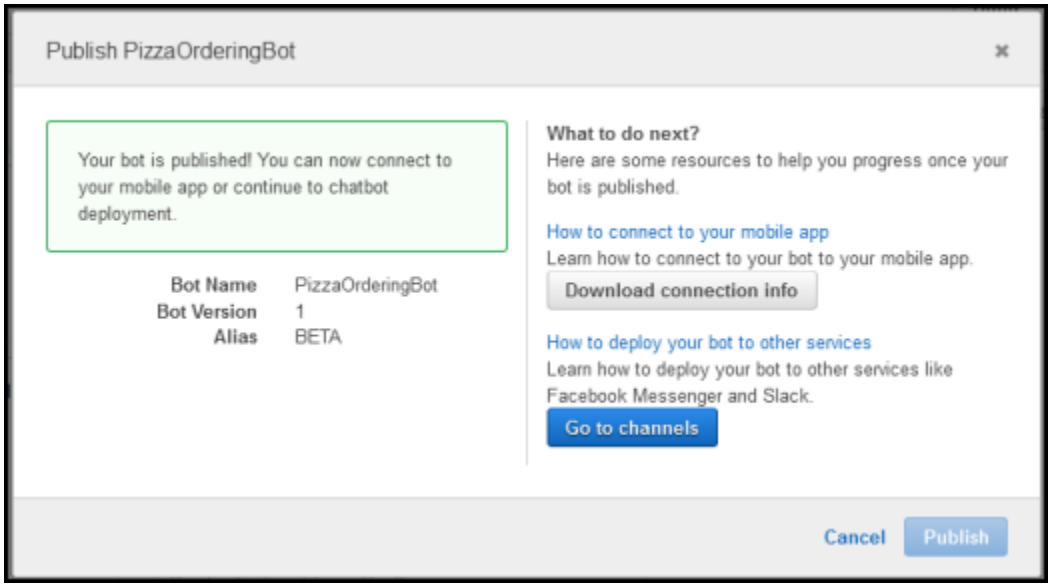
- Terbitkan versi baru dari bot. Amazon Lex mengambil salinan snapshot`$LATEST` versi untuk mempublikasikan versi baru.
- Buat alias yang menunjuk ke versi baru.

Untuk informasi selengkapnya tentang pembuatan versi dan alias, lihatPembuatan Versi dan Alias.

Lakukan hal berikut untuk mempublikasikan versi bot yang Anda buat untuk latihan ini:

1. Di konsol Amazon Lex, pilih salah satu bot yang Anda buat.

   Verifikasi bahwa konsol menunjukkan`$LATEST` sebagai versi bot di sebelah nama bot.
2. Pilih Terbitkan.
3. Pada Wisaya Publish ***botname***, tentukan alias**BETA**, lalu pilih Publikasikan.
4. Verifikasi bahwa konsol Amazon Lex menampilkan versi baru di sebelah nama bot, seperti pada gambar berikut.



Sekarang setelah Anda memiliki bot yang berfungsi dengan versi yang diterbitkan dan alias, Anda dapat menggunakan bot (di aplikasi seluler Anda atau mengintegrasikan bot dengan Facebook Messenger). Sebagai contoh, lihat Mengintegrasikan Amazon Lex Bot dengan Facebook Messenger.

Langkah 4: Memulai (AWS CLI)

Pada langkah ini, Anda menggunakanAWS CLIuntuk membuat, menguji, dan memodifikasi bot Amazon Lex. Untuk menyelesaikan latihan ini, Anda harus terbiasa dengan menggunakan CLI dan memiliki editor teks. Untuk informasi selengkapnya, lihatLangkah 2: Siapkan AWS Command Line Interface

- Latihan 1 - Buat dan uji bot Amazon Lex. Latihan ini menyediakan semua objek JSON yang Anda butuhkan untuk membuat jenis slot kustom, maksud, dan bot. Untuk informasi selengkapnya, lihatAmazon Lex: Cara Kerjanya

- Latihan 2 - Perbarui bot yang Anda buat di Latihan 1 untuk menambahkan ucapan sampel tambahan. Amazon Lex menggunakan contoh ucapan untuk membangun model machine learning untuk bot Anda.
- Latihan 3 - Perbarui bot yang Anda buat di Latihan 1 untuk menambahkan fungsi Lambda untuk memvalidasi input pengguna dan untuk memenuhi maksud.
- Latihan 4 - Publikasikan versi jenis slot, maksud, dan sumber bot yang Anda buat di Latihan 1. Versi adalah snapshot sumber daya yang tidak dapat diubah.
- Latihan 5 - Buat alias untuk bot yang Anda buat di Latihan 1.
- Latihan 6 - Bersihkan akun Anda dengan menghapus jenis slot, maksud, dan bot yang Anda buat di Latihan 1, dan alias yang Anda buat di Latihan 5.

Topik

Latihan 1: Buat Amazon Lex Bot (AWS CLI)

Secara umum, ketika Anda membuat bot, Anda:

1. Buat jenis slot untuk menentukan informasi yang bot Anda akan bekerja dengan.
2. Buat maksud yang menentukan tindakan pengguna yang didukung bot Anda. Gunakan jenis slot khusus yang Anda buat sebelumnya untuk menentukan slot, atau parameter yang diperlukan oleh maksud Anda.
3. Buat bot yang menggunakan maksud yang Anda tetapkan.

Dalam latihan ini Anda membuat dan menguji bot Amazon Lex baru menggunakan CLI. Gunakan struktur JSON yang kami sediakan untuk membuat bot. Untuk menjalankan perintah dalam latihan ini, Anda perlu mengetahui wilayah di mana perintah akan dijalankan. Untuk daftar wilayah, lihat Kuotas Bangunan .

Topik

Langkah 1: Membuat Peran Terkait Layanan (AWS CLI)

Amazon Lex mengasumsikanAWS Identity and Access Managementperan yang terhubung dengan layanan untuk dipanggilAWSlayanan atas nama bot Anda. Peran, yang ada di akun Anda, ditautkan ke kasus penggunaan Amazon Lex dan memiliki izin yang telah ditentukan. Untuk informasi selengkapnya, lihat Menggunakan Peran tertaut layanan untuk Amazon Lex.

Jika Anda telah membuat bot Amazon Lex menggunakan konsol, peran tertaut-layanan akan dibuat secara otomatis. Loncat ke Langkah 2: Buat Jenis Slot Kustom (AWS CLI).

Untuk membuat peran terkait layanan (AWS CLI)

1. DiAWS CLI, ketik perintah berikut:

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

2. Periksa kebijakan dengan menggunakan perintah berikut ini:

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

   Tanggapannya adalah:

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

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

Langkah Selanjutnya

Langkah 2: Buat Jenis Slot Kustom (AWS CLI)

Langkah 2: Buat Jenis Slot Kustom (AWS CLI)

Buat jenis slot kustom dengan nilai pencacahan untuk bunga yang dapat dipesan. Anda menggunakan tipe ini di langkah berikutnya ketika Anda membuat`OrderFlowers`niat. SEBUAH`Jenis Slot`mendefinisikan nilai yang mungkin untuk slot, atau parameter, maksud.

Untuk menjalankan perintah dalam latihan ini, Anda perlu mengetahui wilayah di mana perintah akan dijalankan. Untuk daftar wilayah, lihat Kuotas Bangunan .

Membuat tipe kustom (AWS CLI)

1. Membuat file teks bernama**`FlowerTypes.json`**. Salin kode JSON dariFlowertypes.JSONke dalam file teks.
2. PanggilPutSlotTypeoperasi menggunakanAWS CLIuntuk membuat jenis slot. Contohnya diformat untuk Unix, Linux, dan macOS. Untuk Windows, ganti karakter kelanjutan Unix (\) di akhir setiap baris dengan caret (^).

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

   Respons dari server adalah:

   ```
   {
       "enumerationValues": [
   ```

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

Langkah Selanjutnya

[Langkah 3: Buat Intent (AWS CLI)]

Flowertypes.JSON

Kode berikut adalah data JSON yang diperlukan untuk membuat`FlowerTypes`jenis slot kustom:

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

Langkah 3: Buat Intent (AWS CLI)

Membuat maksud untuk`OrderFlowersBot`bot dan memberikan tiga slot, atau parameter. Slot memungkinkan bot untuk memenuhi maksud:

- `FlowerType`adalah jenis slot kustom yang menentukan jenis bunga dapat dipesan.
- `AMAZON.DATE`dan`AMAZON.TIME`built-in jenis Slot yang digunakan untuk mendapatkan tanggal dan waktu untuk memberikan bunga dari pengguna.

Untuk menjalankan perintah dalam latihan ini, Anda perlu mengetahui wilayah di mana perintah akan dijalankan. Untuk daftar wilayah, lihat Kuotas Bangunan .

Untuk membuat**`OrderFlowers`**maksud (AWS CLI)

1. Membuat file teks bernama**`OrderFlowers.json`**. Salin kode JSON dariOrderflowers.jsonke dalam file teks.
2. DiAWS CLI, hubungiPutIntentoperasi untuk membuat maksud. Contohnya diformat untuk Unix, Linux, dan macOS. Untuk Windows, ganti karakter kelanjutan Unix (\) di akhir setiap baris dengan caret (^).

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

   Server merespons dengan berikut ini:

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

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

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

Langkah Selanjutnya

Langkah 4: Buat Bot (AWS CLI)

Orderflowers.json

Kode berikut adalah data JSON yang diperlukan untuk membuat`OrderFlowers`maksud:

```
{
```

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

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

Langkah 4: Buat Bot (AWS CLI)

Parameter`OrderFlowersBot`bot memiliki satu maksud,`OrderFlowers`maksud yang Anda buat di langkah sebelumnya. Untuk menjalankan perintah dalam latihan ini, Anda perlu mengetahui wilayah di mana perintah akan dijalankan. Untuk daftar wilayah, lihat Kuotas Bangunan .

> **Note**
>
> BerikutAWS CLIcontoh diformat untuk Unix, Linux, dan macOS. Untuk Windows, ubah"\ `$LATEST"`kepada`$LATEST`.

Untuk membuat**`OrderFlowersBot`**Bot (AWS CLI)

1. Membuat file teks bernama**`OrderFlowersBot.json`**. Salin kode JSON dariOrderflowersbot.jsonke dalam file teks.
2. DiAWS CLI, hubungiPutBotoperasi untuk membuat bot. Contohnya diformat untuk Unix, Linux, dan macOS. Untuk Windows, ganti karakter kelanjutan Unix (\) di akhir setiap baris dengan caret (^).

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

   Respon dari server berikut. Ketika Anda membuat atau memperbarui bot,`status`bidang diatur ke`BUILDING`. Hal ini menunjukkan bahwa bot belum siap untuk digunakan. Untuk menentukan kapan bot siap digunakan, gunakanGetBotoperasi di langkah selanjutnya.

   ```
   {
       "status": "BUILDING",
       "intents": [
           {
               "intentVersion": "$LATEST",
               "intentName": "OrderFlowers"
           }
       ],
       "name": "OrderFlowersBot",
       "locale": "en-US",
       "checksum": "checksum",
   ```

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

3. Untuk menentukan apakah bot baru Anda siap digunakan, jalankan perintah berikut. Ulangi perintah ini sampai`status`kembali bidang`READY`. Contohnya diformat untuk Unix, Linux, dan macOS. Untuk Windows, ganti karakter kelanjutan Unix (\) di akhir setiap baris dengan caret (^).

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

Cari`status`bidang dalam respon:

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

    ...
```

```
}
```

Langkah Selanjutnya

[Langkah 5: Uji Bot (AWS CLI)]

Orderflowersbot.json

Kode berikut menyediakan data JSON yang diperlukan untuk membangun`OrderFlowers`Amazon Lex:

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

```
}
```

Langkah 5: Uji Bot (AWS CLI)

Untuk menguji bot, Anda dapat menggunakan tes berbasis teks atau berbasis pidato.

Topik

- Uji Bot Menggunakan Input Teks (AWS CLI)
- Uji Bot Menggunakan Input Ucapan (AWS CLI)

Uji Bot Menggunakan Input Teks (AWS CLI)

Untuk memverifikasi bahwa bot bekerja dengan benar dengan input teks, gunakanPostTextOperasi. Untuk menjalankan perintah dalam latihan ini, Anda perlu mengetahui wilayah di mana perintah akan dijalankan. Untuk daftar wilayah, lihatService Quotas Runtime.

> **Note**
>
> BerikutAWS CLIcontoh diformat untuk Unix, Linux, dan macOS. Untuk Windows, ubah"\$LATEST"kepada$LATESTdan ganti karakter kelanjutan backslash (\) di akhir setiap baris dengan caret (^).

Untuk menggunakan teks untuk menguji bot (AWS CLI)

1. DiAWS CLI, memulai percakapan dengan`OrderFlowersBot`Bot. Contohnya diformat untuk Unix, Linux, dan macOS. Untuk Windows, ganti karakter kelanjutan Unix (\) di akhir setiap baris dengan caret (^).

   ```
   aws lex-runtime post-text \
       --region region \
       --bot-name OrderFlowersBot \
       --bot-alias "\$LATEST" \
       --user-id UserOne \
       --input-text "i would like to order flowers"
   ```

   Amazon Lex mengenali maksud pengguna dan memulai percakapan dengan mengembalikan respons berikut:

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

2. Jalankan perintah berikut untuk menyelesaikan percakapan dengan bot.

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

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

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

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

Setelah Anda mengkonfirmasi pesanan, Amazon Lex mengirimkan respons pemenuhan untuk menyelesaikan percakapan:

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

Langkah Selanjutnya

Uji Bot Menggunakan Input Ucapan (AWS CLI)

Uji Bot Menggunakan Input Ucapan (AWS CLI)

Untuk menguji bot menggunakan file audio, gunakanPostContentOperasi. Anda membuat file audio menggunakan operasi text-to-speech Amazon Polly.

Untuk menjalankan perintah dalam latihan ini, Anda perlu mengetahui wilayah perintah Amazon Lex dan Amazon Polly akan dijalankan. Untuk daftar wilayah Amazon Lex, lihatService Quotas Runtime. Untuk daftar wilayah untuk Amazon Polly lihatAWSWilayah dan Titik AkhirdiReferensi Umum Amazon Web Services.

> **Note**
>
> BerikutAWS CLIcontoh diformat untuk Unix, Linux, dan macOS. Untuk Windows, ubah"\$LATEST"kepada$LATESTdan ganti karakter kelanjutan backslash (\) di akhir setiap baris dengan caret (^).

Untuk menggunakan input pidato untuk menguji bot (AWS CLI)

1. DiAWS CLI, buat file audio menggunakan Amazon Polly. Contohnya diformat untuk Unix, Linux, dan macOS. Untuk Windows, ganti karakter kelanjutan Unix (\) di akhir setiap baris dengan caret (^).

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

2. Untuk mengirim file audio ke Amazon Lex, jalankan perintah berikut. Amazon Lex menyimpan audio dari respons dalam file output yang ditentukan.

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

Amazon Lex merespon dengan permintaan untuk pertama celah. Ini menghemat respon audio dalam file output yang ditentukan.

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

3. Untuk memesan mawar, buat file audio berikut dan kirimkan ke Amazon Lex:

```
aws polly synthesize-speech \
    --region region \
    --output-format pcm \
```

```
    --text "roses" \
    --voice-id "Salli" \
    FlowerTypeSpeech.mpg
```

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

4. Untuk mengatur tanggal pengiriman, buat file audio berikut dan kirimkan ke Amazon Lex:

```
aws polly synthesize-speech \
    --region region \
    --output-format pcm \
    --text "tuesday" \
    --voice-id "Salli" \
    DateSpeech.mpg
```

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

5. Untuk mengatur waktu pengiriman, buat file audio berikut dan kirimkan ke Amazon Lex:

```
aws polly synthesize-speech \
    --region region \
    --output-format pcm \
    --text "10:00 a.m." \
    --voice-id "Salli" \
    TimeSpeech.mpg
```

```
aws lex-runtime post-content \
```

```
    --region region \
    --bot-name OrderFlowersBot \
    --bot-alias "\$LATEST" \
    --user-id UserOne \
    --content-type "audio/l16; rate=16000; channels=1" \
    --input-stream TimeSpeech.mpg \
    TimeOutputSpeech.mpg
```

6. Untuk mengkonfirmasi pengiriman, buat file audio berikut dan kirimkan ke Amazon Lex:

```
aws polly synthesize-speech \
    --region region \
    --output-format pcm \
    --text "yes" \
    --voice-id "Salli" \
    ConfirmSpeech.mpg
```

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

Setelah Anda mengonfirmasi pengiriman, Amazon Lex mengirimkan respons yang mengonfirmasi pemenuhan maksud:

```
{
    "contentType": "text/plain;charset=utf-8",
    "dialogState": "ReadyForFulfillment",
    "intentName": "OrderFlowers",
    "inputTranscript": "yes",
    "slots": {
        "PickupDate": "2017-05-16",
        "PickupTime": "10:00",
        "FlowerType": "roses"
    }
}
```

Langkah Selanjutnya

Latihan 2: Menambahkan Ucapan Baru (AWS CLI)

Latihan 2: Menambahkan Ucapan Baru (AWS CLI)

Untuk meningkatkan model machine learning yang digunakan Amazon Lex untuk mengenali permintaan dari pengguna Anda, tambahkan contoh ucapan lain ke bot.

Menambahkan ucapan baru adalah proses empat langkah.

1. MenggunakanGetIntentoperasi untuk mendapatkan maksud dari Amazon Lex.
2. Perbarui intent.
3. MenggunakanPutIntentoperasi untuk mengirim niat diperbarui kembali ke Amazon Lex.
4. MenggunakanGetBotdanPutBotoperasi untuk membangun kembali bot apa pun yang menggunakan maksud.

Untuk menjalankan perintah dalam latihan ini, Anda perlu mengetahui wilayah di mana perintah akan dijalankan. Untuk daftar wilayah, lihat Kuotas Bangunan .

Respons dari`GetIntent`operasi berisi bidang yang disebut`checksum`yang mengidentifikasi revisi spesifik maksud. Anda harus memberikan nilai checksum saat Anda menggunakanPutIntentoperasi untuk memperbarui maksud. Jika tidak, Anda akan mendapatkan pesan galat berikut ini:

```
An error occurred (PreconditionFailedException) when calling
the PutIntent operation: Intent intent name already exists.
If you are trying to update intent name you must specify the
checksum.
```

> **Note**
>
> BerikutAWS CLIcontoh diformat untuk Unix, Linux, dan macOS. Untuk Windows, ubah"\ `$LATEST`"kepada`$LATEST`dan ganti karakter kelanjutan backslash (\) di akhir setiap baris dengan caret (^).

Untuk memperbarui**OrderFlowers**maksud (AWS CLI)

1. DiAWS CLI, dapatkan maksud dari Amazon Lex. Amazon Lex mengirimkan output ke file bernama**OrderFlowers-V2.json.**

```
aws lex-models get-intent \
    --region region \
    --name OrderFlowers \
    --intent-version "\$LATEST" > OrderFlowers-V2.json
```

2. Buka **OrderFlowers-V2.json** di editor teks.

   1. Menemukan dan menghapus`createdDate`,`lastUpdatedDate`, dan`version`bidang.
   2. Menambahkan berikut ini ke`sampleUtterances`Bidang:

```
I want to order flowers
```

   3. Simpan file.

3. Kirim intent yang diperbarui ke Amazon Lex dengan perintah berikut:

```
aws lex-models put-intent  \
    --region region \
    --name OrderFlowers \
    --cli-input-json file://OrderFlowers-V2.json
```

   Amazon Lex mengirimkan tanggapan berikut:

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

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

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

Sekarang setelah Anda memperbarui intent, membangun kembali bot apa pun yang menggunakannya.

Untuk membangun kembali**`OrderFlowersBot`**Bot (AWS CLI)

1. DiAWS CLI, dapatkan definisi`OrderFlowersBot`bot dan simpan ke file dengan perintah berikut:

```
aws lex-models get-bot \
```

```
    --region region \
    --name OrderFlowersBot \
    --version-or-alias "\$LATEST" > OrderFlowersBot-V2.json
```

2. Di editor teks, buka**`OrderFlowersBot-V2.json`**. Hapus`createdDate`,`lastUpdatedDate`,`status`dan`version`bidang.

3. Pada editor teks, tambahkan baris berikut ke definisi bot:

```
"processBehavior": "BUILD",
```

4. DiAWS CLI, membangun revisi baru bot dengan menjalankan perintah berikut untuk:

```
aws lex-models put-bot \
    --region region \
    --name OrderFlowersBot \
    --cli-input-json file://OrderFlowersBot-V2.json
```

Respons dari server adalah:

```
{
    "status": "BUILDING",
    "intents": [
        {
            "intentVersion": "$LATEST",
            "intentName": "OrderFlowers"
        }
    ],
    "name": "OrderFlowersBot",
    "locale": "en-US",
    "checksum": "checksum",
    "abortStatement": {
        "messages": [
            {
                "content": "Sorry, I'm not able to assist at this time",
                "contentType": "PlainText"
            }
        ]
    },
    "version": "$LATEST",
    "lastUpdatedDate": timestamp,
    "createdDate": timestamp
    "clarificationPrompt": {
```

```
        "maxAttempts": 2,
        "messages": [
            {
                "content": "I didn't understand you, what would you like to do?",
                "contentType": "PlainText"
            }
        ]
    },
    "voiceId": "Salli",
    "childDirected": false,
    "idleSessionTTLInSeconds": 600,
    "description": "Bot to order flowers on the behalf of a user"
}
```

Langkah Selanjutnya

Latihan 3: Tambahkan Fungsi Lambda (AWS CLI)

Latihan 3: Tambahkan Fungsi Lambda (AWS CLI)

Tambahkan fungsi Lambda yang memvalidasi input pengguna dan memenuhi maksud pengguna ke bot.

Menambahkan ekspresi Lambda adalah proses lima langkah.

1. Gunakan LambdaAddPermissionberfungsi untuk mengaktifkan`OrderFlowers`berniat untuk memanggil LambdaAktifkanOperasi.
2. MenggunakanGetIntentoperasi untuk mendapatkan maksud dari Amazon Lex.
3. Perbarui maksud untuk menambahkan fungsi Lambda.
4. MenggunakanPutIntentoperasi untuk mengirim niat diperbarui kembali ke Amazon Lex.
5. MenggunakanGetBotdanPutBotoperasi untuk membangun kembali bot apa pun yang menggunakan maksud.

Untuk menjalankan perintah dalam latihan ini, Anda perlu mengetahui wilayah di mana perintah akan dijalankan. Untuk daftar wilayah, lihat Kuotas Bangunan .

Jika Anda menambahkan fungsi Lambda ke intent sebelum menambahkan`InvokeFunction`izin, Anda mendapatkan pesan galat berikut:

```
An error occurred (BadRequestException) when calling the
PutIntent operation: Lex is unable to access the Lambda
function Lambda function ARN in the context of intent
intent ARN.  Please check the resource-based policy on
the function.
```

Respons dariGetIntentoperasi berisi bidang yang disebutchecksumyang mengidentifikasi revisi spesifik maksud. Saat Anda menggunakanPutIntentoperasi untuk memperbarui maksud, Anda harus memberikan nilai checksum. Jika tidak, Anda mendapatkan pesan galat berikut ini:

```
An error occurred (PreconditionFailedException) when calling
the PutIntent operation: Intent intent name already exists.
If you are trying to update intent name you must specify the
checksum.
```

Latihan ini menggunakan fungsi Lambda dariLatihan 1: Membuat Bot Amazon Lex Menggunakan Cetak Biru (Konsol). Untuk membuat fungsi Lambda, lihatLangkah 3: Buat Fungsi Lambda (Konsol).

> **Note**
>
> BerikutAWS CLIcontoh diformat untuk Unix, Linux, dan macOS. Untuk Windows, ubah"\$LATEST"kepada$LATEST.

Untuk menambahkan fungsi Lambda ke intent

1. DiAWS CLI, tambahkanInvokeFunctionizin untukOrderFlowersmaksud:

```
aws lambda add-permission \
    --region region \
    --function-name OrderFlowersCodeHook \
    --statement-id LexGettingStarted-OrderFlowersBot \
    --action lambda:InvokeFunction \
    --principal lex.amazonaws.com \
    --source-arn "arn:aws:lex:region:account ID:intent:OrderFlowers:*"
    --source-account account ID
```

Lambda mengirimkan respons berikut:

```
{
    "Statement": "{\"Sid\":\"LexGettingStarted-OrderFlowersBot\",
      \"Resource\":\"arn:aws:lambda:region:account ID:function:OrderFlowersCodeHook
\",
      \"Effect\":\"Allow\",
      \"Principal\":{\"Service\":\"lex.amazonaws.com\"},
      \"Action\":[\"lambda:InvokeFunction\"],
      \"Condition\":{\"StringEquals\":
        {\"AWS:SourceAccount\": \"account ID\"},
        {\"AWS:SourceArn\":
          \"arn:aws:lex:region:account ID:intent:OrderFlowers:*\"}}}"
}
```

2. Dapatkan maksud dari Amazon Lex. Amazon Lex mengirimkan output ke file bernama**`OrderFlowers-V3.json`**.

```
aws lex-models get-intent \
    --region region \
    --name OrderFlowers \
    --intent-version "\$LATEST" > OrderFlowers-V3.json
```

3. Di editor teks, buka**`OrderFlowers-V3.json`**.

   1. Menemukan dan menghapus`createdDate`,`lastUpdatedDate`, dan`version`bidang.
   2. Perbarui`fulfillmentActivity`Bidang:

```
   "fulfillmentActivity": {
       "type": "CodeHook",
       "codeHook": {
           "uri": "arn:aws:lambda:region:account
ID:function:OrderFlowersCodeHook",
           "messageVersion": "1.0"
       }
   }
```

   3. Simpan file.

4. DiAWS CLI, kirim maksud yang diperbarui ke Amazon Lex:

```
aws lex-models put-intent \
```

```
    --region region \
    --name OrderFlowers \
    --cli-input-json file://OrderFlowers-V3.json
```

Sekarang setelah Anda memperbarui intent, membangun kembali bot.

Untuk membangun kembali**`OrderFlowersBot`**Bot

1. DiAWS CLI, dapatkan definisi`OrderFlowersBot`bot dan simpan ke file:

   ```
   aws lex-models get-bot \
       --region region \
       --name OrderFlowersBot \
       --version-or-alias "\$LATEST" > OrderFlowersBot-V3.json
   ```

2. Di editor teks, buka**`OrderFlowersBot-V3.json`**. Hapus`createdDate`,`lastUpdatedDate`,`status`, dan`version`bidang.

3. Di editor teks, tambahkan baris berikut ke definisi bot:

   ```
   "processBehavior": "BUILD",
   ```

4. DiAWS CLI, membangun revisi baru bot:

   ```
   aws lex-models put-bot \
       --region region \
       --name OrderFlowersBot \
       --cli-input-json file://OrderFlowersBot-V3.json
   ```

   Respons dari server adalah:

   ```
   {
       "status": "READY",
       "intents": [
           {
               "intentVersion": "$LATEST",
               "intentName": "OrderFlowers"
           }
       ],
       "name": "OrderFlowersBot",
       "locale": "en-US",
   ```

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

Langkah Selanjutnya

Latihan 4: Publikasikan Versi (AWS CLI)

Latihan 4: Publikasikan Versi (AWS CLI)

Sekarang, buat versi bot yang Anda buat di Latihan 1. SEBUAHversiadalah snapshot dari bot. Setelah membuat versi, Anda tidak dapat mengubahnya. Satu-satunya versi bot yang dapat Anda perbarui adalah$LATESTVersi. Untuk informasi selengkapnya tentang versi, lihatPembuatan Versi dan Alias.

Sebelum Anda dapat mempublikasikan versi bot, Anda harus mempublikasikan maksud yang digunakan. Demikian juga, Anda harus mempublikasikan jenis slot yang dimaksud dengan maksud tersebut. Secara umum, untuk mempublikasikan versi bot, Anda melakukan hal berikut:

1. Mempublikasikan versi tipe slot dengan CreateSlotTypeVersion Operasi.
2. Memublikasikan versi intent dengan CreateIntentVersion Operasi.
3. Memublikasikan versi bot dengan CreateBotVersion Operasi.

Untuk menjalankan perintah dalam latihan ini, Anda perlu mengetahui wilayah di mana perintah akan dijalankan. Untuk daftar wilayah, lihat Kuotas Bangunan .

Topik

- Langkah 1: Publikasikan Jenis Slot (AWS CLI)
- Langkah 2: Terbitkan Intent (AWS CLI)
- Langkah 3: Terbitkan Bot (AWS CLI)

Langkah 1: Publikasikan Jenis Slot (AWS CLI)

Sebelum Anda dapat mempublikasikan versi dari maksud apa pun yang menggunakan jenis slot, Anda harus mempublikasikan versi dari jenis slot itu. Dalam kasus ini, Anda mempublikasikan `FlowerTypes` jenis Slot.

> **Note**
>
> Berikut AWS CLI contoh diformat untuk Unix, Linux, dan macOS. Untuk Windows, ubah `"\$LATEST"` kepada `$LATEST` dan ganti karakter kelanjutan backslash (\) di akhir setiap baris dengan caret (^).

Untuk mempublikasikan jenis slot (AWS CLI)

1. Di AWS CLI, dapatkan versi terbaru dari jenis slot:

```
aws lex-models get-slot-type \
    --region region \
    --name FlowerTypes \
    --slot-type-version "\$LATEST"
```

Tanggapan dari Amazon Lex berikut. Catat checksum untuk revisi saat ini `$LATEST` Versi.

```
{
```

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

2. Mempublikasikan versi tipe slot. Gunakan checksum yang Anda catat di langkah sebelumnya.

```
aws lex-models create-slot-type-version \
    --region region \
    --name FlowerTypes \
    --checksum "checksum"
```

Tanggapan dari Amazon Lex berikut. Rekam nomor versi untuk langkah berikutnya.

```
{
    "version": "1",
    "enumerationValues": [
        {
            "value": "tulips"
        },
        {
            "value": "lilies"
        },
        {
            "value": "roses"
        }
    ],
    "name": "FlowerTypes",
    "createdDate": timestamp,
```

```
    "lastUpdatedDate": timestamp,
    "description": "Types of flowers to pick up"
}
```

Langkah Selanjutnya

Langkah 2: Terbitkan Intent (AWS CLI)

Langkah 2: Terbitkan Intent (AWS CLI)

Sebelum Anda dapat mempublikasikan maksud, Anda harus mempublikasikan semua jenis slot yang disebut maksud. Jenis slot harus diberi nomor versi, bukan`$LATEST`Versi.

Pertama, perbarui`OrderFlowers`bermaksud untuk menggunakan versi`FlowerTypes`Jenis Slot yang Anda publikasikan pada langkah sebelumnya. Kemudian publikasikan versi baru`OrderFlowers`niat.

> **Note**
>
> Berikut`AWS CLI`contoh diformat untuk Unix, Linux, dan macOS. Untuk Windows, ubah`"\$LATEST"`kepada`$LATEST`dan ganti karakter kelanjutan backslash (\) di akhir setiap baris dengan caret (^).

Untuk memublikasikan versi intent (AWS CLI)

1. Di`AWS CLI`, dapatkan`$LATEST`versi`OrderFlowers`maksud dan simpan ke file:

   ```
   aws lex-models get-intent \
       --region region \
       --name OrderFlowers \
       --intent-version "\$LATEST" > OrderFlowers_V4.json
   ```

2. Di editor teks, buka**`OrderFlowers_V4.json`**berkas. Hapus`createdDate`,`lastUpdatedDate`, dan`version`bidang. Menemukan`FlowerTypes`Jenis Slot dan mengubah versi ke nomor versi yang Anda catat di langkah sebelumnya. Fragmen berikut dari**`OrderFlowers_V4.json`**file menunjukkan lokasi perubahan:

   ```
           {
   ```

```
            "slotType": "FlowerTypes",
            "name": "FlowerType",
            "slotConstraint": "Required",
            "valueElicitationPrompt": {
                "maxAttempts": 2,
                "messages": [
                    {
                        "content": "What type of flowers?",
                        "contentType": "PlainText"
                    }
                ]
            },
            "priority": 1,
            "slotTypeVersion": "version",
            "sampleUtterances": []
        },
```

3. DiAWS CLI, simpan revisi intent:

```
aws lex-models put-intent \
    --name OrderFlowers \
    --cli-input-json file://OrderFlowers_V4.json
```

4. Dapatkan revisi intent terbaru:

```
aws lex-models get-intent \
    --region region \
    --name OrderFlowers \
    --intent-version "\$LATEST" > OrderFlowers_V4a.json
```

Fragmen respons berikut menunjukkan checksum intent. Rekam ini untuk langkah selanjutnya.

```
    "name": "OrderFlowers",
    "checksum": "checksum",
    "version": "$LATEST",
```

5. Memublikasikan versi baru maksud:

```
aws lex-models create-intent-version \
    --region region \
    --name OrderFlowers \
    --checksum "checksum"
```

Fragmen respons berikut menunjukkan maksud versi baru. Rekam nomor versi untuk langkah berikutnya.

```
    "name": "OrderFlowers",
    "checksum": "checksum",
    "version": "version",
```

Langkah Selanjutnya

[Langkah 3: Terbitkan Bot (AWS CLI)]

Langkah 3: Terbitkan Bot (AWS CLI)

Setelah Anda menerbitkan semua jenis slot dan maksud yang digunakan oleh bot Anda, Anda dapat mempublikasikan bot.

Perbarui`OrderFlowersBot`bot untuk menggunakan`OrderFlowers`maksud yang Anda perbarui di langkah sebelumnya. Kemudian, publikasikan versi baru`OrderFlowersBot`Bot.

> **Note**
>
> BerikutAWS CLIcontoh diformat untuk Unix, Linux, dan macOS. Untuk Windows, ubah`"\$LATEST"`kepada`$LATEST`dan ganti karakter kelanjutan backslash (\) di akhir setiap baris dengan caret (^).

Untuk mempublikasikan versi bot (AWS CLI)

1. DiAWS CLI, dapatkan`$LATEST`versi`OrderFlowersBot`bot dan simpan ke file:

```
aws lex-models get-bot \
    --region region \
    --name OrderFlowersBot \
    --version-or-alias "\$LATEST" > OrderFlowersBot_V4.json
```

2. Di editor teks, buka**`OrderFlowersBot_V4.json`**berkas. Hapus`createdDate`,`lastUpdatedDate`,`status`dan`version`bidang. Menemukan`OrderFlowers`maksud dan ubah versi ke nomor versi yang Anda catat di langkah sebelumnya. Fragmen berikut**`OrderFlowersBot_V4.json`**menunjukkan lokasi perubahan.

```
    "intents": [
        {
            "intentVersion": "version",
            "intentName": "OrderFlowers"
        }
```

3. DiAWS CLI, simpan revisi bot baru. Buat catatan nomor versi yang dikembalikan oleh panggilan keput-bot.

```
aws lex-models put-bot \
    --name OrderFlowersBot \
    --cli-input-json file://OrderFlowersBot_V4.json
```

4. Dapatkan checksum revisi bot terbaru. Gunakan nomor versi yang dikembalikan pada langkah 3.

```
aws lex-models get-bot \
    --region region \
    --version-or-alias version \
    --name OrderFlowersBot > OrderFlowersBot_V4a.json
```

Fragmen respon berikut menunjukkan checksum bot. Rekam ini untuk langkah selanjutnya.

```
    "name": "OrderFlowersBot",
    "locale": "en-US",
    "checksum": "checksum",
```

5. Memublikasikan versi baru bot:

```
aws lex-models create-bot-version \
    --region region \
    --name OrderFlowersBot \
    --checksum "checksum"
```

Fragmen respon berikut menunjukkan versi baru dari bot.

```
    "checksum": "checksum",
    "abortStatement": {
        ...
    },
    "version": "1",
```

```
    "lastUpdatedDate": timestamp,
```

Langkah Selanjutnya

Latihan 5: Membuat Alias (AWS CLI)

Latihan 5: Membuat Alias (AWS CLI)

Alias adalah penunjuk ke versi tertentu bot. Dengan alias Anda dapat dengan mudah memperbarui versi yang digunakan aplikasi klien Anda. Untuk informasi selengkapnya, lihatPembuatan Versi dan Alias.Untuk menjalankan perintah dalam latihan ini, Anda perlu mengetahui wilayah di mana perintah akan dijalankan. Untuk daftar wilayah, lihat Kuotas Bangunan .

Membuat alias (AWS CLI)

1. DiAWS CLI, dapatkan versi`OrderFlowersBot`bot yang Anda buatLatihan 4: Publikasikan Versi (AWS CLI).

   ```
   aws lex-models get-bot \
       --region region \
       --name OrderFlowersBot \
       --version-or-alias version > OrderFlowersBot_V5.json
   ```

2. Di editor teks, buka**`OrderFlowersBot_v5.json`**. Temukan dan rekam nomor versi.
3. DiAWS CLI, buat alias bot:

   ```
   aws lex-models put-bot-alias  \
       --region region \
       --name PROD \
       --bot-name OrderFlowersBot \
       --bot-version version
   ```

   Berikut ini adalah tanggapan dari server:

   ```
   {
       "name": "PROD",
       "createdDate": timestamp,
       "checksum": "checksum",
       "lastUpdatedDate": timestamp,
       "botName": "OrderFlowersBot",
   ```

```
    "botVersion": "1"
}}
```

Langkah Selanjutnya

[Latihan 6: Membersihkan (AWS CLI)]

Latihan 6: Membersihkan (AWS CLI)

Hapus sumber daya yang Anda buat dan bersihkan akun Anda.

Anda dapat menghapus hanya sumber daya yang tidak digunakan. Secara umum, Anda harus menghapus sumber daya dengan urutan berikut.

1. Hapus alias untuk membebaskan sumber daya bot.
2. Hapus bot untuk membebaskan sumber daya maksud.
3. Hapus maksud untuk membebaskan sumber daya jenis slot.
4. Hapus jenis slot.

Untuk menjalankan perintah dalam latihan ini, Anda perlu mengetahui wilayah di mana perintah akan dijalankan. Untuk daftar wilayah, lihat Kuotas Bangunan .

Untuk membersihkan akun Anda (AWS CLI)

1. DiAWS CLIbaris perintah, menghapus alias:

```
aws lex-models delete-bot-alias \
    --region region \
    --name PROD \
    --bot-name OrderFlowersBot
```

2. DiAWS CLIbaris perintah, menghapus bot:

```
aws lex-models delete-bot \
    --region region \
    --name OrderFlowersBot
```

3. DiAWS CLIbaris perintah, hapus intent:

```
aws lex-models delete-intent \
    --region region \
    --name OrderFlowers
```

4. DariAWS CLIbaris perintah, menghapus jenis slot:

```
aws lex-models delete-slot-type \
    --region region \
    --name FlowerTypes
```

Anda telah menghapus semua sumber daya yang Anda buat dan bersihkan akun Anda.

Pembuatan Versi dan Alias

Amazon Lex mendukung versi penerbitan bot, maksud, dan jenis slot sehingga Anda dapat mengontrol implementasi yang digunakan aplikasi klien Anda. SEBUAHversiadalah snapshot bernomor dari pekerjaan Anda yang dapat Anda publikasikan untuk digunakan di berbagai bagian alur kerja Anda, seperti pengembangan, penyebaran beta, dan produksi.

Bot Amazon Lex juga mendukung alias. Sesialiasadalah penunjuk ke versi tertentu dari bot. Dengan alias, Anda dapat dengan mudah memperbarui versi yang digunakan aplikasi klien Anda. Misalnya, Anda dapat mengarahkan alias ke versi 1 dari bot Anda. Saat Anda siap memperbarui bot, Anda mempublikasikan versi 2 dan mengubah alias untuk mengarah ke versi baru. Karena aplikasi Anda menggunakan alias alih-alih versi tertentu, semua klien Anda mendapatkan fungsionalitas baru tanpa perlu diperbarui.

Topik

- Versioning
- Alias

Versioning

Ketika Anda versi sumber daya Amazon Lex Anda membuat snapshot dari sumber daya sehingga Anda dapat menggunakan sumber daya seperti itu ada ketika versi dibuat. Setelah Anda membuat versi itu akan tetap sama saat Anda terus bekerja pada aplikasi Anda.

Versi $LATEST

Saat Anda membuat bot Amazon Lex, tipe slot, hanya ada satu versi,`$LATEST`versi.

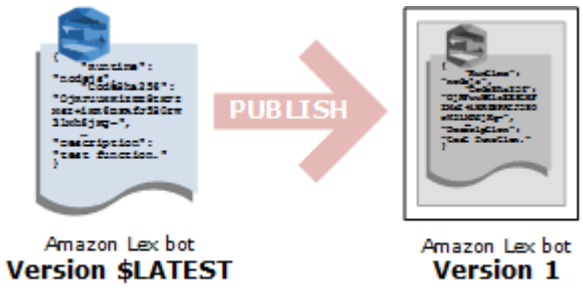
$LATESTadalah salinan kerja sumber daya Anda. Anda dapat memperbarui hanya$LATESTversi dan sampai Anda mempublikasikan versi pertama Anda,$LATESTadalah satu-satunya versi sumber daya yang Anda miliki.

Hanya$LATESTsumber daya dapat menggunakan$LATESTversi sumber daya lain. Misalnya,$LATESTversi bot dapat menggunakan$LATESTversi intent, dan$LATESTversi intent dapat menggunakan$LATESTversi dari jenis slot.

Parameter$LATESTversi bot Anda hanya boleh digunakan untuk pengujian manual. Amazon Lex membatasi jumlah permintaan runtime yang dapat Anda buat ke$LATESTversi bot.

Menerbitkan Versi Sumber Daya Amazon Lex

Ketika Anda mempublikasikan sumber daya, Amazon Lex membuat salinan$LATESTversi dan menyimpannya sebagai versi bernomor. Versi yang dipublikasikan tidak dapat diubah.



Anda membuat dan memublikasikan versi menggunakan konsol Amazon Lex atauCreateBotVersionoperasi. Sebagai contoh, lihat Latihan 3: Publikasikan Versi dan Buat Alias.

Saat Anda memodifikasi$LATESTversi sumber daya, Anda dapat mempublikasikan versi baru untuk membuat perubahan yang tersedia untuk aplikasi klien Anda. Setiap kali Anda mempublikasikan versi, Amazon Lex menyalin$LATESTversi untuk membuat versi baru dan menambah nomor versi dengan 1. Nomor versi tidak pernah digunakan kembali. Misalnya, jika Anda menghapus sumber daya bernomor versi 10 dan kemudian menciptakannya kembali, nomor versi berikutnya yang diberikan Amazon Lex adalah versi 11.

Sebelum dapat memublikasikan bot, Anda harus mengarahkannya ke versi bernomor dari maksud apa pun yang digunakannya. Jika Anda mencoba menerbitkan versi baru bot yang menggunakan intent versi $ TERBARU, Amazon Lex mengembalikan pengecualian HTTP 400 Bad Request. Sebelum dapat memublikasikan maksud versi bernomor, Anda harus mengarahkan intent ke versi bernomor dari jenis slot apa pun yang digunakan. Jika tidak, Anda akan mendapatkan pengecualian HTTP 400 Bad Request.

> **Note**
>
> Amazon Lex menerbitkan versi baru hanya jika versi terakhir yang diterbitkan berbeda dari$LATESTversi. Jika Anda mencoba untuk mempublikasikan$LATESTversi tanpa memodifikasinya, Amazon Lex tidak membuat atau mempublikasikan versi baru.

Memperbarui Sumber Daya Amazon Lex

Anda dapat memperbarui hanya$LATESTversi bot Amazon Lex, maksud, atau jenis slot. Versi yang dipublikasikan tidak dapat diubah. Anda dapat mempublikasikan versi baru kapan saja setelah memperbarui sumber daya di konsol atau dengan[CreateBotVersion](), yang[CreateIntentVersion]()atau[CreateSlotTypeVersion]()operasi.

Menghapus Sumber Daya atau Versi Amazon Lex

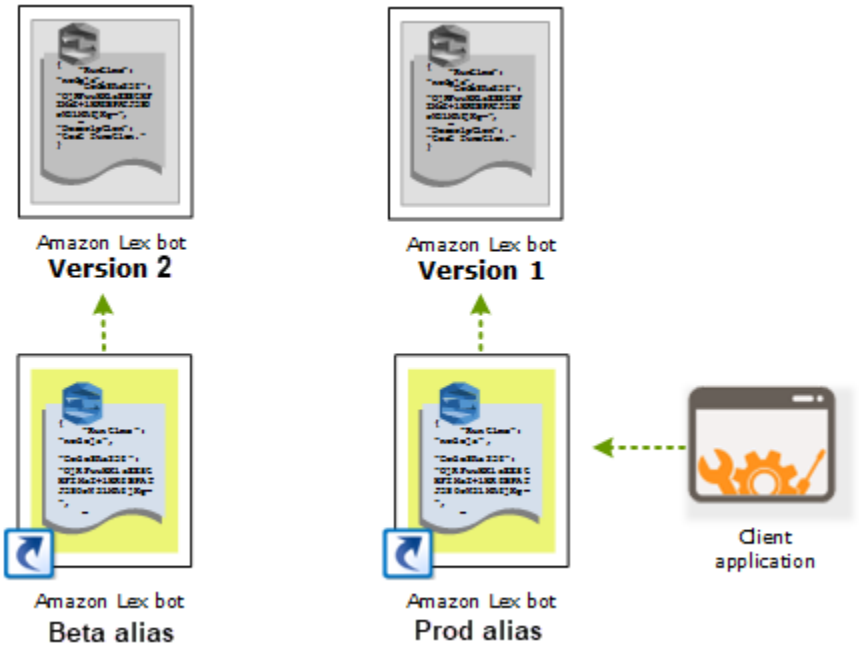
Amazon Lex mendukung menghapus sumber daya atau versi menggunakan konsol atau salah satu operasi API:

- [DeleteBot]()
- [DeleteBotVersion]()
- [DeleteBotAlias]()
- [DeleteBotChannelAssociation]()
- [DeleteIntent]()
- [DeleteIntentVersion]()
- [DeleteSlotType]()
- [DeleteSlotTypeVersion]()

Alias

Alias adalah penunjuk ke versi spesifik dari bot Amazon Lex. Gunakan alias untuk memungkinkan aplikasi klien menggunakan versi bot tertentu tanpa memerlukan aplikasi untuk melacak versi mana yang ada.

Contoh berikut menunjukkan dua versi bot Amazon Lex, 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.



Ketika Anda membuat versi kedua dari bot, Anda dapat memperbarui alias untuk menunjuk ke versi baru dari bot menggunakan konsol atauPutBotoperasi. 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.

> **Note**
>
> Meskipun Anda dapat menguji$LATESTversi bot di konsol, kami sarankan bahwa ketika Anda mengintegrasikan bot dengan aplikasi klien Anda, Anda pertama kali mempublikasikan 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 akan menunggu hingga batas waktu sesi semua sesi saat ini berakhir sebelum mulai menggunakan versi baru. Untuk informasi selengkapnya tentang batas waktu sesi, lihatthe section called “Mengatur Timeout Sesi”

Menggunakan Fungsi Lambda

Anda dapat membuatAWS Lambdafungsi untuk digunakan sebagai pengait kode untuk bot Amazon Lex Anda. Anda dapat mengidentifikasi fungsi Lambda untuk melakukan inisialisasi dan validasi, pemenuhan, atau keduanya dalam konfigurasi maksud Anda.

Kami rekomendasikan Anda menggunakan fungsi Lambda sebagai hook kode untuk bot Anda. Tanpa fungsi Lambda, bot Anda mengembalikan informasi maksud ke aplikasi klien untuk pemenuhan.

Topik

Lambda Fungsi Input Event dan Response Format

Bagian ini menjelaskan struktur data peristiwa yang disediakan Amazon Lex ke fungsi Lambda. Gunakan informasi ini untuk mengurai masukan dalam kode Lambda Anda. Ini juga menjelaskan format respon yang Amazon Lex mengharapkan fungsi Lambda Anda untuk kembali.

Topik

Format Peristiwa Masukan

Berikut ini menunjukkan format umum peristiwa Amazon Lex yang diteruskan ke fungsi Lambda. Gunakan informasi ini saat Anda menulis fungsi Lambda Anda.

> **Note**
>
> Format input dapat berubah tanpa perubahan yang sesuai dalam`messageVersion`. Kode Anda tidak boleh membuang kesalahan jika bidang baru ada.

```
{
  "currentIntent": {
    "name": "intent-name",
    "nluIntentConfidenceScore": score,
    "slots": {
      "slot name": "value",
      "slot name": "value"
    },
    "slotDetails": {
      "slot name": {
        "resolutions" : [
          { "value": "resolved value" },
          { "value": "resolved value" }
        ],
        "originalValue": "original text"
      },
      "slot name": {
        "resolutions" : [
          { "value": "resolved value" },
          { "value": "resolved value" }
        ],
        "originalValue": "original text"
      }
    },
    "confirmationStatus": "None, Confirmed, or Denied (intent confirmation, if
 configured)"
  },
  "alternativeIntents": [
    {
      "name": "intent-name",
      "nluIntentConfidenceScore": score,
      "slots": {
        "slot name": "value",
        "slot name": "value"
      },
      "slotDetails": {
        "slot name": {
          "resolutions" : [
            { "value": "resolved value" },
            { "value": "resolved value" }
          ],
          "originalValue": "original text"
        },
```

```
        "slot name": {
          "resolutions" : [
            { "value": "resolved value" },
            { "value": "resolved value" }
          ],
          "originalValue": "original text"
        }
      },
      "confirmationStatus": "None, Confirmed, or Denied (intent confirmation, if
configured)"
    }
 ],
 "bot": {
   "name": "bot name",
   "alias": "bot alias",
   "version": "bot version"
 },
 "userId": "User ID specified in the POST request to Amazon Lex.",
 "inputTranscript": "Text used to process the request",
 "invocationSource": "FulfillmentCodeHook or DialogCodeHook",
 "outputDialogMode": "Text or Voice, based on ContentType request header in runtime
API request",
 "messageVersion": "1.0",
 "sessionAttributes": {
    "key": "value",
    "key": "value"
 },
 "requestAttributes": {
    "key": "value",
    "key": "value"
 },
 "recentIntentSummaryView": [
   {
       "intentName": "Name",
       "checkpointLabel": Label,
       "slots": {
         "slot name": "value",
         "slot name": "value"
       },
       "confirmationStatus": "None, Confirmed, or Denied (intent confirmation, if
configured)",
       "dialogActionType": "ElicitIntent, ElicitSlot, ConfirmIntent, Delegate, or
Close",
       "fulfillmentState": "Fulfilled or Failed",
```

```
        "slotToElicit": "Next slot to elicit"
    }
  ],
   "sentimentResponse": {
      "sentimentLabel": "sentiment",
      "sentimentScore": "score"
   },
   "kendraResponse": {
       Complete query response from Amazon Kendra
   },
   "activeContexts": [
        {
            "timeToLive": {
                "timeToLiveInSeconds": seconds,
                "turnsToLive": turns
            },
            "name": "name",
            "parameters": {
                "key name": "value"
            }
        }
    ]
}
```

Perhatikan informasi tambahan berikut ini adalah kolom event:

- currentIntent— Menyediakan maksud`name`,`slots`,`slotDetails`dan`confirmationStatus`bidang.

  `nluIntentConfidenceScore`adalah keyakinan bahwa Amazon Lex memiliki bahwa maksud saat ini adalah salah satu yang paling cocok dengan maksud pengguna saat ini.

  `slots`adalah peta nama slot, dikonfigurasi untuk maksud, untuk slot nilai-nilai yang Amazon Lex telah diakui dalam percakapan pengguna. Nilai slot tetap null sampai pengguna memberikan nilai.

  Nilai slot dalam acara input mungkin tidak cocok dengan salah satu nilai yang dikonfigurasi untuk slot. Misalnya, jika pengguna merespon prompt “Mobil warna apa yang Anda inginkan?” dengan

"pizza," Amazon Lex akan kembali "pizza" sebagai nilai slot. Fungsi Anda harus memvalidasi nilai-nilai untuk memastikan bahwa mereka masuk akal dalam konteks.

`slotDetails`memberikan informasi tambahan tentang nilai slot. Parameter`resolutions`array berisi daftar nilai tambahan diakui untuk slot. Setiap slot dapat memiliki maksimum lima nilai.

Parameter`originalValue`bidang berisi nilai yang dimasukkan oleh pengguna untuk slot. Ketika jenis slot dikonfigurasi untuk mengembalikan nilai resolusi atas sebagai nilai slot,`originalValue`mungkin berbeda dari nilai di`slots`Bidang.

`confirmationStatus`memberikan respon pengguna untuk konfirmasi prompt, jika ada satu. Misalnya, jika Amazon Lex bertanya "Apakah Anda ingin memesan pizza keju besar? ," tergantung pada respon pengguna, nilai bidang ini dapat`Confirmed`atau`Denied`. Jika tidak, nilai bidang ini bersifat`None`.

Jika pengguna mengonfirmasi intent, Amazon Lex menetapkan bidang ini ke`Confirmed`. Jika pengguna menolak maksud, Amazon Lex menetapkan nilai ini ke`Denied`.

Dalam tanggapan konfirmasi, ucapan pengguna mungkin memberikan pembaruan slot. Misalnya, pengguna mungkin mengatakan "ya, ubah ukuran menjadi medium." Dalam hal ini, acara Lambda berikutnya memiliki nilai slot yang diperbarui,`PizzaSize`diatur ke`medium`. Amazon Lex menetapkan`confirmationStatus`kepada`None`, karena pengguna memodifikasi beberapa data slot, membutuhkan fungsi Lambda untuk melakukan validasi data pengguna.

- AlternativeIntent— Jika Anda mengaktifkan skor kepercayaan diri, Amazon Lex mengembalikan hingga empat maksud alternatif. Setiap maksud menyertakan skor yang menunjukkan tingkat kepercayaan yang dimiliki Amazon Lex bahwa maksud tersebut adalah maksud yang benar berdasarkan ucapan pengguna.

Isi dari maksud alternatif adalah sama dengan isi dari`currentIntent`Bidang. Untuk informasi selengkapnya, lihat Menggunakan Skor Keyakinan.

- bot— Informasi tentang bot yang memproses permintaan.
  - `name`— Nama bot yang memproses permintaan.
  - `alias`— Alias versi bot yang memproses permintaan.
  - `version`— Versi bot yang memproses permintaan.

- userId— Nilai ini disediakan oleh aplikasi klien. Amazon Lex meneruskannya ke fungsi Lambda.

- inputTranscript— Teks yang digunakan untuk memproses permintaan.

  Jika input adalah teks,`inputTranscript`bidang berisi teks yang masukan oleh pengguna.

  Jika input adalah aliran audio,`inputTranscript`bidang berisi teks yang diekstrak dari aliran audio. Ini adalah teks yang benar-benar diproses untuk mengenali maksud dan nilai slot.

- invocationSource— Untuk menunjukkan mengapa Amazon Lex menerapkan fungsi Lambda, ia menetapkan ini ke salah satu nilai berikut:
  - `DialogCodeHook`— Amazon Lex menetapkan nilai ini untuk mengarahkan fungsi Lambda untuk menginisialisasi fungsi dan memvalidasi input data pengguna.

    Ketika intent dikonfigurasi untuk memanggil fungsi Lambda sebagai hook inisialisasi dan kode validasi, Amazon Lex memanggil fungsi Lambda yang ditentukan pada setiap input pengguna (ucapan) setelah Amazon Lex memahami intent.

> **Note**
>
> Jika intent tidak jelas, Amazon Lex tidak dapat memanggil fungsi Lambda.

- `FulfillmentCodeHook`— Amazon Lex menetapkan nilai ini untuk mengarahkan fungsi Lambda untuk memenuhi maksud.

  Jika intent dikonfigurasi untuk memanggil fungsi Lambda sebagai hook kode pemenuhan, Amazon Lex menetapkan`invocationSource`untuk nilai ini hanya setelah memiliki semua data slot untuk memenuhi maksud.

  Dalam konfigurasi maksud Anda, Anda dapat memiliki dua fungsi Lambda terpisah untuk menginisialisasi dan memvalidasi data pengguna dan untuk memenuhi maksud. Anda juga dapat menggunakan satu fungsi Lambda untuk melakukan keduanya. Dalam hal ini, fungsi Lambda Anda dapat menggunakan`invocationSource`nilai untuk mengikuti jalur kode yang benar.

- OutputDialogMode— Untuk setiap input pengguna, klien mengirimkan permintaan ke Amazon Lex menggunakan salah satu operasi API runtime,PostContentatauPostText. Amazon Lex menggunakan parameter permintaan untuk menentukan apakah respons terhadap klien adalah teks atau suara, dan menetapkan bidang ini sesuai.

  Fungsi Lambda dapat menggunakan informasi ini untuk menghasilkan pesan yang sesuai. Misalnya, jika klien mengharapkan respons suara, fungsi Lambda Anda bisa mengembalikan Speech Synthesis Markup Language (SSIL) alih-alih teks.

- MessageVersion— Versi pesan yang mengidentifikasi format data peristiwa masuk ke fungsi Lambda dan format respon yang diharapkan dari fungsi Lambda.

> **Note**
>
> Anda mengonfigurasi nilai ini saat menentukan intent. Dalam implementasi saat ini, hanya pesan versi 1.0 yang didukung. Oleh karena itu, konsol mengasumsikan nilai default 1.0 dan tidak menampilkan versi pesan.

- sessionAttributes— Atribut sesi aplikasi-spesifik yang dikirim klien dalam permintaan. Jika Anda ingin Amazon Lex memasukkannya ke dalam respons terhadap klien, fungsi Lambda Anda harus mengirimkannya kembali ke Amazon Lex sebagai tanggapan. Untuk informasi selengkapnya, lihatMengatur Atribut Sesi

- requestAttributes— Permintaan-spesifik atribut bahwa klien mengirimkan dalam permintaan. Gunakan atribut request untuk meneruskan informasi yang tidak perlu bertahan untuk seluruh sesi. Jika tidak ada atribut request, nilai akan null. Untuk informasi selengkapnya, lihatMengatur Atribut Permintaan

- RecentIntentSummaryView— Informasi tentang keadaan maksud. Anda dapat melihat informasi tentang tiga intent terakhir yang digunakan. Anda dapat menggunakan informasi ini untuk menetapkan nilai dalam maksud atau kembali ke intent sebelumnya. Untuk informasi selengkapnya, lihat Mengelola Sesi Dengan Amazon Lex API.

- sentimentResponse— Hasil analisis sentimen Amazon Comprehend dari ucapan terakhir. Anda dapat menggunakan informasi ini untuk mengelola alur percakapan bot Anda tergantung pada sentimen yang diungkapkan oleh pengguna. Untuk informasi selengkapnya, lihat Analisis Sentimen.

- kendraResponse— Hasil kueri ke indeks Amazon Kendra. Hanya hadir dalam input ke hook kode pemenuhan dan hanya ketika intent memperluas`AMAZON.KendraSearchIntent`niat bawaan. Bidang ini berisi seluruh respon dari pencarian Amazon Kendra. Untuk informasi selengkapnya, lihat AMAZON.KendraSearchIntent.

- ActiveContext— Satu atau lebih konteks yang aktif selama pergantian percakapan dengan pengguna.
  - TimeTolive— Lamanya waktu atau jumlah putaran dalam percakapan dengan pengguna bahwa konteksnya tetap aktif.
  - nama— nama konteksnya.
  - parameterdaftar pasangan kunci/nilai berisi nama dan nilai slot dari intent yang mengaktifkan konteks.

  Untuk informasi selengkapnya, lihat Konteks Maksud Konteks Maksud Maksud.

Format Respons

Amazon Lex mengharapkan respons dari fungsi Lambda dalam format berikut:

```
{
  "sessionAttributes": {
    "key1": "value1",
    "key2": "value2"
    ...
  },
  "recentIntentSummaryView": [
    {
       "intentName": "Name",
       "checkpointLabel": "Label",
       "slots": {
         "slot name": "value",
         "slot name": "value"
        },
       "confirmationStatus": "None, Confirmed, or Denied (intent confirmation, if
configured)",
        "dialogActionType": "ElicitIntent, ElicitSlot, ConfirmIntent, Delegate, or
Close",
        "fulfillmentState": "Fulfilled or Failed",
        "slotToElicit": "Next slot to elicit"
    }
  ],
  "activeContexts": [
     {
       "timeToLive": {
          "timeToLiveInSeconds": seconds,
          "turnsToLive": turns
```

```
      },
      "name": "name",
      "parameters": {
        "key name": "value"
      }
    }
  ],
  "dialogAction": {
    "type": "ElicitIntent, ElicitSlot, ConfirmIntent, Delegate, or Close",
    Full structure based on the type field. See below for details.
  }
}
```

Tanggapan terdiri dari empat bidang. Parameter`sessionAttributes`,`recentIntentSummaryView`, dan`activeContexts`kolom bersifat opsional,`dialogAction`Bidang wajib diisi. Isi`dialogAction`bidang tergantung pada nilai`type`Bidang. Untuk detailnya, lihat DialogAction.

sessionAttributes

Tidak wajib. Jika Anda menyertakan`sessionAttributes`Bidang dapat kosong. Jika fungsi Lambda Anda tidak mengembalikan atribut sesi, yang terakhir diketahui`sessionAttributes`melewati fungsi API atau Lambda tetap. Untuk informasi selengkapnya, lihatPostContentdanPostTextoperasi.

```
  "sessionAttributes": {
     "key1": "value1",
     "key2": "value2"
  }
```

RecentIntentSummaryView

Tidak wajib. Jika disertakan, tetapkan nilai untuk satu atau lebih intent terbaru. Anda dapat menyertakan informasi hingga tiga maksud. Misalnya, Anda dapat menetapkan nilai untuk maksud sebelumnya berdasarkan informasi yang dikumpulkan oleh maksud saat ini. Informasi dalam ringkasan harus berlaku untuk maksud. Misalnya, nama maksud harus berupa maksud dalam bot. Jika Anda menyertakan nilai slot dalam tampilan ringkasan, slot harus ada dalam maksud. Jika Anda tidak menyertakan`recentIntentSummaryView`dalam tanggapan Anda, semua nilai untuk intent terbaru tetap tidak berubah. Untuk informasi selengkapnya, lihatPutSessionoperasi atauIntentSummarytipe data.

```
"recentIntentSummaryView": [
```

```
   {
      "intentName": "Name",
      "checkpointLabel": "Label",
      "slots": {
        "slot name": "value",
        "slot name": "value"
       },
      "confirmationStatus": "None, Confirmed, or Denied (intent confirmation, if
configured)",
       "dialogActionType": "ElicitIntent, ElicitSlot, ConfirmIntent, Delegate, or
Close",
       "fulfillmentState": "Fulfilled or Failed",
       "slotToElicit": "Next slot to elicit"
   }
 ]
```

ActiveContext

Tidak wajib. Jika disertakan, tetapkan nilai untuk satu atau lebih konteks. Misalnya, Anda dapat menyertakan konteks untuk membuat satu atau lebih maksud yang memiliki konteks tersebut sebagai masukan yang memenuhi syarat untuk dikenali pada giliran percakapan berikutnya.

Konteks aktif yang tidak termasuk dalam respons memiliki nilai waktu-ke-hidup mereka berkurang dan mungkin masih aktif pada permintaan berikutnya.

Jika Anda menentukan time-to-live 0 untuk konteks yang disertakan dalam acara input, itu akan menjadi tidak aktif pada permintaan berikutnya.

Untuk informasi selengkapnya, lihat Konteks Maksud Konteks Maksud Maksud.

DialogAction

Diperlukan. Parameter`dialogAction`bidang mengarahkan Amazon Lex ke kursus berikutnya tindakan, dan menjelaskan apa yang diharapkan dari pengguna setelah Amazon Lex mengembalikan respon kepada klien.

Parameter`type`bidang menunjukkan tindakan berikutnya. Ini juga menentukan bidang lain yang perlu disediakan oleh fungsi Lambda sebagai bagian dari`dialogAction`nilai.

- `Close`— Menginformasikan Amazon Lex tidak mengharapkan respon dari pengguna. Misalnya, “Pesanan pizza Anda telah ditempatkan” tidak memerlukan respons.

Bidang `fulfillmentState` wajib diisi. Amazon Lex menggunakan nilai ini untuk mengatur`dialogState`Bidang di bidangPostContentatauPostTextmenanggapi aplikasi klien. Parameter`message`dan`responseCard`Bidang bersifat opsional. Jika Anda tidak menentukan pesan, Amazon Lex menggunakan pesan selamat tinggal atau pesan tindak lanjut yang dikonfigurasi untuk intent.

```
"dialogAction": {
    "type": "Close",
    "fulfillmentState": "Fulfilled or Failed",
    "message": {
      "contentType": "PlainText or SSML or CustomPayload",
      "content": "Message to convey to the user. For example, Thanks, your pizza has
 been ordered."
    },
   "responseCard": {
      "version": integer-value,
      "contentType": "application/vnd.amazonaws.card.generic",
      "genericAttachments": [
          {
             "title":"card-title",
             "subTitle":"card-sub-title",
             "imageUrl":"URL of the image to be shown",
             "attachmentLinkUrl":"URL of the attachment to be associated with the
 card",
             "buttons":[
                 {
                    "text":"button-text",
                    "value":"Value sent to server on button click"
                 }
              ]
           }
       ]
     }
  }
```

- `ConfirmIntent`— Menginformasikan Amazon Lex bahwa pengguna diharapkan untuk memberikan jawaban ya atau tidak untuk mengkonfirmasi atau menolak maksud saat ini.

Anda harus menyertakan`intentName`dan`slots`bidang. Parameter`slots`bidang harus berisi entri untuk masing-masing slot yang diisi untuk maksud yang ditentukan. Anda tidak perlu menyertakan entri di`slots`lapangan untuk slot yang tidak diisi. Anda harus menyertakan`message`bidang jika intent`confirmationPrompt`Bidang adalah null. Isi`message`Bidang yang dikembalikan oleh fungsi Lambda diutamakan atas`confirmationPrompt`ditentukan dalam maksud. Parameter`responseCard`Bidang bersifat opsional.

```
"dialogAction": {
    "type": "ConfirmIntent",
    "message": {
      "contentType": "PlainText or SSML or CustomPayload",
      "content": "Message to convey to the user. For example, Are you sure you want a
 large pizza?"
    },
   "intentName": "intent-name",
   "slots": {
      "slot-name": "value",
      "slot-name": "value",
      "slot-name": "value"
   },
   "responseCard": {
      "version": integer-value,
      "contentType": "application/vnd.amazonaws.card.generic",
      "genericAttachments": [
          {
             "title":"card-title",
             "subTitle":"card-sub-title",
             "imageUrl":"URL of the image to be shown",
             "attachmentLinkUrl":"URL of the attachment to be associated with the
 card",
             "buttons":[
                 {
                    "text":"button-text",
                    "value":"Value sent to server on button click"
                 }
              ]
           }
       ]
     }
  }
```

- `Delegate`— Mengarahkan Amazon Lex untuk memilih tindakan berikutnya berdasarkan konfigurasi bot. Jika respon tidak termasuk atribut sesi apa pun Amazon Lex mempertahankan atribut yang ada. Jika Anda ingin nilai slot menjadi nol, Anda tidak perlu menyertakan bidang slot dalam permintaan. Anda akan mendapatkan`DependencyFailedException`pengecualian jika fungsi pemenuhan Anda mengembalikan`Delegate`tindakan dialog tanpa menghapus slot apapun.

  Parameter`kendraQueryRequestPayload`dan`kendraQueryFilterString`field bersifat opsional dan hanya digunakan ketika intent berasal dari`AMAZON.KendraSearchIntent`maksud bawaan. Untuk informasi selengkapnya, lihatAMAZON.KendraSearchIntent.

  ```
  "dialogAction": {
   "type": "Delegate",
   "slots": {
      "slot-name": "value",
      "slot-name": "value",
      "slot-name": "value"
   },
   "kendraQueryRequestPayload": "Amazon Kendra query",
   "kendraQueryFilterString": "Amazon Kendra attribute filters"
  }
  ```

- `ElicitIntent`— Menginformasikan Amazon Lex bahwa pengguna diharapkan untuk menanggapi dengan ucapan yang mencakup maksud. Misalnya, “Saya ingin pizza besar,” yang menunjukkan`OrderPizzaIntent`. Ucapan “besar,” di sisi lain, tidak cukup bagi Amazon Lex untuk menyimpulkan maksud pengguna.

  Parameter`message`dan`responseCard`Bidang bersifat opsional. Jika Anda tidak memberikan pesan, Amazon Lex menggunakan salah satu perintah klarifikasi bot. Jika tidak ada prompt klarifikasi yang ditentukan, Amazon Lex mengembalikan pengecualian 400 Bad Request.

  ```
  {
    "dialogAction": {
      "type": "ElicitIntent",
      "message": {
        "contentType": "PlainText or SSML or CustomPayload",
        "content": "Message to convey to the user. For example, What can I help you
   with?"
      },
      "responseCard": {
  ```

```
      "version": integer-value,
      "contentType": "application/vnd.amazonaws.card.generic",
      "genericAttachments": [
          {
             "title":"card-title",
             "subTitle":"card-sub-title",
             "imageUrl":"URL of the image to be shown",
             "attachmentLinkUrl":"URL of the attachment to be associated with the
card",
             "buttons":[
                 {
                    "text":"button-text",
                    "value":"Value sent to server on button click"
                 }
              ]
           }
       ]
    }
}
```

- `ElicitSlot`— Menginformasikan Amazon Lex bahwa pengguna diharapkan untuk memberikan nilai slot dalam respon.

  Parameter`intentName`,`slotToElicit`, dan`slots`bidang yang diperlukan. Parameter`message`dan`responseCard`Bidang bersifat opsional. Jika Anda tidak menentukan pesan, Amazon Lex menggunakan salah satu perintah elisitasi slot yang dikonfigurasi untuk slot.

```
 "dialogAction": {
   "type": "ElicitSlot",
   "message": {
     "contentType": "PlainText or SSML or CustomPayload",
     "content": "Message to convey to the user. For example, What size pizza would
you like?"
   },
  "intentName": "intent-name",
  "slots": {
     "slot-name": "value",
     "slot-name": "value",
     "slot-name": "value"
  },
```

```
  "slotToElicit" : "slot-name",
  "responseCard": {
     "version": integer-value,
     "contentType": "application/vnd.amazonaws.card.generic",
     "genericAttachments": [
         {
            "title":"card-title",
            "subTitle":"card-sub-title",
            "imageUrl":"URL of the image to be shown",
            "attachmentLinkUrl":"URL of the attachment to be associated with the
card",
            "buttons":[
                {
                   "text":"button-text",
                   "value":"Value sent to server on button click"
                }
             ]
          }
       ]
    }
 }
```

Amazon Lex danAWS LambdaCetak Biru

Konsol Amazon Lex menyediakan contoh bot (disebut cetak biru bot) yang telah dikonfigurasi sebelumnya sehingga Anda dapat dengan cepat membuat dan menguji bot di konsol. Untuk masing-masing cetak biru bot ini, cetak biru fungsi Lambda juga disediakan. Cetak biru ini menyediakan kode contoh yang bekerja dengan bot yang sesuai. Anda dapat menggunakan cetak biru ini untuk dengan cepat membuat bot yang dikonfigurasi dengan fungsi Lambda sebagai hook kode, dan menguji pengaturan end-to-end tanpa harus menulis kode.

Anda dapat menggunakan cetak biru bot Amazon Lex dan yang sesuaiAWS Lambdacetak biru fungsi sebagai kait kode untuk bot:

- Cetak biru Amazon Lex —`OrderFlowers`
  - AWS LambdaCetak Biru —`lex-order-flowers-python`
- Cetak biru Amazon Lex —`ScheduleAppointment`
  - AWS LambdaCetak Biru —`lex-make-appointment-python`
- Cetak biru Amazon Lex —`BookTrip`

- AWS LambdaCetak Biru —`lex-book-trip-python`

Untuk membuat bot menggunakan cetak biru dan mengkonfigurasinya untuk menggunakan fungsi Lambda sebagai hook kode, lihatLatihan 1: Membuat Bot Amazon Lex Menggunakan Cetak Biru (Konsol). Untuk contoh menggunakan cetak biru lainnya, lihatContoh Tambahan: Membuat Amazon Lex Bots.

Memperbarui Cetak Biru untuk Lokal Tertentu

Jika Anda menggunakan cetak biru di lokal selain bahasa Inggris (AS) (id), Anda perlu memperbarui nama maksud apa pun untuk menyertakan lokal. Misalnya, jika Anda menggunakan`OrderFlowers`Cetak biru, Anda perlu melakukan hal berikut.

- Menemukan`dispatch`fungsi dekat akhir kode fungsi Lambda.
- Di`dispatch`fungsi, perbarui nama maksud untuk menyertakan lokal yang Anda gunakan. Misalnya, jika Anda menggunakan lokal bahasa Inggris (Australia) (en-AU), ubah baris:

  ```
  if intent_name == 'OrderFlowers':
  ```

  kepada

  ```
  if intent_name == 'OrderFlowers_enAU':
  ```

Cetak biru lain menggunakan nama maksud lain, mereka harus diperbarui seperti di atas sebelum Anda menggunakannya.

Menerapkan Bot Amazon Lex

Bagian ini memberikan contoh penerapan bot Amazon Lex pada berbagai platform perpesanan dan aplikasi seluler.

Topik

Menyebarkan Amazon Lex Bot pada Platform Pesan

Bagian ini menjelaskan cara menyebarkan bot Amazon Lex di platform pesan Facebook, Slack, dan Twilio.

> **Note**
>
> Saat menyimpan konfigurasi Facebook, Slack, atau Twilio, Amazon Lex menggunakanAWS Key Management Servicepelanggan berhasil kunci untuk mengenkripsi informasi. Pertama kali Anda membuat saluran ke salah satu platform pesan ini, Amazon Lex membuat kunci terkelola pelanggan default (`aws/lex`). Atau, Anda dapat membuat kunci terkelola pelanggan denganAWS KMS. Ini akan memberikan Anda fleksibilitas yang lebih baik, termasuk kemampuan untuk membuat, memutar, dan menonaktifkan tombol. Anda juga dapat menentukan kontrol akses dan audit kunci enkripsi yang digunakan untuk melindungi data Anda. Untuk informasi selengkapnya, lihat Panduan Developer AWS Key Management Service.

Ketika platform perpesanan mengirimkan permintaan ke Amazon Lex, platform tersebut menyertakan informasi spesifik platform sebagai atribut permintaan ke fungsi Lambda Anda. Gunakan atribut ini untuk menyesuaikan cara bot Anda berperilaku. Untuk informasi selengkapnya, lihat Mengatur Atribut Permintaan.

Semua atribut mengambil namespace,`x-amz-lex:`, sebagai awalan. Misalnya,`user-id`atribut disebut`x-amz-lex:user-id`. Ada atribut umum yang dikirim oleh semua platform pesan selain atribut yang spesifik untuk platform tertentu. Tabel berikut mencantumkan atribut permintaan yang dikirim platform pesan ke fungsi Lambda bot Anda.

Atribut Permintaan Umum

Atribut	Deskripsi
`channel-id`	Saluran endpoint identifier dari Amazon Lex.
`channel-name`	Nama saluran dari Amazon Lex.
`channel-type`	Salah satu nilai berikut: • `Facebook` • `Kik` • `Slack` • `Twilio-SMS`
`webhook-endpoint-url`	Titik akhir Amazon Lex untuk saluran.

Atribut Permintaan Facebook

Atribut	Deskripsi
`user-id`	Identifier Facebook pengirim. Lihathttps://developers.facebook.com/docs/messenger-platform/webhook-reference/message-received.
`facebook-page-id`	Pengenal halaman Facebook penerima. Lihathttps://developers.facebook.com/docs/messenger-platform/webhook-reference/message-received.

Kik Permintaan Atribut

Atribut	Deskripsi
`kik-chat-id`	Pengenal untuk percakapan yang dilibatkan oleh bot Anda. Untuk informasi selengkapnya, lihathttps://dev.kik.com/#/docs/messaging#message-formats.

Atribut	Deskripsi
`kik-chat-type`	Jenis percakapan yang berasal dari pesan. Untuk informasi selengkapnya, lihathttps://dev.kik.com/#/docs/messaging#message-formats.
`kik-message-id`	Sebuah UUID mengidentifikasi pesan. Untuk informasi selengkapnya, lihathttps://dev.kik.com/#/docs/messaging#message-formats.
`kik-message-type`	Jenis pesan. Untuk informasi selengkapnya, lihathttps://dev.kik.com/#/docs/messaging#message-types.

Atribut Permintaan Twilio

Atribut	Deskripsi
`user-id`	Nomor telepon pengirim (“Dari”). Lihathttps://www.twilio.com/docs/api/rest/message.
`twilio-target-phone-number`	Nomor telepon penerima (“Untuk”). Lihathttps://www.twilio.com/docs/api/rest/message.

Atribut Permintaan Slack

Atribut	Deskripsi
`user-id`	Pengenal pengguna Slack. Lihathttps://api.slack.com/types/user.
`slack-team-id`	Pengenal tim yang mengirim pesan. Lihathttps://api.slack.com/methods/team.info.
`slack-bot-token`	Token pengembang yang memberikan akses bot ke API Slack. Lihathttps://api.slack.com/docs/token-types.

Mengintegrasikan Amazon Lex Bot dengan Facebook Messenger

Latihan ini menunjukkan cara mengintegrasikan Facebook Messenger dengan bot Amazon Lex Anda. Anda harus melakukan langkah-langkah berikut ini:

1. Buat bot Amazon Lex
2. Buat aplikasi Facebook
3. Integrasikan Facebook Messenger dengan bot Amazon Lex Anda
4. Validasi integrasi

Topik

Langkah 1: Buat Bot Amazon Lex Lex Lex

Jika Anda belum memiliki bot Amazon Lex, buat satu bot bot. Dalam topik ini, kami berasumsi bahwa Anda menggunakan bot yang Anda buat di Getting Started Exercise 1. Namun demikian, Anda dapat menggunakan hal berikut yang disediakan dalam panduan ini. Untuk Memulai Latihan 1, lihatLatihan 1: Membuat Bot Amazon Lex Menggunakan Cetak Biru (Konsol).

1. Buat bot Amazon Lex. Untuk petunjuk, lihat Latihan 1: Membuat Bot Amazon Lex Menggunakan Cetak Biru (Konsol).
2. Menyebarkan bot dan membuat alias. Untuk petunjuk, lihat Latihan 3: Publikasikan Versi dan Buat Alias.

Langkah 2: Buat Aplikasi Facebook Facebook

Di portal pengembang Facebook, buat aplikasi Facebook dan halaman Facebook. Untuk petunjuknya, lihat Mulai Cepat di dokumentasi platform Facebook Messenger. Tulis hal berikut:

- Rahasia Aplikasi untuk Aplikasi Facebook
- Token Akses Halaman untuk halaman Facebook

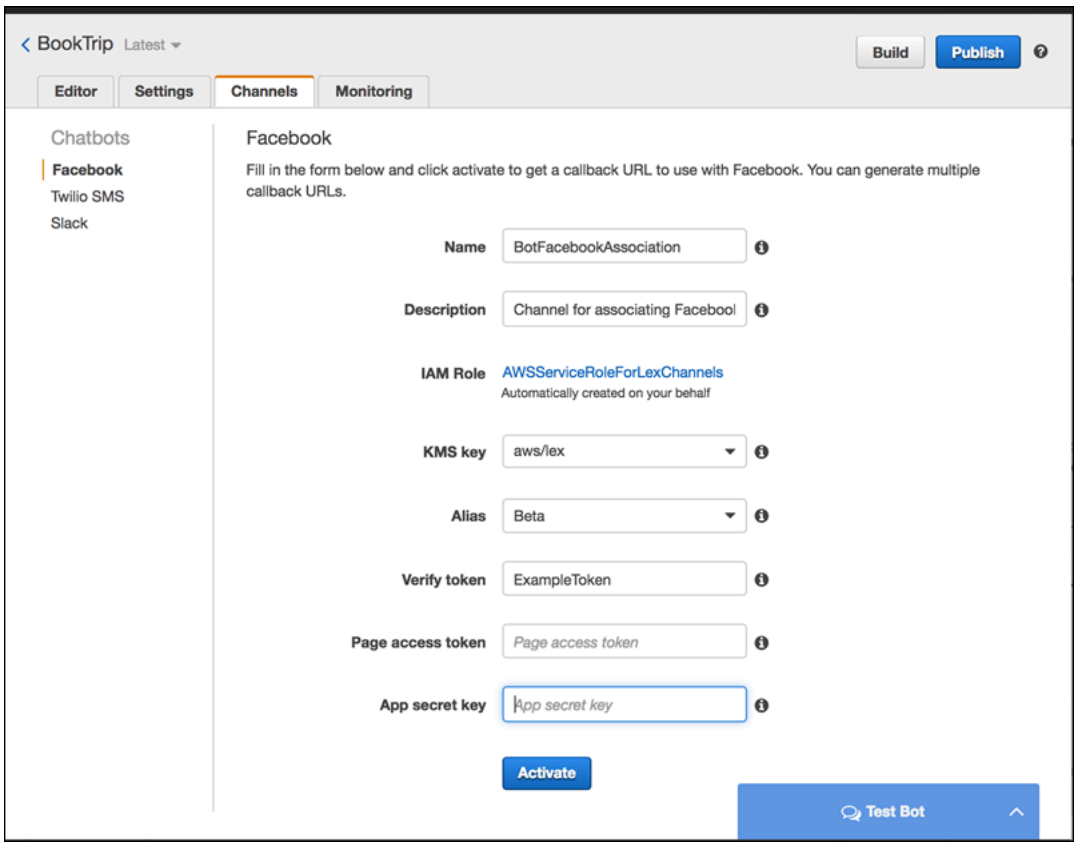
Langkah 3: Integrasikan Facebook Messenger dengan Amazon Lex Bot

Di bagian ini, Anda mengintegrasikan Facebook Messenger dengan bot Amazon Lex Anda.

Setelah menyelesaikan langkah ini, konsol menyediakan URL callback. Tuliskan URL ini.

Untuk mengintegrasikan Facebook Messenger dengan bot Anda

1. a. Masuk keAWS Management Console dan buka konsol Amazon Lex Lex di https://console.aws.amazon.com/lex/.
   b. Pilih bot Amazon Lex Anda.
   c. Pilih Saluran.
   d. Pilih Facebook di bawah Chatbots. Konsol menampilkan halaman integrasi Facebook.
   e. Di halaman integrasi Facebook, lakukan hal berikut:
      - Ketik nama berikut:`BotFacebookAssociation`.
      - Untuk kunci KMS, pilih aws/lex.
      - Untuk Alias, pilih alias bot.
      - Untuk Verifikasi token, ketik token. Ini bisa berupa string yang Anda pilih (misalnya,`ExampleToken`). Anda menggunakan token ini nanti di portal pengembang Facebook saat Anda mengatur webhook.
      - Untuk token akses Halaman, ketik token yang Anda peroleh dari Facebook di Langkah 2.
      - Untuk kunci rahasia App, ketik kunci yang Anda peroleh dari Facebook di Langkah 2.



f. Pilih Aktifkan.

Konsol membuat asosiasi saluran bot dan mengembalikan URL callback. Tuliskan URL ini.

2. Di portal pengembang Facebook, pilih aplikasi Anda.
3. Pilih produk Messenger, dan pilih Pengaturan webhook di bagian Webhooks pada halaman.

Untuk petunjuknya, lihat Mulai Cepat di dokumentasi platform Facebook Messenger.

4. Pada halaman webhook wizard langganan, lakukan hal berikut:

- Untuk URL Callback, ketik URL callback yang disediakan di konsol Amazon Lex sebelumnya dalam prosedur.
- Untuk Verifikasi Token, ketik token yang sama dengan yang Anda gunakan di Amazon Lex.
- Pilih Bidang Langganan (pesan, messaging_postbacks, dan messaging_optins).
- Pilih Verifikasi dan Simpan. Ini memulai jabat tangan antara Facebook dan Amazon Lex.

5. Aktifkan integrasi Webhooks. Pilih halaman yang Anda buat, lalu pilih berlangganan.

> **Note**
>
> Jika Anda memperbarui atau membuat ulang webhook, berhenti berlangganan dan kemudian berlangganan kembali ke halaman.

Langkah 4: Uji integrasi

Anda sekarang dapat memulai percakapan dari Facebook Messenger dengan bot Amazon Lex Anda.

1. Buka halaman Facebook Anda, lalu pilih Pesan.
2. Di jendela Messenger, gunakan ucapan tes yang sama yang disediakan di[Langkah 1: Buat Bot Amazon Lex (Konsol)](#).

Mengintegrasikan Amazon Lex Bot dengan Kik

Latihan ini memberikan instruksi untuk mengintegrasikan bot Amazon Lex dengan aplikasi perpesanan Kik. Anda harus melakukan langkah-langkah berikut ini:

1. Buat bot Amazon Lex.
2. Buat bot Kik menggunakan aplikasi dan situs web Kik.
3. Integrasikan bot Amazon Lex Anda dengan bot Kik menggunakan konsol Amazon Lex.
4. Terlibat dalam percakapan dengan bot Amazon Lex Anda menggunakan Kik untuk menguji hubungan antara bot Amazon Lex dan Kik Anda.

Topik

Langkah 1: Buat Amazon Lex Bot

Jika Anda belum memiliki bot Amazon Lex, buat dan terapkan bot. Dalam topik ini, kami berasumsi bahwa Anda menggunakan bot yang Anda buat di Getting Started Exercise 1. Namun demikian

demikian, Anda dapat menggunakan hal apa pun yang disediakan dalam panduan ini. Untuk Memulai Latihan 1, lihatLatihan 1: Membuat Bot Amazon Lex Menggunakan Cetak Biru (Konsol)

1. Buat bot Amazon Lex. Untuk petunjuk, lihat Latihan 1: Membuat Bot Amazon Lex Menggunakan Cetak Biru (Konsol).
2. Menyebarkan bot dan membuat alias. Untuk petunjuk, lihat Latihan 3: Publikasikan Versi dan Buat Alias.

Langkah Selanjutnya

Langkah 2: Buat Bot Kik

Langkah 2: Buat Bot Kik

Pada langkah ini Anda menggunakan antarmuka pengguna Kik untuk membuat bot Kik. Anda menggunakan informasi yang dihasilkan saat membuat bot untuk menghubungkannya ke bot Amazon Lex Anda.

1. Jika belum memilikinya, unduh dan instal aplikasi Kik dan daftar untuk akun Kik. Jika Anda memiliki akun, masuk.
2. Buka situs web Kik di https://dev.kik.com/. Biarkan jendela browser terbuka.
3. Di aplikasi Kik, pilih ikon roda gigi untuk membuka pengaturan, lalu pilih Kode Kik Anda.
4. Pindai kode Kik di situs web Kik untuk membuka chatbot Botsworth. Pilih Ya untuk membuka Dasbor Bot.
5. Di aplikasi Kik, pilih Buat Bot. Ikuti petunjuk untuk membuat bot Kik Anda.
6. Setelah bot dibuat, pilih Konfigurasi di browser Anda. Pastikan bot baru Anda dipilih.
7. Perhatikan nama bot dan kunci API untuk bagian selanjutnya.

Langkah Selanjutnya

Langkah 3: Integrasikan Kik Bot dengan Amazon Lex Bot

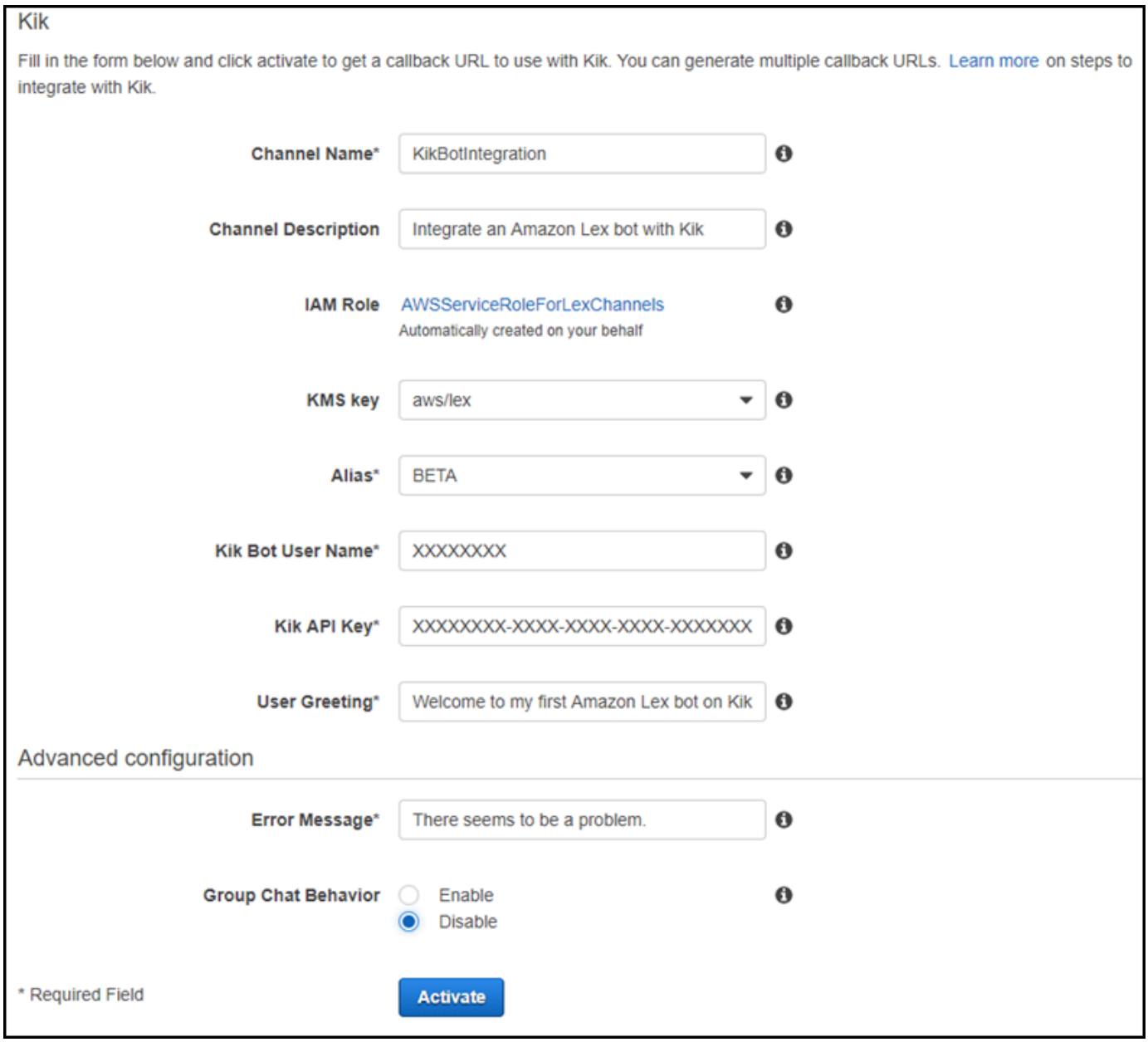
Langkah 3: Integrasikan Kik Bot dengan Amazon Lex Bot

Sekarang setelah Anda membuat bot Amazon Lex dan bot Kik, Anda siap untuk membuat hubungan saluran di antara mereka di Amazon Lex. Saat asosiasi diaktifkan, Amazon Lex secara otomatis menyiapkan URL callback dengan Kik.

1. Masuk ke AWS Management Console, dan buka konsol Amazon Lex di https://console.aws.amazon.com/lex/.
2. Pilih bot Amazon Lex yang Anda buat di Langkah 1.
3. Pilih tab Saluran.
4. Di bagian Saluran, pilih KIK.
5. Di halaman Kik, berikan hal berikut:

   - Ketik nama. Sebagai contoh, `BotKikIntegration`.
   - Ketik deskripsi.
   - Pilih “aws/lex” dari drop-down tombol KMS.
   - Untuk Alias, pilih alias dari menu drop-down.
   - Untuk nama pengguna bot Kik, ketik nama yang Anda berikan bot pada Kik.
   - Untuk kunci API Kik, ketik kunci API yang ditetapkan ke bot di Kik.
   - Untuk ucapan Pengguna, ketik sapaan yang Anda ingin bot Anda kirim saat pertama kali pengguna mengobrol dengannya.
   - Untuk Pesan kesalahan, masukkan pesan kesalahan yang ditampilkan kepada pengguna ketika bagian dari percakapan tidak dipahami.
   - Untuk perilaku obrolan grup, pilih salah satu opsi:
     - Aktifkan - Memungkinkan seluruh grup obrolan untuk berinteraksi dengan bot Anda dalam satu percakapan.
     - Nonaktifkan - Membatasi percakapan ke satu pengguna di grup obrolan.

   - Pilih Aktifkan untuk membuat asosiasi dan menautkannya ke bot Kik.



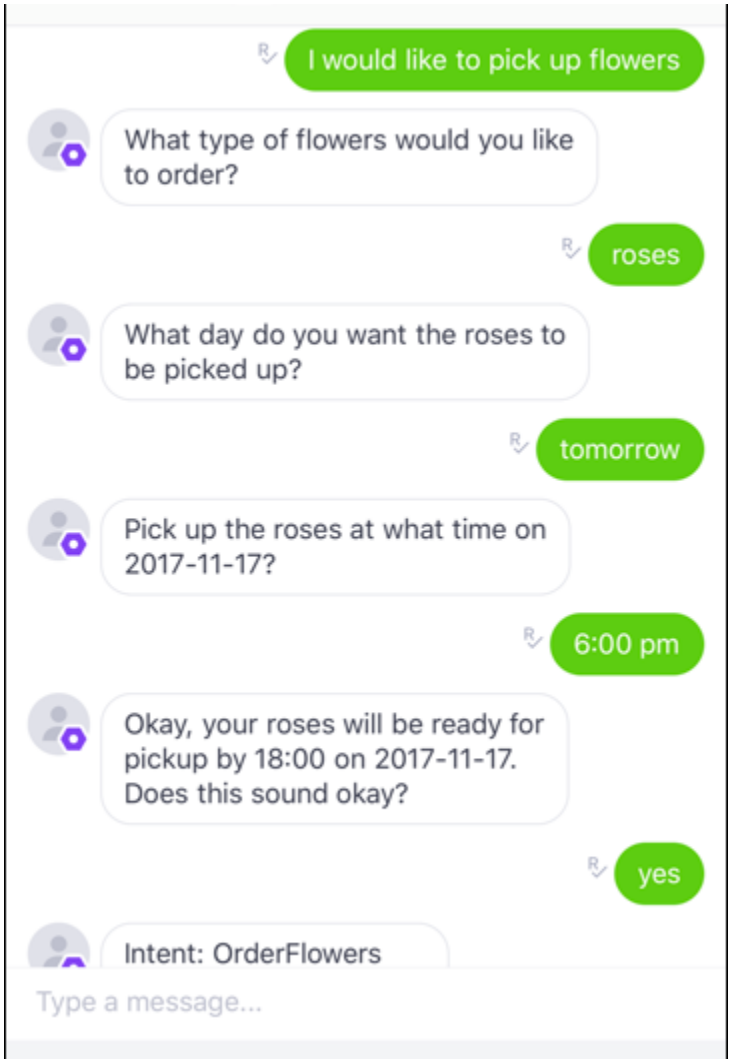
Langkah Selanjutnya

[Langkah 4: Uji Integrasi](#)

Langkah 4: Uji Integrasi

Sekarang setelah Anda membuat hubungan antara bot Amazon Lex dan Kik, Anda dapat menggunakan aplikasi Kik untuk menguji asosiasi.

1. Mulai aplikasi Kik dan masuk. Pilih bot yang Anda buat.
2. Anda dapat menguji bot dengan yang berikut:



Saat Anda memasukkan setiap frasa, bot Amazon Lex Anda akan merespons melalui Kik dengan prompt yang Anda buat untuk setiap slot.

Mengintegrasikan Amazon Lex Bot dengan Slack

Latihan ini memberikan instruksi untuk mengintegrasikan bot Amazon Lex dengan aplikasi perpesanan Slack. Anda harus melakukan langkah-langkah berikut ini:

1. Buat bot Amazon Lex.
2. Buat aplikasi perpesanan Slack.
3. Integrasikan aplikasi Slack dengan bot Anda Amazon Lex.

4. Uji integrasi dengan terlibat dalam percakapan dengan bot Amazon Lex Anda. Anda mengirim pesan dengan aplikasi Slack dan menguji di jendela browser.

Topik

Langkah 1: Buat Amazon Lex Bot

Jika Anda belum memiliki bot Amazon Lex, buat dan terapkan satu bot. Dalam topik ini, kami berasumsi bahwa Anda menggunakan bot yang Anda buat di Getting Started Exercise 1. Namun demikian, Anda dapat menggunakan hal apa pun yang disediakan dalam panduan ini. Untuk Memulai Latihan 1, lihatLatihan 1: Membuat Bot Amazon Lex Menggunakan Cetak Biru (Konsol)

1. Buat bot Amazon Lex. Untuk petunjuk, lihat Latihan 1: Membuat Bot Amazon Lex Menggunakan Cetak Biru (Konsol).
2. Menyebarkan bot dan membuat alias. Untuk petunjuk, lihat Latihan 3: Publikasikan Versi dan Buat Alias.

Langkah Selanjutnya

Langkah 2: Mendaftar untuk Slack dan Buat Tim Slack

Langkah 2: Mendaftar untuk Slack dan Buat Tim Slack

Daftar untuk akun Slack dan buat tim Slack. Untuk petunjuknya, lihat Menggunakan Slack. Di bagian selanjutnya, Anda membuat aplikasi Slack, yang dapat diinstal oleh tim Slack mana pun.

Langkah Selanjutnya

Langkah 3: Buat aplikasi Slack

Langkah 3: Buat aplikasi Slack

Pada bagian ini, lakukan hal berikut:

1. Membuat aplikasi Slack pada Slack API Console
2. Konfigurasikan aplikasi untuk menambahkan pesan interaktif ke bot Anda:

Di akhir bagian ini, Anda mendapatkan kredensyal aplikasi (Client Id, Client Secret, dan Verification Token). Pada bagian berikutnya, Anda menggunakan informasi ini untuk mengonfigurasikan asosiasi saluran bot di konsol Amazon Lex.

1. Masuk ke Slack API Console di http://api.slack.com.
2. Buat 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 tombol 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 Anda menambahkan asosiasi saluran bot di konsol Amazon Lex.

     - Pilih Simpan Perubahan.
4. Di menu sebelah kiri, di Pengaturan, pilih Informasi Dasar. Catat kredensyal aplikasi berikut:

   - ID klien
   - Rahasia Klien
   - Token Verifikasi

Langkah Selanjutnya

Langkah 4: Integrasikan Aplikasi Slack dengan Amazon Lex Bot

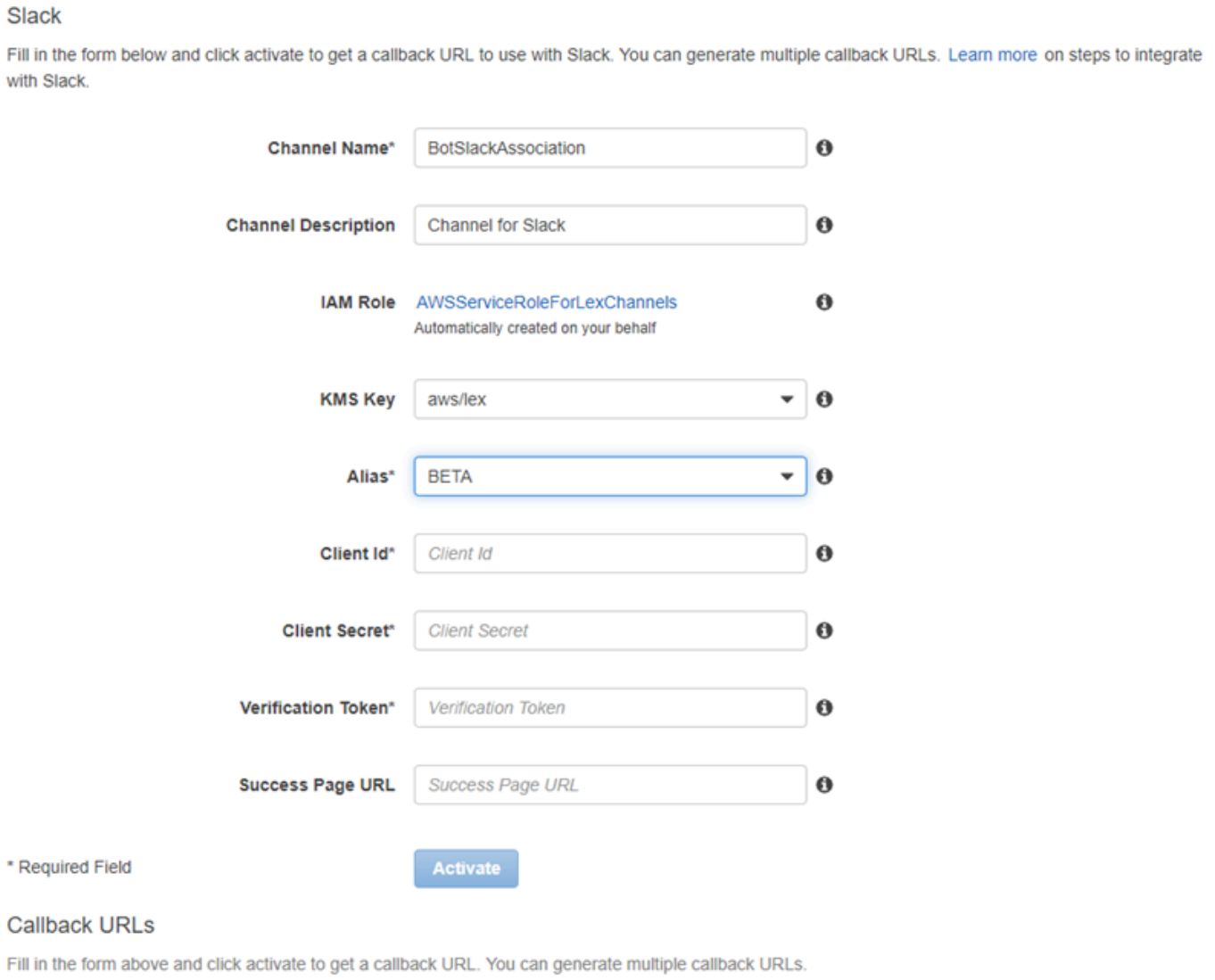
Langkah 4: Integrasikan Aplikasi Slack dengan Amazon Lex Bot

Sekarang setelah Anda memiliki kredensyal aplikasi Slack, Anda dapat mengintegrasikan aplikasi dengan bot Amazon Lex Anda. Untuk mengaitkan aplikasi Slack dengan bot Anda, tambahkan asosiasi saluran bot di Amazon Lex.

Di konsol Amazon Lex, aktifkan asosiasi saluran bot untuk mengaitkan bot dengan aplikasi Slack Anda. Saat asosiasi saluran bot diaktifkan, Amazon Lex mengembalikan dua URL (URL Postback dan URL OAuth). Rekam URL ini karena Anda membutuhkannya nanti.

Untuk mengintegrasikan aplikasi Slack dengan bot Amazon Lex Anda

1. Masuk ke AWS Management Console, dan buka konsol Amazon Lex di https://console.aws.amazon.com/lex/.
2. Pilih bot Amazon Lex yang Anda buat di Langkah 1.
3. Pilih tab Saluran.
4. Pada menu sebelah kiri, pilih Slack.
5. Pada halaman Slack, berikan yang berikut ini:
   - Ketik nama. Sebagai contoh, `BotSlackIntegration`.
   - Pilih “aws/lex” dari drop-down tombol KMS.
   - Untuk Alias, pilih alias bot.
   - Ketik Id Klien, Rahasia Klien, dan Token Verifikasi, yang Anda catat pada langkah sebelumnya. Ini adalah kredensyal dari aplikasi Slack.



6. Pilih Aktifkan.

   Konsol membuat asosiasi saluran bot dan mengembalikan dua URL (URL Postback dan URL OAuth). Rekam mereka. Di bagian selanjutnya, Anda memperbarui konfigurasi aplikasi Slack Anda untuk menggunakan titik akhir ini sebagai berikut:

   - URL Postback adalah titik akhir bot Amazon Lex yang mendengarkan peristiwa Slack. Anda menggunakan URL ini:
     - Sebagai URL permintaan di fitur Event Subscriptions dari aplikasi Slack.
     - Untuk mengganti nilai placeholder untuk URL permintaan di fitur Pesan Interaktif aplikasi Slack.

- URL OAuth adalah titik akhir bot Amazon Lex Anda untuk jabat tangan OAuth dengan Slack.

Langkah Selanjutnya

Langkah 5: Selesaikan Slack

Langkah 5: Selesaikan Slack

Di bagian ini, gunakan konsol API Slack untuk menyelesaikan integrasi aplikasi Slack.

1. Masuk ke konsol Slack API di http://api.slack.com. Pilih aplikasi yang akan dibuatkanLangkah 3: Buat aplikasi Slack.
2. Perbarui fitur OAuth & Izin sebagai berikut:

   a. Di menu sebelah kiri, pilih OAuth & Permissions.

   b. Di bagian Redirect URL, tambahkan URL OAuth yang disediakan Amazon Lex pada langkah sebelumnya. Pilih Tambahkan URL Pengalihan baru, lalu pilih Simpan URL.

   c. Di bagian Bot Token Scopes, tambahkan dua izin dengan tombol Tambahkan Lingkup OAuth. Saring daftar dengan teks berikut:

      - **`chat:write`**
      - **`team:read`**
3. Perbarui fitur Interaktivitas & Pintasan dengan memperbarui nilai URL Permintaan ke URL Postback yang disediakan Amazon Lex pada langkah sebelumnya. Masukkan URL postback yang Anda simpan di langkah 4, lalu pilih Simpan Perubahan.
4. Berlangganan fitur Langganan Acara sebagai berikut:

   - Aktifkan acara dengan memilih opsi On.
   - Tetapkan nilai URL Permintaan ke URL Postback yang disediakan Amazon Lex pada langkah sebelumnya.
   - Di bagian Berlangganan Acara Bot, berlangganan 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 dan pesan Slash dari tab pesan.

Langkah Selanjutnya

Langkah 6: Uji integrasi

Langkah 6: Uji integrasi

Sekarang gunakan jendela browser untuk menguji integrasi Slack dengan bot Amazon Lex Anda.

1. Pilih Kelola Distribusi di bawah Pengaturan. Pilih Add to Slack untuk menginstal aplikasi. Otorisasi bot untuk menanggapi pesan.
2. Anda diarahkan ke tim Slack Anda. Di menu sebelah kiri, di bagian Direct Message, pilih bot Anda. Jika bot Anda tidak melihat, pilih ikon plus (+) di sebelah Direct Message untuk mencarinya.
3. Terlibat dalam obrolan dengan aplikasi Slack Anda, yang ditautkan ke bot Amazon Lex. Bot Anda sekarang merespons pesan.

   Jika Anda membuat bot menggunakan Latihan Memulai 1, Anda dapat menggunakan contoh percakapan yang disediakan dalam latihan itu. Untuk informasi selengkapnya, lihat Langkah 4: Tambahkan Fungsi Lambda sebagai Kode Hook (Konsol).

Mengintegrasikan Amazon Lex Bot dengan Twilio Programmable SMS

Latihan ini memberikan instruksi untuk mengintegrasikan bot Amazon Lex dengan layanan pesan sederhana Twilio (SMS). Anda harus melakukan langkah-langkah berikut ini:

1. Buat bot Amazon Lex
2. Integrasikan Twilio SMS yang dapat diprogram dengan bot Anda Amazon Lex
3. Terlibat dalam interaksi dengan bot Amazon Lex dengan menguji pengaturan menggunakan layanan SMS di ponsel Anda
4. Uji integrasi

Topik

- Langkah 1: Buat Bot Amazon Lex

- Langkah 2: Buat akun Twilio o o
- Langkah 3: Integrasikan Endpoint Layanan Pesan Twilio dengan Amazon Lex Bot
- Langkah 4: Uji integrasi

Langkah 1: Buat Bot Amazon Lex

Jika Anda belum memiliki bot Amazon Lex, buat dan terapkan. Dalam topik ini, kami berasumsi bahwa Anda menggunakan bot yang Anda buat di Getting Started Exercise 1. Namun demikian, Anda dapat menggunakan hal apa pun yang disediakan dalam panduan ini. Untuk Memulai Latihan 1, lihatLatihan 1: Membuat Bot Amazon Lex Menggunakan Cetak Biru (Konsol).

1. Buat bot Amazon Lex. Untuk petunjuk, lihat Latihan 1: Membuat Bot Amazon Lex Menggunakan Cetak Biru (Konsol).
2. Terapkan bot dan membuat alias. Untuk petunjuk, lihat Latihan 3: Publikasikan Versi dan Buat Alias.

Langkah 2: Buat akun Twilio o o

Daftar akun Twilio dan catat informasi akun berikut:

- AKUN SID
- TOKEN AUTENTIKASI

Untuk petunjuk pendaftaran, lihat https://www.twilio.com/console.

Langkah 3: Integrasikan Endpoint Layanan Pesan Twilio dengan Amazon Lex Bot

Untuk mengintegrasikan Twilio dengan bot Amazon Lex Anda

1. Untuk mengaitkan bot Amazon Lex dengan titik akhir SMS Twilio yang dapat diprogram, aktifkan asosiasi saluran bot di konsol Amazon Lex. Ketika asosiasi saluran bot telah diaktifkan, Amazon Lex mengembalikan URL callback. Rekam URL callback ini karena Anda membutuhkannya nanti.

    a. Masuk keAWS Management Console dan buka konsol Amazon Lex Lex di https://console.aws.amazon.com/lex/.

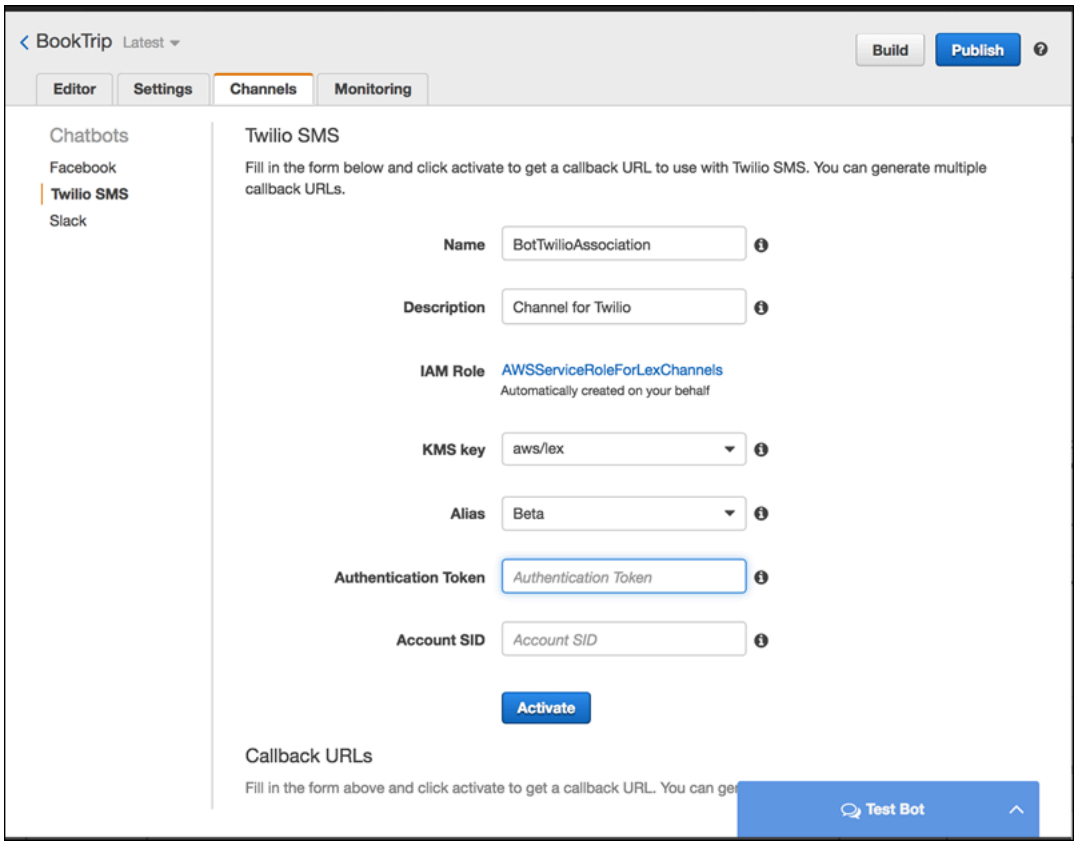
    b. Pilih bot Amazon Lex yang Anda buat di Langkah 1.

c. Pilih tab Saluran.

d. Di bagian Chatbots, pilih Twilio SMS.

e. Di halaman Twilio SMS, berikan informasi berikut ini:

- Ketik nama. Sebagai contoh, `BotTwilioAssociation`.
- Pilih “aws/lex” dari tombol KMS.
- Untuk Alias, pilih alias bot.
- Untuk Token Otentikasi, ketik TOKEN AUTH untuk akun Twilio Anda.
- Untuk Akun SID, ketik SID AKUN untuk akun Twilio Anda.



f. Pilih Aktifkan.

Konsol membuat asosiasi saluran bot dan mengembalikan URL callback. Rekam URL ini.

2. Pada konsol Twilio, hubungkan titik akhir Twilio SMS ke bot Amazon Lex.

a. Masuk ke konsol Twilio di https://www.twilio.com/console.

b. Jika Anda tidak memiliki titik akhir Twilio SMS, buat titik akhir Twilio SMS, buat titik akhir Twilio SMS, buatlah.

c. Perbarui konfigurasi Pengaturan Masuk layanan pesan dengan menyetel nilai URL PERMINTAAN ke URL callback yang disediakan Amazon Lex pada langkah sebelumnya.

Langkah 4: Uji integrasi

Gunakan ponsel Anda untuk menguji integrasi antara Twilio SMS dan bot Anda.

Untuk menguji integrasi

1. Masuk ke konsol Twilio di https://www.twilio.com/console dan lakukan hal-hal berikut ini:

   a. Pastikan Anda memiliki nomor Twilio yang terkait dengan layanan pesan di bawah Kelola Nomor.

      Anda mengirim pesan ke nomor ini dan terlibat dalam interaksi SMS dengan bot Amazon Lex dari ponsel Anda.

   b. Pastikan ponsel Anda terdaftar sebagai Verified Caller ID.

      Jika tidak, ikuti petunjuk di konsol Twilio untuk mengaktifkan ponsel yang akan digunakan untuk pengujian.

      Sekarang Anda dapat menggunakan ponsel Anda untuk mengirim pesan ke titik akhir Twilio SMS, yang dipetakan ke bot Amazon Lex.

2. Menggunakan ponsel Anda, kirim pesan ke nomor Twilio.

   Bot Amazon Lex merespons. Jika Anda membuat bot menggunakan Latihan Memulai 1, Anda dapat menggunakan contoh percakapan yang disediakan dalam latihan itu. Untuk informasi selengkapnya, lihat Langkah 4: Tambahkan Fungsi Lambda sebagai Kode Hook (Konsol).

Menyebarkan Amazon Lex Bot di Aplikasi Seluler

MenggunakanAWS Amplify, Anda dapat mengintegrasikan bot Amazon Lex Anda dengan aplikasi mobile atau web. Untuk informasi selengkapnya, lihatInteraksi — MemulaidiAWS AmplifyDokumen.

Mengimpor dan Mengekspor Amazon Lex Bots, Maksud, dan Jenis Slot

Anda dapat mengimpor atau mengekspor bot, maksud, atau jenis slot. Misalnya, jika Anda ingin berbagi bot dengan kolega di akun AWS yang berbeda, Anda dapat mengekspornya, lalu mengirimkannya kepadanya. Jika Anda ingin menambahkan beberapa ucapan ke bot, Anda dapat mengekspornya, menambahkan ucapannya, lalu mengimpornya kembali ke akun Anda.

Anda dapateksporbot, maksud, dan jenis slot baik di Amazon Lex (untuk berbagi atau memodifikasinya) atau format keterampilan Alexa. Anda dapatimporhanya dalam format Amazon Lex.

Ketika Anda mengekspor sumber daya, Anda harus mengekspornya dalam format yang kompatibel dengan layanan yang Anda ekspor ke, Amazon Lex atau Alexa Skills Kit. Jika Anda mengekspor bot dalam format Amazon Lex, Anda dapat mengimpor ulang ke akun Anda, atau pengguna Amazon Lex di akun lain dapat mengimpornya ke akunnya. Anda juga dapat mengekspor bot dalam format yang kompatibel dengan keterampilan Alexa. Kemudian Anda dapat mengimpor bot menggunakan Alexa Skills Kit untuk membuat bot Anda tersedia dengan Alexa. Untuk informasi selengkapnya, lihat Mengekspor ke Keterampilan Alexa.

Saat Anda mengekspor jenis bot, maksud, atau slot, sumber dayanya ditulis ke file JSON. Untuk mengekspor tipe bot, maksud, atau slot, Anda dapat menggunakan konsol Amazon Lex atauGetExportoperasi. Mengimpor bot, maksud, atau jenis slot menggunakanStartImport.

Topik

- Mengekspor dan Mengimpor dalam Amazon Lex Format
- Mengekspor ke Keterampilan Alexa

Mengekspor dan Mengimpor dalam Amazon Lex Format

Untuk mengekspor bot, maksud, dan jenis slot, dari Amazon Lex dengan tujuan mengimpor ulang ke Amazon Lex, Anda menggunakan membuat file JSON dalam format Amazon Lex. Anda dapat mengedit sumber daya Anda dalam file ini dan mengimpornya kembali ke Amazon Lex. Misalnya, Anda dapat menambahkan ucapan ke intent dan kemudian mengimpor intent yang diubah kembali ke akun Anda. Anda juga dapat menggunakan format JSON untuk berbagi sumber daya. Misalnya,

Anda dapat mengekspor bot dari satu Wilayah AWS dan kemudian mengimpornya ke Wilayah lain. Atau Anda dapat mengirim file JSON ke kolega untuk berbagi bot.

Topik

Mengekspor dalam Amazon Lex Format

Ekspor bot Amazon Lex Anda, maksud, dan jenis slot ke format yang dapat Anda impor keAWSakun. Anda dapat mengekspor sumber daya berikut:

- Bot, termasuk semua maksud dan jenis slot khusus yang digunakan oleh bot
- Maksud, termasuk semua jenis slot khusus yang digunakan oleh intent
- Jenis slot khusus, termasuk semua nilai untuk jenis slot

Anda dapat mengekspor hanya versi sumber daya bernomor. Anda tidak dapat mengekspor sumber daya`$LATEST`versi.

Mengekspor adalah proses asynchronous. Saat ekspor selesai, Anda mendapatkan URL Amazon S3 yang ditandatangani sebelumnya. URL menyediakan lokasi arsip .zip yang berisi sumber daya yang diekspor dalam format JSON.

Anda menggunakan konsol atauGetExportoperasi untuk mengekspor bot, maksud, dan jenis slot khusus.

Proses untuk mengekspor, bot, maksud, atau jenis slot adalah sama. Dalam prosedur berikut, maksud pengganti atau jenis slot untuk bot.

Mengekspor Bot

Untuk mengekspor bot

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon Lex dihttps://console.aws.amazon.com/lex/.
2. MemiilihBot, lalu pilih bot untuk diekspor.
3. PadaTindakanmenu, pilihEkspor.

4. DiEkspor Botdialog, pilih versi bot untuk diekspor. UntukPlatform, pilihAmazon Lex.
5. MemiilihEkspor.
6. Unduh dan simpan arsip.zip.

Amazon Lex mengekspor bot ke file JSON yang terkandung dalam arsip.zip. Untuk memperbarui bot, ubah teks JSON, lalu impor kembali ke Amazon Lex.

Langkah selanjutnya

Mengimpor dalam Amazon Lex Format

Mengimpor dalam Amazon Lex Format

Setelah Anda mengekspor sumber daya ke file JSON dalam format Amazon Lex, Anda dapat mengimpor file JSON yang berisi sumber daya menjadi satu atau lebihAWSakun. Misalnya, Anda dapat mengekspor bot, lalu mengimpornya ke Wilayah AWS lain. Atau Anda dapat mengirim bot ke kolega sehingga dia dapat mengimpornya ke akunnya.

Ketika Anda mengimpor bot, maksud, atau jenis slot, Anda harus memutuskan apakah Anda ingin menimpa`$LATEST`versi sumber daya, seperti maksud atau jenis slot, selama impor, atau jika Anda ingin impor gagal jika Anda ingin mempertahankan sumber daya yang ada di akun Anda. Misalnya, jika Anda mengunggah versi sumber daya yang diedit ke akun Anda, Anda akan memilih untuk menimpa`$LATEST`versi. Jika Anda mengunggah sumber daya yang dikirimkan kepada Anda oleh rekan kerja, Anda dapat memilih agar impor gagal jika ada konflik sumber daya sehingga sumber daya Anda sendiri tidak diganti.

Saat mengimpor sumber daya, izin yang ditetapkan kepada pengguna yang membuat permintaan impor berlaku. Pengguna harus memiliki izin untuk semua sumber daya di akun yang impor mempengaruhi. Pengguna juga harus memiliki izin untukGetBot,PutBot,GetIntent PutIntent,GetSlotType,PutSlotTypeoperasi. Untuk informasi lebih lanjut tentang izin, lihat Bagaimana Amazon Lex bekerja dengan IAM.

Impor melaporkan kesalahan yang terjadi selama pemrosesan. Beberapa kesalahan dilaporkan sebelum impor dimulai, yang lain dilaporkan selama proses impor. Misalnya, jika akun yang mengimpor maksud tidak memiliki izin untuk memanggil fungsi Lambda yang digunakan intent, impor gagal sebelum perubahan dilakukan pada jenis slot atau maksud. Jika impor gagal selama proses impor,`$LATEST`versi intent atau jenis slot apa pun yang diimpor sebelum proses gagal diubah. Anda tidak dapat memutar kembali perubahan yang dibuat ke`$LATEST`versi.

Ketika Anda mengimpor sumber daya, semua sumber daya tergantung diimpor ke`$LATEST`versi sumber daya dan kemudian diberi versi bernomor. Misalnya, jika bot menggunakan intent, intent diberi versi bernomor. Jika maksud menggunakan jenis slot khusus, jenis slot diberi versi bernomor.

Sumber daya diimpor hanya sekali. Misalnya, jika bot berisi`OrderPizza`maksud dan`OrderDrink`maksud bahwa keduanya bergantung pada jenis slot khusus`Size`, yang`Size`Jenis Slot diimpor sekali dan digunakan untuk kedua maksud.

> **Note**
>
> Jika Anda mengekspor bot Anda dengan`enableModelImprovements`parameter diatur ke`false`, Anda harus membuka file.zip yang berisi definisi bot dan mengubah`enableModelImprovements`parameter`true`di Wilayah berikut:
>
> - Asia Pacific (Singapore) (ap-southeast-1)
> - Asia Pacific (Tokyo) (ap-northeast-1)
> - EU (Frankfurt) (eu-central-1)
> - EU (London) (eu-west-2)

Proses untuk mengimpor bot, maksud, atau jenis slot khusus adalah sama. Dalam prosedur berikut, maksud pengganti atau jenis slot, yang sesuai.

Mengimpor Bot

Untuk mengimpor bot

1. Masuk ke Konsol Manajemen AWS, dan buka konsol Amazon Lex dihttps://console.aws.amazon.com/lex/.
2. MemiilihBot, lalu pilih bot untuk diimpor. Untuk mengimpor bot baru, lewati langkah ini.
3. UntukTindakan, pilihImpor.
4. UntukImpor Bot, pilih arsip.zip yang berisi file JSON yang berisi bot untuk diimpor. Jika Anda ingin melihat konflik penggabungan sebelum penggabungan, pilihBeritahu saya tentang konflik penggabungan. Jika Anda mematikan pemeriksaan konflik,`$LATEST`versi dari semua sumber daya yang digunakan oleh bot ditimpa.
5. Pilih Import (Impor). Jika Anda telah memilih untuk diberi tahu tentang konflik penggabungan dan ada konflik, sebuah dialog akan muncul yang mencantumkan daftar mereka. Untuk

menimpa$LATESTversi semua sumber daya yang saling bertentangan, pilihTimpa dan lanjutkan. Untuk menghentikan impor, pilihMembatalkan.

Anda sekarang dapat menguji bot di akun Anda.

Format JSON untuk Mengimpor dan Mengekspor

Contoh berikut menunjukkan struktur JSON untuk mengekspor dan mengimpor jenis slot, maksud, dan bot dalam format Amazon Lex.

Struktur Jenis Slot

Berikut ini adalah struktur JSON untuk jenis slot kustom. Gunakan struktur ini saat Anda mengimpor atau mengekspor jenis slot, dan saat Anda mengekspor maksud yang bergantung pada jenis slot khusus.

```
{
  "metadata": {
    "schemaVersion": "1.0",
    "importType": "LEX",
    "importFormat": "JSON"
  },
  "resource": {
    "name": "slot type name",
    "version": "version number",
    "enumerationValues": [
      {
        "value": "enumeration value",
        "synonyms": []
      },
      {
        "value": "enumeration value",
        "synonyms": []
      }
    ],
    "valueSelectionStrategy": "ORIGINAL_VALUE or TOP_RESOLUTION"
  }
}
```

Struktur maksud

Berikut ini adalah struktur JSON untuk maksud. Gunakan struktur ini saat Anda mengimpor atau mengekspor maksud dan bot yang bergantung pada maksud.

```
{
  "metadata": {
    "schemaVersion": "1.0",
    "importType": "LEX",
    "importFormat": "JSON"
  },
  "resource": {
    "description": "intent description",
    "rejectionStatement": {
      "messages": [
        {
          "contentType": "PlainText or SSML or CustomPayload",
          "content": "string"
        }
      ]
    },
    "name": "intent name",
    "version": "version number",
    "fulfillmentActivity": {
      "type": "ReturnIntent or CodeHook"
    },
    "sampleUtterances": [
      "string",
      "string"
    ],
    "slots": [
      {
        "name": "slot name",
        "description": "slot description",
        "slotConstraint": "Required or Optional",
        "slotType": "slot type",
        "valueElicitationPrompt": {
          "messages": [
            {
              "contentType": "PlainText or SSML or CustomPayload",
              "content": "string"
            }
          ],
```

```
          "maxAttempts": value
        },
        "priority": value,
        "sampleUtterances": []
      }
    ],
    "confirmationPrompt": {
      "messages": [
        {
          "contentType": "PlainText or SSML or CustomPayload",
          "content": "string"
        },
        {
          "contentType": "PlainText or SSML or CustomPayload",
          "content": "string"
        }
      ],
      "maxAttempts": value
    },
    "slotTypes": [
        List of slot type JSON structures.
        For more information, see Struktur Jenis Slot.
    ]
  }
}
```

Struktur bot

Berikut ini adalah struktur JSON untuk bot. Gunakan struktur ini saat Anda mengimpor atau mengekspor bot.

```
{
  "metadata": {
    "schemaVersion": "1.0",
    "importType": "LEX",
    "importFormat": "JSON"
  },
  "resource": {
    "name": "bot name",
    "version": "version number",,
    "nluIntentConfidenceThreshold": 0.00-1.00,
    "enableModelImprovements": true | false,
    "intents": [
```

```
        List of intent JSON structures.
        For more information, see Struktur maksud.
    ],
    "slotTypes": [
        List of slot type JSON structures.
        For more information, see Struktur Jenis Slot.
    ],
    "voiceId": "output voice ID",
    "childDirected": boolean,
    "locale": "en-US",
    "idleSessionTTLInSeconds": timeout,
    "description": "bot description",
    "clarificationPrompt": {
      "messages": [
        {
          "contentType": "PlainText or SSML or CustomPayload",
          "content": "string"
        }
      ],
      "maxAttempts": value
    },
    "abortStatement": {
      "messages": [
        {
          "contentType": "PlainText or SSML or CustomPayload",
          "content": "string"
        }
      ]
    }
  }
}
```

Mengekspor ke Keterampilan Alexa

Anda dapat mengekspor skema bot Anda dalam format yang kompatibel dengan keterampilan Alexa. Setelah Anda mengekspor bot ke file JSON, Anda mengunggahnya ke Alexa menggunakan pembangun keterampilan.

Untuk mengekspor bot dan skema (model interaksi)

1. Masuk keAWS Management Consoledan buka konsol Amazon Lex dihttps://console.aws.amazon.com/lex/.

2. Memiilih bot yang ingin Anda ekspor.
3. UntukTindakan, pilihEkspor.
4. Memiilih versi bot yang ingin Anda ekspor. Untuk formatnya, pilihAlexa Keterampilan Kit, kemudian pilihEkspor.
5. Jika kotak dialog unduhan muncul, pilih lokasi untuk menyimpan file, lalu pilihSimpan.

File yang diunduh adalah arsip .zip yang berisi satu file dengan nama bot yang diekspor. Ini berisi informasi yang diperlukan untuk mengimpor bot sebagai keterampilan Alexa.

> **Note**
>
> Amazon Lex dan Alexa Skills Kit berbeda dengan cara berikut:
>
> - Atribut sesi, dilambangkan dengan tanda kurung persegi ([]), tidak didukung oleh Alexa Skills Kit. Anda perlu memperbarui petunjuk yang menggunakan atribut sesi.
> - Tanda baca tidak didukung oleh Alexa Skills Kit. Anda perlu memperbarui ucapan yang menggunakan tanda baca.

Untuk mengunggah bot ke Alexa Skill

1. Masuk ke portal pengembang dihttps://developer.amazon.com/.
2. PadaKeterampilan Alexahalaman, pilihMembuat Keterampilan.
3. PadaMembuat keterampilan baruhalaman, masukkan nama keterampilan dan bahasa default untuk keterampilan. PastikanKhususdipilih untuk model keterampilan, dan kemudian memilihMembuat keterampilan.
4. PastikanMulai dari awaldipilih dan memilihMemiilih.
5. Dari menu sebelah kiri, pilihPenyunting JSON. Seret file JSON yang Anda ekspor dari Amazon Lex ke editor JSON.
6. MemiilihSimpan Modeluntuk menyimpan model interaksi Anda.

Setelah mengunggah skema ke dalam keterampilan Alexa, buat perubahan yang diperlukan untuk menjalankan keterampilan dengan Alexa. Untuk informasi selengkapnya tentang membuat keterampilan Alexa, lihatGunakan Skill Builder (Beta)diAlexa Keterampilan Kit.

Contoh Tambahan: Membuat Amazon Lex Bots

Bagian berikut menyediakan latihan Amazon Lex tambahan dengan step-by-step instruksi.

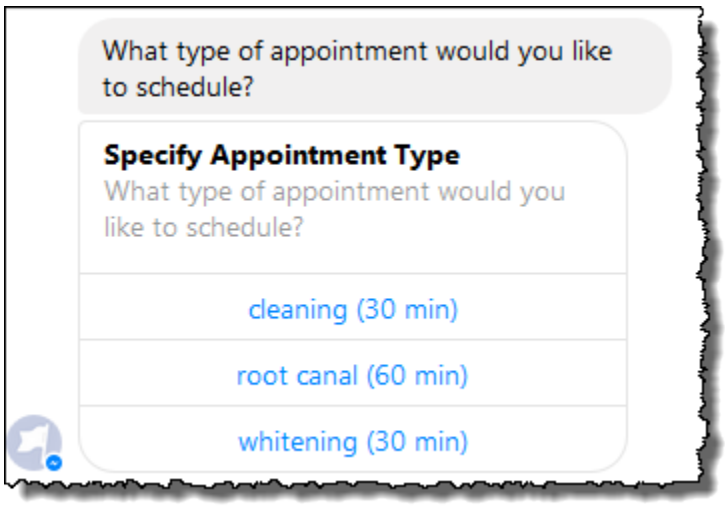
Topik

Jadwalkan Janji

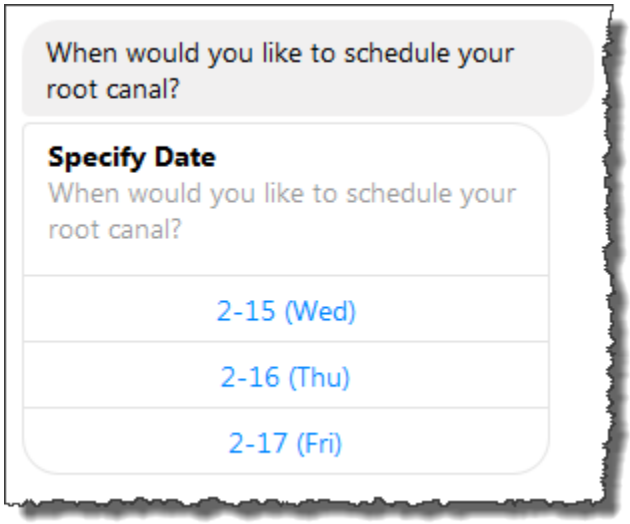
Contoh bot dalam latihan ini menjadwalkan janji temu untuk kantor dokter gigi. Contoh ini juga menggambarkan menggunakan kartu respon untuk mendapatkan input pengguna dengan tombol. Secara khusus, contoh menggambarkan menghasilkan kartu respon secara dinamis pada saat runtime.

Anda dapat mengonfigurasi kartu respons pada waktu pembuatan (juga disebut sebagai kartu respons statis) atau membuatnya secara dinamis dalam suatu AWS Lambda fungsi. Dalam contoh ini, bot menggunakan kartu respons berikut:

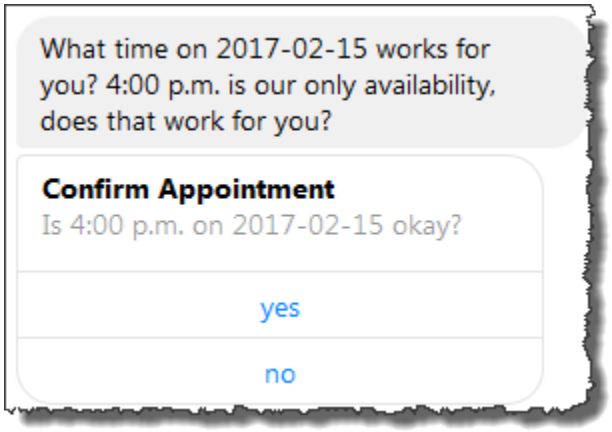
- Kartu respons yang mencantumkan tombol untuk jenis janji temu. Lihat gambar berikut untuk contoh:



- Kartu respons yang mencantumkan tombol untuk tanggal janji temu. Lihat gambar berikut untuk contoh:



- Kartu respons yang mencantumkan tombol untuk mengonfirmasi waktu janji temu yang disarankan. Lihat gambar berikut untuk contoh:



Tanggal dan waktu janji temu yang tersedia bervariasi, yang mengharuskan Anda menghasilkan kartu respons saat runtime. Anda menggunakan AWS Lambda fungsi untuk menghasilkan kartu respons ini secara dinamis. Fungsi Lambda mengembalikan kartu respons dalam responsnya terhadap Amazon Lex. Amazon Lex menyertakan kartu respons dalam responsnya terhadap klien.

Jika klien (misalnya, Facebook Messenger) mendukung kartu respons, pengguna dapat memilih dari daftar tombol atau mengetik respons. Jika tidak, pengguna cukup mengetikkan respons.

Selain tombol yang ditunjukkan pada contoh sebelumnya, Anda juga dapat menyertakan gambar, lampiran, dan informasi berguna lainnya untuk ditampilkan pada kartu respons. Untuk informasi tentang kartu respons, lihatKartu Respon.

Dalam latihan ini, Anda melakukan hal berikut:

- Buat dan uji bot (menggunakan ScheduleAppointment cetak biru). Untuk latihan ini, Anda menggunakan cetak biru bot untuk mengatur dan menguji bot dengan cepat. Untuk daftar cetak biru yang tersedia, lihat Bot Amazon Lex danAWS LambdaCetak Biru ini telah dikonfigurasi sebelumnya dengan satu intent (). `MakeAppointment`

- Buat dan uji fungsi Lambda (menggunakan lex-make-appointment-python cetak biru yang disediakan oleh Lambda). Anda mengonfigurasi `MakeAppointment` maksud untuk menggunakan fungsi Lambda ini sebagai pengait kode untuk melakukan tugas inisialisasi, validasi, dan pemenuhan.

  > **Note**
  >
  > Contoh yang disediakan fungsi Lambda menampilkan percakapan dinamis berdasarkan ketersediaan mocked-up janji dokter gigi. Dalam aplikasi nyata, Anda mungkin menggunakan kalender nyata untuk mengatur janji temu.

- Perbarui konfigurasi `MakeAppointment` maksud untuk menggunakan fungsi Lambda sebagai pengait kode. Kemudian, uji pengalaman end-to-end.
- Publikasikan bot janji temu jadwal ke Facebook Messenger sehingga Anda dapat melihat kartu respons beraksi (klien di konsol Amazon Lex saat ini tidak mendukung kartu respons).

Bagian berikut memberikan informasi ringkasan tentang cetak biru yang Anda gunakan dalam latihan ini.

Topik

Ikhtisar Cetak Biru Bot () ScheduleAppointment

ScheduleAppointmentCetak biru yang Anda gunakan untuk membuat bot untuk latihan ini telah dikonfigurasi sebelumnya dengan yang berikut ini:

- jenis Slot - Satu jenis slot kustom disebut`AppointmentTypeValue`, dengan nilai-nilai pencacahan`root canal`,, `cleaning` dan. `whitening`
- Intent - Satu intent (`MakeAppointment`), yang telah dikonfigurasi sebelumnya sebagai berikut:
  - Slot - Maksud dikonfigurasi dengan slot berikut:
    - Slot`AppointmentType`, dari jenis `AppointmentTypes` kustom.
    - Slot`Date`, dari tipe `AMAZON.DATE` built-in.
    - Slot`Time`, dari tipe `AMAZON.TIME` built-in.
  - Ucapan - Maksud telah dikonfigurasi sebelumnya dengan ucapan berikut:
    - “Saya ingin memesan janji temu”
    - “Pesan janji”
    - “Pesan {AppointmentType}”

    Jika pengguna mengucapkan salah satu dari ini, Amazon Lex menentukan `MakeAppointment` bahwa maksud, dan kemudian menggunakan petunjuk untuk mendapatkan data slot.
  - Anjuran - Maksud telah dikonfigurasi sebelumnya dengan petunjuk berikut:
    - Prompt untuk `AppointmentType` slot - “Jenis janji apa yang ingin Anda jadwalkan?”

- Prompt untuk `Date` slot - “Kapan saya harus menjadwalkan {AppointmentType} Anda?”
- Prompt untuk `Time` slot — “Pada jam berapa Anda ingin menjadwalkan {AppointmentType}?” and

  “Pada jam berapa pada {Date}?”
- Prompt konfirmasi - “{Waktu} tersedia, haruskah saya melanjutkan dan memesan janji temu Anda?”
- Batalkan pesan— “Oke, saya tidak akan menjadwalkan janji temu.”

Ikhtisar Cetak Biru Fungsi Lambda () lex-make-appointment-python

Cetak biru fungsi Lambda (lex-make-appointment-python) adalah hook kode untuk bot yang Anda buat menggunakan cetak biru bot. ScheduleAppointment

Kode cetak biru fungsi Lambda ini dapat melakukan tugas inisialisasi/validasi dan pemenuhan.

- Kode fungsi Lambda menampilkan percakapan dinamis yang didasarkan pada ketersediaan contoh untuk janji dokter gigi (dalam aplikasi nyata, Anda mungkin menggunakan kalender). Untuk hari atau tanggal yang ditentukan pengguna, kode dikonfigurasi sebagai berikut:
  - Jika tidak ada janji temu yang tersedia, fungsi Lambda mengembalikan respons yang mengarahkan Amazon Lex untuk meminta pengguna untuk hari atau tanggal lain (dengan menyetel jenisnya. `dialogAction` `ElicitSlot`) Untuk informasi selengkapnya, lihat Format Respons.
  - Jika hanya ada satu janji temu yang tersedia pada hari atau tanggal yang ditentukan, fungsi Lambda menyarankan waktu yang tersedia dalam respons dan mengarahkan Amazon Lex untuk mendapatkan konfirmasi pengguna dengan menyetel respons ke. `dialogAction` `ConfirmIntent` Ini menggambarkan bagaimana Anda dapat meningkatkan pengalaman pengguna dengan secara proaktif menyarankan waktu yang tersedia untuk janji temu.
  - Jika ada beberapa janji temu yang tersedia, fungsi Lambda mengembalikan daftar waktu yang tersedia dalam respons terhadap Amazon Lex. Amazon Lex mengembalikan respons ke klien dengan pesan dari fungsi Lambda.
- Sebagai hook kode pemenuhan, fungsi Lambda mengembalikan pesan ringkasan yang menunjukkan bahwa janji temu dijadwalkan (yaitu, intent terpenuhi).

> **Note**
>
> Dalam contoh ini, kami menunjukkan cara menggunakan kartu respons. Fungsi Lambda membangun dan mengembalikan kartu respons ke Amazon Lex. Kartu respons mencantumkan hari dan waktu yang tersedia sebagai tombol untuk dipilih. Saat menguji bot menggunakan klien yang disediakan oleh konsol Amazon Lex, Anda tidak dapat melihat kartu respons. Untuk melihatnya, Anda harus mengintegrasikan bot dengan platform perpesanan, seperti Facebook Messenger. Untuk petunjuk, lihat Mengintegrasikan Amazon Lex Bot dengan Facebook Messenger. Untuk informasi selengkapnya tentang kartu respons, lihatMengelola Pesan .

Saat Amazon Lex memanggil fungsi Lambda, ia meneruskan data peristiwa sebagai input. Salah satu bidang acara adalah`invocationSource`, yang digunakan fungsi Lambda untuk memilih antara validasi input dan aktivitas pemenuhan. Untuk informasi selengkapnya, lihat Format Peristiwa Masukan.

Langkah Selanjutnya

Langkah 1: Buat Amazon Lex Bot

Langkah 1: Buat Amazon Lex Bot

Di bagian ini, Anda membuat bot Amazon Lex menggunakan ScheduleAppointment cetak biru, yang disediakan di konsol Amazon Lex.

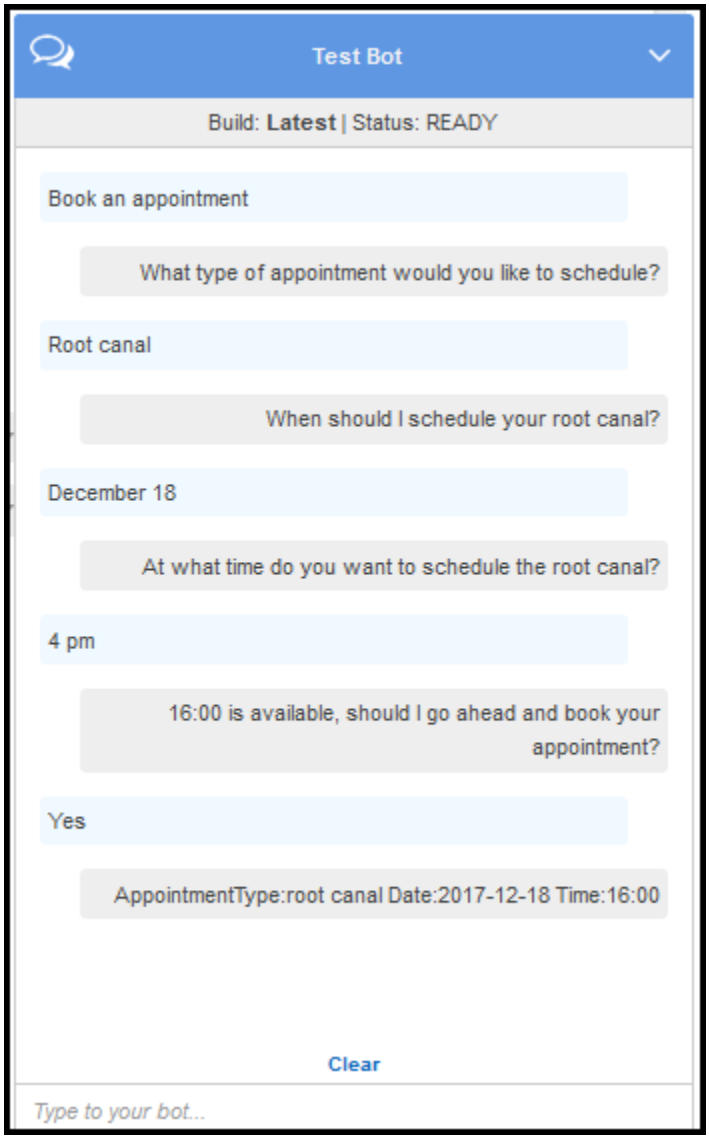
1. Masuk ke AWS Management Console dan buka konsol Amazon Lex di https://console.aws.amazon.com/lex/.
2. Pada halaman Bot, pilih Buat.
3. Pada halaman Create your Lex bot, lakukan hal berikut:
   - Pilih ScheduleAppointmentcetak biru.
   - Tinggalkan nama bot default (ScheduleAppointment).
4. Pilih Create (Buat).

   Langkah ini menyimpan dan membangun bot. Konsol mengirimkan permintaan berikut ke Amazon Lex selama proses pembangunan:

- Buat versi baru dari jenis slot (dari versi $ TERBARU). Untuk informasi tentang jenis slot yang didefinisikan dalam cetak biru bot ini, lihat[Ikhtisar Cetak Biru Bot () ScheduleAppointment](#).
- Buat versi `MakeAppointment` maksud (dari versi $LATEST). Dalam beberapa kasus, konsol mengirimkan permintaan untuk operasi `update` API sebelum membuat versi baru.
- Perbarui versi $LATEST dari bot.

  Pada saat ini, Amazon Lex membangun model pembelajaran mesin untuk bot. Saat Anda menguji bot di konsol, konsol menggunakan API waktu proses untuk mengirim input pengguna kembali ke Amazon Lex. Amazon Lex kemudian menggunakan model machine learning untuk menafsirkan input pengguna.

5. Konsol menunjukkan ScheduleAppointment bot. Pada tab Editor, tinjau detail intent (`MakeAppointment`) yang telah dikonfigurasi sebelumnya.
6. Uji bot di jendela pengujian. Gunakan screen shot berikut untuk terlibat dalam percakapan uji dengan bot Anda:



Perhatikan hal berikut:

- Dari input pengguna awal (“Pesan janji temu”), bot menyimpulkan intent (`MakeAppointment`).
- Bot kemudian menggunakan petunjuk yang dikonfigurasi untuk mendapatkan data slot dari pengguna.
- Cetak biru bot memiliki `MakeAppointment` maksud yang dikonfigurasi dengan prompt konfirmasi berikut:

```
{Time} is available, should I go ahead and book your appointment?
```

Setelah pengguna menyediakan semua data slot, Amazon Lex mengembalikan respons kepada klien dengan konfirmasi prompt sebagai pesan. Klien menampilkan pesan untuk pengguna:

```
16:00 is available, should I go ahead and book your appointment?
```

Perhatikan bahwa bot menerima nilai tanggal dan waktu janji temu karena Anda tidak memiliki kode apa pun untuk menginisialisasi atau memvalidasi data pengguna. Di bagian selanjutnya, Anda menambahkan fungsi Lambda untuk melakukan ini.

Langkah Selanjutnya

Langkah 2: Buat Fungsi Lambda

Langkah 2: Buat Fungsi Lambda

Di bagian ini, Anda membuat fungsi Lambda menggunakan blueprint (lex-make-appointment-python) yang disediakan di konsol Lambda. Anda juga menguji fungsi Lambda dengan memanggilnya menggunakan contoh data peristiwa Amazon Lex yang disediakan oleh konsol.

1. Masuk ke AWS Management Console dan buka konsol AWS Lambda di https://console.aws.amazon.com/lambda/.
2. Pilih Buat fungsi Lambda.
3. Untuk Pilih cetak biru, ketik **`lex`** untuk menemukan cetak biru, lalu pilih cetak biru. lex-make-appointment-python
4. Konfigurasikan fungsi Lambda sebagai berikut.
   - Ketik nama fungsi Lambda ()`MakeAppointmentCodeHook`.
   - Untuk peran tersebut, pilih Buat peran baru dari template, lalu ketik nama peran.
   - Tinggalkan nilai default lainnya.
5. Pilih Buat Fungsi.
6. Jika Anda menggunakan lokal selain bahasa Inggris (AS) (id), perbarui nama maksud seperti yang dijelaskan di. Memperbarui Cetak Biru untuk Lokal Tertentu
7. Uji fungsi Lambda.

a. Pilih Tindakan, lalu pilih Konfigurasikan peristiwa pengujian.
b. Dari daftar template acara Contoh, pilih Lex-Make Appointment (pratinjau). Peristiwa contoh ini menggunakan model permintaan/respons Amazon Lex, dengan nilai yang ditetapkan agar sesuai dengan permintaan dari bot Amazon Lex Anda. Untuk informasi tentang model permintaan/respons Amazon Lex, lihat. Menggunakan Fungsi Lambda
c. Pilih Simpan dan uji.
d. Verifikasi bahwa fungsi Lambda berjalan dengan sukses. Tanggapan dalam hal ini cocok dengan model respons Amazon Lex.

Langkah Selanjutnya

Langkah 3: Perbarui Intent: Konfigurasikan Hook Kode

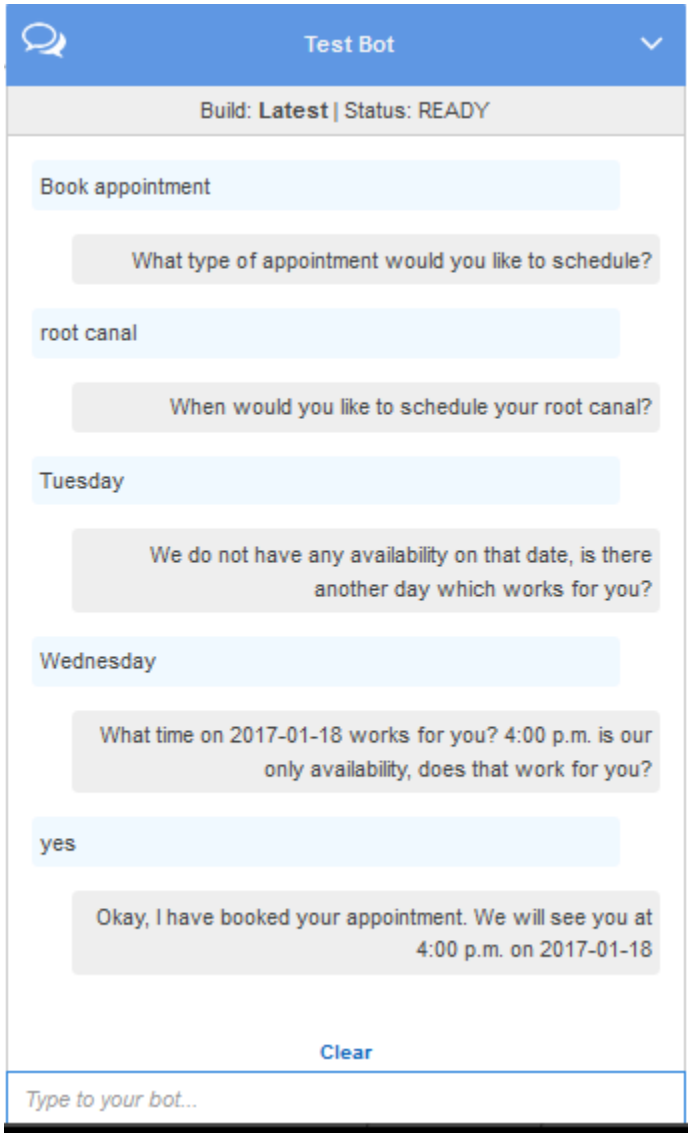
Langkah 3: Perbarui Intent: Konfigurasikan Hook Kode

Di bagian ini, Anda memperbarui konfigurasi `MakeAppointment` maksud untuk menggunakan fungsi Lambda sebagai pengait kode untuk aktivitas validasi dan pemenuhan.

1. Di konsol Amazon Lex, pilih ScheduleAppointment bot. Konsol menunjukkan MakeAppointmentintent. Ubah konfigurasi maksud sebagai berikut.

   > **Note**
   >
   > Anda hanya dapat memperbarui versi $LATEST dari salah satu sumber daya Amazon Lex, termasuk maksud. Pastikan versi intent disetel ke $LATEST. Anda belum menerbitkan versi bot Anda, jadi itu harus tetap menjadi versi $LATEST di konsol.

   a. Di bagian Opsi, pilih Pengait kode inisialisasi dan validasi, lalu pilih fungsi Lambda dari daftar.
   b. Di bagian Fulfillment, pilih fungsi AWS Lambda, lalu pilih fungsi Lambda dari daftar.
   c. Pilih Pesan selamat tinggal, dan ketik pesan.
2. Pilih Simpan, lalu pilih Bangun.
3. Uji bot, seperti pada gambar berikut:



Langkah Selanjutnya

Langkah 4: Menyebarkan Bot di Platform Facebook Messenger

Langkah 4: Menyebarkan Bot di Platform Facebook Messenger

Di bagian sebelumnya, Anda menguji ScheduleAppointment bot menggunakan klien di konsol Amazon Lex. Saat ini, konsol Amazon Lex tidak mendukung kartu respons. Untuk menguji kartu respons yang dihasilkan secara dinamis yang didukung bot, gunakan bot di platform Facebook Messenger dan uji.

Untuk petunjuk, lihat Mengintegrasikan Amazon Lex Bot dengan Facebook Messenger.

Langkah Selanjutnya

Rincian Alur Informasi

Rincian Alur Informasi

Cetak biru `ScheduleAppointment` bot terutama menampilkan penggunaan kartu respons yang dihasilkan secara dinamis. Fungsi Lambda dalam latihan ini mencakup kartu respons dalam responsnya terhadap Amazon Lex. Amazon Lex menyertakan kartu respons dalam balasannya kepada klien. Bagian ini menjelaskan kedua hal berikut:

- Aliran data antara klien dan Amazon Lex.

  Bagian ini mengasumsikan klien mengirimkan permintaan ke Amazon Lex menggunakan API `PostText` waktu proses dan menampilkan detail permintaan/respons yang sesuai. Untuk informasi selengkapnya tentang API `PostText` runtime, lihatPostText.

  > **Note**
  >
  > Untuk contoh alur informasi antara klien dan Amazon Lex di mana klien menggunakan `PostContent` API, lihatLangkah 2a (Opsional): Tinjau Rincian Aliran Informasi Lisan (Konsol) .

- Aliran data antara Amazon Lex dan fungsi Lambda. Untuk informasi selengkapnya, lihat Lambda Fungsi Input Event dan Response Format.

> **Note**
>
> Contohnya mengasumsikan bahwa Anda menggunakan klien Facebook Messenger, yang tidak melewati atribut sesi dalam permintaan ke Amazon Lex. Dengan demikian, contoh permintaan yang ditampilkan di bagian ini menunjukkan kosong`sessionAttributes`. Jika Anda menguji bot menggunakan klien yang disediakan di konsol Amazon Lex, klien menyertakan atribut sesi.

Bagian ini menjelaskan apa yang terjadi setelah setiap input pengguna.

1. Pengguna: Jenis**`Book an appointment`**.

   a. Klien (konsol) mengirimkan PostContent permintaan berikut ke Amazon Lex:

```
POST /bot/ScheduleAppointment/alias/$LATEST/
user/bijt6rovckwecnzesbthrr1d7lv3ja3n/text
"Content-Type":"application/json"
"Content-Encoding":"amz-1.0"

{
   "inputText":"book appointment",
   "sessionAttributes":{}
}
```

   URI permintaan dan badan memberikan informasi kepada Amazon Lex:

   - Permintaan URI - Menyediakan nama bot (ScheduleAppointment), alias bot ($ TERBARU), dan ID nama pengguna. Trailing `text` menunjukkan bahwa itu adalah permintaan API `PostText` (bukan`PostContent`).
   - Permintaan tubuh - Termasuk input pengguna (`inputText`) dan kosong`sessionAttributes`.

   b. Dari`inputText`, Amazon Lex mendeteksi maksud ()`MakeAppointment`. Layanan memanggil fungsi Lambda, yang dikonfigurasi sebagai hook kode, untuk melakukan inisialisasi dan validasi dengan meneruskan peristiwa berikut. Untuk detailnya, lihat Format Peristiwa Masukan.

```
{
    "currentIntent": {
        "slots": {
            "AppointmentType": null,
            "Date": null,
            "Time": null
        },
        "name": "MakeAppointment",
        "confirmationStatus": "None"
    },
    "bot": {
        "alias": null,
        "version": "$LATEST",
```

```
        "name": "ScheduleAppointment"
    },
    "userId": "bijt6rovckwecnzesbthrr1d7lv3ja3n",
    "invocationSource": "DialogCodeHook",
    "outputDialogMode": "Text",
    "messageVersion": "1.0",
    "sessionAttributes": {}
}
```

Selain informasi yang dikirim oleh klien, Amazon Lex juga menyertakan data berikut:

- `currentIntent`- Menyediakan informasi maksud saat ini.
- `invocationSource`- Menunjukkan tujuan pemanggilan fungsi Lambda. Dalam hal ini, tujuannya adalah untuk melakukan inisialisasi data pengguna dan validasi. (Amazon Lex tahu bahwa pengguna belum menyediakan semua data slot untuk memenuhi maksud belum.)
- `messageVersion`- Saat ini Amazon Lex hanya mendukung versi 1.0.

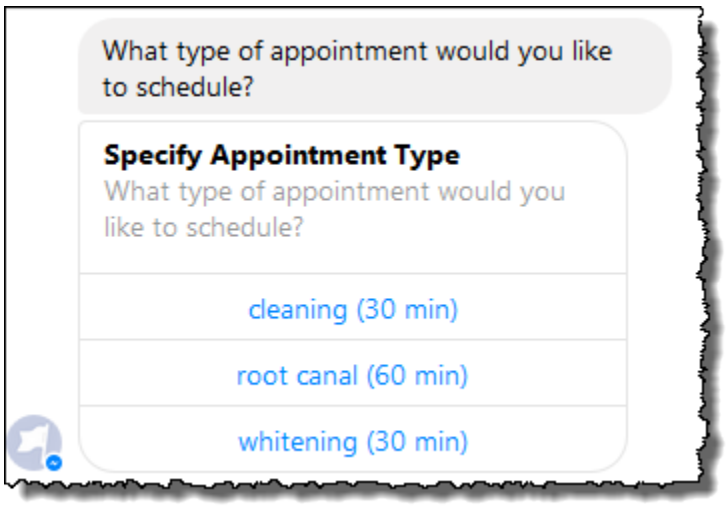
c. Pada saat ini, semua nilai slot yang nol (tidak ada yang memvalidasi). Fungsi Lambda mengembalikan respons berikut ke Amazon Lex, mengarahkan layanan untuk mendapatkan informasi untuk slot. `AppointmentType` Untuk informasi tentang format respons, lihatFormat Respons.

```
{
    "dialogAction": {
        "slotToElicit": "AppointmentType",
        "intentName": "MakeAppointment",
        "responseCard": {
            "genericAttachments": [
                {
                    "buttons": [
                        {
                            "text": "cleaning (30 min)",
                            "value": "cleaning"
                        },
                        {
                            "text": "root canal (60 min)",
                            "value": "root canal"
                        },
                        {
                            "text": "whitening (30 min)",
                            "value": "whitening"
```

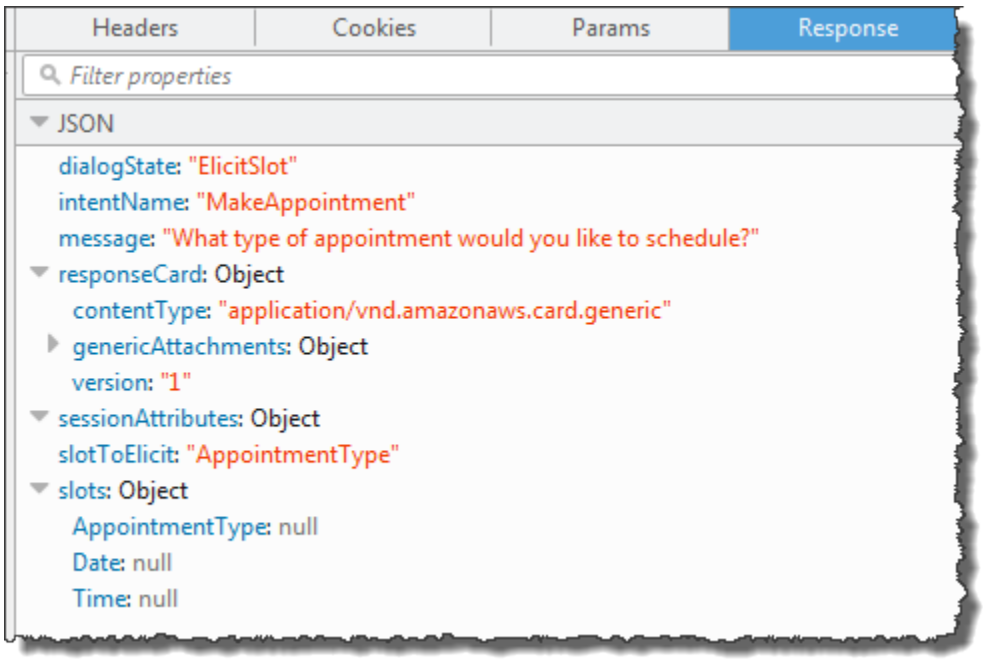
```
                        }
                    ],
                    "subTitle": "What type of appointment would you like to
 schedule?",
                    "title": "Specify Appointment Type"
                }
            ],
            "version": 1,
            "contentType": "application/vnd.amazonaws.card.generic"
        },
        "slots": {
            "AppointmentType": null,
            "Date": null,
            "Time": null
        },
        "type": "ElicitSlot",
        "message": {
            "content": "What type of appointment would you like to schedule?",
            "contentType": "PlainText"
        }
    },
    "sessionAttributes": {}
}
```

Tanggapannya meliputi `dialogAction` dan `sessionAttributes` bidang. Antara lain, `dialogAction` bidang mengembalikan bidang berikut:

- `type`- Dengan menetapkan bidang ini`ElicitSlot`, fungsi Lambda mengarahkan Amazon Lex untuk mendapatkan nilai untuk slot yang ditentukan di bidang. `slotToElicit` Fungsi Lambda juga menyediakan untuk menyampaikan `message` kepada pengguna.
- `responseCard`- Mengidentifikasi daftar nilai yang mungkin untuk `AppointmentType` slot. Klien yang mendukung kartu respons (misalnya, Facebook Messenger) menampilkan kartu respons untuk memungkinkan pengguna memilih jenis janji temu, seperti pada gambar berikut:



d. Seperti yang `dialogAction.type` ditunjukkan oleh respons dari fungsi Lambda, Amazon Lex mengirimkan respons berikut kembali ke klien:



Klien membaca respons, dan kemudian menampilkan pesan: “Jenis janji apa yang ingin Anda jadwalkan?” dan kartu respons (jika klien mendukung kartu respons).

2. Pengguna: Tergantung pada klien, pengguna memiliki dua opsi:

   - Jika kartu respons ditampilkan, pilih saluran akar (60 menit) atau ketik**`root canal`**.
   - Jika klien tidak mendukung kartu respons, ketik**`root canal`**.

a. Klien mengirimkan `PostText` permintaan berikut ke Amazon Lex (jeda baris telah ditambahkan untuk keterbacaan):

```
POST /bot/BookTrip/alias/$LATEST/user/bijt6rovckwecnzesbthrr1d7lv3ja3n/text
"Content-Type":"application/json"
"Content-Encoding":"amz-1.0"

{
   "inputText": "root canal",
    "sessionAttributes": {}
}
```

b. Amazon Lex memanggil fungsi Lambda untuk validasi data pengguna dengan mengirimkan peristiwa berikut sebagai parameter:

```
{
    "currentIntent": {
        "slots": {
            "AppointmentType": "root canal",
            "Date": null,
            "Time": null
        },
        "name": "MakeAppointment",
        "confirmationStatus": "None"
    },
    "bot": {
        "alias": null,
        "version": "$LATEST",
        "name": "ScheduleAppointment"
    },
    "userId": "bijt6rovckwecnzesbthrr1d7lv3ja3n",
    "invocationSource": "DialogCodeHook",
    "outputDialogMode": "Text",
    "messageVersion": "1.0",
    "sessionAttributes": {}
}
```

Dalam data acara, perhatikan hal berikut:

- `invocationSource`terus menjadi`DialogCodeHook`. Pada langkah ini, kita hanya memvalidasi data pengguna.
- Amazon Lex menetapkan `AppointmentType` lapangan di `currentIntent.slots` slot untuk`root canal`.
- Amazon Lex hanya melewati `sessionAttributes` bidang antara klien dan fungsi Lambda.

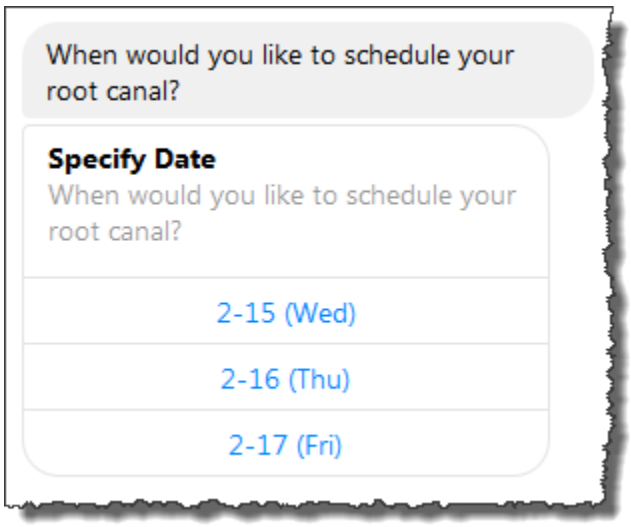
c. Fungsi Lambda memvalidasi input pengguna dan mengembalikan respons berikut ke Amazon Lex, mengarahkan layanan untuk mendapatkan nilai untuk tanggal janji temu.

```
{
    "dialogAction": {
        "slotToElicit": "Date",
        "intentName": "MakeAppointment",
        "responseCard": {
            "genericAttachments": [
                {
                    "buttons": [
                        {
                            "text": "2-15 (Wed)",
                            "value": "Wednesday, February 15, 2017"
                        },
                        {
                            "text": "2-16 (Thu)",
                            "value": "Thursday, February 16, 2017"
                        },
                        {
                            "text": "2-17 (Fri)",
                            "value": "Friday, February 17, 2017"
                        },
                        {
                            "text": "2-20 (Mon)",
                            "value": "Monday, February 20, 2017"
                        },
                        {
                            "text": "2-21 (Tue)",
                            "value": "Tuesday, February 21, 2017"
                        }
                    ],
                    "subTitle": "When would you like to schedule your root
 canal?",
```

```
                    "title": "Specify Date"
                }
            ],
            "version": 1,
            "contentType": "application/vnd.amazonaws.card.generic"
        },
        "slots": {
            "AppointmentType": "root canal",
            "Date": null,
            "Time": null
        },
        "type": "ElicitSlot",
        "message": {
            "content": "When would you like to schedule your root canal?",
            "contentType": "PlainText"
        }
    },
    "sessionAttributes": {}
}
```

Sekali lagi, respon termasuk `dialogAction` dan `sessionAttributes` bidang. Antara lain, `dialogAction` bidang mengembalikan bidang berikut:

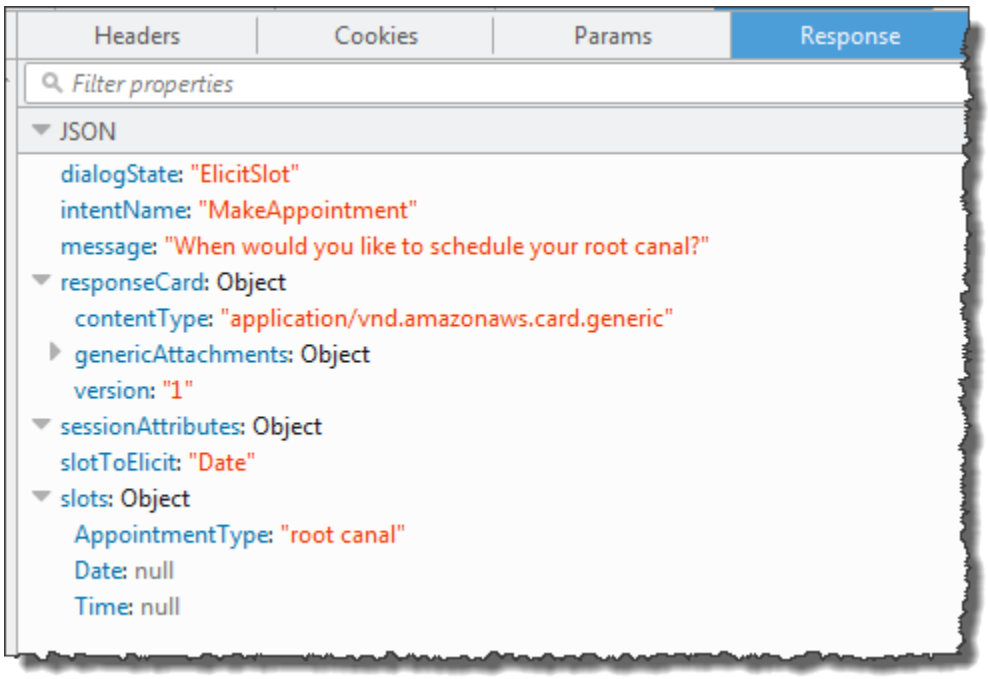
- `type`- Dengan menetapkan bidang ini`ElicitSlot`, fungsi Lambda mengarahkan Amazon Lex untuk mendapatkan nilai untuk slot yang ditentukan di bidang. `slotToElicit` Fungsi Lambda juga menyediakan untuk menyampaikan `message` kepada pengguna.
- `responseCard`- Mengidentifikasi daftar nilai yang mungkin untuk `Date` slot. Klien yang mendukung kartu respons (misalnya, Facebook Messenger) menampilkan kartu respons yang memungkinkan pengguna memilih tanggal janji temu, seperti pada gambar berikut:



Meskipun fungsi Lambda mengembalikan lima tanggal, klien (Facebook Messenger) memiliki batas tiga tombol untuk kartu respons. Oleh karena itu, Anda hanya melihat tiga nilai pertama dalam tangkapan layar.

Tanggal ini sulit dikodekan dalam fungsi Lambda. Dalam aplikasi produksi, Anda dapat menggunakan kalender untuk mendapatkan tanggal yang tersedia secara real time. Karena tanggalnya dinamis, Anda harus menghasilkan kartu respons secara dinamis dalam fungsi Lambda.

d. Amazon Lex memperhatikan `dialogAction.type` dan mengembalikan respons berikut ke klien yang menyertakan informasi dari respons fungsi Lambda.



Klien menampilkan pesan: Kapan Anda ingin menjadwalkan saluran akar Anda? dan kartu respons (jika klien mendukung kartu respons).

3. Pengguna: Jenis**`Thursday`**.

   a. Klien mengirimkan `PostText` permintaan berikut ke Amazon Lex (jeda baris telah ditambahkan untuk keterbacaan):

```
POST /bot/BookTrip/alias/$LATEST/user/bijt6rovckwecnzesbthrr1d7lv3ja3n/text
"Content-Type":"application/json"
"Content-Encoding":"amz-1.0"

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

   b. Amazon Lex memanggil fungsi Lambda untuk validasi data pengguna dengan mengirimkan peristiwa berikut sebagai parameter:

```
{
    "currentIntent": {
        "slots": {
            "AppointmentType": "root canal",
            "Date": "2017-02-16",
```

```
            "Time": null
        },
        "name": "MakeAppointment",
        "confirmationStatus": "None"
    },
    "bot": {
        "alias": null,
        "version": "$LATEST",
        "name": "ScheduleAppointment"
    },
    "userId": "u3fpr9gghj02zts7y5tpq5mm4din2xqy",
    "invocationSource": "DialogCodeHook",
    "outputDialogMode": "Text",
    "messageVersion": "1.0",
    "sessionAttributes": {}
}
```

Dalam data acara, perhatikan hal berikut:

- `invocationSource`terus menjadi`DialogCodeHook`. Pada langkah ini, kita hanya memvalidasi data pengguna.
- Amazon Lex menetapkan `Date` lapangan di `currentIntent.slots` slot untuk`2017-02-16`.
- Amazon Lex hanya melewati `sessionAttributes` antara klien dan fungsi Lambda.

c. Fungsi Lambda memvalidasi input pengguna. Kali ini fungsi Lambda menentukan bahwa tidak ada janji temu yang tersedia pada tanggal yang ditentukan. Ia mengembalikan respons berikut ke Amazon Lex, mengarahkan layanan untuk kembali mendapatkan nilai untuk tanggal janji temu.

```
{
    "dialogAction": {
        "slotToElicit": "Date",
        "intentName": "MakeAppointment",
        "responseCard": {
            "genericAttachments": [
                {
                    "buttons": [
                        {
                            "text": "2-15 (Wed)",
                            "value": "Wednesday, February 15, 2017"
                        },
```

```
                        {
                            "text": "2-17 (Fri)",
                            "value": "Friday, February 17, 2017"
                        },
                        {
                            "text": "2-20 (Mon)",
                            "value": "Monday, February 20, 2017"
                        },
                        {
                            "text": "2-21 (Tue)",
                            "value": "Tuesday, February 21, 2017"
                        }
                    ],
                    "subTitle": "When would you like to schedule your root
 canal?",
                    "title": "Specify Date"
                }
            ],
            "version": 1,
            "contentType": "application/vnd.amazonaws.card.generic"
        },
        "slots": {
            "AppointmentType": "root canal",
            "Date": null,
            "Time": null
        },
        "type": "ElicitSlot",
        "message": {
            "content": "We do not have any availability on that date, is there
 another day which works for you?",
            "contentType": "PlainText"
        }
    },
    "sessionAttributes": {
     "bookingMap": "{\"2017-02-16\": []}"
   }
}
```

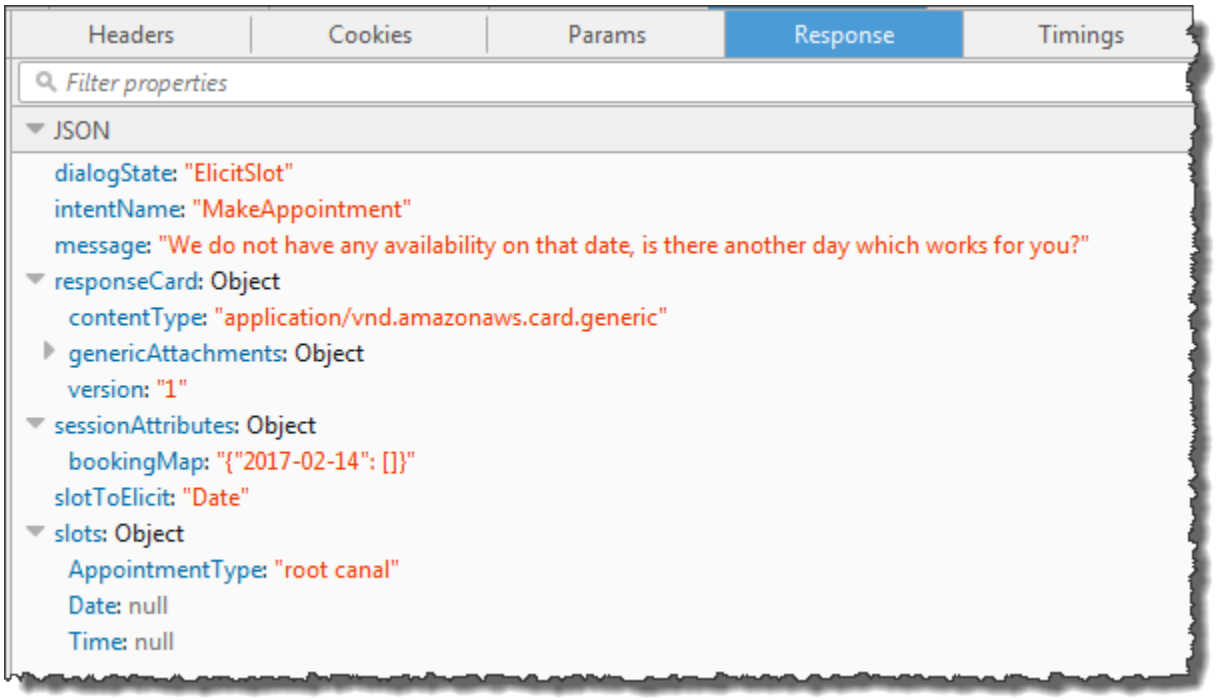
Sekali lagi, respon termasuk `dialogAction` dan `sessionAttributes` bidang. Antara lain, `dialogAction` mengembalikan bidang berikut:

- `dialogAction`bidang:

- `type`- Fungsi Lambda menetapkan nilai ini `ElicitSlot` dan mengatur ulang bidang ke. `slotToElicit` `Date` Fungsi Lambda juga menyediakan yang tepat `message` untuk disampaikan kepada pengguna.
- `responseCard`- Mengembalikan daftar nilai untuk `Date` slot.

- `sessionAttributes`- Kali ini fungsi Lambda menyertakan atribut `bookingMap` session. Nilainya adalah tanggal janji temu yang diminta dan janji temu yang tersedia (objek kosong menunjukkan bahwa tidak ada janji yang tersedia).

d. Amazon Lex memperhatikan `dialogAction.type` dan mengembalikan respons berikut ke klien yang menyertakan informasi dari respons fungsi Lambda.



Klien menampilkan pesan: Kami tidak memiliki ketersediaan pada tanggal itu, apakah ada hari lain yang sesuai untuk Anda? dan kartu respons (jika klien mendukung kartu respons).

4. Pengguna: Tergantung pada klien, pengguna memiliki dua opsi:

- Jika kartu respons ditampilkan, pilih 2-15 (Rabu) atau ketik**`Wednesday`**.
- Jika klien tidak mendukung kartu respons, ketik**`Wednesday`**.

a. Klien mengirimkan `PostText` permintaan berikut ke Amazon Lex:

```
POST /bot/BookTrip/alias/$LATEST/user/bijt6rovckwecnzesbthrr1d7lv3ja3n/text
"Content-Type":"application/json"
```

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

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

> **Note**
>
> Klien Facebook Messenger tidak menetapkan atribut sesi apa pun. Jika Anda ingin mempertahankan status sesi di antara permintaan, Anda harus melakukannya di fungsi Lambda. Dalam aplikasi nyata, Anda mungkin perlu mempertahankan atribut sesi ini dalam database backend.

b. Amazon Lex memanggil fungsi Lambda untuk validasi data pengguna dengan mengirimkan peristiwa berikut sebagai parameter:

```
{
    "currentIntent": {
        "slots": {
            "AppointmentType": "root canal",
            "Date": "2017-02-15",
            "Time": null
        },
        "name": "MakeAppointment",
        "confirmationStatus": "None"
    },
    "bot": {
        "alias": null,
        "version": "$LATEST",
        "name": "ScheduleAppointment"
    },
    "userId": "u3fpr9gghj02zts7y5tpq5mm4din2xqy",
    "invocationSource": "DialogCodeHook",
    "outputDialogMode": "Text",
    "messageVersion": "1.0",
    "sessionAttributes": {
    }
}
```

Amazon Lex diperbarui `currentIntent.slots` dengan mengatur `Date` slot untuk`2017-02-15`.

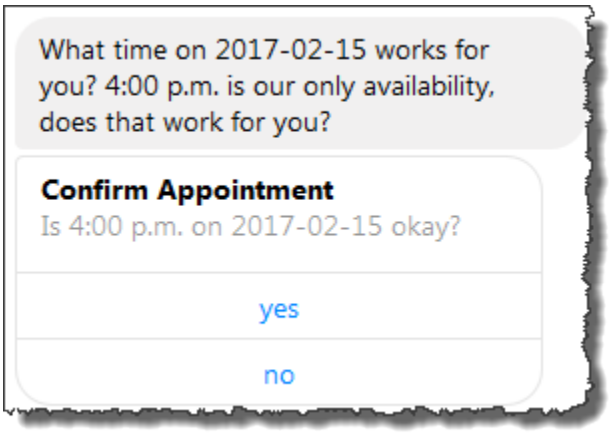
c. Fungsi Lambda memvalidasi input pengguna dan mengembalikan respons berikut ke Amazon Lex, mengarahkannya untuk mendapatkan nilai untuk waktu janji temu.

```
{
    "dialogAction": {
        "slots": {
            "AppointmentType": "root canal",
            "Date": "2017-02-15",
            "Time": "16:00"
        },
        "message": {
            "content": "What time on 2017-02-15 works for you? 4:00 p.m. is our
 only availability, does that work for you?",
            "contentType": "PlainText"
        },
        "type": "ConfirmIntent",
        "intentName": "MakeAppointment",
        "responseCard": {
            "genericAttachments": [
                {
                    "buttons": [
                        {
                            "text": "yes",
                            "value": "yes"
                        },
                        {
                            "text": "no",
                            "value": "no"
                        }
                    ],
                    "subTitle": "Is 4:00 p.m. on 2017-02-15 okay?",
                    "title": "Confirm Appointment"
                }
            ],
            "version": 1,
            "contentType": "application/vnd.amazonaws.card.generic"
        }
    },
    "sessionAttributes": {
        "bookingMap": "{\"2017-02-15\": [\"10:00\", \"16:00\", \"16:30\"]}"
```

```
    }
}
```

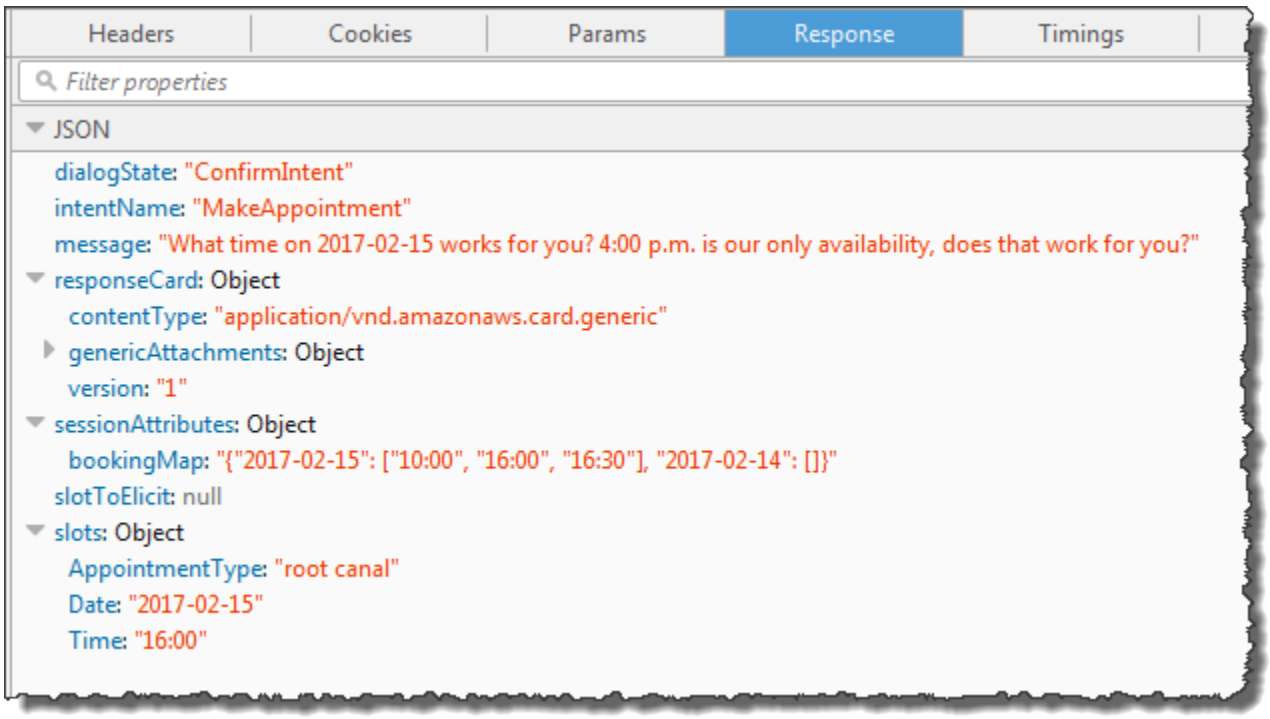
Sekali lagi, respon termasuk `dialogAction` dan `sessionAttributes` bidang. Antara lain, `dialogAction` mengembalikan bidang berikut:

- `dialogAction`bidang:
  - `type`— `Lambda` Fungsi menetapkan nilai ini`ConfirmIntent`, mengarahkan Amazon Lex untuk mendapatkan konfirmasi pengguna dari waktu janji yang disarankan dalam`message`.
  - `responseCard`- Mengembalikan daftar ya/tidak ada nilai bagi pengguna untuk memilih dari. Jika klien mendukung kartu respon, itu akan menampilkan kartu respon, seperti yang ditunjukkan pada contoh berikut:



  - `sessionAttributes`- Fungsi Lambda menetapkan atribut `bookingMap` sesi dengan nilainya ditetapkan ke tanggal janji temu dan janji temu yang tersedia pada tanggal tersebut. Dalam contoh ini, ini adalah janji temu 30 menit. Untuk saluran akar yang membutuhkan waktu satu jam, hanya jam 4 sore yang bisa dipesan.

d. Seperti yang ditunjukkan `dialogAction.type` dalam respons fungsi Lambda, Amazon Lex mengembalikan respons berikut ke klien:



Klien menampilkan pesan: Jam berapa 2017-02-15 bekerja untuk Anda? 4:00 sore adalah satu-satunya ketersediaan kami, apakah itu bekerja untuk Anda?

5. Pengguna: Pilih**yes**.

   Amazon Lex memanggil fungsi Lambda dengan data peristiwa berikut. Karena pengguna menjawab**yes**, Amazon Lex menetapkan `confirmationStatus` ke`Confirmed`, dan menetapkan `Time` bidang `currentIntent.slots` ke`4 p.m.`.

```
{
    "currentIntent": {
        "slots": {
            "AppointmentType": "root canal",
            "Date": "2017-02-15",
            "Time": "16:00"
        },
        "name": "MakeAppointment",
        "confirmationStatus": "Confirmed"
    },
    "bot": {
        "alias": null,
        "version": "$LATEST",
```

```
        "name": "ScheduleAppointment"
    },
    "userId": "u3fpr9gghj02zts7y5tpq5mm4din2xqy",
    "invocationSource": "FulfillmentCodeHook",
    "outputDialogMode": "Text",
    "messageVersion": "1.0",
    "sessionAttributes": {
   }
}
```

Karena `confirmationStatus` dikonfirmasi, fungsi Lambda memproses maksud (buku janji gigi) dan mengembalikan respons berikut ke Amazon Lex:

```
{
    "dialogAction": {
        "message": {
            "content": "Okay, I have booked your appointment.  We will see you at
 4:00 p.m. on 2017-02-15",
            "contentType": "PlainText"
        },
        "type": "Close",
        "fulfillmentState": "Fulfilled"
    },
    "sessionAttributes": {
        "formattedTime": "4:00 p.m.",
        "bookingMap": "{\"2017-02-15\": [\"10:00\"]}"
    }
}
```

Perhatikan hal berikut:

- Fungsi Lambda telah memperbarui file. `sessionAttributes`
- `dialogAction.type`diatur ke`Close`, yang mengarahkan Amazon Lex untuk tidak mengharapkan respons pengguna.
- `dialogAction.fulfillmentState`diatur ke`Fulfilled`, menunjukkan bahwa maksud berhasil dipenuhi.

Klien menampilkan pesan: Oke, saya telah memesan janji temu Anda. Kami akan melihat Anda di 4:00 p.m. pada 2017-02-15.

Buku Perjalanan

Contoh ini menggambarkan pembuatan bot yang dikonfigurasi untuk mendukung beberapa intent. Contoh ini juga menggambarkan bagaimana Anda dapat menggunakan atribut sesi untuk berbagi informasi lintas-maksud. Setelah membuat bot, Anda menggunakan klien uji di konsol Amazon Lex untuk menguji bot (BookTrip). Klien menggunakan operasi APIPostText runtime untuk mengirim permintaan ke Amazon Lex untuk setiap input pengguna.

BookTrip Bot dalam contoh ini dikonfigurasi dengan dua intent (BookHotel dan BookCar). Misalnya, misalkan pengguna memesan hotel terlebih dahulu. Selama interaksi, pengguna memberikan informasi seperti tanggal check-in, lokasi, dan jumlah malam. Setelah intent terpenuhi, klien dapat mempertahankan informasi ini menggunakan atribut sesi. Untuk informasi selengkapnya tentang atribut sesi, lihatPostText.

Sekarang anggaplah pengguna terus memesan mobil. Menggunakan informasi yang diberikan pengguna dalam BookHotel maksud sebelumnya (yaitu, kota tujuan, dan tanggal check-in dan check-out), pengait kode (fungsi Lambda) yang Anda konfigurasikan untuk menginisialisasi dan memvalidasi BookCar maksud, menginisialisasi data slot untuk BookCar maksud (yaitu, tujuan, kota penjemputan, tanggal pengambilan, dan tanggal pengembalian). Ini menggambarkan bagaimana berbagi informasi lintas-maksud memungkinkan Anda untuk membangun bot yang dapat terlibat dalam percakapan dinamis dengan pengguna.

Dalam contoh ini, kami menggunakan atribut sesi berikut. Hanya klien dan fungsi Lambda yang dapat mengatur dan memperbarui atribut sesi. Amazon Lex hanya meneruskan hal tersebut di antara klien dan fungsi Lambda. Amazon Lex tidak mempertahankan atau memodifikasi atribut sesi apa pun.

- `currentReservation`- Berisi data slot untuk reservasi yang sedang berlangsung dan informasi relevan lainnya. Misalnya, berikut ini adalah permintaan sampel dari klien ke Amazon Lex. Ini menunjukkan atribut`currentReservation` sesi di bodi permintaan.

```
POST /bot/BookTrip/alias/$LATEST/user/wch89kjqcpkds8seny7dly5x3otq68j3/text
"Content-Type":"application/json"
"Content-Encoding":"amz-1.0"

{
   "inputText":"Chicago",
   "sessionAttributes":{
       "currentReservation":"{\"ReservationType\":\"Hotel\",
```

```
                                \"Location\":\"Moscow\",
                                \"RoomType\":null,
                                \"CheckInDate\":null,
                                \"Nights\":null}"
    }
}
```

- `lastConfirmedReservation`- Berisi informasi serupa untuk maksud sebelumnya, jika ada. Misalnya, jika pengguna memesan hotel dan kemudian sedang dalam proses pemesanan mobil, atribut sesi ini menyimpan data slot untuk BookHotel maksud sebelumnya.

- `confirmationContext`- Fungsi Lambda menyetel ini`AutoPopulate` saat mengisi ulang beberapa data slot berdasarkan data slot dari reservasi sebelumnya (jika ada). Hal ini memungkinkan berbagi informasi cross-intent. Misalnya, jika pengguna sebelumnya memesan hotel dan sekarang ingin memesan mobil, Amazon Lex dapat meminta pengguna untuk mengonfirmasi (atau menolak) bahwa mobil tersebut dipesan untuk kota dan tanggal yang sama dengan reservasi hotel mereka

Dalam latihan ini Anda menggunakan cetak biru untuk membuat bot Amazon Lex dan fungsi Lambda. Untuk informasi selengkapnya tentang cetak biru, lihat Amazon Lex danAWS LambdaCetak Biru.

Langkah Selanjutnya

Langkah 1: Tinjau Cetak Biru yang Digunakan dalam Latihan ini

Langkah 1: Tinjau Cetak Biru yang Digunakan dalam Latihan ini

Topik

- Ikhtisar Cetak Biru Bot (BookTrip)
- Ikhtisar Cetak Biru Fungsi Lambda (lex-book-trip-python)

Ikhtisar Cetak Biru Bot (BookTrip)

Blueprint (BookTrip) yang Anda gunakan untuk membuat bot menyediakan prekonfigurasi berikut:

- Jenis slot - Dua jenis slot khusus:
  - `RoomTypes`dengan nilai pencacahan:`kingqueen`,, dan`deluxe`, untuk digunakan dalam`BookHotel` maksud.
  - `CarTypes`dengan nilai pencacahan:`economy`,`standard`,`midsizefull size`,`luxury`,`minivan`, dan, untuk digunakan dalam`BookCar` maksud.

- Maksud 1 (BookHotel) - Hal ini telah dikonfigurasi sebagai berikut:
  - Slot yang telah dikonfigurasi
    - `RoomType`, dari jenis slot`RoomTypes` khusus
    - `Location`, dari jenis slot`AMAZON.US_CITY` built-in
    - `CheckInDate`, dari jenis slot`AMAZON.DATE` built-in
    - `Nights`, dari jenis slot`AMAZON.NUMBER` built-in
  - Ucapan yang telah dikonfigurasi
    - “Pesan hotel”
    - “Saya ingin membuat reservasi hotel”
    - “Pesan {Malam} menginap di {Lokasi}”

    Jika pengguna mengucapkan salah satu dari ini, Amazon Lex menentukan`BookHotel` bahwa maksud dan kemudian meminta pengguna untuk data slot.
  - Petunjuk yang telah dikonfigurasi
    - Prompt untuk`Location` slot - “Kota apa yang akan Anda tinggali?”
    - Prompt untuk`CheckInDate` slot - “Hari apa Anda ingin check-in?”
    - Prompt untuk`Nights` slot - “Berapa malam Anda akan tinggal?”
    - Prompt untuk`RoomType` slot - “Jenis kamar apa yang Anda inginkan, ratu, raja, atau deluxe?”
    - Pernyataan konfirmasi - “Oke, saya meminta Anda untuk menginap malam {Malam} di {Lokasi} mulai {CheckInDate}. Haruskah saya memesan reservasi?”
    - Penolakan - “Oke, saya telah membatalkan reservasi Anda dalam proses.”

- Maksud 2 (BookCar) - Hal ini telah dikonfigurasi sebagai berikut:
  - Slot yang telah dikonfigurasi
    - `PickUpCity`, dari tipe`AMAZON.US_CITY` bawaan

  - `PickUpDate`, dari tipe`AMAZON.DATE` bawaan
  - `ReturnDate`, dari tipe`AMAZON.DATE` bawaan
  - `DriverAge`, dari tipe`AMAZON.NUMBER` bawaan
  - `CarType`, dari jenis`CarTypes` kustom
- Ucapan yang telah dikonfigurasi
  - “Pesan mobil”
  - “Pesan mobil”
  - “Buat reservasi mobil”

  Jika pengguna mengucapkan salah satu dari ini, Amazon Lex menentukan BookCar adalah maksud dan kemudian meminta pengguna untuk data slot.
- Petunjuk yang telah dikonfigurasi
  - Prompt untuk`PickUpCity` slot — “Di kota apa Anda perlu menyewa mobil?”
  - Prompt untuk`PickUpDate` slot - “Hari apa Anda ingin memulai sewa Anda?"”
  - Prompt untuk`ReturnDate` slot - “Hari apa Anda ingin mengembalikan mobil ini?”
  - Prompt untuk`DriverAge` slot — “Berapa umur pengemudi untuk sewa ini?”
  - Prompt untuk`CarType` slot — “Apa jenis mobil yang ingin Anda sewa? Pilihan kami yang paling populer adalah ekonomi, menengah, dan mewah”
  - Pernyataan konfirmasi - “Oke, saya memiliki Anda turun untuk {CarType} sewa di {PickUpCity} dari {PickUpDate} ke {ReturnDate}. Haruskah saya memesan reservasi?”
  - Penolakan - “Oke, saya telah membatalkan reservasi Anda dalam proses.”

Ikhtisar Cetak Biru Fungsi Lambda (lex-book-trip-python)

Selain cetak biru bot,AWS Lambda sediakan cetak biru (lex-book-trip-python) yang dapat Anda gunakan sebagai pengait kode dengan cetak biru bot. Untuk daftar cetak biru bot dan cetak biru fungsi Lambda yang sesuai, lihatAmazon Lex danAWS LambdaCetak Biru.

Saat Anda membuat bot menggunakan BookTrip cetak biru, Anda memperbarui konfigurasi intent (BookCar dan BookHotel) dengan menambahkan fungsi Lambda ini sebagai pengait kode untuk inisialisasi/validasi input data pengguna dan pemenuhan intent.

Kode fungsi Lambda ini disediakan menampilkan percakapan dinamis menggunakan informasi yang diketahui sebelumnya (disimpan dalam atribut sesi) tentang pengguna untuk menginisialisasi nilai slot untuk maksud. Untuk informasi selengkapnya, lihat Mengelola Konteks Percakapan.

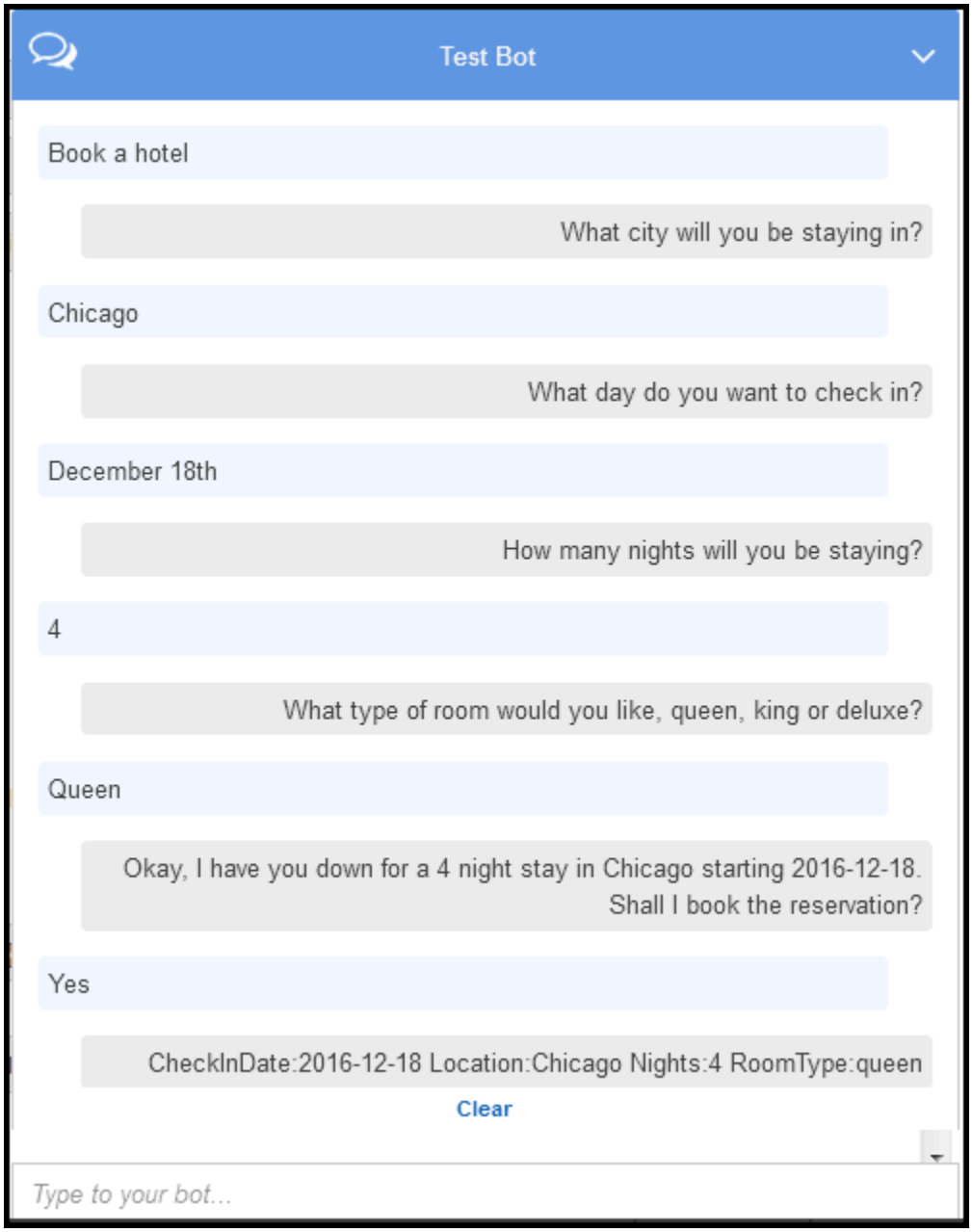
Langkah Selanjutnya

Langkah 2: Membuat Amazon Lex Bot

Langkah 2: Membuat Amazon Lex Bot

Dalam bagian ini, Anda membuat bot Amazon Lex (BookTrip).

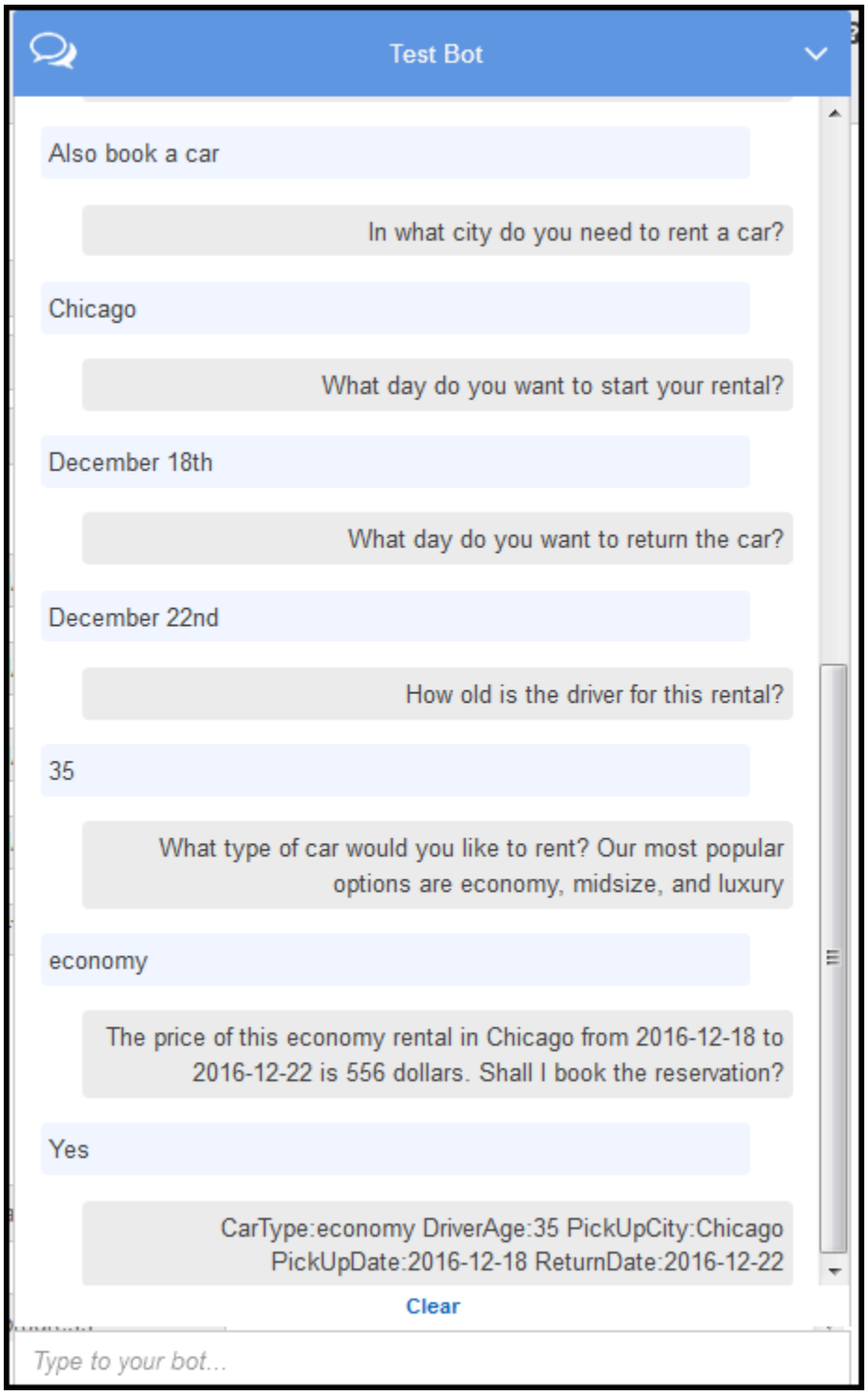
1. Masuk keAWS Management Console dan buka konsol Amazon Lex di https://console.aws.amazon.com/lex/.
2. Pada halaman Bot, pilih Buat.
3. Pada halaman Buat bot Lex Anda,
   - Pilih BookTripcetak biru.
   - Tinggalkan nama bot default (BookTrip).
4. Pilih Create (Buat). Konsol mengirimkan serangkaian permintaan ke Amazon Lex untuk membuat bot. Perhatikan hal berikut:
5. Konsol menunjukkan BookTrip bot. Pada tab Editor, tinjau detail maksud (BookCar dan BookHotel) yang telah dikonfigurasikan sebelumnya.
6. Uji bot di jendela pengujian. Gunakan yang berikut untuk terlibat dalam percakapan uji dengan bot Anda:



Dari input pengguna awal (“Pesan hotel”), Amazon Lex menyimpulkan intent (BookHotel). Bot kemudian menggunakan petunjuk yang telah dikonfigurasi sebelumnya dalam maksud ini untuk mendapatkan data slot dari pengguna. Setelah pengguna memberikan semua data slot, Amazon Lex mengembalikan respons kembali ke klien dengan pesan yang mencakup semua input pengguna sebagai pesan. Klien menampilkan pesan dalam respon seperti yang ditunjukkan.

```
CheckInDate:2016-12-18 Location:Chicago Nights:5 RoomType:queen
```

Sekarang Anda melanjutkan percakapan dan mencoba memesan mobil dalam percakapan berikut.



Perhatikan bahwa,

- Tidak ada validasi data pengguna saat ini. Misalnya, Anda dapat menyediakan kota mana pun untuk memesan hotel.
- Anda memberikan beberapa informasi yang sama lagi (tujuan, kota penjemputan, tanggal penjemputan, dan tanggal kembali) untuk memesan mobil. Dalam percakapan dinamis, bot Anda harus menginisialisasi beberapa informasi ini berdasarkan masukan sebelumnya pengguna yang disediakan untuk pemesanan hotel.

Pada bagian selanjutnya, Anda membuat fungsi Lambda untuk melakukan beberapa validasi data pengguna, dan inisialisasi menggunakan berbagi informasi cross-intent melalui atribut sesi. Kemudian Anda memperbarui konfigurasi maksud dengan menambahkan fungsi Lambda sebagai pengait kode untuk melakukan inisialisasi/validasi input pengguna dan memenuhi maksud.

Langkah Selanjutnya

Langkah 3: Membuat fungsi Lambda

Langkah 3: Membuat fungsi Lambda

Di bagian ini Anda membuat fungsi Lambda menggunakan blueprint (lex-book-trip-python) yang disediakan diAWS Lambda konsol. Anda juga menguji fungsi Lambda dengan memanggilnya menggunakan data kejadian sampel yang disediakan oleh konsol.

Fungsi Lambda ini ditulis dengan Python.

1. Masuk ke AWS Management Console dan buka konsol AWS Lambda di https://console.aws.amazon.com/lambda/.
2. Pilih Buat fungsi.
3. Pilih Gunakan cetak biru. Ketik**`lex`** untuk menemukan cetak biru, pilih`lex-book-trip-python` cetak birunya.
4. Pilih Konfigurasi fungsi Lambda sebagai berikut.
   - Ketik nama fungsi Lambda (`BookTripCodeHook`).
   - Untuk peran tersebut, pilih Buat peran baru lalu ketikkan nama untuk peran.
   - Biarkan nilai default lainnya.

5. Pilih Buat fungsi.
6. Jika Anda menggunakan lokal selain bahasa Inggris (AS) (id), perbarui nama maksud seperti yang dijelaskan diMemperbarui Cetak Biru untuk Lokal Tertentu.
7. Uji fungsi Lambda. Anda memanggil fungsi Lambda dua kali, menggunakan data sampel untuk pemesanan mobil dan pemesanan hotel.

    a. Pilih Konfigurasikan acara uji dari drop down Pilih acara uji.

    b. Pilih Amazon Lex Book Hotel dari daftar template acara Contoh.

    Contoh kejadian ini cocok dengan model permintaan/respons Amazon Lex. Untuk informasi selengkapnya, lihat Menggunakan Fungsi Lambda.

    c. Pilih Simpan dan uji.

    d. Verifikasikan bahwa fungsi Lambda berhasil berjalan. Tanggapan dalam hal ini cocok dengan model respons Amazon Lex.

    e. Ulangi langkahnya. Kali ini Anda memilih Amazon Lex Book Car dari daftar template acara Contoh. Fungsi Lambda memproses reservasi mobil.

Langkah Selanjutnya

Langkah 4: Tambahkan fungsi Lambda sebagai Kode Hook

Langkah 4: Tambahkan fungsi Lambda sebagai Kode Hook

Di bagian ini, Anda memperbarui konfigurasi BookCar dan BookHotel intent dengan menambahkan fungsi Lambda sebagai pengait kode untuk aktivitas inisialisasi/validasi dan pemenuhan. Pastikan Anda memilih versi $LATEST dari maksud karena Anda hanya dapat memperbarui versi $ TERBARU dari sumber daya Amazon Lex Anda.

1. Di konsol Amazon Lex, pilih BookTripbot.
2. Pada tab Editor, pilih BookHotelintent. Perbarui konfigurasi maksud sebagai berikut:

    a. Pastikan versi maksud (di samping nama maksud) adalah $LATEST.

    b. Tambahkan fungsi Lambda sebagai pengait kode inisialisasi dan validasi sebagai berikut:

    - Dalam Pilihan, pilih Inisialisasi dan kode validasi kait.

- Pilih fungsi Lambda Anda dari daftar.

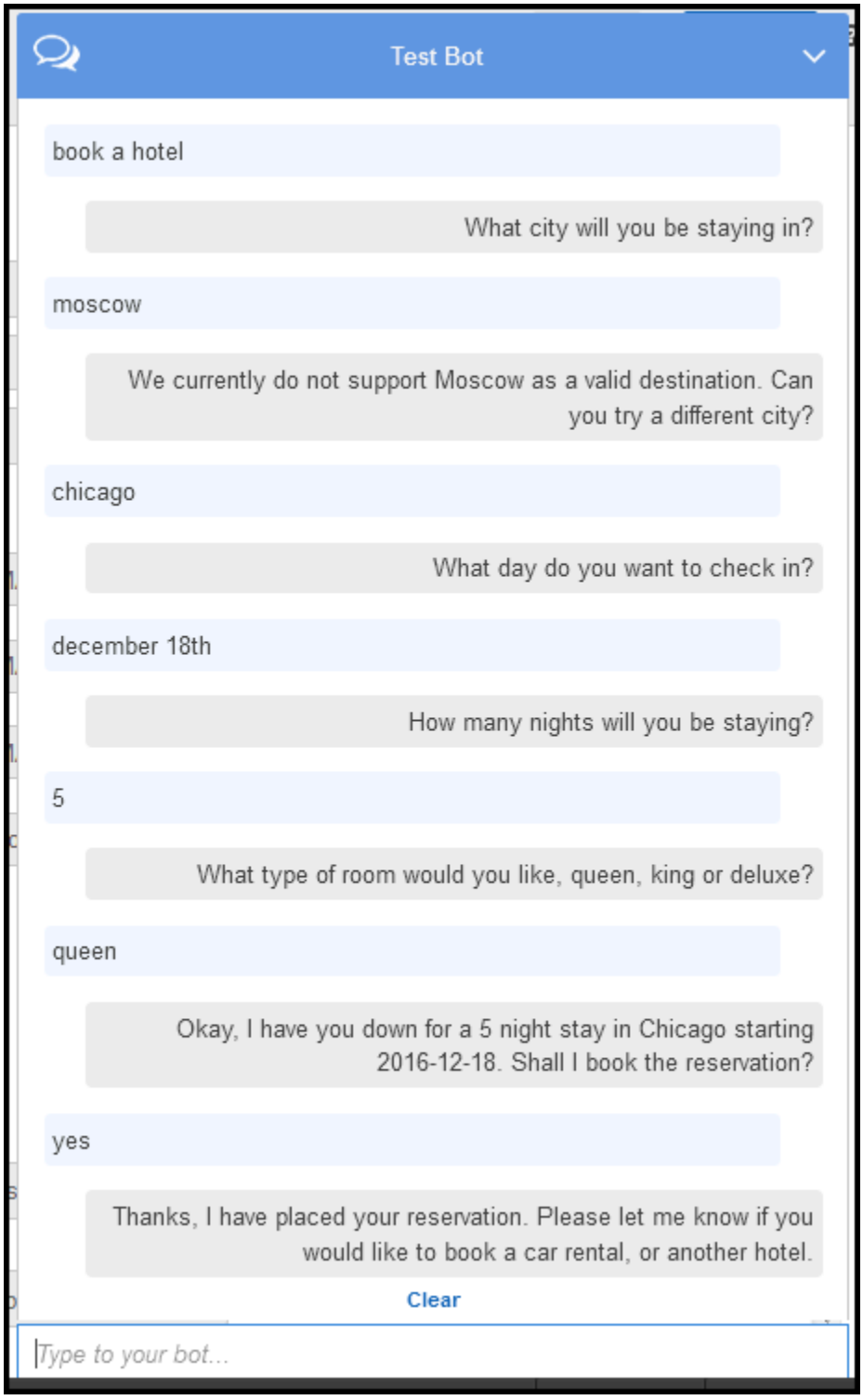
c. Tambahkan fungsi Lambda sebagai pengait kode pemenuhan sebagai berikut:

- Dalam Pemenuhan, pilih fungsi AWS Lambda.
- Pilih fungsi Lambda Anda dari daftar.
- Pilih Pesan selamat tinggal dan ketik pesan.

d. Pilih Save (Simpan).

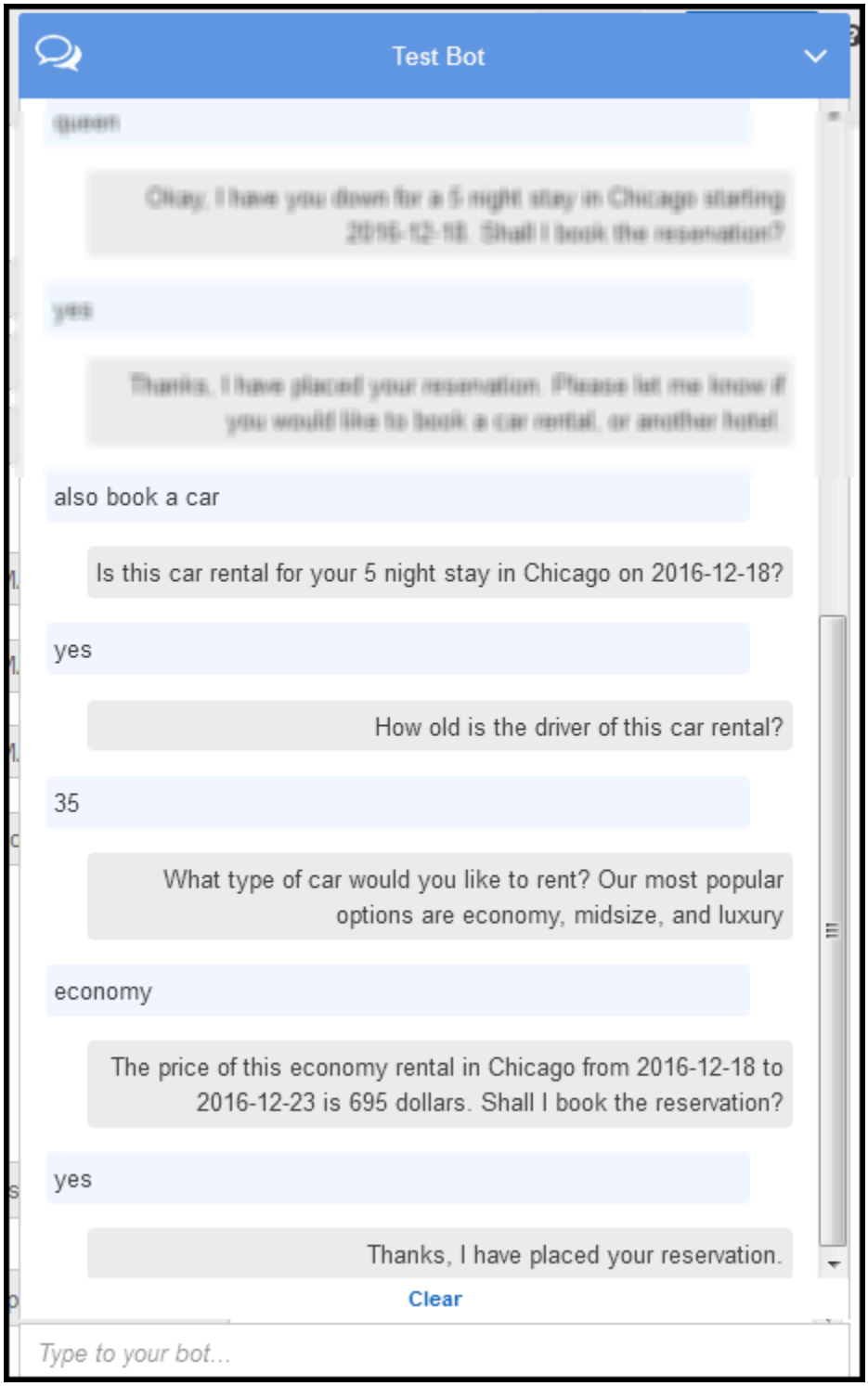
3. Pada tab Editor, pilih BookCar intent. Ikuti langkah sebelumnya untuk menambahkan fungsi Lambda Anda sebagai validasi dan pengait kode pemenuhan.

4. Pilih Build. Konsol mengirimkan serangkaian permintaan ke Amazon Lex untuk menyimpan konfigurasi.
5. Uji bot. Sekarang bahwa Anda memiliki fungsi Lambda melakukan inisialisasi, validasi data pengguna dan pemenuhan, Anda dapat melihat perbedaan dalam interaksi pengguna dalam percakapan berikut:



Untuk informasi selengkapnya tentang aliran data dari klien (konsol) ke Amazon Lex, dan dari Amazon Lex ke fungsi Lambda, lihatAliran Data: Pesan Maksud Hotel.

6. Lanjutkan percakapan dan pesan mobil seperti yang ditunjukkan pada citra berikut:



Saat Anda memilih untuk memesan mobil, klien (konsol) mengirimkan permintaan ke Amazon Lex yang menyertakan atribut sesi (dari percakapan sebelumnya BookHotel). Amazon Lex meneruskan informasi ini ke fungsi Lambda, yang kemudian menginisialisasi (yaitu, prepopulates) beberapa data BookCar slot (yaitu,, PickUpDate ReturnDate, dan PickUpCity).

> **Note**
>
> Ini menggambarkan bagaimana atribut sesi dapat digunakan untuk mempertahankan konteks di seluruh intent. Klien konsol menyediakan tautan Hapus di jendela pengujian yang dapat digunakan pengguna untuk menghapus atribut sesi sebelumnya.

Untuk informasi selengkapnya tentang aliran data dari klien (konsol) ke Amazon Lex, dan dari Amazon Lex ke fungsi Lambda, lihatAliran Data: Maksud Mobil Buku.

Rincian Alur Informasi

Dalam latihan ini, Anda terlibat dalam percakapan dengan BookTrip bot Amazon Lex menggunakan klien jendela pengujian yang disediakan di konsol Amazon Lex. Bagian ini menjelaskan hal-hal berikut:

- Aliran data antara klien dan Amazon Lex.

  Bagian ini mengasumsikan bahwa klien mengirimkan permintaan ke Amazon Lex menggunakan API`PostText` waktu proses dan menampilkan detail permintaan dan respons yang sesuai. Untuk informasi selengkapnya tentang`PostText` runtime API, lihatPostText.

  > **Note**
  >
  > Untuk contoh alur informasi antara klien dan Amazon Lex di mana klien menggunakan`PostContent` API, lihatLangkah 2a (Opsional): Tinjau Rincian Aliran Informasi Lisan (Konsol) .

- Aliran data antara Amazon Lex dan fungsi Lambda. Untuk informasi selengkapnya, lihat Lambda Fungsi Input Event dan Response Format.

Topik

- Aliran Data: Pesan Maksud Hotel
- Aliran Data: Maksud Mobil Buku

Aliran Data: Pesan Maksud Hotel

Bagian ini menjelaskan apa yang terjadi setelah setiap input pengguna.

1. User: “book a hotel”

   a. Klien (konsol) mengirimkanPostText permintaan berikut ke Amazon Lex:

   ```
   POST /bot/BookTrip/alias/$LATEST/user/wch89kjqcpkds8seny7dly5x3otq68j3/text
   "Content-Type":"application/json"
   "Content-Encoding":"amz-1.0"

   {
      "inputText":"book a hotel",
      "sessionAttributes":{}
   }
   ```

   URI permintaan dan badan memberikan informasi ke Amazon Lex:

   - Permintaan URI - Menyediakan nama bot (BookTrip), alias bot ($ TERBARU) dan nama pengguna. Trailing`text` menunjukkan bahwa itu adalah permintaan`PostText` API (dan bukan`PostContent`).
   - Permintaan tubuh - Termasuk input pengguna (`inputText`) dan kosong`sessionAttributes`. Awalnya, ini adalah objek kosong dan fungsi Lambda pertama menetapkan atribut sesi.

   b. Dari`inputText`, Amazon Lex mendeteksi maksud (BookHotel). Maksud ini dikonfigurasi dengan fungsi Lambda sebagai pengait kode untuk inisialisasi/validasi data pengguna. Oleh karena itu, Amazon Lex memanggil fungsi Lambda tersebut dengan meneruskan informasi berikut sebagai parameter peristiwa (lihatFormat Peristiwa Masukan):

   ```
   {
      "messageVersion":"1.0",
      "invocationSource":"DialogCodeHook",
      "userId":"wch89kjqcpkds8seny7dly5x3otq68j3",
   ```

```
   "sessionAttributes":{
   },
   "bot":{
      "name":"BookTrip",
      "alias":null,
      "version":"$LATEST"
   },
   "outputDialogMode":"Text",
   "currentIntent":{
      "name":"BookHotel",
      "slots":{
         "RoomType":null,
         "CheckInDate":null,
         "Nights":null,
         "Location":null
      },
      "confirmationStatus":"None"
   }
}
```

Selain informasi yang dikirim oleh klien, Amazon Lex juga menyertakan data tambahan berikut:

- `messageVersion`- Saat ini Amazon Lex hanya mendukung versi 1.0.
- `invocationSource`- Menunjukkan tujuan pemanggilan fungsi Lambda. Dalam hal ini, itu adalah untuk melakukan inisialisasi data pengguna dan validasi (saat ini Amazon Lex tahu bahwa pengguna belum menyediakan semua data slot untuk memenuhi maksud).
- `currentIntent`- Semua nilai slot diatur ke null.

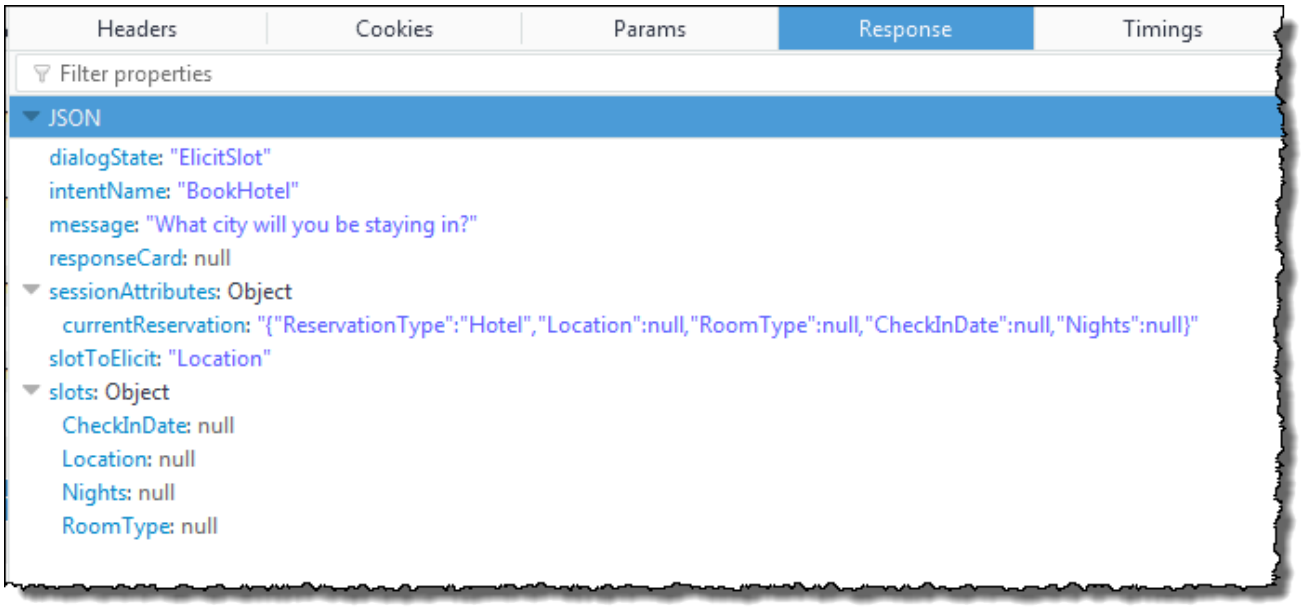
c. Pada saat ini, semua nilai slot yang nol. Tidak ada fungsi Lambda untuk memvalidasi. Fungsi Lambda mengembalikan respons berikut ke Amazon Lex. Untuk informasi selengkapnya tentang respons, lihatFormat Respons.

```
{
   "sessionAttributes":{
      "currentReservation":"{\"ReservationType\":\"Hotel\",\"Location\":null,
\"RoomType\":null,\"CheckInDate\":null,\"Nights\":null}"
   },
   "dialogAction":{
      "type":"Delegate",
      "slots":{
         "RoomType":null,
```

```
            "CheckInDate":null,
            "Nights":null,
            "Location":null
        }
    }
}
```

> **Note**
>
> - `currentReservation`- Fungsi Lambda mencakup atribut sesi ini. Nilainya adalah salinan informasi slot saat ini dan jenis reservasi.
>
>   Hanya fungsi Lambda dan klien yang dapat memperbarui atribut sesi ini. Amazon Lex hanya melewati nilai-nilai ini.
> - `dialogAction.type`- Dengan menetapkan nilai ini`Delegate`, fungsi Lambda mendelegasikan tanggung jawab untuk tindakan berikutnya ke Amazon Lex.
>
>   Jika fungsi Lambda mendeteksi sesuatu dalam validasi data pengguna, itu menginstruksikan Amazon Lex apa yang harus dilakukan selanjutnya.

d. Sesuai`dialogAction.type`, Amazon Lex memutuskan kursus berikutnya tindakan-mendapatkan data dari pengguna untuk`Location` slot. Ini memilih salah satu pesan prompt (“Kota apa yang akan Anda tinggali?”) untuk slot ini, sesuai dengan konfigurasi maksud, dan kemudian mengirimkan respons berikut kepada pengguna:



Atribut sesi diteruskan ke klien.

Klien membaca respons dan kemudian menampilkan pesan: “Kota apa yang akan Anda tinggali?”

2. Pengguna: “Moscow”

   a. Klien mengirimkan`PostText` permintaan berikut ke Amazon Lex (jeda baris ditambahkan untuk keterbacaan):

```
POST /bot/BookTrip/alias/$LATEST/user/wch89kjqcpkds8seny7dly5x3otq68j3/text
"Content-Type":"application/json"
"Content-Encoding":"amz-1.0"

{
   "inputText":"Moscow",
   "sessionAttributes":{
       "currentReservation":"{\"ReservationType\":\"Hotel\",
                              \"Location\":null,
                              \"RoomType\":null,
                              \"CheckInDate\":null,
                              \"Nights\":null}"
   }
}
```

   Selain`inputText`, klien menyertakan atribut`currentReservation` sesi yang sama yang diterima.

   b. Amazon Lex pertama menafsirkan`inputText` dalam konteks maksud saat ini (layanan ingat bahwa ia telah meminta pengguna tertentu untuk informasi tentang`Location` slot). Ini memperbarui nilai slot untuk maksud saat ini dan memanggil fungsi Lambda menggunakan acara berikut:

```
{
    "messageVersion": "1.0",
    "invocationSource": "DialogCodeHook",
    "userId": "wch89kjqcpkds8seny7dly5x3otq68j3",
    "sessionAttributes": {
        "currentReservation": "{\"ReservationType\":\"Hotel\",\"Location
\":null,\"RoomType\":null,\"CheckInDate\":null,\"Nights\":null}"
    },
    "bot": {
```

```
        "name": "BookTrip",
        "alias": null,
        "version": "$LATEST"
    },
    "outputDialogMode": "Text",
    "currentIntent": {
        "name": "BookHotel",
        "slots": {
            "RoomType": null,
            "CheckInDate": null,
            "Nights": null,
            "Location": "Moscow"
        },
        "confirmationStatus": "None"
    }
}
```

> **Note**
>
> - `invocationSource`terus menjadi`DialogCodeHook`. Pada langkah ini, kita hanya memvalidasi data pengguna.
> - Amazon Lex hanya meneruskan atribut session ke fungsi Lambda.
> - Untuk`currentIntent.slots`, Amazon Lex telah memperbarui`Location` slot untuk`Moscow`.

c. Fungsi Lambda melakukan validasi data pengguna dan menentukan lokasi`Moscow` yang tidak valid.

> **Note**
>
> Fungsi Lambda dalam latihan ini memiliki daftar sederhana kota yang valid dan tidak`Moscow` ada dalam daftar. Dalam aplikasi produksi, Anda mungkin menggunakan database back-end untuk mendapatkan informasi ini.

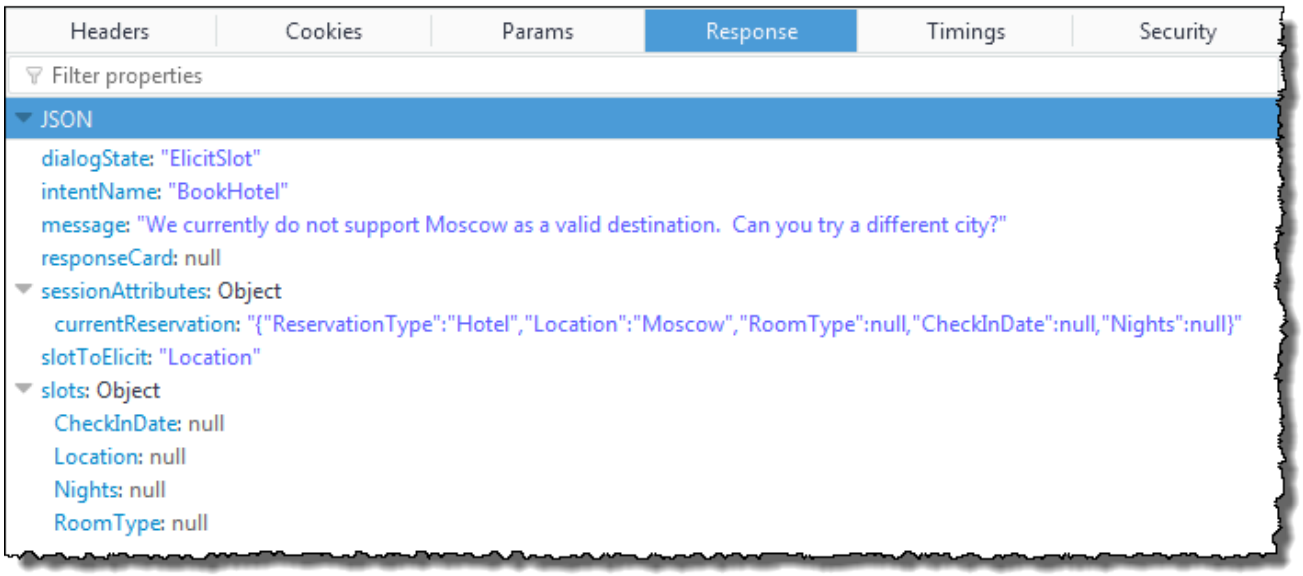
Ini me-reset nilai slot kembali ke null dan mengarahkan Amazon Lex untuk meminta pengguna lagi untuk nilai lain dengan mengirimkan respons berikut:

```
{
```

```
    "sessionAttributes": {
        "currentReservation": "{\"ReservationType\":\"Hotel\",\"Location\":
\"Moscow\",\"RoomType\":null,\"CheckInDate\":null,\"Nights\":null}"
    },
    "dialogAction": {
        "type": "ElicitSlot",
        "intentName": "BookHotel",
        "slots": {
            "RoomType": null,
            "CheckInDate": null,
            "Nights": null,
            "Location": null
        },
        "slotToElicit": "Location",
        "message": {
            "contentType": "PlainText",
            "content": "We currently do not support Moscow as a valid
 destination.  Can you try a different city?"
        }
    }
}
```

> **Note**
>
> - `currentIntent.slots.Location`diatur ulang ke null.
> - `dialogAction.type`diatur ke`ElicitSlot`, yang mengarahkan Amazon Lex untuk meminta pengguna lagi dengan memberikan yang berikut:
>   - `dialogAction.slotToElicit`- Slot untuk mendapatkan data dari pengguna.
>   - `dialogAction.message`- a`message` untuk menyampaikan kepada pengguna.

d. Amazon Lex memperhatikan`dialogAction.type` dan meneruskan informasi kepada klien dalam respons berikut:



Klien hanya menampilkan pesan: “Saat ini kami tidak mendukung Moskow sebagai tujuan yang valid. Bisakah kamu mencoba kota yang berbeda?”

3. Pengguna: “Chicago”

   a. Klien mengirimkan`PostText` permintaan berikut ke Amazon Lex:

```
POST /bot/BookTrip/alias/$LATEST/user/wch89kjqcpkds8seny7dly5x3otq68j3/text
"Content-Type":"application/json"
"Content-Encoding":"amz-1.0"

{
   "inputText":"Chicago",
   "sessionAttributes":{
       "currentReservation":"{\"ReservationType\":\"Hotel\",
                              \"Location\":\"Moscow\",
                              \"RoomType\":null,
                              \"CheckInDate\":null,
                              \"Nights\":null}"
   }
}
```

   b. Amazon Lex tahu konteksnya, bahwa itu memunculkan data untuk`Location` slot. Dalam konteks ini, ia tahu`inputText` nilai adalah untuk`Location` slot. Kemudian memanggil fungsi Lambda dengan mengirimkan acara berikut:

```
{
    "messageVersion": "1.0",
    "invocationSource": "DialogCodeHook",
    "userId": "wch89kjqcpkds8seny7dly5x3otq68j3",
    "sessionAttributes": {
        "currentReservation": "{\"ReservationType\":\"Hotel\",\"Location
\":Moscow,\"RoomType\":null,\"CheckInDate\":null,\"Nights\":null}"
    },
    "bot": {
        "name": "BookTrip",
        "alias": null,
        "version": "$LATEST"
    },
    "outputDialogMode": "Text",
    "currentIntent": {
        "name": "BookHotel",
        "slots": {
            "RoomType": null,
            "CheckInDate": null,
            "Nights": null,
            "Location": "Chicago"
        },
        "confirmationStatus": "None"
    }
}
```

Amazon Lex diperbaruicurrentIntent.slots dengan menetapkanLocation slot untukChicago.

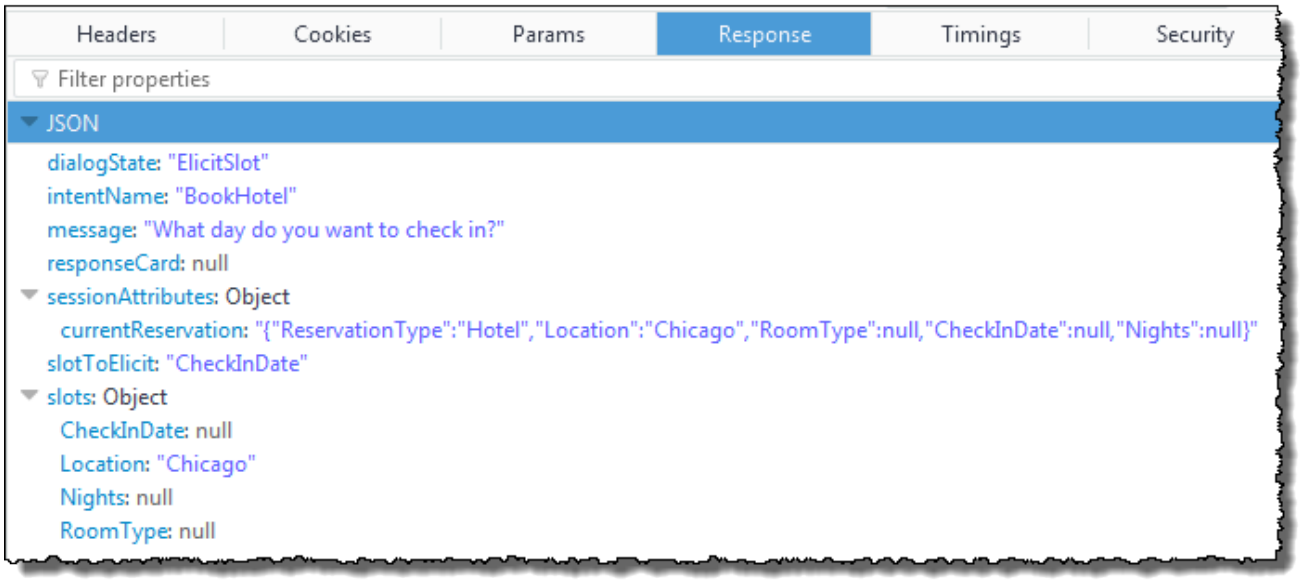
c. MenurutinvocationSource nilaiDialogCodeHook, fungsi Lambda melakukan validasi data pengguna. Ini mengakuiChicago sebagai nilai slot yang valid, memperbarui atribut sesi yang sesuai, dan kemudian mengembalikan respons berikut ke Amazon Lex.

```
{
    "sessionAttributes": {
        "currentReservation": "{\"ReservationType\":\"Hotel\",\"Location\":
\"Chicago\",\"RoomType\":null,\"CheckInDate\":null,\"Nights\":null}"
    },
    "dialogAction": {
        "type": "Delegate",
        "slots": {
            "RoomType": null,
```

```
            "CheckInDate": null,
            "Nights": null,
            "Location": "Chicago"
        }
    }
}
```

> **Note**
>
> - `currentReservation`- Fungsi Lambda memperbarui atribut sesi ini dengan menyetel`Location` to`Chicago`.
> - `dialogAction.type`- Diatur ke`Delegate`. Data pengguna valid, dan fungsi Lambda mengarahkan Amazon Lex untuk memilih tindakan berikutnya.

d. Menurut`dialogAction.type`, Amazon Lex memilih tindakan selanjutnya. Amazon Lex tahu bahwa ia membutuhkan lebih banyak data slot dan memilih slot yang tidak terisi berikutnya (`CheckInDate`) dengan prioritas tertinggi sesuai dengan konfigurasi maksud. Ini memilih salah satu pesan prompt (“Hari apa Anda ingin check-in?”) untuk slot ini sesuai dengan konfigurasi maksud dan kemudian mengirimkan respons berikut kembali ke klien:

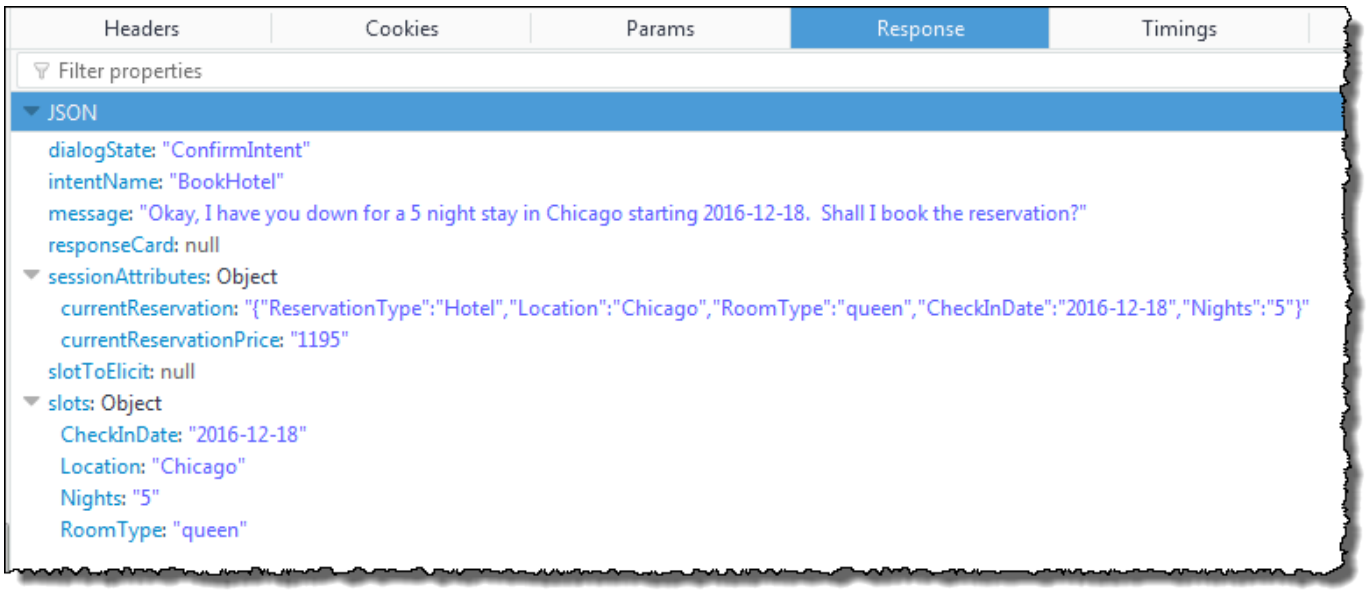


Klien menampilkan pesan: “Hari apa Anda ingin check-in?”

4. Interaksi pengguna terus berlanjut—pengguna menyediakan data, fungsi Lambda memvalidasi data, dan kemudian mendelegasikan tindakan selanjutnya ke Amazon Lex. Akhirnya pengguna menyediakan semua data slot, fungsi Lambda memvalidasi semua input pengguna, dan kemudian Amazon Lex mengakui itu memiliki semua data slot.

   > **Note**
   >
   > Dalam latihan ini, setelah pengguna menyediakan semua data slot, fungsi Lambda menghitung harga reservasi hotel dan mengembalikannya sebagai atribut sesi lainnya (`currentReservationPrice`).

   Pada titik ini, intent siap untuk dipenuhi, tetapi BookHotel intent dikonfigurasi dengan prompt konfirmasi yang memerlukan konfirmasi pengguna sebelum Amazon Lex dapat memenuhi maksud. Oleh karena itu, Amazon Lex mengirimkan pesan berikut kepada klien yang meminta konfirmasi sebelum memesan hotel:



   Klien menampilkan pesan: “Oke, saya telah Anda turun untuk 5 malam di Chicago mulai 2016-12-18. Haruskah saya memesan reservasi?”

5. Pengguna: “ya”

   a. Klien mengirimkan`PostText` permintaan berikut ke Amazon Lex:

```
POST /bot/BookTrip/alias/$LATEST/user/wch89kjqcpkds8seny7dly5x3otq68j3/text
"Content-Type":"application/json"
```

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

{
   "inputText":"Yes",
   "sessionAttributes":{
       "currentReservation":"{\"ReservationType\":\"Hotel\",
                              \"Location\":\"Chicago\",
                              \"RoomType\":\"queen\",
                              \"CheckInDate\":\"2016-12-18\",
                              \"Nights\":\"5\"}",
      "currentReservationPrice":"1195"
   }
}
```

b. Amazon Lex menafsirkan`inputText` dalam konteks mengkonfirmasikan maksud saat ini. Amazon Lex memahami bahwa pengguna ingin melanjutkan reservasi. Kali ini Amazon Lex memanggil fungsi Lambda untuk memenuhi maksud dengan mengirimkan acara berikut. Dengan mengatur`invocationSource` ke`FulfillmentCodeHook` dalam acara tersebut, ia mengirimkan ke fungsi Lambda. Amazon Lex juga menetapkan`confirmationStatus` untuk`Confirmed`.

```
{
    "messageVersion": "1.0",
    "invocationSource": "FulfillmentCodeHook",
    "userId": "wch89kjqcpkds8seny7dly5x3otq68j3",
    "sessionAttributes": {
        "currentReservation": "{\"ReservationType\":\"Hotel\",\"Location\":
\"Chicago\",\"RoomType\":\"queen\",\"CheckInDate\":\"2016-12-18\",\"Nights\":
\"5\"}",
        "currentReservationPrice": "956"
    },
    "bot": {
        "name": "BookTrip",
        "alias": null,
        "version": "$LATEST"
    },
    "outputDialogMode": "Text",
    "currentIntent": {
        "name": "BookHotel",
        "slots": {
            "RoomType": "queen",
            "CheckInDate": "2016-12-18",
            "Nights": "5",
```

```
            "Location": "Chicago"
        },
        "confirmationStatus": "Confirmed"
    }
}
```

> **Note**
>
> - `invocationSource`— Kali ini, Amazon Lex menetapkan nilai ini`FulfillmentCodeHook`, mengarahkan fungsi Lambda untuk memenuhi maksud.
> - `confirmationStatus`- Diatur ke`Confirmed`.

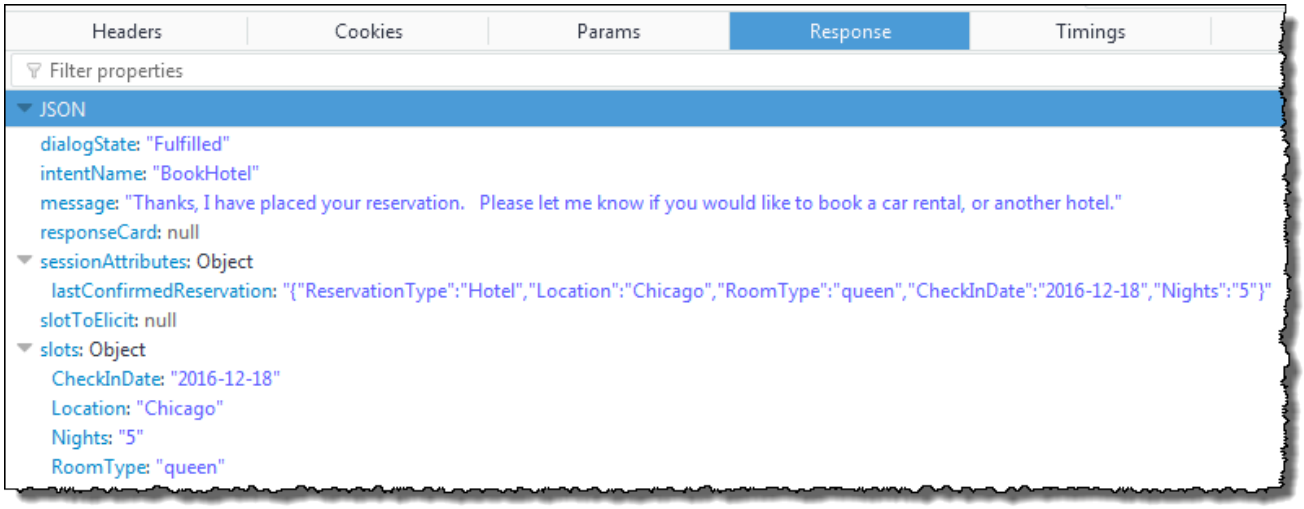
c. Kali ini, fungsi Lambda memenuhi BookHotel maksud, Amazon Lex menyelesaikan reservasi, dan kemudian mengembalikan respons berikut:

```
{
    "sessionAttributes": {
        "lastConfirmedReservation": "{\"ReservationType\":\"Hotel\",\"Location
\":\"Chicago\",\"RoomType\":\"queen\",\"CheckInDate\":\"2016-12-18\",\"Nights
\":\"5\"}"
    },
    "dialogAction": {
        "type": "Close",
        "fulfillmentState": "Fulfilled",
        "message": {
            "contentType": "PlainText",
            "content": "Thanks, I have placed your reservation.   Please let me
 know if you would like to book a car rental, or another hotel."
        }
    }
}
```

> **Note**
>
> - `lastConfirmedReservation`- Adalah atribut sesi baru yang ditambahkan fungsi Lambda (bukan`currentReservation`,`currentReservationPrice`).
> - `dialogAction.type`- Fungsi Lambda menetapkan nilai ini`Close`, menunjukkan bahwa Amazon Lex tidak mengharapkan respons pengguna.

- `dialogAction.fulfillmentState`- Diatur ke`Fulfilled` dan termasuk yang sesuai`message` untuk menyampaikan kepada pengguna.

d. Amazon Lex meninjau`fulfillmentState` dan mengirimkan respons berikut kepada klien:



> **Note**
>
> - `dialogState`- Amazon Lex menetapkan nilai ini ke`Fulfilled`.
> - `message`— Apakah pesan yang sama dengan fungsi Lambda yang disediakan.

Klien menampilkan pesan.

Aliran Data: Maksud Mobil Buku

BookTrip Bot dalam latihan ini mendukung dua maksud (BookHotel dan BookCar). Setelah memesan hotel, pengguna dapat melanjutkan percakapan untuk memesan mobil. Selama sesi belum habis waktu, dalam setiap permintaan berikutnya klien terus mengirim atribut sesi (dalam contoh ini,`lastConfirmedReservation`). Fungsi Lambda dapat menggunakan informasi ini untuk menginisialisasi data slot untuk BookCar maksud tersebut. Ini menunjukkan bagaimana Anda dapat menggunakan atribut sesi dalam berbagi data lintas-maksud.

Secara khusus, saat pengguna memilih BookCar maksud, fungsi Lambda menggunakan informasi yang relevan dalam atribut sesi untuk mengisi ulang slot (PickUpDate, ReturnDate, dan PickUpCity) untuk BookCar maksud tersebut.

> **Note**
>
> Konsol Amazon Lex menyediakan tautan Hapus yang dapat Anda gunakan untuk menghapus atribut sesi sebelumnya.

Ikuti langkah-langkah dalam prosedur ini untuk melanjutkan percakapan.

1. Pengguna: “juga memesan mobil”

   a. Klien mengirimkan`PostText` permintaan berikut ke Amazon Lex.

```
POST /bot/BookTrip/alias/$LATEST/user/wch89kjqcpkds8seny7dly5x3otq68j3/text
"Content-Type":"application/json"
"Content-Encoding":"amz-1.0"

{
   "inputText":"also book a car",
   "sessionAttributes":{
       "lastConfirmedReservation":""{\"ReservationType\":\"Hotel\",
                                    \"Location\":\"Chicago\",
                                    \"RoomType\":\"queen\",
                                    \"CheckInDate\":\"2016-12-18\",
                                    \"Nights\":\"5\"}"
   }
}
```

   Klien termasuk atribut`lastConfirmedReservation` session.

   b. Amazon Lex mendeteksi intent (BookCar) dari`inputText`. Maksud ini juga dikonfigurasi untuk memanggil fungsi Lambda untuk melakukan inisialisasi dan validasi data pengguna. Amazon Lex memanggil fungsi Lambda dengan peristiwa berikut:

```
{
    "messageVersion": "1.0",
    "invocationSource": "DialogCodeHook",
    "userId": "wch89kjqcpkds8seny7dly5x3otq68j3",
    "sessionAttributes": {
```

```
        "lastConfirmedReservation": "{\"ReservationType\":\"Hotel\",\"Location
\":\"Chicago\",\"RoomType\":\"queen\",\"CheckInDate\":\"2016-12-18\",\"Nights
\":\"5\"}"
    },
    "bot": {
        "name": "BookTrip",
        "alias": null,
        "version": "$LATEST"
    },
    "outputDialogMode": "Text",
    "currentIntent": {
        "name": "BookCar",
        "slots": {
            "PickUpDate": null,
            "ReturnDate": null,
            "DriverAge": null,
            "CarType": null,
            "PickUpCity": null
        },
        "confirmationStatus": "None"
    }
}
```

> **Note**
>
> - `messageVersion`- Saat ini Amazon Lex hanya mendukung versi 1.0.
> - `invocationSource`- Menunjukkan tujuan pemanggilan adalah untuk melakukan inisialisasi dan validasi data pengguna.
> - `currentIntent`- Ini termasuk nama maksud dan slot. Pada saat ini, semua nilai slot yang nol.

c. Fungsi Lambda pemberitahuan semua nilai slot null dengan apa-apa untuk memvalidasi. Namun, ia menggunakan atribut sesi untuk menginisialisasi beberapa nilai slot (`PickUpDate`,`ReturnDate`, and`PickUpCity`), dan kemudian mengembalikan respon berikut:

```
{
    "sessionAttributes": {
```

```
        "lastConfirmedReservation": "{\"ReservationType\":\"Hotel\",\"Location
\":\"Chicago\",\"RoomType\":\"queen\",\"CheckInDate\":\"2016-12-18\",\"Nights
\":\"5\"}",
        "currentReservation": "{\"ReservationType\":\"Car\",\"PickUpCity
\":null,\"PickUpDate\":null,\"ReturnDate\":null,\"CarType\":null}",
        "confirmationContext": "AutoPopulate"
    },
    "dialogAction": {
        "type": "ConfirmIntent",
        "intentName": "BookCar",
        "slots": {
            "PickUpCity": "Chicago",
            "PickUpDate": "2016-12-18",
            "ReturnDate": "2016-12-22",
            "CarType": null,
            "DriverAge": null
        },
        "message": {
            "contentType": "PlainText",
            "content": "Is this car rental for your 5 night stay in Chicago on
 2016-12-18?"
        }
    }
}
```

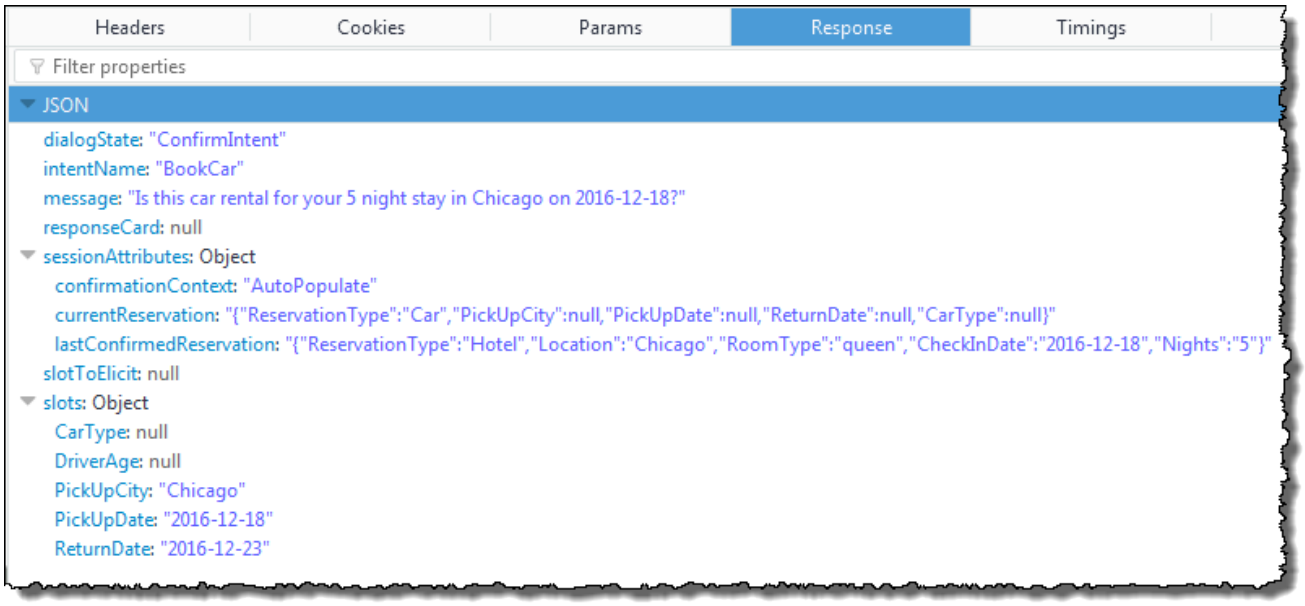
Note

- Selain itu`lastConfirmedReservation`, fungsi Lambda mencakup lebih banyak atribut sesi (`currentReservation`dan`confirmationContext`).
- `dialogAction.type`diatur ke`ConfirmIntent`, yang menginformasikan Amazon Lex bahwa jawaban ya, tidak ada yang diharapkan dari pengguna (confirmationContext diatur ke AutoPopulate, fungsi Lambda tahu bahwa ya/tidak ada balasan pengguna adalah untuk mendapatkan konfirmasi pengguna dari inisialisasi fungsi Lambda yang dilakukan (data slot yang diisi auto).

  Fungsi Lambda juga menyertakan dalam respons pesan informatif di`dialogAction.message` Amazon Lex untuk kembali ke klien.

> **Note**
>
> IstilahConfirmIntent (value of thedialogAction.type) tidak terkait dengan maksud bot apa pun. Dalam contoh, fungsi Lambda menggunakan istilah ini untuk mengarahkan Amazon Lex untuk mendapatkan jawaban yes/no dari pengguna.

d. MenurutdialogAction.type, Amazon Lex mengembalikan respon berikut kepada klien:



Klien menampilkan pesan: “Apakah ini sewa mobil untuk Anda 5 malam menginap di Chicago pada 2016-12-18?”

2. Pengguna: “ya”

a. Klien mengirimkanPostText permintaan berikut ke Amazon Lex.

```
POST /bot/BookTrip/alias/$LATEST/user/wch89kjqcpkds8seny7dly5x3otq68j3/text
"Content-Type":"application/json"
"Content-Encoding":"amz-1.0"

{
   "inputText":"yes",
   "sessionAttributes":{
      "confirmationContext":"AutoPopulate",
      "currentReservation":"{\"ReservationType\":\"Car\",
```

```
                              \"PickUpCity\":null,
                              \"PickUpDate\":null,
                              \"ReturnDate\":null,
                              \"CarType\":null}",
      "lastConfirmedReservation":"{\"ReservationType\":\"Hotel\",
                                   \"Location\":\"Chicago\",
                                   \"RoomType\":\"queen\",
                                   \"CheckInDate\":\"2016-12-18\",
                                   \"Nights\":\"5\"}"
   }
}
```

b. Amazon Lex membaca`inputText` dan tahu konteksnya (meminta pengguna untuk mengkonfirmasi populasi auto). Amazon Lex memanggil fungsi Lambda dengan mengirimkan peristiwa berikut:

```
{
    "messageVersion": "1.0",
    "invocationSource": "DialogCodeHook",
    "userId": "wch89kjqcpkds8seny7dly5x3otq68j3",
    "sessionAttributes": {
        "confirmationContext": "AutoPopulate",
        "currentReservation": "{\"ReservationType\":\"Car\",\"PickUpCity
\":null,\"PickUpDate\":null,\"ReturnDate\":null,\"CarType\":null}",
        "lastConfirmedReservation": "{\"ReservationType\":\"Hotel\",\"Location
\":\"Chicago\",\"RoomType\":\"queen\",\"CheckInDate\":\"2016-12-18\",\"Nights
\":\"5\"}"
    },
    "bot": {
        "name": "BookTrip",
        "alias": null,
        "version": "$LATEST"
    },
    "outputDialogMode": "Text",
    "currentIntent": {
        "name": "BookCar",
        "slots": {
            "PickUpDate": "2016-12-18",
            "ReturnDate": "2016-12-22",
            "DriverAge": null,
            "CarType": null,
            "PickUpCity": "Chicago"
        },
```

```
        "confirmationStatus": "Confirmed"
    }
}
```

Karena pengguna menjawab Ya, Amazon Lex menetapkan`confirmationStatus` untuk`Confirmed`.

c. Dari`confirmationStatus`, fungsi Lambda tahu bahwa nilai prepopulated benar. Fungsi Lambda melakukan hal-hal berikut:

- Memperbarui atribut`currentReservation` sesi ke nilai slot yang telah dihuni sebelumnya.
- Menetapkan`dialogAction.type` ke`ElicitSlot`
- Menetapkan`slotToElicit` nilai untuk`DriverAge`.

Tanggapan berikut dikirim:

```
{
    "sessionAttributes": {
        "currentReservation": "{\"ReservationType\":\"Car\",\"PickUpCity
\":\"Chicago\",\"PickUpDate\":\"2016-12-18\",\"ReturnDate\":\"2016-12-22\",
\"CarType\":null}",
        "lastConfirmedReservation": "{\"ReservationType\":\"Hotel\",\"Location
\":\"Chicago\",\"RoomType\":\"queen\",\"CheckInDate\":\"2016-12-18\",\"Nights
\":\"5\"}"
    },
    "dialogAction": {
        "type": "ElicitSlot",
        "intentName": "BookCar",
        "slots": {
            "PickUpDate": "2016-12-18",
            "ReturnDate": "2016-12-22",
            "DriverAge": null,
            "CarType": null,
            "PickUpCity": "Chicago"
        },
        "slotToElicit": "DriverAge",
        "message": {
            "contentType": "PlainText",
            "content": "How old is the driver of this car rental?"
        }
```

```
    }
}
```

d. Amazon Lex mengembalikan respons berikut:

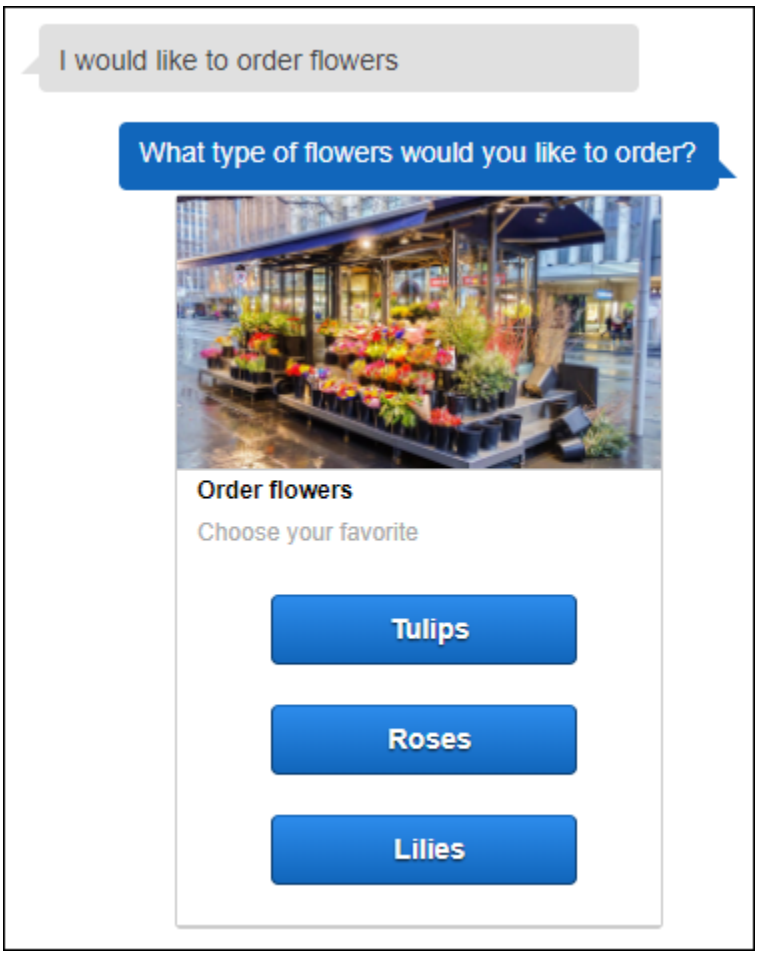
Klien menampilkan pesan “Berapa umur pengemudi sewa mobil ini?” dan percakapan berlanjut.

Menggunakan Kartu Respons

Dalam latihan ini, Anda memperpanjang Latihan Memulai 1 dengan menambahkan kartu respons. Anda membuat bot yang mendukung OrderFlowers maksud, dan kemudian memperbarui maksud dengan menambahkan kartu respons untuk`FlowerType` slot. Selain prompt berikut untuk`FlowerType` slot, pengguna dapat memilih jenis bunga dari kartu respons:

```
What type of flowers would you like to order?
```

Berikut ini adalah kartu responsnya:

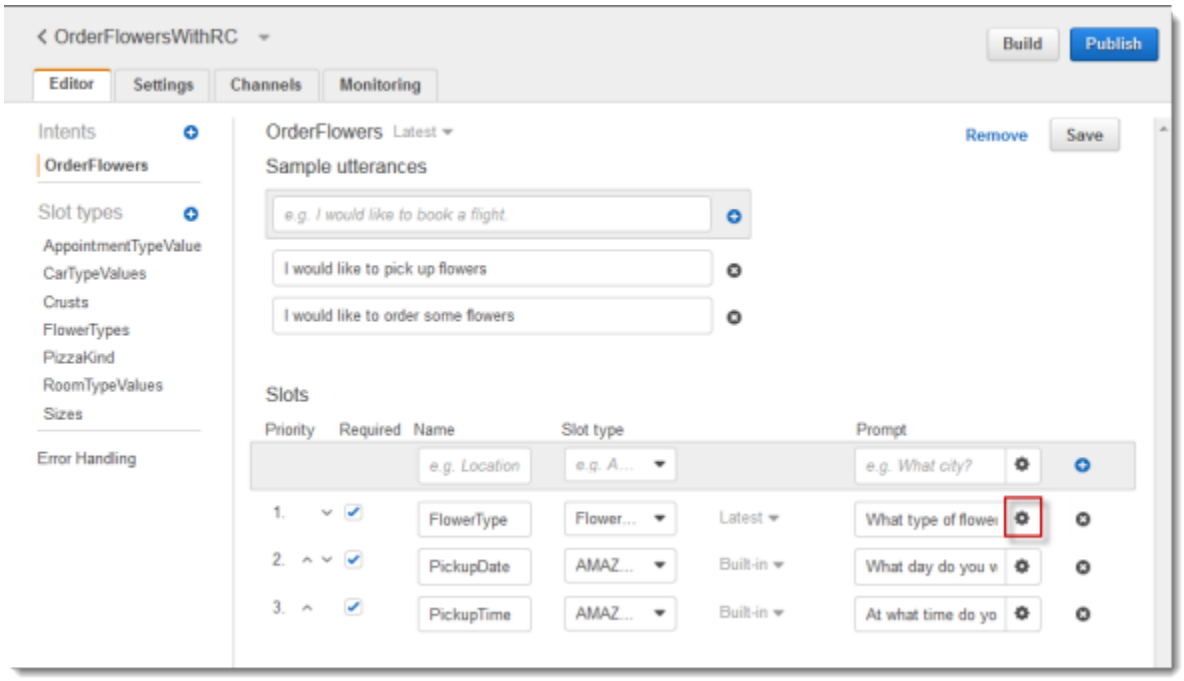


Pengguna bot dapat mengetik teks atau memilih dari daftar jenis bunga. Kartu respons ini dikonfigurasi dengan gambar, yang muncul di klien seperti yang ditunjukkan. Untuk informasi lebih lanjut tentang kartu respon, lihatKartu Respon.

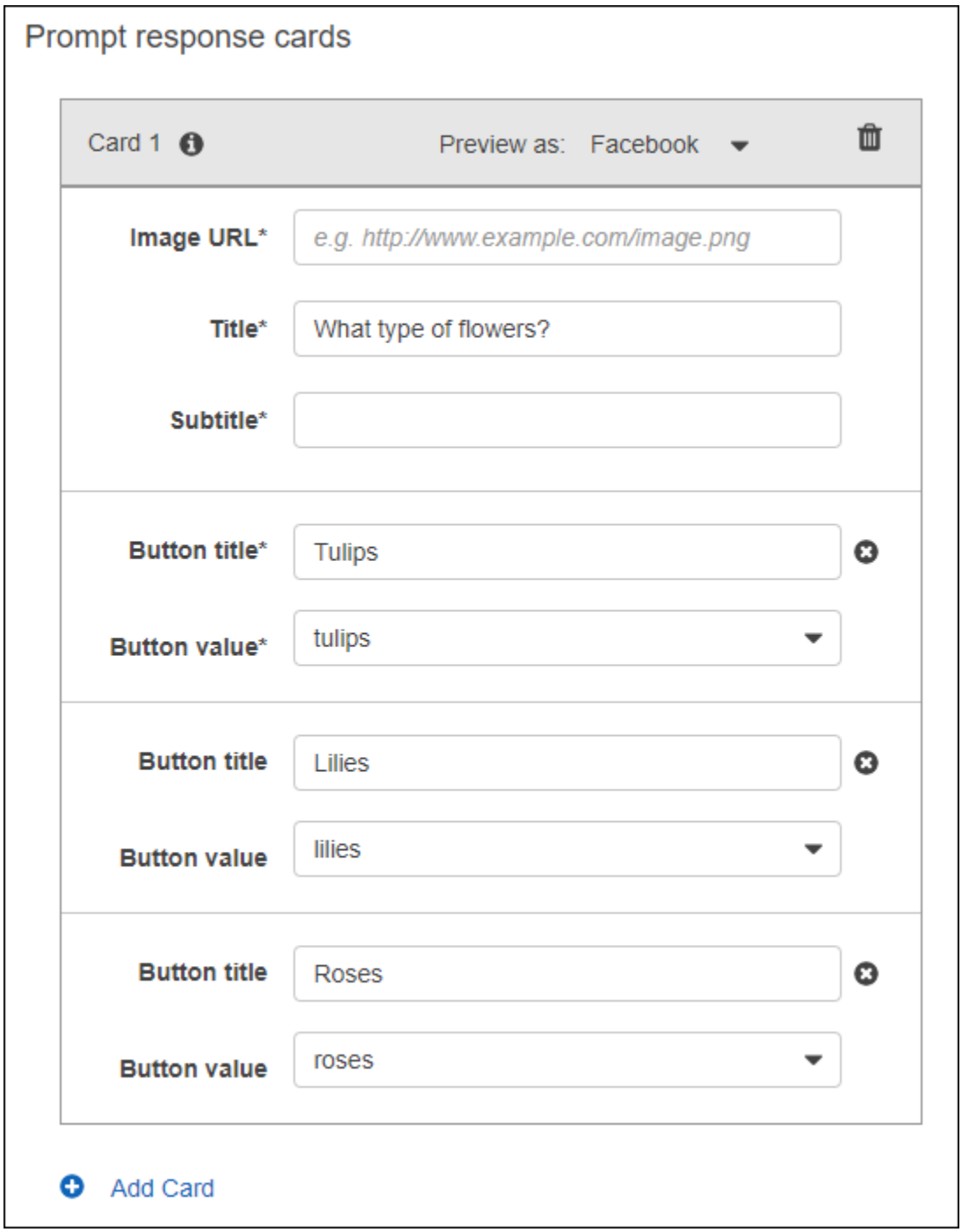
Untuk membuat dan menguji bot dengan kartu respons:

1. Ikuti Latihan Memulai 1 untuk membuat dan menguji OrderFlowers bot. Anda harus menyelesaikan langkah 1, 2, dan 3. Anda tidak perlu menambahkan fungsi Lambda untuk menguji kartu respon. Untuk petunjuk, lihat Latihan 1: Membuat Bot Amazon Lex Menggunakan Cetak Biru (Konsol).
2. Perbarui bot dengan menambahkan kartu respons, lalu publikasikan versi. Saat Anda mempublikasikan versi, tentukan alias (BETA) untuk menunjuk ke versi tersebut.
   a. Di konsol Amazon Lex, pilih bot Anda.
   b. Pilih`OrderFlowers` maksudnya.

c. Pilih ikon pengaturan gigi di sebelah “Apa jenis bunga” Prompt untuk mengkonfigurasi kartu respon untuk`FlowerType`, seperti yang ditunjukkan pada gambar berikut.



d. Beri kartu judul dan konfigurasikan tiga tombol seperti yang ditunjukkan pada screen shot berikut. Anda dapat menambahkan gambar ke kartu respons, asalkan Anda memiliki URL gambar. Jika Anda menerapkan bot Anda menggunakan Twilio SMS, Anda harus memberikan URL gambar.



e. Pilih Simpan untuk menyimpan kartu respons.

f. Pilih Simpan maksud untuk menyimpan konfigurasi maksud.

g. Untuk membangun bot, pilih Build.

h. Untuk mempublikasikan versi bot, pilih Publikasikan. Tentukan BETA sebagai alias yang menunjuk ke versi bot. Untuk informasi tentang pembuatan versi, lihatPembuatan Versi dan Alias.

3. Terapkan bot pada platform perpesanan:

   - Menyebarkan bot pada platform Facebook Messenger dan menguji integrasi. Untuk petunjuk, lihat Mengintegrasikan Amazon Lex Bot dengan Facebook Messenger. Ketika Anda memesan bunga, jendela pesan menunjukkan kartu respons sehingga Anda dapat memilih jenis bunga.
   - Terapkan bot pada platform Slack dan uji integrasi. Untuk petunjuk, lihat Mengintegrasikan Amazon Lex Bot dengan Slack . Ketika Anda memesan bunga, jendela pesan menunjukkan kartu respons sehingga Anda dapat memilih jenis bunga.
   - Menyebarkan bot pada platform SMS Twilio. Untuk petunjuk, lihat Mengintegrasikan Amazon Lex Bot dengan Twilio Programmable SMS . Saat Anda memesan bunga, pesan dari Twilio menunjukkan gambar dari kartu respons. Twilio SMS tidak mendukung tombol dalam respons.

Memperbarui Utanya

Dalam latihan ini, Anda menambahkan ucapan tambahan pada ucapan yang Anda buat di Latihan Memulai 1. Anda menggunakan tab Monitoring di konsol Amazon Lex untuk melihat ucapan yang tidak dikenali bot Anda. Untuk meningkatkan pengalaman bagi pengguna Anda, Anda menambahkan ucapan tersebut ke bot.

Statistik ucapan tidak dihasilkan dalam kondisi berikut:

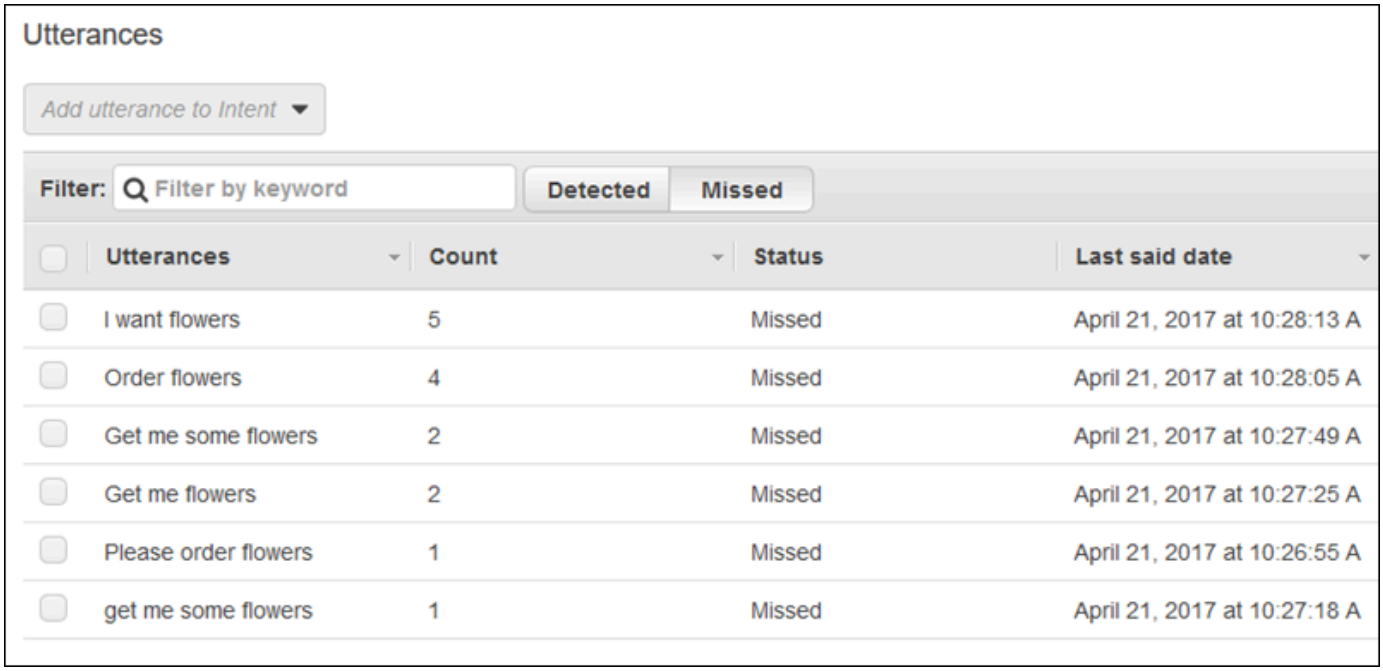
- `childDirected`Bidang diatur ke true ketika bot dibuat.
- Anda menggunakan slot kebingungan dengan satu atau lebih slot.
- Anda memilih untuk tidak berpartisipasi dalam meningkatkan Amazon Lex.

> **Note**
>
> Statistik ucapan dihasilkan sekali sehari. Engkau dapat melihat perkataan yang tidak dikenali, berapa kali ia didengar, dan tanggal dan waktu terakhir ucapan itu didengar. Diperlukan waktu hingga 24 jam jam jam agar ucapan yang tidak terjawab dapat muncul di konsol.

Anda dapat melihat ucapan untuk berbagai versi bot Anda. Untuk mengubah versi bot yang Anda lihat ucapannya, pilih versi yang berbeda dari drop-down di sebelah nama bot.

Untuk melihat dan menambahkan ucapan yang tidak terjawab ke bot:

1. Ikuti langkah pertama Memulai Latihan 1 untuk membuat dan menguji`OrderFlowers` bot. Untuk petunjuk, lihat Latihan 1: Membuat Bot Amazon Lex Menggunakan Cetak Biru (Konsol).
2. Uji bot dengan mengetikkan ucapan berikut di jendela Test Bot. Ketik setiap ucapan beberapa kali. Contoh bot tidak mengenali ucapan berikut:
   - Urutan bunga
   - Dapatkan saya bunga
   - Silakan pesan bunga
   - Ambilkan aku beberapa bunga
3. Tunggu Amazon Lex mengumpulkan data penggunaan tentang ucapan yang tidak terjawab. Data ucapan dihasilkan sekali per hari, umumnya dalam semalam.
4. Masuk keAWS Management Console dan buka dan buka dan buka dan buka dan buka dan buka dan buka konsol Amazon Lex di https://console.aws.amazon.com/lex/.
5. Pilih`OrderFlowers` bot.
6. Pilih tab Monitoring, dan kemudian pilih Ucapan dari menu kiri dan kemudian pilih tombol Missed. Panel berikut menunjukkan maksimum 100 ucapan yang tidak terjawab.



7. Untuk memilih ucapan yang tidak terjawab yang ingin Anda tambahkan kotak centang di samping mereka. Untuk menambahkan ucapan ke`$LATEST` versi intent, pilih panah bawah di sebelah dropdown Tambahkan ucapan ke maksud, lalu pilih intent.

8. Untuk membangun kembali bot Anda, pilih Build dan kemudian Build lagi untuk membangun kembali bot Anda.
9. Untuk memverifikasi bahwa bot Anda mengenali ucapan baru, gunakan panel Test Bot.

Mengintegrasikan dengan situs Web

Dalam contoh ini Anda mengintegrasikan bot dengan situs Web menggunakan teks dan suara. Anda menggunakan JavaScript danAWSlayanan untuk membangun pengalaman interaktif bagi pengunjung ke situs Web Anda. Anda dapat memilih dari contoh-contoh ini yang didokumentasikan padaAWS AI:

- Menerapkan UI Web untuk Chatbot Anda—Menunjukkan UI Web berfitur lengkap yang menyediakan klien Web untuk chatbots Amazon Lex. Anda dapat menggunakan ini untuk mempelajari tentang klien Web, atau sebagai blok bangunan untuk aplikasi Anda sendiri.

- “Salam, pengunjung!” —Terlibat Pengguna Web Anda dengan Amazon Lex—Menunjukkan menggunakan Amazon Lex, AWS SDK for JavaScript in the Browser, dan Amazon Cognito untuk menciptakan pengalaman percakapan di situs Web Anda.

- Menangkap Input Suara di Browser dan Mengirimnya ke Amazon Lex—Menunjukkan embedding chatbot berbasis suara di situs Web menggunakan SDK for JavaScript in the Browser. Aplikasi merekam audio, mengirim audio ke Amazon Lex, dan kemudian memainkan respon.

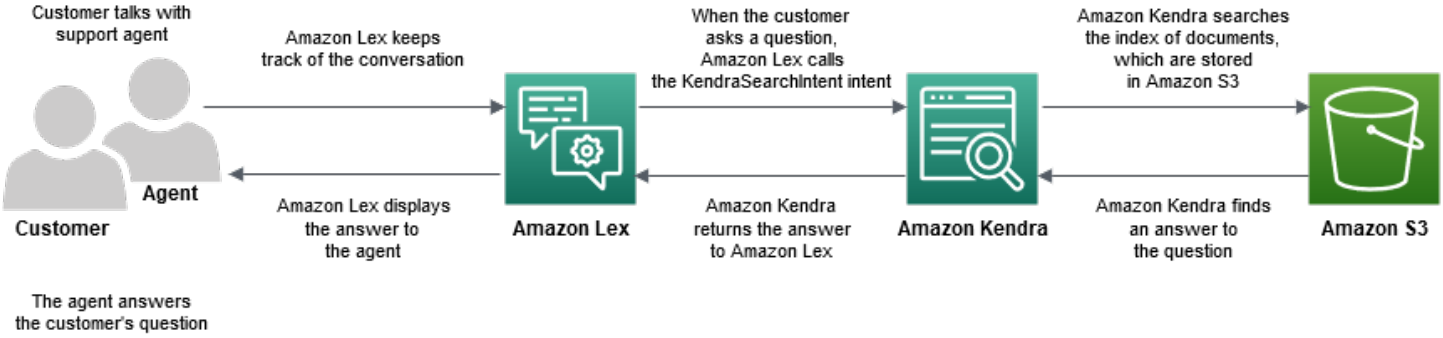
Asisten Agen Call Center

Dalam tutorial ini, Anda menggunakan Amazon Lex 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 melalui dokumen untuk menemukan jawaban. Untuk informasi selengkapnya tentang Amazon Kendra, lihat Panduan Pengembang Amazon Kendra.

Bot Amazon Lex 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 yang digunakan agen untuk menjawab kueri pelanggan secara real time. Dengan membaca jawaban yang disediakan bot, agen terhindar dari mencari jawaban secara manual.

Aplikasi bot dan 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 Amazon Kendra disimpan di bucket Amazon Simple Storage Service (Amazon S3). Jika Anda belum memiliki sebuah bucket S3, Anda dapat mengaturnya saat Anda membuat indeks Amazon Kendra. Selain Amazon S3, Anda akan menggunakan Amazon Cognito untuk tutorial ini. Amazon Cognito mengelola izin untuk menerapkan 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 menerapkan 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. 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 2: Buat Bot Amazon Lex
- Langkah 3: Tambahkan Intent Kustom dan Intent
- Langkah 4: Mengatur 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 bucket S3. Anda dapat mengindeks data tidak terstruktur dan semi-terstruktur dalam format.html, Microsoft Office (.doc, .ppt), PDF, dan teks.

Untuk membuat indeks Amazon Kendra, lihat Memulai dengan 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 file ML_FAQ.csv pada GitHub.

Langkah selanjutnya

Langkah 2: Buat Bot Amazon Lex

Langkah 2: Buat Bot Amazon Lex

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

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

Untuk membuat bot asisten agen

1. Masuk keAWS Management Console dan buka konsol Amazon Lex di https://console.aws.amazon.com/lex/.
2. Di panel navigasi, pilih bot.
3. Pilih Create (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. Batas waktu sesi - Masukkan**`5`**.
   d. COPPA - Pilih No.
5. Pilih Create (Buat). Setelah membuat bot, Amazon Lex menampilkan tab editor bot.

Langkah selanjutnya

Langkah 3: Tambahkan Intent Kustom dan Intent

Langkah 3: Tambahkan Intent Kustom dan Intent

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

Amazon Lex memiliki dua jenis intent: maksud khusus dan intent bawaan. `AMAZON.KendraSearchIntent`adalah maksud bawaan. Bot menggunakan`AMAZON.KendraSearchIntent` maksud untuk melakukan kueri indeks dan menampilkan tanggapan 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. Ini tidak melakukan fungsi lainnya. Ucapan untuk maksud tersebut tidak boleh menjawab pertanyaan apa pun yang mungkin ditanyakan pelanggan. Hal 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 Tambahkan maksud, pilih Buat maksud.
3. Di kotak dialog Buat maksud, masukkan nama deskriptif untuk intent, misalnya**`RequiredIntent`**.
4. Untuk Sampel ucapan, masukkan ucapan deskriptif, seperti**`Required utterance`**.
5. Pilih Simpan maksud.

Untuk menambahkan AMAZON. KendraSearchIntent maksud dan pesan respons

1. Di panel navigasi, pilih tanda tambah (+) di sebelah Maksud.
2. Pilih Cari maksud yang ada.
3. Di kotak Maksud pencarian, masukkan**`AMAZON.KendraSearchIntent`**, lalu pilih dari daftar.
4. Beri maksud nama deskriptif, misalnya**`AgentAssistSearchIntent`**, lalu pilih Tambah.
5. Di editor maksud, pilih kueri Amazon Kendra untuk membuka opsi kueri.
6. Pilih indeks yang Anda inginkan intent untuk dicari,
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 selanjutnya

[Langkah 4: Mengatur Amazon Cognito]

Langkah 4: Mengatur Amazon Cognito

Untuk mengelola izin dan pengguna untuk aplikasi web, Anda perlu menyiapkan Amazon Cognito. Amazon Cognito memastikan bahwa aplikasi web aman dan memiliki kontrol akses. Amazon Cognito menggunakan kumpulan identitas untuk menyediakanAWS kredensial yang memberikan akses keAWS layanan lainnya. Untuk tutorial ini, ia menyediakan akses ke Amazon Lex.

Saat membuat kolam identitas, Amazon Cognito menyediakan peranAWS Identity and Access Management (IAM) bagi pengguna yang diautentikasi dan tidak diautentikasi. Anda mengubah peran IAM dengan menambahkan kebijakan yang memberikan akses ke Amazon Lex.

Mengatur Amazon Cognito

1. Masuk keAWS 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. Konfigurasi kolam identitas.
    a. Nama pangkalan identitas - Masukkan nama yang menunjukkan tujuan pangkalan, seperti**`BotPool`**.
    b. Di bagian Identitas tidak diautentikasi, pilih Aktifkan akses ke identitas yang tidak diautentikasi.
5. Pilih Buat kolam.
6. Pada halaman Identifikasi peran IAM yang akan digunakan dengan pangkalan identitas baru Anda, pilih Lihat Detail.
7. Rekam nama peran IAM. Anda akan memodifikasi mereka nanti.
8. Pilih Izinkan.
9. Pada halaman Memulai dengan Amazon Cognito, untuk Platform, pilih JavaScript.
10. Di bagian GetAWS Credentials, temukan dan rekam ID kumpulan Identity.
11. Untuk mengizinkan akses ke Amazon Lex, 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 Pasang 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 Pasang kebijakan.

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

   iii. Pilih Lampirkan kebijakan.

Langkah selanjutnya

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. Lakukan perubahan berikut.

   a. Di`AWS.config.credentials` bagian tersebut, masukkan nama Wilayah dan ID kumpulan identitas Anda.

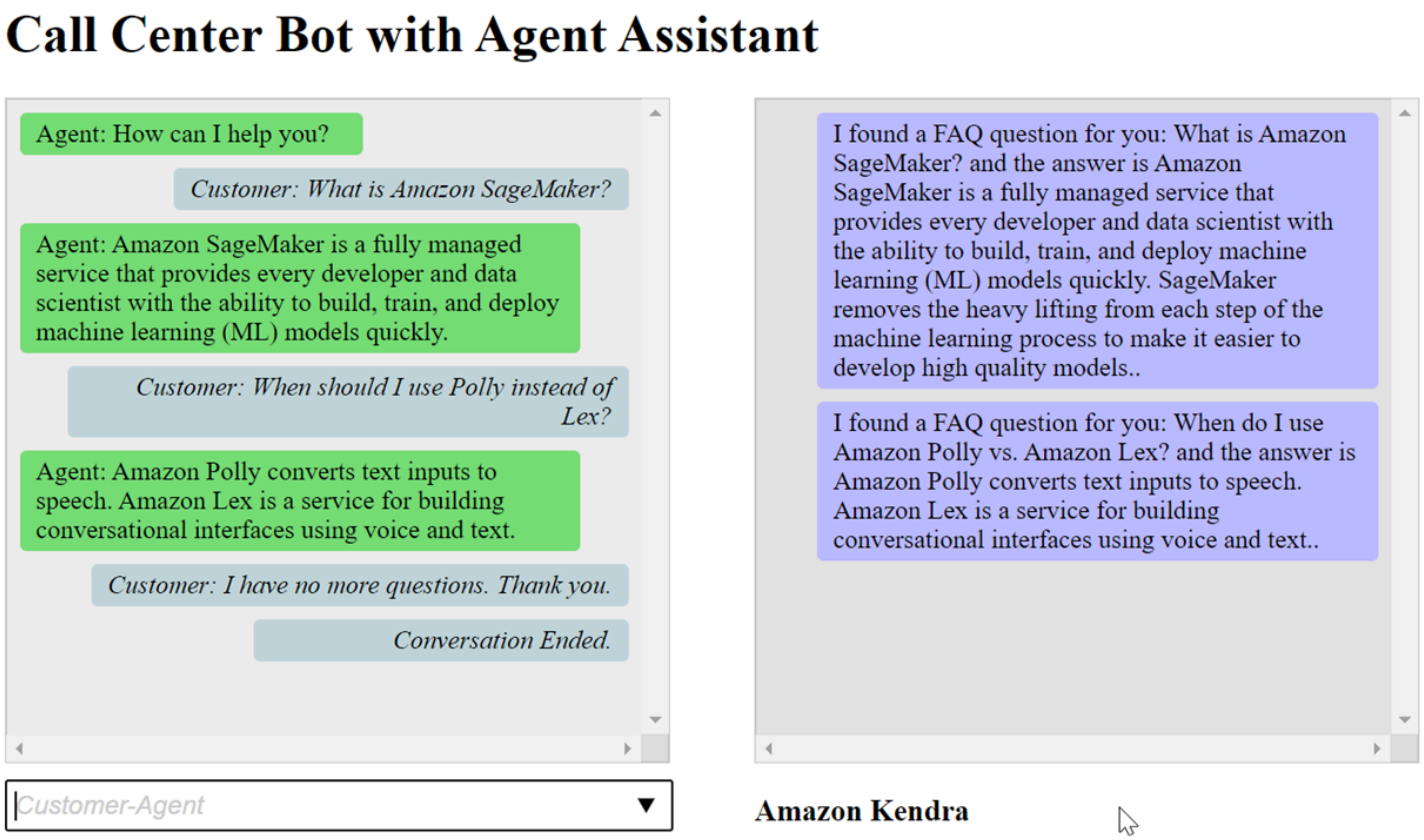
   b. Di `Amazon Lex runtime parameters` bagian tersebut, masukkan nama bot.

   c. Simpan file tersebut.

Langkah 6: Gunakan Bot

Untuk tujuan demo, Anda memberikan masukan ke bot sebagai pelanggan dan sebagai agen. Untuk membedakan 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 yang mirip dengan gambar berikut dengan bot Anda:



`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 mengirimkannya ke Amazon Lex.

`showMessage(daText, senderRequest, displayWindow)`Fungsi 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.

Memigrasikan bot

API Amazon Lex V2 menggunakan arsitektur informasi terbaru yang memungkinkan versi sumber daya yang disederhanakan dan dukungan untuk berbagai bahasa dalam bot. Untuk informasi selengkapnya, lihat panduan Migrasi di Panduan Pengembang Amazon Lex V2.

Untuk menggunakan fitur-fitur baru ini, Anda perlu memigrasi bot Anda. Saat Anda memigrasi bot, Amazon Lex menyediakan hal-hal berikut:

- Migrasi menyalin maksud khusus dan jenis slot Anda ke bot Amazon Lex V2.
- Anda dapat menambahkan beberapa bahasa ke bot Amazon Lex V2 yang sama. Di Amazon Lex V1 Anda membuat bot terpisah untuk setiap bahasa. Anda dapat memigrasi beberapa bot Amazon Lex V1, masing-masing menggunakan bahasa yang berbeda, ke satu bot Amazon Lex V2.
- Amazon Lex memetakan Amazon Lex V1 built-in jenis slot dan maksud untuk Amazon Lex V2 built-in jenis slot dan maksud. Jika built-in tidak dapat dimigrasi, Amazon Lex mengembalikan pesan yang memberi tahu Anda apa yang harus dilakukan selanjutnya.

Proses migrasi tidak memigrasi berikut ini:

- Alias
- Amazon Kendra
- AWS Lambda fungsi
- Pengaturan log percakapan
- Saluran pesan seperti Slack
- Tanda

Untuk memigrasi bot, pengguna atau peran Anda harus memiliki izin IAM untuk operasi Amazon Lex dan Amazon Lex V2 API. Untuk izin yang diperlukan, lihat Izinkan pengguna untuk memigrasikan bot ke Amazon Lex V2 API.

Memigrasi bot (Konsol)

Gunakan konsol Amazon Lex V1 untuk memigrasi struktur bot ke bot Amazon Lex V2.

Menggunakan konsol untuk memigrasi bot ke API Amazon Lex V2

1. Masuk keAWS Management Console dan buka konsol Amazon Lex di https://console.aws.amazon.com/lex/.
2. Pada menu kiri, pilih Alat Migrasi.
3. Dari daftar bot, pilih bot yang ingin Anda migrasi lalu pilih Migrasi.
4. Pilih versi bot yang ingin Anda migrasi, lalu masukkan nama bot yang akan dimigrasi. Jika Anda memasukkan nama bot Amazon Lex V2 yang ada, bot Amazon Lex V1 dimigrasi ke bahasa yang ditunjukkan dalam detail dan menimpa versi Draf bahasa.
5. Pilih Selanjutnya.
6. Pilih peran IAM yang digunakan Amazon Lex untuk menjalankan versi API Amazon Lex V2 dari bot. Anda dapat memilih untuk membuat peran baru dengan menetapkan izin minimum yang diperlukan untuk menjalankan bot, atau Anda dapat memilih peran IAM yang ada.
7. Pilih Selanjutnya.
8. Tinjau pengaturan untuk migrasi. Jika terlihat OK, pilih Mulai migrasi.

Setelah mulai proses migrasi, Anda akan dikembalikan ke halaman alat migrasi. Anda dapat memantau kemajuan migrasi di tabel Riwayat. Ketika kolom Status Migrasi mengatakan Selesai migrasi selesai.

Amazon Lex menggunakan`StartImport` operasi di Amazon Lex V2 API untuk mengimpor bot yang dimigrasi. Anda akan melihat entri di tabel riwayat impor konsol Amazon Lex V2 untuk setiap migrasi.

Selama migrasi, Amazon Lex mungkin menemukan sumber daya dalam bot yang tidak dapat dimigrasi. Anda mendapatkan pesan kesalahan atau peringatan untuk setiap sumber daya yang tidak dapat dimigrasi. Setiap pesan menyertakan tautan ke dokumentasi yang menjelaskan cara mengatasi masalah tersebut.

Memigrasikan fungsi Lambda

Amazon Lex V2 mengubah cara fungsi Lambda didefinisikan untuk bot. Ini hanya memungkinkan satu fungsi Lambda dalam alias untuk setiap bahasa dalam bot. Untuk informasi selengkapnya tentang migrasi fungsi Lambda Anda, lihatMemigrasi fungsi Lambda dari Amazon Lex V1 ke Amazon Lex V2.

pesan migrasi

Selama migrasi, Amazon Lex mungkin menemukan sumber daya, seperti jenis slot bawaan, yang tidak dapat dimigrasi ke sumber daya Amazon Lex V2 yang setara. Ketika ini terjadi, Amazon Lex mengembalikan pesan migrasi yang menjelaskan apa yang terjadi dan menyediakan tautan ke dokumentasi yang memberi tahu Anda cara memperbaiki masalah migrasi. Bagian berikut menjelaskan masalah yang mungkin timbul saat Anda memigrasi bot dan cara memperbaiki masalah.

Topik

Maksud bawaan

Bila Anda menggunakan intent bawaan yang tidak didukung di Amazon Lex V2, intent tersebut dipetakan ke maksud khusus di bot Amazon Lex V2 Anda. Maksud kustom tidak mengandung ucapan. Untuk terus menggunakan maksud, tambahkan contoh ucapan.

Tipe slot bawaan

Setiap slot yang dimigrasi yang menggunakan jenis slot yang tidak didukung di Amazon Lex V2 tidak akan diberikan nilai jenis slot. Untuk menggunakan slot ini:

- Buat jenis slot khusus
- Tambahkan nilai jenis slot yang diharapkan untuk jenis slot
- Perbarui slot untuk menggunakan jenis slot khusus baru

Log percakapan

Migrasi tidak memperbarui pengaturan log percakapan bot Amazon Lex V2.

Untuk mengatur konfigurasi log percakapan

1. Buka konsol Amazon Lex V2 di https://console.aws.amazon.com/lexv2.
2. Dari daftar bot, pilih bot yang log percakapannya ingin Anda konfigurasikan.
3. Pada menu kiri, pilih Alias, lalu pilih alias dari daftar.
4. Di bagian Log percakapan, pilih Kelola log percakapan untuk mengonfigurasi log percakapan untuk alias bot.

Grup pesan

Amazon Lex V2 hanya mendukung satu pesan dan dua pesan alternatif per grup pesan. Jika Anda memiliki lebih dari tiga pesan per grup pesan dalam bot Amazon Lex V1, hanya tiga pesan pertama yang dimigrasi. Untuk menggunakan lebih banyak pesan dalam grup pesan, gunakan fungsi Lambda untuk menampilkan berbagai pesan.

Perintah dan frase

Amazon Lex V2 menggunakan mekanisme yang berbeda untuk menindaklanjuti, klarifikasi, dan menutup telepon.

Untuk petunjuk tindak lanjut, gunakan pengalihan konteks untuk beralih ke maksud yang berbeda setelah pemenuhan.

Misalnya, anggaplah Anda memiliki maksud untuk memesan penyewaan mobil yang dikonfigurasi untuk mengembalikan konteks keluaran yang disebut`book_car_fulfilled`. Saat intent terpenuhi, Amazon Lex menyetel variabel konteks keluaran ke`book_car_fulfilled`. Karena`book_car_fulfilled` adalah konteks aktif, maksud dengan`book_car_fulfilled` konteks masukan dianggap sebagai pengakuan, selama ucapan pengguna dikenali sebagai upaya untuk mendapatkan maksud tersebut. Anda dapat menggunakan ini untuk maksud yang hanya masuk akal setelah memesan mobil, seperti mengirim email tanda terima atau memodifikasi reservasi.

Amazon Lex V2 tidak mendukung petunjuk klarifikasi dan menutup frasa (membatalkan pernyataan). Bot Amazon Lex V2 berisi maksud mundur default yang dipanggil jika tidak ada maksud yang cocok. Untuk mengirim prompt klarifikasi dengan percobaan ulang, konfigurasikan fungsi Lambda dan aktifkan hook kode dialog dalam intent fallback. Fungsi Lambda dapat menampilkan prompt klarifikasi sebagai respons dan nilai coba lagi dalam atribut session. Jika nilai coba lagi melebihi jumlah percobaan ulang maksimum, Anda dapat menampilkan frasa hang up dan menutup percakapan.

Fitur Amazon Lex V1 lainnya

Alat migrasi hanya mendukung migrasi bot Amazon Lex V1 dan maksud, jenis slot, dan slotnya yang mendasarinya. Untuk fitur lainnya, lihat topik berikut dalam dokumentasi Amazon Lex V2.

- Alias bot: Alias
- Saluran bot: Menerapkan bot Amazon Lex V2 di platform perpesanan
- Pengaturan log percakapan: Pemantauan dengan log percakapan
- Indeks Amazon Kendra: AMAZON. KendraSearchIntent
- Fungsi Lambda: MenggunakanAWS Lambda fungsi
- Tags: Tagging sumber daya

Memigrasi fungsi Lambda dari Amazon Lex V1 ke Amazon Lex V2

Amazon Lex V2 hanya mengizinkan satu fungsi Lambda untuk setiap bahasa dalam bot. Fungsi Lambda dan pengaturannya dikonfigurasi untuk alias bot yang Anda gunakan saat runtime.

Fungsi Lambda dipanggil untuk semua maksud dalam bahasa tersebut jika kait kode dialog dan pemenuhan diaktifkan untuk maksud tersebut.

Fungsi Amazon Lex V2 Lambda memiliki format pesan input dan output yang berbeda dari Amazon Lex V1. Ini adalah perbedaan dalam format input fungsi Lambda.

- Amazon Lex V2 menggantikan`currentIntent` dan`alternativeIntents` struktur dengan`interpretations` struktur. Setiap interpretasi berisi maksud, skor kepercayaan NLU untuk maksud, dan analisis sentimen opsional.
- Amazon Lex V2 memindahkan`activeContexts`,`sessionAttributes` di Amazon Lex V1 ke`sessionState` struktur terpadu. Struktur ini memberikan informasi tentang keadaan percakapan saat ini, termasuk ID permintaan asal.
- Amazon Lex V2 tidak mengembalikan`recentIntentSummaryView`. Gunakan informasi dalam`sessionState` struktur sebagai gantinya.
- Input Amazon Lex V2 menyediakan`botId` dan`localeId` dalam`bot` atribut.
- Struktur input berisi`inputMode` atribut yang menyediakan informasi tentang jenis input: teks, ucapan, atau DTMF.

Ini adalah perbedaan dalam format output fungsi Lambda:

- `sessionAttributes`Strukturactive`Contexts` dan di Amazon Lex V1 digantikan oleh`sessionState` struktur di Amazon Lex V2.
- `recentIntentSummaryView`Tidak termasuk dalam output.
- `dialogAction`Struktur Amazon Lex V1 dibagi menjadi dua struktur,`dialogAction` yaitu bagian dari`sessionState` struktur, dan`messages` itu diperlukan ketika`dialogAction.type` is`ElicitIntent`. Amazon Lex memilih pesan dari struktur ini untuk ditampilkan kepada pengguna.

Saat Anda membuat bot dengan API Amazon Lex V2, hanya ada satu fungsi Lambda per alias bot per bahasa, bukan fungsi Lambda untuk setiap maksud. Jika Anda ingin terus menggunakan fungsi terpisah, Anda dapat membuat fungsi router yang mengaktifkan fungsi terpisah untuk setiap intent. Berikut ini adalah 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
```

Daftar bidang yang diperbarui

Tabel berikut memberikan informasi terperinci tentang bidang yang diperbarui dalam permintaan dan respons Amazon Lex V2 Lambda. Anda dapat menggunakan tabel ini untuk memetakan bidang antara versi.

Permintaan

Bidang berikut telah diperbarui dalam format permintaan fungsi Lambda.

Konteks aktif

`activeContexts`Strukturnya sekarang menjadi bagian dari`sessionState` struktur.

Struktur V1	Struktur V2
ActiveContext	SessionState.ActiveContexts
ActiveContext [*]. timeToLive	SessionState.ActiveContexts [*]. timeToLive
ActiveContext [*]. timeToLive. timeToLiv eInSeconds	SessionState.ActiveContexts [*]. timeToLive. timeToLiveInSeconds
ActiveContext [*]. timeToLive. turnsToLive	SessionState.ActiveContexts [*]. timeToLive. turnsToLive
ActiveContexts [*] .nama	SessionState.ActiveContexts [*] .nama
ActiveContexts [*] .parameter	SessionState.ActiveContexts [*] .ContextA ttributes

Maksud alternatif

Daftar interpretasi dari indeks 1 hingga N berisi daftar maksud alternatif yang diprediksi oleh Amazon Lex V2, bersama dengan skor kepercayaan mereka. `recentIntentSummaryView`Itu dihapus dari struktur permintaan di Amazon Lex V2. Untuk melihat detail dari`recentIntentSummaryView`, gunakanGetSession operasi.

Struktur V1	Struktur V2
AlternatifMaksud	interpretasi [1: *]
recentIntentSummaryLihat	T/A

Bot

Di Amazon Lex V2, bot dan alias memiliki pengenal. ID bot adalah bagian dari masukan codehook. ID alias disertakan, tetapi bukan nama alias. Amazon Lex V2 mendukung beberapa lokal untuk bot yang sama sehingga ID lokal disertakan.

Struktur V1	Struktur V2
bot	bot
bot.name	bot.name
T/A	bot.id
alias	T/A
T/A	Bot.aliasid
bot.version	bot.version
T/A	Bot.localeid

Niat saat ini

`sessionState.intent`Struktur berisi rincian maksud aktif. Amazon Lex V2 juga mengembalikan daftar semua maksud, termasuk maksud alternatif, dala`interpretations` struktur. Elemen pertama dalam daftar interpretasi selalu sama dengan`sessionState.intent`.

Struktur V1	Struktur V2
currentIntent	SessionState.intent ATAU interpretasi [0] .maksud

Struktur V1	Struktur V2
CurrentIntent.name	SessionState.intent.name ATAU interpretasi [0] .intent.name
currentIntent. nluConfidenceScore	interpretasi [0] .nluconfidence.Score

Tindakan dialog

`confirmationStatus`Bidang sekarang menjadi bagian dari`sessionState` struktur.

Struktur V1	Struktur V2
CurrentIntent.ConfirmationStatus	SessionState.Intent.ConfirmationState ATAU interpretasi [0] .intent.confirmationState
T/A	SessionState.intent.state ATAU interpretasi [*] .intent.state

Amazon Kendra

`kendraResponse`Bidang sekarang bagian dari`sessionState` dan`interpretations` struktur.

Struktur V1	Struktur V2
KendraResponse	SessionState.Intent.KendraResponse ATAU interpretasi [0] .Intent.kendraResponse

Sentimen

`sentimentResponse`Struktur dipindahkan ke`interpretations` struktur baru.

Struktur V1	Struktur V2
sentimentResponse	interpretasi [0] .sentimentResponse
sentimentResponse.sentimentLabel	interpretasi [0] .sentimentResponse.Sentimen

Struktur V1	Struktur V2
sentimentresponse.sentimentScore	interpretasi [0] .sentimentResponse.sentimentScore

Slot

Amazon Lex V2 menyediakan satu`slots` objek di dalam`sessionState.intent` struktur yang berisi nilai yang diselesaikan, nilai yang ditafsirkan, dan nilai asli dari apa yang dikatakan pengguna. Amazon Lex V2 juga mendukung slot multi-nilai dengan mengatur`slotShape` as`List` dan mengatur`values` daftar. Slot nilai tunggal didukung oleh`value` lapangan, bentuknya diasumsikan`Scalar`.

Struktur V1	Struktur V2
CurrentIntent.slot	SessionState.Intent.Slots ATAU interpretasi [0] .intent.slots
CurrentIntent.slot [*] .nilai	SessionState.Intent.Slots [*] .value.interpretedValue ATAU interpretasi [0] .intent.slots [*] .value.interpretedValue
T/A	SessionState.intent.slots [*] .value.shape ATAU interpretasi [0] .intent.slots [*] .shape
T/A	SessionState.intent.slots [*] .values ATAU interpretasi [0] .intent.slots [*] .values
CurrentIntent.slotDetails	SessionState.Intent.Slots ATAU interpretasi [0] .intent.slots
CurrentIntent.slotDetails [*] .resolusi	SessionState.intent.slots [*] .terselesaikanNilai ATAU interpretasi [0] .intent.slots [*] .terselesaikanNilai
currentIntent.slotDetails [*] .originalValue	SessionState.intent.slots [*] .originalValue ATAU interpretasi [0] .intent.slots [*] .originalValue

Lainnya

`sessionId`Bidang Amazon Lex V2 sama dengan`userId` bidang di Amazon Lex V1. Amazon Lex V2 juga mengirimkan`inputMode` pemanggil: teks, DTMF, atau pidato.

Struktur V1	Struktur V2
userId	sessionId
inputTranscript	inputTranscript
invocationSource	invocationSource
outputDialogMode	responseContentType
messageVersion	messageVersion
SessionAttributes	SessionState.SessionAttributes
requestAttributes	requestAttributes
T/A	Mode masukan
T/A	originatingRequestId

Response

Bidang berikut telah diubah dalam format pesan respons fungsi Lambda.

Konteks aktif

`activeContexts`Struktur pindah ke`sessionState` struktur.

Struktur V1	Struktur V2
ActiveContext	SessionState.ActiveContexts
ActiveContext [*]. timeToLive	SessionState.ActiveContexts [*]. timeToLive

Struktur V1	Struktur V2
ActiveContext [*]. timeToLive. timeToLiv eInSeconds	SessionState.ActiveContexts [*]. timeToLive. timeToLiveInSeconds
ActiveContext [*]. timeToLive. turnsToLive	SessionState.ActiveContexts [*]. timeToLive. turnsToLive
ActiveContexts [*] .nama	SessionState.ActiveContexts [*] .nama
ActiveContexts [*] .parameter	SessionState.ActiveContexts [*] .ContextA ttributes

Tindakan dialog

`dialogAction`Struktur pindah ke`sessionState` struktur. Anda sekarang dapat menentukan beberapa pesan dalam tindakan dialog, dan`genericAttachments` struktur sekarang`imageResponseCard` struktur.

Struktur V1	Struktur V2
dialogAction	SessionState.DialogAction
Dialogaction.type	SessionState.dialogaction.type
dialogAction. slotToElicit	SessionState.Intent.DialogAction. slotToElicit
dialogaction.type.fulfillmentState	SessionState.Intent.State
Dialogaction.message	pesan
DialogAction.message.contentType	pesan [*] .contentType
Dialogaction.message.content	pesan [*] .konten
Dialogaction.responseCard	pesan [*]. imageResponseCard
Dialogaction.responseCard.version	T/A
dialogaction.responseCard.contentType	pesan [*] .contentType

Struktur V1	Struktur V2
dialogaction.responsecard.genericlampiran	T/A
Dialogaction.responseCard.genericAttachments [*] .title	pesan [*]. imageResponseCard.judul
Dialogaction.responseCard.genericAttachments [*] .subtitle	pesan [*]. imageResponseCard.subjudul
Dialogaction.responseCard.genericAttachments [*] .imageUrl	pesan [*]. imageResponseCard.imageUrl
Dialogaction.responseCard.genericAttachments [*] .tombol	pesan [*]. imageResponseCard.tombol
Dialogaction.responseCard.genericAttachments [*] .buttons [*] .nilai	pesan [*]. imageResponseCard.buttons [*] .nilai
Dialogaction.responseCard.genericAttachments [*] .buttons [*] .text	pesan [*]. imageResponseCard.buttons [*] .text
dialogAction. kendraQueryRequestMuatan	dialogAction. kendraQueryRequestMuatan
dialogAction. kendraQueryFilterString	dialogAction. kendraQueryFilterString

Maksud dan slot

Bidang maksud dan slot yang merupakan bagian dari`dialogAction` struktur sekarang menjadi bagian dari`sessionState` struktur.

Struktur V1	Struktur V2
Dialogaction.IntentName	SessionState.Intent.name
Dialogaction.slot	SessionState.Intent.Slots
DialogAction.slot [*] .kunci	SessionState.intent.slots [*] .kunci

Struktur V1	Struktur V2
DialogAction.slot [*] .nilai	SessionState.Intent.Slots [*] .value.interpreted Value
T/A	SessionState.intent.slots [*] .value.shape
T/A	SessionState.intent.slots [*] .nilai

Lainnya

`sessionAttributes`Strukturnya sekarang menjadi bagian dari`sessionState` struktur. `recentIntentSummaryReview`Struktur telah dihapus.

Struktur V1	Struktur V2
SessionAttributes	SessionState.SessionAttributes
recentIntentSummaryLihat	T/A

Keamanan di Amazon Lex

Keamanan cloud di AWS adalah prioritas tertinggi. Sebagai AWS pelanggan, Anda mendapat manfaat dari pusat data dan arsitektur jaringan yang dibangun untuk memenuhi persyaratan organisasi yang paling sensitif terhadap keamanan.

Keamanan adalah tanggung jawab bersama antara Anda AWS dan Anda. Model tanggung jawab bersama menjelaskan hal ini sebagai keamanan dari cloud dan keamanan di cloud:

- Keamanan cloud — AWS bertanggung jawab untuk melindungi infrastruktur yang menjalankan AWS layanan di AWS Cloud. AWS juga memberi Anda layanan yang dapat Anda gunakan dengan aman. Efektivitas keamanan kami diuji dan diverifikasi secara rutin oleh auditor pihak ketiga sebagai bagian dari program kepatuhan AWS. Untuk mempelajari tentang program kepatuhan yang berlaku untuk Amazon Lex, lihat AWS Layanan dalam Lingkup berdasarkan Program Kepatuhan.
- Keamanan di cloud — Tanggung jawab Anda ditentukan oleh AWS layanan yang Anda gunakan. Anda juga bertanggung jawab atas faktor-faktor lain termasuk sensitivitas data Anda, persyaratan organisasi Anda, serta undang-undang dan peraturan yang berlaku.

Dokumentasi ini akan membantu Anda memahami cara menerapkan model tanggung jawab bersama saat menggunakan Amazon Lex. Topik berikut menunjukkan cara mengonfigurasi Amazon Lex untuk memenuhi tujuan keamanan dan kepatuhan Anda. Anda juga akan mempelajari cara menggunakan layanan AWS lain yang dapat membantu Anda memantau dan mengamankan sumber daya Amazon Lex Anda.

Topik

Perlindungan Data di Amazon Lex

Amazon Lex mengumpulkan konten pelanggan untuk pemecahan masalah dan untuk membantu meningkatkan layanan. Konten pelanggan diamankan secara default. Anda dapat menghapus konten untuk pelanggan individu menggunakan Amazon Lex API.

Amazon Lex menyimpan empat jenis konten:

- Contoh ucapan, yang digunakan untuk membangun dan melatih bot
- Ucapan pelanggan dari pengguna yang berinteraksi dengan bot
- Atribut sesi, yang menyediakan informasi khusus aplikasi selama interaksi pengguna dengan bot
- Atribut permintaan, yang berisi informasi yang berlaku untuk satu permintaan ke bot

Setiap bot Amazon Lex yang dirancang untuk digunakan oleh anak-anak diatur oleh Children's Online Privacy Protection Act (COPPA). Anda memberi tahu Amazon Lex bahwa bot tunduk pada COPPA dengan menggunakan konsol atau Amazon Lex API untuk menyetel `childDirected` bidang `ketrue`. Saat `childDirected` bidang disetel `ketrue`, tidak ada ucapan pengguna yang disimpan.

Topik

- Enkripsi saat Data Tidak Berpindah
- Enkripsi Saat Data Berpindah
- Manajemen kunci

Enkripsi saat Data Tidak Berpindah

Amazon Lex mengenkripsi ucapan pengguna yang disimpannya.

Topik

- Sampel Ucapan
- Ucapan Pelanggan
- Atribut Sesi
- Permintaan Atribut

Sampel Ucapan

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 digunakan untuk membangun bot dan untuk menciptakan pengalaman pengguna.

Ucapan Pelanggan

Amazon Lex mengenkripsi ucapan yang dikirim pengguna ke bot Anda kecuali jika `childDirected` bidangnya disetel ke. `true`

Saat `childDirected` bidang disetel ke`true`, tidak ada ucapan pengguna yang disimpan.

Ketika `childDirected` bidang diatur ke `false` (default), ucapan pengguna dienkripsi dan disimpan selama 15 hari untuk digunakan dengan operasi. GetUtterancesView Untuk menghapus ucapan yang disimpan untuk pengguna tertentu, gunakan operasi. DeleteUtterances

Saat bot Anda menerima input suara, input disimpan tanpa batas waktu. Amazon Lex menggunakannya untuk meningkatkan kemampuan bot Anda untuk merespons input pengguna.

Gunakan DeleteUtterances operasi untuk menghapus ucapan tersimpan untuk pengguna tertentu.

Atribut Sesi

Atribut sesi berisi informasi khusus aplikasi yang diteruskan antara Amazon Lex dan aplikasi klien. Amazon Lex meneruskan atribut sesi ke semua AWS Lambda fungsi yang dikonfigurasi untuk bot. Jika fungsi Lambda menambahkan atau memperbarui atribut sesi, Amazon Lex 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.

Permintaan Atribut

Atribut permintaan berisi informasi khusus permintaan dan hanya berlaku untuk permintaan saat ini. Aplikasi klien menggunakan atribut permintaan untuk mengirim informasi ke Amazon Lex 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 Data Berpindah

Amazon Lex 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 kunci

Amazon Lex melindungi konten Anda dari penggunaan yang tidak sah dengan kunci internal.

Identity and Access Management untuk Amazon Lex

(IAM) AWS Identity and Access Management adalah Layanan AWS yang membantu seorang administrator dalam mengendalikan akses ke sumber daya AWS secara aman. Administrator IAM mengontrol siapa yang dapat diautentikasi (masuk) dan diotorisasi (memiliki izin) untuk menggunakan sumber daya Amazon Lex. IAM adalah sebuah layanan Layanan AWS yang dapat Anda gunakan tanpa dikenakan biaya tambahan.

Topik

Audiens

Cara Anda menggunakan AWS Identity and Access Management (IAM) berbeda, tergantung pada pekerjaan yang Anda lakukan di Amazon Lex.

Pengguna layanan - Jika Anda menggunakan layanan Amazon Lex untuk melakukan pekerjaan Anda, administrator Anda memberi Anda kredensi dan izin yang Anda butuhkan. Saat Anda

menggunakan lebih banyak fitur Amazon Lex untuk melakukan pekerjaan Anda, Anda mungkin memerlukan izin tambahan. Memahami cara akses dikelola dapat membantu Anda meminta izin yang tepat dari administrator Anda. Jika Anda tidak dapat mengakses fitur di Amazon Lex, lihatMemecahkan masalah identitas dan akses Amazon Lex.

Administrator layanan - Jika Anda bertanggung jawab atas sumber daya Amazon Lex di perusahaan Anda, Anda mungkin memiliki akses penuh ke Amazon Lex. Tugas Anda adalah menentukan fitur dan sumber daya Amazon Lex mana yang harus diakses pengguna layanan Anda. Kemudian, Anda harus mengirimkan permintaan kepada administrator IAM Anda untuk mengubah izin pengguna layanan Anda. Tinjau informasi di halaman ini untuk memahami konsep dasar IAM. Untuk mempelajari lebih lanjut tentang bagaimana perusahaan Anda dapat menggunakan IAM dengan Amazon Lex, lihatBagaimana Amazon Lex bekerja dengan IAM.

Administrator IAM - Jika Anda seorang administrator IAM, Anda mungkin ingin mempelajari detail tentang cara menulis kebijakan untuk mengelola akses ke Amazon Lex. Untuk melihat contoh kebijakan berbasis identitas Amazon Lex yang dapat Anda gunakan di IAM, lihat. Contoh kebijakan berbasis identitas untuk Amazon Lex

Mengautentikasi dengan identitas

Autentikasi merupakan cara Anda untuk masuk ke AWS dengan menggunakan kredensial identitas Anda. Anda harus terautentikasi (masuk keAWS) sebagai Pengguna root akun AWS, sebagai pengguna IAM, atau dengan mengambil peran IAM.

Anda dapat masuk ke AWS sebagai identitas terfederasi dengan menggunakan kredensial yang disediakan melalui sumber identitas. AWS IAM Identity Center Para pengguna (Pusat Identitas IAM), otentikasi sign-on tunggal perusahaan Anda, dan kredensial Google atau Facebook Anda merupakan contoh identitas terfederasi. Saat Anda masuk sebagai identitas terfederasi, administrator Anda sebelumnya menyiapkan federasi identitas dengan menggunakan peran IAM. Ketika Anda mengakses AWS dengan menggunakan federasi, Anda secara tidak langsung mengambil suatu peran.

Tergantung pada jenis pengguna Anda, Anda dapat masuk ke AWS Management Console atau portal akses AWS. Untuk informasi selengkapnya tentang masuk ke AWS, silakan lihat Cara masuk ke Akun AWS Anda di Panduan Pengguna AWS Sign-In.

Jika Anda mengakses AWS secara terprogram, AWS sediakan kit pengembangan perangkat lunak (SDK) dan antarmuka baris perintah (CLI) untuk menandatangani permintaan Anda secara kriptografis dengan menggunakan kredensial Anda. Jika Anda tidak menggunakan peralatan AWS,

maka Anda harus menandatangani sendiri permintaan tersebut. Untuk informasi selengkapnya tentang penggunaan metode yang disarankan untuk menandatangani permintaan sendiri, silakan lihat Menandatangani permintaan API AWS di Panduan Pengguna IAM.

Terlepas dari metode autentikasi yang Anda gunakan, Anda mungkin juga diminta untuk menyediakan informasi keamanan tambahan. Sebagai contoh, AWS menyarankan supaya Anda menggunakan autentikasi multi-faktor (MFA) untuk meningkatkan keamanan akun Anda. Untuk mempelajari selengkapnya, silakan lihat Autentikasi multi-faktor di Panduan Pengguna AWS IAM Identity Center dan Menggunakan autentikasi multi-faktor (MFA) di AWS di Panduan Pengguna IAM.

Pengguna root Akun AWS

Ketika Anda membuat Akun AWS, Anda memulai dengan satu identitas masuk yang memiliki akses ke semua Layanan AWS dan sumber daya di akun tersebut. Identitas ini disebut pengguna root Akun AWS dan diakses dengan cara masuk ke alamat email dan kata sandi yang Anda gunakan untuk membuat akun. Kami sangat menyarankan Anda tidak menggunakan pengguna root untuk tugas sehari-hari. Lindungi kredensial pengguna root Anda dan gunakan kredensial tersebut untuk melakukan tugas yang hanya dapat dilakukan pengguna root. Untuk daftar lengkap tugas yang mengharuskan Anda masuk sebagai pengguna root, silakan lihat Tugas yang memerlukan kredensial pengguna root di Panduan Pengguna IAM.

Identitas terfederasi

Praktik terbaiknya berupa, mewajibkan pengguna manusia, termasuk pengguna yang memerlukan akses administrator, untuk menggunakan federasi dengan penyedia identitas untuk mengakses Layanan AWS dengan menggunakan kredensial temporer.

Identitas terfederasi adalah pengguna dari direktori pengguna perusahaan Anda, penyedia identitas web, dikenal sebagai AWS Directory Service, direktori Pusat Identitas, atau pengguna mana pun yang mengakses Layanan AWS dengan menggunakan kredensial yang disediakan melalui sumber identitas. Ketika identitas terfederasi mengakses Akun AWS, identitas tersebut mengambil peran, dan peran memberikan kredensial temporer.

Untuk pengelolaan akses terpusat, kami sarankan Anda menggunakanAWS IAM Identity Center. Anda dapat membuat pengguna dan grup di Pusat Identitas IAM, atau Anda dapat menghubungkan dan menyinkronkan ke sekumpulan pengguna dan grup di sumber identitas Anda sendiri untuk digunakan di semua Akun AWS Anda dan aplikasi Anda. Untuk informasi tentang Pusat Identitas IAM, silakan lihat Apakah Pusat Identitas IAM itu? di User Guide AWS IAM Identity Center.

Pengguna dan Grup IAM

Pengguna IAM adalah identitas dalam Akun AWS Anda yang memiliki izin khusus untuk satu orang atau aplikasi. Apabila memungkinkan, kami menyarankan untuk mengandalkan pada kredensial temporer alih-alih membuat pengguna IAM yang memiliki kredensyal jangka panjang seperti kata sandi dan kunci akses. Namun, jika Anda memiliki kasus penggunaan khusus yang memerlukan kredensial jangka panjang dengan pengguna IAM, kami menyarankan Anda memutar kunci akses. Untuk informasi selengkapnya, silakan lihat Memutar kunci akses secara teratur untuk kasus penggunaan yang memerlukan kredensial jangka panjang di Panduan Pengguna IAM.

Grup IAM adalah identitas yang menerangkan secara spesifik kumpulan pengguna IAM. Anda tidak dapat masuk sebagai kelompok. Anda dapat menggunakan grup untuk menerangkan secara spesifik izin untuk beberapa pengguna sekaligus. Grup membuat izin lebih mudah dikelola untuk sekelompok besar pengguna. Sebagai contoh, Anda dapat memiliki grup yang diberi nama AdminIAM dan memberikan izin kepada grup tersebut untuk mengelola sumber daya IAM.

Pengguna berbeda dari peran. Pengguna secara unik terkait dengan satu orang atau aplikasi, tetapi peran tersebut dimaksudkan untuk dapat digunakan oleh siapa pun yang membutuhkannya. Pengguna memiliki kredensial jangka panjang permanen, tetapi peran memberikan kredensial temporer. Untuk mempelajari selengkapnya, silakan lihat Kapan harus membuat pengguna IAM (alih-alih peran) di Panduan Pengguna IAM.

Peran IAM

Peran IAM merupakan identitas dalam Akun AWS Anda yang memiliki izin khusus. Peran ini mirip dengan pengguna IAM, tetapi tidak terkait dengan orang tertentu. Anda dapat menggunakan peran IAM untuk sementara dalam AWS Management Console dengan berganti peran. Anda dapat mengambil peran dengan cara memanggil operasi API AWS CLI atau AWS atau menggunakan URL kustom. Untuk informasi selengkapnya tentang cara menggunakan peran, silakan lihat menggunakan peran IAM di Panduan Pengguna IAM.

IAM role dengan kredensial temporer berguna dalam situasi berikut:

- Akses pengguna gabungan – Untuk menetapkan izin ke sebuah identitas terfederasi, Anda harus membuat sebuah peran dan menentukan izin untuk peran tersebut. Ketika identitas gabungan terfederasi mengautentikasi, identitas tersebut terhubung dengan peran dan diberikan izin yang ditentukan oleh peran. Untuk informasi tentang peran-peran untuk federasi, silakan lihat Membuat sebuah peran untuk Penyedia Identitas pihak ketiga di Panduan Pengguna IAM. Jika Anda menggunakan Pusat Identitas IAM, Anda mengonfigurasi serangkain izin. Untuk mengontrol apa

yang dapat diakses identitas Anda setelah identitas tersebut diautentikasi, Pusat Identitas IAM mengkorelasikan izin yang diatur ke peran dalam IAM. Untuk informasi tentang rangkaian izin, silakan lihat Rangkaian izin di Panduan Pengguna AWS IAM Identity Center.

- Izin pengguna IAM sementara – Pengguna atau peran IAM dapat mengambil peran IAM untuk sementara mengambil izin berbeda untuk tugas tertentu.
- Akses lintas akun – Anda dapat menggunakan peran IAM untuk mengizinkan seseorang (pengguna utama tepercaya) di akun berbeda untuk mengakses sumber daya yang ada di akun Anda. Peran adalah cara utama untuk memberikan akses lintas akun. Namun, pada beberapa Layanan AWS, Anda dapat melampirkan kebijakan secara langsung ke sumber daya (alih-alih menggunakan suatu peran sebagai proksi). Untuk mempelajari perbedaan antara kebijakan peran dan kebijakan berbasis sumber daya untuk akses lintas akun, silakan lihat Bagaimana peran IAM role berbeda dari kebijakan berbasis sumber daya dalam Panduan Pengguna IAM.
- Akses lintas layanan – Sebagian Layanan AWS menggunakan fitur di Layanan AWS lainnya. Sebagai contoh, ketika Anda melakukan panggilan dalam suatu layanan, lazim pada layanan tersebut menjalankan aplikasi di Amazon EC2 atau menyimpan objek di Amazon S3. Suatu layanan mungkin melakukan hal tersebut menggunakan izin pengguna utama panggilan, menggunakan peran layanan, atau peran tertaut layanan.
  - Sesi akses maju (FAS) – Ketika Anda menggunakan pengguna IAM atau peran IAM untuk melakukan tindakan-tindakan di AWS, Anda akan dianggap sebagai seorang pengguna utama. Ketika Anda menggunakan beberapa layanan, Anda mungkin melakukan sebuah tindakan yang kemudian dilanjutkan oleh tindakan lain pada layanan yang berbeda. FAS menggunakan izin dari pengguna utama untuk memanggil Layanan AWS, yang dikombinasikan dengan Layanan AWS yang diminta untuk membuat pengajuan ke layanan hilir. Permintaan FAS hanya diajukan ketika sebuah layanan menerima pengajuan yang memerlukan interaksi dengan Layanan AWS lain atau sumber daya lain untuk diselesaikan. Dalam hal ini, Anda harus memiliki izin untuk melakukan kedua tindakan tersebut. Untuk detail kebijakan ketika mengajukan permintaan FAS, silakan lihat Meneruskan sesi akses.
  - Peran layanan – Sebuah peran layanan adalah sebuah peran IAM yang dijalankan oleh suatu layanan untuk melakukan tindakan atas nama Anda. Administrator IAM dapat membuat, memodifikasi, dan menghapus peran layanan dari dalam IAM. Untuk informasi selengkapnya, silakan lihat Membuat sebuah peran untuk mendelegasikan izin ke Layanan AWS dalam Panduan pengguna IAM.
  - Peran tertaut layanan – Peran tertaut layanan adalah tipe peran layanan yang tertaut dengan Layanan AWS. Layanan tersebut dapat menjalankan peran untuk melakukan sebuah tindakan atas nama Anda. Peran tertaut layanan akan muncul di Akun AWS Anda dan dimiliki oleh

layanan tersebut. Administrator IAM dapat melihat, tetapi tidak dapat mengedit izin untuk peran tertaut layanan.

- Aplikasi yang berjalan di Amazon EC2 – Anda dapat menggunakan peran IAM untuk mengelola kredensial temporer untuk aplikasi yang berjalan di instans EC2 dan mengajukan permintaan AWS CLI atau API AWS. Cara ini lebih baik daripada menyimpan kunci akses dalam instans EC2. Untuk menugaskan sebuah peran AWS ke instans EC2 dan membuatnya tersedia untuk semua aplikasinya, Anda dapat membuat sebuah profil instans yang dilampirkan ke instans. Profil instans berisi peran dan memungkinkan program yang berjalan di instans EC2 untuk mendapatkan kredensial sementara. Untuk informasi selengkapnya, silakan lihat Menggunakan peran IAM untuk memberikan izin ke aplikasi yang berjalan di instans Amazon EC2 di Panduan Pengguna IAM.

Untuk mempelajari apakah kita harus menggunakan peran IAM atau pengguna IAM, silakan lihat Kapan harus membuat peran IAM (alih-alih pengguna) di Panduan Pengguna IAM.

Mengelola kebijakan menggunakan akses

Anda mengendalikan akses di AWS dengan membuat kebijakan dan melampirkannya ke identitas atau sumber daya AWS. Kebijakan adalah objek di AWS yang, ketika terkait dengan identitas atau sumber daya, akan menentukan izinnya. AWS mengevaluasi kebijakan-kebijakan tersebut ketika seorang pengguna utama (pengguna, root user, atau sesi peran) mengajukan permintaan. Izin dalam kebijakan menentukan apakah permintaan diberikan atau ditolak. Sebagian besar kebijakan disimpan di AWS sebagai dokumen JSON. Untuk informasi selengkapnya tentang struktur dan isi dokumen kebijakan JSON, silakan lihat Gambaran Umum kebijakan JSON di Panduan Pengguna IAM.

Administrator dapat menggunakan kebijakan JSON AWS untuk menentukan secara spesifik siapa yang memiliki akses pada apa. Yaitu, pengguna utama manakah yang dapat melakukan tindakan pada sumber daya apa, dan dengan syarat apa.

Secara bawaan, para pengguna dan peran tidak memiliki izin. Untuk mengabulkan izin kepada pengguna untuk melakukan tindakan di sumber daya yang mereka perlukan, administrator IAM dapat membuat kebijakan IAM. Administrator kemudian akan dapat menambahkan kebijakan IAM ke peran, dan para pengguna dapat mengambil peran.

Kebijakan IAM mendefinisikan izin untuk suatu tindakan terlepas dari metode yang Anda gunakan untuk pengoperasiannya. Sebagai contoh, anggap saja Anda memiliki kebijakan yang mengizinkan tindakan `iam:GetRole` . Pengguna dengan kebijakan tersebut dapat memperoleh informasi peran dari AWS Management Console, AWS CLI, atau APIAWS.

Kebijakan berbasis identitas

Kebijakan berbasis identitas adalah dokumen kebijakan izin JSON yang dapat Anda lampirkan ke sebuah identitas, misalnya pengguna IAM, grup pengguna IAM, atau peran IAM. Kebijakan ini mengontrol apa yang pengguna tindakan dan peran dapat kerjakan, pada sumber daya mana, dan dalam keadaan apa. Untuk mempelajari cara membuat kebijakan berbasis identitas, silakan lihat Membuat kebijakan IAM dalam Panduan Pengguna IAM.

Kebijakan berbasis identitas dapat dikategorikan lebih lanjut sebagai kebijakan inline atau kebijakan terkelola. Kebijakan inline ditanam secara langsung ke pengguna tunggal, grup, atau peran. Kebijakan terkelola adalah kebijakan yang berdiri sendiri yang dapat Anda lampirkan ke beberapa pengguna, grup, dan peran di Akun AWS Anda. Kebijakan terkelola mencakup kebijakan terkelola AWS dan kebijakan terkelola pelanggan. Untuk mempelajari cara memilih antara kebijakan terkelola atau kebijakan inline, silakan lihat Memilih antara kebijakan terkelola dan kebijakan inline diPanduan Pengguna IAM.

Kebijakan berbasis sumber daya

Kebijakan berbasis sumber daya adalah dokumen kebijakan JSON yang Anda lampirkan ke sumber daya. Contoh kebijakan-kebijakan berbasis sumber daya adalah kebijakan terpercaya 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 yang dilampiri kebijakan, kebijakan tersebut menentkan tindakan apa yang dapat dilakukan oleh pengguna utama yang ditentukan di sumber daya tersebut dan dalam kondisi apa. Anda harus menentukan pengguna utama dalam kebijakan berbasis sumber daya. Pengguna utama dapat mencakup akun, pengguna, peran, pengguna gabungan, atau Layanan AWS.

Kebijakan berbasis sumber daya merupakan kebijakan inline yang terletak di layanan tersebut. Anda tidak dapat menggunakan kebijakan terkelola AWS dari IAM dalam kebijakan berbasis sumber daya.

Daftar kontrol akses (ACL)

Daftar kontrol akses (ACL) mengendalikan pengguna utama mana (anggota akun, pengguna, atau peran) yang memiliki izin untuk mengakses sumber daya. ACL serupa dengan kebijakan berbasis sumber daya, meskipun kebijakan-kebijakan tersebut tidak menggunakan format dokumen kebijakan JSON.

Amazon S3, AWS WAF, dan Amazon VPC adalah contoh-contoh layanan yang mendukung ACL. Untuk mempelajari ACL selengkapnya, silakan lihat Gambaran umum daftar kontrol akses (ACL) di Panduan Pengembang Layanan Penyimpanan Ringkas Amazon.

Tipe-tipe kebijakan lain

AWS mendukung tipe kebijakan tambahan, yang kurang umum. Tipe-tipe kebijakan ini dapat mengatur izin maksimum yang diberikan kepada Anda oleh tipe kebijakan yang lebih umum.

- Batasan izin – Batasan izin adalah fitur lanjutan tempat Anda mengatur izin maksimum yang dapat diberikan oleh kebijakan berbasis identitas kepada entitas IAM (pengguna IAM atau peran IAM). Anda dapat menetapkan batasan izin untuk suatu entitas. Izin yang dihasilkan adalah perpotongan antara kebijakan berbasis identitas milik entitas dan batas izinnya. Kebijakan berbasis sumber daya yang menentukan pengguna atau peran dalam bidang `Principal` tidak dibatasi oleh batasan izin. Penolakan eksplisit dalam salah satu kebijakan ini menindihi izin. Untuk informasi selengkapnya tentang batasan izin, silakan lihat Batasan izin untuk entitas IAM di Panduan Pengguna IAM.
- Kebijakan kontrol layanan (SCP) – SCP adalah kebijakan JSON yang menentukan izin maksimum untuk sebuah organisasi atau unit organisasi (OU) di AWS Organizations. AWS Organizations adalah layanan untuk mengelompokkan dan secara terpusat mengelola beberapa Akun AWS yang dimiliki bisnis Anda. Jika Anda mengaktifkan semua fitur di suatu organisasi, maka Anda dapat menerapkan kebijakan kontrol layanan (SCP) ke salah satu atau ke semua akun Anda. SCP membatasi izin untuk entitas dalam akun anggota, termasuk setiap Pengguna root akun AWS. Untuk informasi selengkapnya tentang Organisasi dan SCP, silakan lihat Cara kerja SCP di Panduan Pengguna AWS Organizations.
- Kebijakan sesi – Kebijakan sesi adalah kebijakan lanjutan yang Anda berikan sebagai parameter ketika Anda membuat sesi sementara secara terprogram untuk peran atau pengguna gabungan. Izin sesi yang dihasilkan adalah perpotongan kebijakan berbasis identitas pengguna atau peran dan kebijakan sesi. Izin juga dapat berasal dari kebijakan berbasis sumber daya. Penolakan eksplisit dalam salah satu kebijakan ini menindihi izin. Untuk informasi selengkapnya, silakan lihat Kebijakan sesi di Panduan Pengguna IAM.

Berbagai tipe kebijakan

Ketika beberapa tipe kebijakan berlaku pada suatu permintaan, izin yang dihasilkan lebih rumit untuk dipahami. Untuk mempelajari cara AWS menentukan apakah akan mengizinkan permintaan ketika beberapa tipe kebijakan dilibatkan, silakan lihat Logika evaluasi kebijakan di Panduan Pengguna IAM.

Bagaimana Amazon Lex bekerja dengan IAM

Sebelum Anda menggunakan IAM untuk mengelola akses ke Amazon Lex, pelajari fitur IAM apa yang tersedia untuk digunakan dengan Amazon Lex.

Fitur IAM yang dapat Anda gunakan dengan Amazon Lex

Fitur IAM	Dukungan Amazon Lex
Kebijakan berbasis identitas	Ya
Kebijakan berbasis sumber daya	Tidak
Tindakan kebijakan	Ya
Sumber daya kebijakan	Ya
kunci-kunci persyaratan kebijakan (spesifik layanan)	Ya
ACL	Tidak
ABAC (tag dalam kebijakan)	Parsial
Kredensial temporer	Ya
Izin-izin pengguna utama	Ya
Peran layanan	Ya
Peran tertaut layanan	Ya

Untuk mendapatkan tampilan tingkat tinggi tentang cara kerja Amazon Lex dan AWS layanan lainnya dengan sebagian besar fitur IAM, lihat AWSlayanan yang bekerja dengan IAM di Panduan Pengguna IAM.

Kebijakan berbasis identitas untuk Amazon Lex

Mendukung kebijakan berbasis identitas	Ya

Kebijakan berbasis identitas adalah dokumen kebijakan izin JSON yang dapat Anda lampirkan ke sebuah identitas, misalnya pengguna IAM, grup pengguna IAM, atau peran IAM. Kebijakan ini mengontrol apa yang pengguna tindakan dan peran dapat kerjakan, pada sumber daya mana, dan dalam keadaan apa. Untuk mempelajari cara membuat kebijakan berbasis identitas, silakan lihat Membuat kebijakan IAM dalam Panduan Pengguna IAM.

Dengan kebijakan berbasis identitas IAM, Anda dapat menentukan secara spesifik apakah tindakan dan sumber daya diizinkan atau ditolak, serta persyaratan yang menjadi dasar dikabulkan atau ditolaknya tindakan tersebut. Anda tidak dapat menentukan secara spesifik pengguna utama dalam sebuah kebijakan berbasis identitas karena pengguna utama berlaku bagi pengguna atau peran yang melekat kepadanya. Untuk mempelajari semua elemen yang dapat Anda gunakan dalam kebijakan JSON, silakan lihat Referensi elemen kebijakan JSON IAM dalam Panduan Pengguna IAM.

Contoh kebijakan berbasis identitas untuk Amazon Lex

Untuk melihat contoh kebijakan berbasis identitas Amazon Lex, lihat. Contoh kebijakan berbasis identitas untuk Amazon Lex

Kebijakan berbasis sumber daya dalam Amazon Lex

Mendukung kebijakan berbasis sumber daya	Tidak

Kebijakan berbasis sumber daya adalah dokumen kebijakan JSON yang Anda lampirkan ke sumber daya. Contoh kebijakan-kebijakan berbasis sumber daya adalah kebijakan terpercaya 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 yang dilampiri kebijakan, kebijakan tersebut menentkan tindakan apa yang dapat dilakukan oleh pengguna utama yang ditentukan di sumber daya tersebut dan dalam kondisi apa. Anda harus menentukan pengguna utama dalam kebijakan berbasis sumber daya. Pengguna utama dapat mencakup akun, pengguna, peran, pengguna gabungan, atau Layanan AWS.

Untuk mengaktifkan akses lintas akun, Anda dapat menentukan secara spesifik seluruh akun atau entitas IAM di akun lain sebagai pengguna utama dalam kebijakan berbasis sumber daya. Menambahkan pengguna utama akun silang ke kebijakan berbasis sumber daya hanya setengah dari membangun hubungan kepercayaan. Ketika pengguna utama dan sumber daya berada dalam Akun AWS yang berbeda, Administrator IAM di akun tepercaya juga harus memberikan izin kepada entitas pengguna utama (pengguna atau peran) untuk mengakses sumber daya. Mereka

memberikan izin melampirkan kebijakan berbasis identitas kepada entitas. Namun, jika kebijakan berbasis sumber daya memberikan akses kepada pengguna utama dalam akun yang sama, tidak diperlukan kebijakan berbasis identitas tambahan. Untuk informasi selengkapnya, silakan lihat Bagaimana peran IAM berbeda dari kebijakan berbasis sumber daya di Panduan Pengguna IAM.

Tindakan kebijakan untuk Amazon Lex

Mendukung tindakan kebijakan	Ya

Administrator dapat menggunakan kebijakan JSON AWS untuk menentukan secara spesifik siapa yang memiliki akses pada apa. Yaitu, pengguna utama manakah yang dapat melakukan tindakan pada sumber daya apa, dan dengan syarat apa.

Elemen `Action` dari kebijakan JSON menjelaskan tindakan-tindakan yang dapat Anda gunakan untuk mengizinkan atau menolak akses dalam sebuah kebijakan. Tindakan-tindakan kebijakan biasanya memiliki nama yang sama sebagaimana operasi API AWS yang dikaitkan padanya. Ada beberapa pengecualian, misalnya tindakan yang memiliki izin saja yang tidak memiliki operasi API yang cocok. Ada juga beberapa operasi yang memerlukan beberapa tindakan dalam sebuah kebijakan. Tindakan-tindakan tambahan ini disebut tindakan dependen.

Menyertakan tindakan dalam suatu kebijakan untuk memberikan izin guna melakukan operasi yang terkait.

Untuk melihat daftar tindakan Amazon Lex, lihat Tindakan yang ditentukan oleh Amazon Lex di Referensi Otorisasi Layanan.

Tindakan kebijakan di Amazon Lex menggunakan awalan berikut sebelum tindakan:

```
lex
```

Untuk menetapkan secara spesifik beberapa tindakan dalam satu pernyataan, pisahkan tindakan-tindakan tersebut dengan koma.

```
"Action": [
      "lex:action1",
      "lex:action2"
         ]
```

Anda juga dapat menentukan beberapa tindakan menggunakan wildcard (*). Sebagai contoh, untuk menentukan semua tindakan yang dimulai dengan kata `Describe`, sertakan tindakan berikut:

```
"Action": "lex:Describe*"
```

Sumber daya kebijakan untuk Amazon Lex

Mendukung sumber daya kebijakan	Ya

Administrator dapat menggunakan kebijakan JSON AWS untuk menentukan secara spesifik siapa yang memiliki akses pada apa. Yaitu, pengguna utama manakah yang dapat melakukan tindakan pada sumber daya apa, dan dengan syarat apa.

Elemen kebijakan JSON `Resource` menentukan objek atau objek-objek yang menjadi target penerapan tindakan. Pernyataan harus menyertakan entah elemen `Resource` atau `NotResource`. Praktik terbaiknya, tentukan sumber daya menggunakan Amazon Resource Name (ARN). Anda dapat melakukan ini untuk tindakan-tindakan yang mendukung jenis sumber daya tertentu, yang dikenal sebagai izin tingkat sumber daya.

Untuk tindakan yang tidak mendukung izin di tingkat sumber daya, misalnya operasi pencantuman, gunakan wildcard (*) untuk mengindikasikan bahwa pernyataan tersebut berlaku bagi semua sumber daya.

```
"Resource": "*"
```

Sumber daya bot Amazon Lex ARN memiliki format berikut.

```
arn:aws:lex:${Region}:${Account}:bot:${Bot-Name}
```

Untuk informasi lebih lanjut tentang format ARN, lihat Amazon Resource Name (ARN) dan Namespace Layanan AWS.

Misalnya, untuk menentukan `OrderFlowers` bot dalam pernyataan Anda, gunakan ARN berikut.

```
"Resource": "arn:aws:lex:us-east-2:123456789012:bot:OrderFlowers"
```

Untuk menentukan semua bot milik akun tertentu, gunakan wildcard (*).

```
"Resource": "arn:aws:lex:us-east-2:123456789012:bot:*"
```

Beberapa tindakan Amazon Lex, seperti untuk membuat sumber daya, tidak dapat dilakukan pada sumber daya tertentu. Dalam kasus tersebut, Anda harus menggunakan wildcard, (*).

```
"Resource": "*"
```

Untuk melihat daftar jenis sumber daya Amazon Lex dan ARNnya, lihat Sumber daya yang ditentukan oleh Amazon Lex di Referensi Otorisasi Layanan. Untuk mempelajari tindakan mana yang dapat Anda tentukan ARN dari setiap sumber daya, lihat Tindakan yang ditentukan oleh Amazon Lex.

Kunci kondisi kebijakan untuk Amazon Lex

Mendukung kunci-kunci persyaratan kebijakan spesifik layanan	Ya

Administrator dapat menggunakan kebijakan JSON AWS untuk menentukan secara spesifik siapa yang memiliki akses pada apa. Yaitu, pengguna utama manakah yang dapat melakukan tindakan pada sumber daya apa, dan dengan syarat apa.

Elemen `Condition` (atau blok `Condition`) akan memungkinkan Anda menentukan syarat yang menjadi dasar suatu pernyataan berlaku. Elemen `Condition` bersifat opsional. Anda dapat membuat ekspresi bersyarat yang menggunakan operator syarat, misalnya sama dengan atau kurang dari, untuk mencocokkan syarat dalam kebijakan dengan nilai-nilai yang diminta.

Jika Anda menentukan beberapa elemen `Condition` dalam sebuah pernyataan, atau beberapa kunci dalam elemen `Condition` tunggal, maka AWS akan mengevaluasinya dengan menggunakan operasi `AND` yang logis. Jika Anda menentukan beberapa niilai untuk satu kunci persyaratan, maka AWS akan mengevaluasi syarat tersebut menggunakan operasi `OR` yang logis. Semua persyaratan harus dipenuhi sebelum izin pernyataan diberikan.

Anda juga dapat menggunakan variabel placeholder saat menentukan syarat. Sebagai contoh, Anda dapat memberikan izin kepada pengguna IAM untuk mengakses sumber daya hanya jika izin tersebut mempunyai tag yang sesuai dengan nama pengguna IAM mereka. Untuk informasi selengkapnya, silakan lihat Elemen kebijakan IAM: variabel dan tag di Panduan Pengguna IAM.

AWS mendukung kunci-kunci syarat global dan kunci-kunci syarat spesifik layanan. Untuk melihat semua kunci persyaratan global AWS, silakan lihat kunci konteks syarat global AWS di Panduan Pengguna IAM.

Untuk melihat daftar kunci kondisi Amazon Lex, lihat Kunci kondisi untuk Amazon Lex di Referensi Otorisasi Layanan. Untuk mempelajari tindakan dan sumber daya yang dapat Anda gunakan kunci kondisi, lihat Tindakan yang ditentukan oleh Amazon Lex.

Tabel berikut mencantumkan kunci kondisi Amazon Lex yang berlaku untuk sumber daya Amazon Lex. Anda dapat menyertakan kunci ini dalam `Condition` elemen dalam kebijakan izin IAM.

ACL di Amazon Lex

Mendukung ACL	Tidak

Daftar kontrol akses (ACL) mengendalikan pengguna utama mana (anggota akun, pengguna, atau peran) yang memiliki izin untuk mengakses sumber daya. ACL serupa dengan kebijakan berbasis sumber daya, meskipun kebijakan-kebijakan tersebut tidak menggunakan format dokumen kebijakan JSON.

ABAC dengan Amazon Lex

Mendukung ABAC (tag dalam kebijakan)	Parsial

Kontrol akses berbasis atribut (ABAC) adalah strategi otorisasi yang menentukan izin berdasarkan atribut. Di AWS, atribut-atribut ini disebut tag. Anda dapat melampirkan tag ke entitas IAM (pengguna atau peran) dan ke banyak sumber daya AWS. Pemberian tag ke entitas dan sumber daya adalah langkah pertama dari ABAC. Kemudian rancanglah kebijakan ABAC untuk mengizinkan operasi-operasi ketika tag milik pengguna utama cocok dengan tag yang ada di sumber daya yang ingin diakses.

ABAC sangat berguna di lingkungan yang berkembang dengan cepat dan berguna di situasi dimana pengelolaan kebijakan menjadi rumit.

Untuk mengendalikan akses berdasarkan tag, berikan informasi tentang tag di elemen syarat dari sebuh kebijakan dengan menggunakan kunci-kunci persyaratan `aws:ResourceTag/`*`key-name`*, `aws:RequestTag/`*`key-name`*, atau `aws:TagKeys`.

Jika sebuah layanan mendukung ketiga kunci-kunci persyaratan untuk setiap jenis sumber daya, maka nilainya adalah Ya untuk layanan tersebut. Jika suatu layanan mendukung ketiga kunci persyaratan untuk hanya beberapa jenis sumber daya, maka nilainya adalah Parsial.

Untuk informasi selengkapnya tentang ABAC, silakan lihat Apa itu ABAC? di Panduan Pengguna IAM. Untuk melihat tutorial yang menguraikan langkah-langkah pengaturan ABAC, silakan lihat Menggunakan kontrol akses berbasis atribut (ABAC) di Panduan Pengguna IAM.

Anda dapat mengaitkan tag dengan jenis sumber daya Amazon Lex tertentu untuk otorisasi. Untuk mengontrol akses berdasarkan tag, berikan informasi tag dalam elemen kondisi kebijakan dengan menggunakan`lex:ResourceTag/${TagKey}`,`aws:RequestTag/${TagKey}`, atau kunci `aws:TagKeys` kondisi.

Untuk informasi selengkapnya tentang menandai sumber daya Amazon Lex, lihatMenandai Sumber Daya Amazon Lex.

Untuk melihat contoh kebijakan berbasis identitas untuk membatasi akses ke sumber daya berdasarkan tag pada sumber daya tersebut, lihat Menggunakan Tag untuk Mengakses Sumber Daya.

Tabel berikut mencantumkan tindakan dan jenis sumber daya yang sesuai untuk kontrol akses berbasis tag. Setiap tindakan diotorisasi berdasarkan tanda yang terkait dengan jenis sumber daya yang sesuai.

Tindakan	Tipe sumber daya	Kunci syarat	Catatan
CreateBotVersion	bot	`lex:ResourceTag`	
DeleteBot	bot	`lex:ResourceTag`	
DeleteBotAlias	alias	`lex:ResourceTag`	
DeleteBotChannelAssociation	saluran	`lex:ResourceTag`	
DeleteBotVersion	bot	`lex:ResourceTag`	
DeleteSession	bot atau alias	`lex:ResourceTag`	Menggunakan tag yang terkait dengan bot saat alias

Tindakan	Tipe sumber daya	Kunci syarat	Catatan
			disetel ke$LATEST. Menggunakan tag yang terkait dengan alias yang ditentuka n saat digunakan dengan alias lain.
DeleteUtterances	bot	lex:ResourceTag	
GetBot	bot atau alias	lex:ResourceTag	Menggunakan tag yang terkait dengan bot saat versionOr Alias diatur ke $LATEST atau versi numerik. Menggunak an tag yang terkait dengan alias yang ditentukan saat digunakan dengan alias
GetBotAlias	alias	lex:ResourceTag	
GetBotChannelAssoc iation	chanel	lex:ResourceTag	
GetBotChannelAssoc iations	chanel	lex:ResourceTag	Menggunakan tag yang terkait dengan bot ketika alias diatur ke “-”. Menggunak an tag yang terkait dengan alias yang ditentukan ketika alias bot ditentukan
GetBotVersions	bot	lex:ResourceTag	

Tindakan	Tipe sumber daya	Kunci syarat	Catatan
GetExport	bot	`lex:ResourceTag`	
GetSession	bot atau alias	`lex:ResourceTag`	Menggunakan tag yang terkait dengan bot saat alias disetel ke`$LATEST`. Menggunakan tag yang terkait dengan alias yang ditentukan saat digunakan dengan alias lain.
GetUtterancesView	bot	`lex:ResourceTag`	
ListTagsForResource	bot, alias, atau saluran	`lex:ResourceTag`	
PostContent	bot atau alias	`lex:ResourceTag`	Menggunakan tag yang terkait dengan bot saat alias disetel ke`$LATEST`. Menggunakan tag yang terkait dengan alias yang ditentukan saat digunakan dengan alias lain.
PostText	bot atau alias	`lex:ResourceTag`	Menggunakan tag yang terkait dengan bot saat alias disetel ke`$LATEST`. Menggunakan tag yang terkait dengan alias yang ditentukan saat digunakan dengan alias lain.

Tindakan	Tipe sumber daya	Kunci syarat	Catatan
PutBot	bot	`lex:ResourceTag`, `aws:RequestTag`, `aws:TagKeys`	
PutBotAlias	alias	`lex:ResourceTag`, `aws:RequestTag`, `aws:TagKeys`	
PutSession	bot atau alias	`lex:ResourceTag`	Menggunakan tag yang terkait dengan bot saat alias disetel ke`$LATEST`. Menggunakan tag yang terkait dengan alias yang ditentukan saat digunakan dengan alias lain.
StartImport	bot	`lex:ResourceTag`	Bergantung pada kebijakan akses untuk `PutBot` operasi. Tag dan izin khusus untuk `StartImport` operasi diabaikan.
TagResource	bot, alias, atau saluran	`lex:ResourceTag`, `aws:RequestTag`, `aws:TagKeys`	

Tindakan	Tipe sumber daya	Kunci syarat	Catatan
UntagResource	bot, alias, atau saluran	`lex:ResourceTag`, `aws:RequestTag`, `aws:TagKeys`	

Menggunakan kredensyal sementara dengan Amazon Lex

Mendukung kredensial temporer	Ya

Beberapa Layanan AWS tidak berfungsi saat Anda masuk dengan menggunakan kredensial temporer. Sebagai informasi tambahan, termasuk tentang Layanan AWS mana saja yang berfungsi dengan kredensial temporer, silakan lihat Layanan AWS yang berfungsi dengan IAM di Panduan Pengguna IAM.

Anda menggunakan kredensial temporer jika Anda masuk ke AWS Management Console dengan menggunakan metode apa pun kecuali nama pengguna dan kata sandi. Sebagai contoh, ketika Anda mengakses AWS dengan menggunakan tautan masuk tunggal (SSO) milik perusahaan Anda, proses itu secara otomatis akan membuat kredensial temporer. Anda juga akan secara otomatis membuat kredensial temporer ketika Anda masuk ke konsol sebagai seorang pengguna dan kemudian beralih peran. Untuk informasi selengkapnya tentang peralihan peran, silakan lihat Peralihan peran (konsol) di Panduan Pengguna IAM.

Anda dapat secara manual membuat kredensial temporer menggunakan AWS CLI atau API AWS. Anda kemudian dapat menggunakan kredensial temporer tersebut untuk mengakses AWS. AWS menyarankan agar Anda secara dinamis membuat kredensial temporer alih-alih menggunakan kunci akses jangka panjang. Untuk informasi selengkapnya, silakan lihat Kredensial keamanan temporer di IAM.

Anda dapat menggunakan kredensial sementara untuk masuk dengan gabungan, menjalankan IAM role, atau menjalankan peran lintas akun. Anda memperoleh kredensil keamanan sementara dengan memanggil operasi AWS STS API seperti AssumeRoleatau. GetFederationToken

Izin utama lintas layanan untuk Amazon Lex

Mendukung sesi akses maju (FAS)	Ya

Saat Anda menggunakan pengguna IAM atau peran IAM untuk mengerjakan tindakan di AWS, Anda akan dianggap sebagai pengguna utama. Ketika Anda menggunakan beberapa layanan, Anda mungkin melakukan sebuah tindakan yang kemudian dilanjutkan oleh tindakan lain pada layanan yang berbeda. FAS menggunakan izin dari pengguna utama untuk memanggil Layanan AWS, yang dikombinasikan dengan Layanan AWS yang diminta untuk membuat pengajuan ke layanan hilir. Permintaan FAS hanya diajukan ketika sebuah layanan menerima pengajuan yang memerlukan interaksi dengan Layanan AWS lain atau sumber daya lain untuk diselesaikan. Dalam hal ini, Anda harus memiliki izin untuk melakukan kedua tindakan tersebut. Untuk detail kebijakan ketika mengajukan permintaan FAS, silakan lihat Meneruskan sesi akses.

Peran layanan untuk Amazon Lex

Mendukung peran layanan	Ya

Peran layanan adalah sebuah peran IAM yang diambil oleh sebuah layanan untuk melakukan tindakan atas nama Anda. Administrator IAM dapat membuat, memodifikasi, dan menghapus peran layanan dari dalam IAM. Untuk informasi selengkapnya, lihat Membuat peran untuk mendelegasikan izin ke Layanan AWS dalam Panduan pengguna IAM.

> **Warning**
>
> Mengubah izin untuk peran layanan dapat merusak fungsionalitas Amazon Lex. Edit peran layanan hanya jika Amazon Lex memberikan panduan untuk melakukannya.

Memilih peran IAM di Amazon Lex

Amazon Lex menggunakan peran terkait layanan untuk memanggil Amazon Comprehend dan Amazon Polly. Ini menggunakan izin tingkat sumber daya pada fungsi Anda AWS Lambda untuk memanggilnya.

Anda harus memberikan peran IAM untuk mengaktifkan penandaan percakapan. Untuk informasi selengkapnya, lihat Membuat Kebijakan IAM Role dan Kebijakan untuk Log Percakapan.

Peran terkait layanan untuk Amazon Lex

Mendukung peran yang terhubung dengan layanan	Ya

Peran yang tertaut layanan adalah jenis peran layanan yang tertaut dengan Layanan AWS. Layanan tersebut dapat menjalankan peran untuk melakukan sebuah tindakan atas nama Anda. Peran tertaut layanan akan muncul di Akun AWS Anda dan dimiliki oleh layanan tersebut. Administrator IAM dapat melihat, tetapi tidak dapat mengedit izin untuk peran tertaut layanan.

Untuk detail tentang membuat atau mengelola peran terkait layanan Amazon Lex, lihat. Menggunakan Peran tertaut layanan untuk Amazon Lex

Contoh kebijakan berbasis identitas untuk Amazon Lex

Secara default, pengguna dan peran tidak memiliki izin untuk membuat atau memodifikasi sumber daya Amazon Lex. Pengguan dan peran tersebut juga tidak dapat melakukan tugas dengan menggunakan API AWS Management Console, AWS Command Line Interface (AWS CLI), atau AWS. Untuk mengabulkan izin kepada pengguna untuk melakukan tindakan di sumber daya yang mereka perlukan, administrator IAM dapat membuat kebijakan IAM. Administrator kemudian akan dapat menambahkan kebijakan IAM ke peran, dan para pengguna dapat mengambil peran.

Untuk mempelajari cara membuat kebijakan berbasis identitas IAM menggunakan contoh dokumen kebijakan JSON ini, silakan lihat Membuat kebijakan IAM di Panduan Pengguna IAM.

Untuk detail tentang tindakan dan jenis sumber daya yang ditentukan oleh Amazon Lex, termasuk format ARN untuk setiap jenis sumber daya, lihat Tindakan, sumber daya, dan kunci kondisi untuk Amazon Lex di Referensi Otorisasi Layanan.

Topik

- Praktik terbaik kebijakan
- Menggunakan konsol Amazon Lex
- Izinkan para pengguna untuk melihat izin mereka sendiri

- Hapus Semua Bot Amazon Lex
- Izinkan pengguna untuk memigrasikan bot ke Amazon Lex V2 API
- Menggunakan Tag untuk Mengakses Sumber Daya

Praktik terbaik kebijakan

Kebijakan berbasis identitas menentukan apakah seseorang dapat membuat, mengakses, atau menghapus sumber daya Amazon Lex di akun Anda. Tindakan ini mengenakan biaya kepada Anda Akun AWS. Ketika Anda membuat atau mengedit kebijakan berbasis identitas, ikuti panduan dan rekomendasi ini:

- Mulailah dengan kebijakan terkelola AWS dan beralih ke izin dengan hak akses paling rendah – Untuk mulai memberikan izin kepada pengguna dan beban kerja Anda, gunakan kebijakan terkelola AWS yang memberikan izin untuk banyak kasus penggunaan umum. Kebijakan tedapat di Akun AWS Anda. Kami menyarankan Anda untuk mengurangi izin lebih lanjut dengan menentukan kebijakan yang dikelola pelanggan AWS yang khusus untuk kasus penggunaan Anda. Untuk informasi selengkapnya, silakan lihat kebijakan-kebijakan terkelola AWS atau kebijakan-kebijakan terkelola AWS untuk fungsi tugas 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 melakukan ini dengan mendefinisikan tindakan yang dapat diambil pada sumber daya tertentu dalam kondisi tertentu, juga dikenal sebagai izin dengan hak akses paling rendah. Untuk informasi selengkapnya tentang cara menggunakan pengguna IAM untuk mengajukan izin, silakan lihat Kebijakan dan izin di IAM di Panduan Pengguna IAM.
- Gunakan syarat dalam kebijakan IAM untuk membatasi akses lebih lanjut – Anda dapat menambahkan suatu syarat ke kebijakan Anda untuk membatasi akses ke tindakan dan sumber daya. Sebagai contoh, Anda dapat menulis syarat kebijakan untuk menentukan bahwa semua pengajuan harus dikirim menggunakan SSL. Anda juga dapat menggunakan syarat untuk memberi akses ke tindakan layanan jika digunakan melalui Layanan AWS yang spesifk, sepertiAWS CloudFormation. Untuk informasi selengkapnya, silakan lihat Elemen kebijakan JSON IAM: Syarat di Panduan Pengguna IAM.
- Gunakan Analizer Akses IAM untuk memvalidasi kebijakan IAM Anda untuk memastikan izin yang aman dan fungsional – Analizer Akses IAM memvalidasi kebijakan baru dan yang sudah ada sehingga kebijakan tersebut mematuhi bahasa kebijakan IAM (JSON) dan praktik terbaik IAM. Analizer Akses IAM menyediakan lebih dari 100 pemeriksaan kebijakan dan rekomendasi yang dapat ditindaklanjuti untuk membantu Anda membuat kebijakan yang aman dan fungsional. Untuk

informasi selengkapnya, silakan lihat validasi kebijakan Analizer Akses IAM di Panduan Pengguna IAM.

- Memerlukan autentikasi multi-faktor (MFA) – Jika Anda memiliki skenario yang mengharuskan pengguna IAM atau pengguna root di Akun AWS Anda, aktifkan MFA untuk keamanan tambahan. Untuk meminta MFA ketika operasi API dipanggil, tambahkan syarat MFA pada kebijakan Anda. Untuk informasi selengkapnya, silakan lihat Mengonfigurasi akses API yang diproteksi MFA di Panduan Pengguna IAM.

Untuk informasi selengkapnya tentang praktik terbaik dalam IAM, silakan lihat Praktik terbaik keamanan di IAM di Panduan Pengguna IAM.

Menggunakan konsol Amazon Lex

Untuk mengakses konsol Amazon Lex, Anda harus memiliki set izin minimum. Izin ini harus memungkinkan Anda untuk membuat daftar dan melihat detail tentang sumber daya Amazon Lex di AndaAkun 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 meloloskan izin konsol minimum bagi pengguna yang hanya melakukan panggilan ke API AWS CLI atau AWS. Jika tidak, akses hanya diizinkan ke tindakan-tindakan yang sesuai dengan operasi API yang sedang mereka coba lakukan.

AWS menangani banyak kasus penggunaan umum dengan menyediakan kebijakan IAM mandiri yang dibuat dan dikelola oleh AWS. Kebijakan ini disebut kebijakan yang dikelola AWS. Kebijakan terkelola AWS membuat lebih mudah bagi anda untuk menetapkan izin yang sesuai untuk pengguna, grup, dan peran dibandingkan jika Anda harus menulis kebijakan tersebut sendiri. Untuk informasi selengkapnya, lihat Kebijakan yang Dikelola AWS dalam Panduan Pengguna IAM.

Kebijakan AWS terkelola berikut, yang dapat Anda lampirkan ke grup dan peran di akun Anda, khusus untuk Amazon Lex:

- AmazonLexReadOnly— Memberikan akses hanya-baca ke sumber daya Amazon Lex.
- AmazonLexRunBotsOnly— Memberikan akses untuk menjalankan bot percakapan Amazon Lex.
- AmazonLexFullAccess— Memberikan akses penuh untuk membuat, membaca, memperbarui, menghapus, dan menjalankan semua sumber daya Amazon Lex. Juga memberikan kemampuan untuk mengaitkan fungsi Lambda yang namanya dimulai `AmazonLex` dengan maksud Amazon Lex.

> **Note**
>
> Anda dapat meninjau kebijakan izin ini dengan masuk ke konsol IAM dan mencari kebijakan tertentu.

AmazonLexFullAccessKebijakan ini tidak memberikan izin kepada pengguna untuk menggunakan `KendraSearchIntent` intent untuk menanyakan indeks Amazon Kendra. Untuk membuat kueri indeks, Anda harus menambahkan izin tambahan ke kebijakan. Untuk izin yang diperlukan, lihat Kebijakan IAM untuk Pencarian Amazon Kendra.

Anda juga dapat membuat kebijakan IAM kustom Anda sendiri untuk mengizinkan izin untuk tindakan Amazon Lex API. Anda dapat melampirkan kebijakan khusus ini ke peran atau grup IAM yang memerlukan izin tersebut.

Untuk detail tentang kebijakan terkelola AWS untuk Amazon Lex, lihatAWSkebijakan terkelola untuk Amazon Lex.

Izinkan para pengguna untuk melihat izin mereka sendiri

Contoh ini menunjukkan cara Anda dapat membuat kebijakan yang mengizinkan para pengguna IAM untuk melihat kebijakan inline dan terkelola yang dilampirkan ke identitas pengguna mereka. Kebijakan ini mencakup izin untuk menyelesaikan tindakan pada konsol atau secara terprogram menggunakan API AWS CLI atau AWS.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
```

```
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Hapus Semua Bot Amazon Lex

Contoh kebijakan ini memberikan izin kepada pengguna di akun AWS Anda untuk menghapus bot apa pun di akun Anda.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "lex:DeleteBot"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

Izinkan pengguna untuk memigrasikan bot ke Amazon Lex V2 API

Kebijakan izin IAM berikut memungkinkan pengguna untuk mulai memigrasikan bot dari Amazon Lex ke Amazon Lex V2 API dan untuk melihat daftar migrasi dan kemajuannya.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "startMigration",
            "Effect": "Allow",
            "Action": "lex:StartMigration",
            "Resource": "arn:aws:lex:<Region>:<123456789012>:bot:*"
        },
        {
            "Sid": "passRole",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::<123456789012>:role/<v2 bot role>"
        },
        {
            "Sid": "allowOperations",
            "Effect": "Allow",
            "Action": [
                "lex:CreateBot",
                "lex:CreateIntent",
                "lex:UpdateSlot",
                "lex:DescribeBotLocale",
                "lex:UpdateBotAlias",
                "lex:CreateSlotType",
                "lex:DeleteBotLocale",
                "lex:DescribeBot",
                "lex:UpdateBotLocale",
                "lex:CreateSlot",
                "lex:DeleteSlot",
                "lex:UpdateBot",
                "lex:DeleteSlotType",
                "lex:DescribeBotAlias",
                "lex:CreateBotLocale",
                "lex:DeleteIntent",
                "lex:StartImport",
                "lex:UpdateSlotType",
                "lex:UpdateIntent",
                "lex:DescribeImport",
                "lex:CreateCustomVocabulary",
                "lex:UpdateCustomVocabulary",
                "lex:DeleteCustomVocabulary",
                "lex:DescribeCustomVocabulary",
```

```
                "lex:DescribeCustomVocabularyMetadata"
            ],
            "Resource": [
                "arn:aws:lex:<Region>:<123456789012>:bot/*",
                "arn:aws:lex:<Region>:<123456789012>:bot-alias/*/*"
            ]
        },
        {
            "Sid": "showBots",
            "Effect": "Allow",
            "Action": [
                "lex:CreateUploadUrl",
                "lex:ListBots"
            ],
            "Resource": "*"
        },
        {
            "Sid": "showMigrations",
            "Effect": "Allow",
            "Action": [
                "lex:GetMigration",
                "lex:GetMigrations"
            ],
            "Resource": "*"
        }
    ]
}
```

Menggunakan Tag untuk Mengakses Sumber Daya

Kebijakan contoh ini memberikan izin kepada pengguna atau peran di AWS akun Anda untuk menggunakan `PostText` operasi dengan sumber daya apa pun yang ditandai dengan kunci **`Department`** dan nilainya. **`Support`**

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": "lex:PostText",
            "Effect": "Allow",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
```

```
                    "lex:ResourceTag/Department": "Support"
                }
            }
        }
    ]
}
```

AWSkebijakan terkelola untuk Amazon Lex

SebuahAWSkebijakan terkelola adalah kebijakan mandiri yang dibuat dan dikelola olehAWS.AWSkebijakan terkelola dirancang untuk memberikan izin untuk banyak kasus penggunaan umum sehingga Anda dapat mulai menetapkan izin kepada pengguna, grup, dan peran.

Perlu diingat bahwaAWSkebijakan terkelola mungkin tidak memberikan izin paling sedikit hak istimewa untuk kasus penggunaan spesifik Anda karena tersedia untuk semuaAWSpelanggan untuk digunakan. Kami menyarankan Anda mengurangi izin lebih lanjut dengan mendefinisikankebijakan yang dikelola pelanggan yang khusus untuk kasus penggunaan Anda.

Anda tidak dapat mengubah izin yang ditentukan dalamAWSkebijakan yang dikelola. JikaAWSmemperbarui izin yang didefinisikan dalamAWSkebijakan terkelola, pembaruan mempengaruhi semua identitas utama (pengguna, grup, dan peran) yang dilampirkan kebijakan.AWSkemungkinan besar akan memperbaruiAWSkebijakan terkelola saat baruLayanan AWSdiluncurkan atau operasi API baru tersedia untuk layanan yang ada.

Untuk informasi selengkapnya, lihat Kebijakan terkelola AWS dalam Panduan Pengguna IAM.

Kebijakan yang dikelola AWS:AmazonLexReadOnly

Anda dapat melampirkan kebijakan `AmazonLexReadOnly` ke identitas-identitas IAM Anda.

Kebijakan ini memberikan izin hanya-baca yang memungkinkan pengguna melihat semua tindakan dalam layanan pembuatan model Amazon Lex dan Amazon Lex V2.

Rincian izin

Kebijakan ini mencakup izin berikut:

- `lex`— Akses hanya-baca ke sumber daya Amazon Lex dan Amazon Lex V2 dalam layanan pembuatan model.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "lex:GetBot",
                "lex:GetBotAlias",
                "lex:GetBotAliases",
                "lex:GetBots",
                "lex:GetBotChannelAssociation",
                "lex:GetBotChannelAssociations",
                "lex:GetBotVersions",
                "lex:GetBuiltinIntent",
                "lex:GetBuiltinIntents",
                "lex:GetBuiltinSlotTypes",
                "lex:GetIntent",
                "lex:GetIntents",
                "lex:GetIntentVersions",
                "lex:GetSlotType",
                "lex:GetSlotTypes",
                "lex:GetSlotTypeVersions",
                "lex:GetUtterancesView",
                "lex:DescribeBot",
                "lex:DescribeBotAlias",
                "lex:DescribeBotChannel",
                "lex:DescribeBotLocale",
                "lex:DescribeBotVersion",
                "lex:DescribeExport",
                "lex:DescribeImport",
                "lex:DescribeIntent",
                "lex:DescribeResourcePolicy",
                "lex:DescribeSlot",
                "lex:DescribeSlotType",
                "lex:ListBots",
```

```
                "lex:ListBotLocales",
                "lex:ListBotAliases",
                "lex:ListBotChannels",
                "lex:ListBotVersions",
                "lex:ListBuiltInIntents",
                "lex:ListBuiltInSlotTypes",
                "lex:ListExports",
                "lex:ListImports",
                "lex:ListIntents",
                "lex:ListSlots",
                "lex:ListSlotTypes",
                "lex:ListTagsForResource"
            ],
            "Resource": "*"
        }
    ]
}
```

Kebijakan yang dikelola AWS:AmazonLexRunBotsOnly

Anda dapat melampirkan kebijakan `AmazonLexRunBotsOnly` ke identitas-identitas IAM Anda.

Kebijakan ini memberikan izin hanya-baca yang memungkinkan akses untuk menjalankan bot percakapan Amazon Lex dan Amazon Lex V2.

Rincian izin

Kebijakan ini mencakup izin berikut:

- `lex`— Akses hanya-baca ke semua tindakan dalam waktu proses Amazon Lex dan Amazon Lex V2.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "lex:PostContent",
                "lex:PostText",
                "lex:PutSession",
                "lex:GetSession",
```

```
                "lex:DeleteSession",
                "lex:RecognizeText",
                "lex:RecognizeUtterance",
                "lex:StartConversation"
            ],
            "Resource": "*"
        }
    ]
}
```

Kebijakan yang dikelola AWS:AmazonLexFullAccess

Anda dapat melampirkan kebijakan `AmazonLexFullAccess` ke identitas-identitas IAM Anda.

Kebijakan ini memberikan izin administratif yang memungkinkan izin pengguna untuk membuat, membaca, memperbarui, dan menghapus sumber daya Amazon Lex dan Amazon Lex V2, serta menjalankan bot percakapan Amazon Lex dan Amazon Lex V2.

Rincian izin

Kebijakan ini mencakup izin berikut:

- `lex`- Memungkinkan prinsipal membaca dan menulis akses ke semua tindakan di Amazon Lex dan Amazon Lex V2 model bangunan dan runtime layanan.
- `cloudwatch`- Memungkinkan prinsipal untuk melihat AmazonCloudWatchmetrik dan alarm.
- `iam`- Memungkinkan prinsipal untuk membuat dan menghapus peran terkait layanan, meneruskan peran, dan melampirkan dan 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 prinsipal untuk mencantumkan indeks Amazon Kendra.
- `kms`- Memungkinkan prinsipal untuk menggambarkanAWS KMSkunci dan alias.
- `lambda`- Memungkinkan kepala sekolah untuk daftarAWS Lambdafungsi dan mengelola izin yang melekat pada setiap fungsi Lambda.
- `polly`- Memungkinkan kepala sekolah untuk menggambarkan suara Amazon Polly dan mensintesis ucapan.

```
{
    "Version": "2012-10-17",
```

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:DescribeAlarms",
                "cloudwatch:DescribeAlarmsForMetric",
                "kms:DescribeKey",
                "kms:ListAliases",
                "lambda:GetPolicy",
                "lambda:ListFunctions",
                "lex:*",
                "polly:DescribeVoices",
                "polly:SynthesizeSpeech",
                "kendra:ListIndices",
                "iam:ListRoles",
                "s3:ListAllMyBuckets",
                "logs:DescribeLogGroups",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "lambda:AddPermission",
                "lambda:RemovePermission"
            ],
            "Resource": "arn:aws:lambda:*:*:function:AmazonLex*",
            "Condition": {
                "StringEquals": {
                    "lambda:Principal": "lex.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:GetRole"
            ],
            "Resource": [
```

```
                "arn:aws:iam::*:role/aws-service-role/lex.amazonaws.com/
AWSServiceRoleForLexBots",
                "arn:aws:iam::*:role/aws-service-role/channels.lex.amazonaws.com/
AWSServiceRoleForLexChannels",
                "arn:aws:iam::*:role/aws-service-role/lexv2.amazonaws.com/
AWSServiceRoleForLexV2Bots*",
                "arn:aws:iam::*:role/aws-service-role/channels.lexv2.amazonaws.com/
AWSServiceRoleForLexV2Channels*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/lex.amazonaws.com/
AWSServiceRoleForLexBots"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "lex.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/channels.lex.amazonaws.com/
AWSServiceRoleForLexChannels"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "channels.lex.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
```

```
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/lexv2.amazonaws.com/
AWSServiceRoleForLexV2Bots*"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "lexv2.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/channels.lexv2.amazonaws.com/
AWSServiceRoleForLexV2Channels*"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "channels.lexv2.amazonaws.com"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:DeleteServiceLinkedRole",
                "iam:GetServiceLinkedRoleDeletionStatus"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/lex.amazonaws.com/
AWSServiceRoleForLexBots",
                "arn:aws:iam::*:role/aws-service-role/channels.lex.amazonaws.com/
AWSServiceRoleForLexChannels",
                "arn:aws:iam::*:role/aws-service-role/lexv2.amazonaws.com/
AWSServiceRoleForLexV2Bots*",
                "arn:aws:iam::*:role/aws-service-role/channels.lexv2.amazonaws.com/
AWSServiceRoleForLexV2Channels*"
            ]
        },
        {
```

```
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/lex.amazonaws.com/
AWSServiceRoleForLexBots"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "lex.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/lexv2.amazonaws.com/
AWSServiceRoleForLexV2Bots*"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "lexv2.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/channels.lexv2.amazonaws.com/
AWSServiceRoleForLexV2Channels*"
            ],
            "Condition": {
                "StringEquals": {
```

```
                    "iam:PassedToService": [
                        "channels.lexv2.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

Amazon Lex memperbaruiAWSkebijakan terkelola

Lihat detail tentang pembaruanAWSkebijakan terkelola untuk Amazon Lex sejak layanan ini mulai melacak perubahan ini. Untuk peringatan otomatis tentang perubahan pada halaman ini, berlangganan umpan RSS di Amazon LexRiwayat Dokumen untuk Amazon Lexhalaman.

Perubahan	Deskripsi	Tanggal
AmazonLexFullAccess — Permbaruan ke kebijakan yang sudah ada	Amazon Lex menambahkan izin baru untuk mengizinkan akses hanya-baca ke operasi layanan pembuatan model Amazon Lex V2.	Agustus 18, 2021
AmazonLexReadOnly — Permbaruan ke kebijakan yang sudah ada	Amazon Lex menambahkan izin baru untuk mengizinkan akses hanya-baca ke operasi layanan pembuatan model Amazon Lex V2.	Agustus 18, 2021
AmazonLexRunBotsOnly — Permbaruan ke kebijakan yang sudah ada	Amazon Lex menambahkan izin baru untuk mengizinkan akses hanya-baca ke operasi layanan runtime Amazon Lex V2.	Agustus 18, 2021

Perubahan	Deskripsi	Tanggal
Amazon Lex mulai melacak perubahan	Amazon Lex mulai melacak perubahannyaAWSkebijakan yang dikelola.	Agustus 18, 2021

Menggunakan Peran tertaut layanan untuk Amazon Lex

Amazon Lex menggunakan peran terkait layananAWS Identity and Access Management (IAM). Peran tertaut layanan adalah jenis IAM role unik yang ditautkan langsung ke Amazon Lex. Peran tertaut layanan ditentukan sebelumnya oleh Amazon Lex dan mencakup semua izin yang diperlukan layanan untuk memanggilAWS layanan lainnya atas nama Anda.

Peran tertaut layanan mempermudah pengaturan Amazon Lex karena Anda tidak perlu menambahkan izin yang diperlukan secara manual. Amazon Lex menentukan izin peran tertaut layanan, dan kecuali ditentukan lain, hanya Amazon Lex yang dapat menjalankan perannya. Izin yang ditentukan mencakup kebijakan trust dan kebijakan izin, serta bahwa kebijakan izin tidak dapat diberikan ke entitas IAM lainnya.

Anda dapat menghapus peran tertaut layanan hanya setelah terlebih dahulu menghapus sumber dayanya yang terkait. Tindakan ini akan melindungi sumber daya Amazon Lex karena Anda tidak bisa secara tidak sengaja menghapus izin untuk mengakses sumber daya.

Izin Peran tertaut layanan untuk Amazon Lex

Amazon Lex menggunakan dua peran tertaut layanan:

- AWSServiceRoleForLexBots- Amazon Lex menggunakan peran terkait layanan ini untuk memanggil Amazon Polly untuk mensintesis tanggapan ucapan untuk bot Anda, untuk memanggil Amazon Comprehend untuk analisis sentimen, dan secara opsional Amazon Kendra untuk mencari indeks.
- AWSServiceRoleForLexChannels— Amazon Lex menggunakan peran terkait layanan ini untuk memposting teks ke bot Anda saat mengelola saluran.

Anda harus mengonfigurasi izin untuk mengizinkan entitas IAM (seperti pengguna, grup, atau peran) untuk membuat, mengedit, atau menghapus peran terkait layanan. Untuk informasi lebih lanjut, lihat Izin Peran Tertaut Layanan di Panduan Pengguna IAM.

Membuat Peran tertaut layanan untuk Amazon Lex

Anda tidak perlu membuat peran terkait layanan secara manual. Ketika Anda membuat bot, saluran bot, atau maksud pencarian Amazon Kendra diAWS Management Console, Amazon Lex membuat peran tertaut layanan untuk Anda.

Jika Anda menghapus peran tertaut layanan ini, dan ingin membuatnya lagi, Anda dapat mengulangi proses yang sama untuk membuat kembali peran tersebut di akun Anda. Ketika Anda membuat bot baru, asosiasi saluran, atau maksud pencarian Amazon Kendra, Amazon Lex membuat peran terkait layanan bagi Anda kembali.

Anda juga dapat menggunakanAWS CLI untuk membuat peran tertaut layanan dengan kasus AWSServiceRoleForLexBotspenggunaan. DiAWS CLI membuat peran tertaut layanan dengan nama layanan Amazon Lex`lex.amazonaws.com`. Untuk informasi selengkapnya, lihat Langkah 1: Membuat Peran Terkait Layanan (AWS CLI). Jika Anda menghapus peran tertaut layanan ini, Anda dapat mengulang proses yang sama untuk membuat peran tersebut lagi.

Mengedit Peran tertaut layanan untuk Amazon Lex

Amazon Lex tidak mengizinkan Anda mengedit peran tertaut layanan Amazon Lex. Setelah Anda membuat peran terkait layanan, Anda tidak dapat mengubah nama peran karena berbagai entitas mungkin mereferensikan peran tersebut. Namun, Anda dapat mengedit penjelasan peran menggunakan IAM. Untuk informasi lebih lanjut, lihat Mengedit Peran Tertaut Layanan di Panduan Pengguna IAM.

Menghapus Peran tertaut layanan untuk Amazon Lex

Jika Anda tidak perlu lagi menggunakan fitur atau layanan yang memerlukan peran terkait layanan, kami merekomendasikan Anda menghapus 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 menggunakan peran saat Anda mencoba untuk menghapus sumber daya, maka penghapusan tersebut kemungkinan gagal. Jika hal itu terjadi, tunggu beberapa menit dan coba mengoperasikannya lagi.

Untuk menghapus sumber daya Amazon Lex yang digunakan oleh peran terkait layanan:

1. Hapus saluran bot apa pun yang Anda gunakan.
2. Hapus bot apa pun di akun Anda.

Untuk menghapus peran tertaut layanan secara manual menggunakan IAM

Menggunakan konsol IAM,AWS CLI, atauAWS API untuk menghapus peran tertaut layanan Amazon Lex. Untuk informasi lebih lanjut, lihat Menghapus Peran Tertaut Layanan di Panduan Pengguna IAM.

Wilayah yang disupport untuk Peran tertaut layanan Amazon Lex

Amazon Lex mendukung penggunaan peran tertaut layanan di semua wilayah tempat layanan tersedia. Untuk informasi lebih lanjut, lihat Endpoint dan kuota Amazon Lex.

Memecahkan masalah identitas dan akses Amazon Lex

Gunakan informasi berikut untuk membantu Anda mendiagnosis dan memperbaiki masalah umum yang mungkin Anda temui saat bekerja dengan Amazon Lex dan IAM.

Topik

- Saya tidak berwenang untuk melakukan tindakan di Amazon Lex
- Saya tidak berwenang untuk melakukan iam: PassRole
- Saya ingin mengizinkan orang di luar saya Akun AWS untuk mengakses sumber daya Amazon Lex saya

Saya tidak berwenang untuk melakukan tindakan di Amazon Lex

Jika Anda menerima pesan galat bahwa Anda tidak memiliki otorisasi untuk melakukan tindakan, kebijakan Anda harus diperbarui agar Anda dapat melakukan tindakan tersebut.

Contoh galat berikut terjadi ketika pengguna IAM `mateojackson` mencoba menggunakan konsol untuk melihat detail tentang suatu sumber daya *`my-example-widget`* rekaan, tetapi tidak memiliki izin `lex:`*`GetWidget`* rekaan.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 lex:GetWidget on resource: my-example-widget
```

Dalam hal ini, kebijakan untuk pengguna `mateojackson` harus diperbarui untuk mengizinkan akses ke sumber daya *`my-example-widget`* dengan menggunakan tindakan `lex:`*`GetWidget`*.

Jika Anda membutuhkan bantuan, hubungi administrator AWS Anda. Administrator Anda adalah orang yang memberikan kredensial masuk Anda.

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.

Sebagian Layanan AWS mengizinkan Anda untuk memberikan peran yang sudah ada ke layanan tersebut alih-alih membuat peran layanan baru atau peran tertaut-layanan. Untuk melakukan tindakan tersebut, Anda harus memiliki izin untuk memberikan peran pada layanan tersebut.

Contoh kesalahan berikut terjadi ketika pengguna IAM bernama `marymajor` mencoba menggunakan konsol untuk melakukan tindakan di Amazon Lex. 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 membutuhkan bantuan, hubungi administrator AWS Anda. Administrator Anda adalah orang yang memberikan kredensial masuk Anda.

Saya ingin mengizinkan orang di luar saya Akun AWS untuk mengakses sumber daya Amazon Lex saya

Anda dapat membuat peran yang dapat digunakan para pengguna di akun lain atau orang-orang di luar organisasi Anda untuk mengakses sumber daya Anda. Anda dapat menentukan siapa 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 akses kepada orang ke sumber daya Anda.

Untuk mempelajari selengkapnya, periksa hal berikut:

- Untuk mengetahui apakah Amazon Lex mendukung fitur-fitur ini, lihatBagaimana Amazon Lex bekerja dengan IAM.
- Untuk mempelajari cara memberikan akses ke sumber daya di seluruh Akun AWS yang Anda miliki, silakan lihat Menyediakan akses ke pengguna IAM di Akun AWS lainnya yang Anda miliki di Panduan Pengguna IAM.
- Untuk mempelajari cara memberikan akses ke sumber daya Anda ke pihak ketiga Akun AWS, silakan lihat Menyediakan akses ke akun Akun AWS yang dimiliki oleh pihak ketiga dalam Panduan Pengguna IAM.
- Untuk mempelajari cara memberikan akses melalui federasi identitas, silakan lihat Menyediakan akses ke pengguna terautentikasi eksternal (gabungan identitas) di Panduan Pengguna IAM .
- Untuk mempelajari perbedaan antara penggunaan peran dan kebijakan berbasis sumber daya untuk akses lintas akun, lihat Perbedaan IAM role dan kebijakan berbasis sumber daya di Panduan Pengguna IAM.

Pemantauan di Amazon Lex

Pemantauan penting untuk menjaga keandalan, ketersediaan, dan kinerja chatbots Amazon Lex Anda. Topik ini menjelaskan cara menggunakan Amazon CloudWatch Logs dan AWS CloudTrail memantau Amazon Lex serta menjelaskan metrik runtime dan asosiasi saluran Amazon Lex.

Topik

- Memantau Amazon Lex dengan Amazon CloudWatch
- Memantau Panggilan Amazon Lex API dengan AWS CloudTrail Log

Memantau Amazon Lex dengan Amazon CloudWatch

Untuk melacak kesehatan bot Amazon Lex Anda, gunakan Amazon CloudWatch. Dengan CloudWatch, Anda bisa mendapatkan metrik untuk operasi Amazon Lex individual atau untuk operasi Amazon Lex global untuk akun Anda. Anda juga dapat mengatur CloudWatch alarm untuk diberi tahu ketika satu atau beberapa metrik melebihi ambang batas yang Anda tentukan. Misalnya, Anda dapat memantau jumlah permintaan yang dibuat ke bot selama periode waktu tertentu, melihat latensi permintaan yang berhasil, atau menaikkan alarm ketika kesalahan melebihi ambang batas.

CloudWatch Metrik untuk Amazon Lex

Untuk mendapatkan metrik untuk operasi Amazon Lex Anda, Anda harus menentukan informasi berikut:

- Dimensi metrik. Dimensi adalah sekumpulan pasangan nama-nilai yang Anda gunakan untuk mengidentifikasi metrik. Amazon Lex memiliki tiga dimensi:
  - `BotAlias, BotName, Operation`
  - `BotAlias, BotName, InputMode, Operation`
  - `BotName, BotVersion, InputMode, Operation`
- Nama metrik, seperti `MissedUtteranceCount` atau`RuntimeRequestCount`.

Anda bisa mendapatkan metrik untuk Amazon Lex denganAWS Management Console, theAWS CLI, atau CloudWatch API. Anda dapat menggunakan CloudWatch API melalui salah satu Kit Pengembangan Perangkat Lunak Amazon AWS (SDK) atau alat CloudWatch API. Konsol Amazon Lex menampilkan grafik berdasarkan data mentah dari CloudWatch API.

Anda harus memiliki CloudWatch izin yang sesuai untuk memantau Amazon Lex. CloudWatch Untuk informasi selengkapnya, lihat Otentikasi dan Kontrol Akses untuk Amazon CloudWatch di Panduan CloudWatch Pengguna Amazon.

Melihat Metrik Amazon Lex

Lihat metrik Amazon Lex menggunakan konsol Amazon Lex atau CloudWatch konsol.

Untuk melihat metrik (konsol Amazon Lex)

1. Masuk ke AWS Management Console dan buka konsol Amazon Lex di https://console.aws.amazon.com/lex/.
2. Dari daftar bot, pilih salah satu yang metriknya ingin Anda lihat.
3. Pilih Pemantauan. Metrik ditampilkan dalam grafik.

Untuk melihat metrik (CloudWatch konsol)

1. Masuk ke AWS Management Console dan buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pilih Metrik, pilih Semua Metrik, lalu pilih AWS/Lex.

3. Pilih dimensi, pilih nama metrik, lalu pilih Tambahkan ke grafik.
4. Pilih nilai untuk rentang tanggal. Hitungan metrik untuk rentang tanggal yang dipilih akan ditampilkan dalam grafik.

Membuat Alarm

CloudWatch Alarm mengawasi satu metrik selama periode waktu tertentu, dan melakukan satu atau beberapa tindakan: mengirim pemberitahuan ke topik Amazon Simple Notification Service (Amazon SNS) atau kebijakan Auto Scaling. Tindakan atau tindakan didasarkan pada nilai metrik relatif terhadap ambang batas tertentu selama sejumlah periode waktu yang Anda tentukan. CloudWatchjuga dapat mengirimi Anda pesan Amazon SNS saat alarm berubah status.

CloudWatch alarm memanggil tindakan hanya ketika status berubah dan telah bertahan selama periode yang Anda tentukan.

Untuk mengatur alarm

1. Masuk ke AWS Management Console dan buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pilih Alarm, lalu pilih Buat Alarm.
3. Pilih AWS/Lex Metrics, lalu pilih metrik.
4. Untuk Rentang Waktu, pilih rentang waktu untuk dipantau, lalu pilih Selanjutnya.
5. Masukkan Nama dan Deskripsi.
6. Untuk Kapan pun pilih >=, dan ketik nilai maksimum.
7. Jika Anda CloudWatch ingin mengirim email saat status alarm tercapai, di bagian Tindakan, untuk Setiap kali alarm ini, pilih Status adalah ALARM. Untuk Kirim pemberitahuan ke, pilih milis atau pilih Daftar baru dan buat milis baru.
8. Pratinjau alarm di bagian Pratinjau Alarm. Jika Anda puas dengan alarm, pilih Buat Alarm.

CloudWatchMetrik untuk Runtime Amazon Lex

Tabel berikut menjelaskan metrik runtime Amazon Lex.

Metrik	Deskripsi
KendraIndexAccessError	Berapa kali Amazon Lex tidak dapat mengakses indeks Amazon Kendra Anda. Dimensi yang valid untuk PostContent operasi dengan Text atau SpeechInputMode : • BotName, BotAlias, Operation, InputMode Dimensi yang valid untuk PostText operasi: • BotName, BotAlias, Operation Unit: Jumlah
KendraLatency	Jumlah waktu yang dibutuhkan Amazon Kendra untuk menanggapi permintaan dari. AMAZON.KendraSearchIntent Dimensi yang valid untuk PostContent operasi dengan Text atau SpeechInputMode : • BotName, BotVersion, Operation, InputMode • BotName, BotAlias, Operation, InputMode Dimensi yang valid untuk PostText operasi: • BotName, BotVersion, Operation • BotName, BotAlias, Operation Unit: Milidetik
KendraSuccess	Jumlah permintaan yang berhasil dari AMAZON.KendraSearchIntent ke indeks Amazon Kendra Anda. Dimensi yang valid untuk PostContent operasi dengan Text atau SpeechInputMode :

Metrik	Deskripsi
	• BotName, BotVersion, Operation, InputMode • BotName, BotAlias, Operation, InputMode Dimensi yang valid untuk PostText operasi: • BotName, BotVersion, Operation • BotName, BotAlias, Operation Unit: Jumlah
KendraSystemErrors	Berapa kali Amazon Lex tidak bisa menanyakan indeks Amazon Kendra. Dimensi yang valid untuk PostContent operasi dengan Text atau SpeechInputMode : • BotName, BotAlias, Operation, InputMode Dimensi yang valid untuk PostText operasi: • BotName, BotAlias, Operation Unit: Jumlah

Metrik	Deskripsi
KendraThrottledEvents	Berapa kali Amazon Kendra membatasi permintaan dari. AMAZON.KendraSearchIntent Dimensi yang valid untuk PostContent operasi dengan Text atau SpeechInputMode : • BotName, BotAlias, Operation, InputMode Dimensi yang valid untuk PostText operasi: • BotName, BotAlias, Operation Unit: Jumlah
MissedUtteranceCount	Jumlah ucapan yang tidak diakui dalam periode yang ditentukan. Dimensi yang valid untuk PostContent operasi dengan Text atau SpeechInputMode : • BotName, BotVersion, Operation, InputMode • BotName, BotAlias, Operation, InputMode Dimensi yang valid untuk PostText operasi: • BotName, BotVersion, Operation • BotName, BotAlias, Operation

Metrik	Deskripsi
`RuntimeConcurrency`	Jumlah koneksi bersamaan dalam periode waktu yang ditentukan. `RuntimeConcurrency` dilaporkan sebagai a`StatisticSet` . Dimensi yang valid untuk `PostContent` operasi dengan `Text` atau `SpeechInputMode` : • Operasi, BotName, BotVersion, InputMode • Operasi, BotName, BotAlias, InputMode Dimensi yang valid untuk operasi lain: • Operasi, BotName, BotVersion • Operasi, BotName, BotAlias Unit: Jumlah
`RuntimeInvalidLambdaResponses`	Jumlah tanggapan (AWS LambdaLambda) yang tidak valid dalam periode yang ditentukan. Dimensi yang valid untuk `PostContent` operasi dengan `Text` atau `SpeechInputMode` : • `BotName, BotAlias, Operation, InputMode` Dimensi yang valid untuk `PostText` operasi: • `BotName, BotAlias, Operation`

Metrik	Deskripsi
RuntimeLambdaErrors	Jumlah kesalahan runtime Lambda dalam periode yang ditentukan. Dimensi yang valid untuk PostContent operasi dengan Text atau Speech InputMode : • BotName, BotAlias, Operation, InputMode Dimensi yang valid untuk PostText operasi: • BotName, BotAlias, Operation
RuntimePollyErrors	Jumlah tanggapan Amazon Polly yang tidak valid dalam periode yang ditentukan. Dimensi yang valid untuk PostContent operasi dengan Text atau SpeechInputMode : • BotName, BotAlias, Operation, InputMode Dimensi yang valid untuk PostText operasi: • BotName, BotAlias, Operation

Metrik	Deskripsi
`RuntimeRequestCount`	Jumlah permintaan runtime dalam periode yang ditentukan. Dimensi yang valid untuk `PostContent` operasi dengan `Text` atau `SpeechInputMode` : • `BotName, BotVersion, Operation, InputMode` • `BotName, BotAlias, Operation, InputMode` Dimensi yang valid untuk `PostText` operasi: • `BotName, BotVersion, Operation` • `BotName, BotAlias, Operation` Unit: Jumlah
`RuntimeSucessfulRequestLatency` **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 `PostContent` operasi dengan `Text` atau `SpeechInputMode` : • `BotName, BotVersion, Operation, InputMode` • `BotName, BotAlias, Operation, InputMode` Dimensi yang valid untuk `PostText` operasi: • `BotName, BotVersion, Operation` • `BotName, BotAlias, Operation` Unit: Milidetik

Metrik	Deskripsi
`RuntimeSystemErrors`	Jumlah kesalahan sistem dalam periode yang ditentukan. Rentang kode respons untuk kesalahan sistem adalah 500 hingga 599. Dimensi yang valid untuk `PostContent` operasi dengan `Text` atau `SpeechInputMode` : • `BotName, BotAlias, Operation, InputMode` Dimensi yang valid untuk `PostText` operasi: • `BotName, BotAlias, Operation` Unit: Jumlah
`RuntimeThrottledEvents`	Jumlah permintaan yang ditahan. Amazon Lex membatasi permintaan 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 penambahan, lihat Kuota Layanan AWS. Dimensi yang valid untuk `PostContent` operasi dengan `Text` atau `SpeechInputMode` : • BotName, BotAlias, Operasi, InputMode Dimensi yang valid untuk `PostText` operasi: • `BotName, BotAlias, Operation` Unit: Jumlah

Metrik	Deskripsi
RuntimeUserErrors	Jumlah kesalahan pengguna dalam periode yang ditentukan. Kisaran kode respons untuk kesalahan pengguna adalah 400 hingga 499. Dimensi yang valid untuk PostContent operasi dengan Text atau SpeechInputMode : • BotName, BotAlias, Operation, InputMode Dimensi yang valid untuk PostText operasi: • BotName, BotAlias, Operation Unit: Jumlah

Metrik runtime Amazon Lex menggunakan AWS/Lex namespace, dan menyediakan metrik dalam dimensi berikut. Anda dapat mengelompokkan metrik berdasarkan dimensi di CloudWatch konsol:

Dimensi	Deskripsi
BotName, BotAlias, Operation, InputMode	Mengelompokkan metrik berdasarkan alias bot, nama bot, operasi (PostContent), dan apakah inputnya adalah teks atau ucapan.
BotName, BotVersion, Operation, InputMode	Mengelompokkan metrik berdasarkan nama bot, versi bot, operasi (PostContent), dan apakah inputnya adalah teks atau ucapan.
BotName, BotVersion, Operation	Kelompokkan metrik berdasarkan nama bot, versi bot, dan operasi,. PostText
BotName, BotAlias, Operation	Mengelompokkan metrik berdasarkan nama bot, alias bot, dan dengan operasi,. PostText

CloudWatch Metrik untuk Asosiasi Saluran Amazon Lex

Asosiasi saluran adalah hubungan antara Amazon Lex dan saluran perpesanan, seperti Facebook. Tabel berikut menjelaskan metrik asosiasi saluran Amazon Lex.

Metrik	Deskripsi
`BotChannelAuthErrors`	Jumlah kesalahan otentikasi yang dikembalikan oleh saluran pesan dalam periode waktu yang ditentukan. Kesalahan otentikasi menunjukkan bahwa token rahasia yang diberikan selama pembuatan saluran tidak valid atau telah kedaluwarsa.
`BotChannelConfigurationErrors`	Jumlah kesalahan konfigurasi dalam periode yang ditentukan. Kesalahan konfigurasi menunjukkan bahwa satu atau beberapa entri konfigurasi untuk saluran tidak valid.
`BotChannelInboundThrottledEvents`	Berapa kali pesan yang dikirim oleh saluran pesan dibatasi oleh Amazon Lex dalam periode yang ditentukan.
`BotChannelOutboundThrottledEvents`	Berapa kali peristiwa keluar dari Amazon Lex ke saluran pesan dibatasi dalam periode waktu yang ditentukan.
`BotChannelRequestCount`	Jumlah permintaan yang dibuat pada saluran dalam periode waktu yang ditentukan.
`BotChannelResponseCardErrors`	Berapa kali Amazon Lex tidak dapat memposting kartu respons dalam periode yang ditentukan.
`BotChannelSystemErrors`	Jumlah kesalahan internal yang terjadi di Amazon Lex untuk saluran dalam periode yang ditentukan.

Metrik asosiasi saluran Amazon Lex menggunakan `AWS/Lex` namespace, dan menyediakan metrik untuk dimensi berikut. Anda dapat mengelompokkan metrik berdasarkan dimensi di CloudWatch konsol:

Dimensi	Deskripsi
`BotAlias, BotChannelName, BotName, Source`	Metrik grup berdasarkan alias bot, nama saluran, nama bot, dan sumber lalu lintas.

CloudWatch Metrik untuk Log Percakapan

Amazon Lex menggunakan metrik berikut untuk pencatatan percakapan:

Metrik	Deskripsi
`ConversationLogsAudioDeliverySuccess`	Jumlah log audio yang berhasil dikirim ke bucket S3 dalam periode waktu yang ditentukan. Unit: Count (Jumlah)
`ConversationLogsAudioDeliveryFailure`	Jumlah log audio yang gagal dikirim ke bucket S3 dalam periode waktu yang ditentukan. Kegagalan pengiriman menunjukkan kesalahan dengan sumber daya yang dikonfigurasi untuk log percakapan. Kesalahan dapat mencakup izin IAM yang tidak mencukupi, AWS KMS kunci yang tidak dapat diakses, atau bucket S3 yang tidak dapat diakses. Unit: Count (Jumlah)
`ConversationLogsTextDeliverySuccess`	Jumlah log teks yang berhasil dikirim ke CloudWatch Log dalam periode waktu yang ditentukan. Unit: Count (Jumlah)
`ConversationLogsTextDeliveryFailure`	Jumlah log teks yang gagal dikirim ke CloudWatch Log dalam periode waktu yang ditentukan. Kegagalan pengiriman menunjukk

Metrik	Deskripsi
	an kesalahan dengan sumber daya yang dikonfigurasi untuk log percakapan. Kesalahan dapat mencakup izin IAM yang tidak memadai, AWS KMS kunci yang tidak dapat diakses, atau grup log Log yang tidak dapat diakses CloudWatch . Unit: Count (Jumlah)

Metrik log percakapan Amazon Lex menggunakan `AWS/Lex` namespace, dan menyediakan metrik untuk dimensi berikut. Anda dapat mengelompokkan metrik berdasarkan dimensi di CloudWatch konsol.

Dimensi	Deskripsi
`BotAlias`	Metrik grup dengan alias bot.
`BotName`	Metrik grup dengan nama bot.
`BotVersion`	Metrik grup berdasarkan versi bot.

Memantau Panggilan Amazon Lex API dengan AWS CloudTrail Log

Amazon Lex terintegrasi denganAWS CloudTrail, layanan yang menyediakan catatan tindakan yang diambil oleh pengguna, peran, atau AWS layanan di Amazon Lex. CloudTrail menangkap subset panggilan API untuk Amazon Lex sebagai peristiwa, termasuk panggilan dari konsol Amazon Lex dan dari panggilan kode ke Amazon Lex API. Jika Anda membuat jejak, Anda dapat mengaktifkan pengiriman CloudTrail acara secara terus menerus ke bucket Amazon S3, termasuk acara untuk Amazon Lex. Jika Anda tidak mengonfigurasi jejak, Anda masih dapat melihat peristiwa terbaru di CloudTrail konsol dalam Riwayat acara. Dengan menggunakan informasi yang dikumpulkan oleh CloudTrail, Anda dapat menentukan permintaan yang dibuat untuk Amazon Lex, alamat IP dari mana permintaan itu dibuat, siapa yang membuat permintaan, kapan dibuat, dan detail tambahan.

Untuk mempelajari selengkapnya CloudTrail, termasuk cara mengonfigurasi dan mengaktifkannya, lihat Panduan AWS CloudTrail Pengguna.

Informasi Amazon Lex di CloudTrail

CloudTrail diaktifkan di AWS akun Anda saat Anda membuat akun. Ketika aktivitas acara yang didukung terjadi di Amazon Lex, aktivitas tersebut direkam dalam suatu CloudTrail peristiwa bersama dengan peristiwa AWS layanan lainnya dalam riwayat Acara. Anda dapat melihat, mencari, dan mengunduh peristiwa terbaru di akun AWS Anda. Untuk informasi selengkapnya, lihat Melihat Acara dengan Riwayat CloudTrail Acara.

Untuk catatan acara yang sedang berlangsung di AWS akun Anda, termasuk acara untuk Amazon Lex, buat jejak. Jejak memungkinkan CloudTrail untuk mengirimkan file log ke bucket Amazon Simple Storage Service (Amazon S3) Simple Storage Service (Amazon S3). Secara default, ketika Anda membuat jejak di konsol tersebut, jejak tersebut diterapkan ke semua Wilayah AWS. Jejak mencatat peristiwa dari semua Wilayah di partisi AWS dan mengirimkan berkas log ke bucket S3 yang Anda tentukan. Selain itu, Anda dapat mengonfigurasi AWS layanan lain untuk menganalisis lebih lanjut dan menindaklanjuti data peristiwa yang dikumpulkan dalam CloudTrail log. Untuk informasi selengkapnya, lihat :

- Ikhtisar 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 mendukung pencatatan operasi berikut sebagai peristiwa dalam file CloudTrail log:

- CreateBotVersion
- CreateIntentVersion
- CreateSlotTypeVersion
- DeleteBot
- DeleteBotAlias
- DeleteBotChannelAssociation
- DeleteBotVersion
- DeleteIntent
- DeleteIntentVersion
- DeleteSlotType

- DeleteSlotTypeVersion
- DeleteUtterances
- GetBot
- GetBotAlias
- GetBotAliases
- GetBotChannelAssociation
- GetBotChannelAssociations
- GetBots
- GetBotVersions
- GetBuiltinIntent
- GetBuiltinIntents
- GetBuiltinSlotTypes
- GetSlotTypeVersions
- GetUtterancesView
- PutBot
- PutBotAlias
- PutIntent
- PutSlotType

Setiap entri peristiwa atau log berisi informasi tentang siapa yang membuat permintaan tersebut. Informasi ini membantu Anda menentukan hal berikut:

- Apakah permintaan dibuat dengan root atau kredensyal pengguna
- Jika permintaan tersebut dibuat dengan kredensial keamanan sementara untuk peran atau pengguna gabungan
- Jika permintaan tersebut dibuat oleh layanan AWS lainnya

Untuk informasi lain, lihat Elemen userIdentity CloudTrail .

Untuk informasi tentang tindakan Amazon Lex yang masuk CloudTrail log dalam log, lihat Layanan Pembuatan Model Amazon Lex. Misalnya, panggilan kePutBot,GetBot, dan DeleteBot operasi menghasilkan entri di CloudTrail log. Tindakan yang didokumentasikan di Amazon Lex Runtime Service, PostContent danPostText, tidak dicatat.

Contoh: Entri File Log Amazon Lex

Trail adalah konfigurasi yang memungkinkan pengiriman peristiwa sebagai file log ke bucket 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, sehingga file tersebut tidak muncul dalam urutan tertentu.

Contoh entri CloudTrail log berikut menunjukkan hasil panggilan ke `PutBot` operasi.

```
   {
               "eventVersion": "1.05",
               "userIdentity": {
                   "type": "AssumedRole | FederatedUser | IAMUser | Root | SAMLUser |
WebIdentityUser",
                   "principalId": "principal ID",
                   "arn": "ARN",
                   "accountId": "account ID",
                   "accessKeyId": "access key ID",
                   "userName": "user name"
               },
               "eventTime": "timestamp",
               "eventSource": "lex.amazonaws.com",
               "eventName": "PutBot",
               "awsRegion": "region",
               "sourceIPAddress": "source IP address",
               "userAgent": "user agent",
               "requestParameters": {
                   "name": "CloudTrailBot",
                   "intents": [
                       {
                           "intentVersion": "11",
                           "intentName": "TestCloudTrail"
                       }
                   ],
                   "voiceId": "Salli",
                   "childDirected": false,
                   "locale": "en-US",
                   "idleSessionTTLInSeconds": 500,
                   "processBehavior": "BUILD",
                   "description": "CloudTrail test bot",
                   "clarificationPrompt": {
                       "messages": [
```

```
                            {
                                "contentType": "PlainText",
                                "content": "I didn't understand you. What would you
like to do?"
                            }
                        ],
                        "maxAttempts": 2
                    },
                    "abortStatement": {
                        "messages": [
                            {
                                "contentType": "PlainText",
                                "content": "Sorry. I'm not able to assist at this
time."
                            }
                        ]
                    }
                },
                "responseElements": {
                    "voiceId": "Salli",
                    "locale": "en-US",
                    "childDirected": false,
                    "abortStatement": {
                        "messages": [
                            {
                                "contentType": "PlainText",
                                "content": "Sorry. I'm not able to assist at this
time."
                            }
                        ]
                    },
                    "status": "BUILDING",
                    "createdDate": "timestamp",
                    "lastUpdatedDate": "timestamp",
                    "idleSessionTTLInSeconds": 500,
                    "intents": [
                        {
                            "intentVersion": "11",
                            "intentName": "TestCloudTrail"
                        }
                    ],
                    "clarificationPrompt": {
                        "messages": [
                            {
```

```
                                "contentType": "PlainText",
                                "content": "I didn't understand you. What would you
like to do?"
                            }
                        ],
                        "maxAttempts": 2
                    },
                    "version": "$LATEST",
                    "description": "CloudTrail test bot",
                    "checksum": "checksum",
                    "name": "CloudTrailBot"
                },
                "requestID": "request ID",
                "eventID": "event ID",
                "eventType": "AwsApiCall",
                "recipientAccountId": "account ID"
            }
        }
```

Validasi Kepatuhan untuk Amazon Lex

Auditor pihak ketiga menilai keamanan dan kepatuhan Amazon Lex sebagai bagian dari beberapa program AWS kepatuhan. Amazon Lex adalah layanan yang memenuhi syarat HIPAA. Ini sesuai dengan PCI, SOC, dan ISO. 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 Amazon Lex ditentukan oleh sensitivitas data Anda, tujuan kepatuhan organisasi Anda, serta hukum dan peraturan yang berlaku. Jika penggunaan Amazon Lex Anda tunduk pada kepatuhan terhadap standar seperti PCI, AWS sediakan sumber daya berikut untuk membantu:

- Panduan Memulai Cepat Keamanan dan Kepatuhan — Panduan penerapan yang membahas pertimbangan arsitektur dan memberikan langkah-langkah untuk menerapkan lingkungan dasar yang berfokus pada keamanan dan kepatuhan pada AWS
- Perancangan Laporan Resmi Keamanan dan Kepatuhan HIPAA – Laporan resmi ini menjelaskan bagaimana perusahaan dapat menggunakan AWS untuk membuat aplikasi yang patuh terhadap HIPAA.
- AWS Sumber Daya Kepatuhan - Kumpulan buku kerja dan panduan yang mungkin berlaku untuk industri dan lokasi Anda

- AWS Config— Layanan yang menilai seberapa baik konfigurasi sumber daya Anda mematuhi praktik internal, pedoman industri, dan peraturan
- AWS Security Hub— Pandangan komprehensif tentang keadaan keamanan Anda di dalamnya AWS yang membantu Anda memeriksa kepatuhan Anda terhadap standar industri keamanan dan praktik terbaik

Untuk daftar AWS layanan dalam cakupan program kepatuhan tertentu, lihat AWS Services in Scope by Compliance Program. Untuk informasi umum, lihat Program Kepatuhan AWS.

Ketahanan di Amazon Lex

Infrastruktur AWS global dibangun di sekitar AWS Wilayah dan Zona Ketersediaan. AWS Wilayah menyediakan beberapa Availability Zone yang terpisah secara fisik dan terisolasi, yang terhubung dengan latensi rendah, throughput tinggi, dan jaringan yang sangat redundan. Dengan Zona Ketersediaan, Anda dapat merancang dan mengoperasikan aplikasi dan basis data yang melakukan secara otomatis pinda saat gagal/failover di antara zona-zona tanpa terputus. Zona Ketersediaan lebih sangat tersedia, lebih toleran kesalahan, dan lebih dapat diskalakan daripadai infrastruktur pusat data tunggal atau multi tradisional.

Untuk informasi selengkapnya tentang AWS Wilayah dan Availability Zone, lihat Infrastruktur AWS Global.

Selain infrastruktur AWS global, Amazon Lex menawarkan beberapa fitur untuk membantu mendukung ketahanan data dan kebutuhan pencadangan Anda.

Keamanan Infrastruktur di Amazon Lex

Sebagai layanan terkelola, Amazon Lex dilindungi oleh prosedur keamanan jaringan AWS global yang dijelaskan dalam whitepaper Amazon Web Services: Overview of Security Processes.

Anda menggunakan panggilan AWS API yang dipublikasikan untuk mengakses Amazon Lex melalui jaringan. Klien harus mendukung TLS (Transport Layer Security) 1.0. Kami merekomendasikan TLS 1.2 atau versi yang lebih baru. Selain itu, klien harus mendukung cipher suites dengan perfect forward secrecy (PFS) seperti Ephemeral Diffie-Hellman (DHE) atau Elliptic Curve Diffie-Hellman Ephemeral (ECDHE). Sebagian besar sistem modern, misalnya Java 7 dan versi yang lebih baru, mendukung mode ini. Selain itu, permintaan harus ditandatangani menggunakan ID kunci akses dan kunci akses rahasia yang terkait dengan prinsipal IAM. Atau Anda dapat menggunakan AWS

Security Token Service (AWS STS) untuk menghasilkan kredensial keamanan sementara untuk menandatangani permintaan.

Anda dapat memanggil operasi API ini dari lokasi jaringan mana pun, tetapi Amazon Lex mendukung kebijakan akses tingkat sumber daya, yang dapat mencakup pembatasan berdasarkan alamat IP sumber. Anda juga dapat menggunakan kebijakan Amazon Lex untuk mengontrol akses dari titik akhir Amazon Virtual Private Cloud (Amazon VPC) tertentu atau VPC tertentu. Secara efektif, ini mengisolasi akses jaringan ke sumber daya Amazon Lex tertentu hanya dari VPC tertentu dalam AWS jaringan.

Pedoman dan Kuotas di Amazon Lex

Bagian berikut memberikan Pedoman dan kuota saat menggunakan Amazon Lex.

Topik

Wilayah yang didukung

Untuk daftarAWS Wilayah yang terdapat Amazon Lex, lihat Wilayah dan Titik Akhir AWS dalam Referensi Umum Amazon Web Services.

Pedoman Umum

Bagian ini menjelaskan pedoman umum saat menggunakan Amazon Lex.

- Permintaan penandatanganan — Semua operasi API pembuatan model dan runtime Amazon Lex dalamReferensi API penggunaan tanda tangan V4 untuk mengautentikasi permintaan. Untuk informasi selengkapnya tentang melakukan autentikasi permintaan, lihat Proses Penandatanganan Tanda Tangan Versi 4 dalam Referensi Umum Amazon Web Services.

  Untuk ituPostContent, Amazon Lex menggunakan opsi payload yang tidak ditandatangani yang dijelaskan dalam Perhitungan Tanda Tangan untuk Header Otorisasi: Mentransfer Muatan dalam Potongan Tunggal (AWS Signature Version 4) di Referensi API Amazon Simple Storage Service (S3).

  Saat Anda menggunakan opsi payload yang tidak ditandatangani, jangan sertakan hash payload dalam permintaan kanonik. Sebagai gantinya, Anda menggunakan string literal “UNSIGNED-PAYLOAD” sebagai hash payload. Sertakan juga header dengan nama`x-amz-content-sha256` dan nilai`UNSIGNED-PAYLOAD` dalam`PostContent` permintaan.

- Perhatikan hal berikut tentang cara Amazon Lex menangkap nilai slot dari ucapan pengguna:

  Amazon Lex menggunakan nilai pencacahan yang Anda berikan dalam definisi jenis slot untuk melatih model pembelajaran mesinnya. Misalkan Anda mendefinisikan maksud yang dipanggil`GetPredictionIntent` dengan contoh ucapan berikut:

  ```
  "Tell me the prediction for {Sign}"
  ```

  `{Sign}`Dimana slot tipe kustom`ZodiacSign`. Ini memiliki 12 nilai pencacahan,`Aries` melalui`Pisces`. Dari ucapan pengguna “Katakan padaku prediksi untuk...” Amazon Lex mengerti apa yang berikut adalah tanda zodiak.

  Ketika`valueSelectionStrategy` bidang diatur untuk`ORIGINAL_VALUE` menggunakan[PutSlotType](#) operasi, atau jika nilai Expand dipilih di konsol, jika pengguna mengatakan “Beri tahu saya prediksi untuk bumi”, Amazon Lex menyimpulkan bahwa “earth” adalah`ZodiacSign` dan meneruskannya ke aplikasi klien Anda atau fungsi Lambda. Anda harus memeriksa bahwa nilai slot memiliki nilai yang valid sebelum menggunakannya dalam aktivitas pemenuhan Anda.

  Jika Anda mengatur`valueSelectionStrategy` bidang untuk`TOP_RESOLUTION` menggunakan[PutSlotType](#) operasi, atau jika Batasi ke nilai slot dan sinonim dipilih di konsol, nilai yang dikembalikan terbatas pada nilai yang Anda tentukan untuk jenis slot. Misalnya, jika pengguna mengatakan “Beri tahu saya prediksi untuk bumi” nilainya tidak akan dikenali karena itu bukan salah satu nilai yang ditentukan untuk jenis slot. Ketika Anda mendefinisikan sinonim untuk nilai slot, mereka diakui sama dengan nilai slot, namun, nilai slot dikembalikan bukan sinonim.

  Ketika Amazon Lex memanggil fungsi Lambda atau mengembalikan hasil interaksi pidato dengan aplikasi klien Anda, kasus nilai slot tidak dijamin. Misalnya, jika Anda memunculkan nilai untuk [Amazon.Movie](#) built-in jenis Slot, dan pengguna mengatakan atau jenis “Gone with the wind,”

Amazon Lex mungkin kembali “Gone with the Wind,” “pergi dengan angin,” atau “Gone With The Wind.” Dalam interaksi teks, kasus nilai slot cocok dengan teks yang dimasukkan atau nilai slot, tergantung pada nilai`valueResolutionStrategy` bidang.

- Saat mendefinisikan nilai slot yang berisi akronim, gunakan pola berikut:
  - Huruf kapital dipisahkan oleh periode (D.V.D.)
  - Huruf kapital dipisahkan oleh spasi (D V D)

- Amazon Lex tidak mendukung AMAZON.LITERAL built-in jenis slot yang didukung Alexa Skills Kit. Namun, Amazon Lex mendukung pembuatan jenis slot khusus yang dapat Anda gunakan untuk mengimplementasikan fungsionalitas ini. Seperti disebutkan dalam bullet sebelumnya, Anda dapat menangkap nilai-nilai di luar definisi jenis slot kustom. Tambahkan nilai pencacahan yang lebih banyak dan beragam untuk meningkatkan akurasi pengenalan suara otomatis (ASR) dan pemahaman bahasa alami (NLU).

- AMAZON.DATE dan AMAZON.TIME built-in jenis slot menangkap kedua tanggal mutlak dan relatif dan waktu. Tanggal dan waktu relatif diselesaikan di wilayah tempat Amazon Lex memproses permintaan.

  Untuk jenis slot`AMAZON.TIME` bawaan, jika pengguna tidak menentukan bahwa waktu sebelum atau sesudah tengah hari, waktunya ambigu dan Amazon Lex akan meminta pengguna lagi. Kami merekomendasikan prompt yang memunculkan waktu absolut. Misalnya, gunakan prompt seperti “Kapan Anda ingin pizza Anda dikirimkan? Anda dapat mengatakan 6 PM atau 6 di malam hari.”

- Menyediakan data pelatihan yang membingungkan di bot Anda mengurangi kemampuan Amazon Lex untuk memahami masukan pengguna. Pertimbangkan contoh berikut:

  Misalkan Anda memiliki dua maksud (`OrderPizza`dan`OrderDrink`) di bot Anda dan keduanya dikonfigurasi dengan ucapan “Saya ingin memesan”. Ucapan ini tidak memetakan ke maksud tertentu yang dapat dipelajari Amazon Lex sambil membangun model bahasa untuk bot pada

waktu pembuatan. Akibatnya, ketika pengguna memasukkan ucapan ini saat runtime, Amazon Lex tidak dapat memilih maksud dengan tingkat kepercayaan yang tinggi.

Pertimbangkan contoh lain di mana Anda menentukan maksud khusus untuk mendapatkan konfirmasi dari pengguna (misalnya,`MyCustomConfirmationIntent`) dan mengonfigurasi maksud dengan ucapan “Ya” dan “Tidak.” Perhatikan bahwa Amazon Lex juga memiliki model bahasa untuk memahami konfirmasi pengguna. Hal ini dapat menciptakan situasi yang saling bertentangan. Ketika pengguna merespons dengan “Ya”, apakah ini berarti bahwa ini adalah konfirmasi untuk maksud yang sedang berlangsung atau bahwa pengguna meminta maksud khusus yang Anda buat?

Secara umum, contoh ucapan yang Anda berikan harus memetakan ke maksud tertentu dan, secara opsional, ke nilai slot tertentu.

- Operasi API runtimePostContent danPostText mengambil ID pengguna sebagai parameter yang diperlukan. 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 (mungkin ada beberapa pengguna yang memesan pizza).

- Jika aplikasi klien Anda menggunakan Amazon Cognito untuk autentikasi, Anda dapat menggunakan ID pengguna Amazon Cognito sebagai ID pengguna Amazon Lex. Perhatikan bahwa setiap fungsi Lambda yang dikonfigurasi untuk bot Anda harus memiliki mekanisme autentikasi sendiri untuk mengidentifikasi pengguna yang atas nama Amazon Lex menjalankan fungsi Lambda.

- Kami mendorong Anda untuk menentukan maksud yang menangkap niat pengguna untuk menghentikan percakapan. Misalnya, Anda dapat menentukan intent (`NothingIntent`) dengan contoh ucapan (“Saya tidak menginginkan apa pun”, “exit”, “bye bye”), tidak ada slot, dan tidak ada

fungsi Lambda yang dikonfigurasi sebagai pengait kode. Ini memungkinkan pengguna menutup percakapan dengan anggun.

Quotas

Bagian ini menjelaskan kuota saat ini di Amazon Lex. Kuota ini dikelompokkan berdasarkan kategori.

Kuota layanan dapat disesuaikan atau ditingkatkan. Hubungi dukunganAWS pelanggan untuk menambah kuota. Perlu waktu beberapa hari untuk menambah kuota layanan. Jika Anda meningkatkan kuota sebagai bagian dari proyek yang lebih besar, pastikan untuk menambahkan waktu ini ke paket Anda.

Topik

- Service Quotas Runtime
- Kuotas Bangunan

Service Quotas Runtime

Selain kuota yang dijelaskan dalam referensi API, perhatikan hal berikut:

Kuota API

- Masukan ucapan kePostContent operasi bisa sampai 15 detik.

- Dalam operasi API runtimePostContent danPostText, ukuran teks input dapat mencapai 1024 karakter Unicode.

- Ukuran maksimum`PostContent` header adalah 16 KB. Ukuran maksimum header permintaan dan sesi gabungan adalah 12 KB.

- Saat menggunakan`PostContent` atau`PostText` operasi dalam mode teks, jumlah maksimum percakapan bersamaan dengan bot adalah 2 untuk`$LATEST` alias dan 50 untuk semua alias lainnya. Kuota berlaku secara terpisah untuk setiap API.

- Saat menggunakan`PostContent` operasi dalam mode suara, jumlah maksimum percakapan mode teks bersamaan dengan bot adalah 2 untuk`$LATEST` alias dan 125 untuk semua alias lainnya. Kuota berlaku secara terpisah untuk setiap API.

- Jumlah maksimum panggilan manajemen sesi bersamaan (PutSession,GetSession, andDeleteSession) adalah 2 untuk`$LATEST` alias bot dan 50 untuk semua alias lainnya.

- Ukuran input maksimum ke fungsi Lambda adalah 12 KB. Ukuran output maksimum adalah 25 KB, dimana 12 KB dapat berupa atribut sesi.

Menggunakan**`$LATEST`** versi

- `$LATEST`Versi bot Anda hanya boleh digunakan untuk pengujian manual. Amazon Lex membatasi jumlah permintaan waktu proses yang dapat Anda buat ke`$LATEST` versi bot.

- Saat Anda memperbarui`$LATEST` versi bot, Amazon Lex mengakhiri percakapan yang sedang berlangsung untuk aplikasi klien apa pun menggunakan`$LATEST` versi bot. Umumnya, Anda tidak boleh menggunakan`$LATEST` versi bot dalam produksi karena`$LATEST` versi dapat diperbarui. Anda harus mempublikasikan versi dan menggunakannya sebagai gantinya.

- Saat Anda memperbarui alias, Amazon Lex membutuhkan waktu beberapa menit untuk mengambil perubahan. Saat Anda memodifikasi`$LATEST` versi bot, perubahan akan segera diambil.

Waktu Habisnya Sesi

- Waktu tunggu sesi yang ditetapkan saat bot dibuat menentukan berapa lama bot mempertahankan konteks percakapan, seperti maksud pengguna saat ini dan data slot.

- Setelah pengguna memulai percakapan dengan bot Anda dan sampai sesi berakhir, Amazon Lex menggunakan versi bot yang sama, bahkan jika Anda memperbarui alias bot untuk menunjuk ke versi lain.

Kuotas Bangunan

Bangunan model mengacu pada pembuatan dan pengelolaan bot. Ini termasuk membuat dan mengelola bot, maksud, jenis slot, slot, dan asosiasi saluran bot.

Topik

Kuotas Bot

- Anda mengonfigurasi prompt dan pernyataan di seluruh API pembuatan model. Masing-masing petunjuk atau pernyataan ini dapat memiliki hingga lima pesan dan setiap pesan dapat berisi 1 hingga 1000 karakter UTF-8.

- Bila menggunakan grup pesan, Anda dapat menentukan hingga lima grup pesan untuk setiap pesan. Setiap grup pesan dapat berisi maksimal lima pesan, dan Anda dibatasi hingga 15 pesan di semua grup pesan.

- Anda dapat menentukan ucapan sampel untuk maksud dan slot. Anda dapat menggunakan maksimal 200.000 karakter untuk semua ucapan.

- Setiap jenis slot dapat menentukan maksimal 10.000 nilai dan Sinonim. Setiap bot dapat berisi maksimum 50.000 nilai jenis slot dan sinonim.

- Nama asosiasi bot, alias, dan saluran bot tidak peka huruf pada saat pembuatan. Jika Anda membuat`PizzaBot` dan kemudian mencoba membuat`pizzaBot`, Anda akan mendapatkan kesalahan. Namun, saat mengakses sumber daya, nama sumber daya sensitif huruf besar, Anda harus menentukan`PizzaBot` dan tidak`pizzaBot`. Nama-nama ini harus antara 2 dan 50 karakter ASCII.

- Jumlah maksimum versi yang dapat Anda publikasikan untuk semua jenis sumber daya adalah 100. Perhatikan bahwa tidak ada versi untuk alias.

- Dalam bot, nama maksud dan nama slot harus unik, Anda tidak dapat memiliki maksud dan slot dengan nama yang sama.

- Anda dapat membuat bot yang dikonfigurasi untuk mendukung beberapa intent. Jika dua intent memiliki slot dengan nama yang sama, maka jenis slot yang sesuai harus sama.

  Misalnya, Anda membuat bot untuk mendukung dua intent (`OrderPizza`dan`OrderDrink`). Jika kedua maksud ini memiliki`size` slot, maka jenis slot harus sama di kedua tempat.

  Selain itu, ucapan sampel yang Anda berikan untuk slot di salah satu maksud berlaku untuk slot dengan nama yang sama dalam maksud lain.

- Anda dapat mengaitkan maksimal 250 intent dengan bot.

- Saat membuat bot, Anda menentukan waktu Habisnya sesi. Waktu tunggu sesi bisa antara satu menit dan satu hari. Default-nya adalah lima menit.

- Anda dapat membuat hingga lima alias untuk bot.

- Anda dapat membuat hingga 250 bot untuk setiap akun AWS.

- Anda tidak dapat membuat beberapa maksud yang diperluas dari intent bawaan yang sama.

Kuotas Niat

- Intent dan nama slot adalah kasus tidak sensitif pada saat pembuatan. Artinya, jika Anda membuat`0rderPizza` maksud dan sekali lagi mencoba membuat`orderPizza` maksud lain, Anda akan mendapatkan kesalahan. Namun, saat mengakses sumber daya ini, nama sumber daya sensitif huruf, tentukan`0rderPizza` dan tidak`orderPizza`. Panjang nama harus berkisar antara 1 sampai 100 karakter ASCII.

- Maksud dapat memiliki hingga 1.500 ucapan sampel hingga 1.500 ucapan sampel hingga 1.500 ucapan sampel. Minimal satu ucapan sampel diperlukan minimal satu ucapan sampel. Setiap ucapan sampel dapat mencapai 200 karakter UTF-8. Anda dapat menggunakan hingga 200.000 karakter untuk semua ucapan maksud dan slot dalam bot. Contoh ucapan untuk maksud:
  - Dapat merujuk ke nol atau lebih nama slot.
  - Dapat merujuk ke nama slot hanya sekali.

  Misalnya:

  ```
  I want a pizza
  I want a {pizzaSize} pizza
  I want a {pizzaSize} {pizzaTopping} pizza
  ```

- Meskipun setiap maksud mendukung hingga 1.500 ucapan, jika Anda menggunakan lebih sedikit ucapan Amazon Lex mungkin memiliki kemampuan yang lebih baik untuk mengenali input di luar set yang Anda sediakan.

- Anda dapat membuat hingga lima grup pesan untuk setiap pesan dalam satu intent. Ada total 15 pesan di semua grup pesan untuk pesan.

- Konsol hanya dapat membuat grup pesan untuk`conclusionStatement` dan`followUpPrompt` pesan. Anda dapat membuat grup pesan untuk pesan lain menggunakan Amazon Lex API.

- Setiap slot dapat memiliki hingga 10 ucapan sampel hingga 10 ucapan sampel. Setiap ucapan sampel harus mengacu pada nama slot tepat sekali. Misalnya:

  ```
  {pizzaSize} please
  ```

- Setiap bot dapat memiliki maksimal 200.000 karakter untuk gabungan maksud dan ucapan slot.

- Anda tidak dapat memberikan ucapan untuk maksud yang meluas dari intent bawaan. Untuk semua maksud lain, Anda harus memberikan setidaknya satu ucapan sampel. Maksud berisi slot, tetapi ucapan sampel tingkat slot adalah opsional.

- Maksud bawaan
  - Saat ini, Amazon Lex tidak mendukung elicitation slot untuk intent bawaan. Anda tidak dapat membuat fungsi Lambda untuk mengembalikan`ElicitSlot` direktif dalam respons dengan maksud yang berasal dari intent bawaan. Untuk informasi selengkapnya, lihat Format Respons.
  - Layanan ini tidak mendukung penambahan ucapan sampel ke intent bawaan. Demikian pula, Anda tidak dapat menambah atau menghapus slot ke intent bawaan.

- Anda dapat membuat hingga 1.000 intent per akun AWS. Anda dapat membuat hingga 100 slot dalam satu maksud.

Kuota Jenis Slot

- Nama tipe slot adalah kasus tidak sensitif pada saat pembuatan. Jika Anda membuat jenis`PizzaSize` slot dan sekali lagi mencoba untuk membuat jenis`pizzaSize` slot, Anda akan mendapatkan kesalahan. Namun, saat mengakses sumber daya ini, nama sumber daya sensitif huruf (Anda harus menentukan`PizzaSize` dan tidak`pizzaSize`). Panjang nama harus berkisar antara 1 sampai 100 karakter ASCII.

- Jenis slot kustom yang Anda buat dapat memiliki maksimal 10.000 nilai pencacahan dan Sinonim. Setiap nilai dapat memiliki panjang hingga 140 karakter UTF-8 karakter. Nilai pencacahan dan sinonim tidak dapat berisi duplikat.

- Untuk nilai jenis slot, jika sesuai, tentukan huruf besar dan kecil. Misalnya, untuk jenis slot yang disebut`Procedure`, jika nilainya`MRI`, tentukan “MRI” dan “mri” sebagai nilai.

- Jenis slot bawaan — Saat ini, Amazon Lex tidak mendukung penambahan nilai pencacahan atau sinonim untuk jenis slot bawaan.

Referensi API

Bagian ini menyediakan dokumentasi untuk operasi Amazon Lex API. Untuk daftar Wilayah AWS tempat Amazon Lex tersedia, lihat Wilayah dan Titik Akhir AWS di Referensi Umum Amazon Web Services.

Topik

- Tindakan
- Tipe Data

Tindakan

Tindakan berikut didukung oleh Amazon Lex Model Building Service:

- CreateBotVersion
- CreateIntentVersion
- CreateSlotTypeVersion
- DeleteBot
- DeleteBotAlias
- DeleteBotChannelAssociation
- DeleteBotVersion
- DeleteIntent
- DeleteIntentVersion
- DeleteSlotType
- DeleteSlotTypeVersion
- DeleteUtterances
- GetBot
- GetBotAlias
- GetBotAliases
- GetBotChannelAssociation
- GetBotChannelAssociations
- GetBots

- GetBotVersions
- GetBuiltinIntent
- GetBuiltinIntents
- GetBuiltinSlotTypes
- GetExport
- GetImport
- GetIntent
- GetIntents
- GetIntentVersions
- GetMigration
- GetMigrations
- GetSlotType
- GetSlotTypes
- GetSlotTypeVersions
- GetUtterancesView
- ListTagsForResource
- PutBot
- PutBotAlias
- PutIntent
- PutSlotType
- StartImport
- StartMigration
- TagResource
- UntagResource

Tindakan berikut didukung oleh Amazon Lex Runtime Service:

- DeleteSession
- GetSession
- PostContent
- PostText

- PutSession

Amazon Lex Model Bangunan Layanan

Tindakan berikut didukung oleh Amazon Lex Model Building Service:

- CreateBotVersion
- CreateIntentVersion
- CreateSlotTypeVersion
- DeleteBot
- DeleteBotAlias
- DeleteBotChannelAssociation
- DeleteBotVersion
- DeleteIntent
- DeleteIntentVersion
- DeleteSlotType
- DeleteSlotTypeVersion
- DeleteUtterances
- GetBot
- GetBotAlias
- GetBotAliases
- GetBotChannelAssociation
- GetBotChannelAssociations
- GetBots
- GetBotVersions
- GetBuiltinIntent
- GetBuiltinIntents
- GetBuiltinSlotTypes
- GetExport
- GetImport
- GetIntent
- GetIntents

- GetIntentVersions
- GetMigration
- GetMigrations
- GetSlotType
- GetSlotTypes
- GetSlotTypeVersions
- GetUtterancesView
- ListTagsForResource
- PutBot
- PutBotAlias
- PutIntent
- PutSlotType
- StartImport
- StartMigration
- TagResource
- UntagResource

CreateBotVersion

Layanan: Amazon Lex Model Building Service

Membuat versi baru bot berdasarkan `$LATEST` versi. Jika `$LATEST` versi sumber daya ini tidak berubah sejak Anda membuat versi terakhir, Amazon Lex tidak membuat versi baru. Ia mengembalikan versi yang terakhir dibuat.

> **Note**
>
> Anda hanya dapat memperbarui `$LATEST` versi bot. Anda tidak dapat memperbarui versi bernomor yang Anda buat dengan `CreateBotVersion` operasi.

Saat Anda membuat versi pertama bot, Amazon Lex menetapkan versi ke 1. Versi selanjutnya meningkat sebesar 1. Untuk informasi selengkapnya, lihat Versioning.

Operasi ini memerlukan izin untuk tindakan `lex:CreateBotVersion`.

Minta Sintaks

```
POST /bots/name/versions HTTP/1.1
Content-type: application/json

{
   "checksum": "string"
}
```

Parameter Permintaan URI

Permintaan menggunakan parameter URI berikut.

name

Nama bot yang ingin Anda buat versi baru. Namanya peka huruf besar/kecil.

Kendala Panjang: Panjang minimum 2. Panjang maksimum 50.

Pola: `^([A-Za-z]_?)+$`

Wajib: Ya

Isi Permintaan

Permintaan menerima data berikut dalam format JSON.

checksum

Mengidentifikasi revisi spesifik dari $LATEST versi bot. Jika Anda menentukan checksum dan $LATEST versi bot memiliki checksum yang berbeda, PreconditionFailedException pengecualian akan dikembalikan dan Amazon Lex tidak menerbitkan versi baru. Jika Anda tidak menentukan checksum, Amazon Lex menerbitkan versinya. $LATEST

Tipe: String

Wajib: Tidak

Sintaksis Respons

```
HTTP/1.1 201
Content-type: application/json

{
   "abortStatement": {
      "messages": [
         {
            "content": "string",
            "contentType": "string",
            "groupNumber": number
         }
      ],
      "responseCard": "string"
   },
   "checksum": "string",
   "childDirected": boolean,
   "clarificationPrompt": {
      "maxAttempts": number,
      "messages": [
         {
            "content": "string",
            "contentType": "string",
            "groupNumber": number
         }
      ],
      "responseCard": "string"
```

```
    },
    "createdDate": number,
    "description": "string",
    "detectSentiment": boolean,
    "enableModelImprovements": boolean,
    "failureReason": "string",
    "idleSessionTTLInSeconds": number,
    "intents": [
       {
          "intentName": "string",
          "intentVersion": "string"
       }
    ],
    "lastUpdatedDate": number,
    "locale": "string",
    "name": "string",
    "status": "string",
    "version": "string",
    "voiceId": "string"
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respon HTTP 201.

Layanan mengembalikan data berikut dalam format JSON.

abortStatement

Pesan yang digunakan Amazon Lex untuk membatalkan percakapan. Untuk informasi selengkapnya, lihat PutBot.

Tipe: Objek Statement

checksum

Checksum mengidentifikasi versi bot yang dibuat.

Jenis: String

childDirected

Untuk setiap bot Amazon Lex yang dibuat dengan Layanan Pembuatan Model Amazon Lex, Anda harus menentukan apakah penggunaan Amazon Lex Anda terkait dengan situs web, program,

atau aplikasi lain yang diarahkan atau ditargetkan, secara keseluruhan atau sebagian, untuk anak-anak di bawah usia 13 tahun dan tunduk pada Undang-Undang Perlindungan Privasi Online Anak (COPPA) dengan menentukan `true` atau `false` di lapangan. `childDirected` Dengan menentukan **`true`** di **`childDirected`** lapangan, Anda mengonfirmasi bahwa penggunaan Amazon Lex Anda terkait dengan situs web, program, atau aplikasi lain yang diarahkan atau ditargetkan, secara keseluruhan atau sebagian, untuk anak-anak di bawah usia 13 tahun dan tunduk pada COPPA. Dengan menentukan `false` di `childDirected` lapangan, Anda mengonfirmasi bahwa penggunaan Amazon Lex Anda tidak terkait dengan situs web, program, atau aplikasi lain yang diarahkan atau ditargetkan, secara keseluruhan atau sebagian, untuk anak-anak di bawah usia 13 tahun dan tunduk pada COPPA. Anda tidak boleh menentukan nilai default untuk `childDirected` bidang yang tidak secara akurat mencerminkan apakah penggunaan Amazon Lex Anda terkait dengan situs web, program, atau aplikasi lain yang diarahkan atau ditargetkan, secara keseluruhan atau sebagian, untuk anak-anak di bawah usia 13 tahun dan tunduk pada COPPA.

Jika penggunaan Amazon Lex Anda terkait dengan situs web, program, atau aplikasi lain yang diarahkan secara keseluruhan atau sebagian, untuk anak-anak di bawah usia 13 tahun, Anda harus mendapatkan persetujuan orang tua yang dapat diverifikasi berdasarkan COPPA. Untuk informasi mengenai penggunaan Amazon Lex sehubungan dengan situs web, program, atau aplikasi lain yang diarahkan atau ditargetkan, secara keseluruhan atau sebagian, untuk anak-anak di bawah usia 13 tahun, lihat FAQ Amazon Lex.

Jenis: Boolean

clarificationPrompt

Pesan yang digunakan Amazon Lex saat tidak memahami permintaan pengguna. Untuk informasi selengkapnya, lihat PutBot.

Tipe: Objek Prompt

createdDate

Tanggal ketika versi bot dibuat.

Tipe: Timestamp

description

Deskripsi bot.

Jenis: String

Batasan Panjang: Panjang minimum 0. Panjang maksimum 200.

detectSentiment

Menunjukkan apakah ucapan yang dimasukkan oleh pengguna harus dikirim ke Amazon Comprehend untuk analisis sentimen.

Jenis: Boolean

enableModelImprovements

Menunjukkan apakah bot menggunakan peningkatan akurasi. `true`menunjukkan bahwa bot menggunakan perbaikan, jika tidak,`false`.

Jenis: Boolean

failureReason

Jika `status` ya`FAILED`, Amazon Lex memberikan alasan bahwa ia gagal membangun bot.

Jenis: String

idleSessionTTLInSeconds

Waktu maksimum dalam hitungan detik Amazon Lex menyimpan data yang dikumpulkan dalam percakapan. Untuk informasi selengkapnya, lihat PutBot.

Jenis: Integer

Rentang yang Valid: Nilai minimum 60. Nilai maksimum 86400.

intents

Susunan objek `Intent`. Untuk informasi selengkapnya, lihat PutBot.

Tipe: Array objek Intent

lastUpdatedDate

Tanggal ketika `$LATEST` versi bot ini diperbarui.

Tipe: Timestamp

locale

Menentukan lokal target untuk bot.

Jenis: String

Nilai yang Valid: `de-DE | en-AU | en-GB | en-IN | en-US | es-419 | es-ES | es-US | fr-FR | fr-CA | it-IT | ja-JP | ko-KR`

name

Nama bot.

Jenis: String

Kendala Panjang: Panjang minimum 2. Panjang maksimum 50.

Pola: `^([A-Za-z]_?)+$`

status

Saat Anda mengirim permintaan untuk membuat atau memperbarui bot, Amazon Lex menyetel elemen `status` respons ke`BUILDING`. Setelah Amazon Lex membangun bot, itu ditetapkan `status` ke`READY`. Jika Amazon Lex tidak dapat membangun bot, itu akan diatur `status` ke`FAILED`. Amazon Lex mengembalikan alasan kegagalan elemen `failureReason` respons.

Jenis: String

Nilai yang Valid: `BUILDING | READY | READY_BASIC_TESTING | FAILED | NOT_BUILT`

version

Versi bot.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum adalah 64.

Pola: `\$LATEST|[0-9]+`

voiceId

ID suara Amazon Polly yang digunakan Amazon Lex untuk interaksi suara dengan pengguna.

Jenis: String

Kesalahan

BadRequestException

Permintaan tidak terbentuk dengan baik. Misalnya, nilai tidak valid atau bidang wajib hilang. Periksa nilai bidang, dan coba lagi.

Kode Status HTTP: 400

ConflictException

Ada konflik memproses permintaan. Coba permintaan Anda lagi.

Kode Status HTTP: 409

InternalFailureException

Terjadi kesalahan internal Amazon Lex. Coba permintaan Anda lagi.

Kode Status HTTP: 500

LimitExceededException

Permintaan melebihi batas. Coba permintaan Anda lagi.

Kode Status HTTP: 429

NotFoundException

Sumber daya yang ditentukan dalam permintaan tidak ditemukan. Periksa sumber daya dan coba lagi.

Kode Status HTTP: 404

PreconditionFailedException

Checksum sumber daya yang Anda coba ubah tidak cocok dengan checksum dalam permintaan. Periksa checksum sumber daya dan coba lagi.

Kode Status HTTP: 412

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2

- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

CreateIntentVersion

Layanan: Amazon Lex Model Building Service

Membuat versi baru dari intent berdasarkan `$LATEST` versi intent. Jika `$LATEST` versi intent ini tidak berubah sejak terakhir kali Anda memperbaruinya, Amazon Lex tidak membuat versi baru. Ini mengembalikan versi terakhir yang Anda buat.

> **Note**
>
> Anda hanya dapat memperbarui `$LATEST` versi intent. Anda tidak dapat memperbarui versi bernomor yang Anda buat dengan `CreateIntentVersion` operasi.

Saat Anda membuat versi intent, Amazon Lex menyetel versi ke 1. Versi selanjutnya meningkat sebesar 1. Untuk informasi selengkapnya, lihat Versioning.

Operasi ini memerlukan izin untuk menjalankan tindakan `lex:CreateIntentVersion`.

Minta Sintaks

```
POST /intents/name/versions HTTP/1.1
Content-type: application/json

{
   "checksum": "string"
}
```

Parameter Permintaan URI

Permintaan menggunakan parameter URI berikut.

name

Nama maksud yang ingin Anda buat versi baru. Namanya peka huruf besar/kecil.

Batasan Panjang: Panjang minimum 1. Panjang maksimum 100.

Pola: `^([A-Za-z]_?)+$`

Wajib: Ya

Isi Permintaan

Permintaan menerima data berikut dalam format JSON.

checksum

Checksum $LATEST versi intent yang harus digunakan untuk membuat versi baru. Jika Anda menentukan checksum dan $LATEST versi intent memiliki checksum yang berbeda, Amazon Lex mengembalikan PreconditionFailedException pengecualian dan tidak memublikasikan versi baru. Jika Anda tidak menentukan checksum, Amazon Lex menerbitkan versinya. $LATEST

Tipe: String

Wajib: Tidak

Sintaksis Respons

```
HTTP/1.1 201
Content-type: application/json

{
   "checksum": "string",
   "conclusionStatement": {
      "messages": [
         {
            "content": "string",
            "contentType": "string",
            "groupNumber": number
         }
      ],
      "responseCard": "string"
   },
   "confirmationPrompt": {
      "maxAttempts": number,
      "messages": [
         {
            "content": "string",
            "contentType": "string",
            "groupNumber": number
         }
      ],
      "responseCard": "string"
   },
```

```
  "createdDate": number,
  "description": "string",
  "dialogCodeHook": {
     "messageVersion": "string",
     "uri": "string"
  },
  "followUpPrompt": {
     "prompt": {
        "maxAttempts": number,
        "messages": [
           {
              "content": "string",
              "contentType": "string",
              "groupNumber": number
           }
        ],
        "responseCard": "string"
     },
     "rejectionStatement": {
        "messages": [
           {
              "content": "string",
              "contentType": "string",
              "groupNumber": number
           }
        ],
        "responseCard": "string"
     }
  },
  "fulfillmentActivity": {
     "codeHook": {
        "messageVersion": "string",
        "uri": "string"
     },
     "type": "string"
  },
  "inputContexts": [
     {
        "name": "string"
     }
  ],
  "kendraConfiguration": {
     "kendraIndex": "string",
     "queryFilterString": "string",
```

```
      "role": "string"
   },
   "lastUpdatedDate": number,
   "name": "string",
   "outputContexts": [
      {
         "name": "string",
         "timeToLiveInSeconds": number,
         "turnsToLive": number
      }
   ],
   "parentIntentSignature": "string",
   "rejectionStatement": {
      "messages": [
         {
            "content": "string",
            "contentType": "string",
            "groupNumber": number
         }
      ],
      "responseCard": "string"
   },
   "sampleUtterances": [ "string" ],
   "slots": [
      {
         "defaultValueSpec": {
            "defaultValueList": [
               {
                  "defaultValue": "string"
               }
            ]
         },
         "description": "string",
         "name": "string",
         "obfuscationSetting": "string",
         "priority": number,
         "responseCard": "string",
         "sampleUtterances": [ "string" ],
         "slotConstraint": "string",
         "slotType": "string",
         "slotTypeVersion": "string",
         "valueElicitationPrompt": {
            "maxAttempts": number,
            "messages": [
```

```
                {
                   "content": "string",
                   "contentType": "string",
                   "groupNumber": number
                }
             ],
             "responseCard": "string"
          }
       }
    ],
    "version": "string"
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respon HTTP 201.

Layanan mengembalikan data berikut dalam format JSON.

checksum

Checksum dari versi intent yang dibuat.

Jenis: String

conclusionStatement

Setelah fungsi Lambda yang ditentukan dalam `fulfillmentActivity` bidang memenuhi intent, Amazon Lex menyampaikan pernyataan ini kepada pengguna.

Tipe: Objek Statement

confirmationPrompt

Jika ditentukan, prompt yang digunakan Amazon Lex untuk mengonfirmasi maksud pengguna sebelum memenuhinya.

Tipe: Objek Prompt

createdDate

Tanggal dimana niat itu dibuat.

Tipe: Timestamp

description

Deskripsi niat.

Jenis: String

Batasan Panjang: Panjang minimum 0. Panjang maksimum 200.

dialogCodeHook

Jika ditentukan, Amazon Lex memanggil fungsi Lambda ini untuk setiap input pengguna.

Tipe: Objek CodeHook

followUpPrompt

Jika didefinisikan, Amazon Lex menggunakan prompt ini untuk meminta aktivitas pengguna tambahan setelah intent terpenuhi.

Tipe: Objek FollowUpPrompt

fulfillmentActivity

Menjelaskan bagaimana niat terpenuhi.

Tipe: Objek FulfillmentActivity

inputContexts

Larik `InputContext` objek yang mencantumkan konteks yang harus aktif untuk Amazon Lex untuk memilih maksud dalam percakapan dengan pengguna.

Tipe: Array objek InputContext

Anggota Array: Jumlah minimum 0 item. Jumlah maksimum 5 item.

kendraConfiguration

Informasi konfigurasi, jika ada, untuk menghubungkan indeks Amazon Kendra dengan maksud. `AMAZON.KendraSearchIntent`

Tipe: Objek KendraConfiguration

lastUpdatedDate

Tanggal dimana intent diperbarui.

Tipe: Timestamp

name

Nama niat.

Jenis: String

Panjang Batasan: Panjang minimum 1. Panjang maksimum 100.

Pola: `^([A-Za-z]_?)+$`

outputContexts

Array `OutputContext` objek yang mencantumkan konteks yang mengaktifkan intent saat intent terpenuhi.

Tipe: Array objek OutputContext

Anggota Array: Jumlah minimum 0 item. Jumlah maksimum 10 item.

parentIntentSignature

Pengenal unik untuk maksud bawaan.

Jenis: String

rejectionStatement

Jika pengguna menjawab “tidak” untuk pertanyaan yang didefinisikan dalam`confirmationPrompt`, Amazon Lex merespons dengan pernyataan ini untuk mengakui bahwa maksud tersebut dibatalkan.

Tipe: Objek Statement

sampleUtterances

Larik contoh ucapan yang dikonfigurasi untuk maksud.

Tipe: Array string.

Anggota Array: Jumlah minimum 0 item. Jumlah maksimum 1500 item.

Batasan Panjang: Panjang minimum 1. Panjang maksimum 200.

slots

Array jenis slot yang mendefinisikan informasi yang diperlukan untuk memenuhi maksud.

Tipe: Array objek Slot

Anggota Array: Jumlah minimum 0 item. Jumlah maksimum 100 item.

version

Nomor versi yang ditetapkan ke versi baru dari intent.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum adalah 64.

Pola: `\$LATEST|[0-9]+`

Kesalahan

BadRequestException

Permintaan tidak terbentuk dengan baik. Misalnya, nilai tidak valid atau bidang wajib hilang. Periksa nilai bidang, dan coba lagi.

Kode Status HTTP: 400

ConflictException

Ada konflik yang memproses permintaan tersebut. Coba permintaan Anda lagi.

Kode Status HTTP: 409

InternalFailureException

Terjadi kesalahan internal Amazon Lex. Coba permintaan Anda lagi.

Kode Status HTTP: 500

LimitExceededException

Permintaan melebihi batas. Coba permintaan Anda lagi.

Kode Status HTTP: 429

NotFoundException

Sumber daya yang ditentukan dalam permintaan tidak ditemukan. Periksa sumber daya dan coba lagi.

Kode Status HTTP: 404

PreconditionFailedException

Checksum sumber daya yang Anda coba ubah tidak cocok dengan checksum dalam permintaan. Periksa checksum sumber daya dan coba lagi.

Kode Status HTTP: 412

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

CreateSlotTypeVersion

Layanan: Amazon Lex Model Building Service

Membuat versi baru dari jenis slot berdasarkan `$LATEST` versi jenis slot yang ditentukan. Jika `$LATEST` versi sumber daya ini tidak berubah sejak versi terakhir yang Anda buat, Amazon Lex tidak membuat versi baru. Ini mengembalikan versi terakhir yang Anda buat.

> **Note**
>
> Anda hanya dapat memperbarui `$LATEST` versi jenis slot. Anda tidak dapat memperbarui versi bernomor yang Anda buat dengan `CreateSlotTypeVersion` operasi.

Saat Anda membuat versi jenis slot, Amazon Lex menetapkan versi ke 1. Versi selanjutnya meningkat sebesar 1. Untuk informasi selengkapnya, lihat Versioning.

Operasi ini memerlukan izin utuk tindakan `lex:CreateSlotTypeVersion`.

Minta Sintaks

```
POST /slottypes/name/versions HTTP/1.1
Content-type: application/json

{
   "checksum": "string"
}
```

Parameter Permintaan URI

Permintaan menggunakan parameter URI berikut.

name

Nama jenis slot yang ingin Anda buat versi baru. Namanya peka huruf besar/kecil.

Batasan Panjang: Panjang minimum 1. Panjang maksimum 100.

Pola: `^([A-Za-z]_?)+$`

Wajib: Ya

Isi Permintaan

Permintaan menerima data berikut dalam format JSON.

checksum

Checksum untuk $LATEST versi jenis slot yang ingin Anda publikasikan. Jika Anda menentukan checksum dan $LATEST versi jenis slot memiliki checksum yang berbeda, Amazon Lex mengembalikan PreconditionFailedException pengecualian dan tidak mempublikasikan versi baru. Jika Anda tidak menentukan checksum, Amazon Lex menerbitkan versinya. $LATEST

Tipe: String

Wajib: Tidak

Sintaksis Respons

```
HTTP/1.1 201
Content-type: application/json

{
   "checksum": "string",
   "createdDate": number,
   "description": "string",
   "enumerationValues": [
      {
         "synonyms": [ "string" ],
         "value": "string"
      }
   ],
   "lastUpdatedDate": number,
   "name": "string",
   "parentSlotTypeSignature": "string",
   "slotTypeConfigurations": [
      {
         "regexConfiguration": {
            "pattern": "string"
         }
      }
   ],
   "valueSelectionStrategy": "string",
   "version": "string"
```

```
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respon HTTP 201.

Layanan mengembalikan data berikut dalam format JSON.

checksum

Checksum dari `$LATEST` versi jenis slot.

Jenis: String

createdDate

Tanggal jenis slot dibuat.

Tipe: Timestamp

description

Deskripsi jenis slot.

Jenis: String

Batasan Panjang: Panjang minimum 0. Panjang maksimum 200.

enumerationValues

Daftar `EnumerationValue` objek yang mendefinisikan nilai-nilai yang dapat diambil oleh jenis slot.

Tipe: Array objek EnumerationValue

Anggota Array: Jumlah minimum 0 item. Jumlah maksimum 10000 item.

lastUpdatedDate

Tanggal bahwa jenis slot diperbarui. Saat Anda membuat sumber daya, tanggal pembuatan dan tanggal pembaruan terakhir adalah sama.

Tipe: Timestamp

name

Nama jenis slot.

Jenis: String

Panjang Batasan: Panjang minimum 1. Panjang maksimum 100.

Pola: `^([A-Za-z]_?)+$`

parentSlotTypeSignature

Tipe slot bawaan menggunakan induk dari jenis slot.

Jenis: String

Panjang Batasan: Panjang minimum 1. Panjang maksimum 100.

Pola: `^((AMAZON\.)_?|[A-Za-z]_?)+`

slotTypeConfigurations

Informasi konfigurasi yang memperluas tipe slot bawaan induk.

Tipe: Array objek SlotTypeConfiguration

Anggota Array: Jumlah minimum 0 item. Jumlah maksimum 10 item.

valueSelectionStrategy

Strategi yang digunakan Amazon Lex untuk menentukan nilai slot. Untuk informasi selengkapnya, lihat PutSlotType.

Tipe: String

Nilai yang Valid: `ORIGINAL_VALUE | TOP_RESOLUTION`

version

Versi yang ditetapkan untuk versi jenis slot baru.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum adalah 64.

Pola: `\$LATEST|[0-9]+`

Kesalahan

BadRequestException

Permintaan tidak terbentuk dengan baik. Misalnya, nilai tidak valid atau bidang wajib hilang. Periksa nilai bidang, dan coba lagi.

Kode Status HTTP: 400

ConflictException

Ada konflik memproses permintaan. Coba permintaan Anda lagi.

Kode Status HTTP: 409

InternalFailureException

Terjadi kesalahan internal Amazon Lex. Coba permintaan Anda lagi.

Kode Status HTTP: 500

LimitExceededException

Permintaan melebihi batas. Coba permintaan Anda lagi.

Kode Status HTTP: 429

NotFoundException

Sumber daya yang ditentukan dalam permintaan tidak ditemukan. Periksa sumber daya dan coba lagi.

Kode Status HTTP: 404

PreconditionFailedException

Checksum sumber daya yang Anda coba ubah tidak cocok dengan checksum dalam permintaan. Periksa checksum sumber daya dan coba lagi.

Kode Status HTTP: 412

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

DeleteBot

Layanan: Amazon Lex Model Building Service

Menghapus semua versi bot, termasuk `$LATEST` versinya. Untuk menghapus versi bot tertentu, gunakan DeleteBotVersion operasi. `DeleteBot`Operasi tidak segera menghapus skema bot. Sebaliknya, itu ditandai untuk dihapus dan dihapus nanti.

Amazon Lex menyimpan ucapan tanpa batas untuk meningkatkan kemampuan bot Anda untuk merespons input pengguna. Ucapan ini tidak dihapus ketika bot dihapus. Untuk menghapus ucapan, gunakan operasi. DeleteUtterances

Jika bot memiliki alias, Anda tidak dapat menghapusnya. Sebagai gantinya, `DeleteBot` operasi mengembalikan `ResourceInUseException` pengecualian yang menyertakan referensi ke alias yang mengacu pada bot. Untuk menghapus referensi ke bot, hapus alias. Jika Anda mendapatkan pengecualian yang sama lagi, hapus alias rujukan hingga `DeleteBot` operasi berhasil.

Operasi ini memerlukan izin utuk tindakan `lex:DeleteBot`.

Minta Sintaks

```
DELETE /bots/name HTTP/1.1
```

Parameter Permintaan URI

Permintaan menggunakan parameter URI berikut.

name

Nama bot. Namanya peka huruf besar/kecil.

Kendala Panjang: Panjang minimum 2. Panjang maksimum 50.

Pola: `^([A-Za-z]_?)+$`

Wajib: Ya

Isi Permintaan

Permintaan tidak memiliki isi permintaan.

Sintaks Respons

```
HTTP/1.1 204
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 204 dengan isi HTTP kosong.

Kesalahan

BadRequestException

Permintaan tidak terbentuk dengan baik. Misalnya, nilai tidak valid atau bidang wajib hilang. Periksa nilai bidang, dan coba lagi.

Kode Status HTTP: 400

ConflictException

Ada konflik yang memproses permintaan tersebut. Coba permintaan Anda lagi.

Kode Status HTTP: 409

InternalFailureException

Terjadi kesalahan internal Amazon Lex. Coba permintaan Anda lagi.

Kode Status HTTP: 500

LimitExceededException

Permintaan melebihi batas. Coba permintaan Anda lagi.

Kode Status HTTP: 429

NotFoundException

Sumber daya yang ditentukan dalam permintaan tidak ditemukan. Periksa sumber daya dan coba lagi.

Kode Status HTTP: 404

ResourceInUseException

Sumber daya yang Anda coba hapus dirujuk oleh sumber daya lain. Gunakan informasi ini untuk menghapus referensi ke sumber daya yang Anda coba hapus.

Badan pengecualian berisi objek JSON yang menjelaskan sumber daya.

```
{ "resourceType": BOT | BOTALIAS | BOTCHANNEL | INTENT,

"resourceReference": {

"name": string, "version": string } }
```

Kode Status HTTP: 400

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

DeleteBotAlias

Layanan: Amazon Lex Model Building Service

Menghapus alias untuk bot yang ditentukan.

Anda tidak dapat menghapus alias yang digunakan dalam asosiasi antara bot dan saluran pesan. Jika alias digunakan dalam asosiasi saluran, `DeleteBot` operasi mengembalikan `ResourceInUseException` pengecualian yang menyertakan referensi ke asosiasi saluran yang merujuk ke bot. Anda dapat menghapus referensi ke alias dengan menghapus asosiasi saluran. Jika Anda mendapatkan pengecualian yang sama lagi, hapus asosiasi rujukan hingga `DeleteBotAlias` operasi berhasil.

Minta Sintaks

```
DELETE /bots/botName/aliases/name HTTP/1.1
```

Parameter Permintaan URI

Permintaan menggunakan parameter URI berikut.

botName

Nama bot yang ditunjuk alias.

Kendala Panjang: Panjang minimum 2. Panjang maksimum 50.

Pola: `^([A-Za-z]_?)+$`

Diperlukan: Ya

name

Nama alias yang akan dihapus. Namanya peka huruf besar/kecil.

Batasan Panjang: Panjang minimum 1. Panjang maksimum 100.

Pola: `^([A-Za-z]_?)+$`

Wajib: Ya

Isi Permintaan

Permintaan tidak memiliki isi permintaan.

Sintaks Respons

```
HTTP/1.1 204
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 204 dengan isi HTTP kosong.

Kesalahan

BadRequestException

Permintaan tidak terbentuk dengan baik. Misalnya, nilai tidak valid atau bidang wajib hilang. Periksa nilai bidang, dan coba lagi.

Kode Status HTTP: 400

ConflictException

Ada konflik yang memproses permintaan tersebut. Coba permintaan Anda lagi.

Kode Status HTTP: 409

InternalFailureException

Terjadi kesalahan internal Amazon Lex. Coba permintaan Anda lagi.

Kode Status HTTP: 500

LimitExceededException

Permintaan melebihi batas. Coba permintaan Anda lagi.

Kode Status HTTP: 429

NotFoundException

Sumber daya yang ditentukan dalam permintaan tidak ditemukan. Periksa sumber daya dan coba lagi.

Kode Status HTTP: 404

ResourceInUseException

Sumber daya yang Anda coba hapus dirujuk oleh sumber daya lain. Gunakan informasi ini untuk menghapus referensi ke sumber daya yang Anda coba hapus.

Badan pengecualian berisi objek JSON yang menjelaskan sumber daya.

```
{ "resourceType": BOT | BOTALIAS | BOTCHANNEL | INTENT,

"resourceReference": {

"name": string, "version": string } }
```

Kode Status HTTP: 400

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

DeleteBotChannelAssociation

Layanan: Amazon Lex Model Building Service

Menghapus hubungan antara bot Amazon Lex dan platform perpesanan.

Operasi ini memerlukan izin untuk tindakan `lex:DeleteBotChannelAssociation`.

Minta Sintaks

```
DELETE /bots/botName/aliases/aliasName/channels/name HTTP/1.1
```

Parameter Permintaan URI

Permintaan menggunakan parameter URI berikut.

aliasName

Alias yang menunjuk ke versi spesifik bot Amazon Lex tempat asosiasi ini dibuat.

Batasan Panjang: Panjang minimum 1. Panjang maksimum 100.

Pola: `^([A-Za-z]_?)+$`

Diperlukan: Ya

botName

Nama bot Amazon Lex.

Kendala Panjang: Panjang minimum 2. Panjang maksimum 50.

Pola: `^([A-Za-z]_?)+$`

Diperlukan: Ya

name

Nama pengaitan. Namanya peka huruf besar/kecil.

Batasan Panjang: Panjang minimum 1. Panjang maksimum 100.

Pola: `^([A-Za-z]_?)+$`

Wajib: Ya

Isi Permintaan

Permintaan tidak memiliki isi permintaan.

Sintaks Respons

```
HTTP/1.1 204
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 204 dengan isi HTTP kosong.

Kesalahan

BadRequestException

Permintaan tidak terbentuk dengan baik. Misalnya, nilai tidak valid atau bidang wajib hilang. Periksa nilai bidang, dan coba lagi.

Kode Status HTTP: 400

ConflictException

Ada konflik yang memproses permintaan tersebut. Coba permintaan Anda lagi.

Kode Status HTTP: 409

InternalFailureException

Terjadi kesalahan internal Amazon Lex. Coba permintaan Anda lagi.

Kode Status HTTP: 500

LimitExceededException

Permintaan melebihi batas. Coba permintaan Anda lagi.

Kode Status HTTP: 429

NotFoundException

Sumber daya yang ditentukan dalam permintaan tidak ditemukan. Periksa sumber daya dan coba lagi.

Kode Status HTTP: 404

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

DeleteBotVersion

Layanan: Amazon Lex Model Building Service

Menghapus versi bot tertentu. Untuk menghapus semua versi bot, gunakan DeleteBot operasi.

Operasi ini memerlukan izin utuk tindakan `lex:DeleteBotVersion`.

Minta Sintaks

```
DELETE /bots/name/versions/version HTTP/1.1
```

Parameter Permintaan URI

Permintaan menggunakan parameter URI berikut.

name

Nama bot.

Kendala Panjang: Panjang minimum 2. Panjang maksimum 50.

Pola: `^([A-Za-z]_?)+$`

Diperlukan: Ya

version

Versi bot yang akan dihapus. Anda tidak dapat menghapus `$LATEST` versi bot. Untuk menghapus `$LATEST` versi, gunakan DeleteBot operasi.

Batasan Panjang: Panjang minimum 1. Panjang maksimum adalah 64.

Pola: `[0-9]+`

Wajib: Ya

Isi Permintaan

Permintaan tidak memiliki isi permintaan.

Sintaks Respons

```
HTTP/1.1 204
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 204 dengan isi HTTP kosong.

Kesalahan

BadRequestException

Permintaan tidak terbentuk dengan baik. Misalnya, nilai tidak valid atau bidang wajib hilang. Periksa nilai bidang, dan coba lagi.

Kode Status HTTP: 400

ConflictException

Ada konflik memproses permintaan. Coba permintaan Anda lagi.

Kode Status HTTP: 409

InternalFailureException

Terjadi kesalahan internal Amazon Lex. Coba permintaan Anda lagi.

Kode Status HTTP: 500

LimitExceededException

Permintaan melebihi batas. Coba permintaan Anda lagi.

Kode Status HTTP: 429

NotFoundException

Sumber daya yang ditentukan dalam permintaan tidak ditemukan. Periksa sumber daya dan coba lagi.

Kode Status HTTP: 404

ResourceInUseException

Sumber daya yang Anda coba hapus dirujuk oleh sumber daya lain. Gunakan informasi ini untuk menghapus referensi ke sumber daya yang Anda coba hapus.

Badan pengecualian berisi objek JSON yang menjelaskan sumber daya.

```
{ "resourceType": BOT | BOTALIAS | BOTCHANNEL | INTENT,
```

```
"resourceReference": {

"name": string, "version": string } }
```

Kode Status HTTP: 400

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

DeleteIntent

Layanan: Amazon Lex Model Building Service

Menghapus semua versi intent, termasuk versi. `$LATEST` Untuk menghapus versi tertentu dari intent, gunakan DeleteIntentVersion operasi.

Anda dapat menghapus versi intent hanya jika tidak direferensikan. Untuk menghapus maksud yang dirujuk dalam satu atau beberapa bot (lihatAmazon Lex: Cara Kerjanya), Anda harus menghapus referensi tersebut terlebih dahulu.

> **Note**
>
> Jika Anda mendapatkan `ResourceInUseException` pengecualian, ini memberikan contoh referensi yang menunjukkan di mana maksud direferensikan. Untuk menghapus referensi ke intent, perbarui bot atau hapus. Jika Anda mendapatkan pengecualian yang sama saat mencoba menghapus intent lagi, ulangi sampai intent tidak memiliki referensi dan panggilan ke berhasil`DeleteIntent`.

Operasi ini memerlukan izin untuk tindakan `lex:DeleteIntent`.

Minta Sintaks

```
DELETE /intents/name HTTP/1.1
```

Parameter Permintaan URI

Permintaan menggunakan parameter URI berikut.

name

Nama niat. Namanya peka huruf besar/kecil.

Batasan Panjang: Panjang minimum 1. Panjang maksimum 100.

Pola: `^([A-Za-z]_?)+$`

Wajib: Ya

Isi Permintaan

Permintaan tidak memiliki isi permintaan.

Sintaks Respons

```
HTTP/1.1 204
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 204 dengan isi HTTP kosong.

Kesalahan

BadRequestException

Permintaan tidak terbentuk dengan baik. Misalnya, nilai tidak valid atau bidang wajib hilang. Periksa nilai bidang, dan coba lagi.

Kode Status HTTP: 400

ConflictException

Ada konflik memproses permintaan. Coba permintaan Anda lagi.

Kode Status HTTP: 409

InternalFailureException

Terjadi kesalahan internal Amazon Lex. Coba permintaan Anda lagi.

Kode Status HTTP: 500

LimitExceededException

Permintaan melebihi batas. Coba permintaan Anda lagi.

Kode Status HTTP: 429

NotFoundException

Sumber daya yang ditentukan dalam permintaan tidak ditemukan. Periksa sumber daya dan coba lagi.

Kode Status HTTP: 404

ResourceInUseException

Sumber daya yang Anda coba hapus dirujuk oleh sumber daya lain. Gunakan informasi ini untuk menghapus referensi ke sumber daya yang Anda coba hapus.

Badan pengecualian berisi objek JSON yang menjelaskan sumber daya.

```
{ "resourceType": BOT | BOTALIAS | BOTCHANNEL | INTENT,

"resourceReference": {

"name": string, "version": string } }
```

Kode Status HTTP: 400

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

DeleteIntentVersion

Layanan: Amazon Lex Model Building Service

Menghapus versi tertentu dari intent. Untuk menghapus semua versi intent, gunakan DeleteIntent operasi.

Operasi ini memerlukan izin utuk tindakan `lex:DeleteIntentVersion`.

Minta Sintaks

```
DELETE /intents/name/versions/version HTTP/1.1
```

Parameter Permintaan URI

Permintaan menggunakan parameter URI berikut.

name

Nama niat.

Batasan Panjang: Panjang minimum 1. Panjang maksimum 100.

Pola: `^([A-Za-z]_?)+$`

Diperlukan: Ya

version

Versi maksud untuk menghapus. Anda tidak dapat menghapus `$LATEST` versi intent. Untuk menghapus `$LATEST` versi, gunakan DeleteIntent operasi.

Batasan Panjang: Panjang minimum 1. Panjang maksimum adalah 64.

Pola: `[0-9]+`

Wajib: Ya

Isi Permintaan

Permintaan tidak memiliki isi permintaan.

Sintaks Respons

```
HTTP/1.1 204
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 204 dengan isi HTTP kosong.

Kesalahan

BadRequestException

Permintaan tidak terbentuk dengan baik. Misalnya, nilai tidak valid atau bidang wajib hilang. Periksa nilai bidang, dan coba lagi.

Kode Status HTTP: 400

ConflictException

Ada konflik memproses permintaan. Coba permintaan Anda lagi.

Kode Status HTTP: 409

InternalFailureException

Terjadi kesalahan internal Amazon Lex. Coba permintaan Anda lagi.

Kode Status HTTP: 500

LimitExceededException

Permintaan melebihi batas. Coba permintaan Anda lagi.

Kode Status HTTP: 429

NotFoundException

Sumber daya yang ditentukan dalam permintaan tidak ditemukan. Periksa sumber daya dan coba lagi.

Kode Status HTTP: 404

ResourceInUseException

Sumber daya yang Anda coba hapus dirujuk oleh sumber daya lain. Gunakan informasi ini untuk menghapus referensi ke sumber daya yang Anda coba hapus.

Badan pengecualian berisi objek JSON yang menjelaskan sumber daya.

```
{ "resourceType": BOT | BOTALIAS | BOTCHANNEL | INTENT,

"resourceReference": {

"name": string, "version": string } }
```

Kode Status HTTP: 400

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

DeleteSlotType

Layanan: Amazon Lex Model Building Service

Menghapus semua versi jenis slot, termasuk `$LATEST` versi. Untuk menghapus versi tertentu dari jenis slot, gunakan DeleteSlotTypeVersion operasi.

Anda dapat menghapus versi jenis slot hanya jika tidak direferensikan. Untuk menghapus jenis slot yang dirujuk dalam satu atau lebih maksud, Anda harus menghapus referensi tersebut terlebih dahulu.

> **Note**
>
> Jika Anda mendapatkan `ResourceInUseException` pengecualian, pengecualian memberikan contoh referensi yang menunjukkan maksud di mana jenis slot direferensikan. Untuk menghapus referensi ke jenis slot, perbarui maksud atau hapus. Jika Anda mendapatkan pengecualian yang sama ketika Anda mencoba untuk menghapus jenis slot lagi, ulangi sampai jenis slot tidak memiliki referensi dan `DeleteSlotType` panggilan berhasil.

Operasi ini memerlukan izin untuk tindakan `lex:DeleteSlotType`.

Minta Sintaks

```
DELETE /slottypes/name HTTP/1.1
```

Parameter Permintaan URI

Permintaan menggunakan parameter URI berikut.

name

Nama jenis slot. Namanya peka huruf besar/kecil.

Batasan Panjang: Panjang minimum 1. Panjang maksimum 100.

Pola: `^([A-Za-z]_?)+$`

Wajib: Ya

Isi Permintaan

Permintaan tidak memiliki isi permintaan.

Sintaks Respons

```
HTTP/1.1 204
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 204 dengan isi HTTP kosong.

Kesalahan

BadRequestException

Permintaan tidak terbentuk dengan baik. Misalnya, nilai tidak valid atau bidang wajib hilang. Periksa nilai bidang, dan coba lagi.

Kode Status HTTP: 400

ConflictException

Ada konflik yang memproses permintaan tersebut. Coba permintaan Anda lagi.

Kode Status HTTP: 409

InternalFailureException

Terjadi kesalahan internal Amazon Lex. Coba permintaan Anda lagi.

Kode Status HTTP: 500

LimitExceededException

Permintaan melebihi batas. Coba permintaan Anda lagi.

Kode Status HTTP: 429

NotFoundException

Sumber daya yang ditentukan dalam permintaan tidak ditemukan. Periksa sumber daya dan coba lagi.

Kode Status HTTP: 404

ResourceInUseException

Sumber daya yang Anda coba hapus dirujuk oleh sumber daya lain. Gunakan informasi ini untuk menghapus referensi ke sumber daya yang Anda coba hapus.

Badan pengecualian berisi objek JSON yang menjelaskan sumber daya.

```
{ "resourceType": BOT | BOTALIAS | BOTCHANNEL | INTENT,

"resourceReference": {

"name": string, "version": string } }
```

Kode Status HTTP: 400

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

DeleteSlotTypeVersion

Layanan: Amazon Lex Model Building Service

Menghapus versi tertentu dari jenis slot. Untuk menghapus semua versi jenis slot, gunakan DeleteSlotType operasi.

Operasi ini memerlukan izin utuk tindakan `lex:DeleteSlotTypeVersion`.

Minta Sintaks

```
DELETE /slottypes/name/version/version HTTP/1.1
```

Parameter Permintaan URI

Permintaan menggunakan parameter URI berikut.

name

Nama jenis slot.

Batasan Panjang: Panjang minimum 1. Panjang maksimum 100.

Pola: `^([A-Za-z]_?)+$`

Diperlukan: Ya

version

Versi jenis slot yang akan dihapus. Anda tidak dapat menghapus `$LATEST` versi jenis slot. Untuk menghapus `$LATEST` versi, gunakan DeleteSlotType operasi.

Batasan Panjang: Panjang minimum 1. Panjang maksimum adalah 64.

Pola: `[0-9]+`

Wajib: Ya

Isi Permintaan

Permintaan tidak memiliki isi permintaan.

Sintaks Respons

```
HTTP/1.1 204
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 204 dengan isi HTTP kosong.

Kesalahan

BadRequestException

Permintaan tidak terbentuk dengan baik. Misalnya, nilai tidak valid atau bidang wajib hilang. Periksa nilai bidang, dan coba lagi.

Kode Status HTTP: 400

ConflictException

Ada konflik memproses permintaan. Coba permintaan Anda lagi.

Kode Status HTTP: 409

InternalFailureException

Terjadi kesalahan internal Amazon Lex. Coba permintaan Anda lagi.

Kode Status HTTP: 500

LimitExceededException

Permintaan melebihi batas. Coba permintaan Anda lagi.

Kode Status HTTP: 429

NotFoundException

Sumber daya yang ditentukan dalam permintaan tidak ditemukan. Periksa sumber daya dan coba lagi.

Kode Status HTTP: 404

ResourceInUseException

Sumber daya yang Anda coba hapus dirujuk oleh sumber daya lain. Gunakan informasi ini untuk menghapus referensi ke sumber daya yang Anda coba hapus.

Badan pengecualian berisi objek JSON yang menjelaskan sumber daya.

```
{ "resourceType": BOT | BOTALIAS | BOTCHANNEL | INTENT,

"resourceReference": {

"name": string, "version": string } }
```

Kode Status HTTP: 400

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

DeleteUtterances

Layanan: Amazon Lex Model Building Service

Menghapus ucapan yang disimpan.

Amazon Lex menyimpan ucapan yang dikirim pengguna ke bot Anda. Ucapan disimpan selama 15 hari untuk digunakan dengan GetUtterancesView operasi, dan kemudian disimpan tanpa batas waktu untuk digunakan dalam meningkatkan kemampuan bot Anda untuk menanggapi input pengguna.

Gunakan `DeleteUtterances` operasi untuk menghapus ucapan tersimpan secara manual untuk pengguna tertentu. Saat Anda menggunakan `DeleteUtterances` operasi, ucapan yang disimpan untuk meningkatkan kemampuan bot Anda untuk merespons input pengguna segera dihapus. Ucapan yang disimpan untuk digunakan dengan `GetUtterancesView` operasi dihapus setelah 15 hari.

Operasi ini memerlukan izin utuk tindakan `lex:DeleteUtterances`.

Minta Sintaks

```
DELETE /bots/botName/utterances/userId HTTP/1.1
```

Parameter Permintaan URI

Permintaan menggunakan parameter URI berikut.

botName

Nama bot yang menyimpan ucapan.

Kendala Panjang: Panjang minimum 2. Panjang maksimum 50.

Pola: `^([A-Za-z]_?)+$`

Diperlukan: Ya

userId

Pengidentifikasi unik untuk pengguna yang membuat ucapan. Ini adalah ID pengguna yang dikirim dalam permintaan PostContentatau PostTextoperasi yang berisi ucapan.

Kendala Panjang: Panjang minimum 2. Panjang maksimum 100.

Diperlukan: Ya

Isi Permintaan

Permintaan tidak memiliki isi permintaan.

Sintaks Respons

```
HTTP/1.1 204
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 204 dengan isi HTTP kosong.

Kesalahan

BadRequestException

Permintaan tidak terbentuk dengan baik. Misalnya, nilai tidak valid atau bidang wajib hilang. Periksa nilai bidang, dan coba lagi.

Kode Status HTTP: 400

InternalFailureException

Terjadi kesalahan internal Amazon Lex. Coba permintaan Anda lagi.

Kode Status HTTP: 500

LimitExceededException

Permintaan melebihi batas. Coba permintaan Anda lagi.

Kode Status HTTP: 429

NotFoundException

Sumber daya yang ditentukan dalam permintaan tidak ditemukan. Periksa sumber daya dan coba lagi.

Kode Status HTTP: 404

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

GetBot

Layanan: Amazon Lex Model Building Service

Mengembalikan informasi metadata untuk bot tertentu. Anda harus memberikan nama bot dan versi bot atau alias.

Operasi ini memerlukan izin utuk tindakan `lex:GetBot`.

Minta Sintaks

```
GET /bots/name/versions/versionoralias HTTP/1.1
```

Parameter Permintaan URI

Permintaan menggunakan parameter URI berikut.

name

Nama bot. Namanya peka huruf besar/kecil.

Kendala Panjang: Panjang minimum 2. Panjang maksimum 50.

Pola: `^([A-Za-z]_?)+$`

Diperlukan: Ya

versionoralias

Versi atau alias bot.

Diperlukan: Ya

Isi Permintaan

Permintaan tidak memiliki isi permintaan.

Sintaks Respons

```
HTTP/1.1 200
Content-type: application/json

{
   "abortStatement": {
      "messages": [
```

```
            {
               "content": "string",
               "contentType": "string",
               "groupNumber": number
            }
         ],
         "responseCard": "string"
      },
      "checksum": "string",
      "childDirected": boolean,
      "clarificationPrompt": {
         "maxAttempts": number,
         "messages": [
            {
               "content": "string",
               "contentType": "string",
               "groupNumber": number
            }
         ],
         "responseCard": "string"
      },
      "createdDate": number,
      "description": "string",
      "detectSentiment": boolean,
      "enableModelImprovements": boolean,
      "failureReason": "string",
      "idleSessionTTLInSeconds": number,
      "intents": [
         {
            "intentName": "string",
            "intentVersion": "string"
         }
      ],
      "lastUpdatedDate": number,
      "locale": "string",
      "name": "string",
      "nluIntentConfidenceThreshold": number,
      "status": "string",
      "version": "string",
      "voiceId": "string"
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Layanan mengembalikan data berikut dalam format JSON.

abortStatement

Pesan yang dikembalikan Amazon Lex ketika pengguna memilih untuk mengakhiri percakapan tanpa menyelesaikannya. Untuk informasi selengkapnya, lihat PutBot.

Tipe: Objek Statement

checksum

Checksum bot yang digunakan untuk mengidentifikasi revisi spesifik dari versi bot. `$LATEST`

Jenis: String

childDirected

Untuk setiap bot Amazon Lex yang dibuat dengan Layanan Pembuatan Model Amazon Lex, Anda harus menentukan apakah penggunaan Amazon Lex Anda terkait dengan situs web, program, atau aplikasi lain yang diarahkan atau ditargetkan, secara keseluruhan atau sebagian, untuk anak-anak di bawah usia 13 tahun dan tunduk pada Undang-Undang Perlindungan Privasi Online Anak (COPPA) dengan menentukan `true` atau `false` di lapangan. `childDirected` Dengan menentukan **`true`** di **`childDirected`** lapangan, Anda mengonfirmasi bahwa penggunaan Amazon Lex Anda terkait dengan situs web, program, atau aplikasi lain yang diarahkan atau ditargetkan, secara keseluruhan atau sebagian, untuk anak-anak di bawah usia 13 tahun dan tunduk pada COPPA. Dengan menentukan `false` di `childDirected` lapangan, Anda mengonfirmasi bahwa penggunaan Amazon Lex Anda tidak terkait dengan situs web, program, atau aplikasi lain yang diarahkan atau ditargetkan, secara keseluruhan atau sebagian, untuk anak-anak di bawah usia 13 tahun dan tunduk pada COPPA. Anda tidak boleh menentukan nilai default untuk `childDirected` bidang yang tidak secara akurat mencerminkan apakah penggunaan Amazon Lex Anda terkait dengan situs web, program, atau aplikasi lain yang diarahkan atau ditargetkan, secara keseluruhan atau sebagian, untuk anak-anak di bawah usia 13 tahun dan tunduk pada COPPA.

Jika penggunaan Amazon Lex Anda terkait dengan situs web, program, atau aplikasi lain yang diarahkan secara keseluruhan atau sebagian, untuk anak-anak di bawah usia 13 tahun, Anda harus mendapatkan persetujuan orang tua yang dapat diverifikasi berdasarkan COPPA. Untuk

informasi mengenai penggunaan Amazon Lex sehubungan dengan situs web, program, atau aplikasi lain yang diarahkan atau ditargetkan, secara keseluruhan atau sebagian, untuk anak-anak di bawah usia 13 tahun, lihat FAQ Amazon Lex.

Jenis: Boolean

clarificationPrompt

Pesan Amazon Lex menggunakan ketika tidak memahami permintaan pengguna. Untuk informasi selengkapnya, lihat PutBot.

Tipe: Objek Prompt

createdDate

Tanggal bot itu dibuat.

Tipe: Timestamp

description

Deskripsi bot.

Jenis: String

Batasan Panjang: Panjang minimum 0. Panjang maksimum 200.

detectSentiment

Menunjukkan apakah ucapan pengguna harus dikirim ke Amazon Comprehend untuk analisis sentimen.

Jenis: Boolean

enableModelImprovements

Menunjukkan apakah bot menggunakan peningkatan akurasi. `true`menunjukkan bahwa bot menggunakan perbaikan, jika tidak,`false`.

Jenis: Boolean

failureReason

Jika `status` ya`FAILED`, Amazon Lex menjelaskan mengapa gagal membangun bot.

Jenis: String

idleSessionTTLInSeconds

Waktu maksimum dalam hitungan detik Amazon Lex menyimpan data yang dikumpulkan dalam percakapan. Untuk informasi selengkapnya, lihat PutBot.

Jenis: Integer

Rentang yang Valid: Nilai minimum 60. Nilai maksimum 86400.

intents

Susunan objek `intent`. Untuk informasi selengkapnya, lihat PutBot.

Tipe: Array objek Intent

lastUpdatedDate

Tanggal bot diperbarui. Saat Anda membuat sumber daya, tanggal pembuatan dan tanggal terakhir diperbarui adalah sama.

Tipe: Timestamp

locale

Target lokal untuk bot.

Jenis: String

Nilai yang Valid: `de-DE | en-AU | en-GB | en-IN | en-US | es-419 | es-ES | es-US | fr-FR | fr-CA | it-IT | ja-JP | ko-KR`

name

Nama bot.

Jenis: String

Kendala Panjang: Panjang minimum 2. Panjang maksimum 50.

Pola: `^([A-Za-z]_?)+$`

nluIntentConfidenceThreshold

Skor yang menentukan di mana Amazon Lex menyisipkan `AMAZON.FallbackIntentAMAZON.KendraSearchIntent`,, atau keduanya

saat mengembalikan maksud alternatif dalam respons PostContentatau PostText. AMAZON.FallbackIntentdimasukkan jika skor kepercayaan untuk semua maksud di bawah nilai ini. AMAZON.KendraSearchIntenthanya dimasukkan jika dikonfigurasi untuk bot.

Tipe: Ganda

Rentang yang Valid: Nilai minimum 0. Nilai maksimum 1.

status

Status bot.

Ketika statusnya adalah BUILDING Amazon Lex sedang membangun bot untuk pengujian dan penggunaan.

Jika status botnyaREADY_BASIC_TESTING, Anda dapat menguji bot menggunakan ucapan persis yang ditentukan dalam maksud bot. Ketika bot siap untuk pengujian penuh atau dijalankan, statusnya adalahREADY.

Jika ada masalah dengan membangun bot, statusnya FAILED dan failureReason bidang menjelaskan mengapa bot tidak dibangun.

Jika bot disimpan tetapi tidak dibangun, statusnya adalahNOT_BUILT.

Jenis: String

Nilai yang Valid: BUILDING | READY | READY_BASIC_TESTING | FAILED | NOT_BUILT

version

Versi bot. Untuk bot baru, versinya selalu$LATEST.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum adalah 64.

Pola: \$LATEST|[0-9]+

voiceId

ID suara Amazon Polly yang digunakan Amazon Lex untuk interaksi suara dengan pengguna. Untuk informasi selengkapnya, lihat PutBot.

Tipe: String

Kesalahan

BadRequestException

Permintaan tidak terbentuk dengan baik. Misalnya, nilai tidak valid atau bidang wajib hilang. Periksa nilai bidang, dan coba lagi.

Kode Status HTTP: 400

InternalFailureException

Terjadi kesalahan internal Amazon Lex. Coba permintaan Anda lagi.

Kode Status HTTP: 500

LimitExceededException

Permintaan melebihi batas. Coba permintaan Anda lagi.

Kode Status HTTP: 429

NotFoundException

Sumber daya yang ditentukan dalam permintaan tidak ditemukan. Periksa sumber daya dan coba lagi.

Kode Status HTTP: 404

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python

- AWS SDK for Ruby V3

GetBotAlias

Layanan: Amazon Lex Model Building Service

Mengembalikan informasi tentang alias bot Amazon Lex. Untuk informasi selengkapnya tentang alias, lihatPembuatan Versi dan Alias.

Operasi ini memerlukan izin utuk tindakan `lex:GetBotAlias`.

Minta Sintaks

```
GET /bots/botName/aliases/name HTTP/1.1
```

Parameter Permintaan URI

Permintaan menggunakan parameter URI berikut.

botName

Nama bot.

Kendala Panjang: Panjang minimum 2. Panjang maksimum 50.

Pola: `^([A-Za-z]_?)+$`

Diperlukan: Ya

name

Nama bot alias. Namanya peka huruf besar/kecil.

Batasan Panjang: Panjang minimum 1. Panjang maksimum 100.

Pola: `^([A-Za-z]_?)+$`

Wajib: Ya

Isi Permintaan

Permintaan tidak memiliki isi permintaan.

Sintaks Respons

```
HTTP/1.1 200
```

```
Content-type: application/json

{
   "botName": "string",
   "botVersion": "string",
   "checksum": "string",
   "conversationLogs": {
      "iamRoleArn": "string",
      "logSettings": [
         {
            "destination": "string",
            "kmsKeyArn": "string",
            "logType": "string",
            "resourceArn": "string",
            "resourcePrefix": "string"
         }
      ]
   },
   "createdDate": number,
   "description": "string",
   "lastUpdatedDate": number,
   "name": "string"
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Layanan mengembalikan data berikut dalam format JSON.

botName

Nama bot yang ditunjuk alias.

Jenis: String

Kendala Panjang: Panjang minimum 2. Panjang maksimum 50.

Pola: `^([A-Za-z]_?)+$`

botVersion

Versi bot yang ditunjuk alias.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum adalah 64.

Pola: `\$LATEST|[0-9]+`

checksum

Checksum dari alias bot.

Jenis: String

conversationLogs

Pengaturan yang menentukan bagaimana Amazon Lex menggunakan log percakapan untuk alias.

Tipe: Objek ConversationLogsResponse

createdDate

Tanggal alias bot dibuat.

Tipe: Timestamp

description

Deskripsi alias bot.

Jenis: String

Batasan Panjang: Panjang minimum 0. Panjang maksimum 200.

lastUpdatedDate

Tanggal alias bot diperbarui. Saat Anda membuat sumber daya, tanggal pembuatan dan tanggal pembaruan terakhir adalah sama.

Tipe: Timestamp

name

Nama bot alias.

Jenis: String

Panjang Batasan: Panjang minimum 1. Panjang maksimum 100.

Pola: `^([A-Za-z]_?)+$`

Kesalahan

BadRequestException

Permintaan tidak terbentuk dengan baik. Misalnya, nilai tidak valid atau bidang wajib hilang. Periksa nilai bidang, dan coba lagi.

Kode Status HTTP: 400

InternalFailureException

Terjadi kesalahan internal Amazon Lex. Coba permintaan Anda lagi.

Kode Status HTTP: 500

LimitExceededException

Permintaan melebihi batas. Coba permintaan Anda lagi.

Kode Status HTTP: 429

NotFoundException

Sumber daya yang ditentukan dalam permintaan tidak ditemukan. Periksa sumber daya dan coba lagi.

Kode Status HTTP: 404

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python

- AWS SDK for Ruby V3

GetBotAliases

Layanan: Amazon Lex Model Building Service

Mengembalikan daftar alias untuk bot Amazon Lex tertentu.

Operasi ini memerlukan izin utuk tindakan `lex:GetBotAliases`.

Minta Sintaks

```
GET /bots/botName/aliases/?
maxResults=maxResults&nameContains=nameContains&nextToken=nextToken HTTP/1.1
```

Parameter Permintaan URI

Permintaan menggunakan parameter URI berikut.

botName

Nama bot.

Kendala Panjang: Panjang minimum 2. Panjang maksimum 50.

Pola: `^([A-Za-z]_?)+$`

Diperlukan: Ya

maxResults

Jumlah maksimum alias untuk kembali dalam respons. Defaultnya adalah 50.

Rentang yang Valid: Nilai minimum 1. Nilai maksimum 50.

nameContains

Substring untuk mencocokkan nama alias bot. Alias akan dikembalikan jika ada bagian dari namanya yang cocok dengan substring. Misalnya, “xyz” cocok dengan “xyzabc” dan “abcxyz.”

Batasan Panjang: Panjang minimum 1. Panjang maksimum 100.

Pola: `^([A-Za-z]_?)+$`

nextToken

Token pagination untuk mengambil halaman berikutnya dari alias. Jika respons terhadap panggilan ini terpotong, Amazon Lex mengembalikan token pagination dalam respons. Untuk mengambil halaman alias berikutnya, tentukan token pagination di permintaan berikutnya.

Isi Permintaan

Permintaan tidak memiliki isi permintaan.

Sintaks Respons

```
HTTP/1.1 200
Content-type: application/json

{
   "BotAliases": [
      {
         "botName": "string",
         "botVersion": "string",
         "checksum": "string",
         "conversationLogs": {
            "iamRoleArn": "string",
            "logSettings": [
               {
                  "destination": "string",
                  "kmsKeyArn": "string",
                  "logType": "string",
                  "resourceArn": "string",
                  "resourcePrefix": "string"
               }
            ]
         },
         "createdDate": number,
         "description": "string",
         "lastUpdatedDate": number,
         "name": "string"
      }
   ],
   "nextToken": "string"
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Layanan mengembalikan data berikut dalam format JSON.

BotAliases

Sebuah array `BotAliasMetadata` objek, masing-masing menggambarkan alias bot.

Tipe: Array objek BotAliasMetadata

nextToken

Token pagination untuk mengambil halaman berikutnya dari alias. Jika respons terhadap panggilan ini terpotong, Amazon Lex mengembalikan token pagination dalam respons. Untuk mengambil halaman alias berikutnya, tentukan token pagination di permintaan berikutnya.

Jenis: String

Kesalahan

BadRequestException

Permintaan tidak terbentuk dengan baik. Misalnya, nilai tidak valid atau bidang wajib hilang. Periksa nilai bidang, dan coba lagi.

Kode Status HTTP: 400

InternalFailureException

Terjadi kesalahan internal Amazon Lex. Coba permintaan Anda lagi.

Kode Status HTTP: 500

LimitExceededException

Permintaan melebihi batas. Coba permintaan Anda lagi.

Kode Status HTTP: 429

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++

- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

GetBotChannelAssociation

Layanan: Amazon Lex Model Building Service

Mengembalikan informasi tentang hubungan antara bot Amazon Lex dan platform perpesanan.

Operasi ini memerlukan izin untuk tindakan `lex:GetBotChannelAssociation`.

Minta Sintaks

```
GET /bots/botName/aliases/aliasName/channels/name HTTP/1.1
```

Parameter Permintaan URI

Permintaan menggunakan parameter URI berikut.

aliasName

Alias yang menunjuk ke versi spesifik bot Amazon Lex tempat asosiasi ini dibuat.

Batasan Panjang: Panjang minimum 1. Panjang maksimum 100.

Pola: `^([A-Za-z]_?)+$`

Diperlukan: Ya

botName

Nama bot Amazon Lex.

Kendala Panjang: Panjang minimum 2. Panjang maksimum 50.

Pola: `^([A-Za-z]_?)+$`

Diperlukan: Ya

name

Nama asosiasi antara bot dan saluran. Namanya peka huruf besar/kecil.

Batasan Panjang: Panjang minimum 1. Panjang maksimum 100.

Pola: `^([A-Za-z]_?)+$`

Wajib: Ya

Isi Permintaan

Permintaan tidak memiliki isi permintaan.

Sintaks Respons

```
HTTP/1.1 200
Content-type: application/json

{
   "botAlias": "string",
   "botConfiguration": {
      "string" : "string"
   },
   "botName": "string",
   "createdDate": number,
   "description": "string",
   "failureReason": "string",
   "name": "string",
   "status": "string",
   "type": "string"
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Layanan mengembalikan data berikut dalam format JSON.

botAlias

Alias yang menunjuk ke versi spesifik bot Amazon Lex tempat asosiasi ini dibuat.

Jenis: String

Panjang Batasan: Panjang minimum 1. Panjang maksimum 100.

Pola: `^([A-Za-z]_?)+$`

botConfiguration

Memberikan informasi bahwa platform perpesanan perlu berkomunikasi dengan bot Amazon Lex.

Tipe: Peta string ke string

Entri Peta: Jumlah maksimum 10 item.

botName

Nama bot Amazon Lex.

Jenis: String

Kendala Panjang: Panjang minimum 2. Panjang maksimum 50.

Pola: `^([A-Za-z]_?)+$`

createdDate

Tanggal hubungan antara bot dan saluran dibuat.

Tipe: Timestamp

description

Deskripsi hubungan antara bot dan saluran.

Jenis: String

Batasan Panjang: Panjang minimum 0. Panjang maksimum 200.

failureReason

Jika `status` ya`FAILED`, Amazon Lex memberikan alasan bahwa ia gagal membuat asosiasi.

Jenis: String

name

Nama asosiasi antara bot dan saluran.

Jenis: String

Panjang Batasan: Panjang minimum 1. Panjang maksimum 100.

Pola: `^([A-Za-z]_?)+$`

status

Status saluran bot.

- `CREATED`- Saluran telah dibuat dan siap digunakan.

- `IN_PROGRESS`- Pembuatan saluran sedang berlangsung.
- `FAILED`- Terjadi kesalahan saat membuat saluran. Untuk informasi tentang alasan kegagalan, lihat `failureReason` bidangnya.

Jenis: String

Nilai yang Valid: `IN_PROGRESS | CREATED | FAILED`

type

Jenis platform perpesanan.

Jenis: String

Nilai yang Valid: `Facebook | Slack | Twilio-Sms | Kik`

Kesalahan

BadRequestException

Permintaan tidak terbentuk dengan baik. Misalnya, nilai tidak valid atau bidang wajib hilang. Periksa nilai bidang, dan coba lagi.

Kode Status HTTP: 400

InternalFailureException

Terjadi kesalahan internal Amazon Lex. Coba permintaan Anda lagi.

Kode Status HTTP: 500

LimitExceededException

Permintaan melebihi batas. Coba permintaan Anda lagi.

Kode Status HTTP: 429

NotFoundException

Sumber daya yang ditentukan dalam permintaan tidak ditemukan. Periksa sumber daya dan coba lagi.

Kode Status HTTP: 404

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

GetBotChannelAssociations

Layanan: Amazon Lex Model Building Service

Mengembalikan daftar semua saluran yang terkait dengan bot yang ditentukan.

`GetBotChannelAssociations`Operasi memerlukan izin untuk `lex:GetBotChannelAssociations` tindakan tersebut.

Minta Sintaks

```
GET /bots/botName/aliases/aliasName/channels/?
maxResults=maxResults&nameContains=nameContains&nextToken=nextToken HTTP/1.1
```

Parameter Permintaan URI

Permintaan menggunakan parameter URI berikut.

aliasName

Alias yang menunjuk ke versi spesifik bot Amazon Lex tempat asosiasi ini dibuat.

Batasan Panjang: Panjang minimum 1. Panjang maksimum 100.

Pola: `^(-|^([A-Za-z]_?)+$)$`

Diperlukan: Ya

botName

Nama bot Amazon Lex di asosiasi.

Kendala Panjang: Panjang minimum 2. Panjang maksimum 50.

Pola: `^([A-Za-z]_?)+$`

Diperlukan: Ya

maxResults

Jumlah maksimum asosiasi untuk kembali dalam respons. Defaultnya adalah 50.

Rentang yang Valid: Nilai minimum 1. Nilai maksimum 50.

nameContains

Substring agar sesuai dengan nama asosiasi saluran. Asosiasi akan dikembalikan jika ada bagian dari namanya yang cocok dengan substring. Misalnya, “xyz” cocok dengan “xyzabc” dan “abcxyz.” Untuk mengembalikan semua asosiasi saluran bot, gunakan tanda hubung (“-”) sebagai parameter. `nameContains`

Batasan Panjang: Panjang minimum 1. Panjang maksimum 100.

Pola: `^([A-Za-z]_?)+$`

nextToken

Token pagination untuk mengambil halaman asosiasi berikutnya. Jika respons terhadap panggilan ini terpotong, Amazon Lex mengembalikan token pagination dalam respons. Untuk mengambil halaman asosiasi berikutnya, tentukan token pagination di permintaan berikutnya.

Isi Permintaan

Permintaan tidak memiliki isi permintaan.

Sintaks Respons

```
HTTP/1.1 200
Content-type: application/json

{
   "botChannelAssociations": [
      {
         "botAlias": "string",
         "botConfiguration": {
            "string" : "string"
         },
         "botName": "string",
         "createdDate": number,
         "description": "string",
         "failureReason": "string",
         "name": "string",
         "status": "string",
         "type": "string"
      }
   ],
   "nextToken": "string"
```

```
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Layanan mengembalikan data berikut dalam format JSON.

botChannelAssociations

Sebuah array objek, satu untuk setiap asosiasi, yang memberikan informasi tentang bot Amazon Lex dan hubungannya dengan saluran.

Tipe: Array objek BotChannelAssociation

nextToken

Token pagination yang mengambil halaman asosiasi berikutnya. Jika respons terhadap panggilan ini terpotong, Amazon Lex mengembalikan token pagination dalam respons. Untuk mengambil halaman asosiasi berikutnya, tentukan token pagination di permintaan berikutnya.

Jenis: String

Kesalahan

BadRequestException

Permintaan tidak terbentuk dengan baik. Misalnya, nilai tidak valid atau bidang wajib hilang. Periksa nilai bidang, dan coba lagi.

Kode Status HTTP: 400

InternalFailureException

Terjadi kesalahan internal Amazon Lex. Coba permintaan Anda lagi.

Kode Status HTTP: 500

LimitExceededException

Permintaan melebihi batas. Coba permintaan Anda lagi.

Kode Status HTTP: 429

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

GetBots

Layanan: Amazon Lex Model Building Service

Mengembalikan informasi bot sebagai berikut:

- Jika Anda memberikan `nameContains` bidang, respons menyertakan informasi untuk `$LATEST` versi semua bot yang namanya berisi string yang ditentukan.
- Jika Anda tidak menentukan `nameContains` bidang, operasi akan mengembalikan informasi tentang `$LATEST` versi semua bot Anda.

Operasi ini memerlukan izin untuk tindakan `lex:GetBots`.

Minta Sintaks

```
GET /bots/?maxResults=maxResults&nameContains=nameContains&nextToken=nextToken HTTP/1.1
```

Parameter Permintaan URI

Permintaan menggunakan parameter URI berikut.

maxResults

Jumlah maksimum bot yang akan dikembalikan dalam respons bahwa permintaan akan dikembalikan. Default-nya adalah 10.

Rentang yang Valid: Nilai minimum 1. Nilai maksimum 50.

nameContains

Substring agar sesuai dengan nama bot. Bot akan dikembalikan jika ada bagian dari namanya yang cocok dengan substring. Misalnya, “xyz” cocok dengan “xyzabc” dan “abcxyz.”

Kendala Panjang: Panjang minimum 2. Panjang maksimum 50.

Pola: `^([A-Za-z]_?)+$`

nextToken

Token pagination yang mengambil halaman bot berikutnya. Jika respons terhadap panggilan ini terpotong, Amazon Lex mengembalikan token pagination dalam respons. Untuk mengambil halaman bot berikutnya, tentukan token pagination di permintaan berikutnya.

Isi Permintaan

Permintaan tidak memiliki isi permintaan.

Sintaks Respons

```
HTTP/1.1 200
Content-type: application/json

{
   "bots": [
      {
         "createdDate": number,
         "description": "string",
         "lastUpdatedDate": number,
         "name": "string",
         "status": "string",
         "version": "string"
      }
   ],
   "nextToken": "string"
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Layanan mengembalikan data berikut dalam format JSON.

bots

Sebuah array `botMetadata` objek, dengan satu entri untuk setiap bot.

Tipe: Array objek BotMetadata

nextToken

Jika respons terpotong, itu termasuk token pagination yang dapat Anda tentukan dalam permintaan berikutnya untuk mengambil halaman bot berikutnya.

Jenis: String

Kesalahan

BadRequestException

Permintaan tidak terbentuk dengan baik. Misalnya, nilai tidak valid atau bidang wajib hilang. Periksa nilai bidang, dan coba lagi.

Kode Status HTTP: 400

InternalFailureException

Terjadi kesalahan internal Amazon Lex. Coba permintaan Anda lagi.

Kode Status HTTP: 500

LimitExceededException

Permintaan melebihi batas. Coba permintaan Anda lagi.

Kode Status HTTP: 429

NotFoundException

Sumber daya yang ditentukan dalam permintaan tidak ditemukan. Periksa sumber daya dan coba lagi.

Kode Status HTTP: 404

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python

- AWS SDK for Ruby V3

GetBotVersions

Layanan: Amazon Lex Model Building Service

Mendapat informasi tentang semua versi bot.

`GetBotVersions`Operasi mengembalikan `BotMetadata` objek untuk setiap versi bot. Misalnya, jika bot memiliki tiga versi bernomor, `GetBotVersions` operasi mengembalikan empat `BotMetadata` objek dalam respons, satu untuk setiap versi bernomor dan satu untuk versi. `$LATEST`

`GetBotVersions`Operasi selalu mengembalikan setidaknya satu versi, `$LATEST` versi.

Operasi ini memerlukan izin utuk tindakan `lex:GetBotVersions`.

Minta Sintaks

```
GET /bots/name/versions/?maxResults=maxResults&nextToken=nextToken HTTP/1.1
```

Parameter Permintaan URI

Permintaan menggunakan parameter URI berikut.

maxResults

Jumlah maksimum versi bot untuk kembali dalam respons. Default-nya adalah 10.

Rentang yang Valid: Nilai minimum 1. Nilai maksimum 50.

name

Nama bot untuk versi mana yang harus dikembalikan.

Kendala Panjang: Panjang minimum 2. Panjang maksimum 50.

Pola: `^([A-Za-z]_?)+$`

Diperlukan: Ya

nextToken

Token pagination untuk mengambil halaman berikutnya dari versi bot. Jika respons terhadap panggilan ini terpotong, Amazon Lex mengembalikan token pagination dalam respons. Untuk mengambil halaman versi berikutnya, tentukan token pagination di permintaan berikutnya.

Isi Permintaan

Permintaan tidak memiliki isi permintaan.

Sintaks Respons

```
HTTP/1.1 200
Content-type: application/json

{
   "bots": [
      {
         "createdDate": number,
         "description": "string",
         "lastUpdatedDate": number,
         "name": "string",
         "status": "string",
         "version": "string"
      }
   ],
   "nextToken": "string"
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Layanan mengembalikan data berikut dalam format JSON.

bots

Sebuah array `BotMetadata` objek, satu untuk setiap versi bernomor bot ditambah satu untuk `$LATEST` versi.

Tipe: Array objek BotMetadata

nextToken

Token pagination untuk mengambil halaman berikutnya dari versi bot. Jika respons terhadap panggilan ini terpotong, Amazon Lex mengembalikan token pagination dalam respons. Untuk mengambil halaman versi berikutnya, tentukan token pagination di permintaan berikutnya.

Jenis: String

Kesalahan

BadRequestException

Permintaan tidak terbentuk dengan baik. Misalnya, nilai tidak valid atau bidang wajib hilang. Periksa nilai bidang, dan coba lagi.

Kode Status HTTP: 400

InternalFailureException

Terjadi kesalahan internal Amazon Lex. Coba permintaan Anda lagi.

Kode Status HTTP: 500

LimitExceededException

Permintaan melebihi batas. Coba permintaan Anda lagi.

Kode Status HTTP: 429

NotFoundException

Sumber daya yang ditentukan dalam permintaan tidak ditemukan. Periksa sumber daya dan coba lagi.

Kode Status HTTP: 404

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python

- AWS SDK for Ruby V3

GetBuiltinIntent

Layanan: Amazon Lex Model Building Service

Mengembalikan informasi tentang maksud bawaan.

Operasi ini memerlukan izin untuk tindakan `lex:GetBuiltinIntent`.

Minta Sintaks

```
GET /builtins/intents/signature HTTP/1.1
```

Parameter Permintaan URI

Permintaan menggunakan parameter URI berikut.

signature

Pengidentifikasi unik untuk maksud bawaan. Untuk menemukan tanda tangan untuk intent, lihat Intent Bawaan Standar di Alexa Skills Kit.

Diperlukan: Ya

Isi Permintaan

Permintaan tidak memiliki isi permintaan.

Sintaks Respons

```
HTTP/1.1 200
Content-type: application/json

{
   "signature": "string",
   "slots": [
      {
         "name": "string"
      }
   ],
   "supportedLocales": [ "string" ]
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Layanan mengembalikan data berikut dalam format JSON.

signature

Pengidentifikasi unik untuk maksud bawaan.

Jenis: String

slots

Sebuah array `BuiltinIntentSlot` objek, satu entri untuk setiap jenis slot dalam maksud.

Tipe: Array objek BuiltinIntentSlot

supportedLocales

Daftar lokal yang didukung oleh intent.

Tipe: Array string

Nilai yang Valid: `de-DE | en-AU | en-GB | en-IN | en-US | es-419 | es-ES | es-US | fr-FR | fr-CA | it-IT | ja-JP | ko-KR`

Kesalahan

BadRequestException

Permintaan tidak terbentuk dengan baik. Misalnya, nilai tidak valid atau bidang wajib hilang. Periksa nilai bidang, dan coba lagi.

Kode Status HTTP: 400

InternalFailureException

Terjadi kesalahan internal Amazon Lex. Coba permintaan Anda lagi.

Kode Status HTTP: 500

LimitExceededException

Permintaan melebihi batas. Coba permintaan Anda lagi.

Kode Status HTTP: 429

NotFoundException

Sumber daya yang ditentukan dalam permintaan tidak ditemukan. Periksa sumber daya dan coba lagi.

Kode Status HTTP: 404

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

GetBuiltinIntents

Layanan: Amazon Lex Model Building Service

Mendapat daftar maksud bawaan yang memenuhi kriteria yang ditentukan.

Operasi ini memerlukan izin untuk tindakan `lex:GetBuiltinIntents`.

Minta Sintaks

```
GET /builtins/intents/?
locale=locale&maxResults=maxResults&nextToken=nextToken&signatureContains=signatureContains
 HTTP/1.1
```

Parameter Permintaan URI

Permintaan menggunakan parameter URI berikut.

locale

Daftar lokal yang didukung oleh intent.

Nilai yang Valid: `de-DE | en-AU | en-GB | en-IN | en-US | es-419 | es-ES | es-US | fr-FR | fr-CA | it-IT | ja-JP | ko-KR`

maxResults

Jumlah maksimum maksud untuk kembali dalam respons. Default-nya adalah 10.

Rentang yang Valid: Nilai minimum 1. Nilai maksimum 50.

nextToken

Token pagination yang mengambil halaman berikutnya dari intent. Jika panggilan API ini terpotong, Amazon Lex mengembalikan token pagination dalam respons. Untuk mengambil halaman intent berikutnya, gunakan token pagination di permintaan berikutnya.

signatureContains

Substring agar sesuai dengan tanda tangan intent bawaan. Maksud akan dikembalikan jika ada bagian dari tanda tangannya yang cocok dengan substring. Misalnya, “xyz” cocok dengan “xyzabc” dan “abcxyz.” Untuk menemukan tanda tangan untuk intent, lihat Intent Bawaan Standar di Alexa Skills Kit.

Isi Permintaan

Permintaan tidak memiliki isi permintaan.

Sintaks Respons

```
HTTP/1.1 200
Content-type: application/json

{
   "intents": [
      {
         "signature": "string",
         "supportedLocales": [ "string" ]
      }
   ],
   "nextToken": "string"
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Layanan mengembalikan data berikut dalam format JSON.

intents

Sebuah array `builtinIntentMetadata` objek, satu untuk setiap maksud dalam respon.

Tipe: Array objek BuiltinIntentMetadata

nextToken

Token pagination yang mengambil halaman berikutnya dari intent. Jika respons terhadap panggilan API ini terpotong, Amazon Lex mengembalikan token pagination dalam respons. Untuk mengambil halaman intent berikutnya, tentukan token pagination di permintaan berikutnya.

Jenis: String

Kesalahan

BadRequestException

Permintaan tidak terbentuk dengan baik. Misalnya, nilai tidak valid atau bidang wajib hilang. Periksa nilai bidang, dan coba lagi.

Kode Status HTTP: 400

InternalFailureException

Terjadi kesalahan internal Amazon Lex. Coba permintaan Anda lagi.

Kode Status HTTP: 500

LimitExceededException

Permintaan melebihi batas. Coba permintaan Anda lagi.

Kode Status HTTP: 429

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

GetBuiltinSlotTypes

Layanan: Amazon Lex Model Building Service

Mendapat daftar jenis slot bawaan yang memenuhi kriteria yang ditentukan.

Untuk daftar jenis slot bawaan, lihat Referensi Jenis Slot di Alexa Skills Kit.

Operasi ini memerlukan izin untuk tindakan `lex:GetBuiltInSlotTypes`.

Minta Sintaks

```
GET /builtins/slottypes/?
locale=locale&maxResults=maxResults&nextToken=nextToken&signatureContains=signatureContains
 HTTP/1.1
```

Parameter Permintaan URI

Permintaan menggunakan parameter URI berikut.

locale

Daftar lokal yang didukung oleh jenis slot.

Nilai yang Valid: `de-DE | en-AU | en-GB | en-IN | en-US | es-419 | es-ES | es-US | fr-FR | fr-CA | it-IT | ja-JP | ko-KR`

maxResults

Jumlah maksimum jenis slot untuk kembali dalam respons. Default-nya adalah 10.

Rentang yang Valid: Nilai minimum 1. Nilai maksimum 50.

nextToken

Token pagination yang mengambil halaman berikutnya dari jenis slot. Jika respons terhadap panggilan API ini terpotong, Amazon Lex mengembalikan token pagination dalam respons. Untuk mengambil halaman berikutnya dari jenis slot, tentukan token pagination di permintaan berikutnya.

signatureContains

Substring agar sesuai dengan tanda tangan tipe slot bawaan. Jenis slot akan dikembalikan jika ada bagian dari tanda tangannya yang cocok dengan substring. Misalnya, “xyz” cocok dengan “xyzabc” dan “abcxyz.”

Isi Permintaan

Permintaan tidak memiliki isi permintaan.

Sintaks Respons

```
HTTP/1.1 200
Content-type: application/json

{
   "nextToken": "string",
   "slotTypes": [
      {
         "signature": "string",
         "supportedLocales": [ "string" ]
      }
   ]
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Layanan mengembalikan data berikut dalam format JSON.

nextToken

Jika respons terpotong, respons menyertakan token pagination yang dapat Anda gunakan dalam permintaan berikutnya untuk mengambil halaman berikutnya dari jenis slot.

Jenis: String

slotTypes

Sebuah array `BuiltInSlotTypeMetadata` objek, satu entri untuk setiap jenis slot dikembalikan.

Tipe: Array objek BuiltinSlotTypeMetadata

Kesalahan

BadRequestException

Permintaan tidak terbentuk dengan baik. Misalnya, nilai tidak valid atau bidang wajib hilang. Periksa nilai bidang, dan coba lagi.

Kode Status HTTP: 400

InternalFailureException

Terjadi kesalahan internal Amazon Lex. Coba permintaan Anda lagi.

Kode Status HTTP: 500

LimitExceededException

Permintaan melebihi batas. Coba permintaan Anda lagi.

Kode Status HTTP: 429

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

GetExport

Layanan: Amazon Lex Model Building Service

Mengekspor konten sumber daya Amazon Lex dalam format tertentu.

Minta Sintaks

```
GET /exports/?exportType=exportType&name=name&resourceType=resourceType&version=version
 HTTP/1.1
```

Parameter Permintaan URI

Permintaan menggunakan parameter URI berikut.

exportType

Format data yang diekspor.

Nilai yang Valid: `ALEXA_SKILLS_KIT | LEX`

Diperlukan: Ya

name

Nama bot yang akan diekspor.

Batasan Panjang: Panjang minimum 1. Panjang maksimum 100.

Pola: `[a-zA-Z_]+`

Diperlukan: Ya

resourceType

Jenis sumber daya untuk diekspor.

Nilai yang Valid: `BOT | INTENT | SLOT_TYPE`

Diperlukan: Ya

version

Versi bot yang akan diekspor.

Batasan Panjang: Panjang minimum 1. Panjang maksimum adalah 64.

Pola: `[0-9]+`

Wajib: Ya

Isi Permintaan

Permintaan tidak memiliki isi permintaan.

Sintaks Respons

```
HTTP/1.1 200
Content-type: application/json

{
   "exportStatus": "string",
   "exportType": "string",
   "failureReason": "string",
   "name": "string",
   "resourceType": "string",
   "url": "string",
   "version": "string"
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Layanan mengembalikan data berikut dalam format JSON.

exportStatus

Status ekspor.

- `IN_PROGRESS`- Ekspor sedang berlangsung.
- `READY`- Ekspor selesai.
- `FAILED`- Ekspor tidak dapat diselesaikan.

Jenis: String

Nilai yang Valid: `IN_PROGRESS | READY | FAILED`

exportType

Format data yang diekspor.

Jenis: String

Nilai yang Valid: `ALEXA_SKILLS_KIT | LEX`

failureReason

Jika `status` ya`FAILED`, Amazon Lex memberikan alasan bahwa ia gagal mengekspor sumber daya.

Jenis: String

name

Nama bot yang diekspor.

Jenis: String

Panjang Batasan: Panjang minimum 1. Panjang maksimum 100.

Pola: `[a-zA-Z_]+`

resourceType

Jenis sumber daya yang diekspor.

Jenis: String

Nilai yang Valid: `BOT | INTENT | SLOT_TYPE`

url

URL pra-ditandatangani S3 yang menyediakan lokasi sumber daya yang diekspor. Sumber daya yang diekspor adalah arsip ZIP yang berisi sumber daya yang diekspor dalam format JSON. Struktur arsip dapat berubah. Kode Anda seharusnya tidak bergantung pada struktur arsip.

Jenis: String

version

Versi bot yang diekspor.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum adalah 64.

Pola: `[0-9]+`

Kesalahan

BadRequestException

Permintaan tidak terbentuk dengan baik. Misalnya, nilai tidak valid atau bidang wajib hilang. Periksa nilai bidang, dan coba lagi.

Kode Status HTTP: 400

InternalFailureException

Terjadi kesalahan internal Amazon Lex. Coba permintaan Anda lagi.

Kode Status HTTP: 500

LimitExceededException

Permintaan melebihi batas. Coba permintaan Anda lagi.

Kode Status HTTP: 429

NotFoundException

Sumber daya yang ditentukan dalam permintaan tidak ditemukan. Periksa sumber daya dan coba lagi.

Kode Status HTTP: 404

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2

- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

GetImport

Layanan: Amazon Lex Model Building Service

Mendapat informasi tentang pekerjaan impor yang dimulai dengan `StartImport` operasi.

Minta Sintaks

```
GET /imports/importId HTTP/1.1
```

Parameter Permintaan URI

Permintaan menggunakan parameter URI berikut.

importId

Pengidentifikasi informasi pekerjaan impor untuk dikembalikan.

Diperlukan: Ya

Isi Permintaan

Permintaan tidak memiliki isi permintaan.

Sintaks Respons

```
HTTP/1.1 200
Content-type: application/json

{
   "createdDate": number,
   "failureReason": [ "string" ],
   "importId": "string",
   "importStatus": "string",
   "mergeStrategy": "string",
   "name": "string",
   "resourceType": "string"
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Layanan mengembalikan data berikut dalam format JSON.

createdDate

Stempel waktu untuk tanggal dan waktu pekerjaan impor dibuat.

Tipe: Timestamp

failureReason

String yang menjelaskan mengapa pekerjaan impor gagal diselesaikan.

Tipe: Array string

importId

Pengidentifikasi untuk pekerjaan impor tertentu.

Jenis: String

importStatus

Status pekerjaan impor. Jika statusnya`FAILED`, Anda bisa mendapatkan alasan kegagalan dari `failureReason` lapangan.

Jenis: String

Nilai yang Valid: `IN_PROGRESS | COMPLETE | FAILED`

mergeStrategy

Tindakan yang diambil ketika ada konflik antara sumber daya yang ada dan sumber daya dalam file impor.

Jenis: String

Nilai yang Valid: `OVERWRITE_LATEST | FAIL_ON_CONFLICT`

name

Nama yang diberikan untuk pekerjaan impor.

Jenis: String

Panjang Batasan: Panjang minimum 1. Panjang maksimum 100.

Pola: `[a-zA-Z_]+`

resourceType

Jenis sumber daya yang diimpor.

Jenis: String

Nilai yang Valid: `BOT | INTENT | SLOT_TYPE`

Kesalahan

BadRequestException

Permintaan tidak terbentuk dengan baik. Misalnya, nilai tidak valid atau bidang wajib hilang. Periksa nilai bidang, dan coba lagi.

Kode Status HTTP: 400

InternalFailureException

Terjadi kesalahan internal Amazon Lex. Coba permintaan Anda lagi.

Kode Status HTTP: 500

LimitExceededException

Permintaan melebihi batas. Coba permintaan Anda lagi.

Kode Status HTTP: 429

NotFoundException

Sumber daya yang ditentukan dalam permintaan tidak ditemukan. Periksa sumber daya dan coba lagi.

Kode Status HTTP: 404

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET

- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

GetIntent

Layanan: Amazon Lex Model Building Service

Mengembalikan informasi tentang maksud. Selain nama maksud, Anda harus menentukan versi intent.

Operasi ini memerlukan izin untuk menjalankan tindakan `lex:GetIntent`.

Minta Sintaks

```
GET /intents/name/versions/version HTTP/1.1
```

Parameter Permintaan URI

Permintaan menggunakan parameter URI berikut.

name

Nama niat. Namanya peka huruf besar/kecil.

Batasan Panjang: Panjang minimum 1. Panjang maksimum 100.

Pola: `^([A-Za-z]_?)+$`

Diperlukan: Ya

version

Versi maksudnya.

Batasan Panjang: Panjang minimum 1. Panjang maksimum adalah 64.

Pola: `\$LATEST|[0-9]+`

Wajib: Ya

Isi Permintaan

Permintaan tidak memiliki isi permintaan.

Sintaks Respons

```
HTTP/1.1 200
Content-type: application/json
```

```
{
   "checksum": "string",
   "conclusionStatement": {
      "messages": [
         {
            "content": "string",
            "contentType": "string",
            "groupNumber": number
         }
      ],
      "responseCard": "string"
   },
   "confirmationPrompt": {
      "maxAttempts": number,
      "messages": [
         {
            "content": "string",
            "contentType": "string",
            "groupNumber": number
         }
      ],
      "responseCard": "string"
   },
   "createdDate": number,
   "description": "string",
   "dialogCodeHook": {
      "messageVersion": "string",
      "uri": "string"
   },
   "followUpPrompt": {
      "prompt": {
         "maxAttempts": number,
         "messages": [
            {
               "content": "string",
               "contentType": "string",
               "groupNumber": number
            }
         ],
         "responseCard": "string"
      },
      "rejectionStatement": {
         "messages": [
```

```
            {
               "content": "string",
               "contentType": "string",
               "groupNumber": number
            }
         ],
         "responseCard": "string"
      }
   },
   "fulfillmentActivity": {
      "codeHook": {
         "messageVersion": "string",
         "uri": "string"
      },
      "type": "string"
   },
   "inputContexts": [
      {
         "name": "string"
      }
   ],
   "kendraConfiguration": {
      "kendraIndex": "string",
      "queryFilterString": "string",
      "role": "string"
   },
   "lastUpdatedDate": number,
   "name": "string",
   "outputContexts": [
      {
         "name": "string",
         "timeToLiveInSeconds": number,
         "turnsToLive": number
      }
   ],
   "parentIntentSignature": "string",
   "rejectionStatement": {
      "messages": [
         {
            "content": "string",
            "contentType": "string",
            "groupNumber": number
         }
      ],
```

```
      "responseCard": "string"
   },
   "sampleUtterances": [ "string" ],
   "slots": [
      {
         "defaultValueSpec": {
            "defaultValueList": [
               {
                  "defaultValue": "string"
               }
            ]
         },
         "description": "string",
         "name": "string",
         "obfuscationSetting": "string",
         "priority": number,
         "responseCard": "string",
         "sampleUtterances": [ "string" ],
         "slotConstraint": "string",
         "slotType": "string",
         "slotTypeVersion": "string",
         "valueElicitationPrompt": {
            "maxAttempts": number,
            "messages": [
               {
                  "content": "string",
                  "contentType": "string",
                  "groupNumber": number
               }
            ],
            "responseCard": "string"
         }
      }
   ],
   "version": "string"
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Layanan mengembalikan data berikut dalam format JSON.

checksum

Checksum dari maksud.

Jenis: String

conclusionStatement

Setelah fungsi Lambda yang ditentukan dalam `fulfillmentActivity` elemen memenuhi intent, Amazon Lex menyampaikan pernyataan ini kepada pengguna.

Tipe: Objek Statement

confirmationPrompt

Jika didefinisikan dalam bot, Amazon Lex menggunakan prompt untuk mengonfirmasi maksud sebelum memenuhi permintaan pengguna. Untuk informasi selengkapnya, lihat PutIntent.

Tipe: Objek Prompt

createdDate

Tanggal dimana niat itu dibuat.

Tipe: Timestamp

description

Deskripsi niat.

Jenis: String

Batasan Panjang: Panjang minimum 0. Panjang maksimum 200.

dialogCodeHook

Jika didefinisikan dalam bot, Amazon Amazon Lex memanggil fungsi Lambda ini untuk setiap input pengguna. Untuk informasi selengkapnya, lihat PutIntent.

Tipe: Objek CodeHook

followUpPrompt

Jika didefinisikan dalam bot, Amazon Lex menggunakan prompt ini untuk meminta aktivitas pengguna tambahan setelah maksud terpenuhi. Untuk informasi selengkapnya, lihat PutIntent.

Tipe: Objek FollowUpPrompt

fulfillmentActivity

Menjelaskan bagaimana niat terpenuhi. Untuk informasi selengkapnya, lihat PutIntent.

Tipe: Objek FulfillmentActivity

inputContexts

Larik `InputContext` objek yang mencantumkan konteks yang harus aktif untuk Amazon Lex untuk memilih maksud dalam percakapan dengan pengguna.

Tipe: Array objek InputContext

Anggota Array: Jumlah minimum 0 item. Jumlah maksimum 5 item.

kendraConfiguration

Informasi konfigurasi, jika ada, untuk terhubung ke indeks Amazon Kendra dengan maksud. `AMAZON.KendraSearchIntent`

Tipe: Objek KendraConfiguration

lastUpdatedDate

Tanggal niat diperbarui. Saat Anda membuat sumber daya, tanggal pembuatan dan tanggal terakhir diperbarui adalah sama.

Tipe: Timestamp

name

Nama niat.

Jenis: String

Panjang Batasan: Panjang minimum 1. Panjang maksimum 100.

Pola: `^([A-Za-z]_?)+$`

outputContexts

Array `OutputContext` objek yang mencantumkan konteks yang mengaktifkan intent saat intent terpenuhi.

Tipe: Array objek OutputContext

Anggota Array: Jumlah minimum 0 item. Jumlah maksimum 10 item.

parentIntentSignature

Pengenal unik untuk maksud bawaan.

Jenis: String

rejectionStatement

Jika pengguna menjawab “tidak” untuk pertanyaan yang didefinisikan dalam`confirmationPrompt`, Amazon Lex merespons dengan pernyataan ini untuk mengakui bahwa maksud tersebut dibatalkan.

Tipe: Objek Statement

sampleUtterances

Larik contoh ucapan yang dikonfigurasi untuk maksud.

Tipe: Array string.

Anggota Array: Jumlah minimum 0 item. Jumlah maksimum 1500 item.

Batasan Panjang: Panjang minimum 1. Panjang maksimum 200.

slots

Array slot intent yang dikonfigurasi untuk maksud tersebut.

Tipe: Array objek Slot

Anggota Array: Jumlah minimum 0 item. Jumlah maksimum 100 item.

version

Versi maksudnya.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum adalah 64.

Pola: `\$LATEST|[0-9]+`

Kesalahan

BadRequestException

Permintaan tidak terbentuk dengan baik. Misalnya, nilai tidak valid atau bidang wajib hilang. Periksa nilai bidang, dan coba lagi.

Kode Status HTTP: 400

InternalFailureException

Terjadi kesalahan internal Amazon Lex. Coba permintaan Anda lagi.

Kode Status HTTP: 500

LimitExceededException

Permintaan melebihi batas. Coba permintaan Anda lagi.

Kode Status HTTP: 429

NotFoundException

Sumber daya yang ditentukan dalam permintaan tidak ditemukan. Periksa sumber daya dan coba lagi.

Kode Status HTTP: 404

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python

- AWS SDK for Ruby V3

GetIntents

Layanan: Amazon Lex Model Building Service

Mengembalikan informasi maksud sebagai berikut:

- Jika Anda menentukan `nameContains` bidang, mengembalikan `$LATEST` versi semua maksud yang berisi string tertentu.
- Jika Anda tidak menentukan `nameContains` bidang, mengembalikan informasi tentang `$LATEST` versi semua maksud.

Operasi membutuhkan izin untuk `lex:GetIntents` tindakan tersebut.

Minta Sintaks

```
GET /intents/?maxResults=maxResults&nameContains=nameContains&nextToken=nextToken
 HTTP/1.1
```

Parameter Permintaan URI

Permintaan menggunakan parameter URI berikut.

maxResults

Jumlah maksimum maksud untuk kembali dalam respons. Default-nya adalah 10.

Rentang yang Valid: Nilai minimum 1. Nilai maksimum 50.

nameContains

Substring untuk mencocokkan nama maksud. Maksud akan dikembalikan jika ada bagian dari namanya yang cocok dengan substring. Misalnya, “xyz” cocok dengan “xyzabc” dan “abcxyz.”

Batasan Panjang: Panjang minimum 1. Panjang maksimum 100.

Pola: `^([A-Za-z]_?)+$`

nextToken

Token pagination yang mengambil halaman berikutnya dari intent. Jika respons terhadap panggilan API ini terpotong, Amazon Lex mengembalikan token pagination dalam respons. Untuk mengambil halaman intent berikutnya, tentukan token pagination di permintaan berikutnya.

Isi Permintaan

Permintaan tidak memiliki isi permintaan.

Sintaks Respons

```
HTTP/1.1 200
Content-type: application/json

{
   "intents": [
      {
         "createdDate": number,
         "description": "string",
         "lastUpdatedDate": number,
         "name": "string",
         "version": "string"
      }
   ],
   "nextToken": "string"
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Layanan mengembalikan data berikut dalam format JSON.

intents

Susunan objek `Intent`. Untuk informasi selengkapnya, lihat PutBot.

Tipe: Array objek IntentMetadata

nextToken

Jika respons terpotong, respons menyertakan token pagination yang dapat Anda tentukan dalam permintaan berikutnya untuk mengambil halaman intent berikutnya.

Jenis: String

Kesalahan

BadRequestException

Permintaan tidak terbentuk dengan baik. Misalnya, nilai tidak valid atau bidang wajib hilang. Periksa nilai bidang, dan coba lagi.

Kode Status HTTP: 400

InternalFailureException

Terjadi kesalahan internal Amazon Lex. Coba permintaan Anda lagi.

Kode Status HTTP: 500

LimitExceededException

Permintaan melebihi batas. Coba permintaan Anda lagi.

Kode Status HTTP: 429

NotFoundException

Sumber daya yang ditentukan dalam permintaan tidak ditemukan. Periksa sumber daya dan coba lagi.

Kode Status HTTP: 404

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python

- AWS SDK for Ruby V3

GetIntentVersions

Layanan: Amazon Lex Model Building Service

Mendapat informasi tentang semua versi maksud.

`GetIntentVersions`Operasi mengembalikan `IntentMetadata` objek untuk setiap versi maksud. Misalnya, jika intent memiliki tiga versi bernomor, `GetIntentVersions` operasi akan mengembalikan empat `IntentMetadata` objek dalam respons, satu untuk setiap versi bernomor dan satu untuk versi. `$LATEST`

`GetIntentVersions`Operasi selalu mengembalikan setidaknya satu versi, `$LATEST` versi.

Operasi ini memerlukan izin utuk tindakan `lex:GetIntentVersions`.

Minta Sintaks

```
GET /intents/name/versions/?maxResults=maxResults&nextToken=nextToken HTTP/1.1
```

Parameter Permintaan URI

Permintaan menggunakan parameter URI berikut.

maxResults

Jumlah maksimum versi intent yang akan ditampilkan dalam respons. Default-nya adalah 10.

Rentang yang Valid: Nilai minimum 1. Nilai maksimum 50.

name

Nama maksud untuk versi mana yang harus dikembalikan.

Batasan Panjang: Panjang minimum 1. Panjang maksimum 100.

Pola: `^([A-Za-z]_?)+$`

Diperlukan: Ya

nextToken

Token pagination untuk mengambil halaman berikutnya dari versi intent. Jika respons terhadap panggilan ini terpotong, Amazon Lex mengembalikan token pagination dalam respons. Untuk mengambil halaman versi berikutnya, tentukan token pagination di permintaan berikutnya.

Isi Permintaan

Permintaan tidak memiliki isi permintaan.

Sintaks Respons

```
HTTP/1.1 200
Content-type: application/json

{
   "intents": [
      {
         "createdDate": number,
         "description": "string",
         "lastUpdatedDate": number,
         "name": "string",
         "version": "string"
      }
   ],
   "nextToken": "string"
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Layanan mengembalikan data berikut dalam format JSON.

intents

Sebuah array `IntentMetadata` objek, satu untuk setiap versi bernomor dari intent ditambah satu untuk versi. `$LATEST`

Tipe: Array objek IntentMetadata

nextToken

Token pagination untuk mengambil halaman berikutnya dari versi intent. Jika respons terhadap panggilan ini terpotong, Amazon Lex mengembalikan token pagination dalam respons. Untuk mengambil halaman versi berikutnya, tentukan token pagination di permintaan berikutnya.

Jenis: String

Kesalahan

BadRequestException

Permintaan tidak terbentuk dengan baik. Misalnya, nilai tidak valid atau bidang wajib hilang. Periksa nilai bidang, dan coba lagi.

Kode Status HTTP: 400

InternalFailureException

Terjadi kesalahan internal Amazon Lex. Coba permintaan Anda lagi.

Kode Status HTTP: 500

LimitExceededException

Permintaan melebihi batas. Coba permintaan Anda lagi.

Kode Status HTTP: 429

NotFoundException

Sumber daya yang ditentukan dalam permintaan tidak ditemukan. Periksa sumber daya dan coba lagi.

Kode Status HTTP: 404

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python

- AWS SDK for Ruby V3

GetMigration

Layanan: Amazon Lex Model Building Service

Memberikan detail tentang migrasi yang sedang berlangsung atau lengkap dari bot Amazon Lex V1 ke bot Amazon Lex V2. Gunakan operasi ini untuk melihat peringatan migrasi dan peringatan yang terkait dengan migrasi.

Minta Sintaks

```
GET /migrations/migrationId HTTP/1.1
```

Parameter Permintaan URI

Permintaan menggunakan parameter URI berikut.

migrationId

Pengidentifikasi unik migrasi yang akan dilihat. `migrationID`Itu dikembalikan oleh StartMigration operasi.

Kendala Panjang: Panjang tetap 10.

Pola: `^[0-9a-zA-Z]+$`

Wajib: Ya

Isi Permintaan

Permintaan tidak memiliki isi permintaan.

Sintaks Respons

```
HTTP/1.1 200
Content-type: application/json

{
   "alerts": [
      {
         "details": [ "string" ],
         "message": "string",
         "referenceURLs": [ "string" ],
         "type": "string"
```

```
      }
    ],
    "migrationId": "string",
    "migrationStatus": "string",
    "migrationStrategy": "string",
    "migrationTimestamp": number,
    "v1BotLocale": "string",
    "v1BotName": "string",
    "v1BotVersion": "string",
    "v2BotId": "string",
    "v2BotRole": "string"
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Layanan mengembalikan data berikut dalam format JSON.

alerts

Daftar peringatan dan peringatan yang menunjukkan masalah dengan migrasi bot Amazon Lex V1 ke Amazon Lex V2. Anda menerima peringatan ketika fitur Amazon Lex V1 memiliki implementasi yang berbeda di Amazon Lex V2.

Untuk informasi selengkapnya, lihat Memigrasi bot di panduan pengembang Amazon Lex V2.

Tipe: Array objek MigrationAlert

migrationId

Pengidentifikasi unik migrasi. Ini sama dengan pengidentifikasi yang digunakan saat memanggil `GetMigration` operasi.

Jenis: String

Kendala Panjang: Panjang tetap 10.

Pola: `^[0-9a-zA-Z]+$`

migrationStatus

Menunjukkan status migrasi. Ketika status migrasi selesai dan bot tersedia di Amazon Lex V2. `COMPLETE` Mungkin ada peringatan dan peringatan yang perlu diselesaikan untuk menyelesaikan migrasi.

Jenis: String

Nilai yang Valid: `IN_PROGRESS | COMPLETED | FAILED`

migrationStrategy

Strategi yang digunakan untuk melakukan migrasi.

- `CREATE_NEW`- Membuat bot Amazon Lex V2 baru dan memigrasikan bot Amazon Lex V1 ke bot baru.
- `UPDATE_EXISTING`- Menimpa metadata bot Amazon Lex V2 yang ada dan lokal yang sedang dimigrasikan. Itu tidak mengubah lokal lain di bot Amazon Lex V2. Jika lokal tidak ada, lokal baru dibuat di bot Amazon Lex V2.

Jenis: String

Nilai yang Valid: `CREATE_NEW | UPDATE_EXISTING`

migrationTimestamp

Tanggal dan waktu migrasi dimulai.

Tipe: Timestamp

v1BotLocale

Lokal bot Amazon Lex V1 bermigrasi ke Amazon Lex V2.

Jenis: String

Nilai yang Valid: `de-DE | en-AU | en-GB | en-IN | en-US | es-419 | es-ES | es-US | fr-FR | fr-CA | it-IT | ja-JP | ko-KR`

v1BotName

Nama bot Amazon Lex V1 bermigrasi ke Amazon Lex V2.

Jenis: String

Kendala Panjang: Panjang minimum 2. Panjang maksimum 50.

Pola: `^([A-Za-z]_?)+$`

v1BotVersion

Versi bot Amazon Lex V1 bermigrasi ke Amazon Lex V2.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum adalah 64.

Pola: `\$LATEST|[0-9]+`

v2BotId

Pengenal unik bot Amazon Lex V2 tempat Amazon Lex V1 dimigrasikan.

Jenis: String

Kendala Panjang: Panjang tetap 10.

Pola: `^[0-9a-zA-Z]+$`

v2BotRole

Peran IAM yang digunakan Amazon Lex untuk menjalankan bot Amazon Lex V2.

Jenis: String

Batasan Panjang: Panjang minimum 20. Panjang maksimum 2048.

Pola: `^arn:[\w\-]+:iam::[\d]{12}:role/.+$`

Kesalahan

BadRequestException

Permintaan tidak terbentuk dengan baik. Misalnya, nilai tidak valid atau bidang wajib hilang. Periksa nilai bidang, dan coba lagi.

Kode Status HTTP: 400

InternalFailureException

Terjadi kesalahan internal Amazon Lex. Coba permintaan Anda lagi.

Kode Status HTTP: 500

LimitExceededException

Permintaan melebihi batas. Coba permintaan Anda lagi.

Kode Status HTTP: 429

NotFoundException

Sumber daya yang ditentukan dalam permintaan tidak ditemukan. Periksa sumber daya dan coba lagi.

Kode Status HTTP: 404

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

GetMigrations

Layanan: Amazon Lex Model Building Service

Mendapat daftar migrasi antara Amazon Lex V1 dan Amazon Lex V2.

Minta Sintaks

```
GET /migrations?
maxResults=maxResults&migrationStatusEquals=migrationStatusEquals&nextToken=nextToken&sortByAtt
 HTTP/1.1
```

Parameter Permintaan URI

Permintaan menggunakan parameter URI berikut.

maxResults

Jumlah maksimum migrasi untuk kembali dalam respons. Default-nya adalah 10.

Rentang yang Valid: Nilai minimum 1. Nilai maksimum 50.

migrationStatusEquals

Memfilter daftar untuk hanya berisi migrasi dalam keadaan tertentu.

Nilai yang Valid: `IN_PROGRESS | COMPLETED | FAILED`

nextToken

Token pagination yang mengambil halaman migrasi berikutnya. Jika respons terhadap operasi ini terpotong, Amazon Lex mengembalikan token pagination dalam respons. Untuk mengambil halaman migrasi berikutnya, tentukan token pagination dalam permintaan.

sortByAttribute

Bidang untuk mengurutkan daftar migrasi berdasarkan. Anda dapat mengurutkan berdasarkan nama bot Amazon Lex V1 atau tanggal dan waktu migrasi dimulai.

Nilai yang Valid: `V1_BOT_NAME | MIGRATION_DATE_TIME`

sortByOrder

Urutan jadi urutkan daftar.

Nilai yang Valid: `ASCENDING | DESCENDING`

v1BotNameContains

Filter daftar untuk hanya berisi bot yang namanya berisi string yang ditentukan. String dicocokkan di mana saja dalam nama bot.

Kendala Panjang: Panjang minimum 2. Panjang maksimum 50.

Pola: `^([A-Za-z]_?)+$`

Isi Permintaan

Permintaan tidak memiliki isi permintaan.

Sintaks Respons

```
HTTP/1.1 200
Content-type: application/json

{
   "migrationSummaries": [
      {
         "migrationId": "string",
         "migrationStatus": "string",
         "migrationStrategy": "string",
         "migrationTimestamp": number,
         "v1BotLocale": "string",
         "v1BotName": "string",
         "v1BotVersion": "string",
         "v2BotId": "string",
         "v2BotRole": "string"
      }
   ],
   "nextToken": "string"
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Layanan mengembalikan data berikut dalam format JSON.

migrationSummaries

Array ringkasan untuk migrasi dari Amazon Lex V1 ke Amazon Lex V2. Untuk melihat detail migrasi, gunakan `migrationId` dari ringkasan dalam panggilan ke GetMigration operasi.

Tipe: Array objek MigrationSummary

nextToken

Jika respons terpotong, itu termasuk token pagination yang dapat Anda tentukan dalam permintaan berikutnya untuk mengambil halaman migrasi berikutnya.

Jenis: String

Kesalahan

BadRequestException

Permintaan tidak terbentuk dengan baik. Misalnya, nilai tidak valid atau bidang wajib hilang. Periksa nilai bidang, dan coba lagi.

Kode Status HTTP: 400

InternalFailureException

Terjadi kesalahan internal Amazon Lex. Coba permintaan Anda lagi.

Kode Status HTTP: 500

LimitExceededException

Permintaan melebihi batas. Coba permintaan Anda lagi.

Kode Status HTTP: 429

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++

- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

GetSlotType

Layanan: Amazon Lex Model Building Service

Mengembalikan informasi tentang versi tertentu dari jenis slot. Selain menentukan nama jenis slot, Anda harus menentukan versi jenis slot.

Operasi ini memerlukan izin utuk tindakan `lex:GetSlotType`.

Minta Sintaks

```
GET /slottypes/name/versions/version HTTP/1.1
```

Parameter Permintaan URI

Permintaan menggunakan parameter URI berikut.

name

Nama jenis slot. Namanya peka huruf besar/kecil.

Batasan Panjang: Panjang minimum 1. Panjang maksimum 100.

Pola: `^([A-Za-z]_?)+$`

Diperlukan: Ya

version

Versi dari jenis slot.

Batasan Panjang: Panjang minimum 1. Panjang maksimum adalah 64.

Pola: `\$LATEST|[0-9]+`

Wajib: Ya

Isi Permintaan

Permintaan tidak memiliki isi permintaan.

Sintaks Respons

```
HTTP/1.1 200
Content-type: application/json
```

```
{
   "checksum": "string",
   "createdDate": number,
   "description": "string",
   "enumerationValues": [
      {
         "synonyms": [ "string" ],
         "value": "string"
      }
   ],
   "lastUpdatedDate": number,
   "name": "string",
   "parentSlotTypeSignature": "string",
   "slotTypeConfigurations": [
      {
         "regexConfiguration": {
            "pattern": "string"
         }
      }
   ],
   "valueSelectionStrategy": "string",
   "version": "string"
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Layanan mengembalikan data berikut dalam format JSON.

checksum

Checksum dari $LATEST versi jenis slot.

Jenis: String

createdDate

Tanggal jenis slot dibuat.

Tipe: Timestamp

description

Deskripsi jenis slot.

Jenis: String

Batasan Panjang: Panjang minimum 0. Panjang maksimum 200.

enumerationValues

Daftar `EnumerationValue` objek yang mendefinisikan nilai-nilai yang dapat diambil oleh jenis slot.

Tipe: Array objek EnumerationValue

Anggota Array: Jumlah minimum 0 item. Jumlah maksimum 10000 item.

lastUpdatedDate

Tanggal bahwa jenis slot diperbarui. Saat Anda membuat sumber daya, tanggal pembuatan dan tanggal pembaruan terakhir adalah sama.

Tipe: Timestamp

name

Nama jenis slot.

Jenis: String

Panjang Batasan: Panjang minimum 1. Panjang maksimum 100.

Pola: `^([A-Za-z]_?)+$`

parentSlotTypeSignature

Jenis slot built-in digunakan sebagai induk untuk jenis slot.

Jenis: String

Panjang Batasan: Panjang minimum 1. Panjang maksimum 100.

Pola: `^((AMAZON\.)_?|[A-Za-z]_?)+`

slotTypeConfigurations

Informasi konfigurasi yang memperluas tipe slot bawaan induk.

Tipe: Array objek SlotTypeConfiguration

Anggota Array: Jumlah minimum 0 item. Jumlah maksimum 10 item.

valueSelectionStrategy

Strategi yang digunakan Amazon Lex untuk menentukan nilai slot. Untuk informasi selengkapnya, lihat PutSlotType.

Tipe: String

Nilai yang Valid: `ORIGINAL_VALUE | TOP_RESOLUTION`

version

Versi dari jenis slot.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum adalah 64.

Pola: `\$LATEST|[0-9]+`

Kesalahan

BadRequestException

Permintaan tidak terbentuk dengan baik. Misalnya, nilai tidak valid atau bidang wajib hilang. Periksa nilai bidang, dan coba lagi.

Kode Status HTTP: 400

InternalFailureException

Terjadi kesalahan internal Amazon Lex. Coba permintaan Anda lagi.

Kode Status HTTP: 500

LimitExceededException

Permintaan melebihi batas. Coba permintaan Anda lagi.

Kode Status HTTP: 429

NotFoundException

Sumber daya yang ditentukan dalam permintaan tidak ditemukan. Periksa sumber daya dan coba lagi.

Kode Status HTTP: 404

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

GetSlotTypes

Layanan: Amazon Lex Model Building Service

Mengembalikan informasi jenis slot sebagai berikut:

- Jika Anda menentukan `nameContains` bidang, mengembalikan `$LATEST` versi semua jenis slot yang berisi string yang ditentukan.
- Jika Anda tidak menentukan `nameContains` bidang, mengembalikan informasi tentang `$LATEST` versi semua jenis slot.

Operasi membutuhkan izin untuk `lex:GetSlotTypes` tindakan tersebut.

Minta Sintaks

```
GET /slottypes/?maxResults=maxResults&nameContains=nameContains&nextToken=nextToken
 HTTP/1.1
```

Parameter Permintaan URI

Permintaan menggunakan parameter URI berikut.

maxResults

Jumlah maksimum jenis slot untuk kembali dalam respons. Default-nya adalah 10.

Rentang yang Valid: Nilai minimum 1. Nilai maksimum 50.

nameContains

Substring agar sesuai dengan nama tipe slot. Jenis slot akan dikembalikan jika ada bagian dari namanya yang cocok dengan substring. Misalnya, “xyz” cocok dengan “xyzabc” dan “abcxyz.”

Batasan Panjang: Panjang minimum 1. Panjang maksimum 100.

Pola: `^([A-Za-z]_?)+$`

nextToken

Token pagination yang mengambil halaman berikutnya dari jenis slot. Jika respons terhadap panggilan API ini terpotong, Amazon Lex mengembalikan token pagination dalam respons. Untuk mengambil halaman berikutnya dari jenis slot, tentukan token pagination di permintaan berikutnya.

Isi Permintaan

Permintaan tidak memiliki isi permintaan.

Sintaks Respons

```
HTTP/1.1 200
Content-type: application/json

{
   "nextToken": "string",
   "slotTypes": [
      {
         "createdDate": number,
         "description": "string",
         "lastUpdatedDate": number,
         "name": "string",
         "version": "string"
      }
   ]
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Layanan mengembalikan data berikut dalam format JSON.

nextToken

Jika respons terpotong, itu termasuk token pagination yang dapat Anda tentukan dalam permintaan berikutnya untuk mengambil halaman berikutnya dari jenis slot.

Jenis: String

slotTypes

Sebuah array objek, satu untuk setiap jenis slot, yang menyediakan informasi seperti nama jenis slot, versi, dan deskripsi.

Tipe: Array objek SlotTypeMetadata

Kesalahan

BadRequestException

Permintaan tidak terbentuk dengan baik. Misalnya, nilai tidak valid atau bidang wajib hilang. Periksa nilai bidang, dan coba lagi.

Kode Status HTTP: 400

InternalFailureException

Terjadi kesalahan internal Amazon Lex. Coba permintaan Anda lagi.

Kode Status HTTP: 500

LimitExceededException

Permintaan melebihi batas. Coba permintaan Anda lagi.

Kode Status HTTP: 429

NotFoundException

Sumber daya yang ditentukan dalam permintaan tidak ditemukan. Periksa sumber daya dan coba lagi.

Kode Status HTTP: 404

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python

- AWS SDK for Ruby V3

GetSlotTypeVersions

Layanan: Amazon Lex Model Building Service

Mendapat informasi tentang semua versi dari jenis slot.

`GetSlotTypeVersions`Operasi mengembalikan `SlotTypeMetadata` objek untuk setiap versi dari jenis slot. Misalnya, jika jenis slot memiliki tiga versi bernomor, `GetSlotTypeVersions` operasi mengembalikan empat `SlotTypeMetadata` objek dalam respons, satu untuk setiap versi bernomor dan satu untuk versi. `$LATEST`

`GetSlotTypeVersions`Operasi selalu mengembalikan setidaknya satu versi, `$LATEST` versi.

Operasi ini memerlukan izin utuk tindakan `lex:GetSlotTypeVersions`.

Minta Sintaks

```
GET /slottypes/name/versions/?maxResults=maxResults&nextToken=nextToken HTTP/1.1
```

Parameter Permintaan URI

Permintaan menggunakan parameter URI berikut.

maxResults

Jumlah maksimum versi jenis slot untuk kembali dalam respons. Default-nya adalah 10.

Rentang yang Valid: Nilai minimum 1. Nilai maksimum 50.

name

Nama jenis slot untuk versi mana yang harus dikembalikan.

Batasan Panjang: Panjang minimum 1. Panjang maksimum 100.

Pola: `^([A-Za-z]_?)+$`

Diperlukan: Ya

nextToken

Token pagination untuk mengambil halaman berikutnya dari versi jenis slot. Jika respons terhadap panggilan ini terpotong, Amazon Lex mengembalikan token pagination dalam respons. Untuk mengambil halaman versi berikutnya, tentukan token pagination di permintaan berikutnya.

Isi Permintaan

Permintaan tidak memiliki isi permintaan.

Sintaks Respons

```
HTTP/1.1 200
Content-type: application/json

{
   "nextToken": "string",
   "slotTypes": [
      {
         "createdDate": number,
         "description": "string",
         "lastUpdatedDate": number,
         "name": "string",
         "version": "string"
      }
   ]
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Layanan mengembalikan data berikut dalam format JSON.

nextToken

Token pagination untuk mengambil halaman berikutnya dari versi jenis slot. Jika respons terhadap panggilan ini terpotong, Amazon Lex mengembalikan token pagination dalam respons. Untuk mengambil halaman versi berikutnya, tentukan token pagination di permintaan berikutnya.

Jenis: String

slotTypes

Sebuah array `SlotTypeMetadata` objek, satu untuk setiap versi nomor dari jenis slot ditambah satu untuk `$LATEST` versi.

Tipe: Array objek SlotTypeMetadata

Kesalahan

BadRequestException

Permintaan tidak terbentuk dengan baik. Misalnya, nilai tidak valid atau bidang wajib hilang. Periksa nilai bidang, dan coba lagi.

Kode Status HTTP: 400

InternalFailureException

Terjadi kesalahan internal Amazon Lex. Coba permintaan Anda lagi.

Kode Status HTTP: 500

LimitExceededException

Permintaan melebihi batas. Coba permintaan Anda lagi.

Kode Status HTTP: 429

NotFoundException

Sumber daya yang ditentukan dalam permintaan tidak ditemukan. Periksa sumber daya dan coba lagi.

Kode Status HTTP: 404

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python

- AWS SDK for Ruby V3

GetUtterancesView

Layanan: Amazon Lex Model Building Service

Gunakan `GetUtterancesView` operasi untuk mendapatkan informasi tentang ucapan yang telah dibuat pengguna Anda ke bot Anda. Anda dapat menggunakan daftar ini untuk menyetel ucapan yang ditanggapi bot Anda.

Misalnya, katakan bahwa Anda telah membuat bot untuk memesan bunga. Setelah pengguna Anda menggunakan bot Anda untuk sementara waktu, gunakan `GetUtterancesView` operasi untuk melihat permintaan yang telah mereka buat dan apakah mereka telah berhasil. Anda mungkin menemukan bahwa ucapan “Saya ingin bunga” tidak dikenali. Anda dapat menambahkan ucapan ini ke `OrderFlowers` maksud sehingga bot Anda mengenali ucapan itu.

Setelah Anda menerbitkan versi baru bot, Anda bisa mendapatkan informasi tentang versi lama dan yang baru sehingga Anda dapat membandingkan kinerja di kedua versi.

Statistik ucapan dihasilkan sekali sehari. Data tersedia selama 15 hari terakhir. Anda dapat meminta informasi hingga 5 versi bot Anda di setiap permintaan. Amazon Lex mengembalikan ucapan yang paling sering diterima oleh bot dalam 15 hari terakhir. Respons berisi informasi tentang maksimum 100 ucapan untuk setiap versi.

Statistik ucapan tidak dihasilkan dalam kondisi berikut:

- `childDirected`Bidang disetel ke true saat bot dibuat.
- Anda menggunakan slot obfuscation dengan satu atau lebih slot.
- Anda memilih untuk tidak berpartisipasi dalam meningkatkan Amazon Lex.

Operasi ini memerlukan izin utuk tindakan `lex:GetUtterancesView`.

Minta Sintaks

```
GET /bots/botname/utterances?
view=aggregation&bot_versions=botVersions&status_type=statusType HTTP/1.1
```

Parameter Permintaan URI

Permintaan menggunakan parameter URI berikut.

botname

Nama bot yang informasi ucapannya harus dikembalikan.

Kendala Panjang: Panjang minimum 2. Panjang maksimum 50.

Pola: `^([A-Za-z]_?)+$`

Diperlukan: Ya

botVersions

Array versi bot yang informasi ucapannya harus dikembalikan. Batasnya adalah 5 versi per permintaan.

Anggota Array: Jumlah minimum 1 item. Jumlah maksimum 5 item.

Batasan Panjang: Panjang minimum 1. Panjang maksimum adalah 64.

Pola: `\$LATEST|[0-9]+`

Diperlukan: Ya

statusType

Untuk mengembalikan ucapan yang diakui dan ditangani, gunakan. `Detected` Untuk mengembalikan ucapan yang tidak dikenali, gunakan. `Missed`

Nilai yang Valid: `Detected | Missed`

Diperlukan: Ya

Isi Permintaan

Permintaan tidak memiliki isi permintaan.

Sintaks Respons

```
HTTP/1.1 200
Content-type: application/json

{
   "botName": "string",
   "utterances": [
      {
         "botVersion": "string",
```

```
         "utterances": [
            {
               "count": number,
               "distinctUsers": number,
               "firstUtteredDate": number,
               "lastUtteredDate": number,
               "utteranceString": "string"
            }
         ]
      }
   ]
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Layanan mengembalikan data berikut dalam format JSON.

botName

Nama bot yang informasi ucapannya dikembalikan.

Jenis: String

Kendala Panjang: Panjang minimum 2. Panjang maksimum 50.

Pola: `^([A-Za-z]_?)+$`

utterances

Array UtteranceList objek, masing-masing berisi daftar UtteranceData objek yang menggambarkan ucapan yang diproses oleh bot Anda. Respons berisi maksimal 100 `UtteranceData` objek untuk setiap versi. Amazon Lex mengembalikan ucapan yang paling sering diterima oleh bot dalam 15 hari terakhir.

Tipe: Array objek UtteranceList

Kesalahan

BadRequestException

Permintaan tidak terbentuk dengan baik. Misalnya, nilai tidak valid atau bidang wajib hilang. Periksa nilai bidang, dan coba lagi.

Kode Status HTTP: 400

InternalFailureException

Terjadi kesalahan internal Amazon Lex. Coba permintaan Anda lagi.

Kode Status HTTP: 500

LimitExceededException

Permintaan melebihi batas. Coba permintaan Anda lagi.

Kode Status HTTP: 429

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

ListTagsForResource

Layanan: Amazon Lex Model Building Service

Mendapat daftar tag yang terkait dengan sumber daya yang ditentukan. Hanya bot, alias bot, dan saluran bot yang dapat memiliki tag yang terkait dengannya.

Minta Sintaks

```
GET /tags/resourceArn HTTP/1.1
```

Parameter Permintaan URI

Permintaan menggunakan parameter URI berikut.

resourceArn

Nama Sumber Daya Amazon (ARN) dari sumber daya untuk mendapatkan daftar tag untuk.

Batasan Panjang: Panjang minimum 1. Panjang maksimum 1011.

Diperlukan: Ya

Isi Permintaan

Permintaan tidak memiliki isi permintaan.

Sintaks Respons

```
HTTP/1.1 200
Content-type: application/json

{
   "tags": [
      {
         "key": "string",
         "value": "string"
      }
   ]
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Layanan mengembalikan data berikut dalam format JSON.

tags

Tag yang terkait dengan sumber daya.

Tipe: Array objek Tag

Anggota Array: Jumlah minimum 0 item. Jumlah maksimum 200 item.

Kesalahan

BadRequestException

Permintaan tidak terbentuk dengan baik. Misalnya, nilai tidak valid atau bidang wajib hilang. Periksa nilai bidang, dan coba lagi.

Kode Status HTTP: 400

InternalFailureException

Terjadi kesalahan internal Amazon Lex. Coba permintaan Anda lagi.

Kode Status HTTP: 500

LimitExceededException

Permintaan melebihi batas. Coba permintaan Anda lagi.

Kode Status HTTP: 429

NotFoundException

Sumber daya yang ditentukan dalam permintaan tidak ditemukan. Periksa sumber daya dan coba lagi.

Kode Status HTTP: 404

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah

- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

PutBot

Layanan: Amazon Lex Model Building Service

Membuat bot percakapan Amazon Lex atau menggantikan bot yang ada. Saat Anda membuat atau memperbarui bot, Anda hanya diminta untuk menentukan nama, lokal, dan apakah bot diarahkan ke anak-anak di bawah usia 13 tahun. Anda dapat menggunakan ini untuk menambahkan intent nanti, atau untuk menghapus intent dari bot yang ada. Saat Anda membuat bot dengan informasi minimum, bot dibuat atau diperbarui tetapi Amazon Lex mengembalikan responsFAILED. Anda dapat membangun bot setelah Anda menambahkan satu atau lebih maksud. Untuk informasi lebih lanjut tentang bot Amazon Lex, lihatAmazon Lex: Cara Kerjanya.

Jika Anda menentukan nama bot yang ada, bidang dalam permintaan menggantikan nilai yang ada di $LATEST versi bot. Amazon Lex menghapus semua bidang yang tidak Anda berikan nilainya dalam permintaan, kecuali `privacySettings` bidang `idleTTLInSeconds` dan, yang disetel ke nilai defaultnya. Jika Anda tidak menentukan nilai untuk bidang wajib, Amazon Lex melempar pengecualian.

Operasi ini memerlukan izin utuk tindakan `lex:PutBot`. Untuk informasi selengkapnya, lihat Identity and Access Management untuk Amazon Lex.

Sintaks Permintaan

```
PUT /bots/name/versions/$LATEST HTTP/1.1
Content-type: application/json

{
   "abortStatement": {
      "messages": [
         {
            "content": "string",
            "contentType": "string",
            "groupNumber": number
         }
      ],
      "responseCard": "string"
   },
   "checksum": "string",
   "childDirected": boolean,
   "clarificationPrompt": {
      "maxAttempts": number,
      "messages": [
```

```
            {
               "content": "string",
               "contentType": "string",
               "groupNumber": number
            }
         ],
         "responseCard": "string"
      },
      "createVersion": boolean,
      "description": "string",
      "detectSentiment": boolean,
      "enableModelImprovements": boolean,
      "idleSessionTTLInSeconds": number,
      "intents": [
         {
            "intentName": "string",
            "intentVersion": "string"
         }
      ],
      "locale": "string",
      "nluIntentConfidenceThreshold": number,
      "processBehavior": "string",
      "tags": [
         {
            "key": "string",
            "value": "string"
         }
      ],
      "voiceId": "string"
}
```

Parameter Permintaan URI

Permintaan menggunakan parameter URI berikut.

name

Nama bot. Namanya tidak peka huruf besar/kecil.

Kendala Panjang: Panjang minimum 2. Panjang maksimum 50.

Pola: `^([A-Za-z]_?)+$`

Wajib: Ya

Isi Permintaan

Permintaan menerima data berikut dalam format JSON.

abortStatement

Ketika Amazon Lex tidak dapat memahami masukan pengguna dalam konteks, Amazon Lex mencoba memperoleh informasi beberapa kali. Setelah itu, Amazon Lex mengirimkan pesan yang ditentukan `abortStatement` ke pengguna, dan kemudian membatalkan percakapan. Untuk mengatur jumlah percobaan ulang, gunakan `valueElicitationPrompt` bidang untuk jenis slot.

Misalnya, dalam bot pemesanan pizza, Amazon Lex mungkin bertanya kepada pengguna "Jenis kerak apa yang Anda inginkan?" Jika respons pengguna bukan salah satu respons yang diharapkan (misalnya, "kerak tipis," piringan dalam, "dll.), Amazon Lex mencoba memperoleh respons yang benar beberapa kali lagi.

Misalnya, dalam aplikasi pemesanan pizza, `OrderPizza` mungkin salah satu maksudnya. Maksud ini mungkin memerlukan `CrustType` slot. Anda menentukan `valueElicitationPrompt` bidang saat Anda membuat `CrustType` slot.

Jika Anda telah menetapkan maksud fallback, pernyataan pembatalan tidak akan dikirim ke pengguna, maksud fallback digunakan sebagai gantinya. Untuk informasi selengkapnya, lihat AMAZON. FallbackIntent.

Tipe: Objek Statement

Diperlukan: Tidak

checksum

Mengidentifikasi revisi spesifik dari versi. `$LATEST`

Saat Anda membuat bot baru, biarkan `checksum` bidang kosong. Jika Anda menentukan checksum Anda mendapatkan `BadRequestException` pengecualian.

Saat Anda ingin memperbarui bot, atur `checksum` bidang ke checksum dari revisi versi terbaru. `$LATEST` Jika Anda tidak menentukan `checksum` bidang, atau jika checksum tidak cocok dengan `$LATEST` versi, Anda mendapatkan `PreconditionFailedException` pengecualian.

Tipe: String

Wajib: Tidak

childDirected

Untuk setiap bot Amazon Lex yang dibuat dengan Layanan Pembuatan Model Amazon Lex, Anda harus menentukan apakah penggunaan Amazon Lex Anda terkait dengan situs web, program, atau aplikasi lain yang diarahkan atau ditargetkan, secara keseluruhan atau sebagian, untuk anak-anak di bawah usia 13 tahun dan tunduk pada Undang-Undang Perlindungan Privasi Online Anak (COPPA) dengan menentukan `true` atau `false` di lapangan. `childDirected` Dengan menentukan **`true`** di **`childDirected`** lapangan, Anda mengonfirmasi bahwa penggunaan Amazon Lex Anda terkait dengan situs web, program, atau aplikasi lain yang diarahkan atau ditargetkan, secara keseluruhan atau sebagian, untuk anak-anak di bawah usia 13 tahun dan tunduk pada COPPA. Dengan menentukan `false` di `childDirected` lapangan, Anda mengonfirmasi bahwa penggunaan Amazon Lex Anda tidak terkait dengan situs web, program, atau aplikasi lain yang diarahkan atau ditargetkan, secara keseluruhan atau sebagian, untuk anak-anak di bawah usia 13 tahun dan tunduk pada COPPA. Anda tidak boleh menentukan nilai default untuk `childDirected` bidang yang tidak secara akurat mencerminkan apakah penggunaan Amazon Lex Anda terkait dengan situs web, program, atau aplikasi lain yang diarahkan atau ditargetkan, secara keseluruhan atau sebagian, untuk anak-anak di bawah usia 13 tahun dan tunduk pada COPPA.

Jika penggunaan Amazon Lex Anda terkait dengan situs web, program, atau aplikasi lain yang diarahkan secara keseluruhan atau sebagian, untuk anak-anak di bawah usia 13 tahun, Anda harus mendapatkan persetujuan orang tua yang dapat diverifikasi berdasarkan COPPA. Untuk informasi mengenai penggunaan Amazon Lex sehubungan dengan situs web, program, atau aplikasi lain yang diarahkan atau ditargetkan, secara keseluruhan atau sebagian, untuk anak-anak di bawah usia 13 tahun, lihat FAQ Amazon Lex.

Jenis: Boolean

Diperlukan: Ya

clarificationPrompt

Ketika Amazon Lex tidak memahami maksud pengguna, Amazon Lex menggunakan pesan ini untuk mendapatkan klarifikasi. Untuk menentukan berapa kali Amazon Lex harus mengulangi prompt klarifikasi, gunakan `maxAttempts` bidang. Jika Amazon Lex masih tidak mengerti, ia mengirimkan pesan di `abortStatement` lapangan.

Saat Anda membuat prompt klarifikasi, pastikan itu menyarankan respons yang benar dari pengguna. misalnya, untuk bot yang memesan pizza dan minuman, Anda dapat membuat prompt

klarifikasi ini: “Apa yang ingin Anda lakukan? Anda bisa mengatakan 'Pesan pizza' atau 'Pesan minuman'.”

Jika Anda telah menentukan maksud fallback, itu akan dipanggil jika prompt klarifikasi diulang berapa kali yang ditentukan di bidang. `maxAttempts` Untuk informasi selengkapnya, lihat AMAZON. FallbackIntent.

Jika Anda tidak menentukan prompt klarifikasi, saat runtime Amazon Lex akan mengembalikan pengecualian 400 Permintaan Buruk dalam tiga kasus:

- Follow-up prompt - Ketika pengguna merespons prompt tindak lanjut tetapi tidak memberikan maksud. Misalnya, sebagai tanggapan atas prompt tindak lanjut yang mengatakan “Apakah Anda ingin hal lain hari ini?” pengguna mengatakan “Ya.” Amazon Lex akan mengembalikan pengecualian 400 Bad Request karena tidak memiliki prompt klarifikasi untuk dikirim ke pengguna untuk mendapatkan maksud.
- Fungsi Lambda - Saat menggunakan fungsi Lambda, Anda mengembalikan jenis dialog. `ElicitIntent` Karena Amazon Lex tidak memiliki prompt klarifikasi untuk mendapatkan maksud dari pengguna, Amazon Lex mengembalikan pengecualian 400 Bad Request.
- PutSession operasi - Saat menggunakan `PutSession` operasi, Anda mengirim jenis `ElicitIntent` dialog. Karena Amazon Lex tidak memiliki prompt klarifikasi untuk mendapatkan maksud dari pengguna, Amazon Lex mengembalikan pengecualian 400 Bad Request.

Tipe: Objek Prompt

Diperlukan: Tidak

createVersion

Ketika diatur ke `true` versi bernomor baru dari bot dibuat. Ini sama dengan memanggil `CreateBotVersion` operasi. Jika Anda tidak menentukan`createVersion`, defaultnya adalah`false`.

Tipe: Boolean

Wajib: Tidak

description

Deskripsi bot.

Jenis: String

Batasan Panjang: Panjang minimum 0. Panjang maksimum 200.

Diperlukan: Tidak

detectSentiment

Saat disetel ke `true` pengguna, ucapan dikirim ke Amazon Comprehend untuk analisis sentimen. Jika Anda tidak menentukan`detectSentiment`, defaultnya adalah`false`.

Tipe: Boolean

Wajib: Tidak

enableModelImprovements

Setel `true` untuk mengaktifkan akses ke peningkatan pemahaman bahasa alami.

Ketika Anda mengatur `enableModelImprovements` parameter untuk `true` Anda dapat menggunakan `nluIntentConfidenceThreshold` parameter untuk mengkonfigurasi skor kepercayaan. Untuk informasi lebih lanjut, lihat Skor Keyakinan.

Anda hanya dapat mengatur `enableModelImprovements` parameter di Wilayah tertentu. Jika Anda mengatur parameternya`true`, bot Anda memiliki akses ke peningkatan akurasi.

Wilayah tempat Anda dapat mengatur `enableModelImprovements` parameter `false` untuk lokal en-US adalah:

- US East (N. Virginia) (us-east-1)
- US West (Oregon) (us-west-2)
- Asia Pasifik (Sydney) (ap-southeast-2)
- EU (Ireland) (eu-west-1)

Di Wilayah dan lokal lain, `enableModelImprovements` parameter diatur ke secara `true` default. Di Wilayah dan lokal ini, menyetel parameter untuk `false` melempar pengecualian. `ValidationException`

Tipe: Boolean

Wajib: Tidak

idleSessionTTLInSeconds

Waktu maksimum dalam hitungan detik Amazon Lex menyimpan data yang dikumpulkan dalam percakapan.

Sesi interaksi pengguna tetap aktif untuk jumlah waktu yang ditentukan. Jika tidak ada percakapan yang terjadi selama waktu ini, sesi berakhir dan Amazon Lex menghapus data apa pun yang diberikan sebelum batas waktu.

Misalnya, anggaplah pengguna memilih OrderPizza intent, tetapi teralihkan di tengah jalan dengan menempatkan pesanan. Jika pengguna tidak menyelesaikan pesanan dalam waktu yang ditentukan, Amazon Lex membuang informasi slot yang dikumpulkannya, dan pengguna harus memulai dari awal.

Jika Anda tidak menyertakan `idleSessionTTLInSeconds` elemen dalam permintaan `PutBot` operasi, Amazon Lex menggunakan nilai default. Ini juga berlaku jika permintaan menggantikan bot yang ada.

Default-nya adalah 300 detik (5 menit).

Jenis: Integer

Rentang yang Valid: Nilai minimum 60. Nilai maksimum 86400.

Diperlukan: Tidak

intents

Susunan objek `Intent`. Setiap intent mewakili perintah yang dapat diekspresikan pengguna. Misalnya, bot pemesanan pizza mungkin mendukung OrderPizza niat. Untuk informasi selengkapnya, lihat Amazon Lex: Cara Kerjanya.

Tipe: Array objek Intent

Diperlukan: Tidak

locale

Menentukan lokal target untuk bot. Maksud apa pun yang digunakan dalam bot harus kompatibel dengan lokal bot.

Default-nya adalah `en-US`.

Jenis: String

Nilai yang Valid: `de-DE | en-AU | en-GB | en-IN | en-US | es-419 | es-ES | es-US | fr-FR | fr-CA | it-IT | ja-JP | ko-KR`

Diperlukan: Ya

nluIntentConfidenceThreshold

Menentukan ambang batas tempat Amazon Lex akan menyisipkan `AMAZON.FallbackIntent`AMAZON.KendraSearchIntent,, atau keduanya saat mengembalikan maksud alternatif dalam PostTextrespons PostContentatau. `AMAZON.FallbackIntent`dan hanya `AMAZON.KendraSearchIntent` dimasukkan jika dikonfigurasi untuk bot.

Anda harus mengatur `enableModelImprovements` parameter `true` untuk menggunakan skor kepercayaan di wilayah berikut.

- US East (N. Virginia) (us-east-1)
- US West (Oregon) (us-west-2)
- Asia Pasifik (Sydney) (ap-southeast-2)
- EU (Ireland) (eu-west-1)

Di Wilayah lain, `enableModelImprovements` parameter diatur ke secara `true` default.

Misalnya, bot dikonfigurasi dengan ambang kepercayaan 0,80 dan. `AMAZON.FallbackIntent` Amazon Lex mengembalikan tiga maksud alternatif dengan skor kepercayaan berikut: Intenta (0,70), IntenTB (0,60), IntentC (0,50). Tanggapan dari `PostText` operasi tersebut adalah:

- AMAZON. FallbackIntent
- IntenTA
- IntentB
- IntentC

Tipe: Ganda

Rentang yang Valid: Nilai minimum 0. Nilai maksimum 1.

Diperlukan: Tidak

processBehavior

Jika Anda mengatur `processBehavior` elemen ke`BUILD`, Amazon Lex membangun bot sehingga dapat dijalankan. Jika Anda mengatur elemen ke `SAVE` Amazon Lex menyimpan bot, tetapi tidak membangunnya.

Jika Anda tidak menentukan nilai ini, nilai defaultnya adalah`BUILD`.

Jenis: String

Nilai yang Valid: `SAVE | BUILD`

Diperlukan: Tidak

tags

Daftar tag untuk ditambahkan ke bot. Anda hanya dapat menambahkan tag saat membuat bot, Anda tidak dapat menggunakan `PutBot` operasi untuk memperbarui tag pada bot. Untuk memperbarui tag, gunakan `TagResource` operasi.

Tipe: Array objek Tag

Anggota Array: Jumlah minimum 0 item. Jumlah maksimum 200 item.

Diperlukan: Tidak

voiceId

ID suara Amazon Polly yang Anda ingin Amazon Lex gunakan untuk interaksi suara dengan pengguna. Lokal yang dikonfigurasi untuk suara harus cocok dengan lokal bot. Untuk informasi selengkapnya, lihat Suara di Amazon Polly di Panduan Pengembang Amazon Polly.

Tipe: String

Wajib: Tidak

Sintaksis Respons

```
HTTP/1.1 200
Content-type: application/json

{
   "abortStatement": {
      "messages": [
         {
            "content": "string",
            "contentType": "string",
            "groupNumber": number
         }
      ],
      "responseCard": "string"
   },
   "checksum": "string",
   "childDirected": boolean,
```

```
    "clarificationPrompt": {
       "maxAttempts": number,
       "messages": [
          {
             "content": "string",
             "contentType": "string",
             "groupNumber": number
          }
       ],
       "responseCard": "string"
    },
    "createdDate": number,
    "createVersion": boolean,
    "description": "string",
    "detectSentiment": boolean,
    "enableModelImprovements": boolean,
    "failureReason": "string",
    "idleSessionTTLInSeconds": number,
    "intents": [
       {
          "intentName": "string",
          "intentVersion": "string"
       }
    ],
    "lastUpdatedDate": number,
    "locale": "string",
    "name": "string",
    "nluIntentConfidenceThreshold": number,
    "status": "string",
    "tags": [
       {
          "key": "string",
          "value": "string"
       }
    ],
    "version": "string",
    "voiceId": "string"
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Layanan mengembalikan data berikut dalam format JSON.

abortStatement

Pesan yang digunakan Amazon Lex untuk membatalkan percakapan. Untuk informasi selengkapnya, lihat PutBot.

Tipe: Objek Statement

checksum

Checksum bot yang Anda buat.

Jenis: String

childDirected

Untuk setiap bot Amazon Lex yang dibuat dengan Layanan Pembuatan Model Amazon Lex, Anda harus menentukan apakah penggunaan Amazon Lex Anda terkait dengan situs web, program, atau aplikasi lain yang diarahkan atau ditargetkan, secara keseluruhan atau sebagian, untuk anak-anak di bawah usia 13 tahun dan tunduk pada Undang-Undang Perlindungan Privasi Online Anak (COPPA) dengan menentukan `true` atau `false` di lapangan. `childDirected` Dengan menentukan **`true`** di **`childDirected`** lapangan, Anda mengonfirmasi bahwa penggunaan Amazon Lex Anda terkait dengan situs web, program, atau aplikasi lain yang diarahkan atau ditargetkan, secara keseluruhan atau sebagian, untuk anak-anak di bawah usia 13 tahun dan tunduk pada COPPA. Dengan menentukan `false` di `childDirected` lapangan, Anda mengonfirmasi bahwa penggunaan Amazon Lex Anda tidak terkait dengan situs web, program, atau aplikasi lain yang diarahkan atau ditargetkan, secara keseluruhan atau sebagian, untuk anak-anak di bawah usia 13 tahun dan tunduk pada COPPA. Anda tidak boleh menentukan nilai default untuk `childDirected` bidang yang tidak secara akurat mencerminkan apakah penggunaan Amazon Lex Anda terkait dengan situs web, program, atau aplikasi lain yang diarahkan atau ditargetkan, secara keseluruhan atau sebagian, untuk anak-anak di bawah usia 13 tahun dan tunduk pada COPPA.

Jika penggunaan Amazon Lex Anda terkait dengan situs web, program, atau aplikasi lain yang diarahkan secara keseluruhan atau sebagian, untuk anak-anak di bawah usia 13 tahun, Anda harus mendapatkan persetujuan orang tua yang dapat diverifikasi berdasarkan COPPA. Untuk informasi mengenai penggunaan Amazon Lex sehubungan dengan situs web, program, atau aplikasi lain yang diarahkan atau ditargetkan, secara keseluruhan atau sebagian, untuk anak-anak di bawah usia 13 tahun, lihat FAQ Amazon Lex.

Jenis: Boolean

clarificationPrompt

Permintaan yang digunakan Amazon Lex saat tidak memahami maksud pengguna. Untuk informasi selengkapnya, lihat PutBot.

Tipe: Objek Prompt

createdDate

Tanggal pembuatan bot.

Tipe: Timestamp

createVersion

`True`jika versi baru bot dibuat. Jika `createVersion` bidang tidak ditentukan dalam permintaan, `createVersion` bidang diatur ke false dalam respons.

Jenis: Boolean

description

Deskripsi bot.

Jenis: String

Batasan Panjang: Panjang minimum 0. Panjang maksimum 200.

detectSentiment

`true`jika bot dikonfigurasi untuk mengirim ucapan pengguna ke Amazon Comprehend untuk analisis sentimen. Jika `detectSentiment` bidang tidak ditentukan dalam permintaan, `detectSentiment` bidang ada `false` dalam respons.

Jenis: Boolean

enableModelImprovements

Menunjukkan apakah bot menggunakan peningkatan akurasi. `true`menunjukkan bahwa bot menggunakan perbaikan, jika tidak,`false`.

Jenis: Boolean

failureReason

Jika `status` ya`FAILED`, Amazon Lex memberikan alasan bahwa ia gagal membangun bot.

Jenis: String

idleSessionTTLInSeconds

Durasi maksimum waktu Amazon Lex menyimpan data yang dikumpulkan dalam percakapan. Untuk informasi selengkapnya, lihat PutBot.

Jenis: Integer

Rentang yang Valid: Nilai minimum 60. Nilai maksimum 86400.

intents

Susunan objek `Intent`. Untuk informasi selengkapnya, lihat PutBot.

Tipe: Array objek Intent

lastUpdatedDate

Tanggal bot diperbarui. Saat Anda membuat sumber daya, tanggal pembuatan dan tanggal terakhir diperbarui adalah sama.

Tipe: Timestamp

locale

Target lokal untuk bot.

Jenis: String

Nilai yang Valid: `de-DE | en-AU | en-GB | en-IN | en-US | es-419 | es-ES | es-US | fr-FR | fr-CA | it-IT | ja-JP | ko-KR`

name

Nama bot.

Jenis: String

Kendala Panjang: Panjang minimum 2. Panjang maksimum 50.

Pola: `^([A-Za-z]_?)+$`

nluIntentConfidenceThreshold

Skor yang menentukan di mana Amazon Lex menyisipkan `AMAZON.FallbackIntentAMAZON.KendraSearchIntent`,, atau keduanya

saat mengembalikan maksud alternatif dalam respons PostContentatau PostText. AMAZON.FallbackIntentdimasukkan jika skor kepercayaan untuk semua maksud di bawah nilai ini. AMAZON.KendraSearchIntenthanya dimasukkan jika dikonfigurasi untuk bot.

Tipe: Ganda

Rentang yang Valid: Nilai minimum 0. Nilai maksimum 1.

status

Saat Anda mengirim permintaan untuk membuat bot dengan processBehavior set toBUILD, Amazon Lex menyetel elemen status respons keBUILDING.

Dalam READY_BASIC_TESTING keadaan Anda dapat menguji bot dengan input pengguna yang sama persis dengan ucapan yang dikonfigurasi untuk maksud dan nilai bot dalam jenis slot.

Jika Amazon Lex tidak dapat membuat bot, Amazon Lex menetapkan status keFAILED. Amazon Lex mengembalikan alasan kegagalan elemen failureReason respons.

Saat Anda menyetel processBehavior keSAVE, Amazon Lex menyetel kode status keNOT BUILT.

Ketika bot dalam READY keadaan Anda dapat menguji dan menerbitkan bot.

Jenis: String

Nilai yang Valid: BUILDING | READY | READY_BASIC_TESTING | FAILED | NOT_BUILT

tags

Daftar tag yang terkait dengan bot.

Tipe: Array objek Tag

Anggota Array: Jumlah minimum 0 item. Jumlah maksimum 200 item.

version

Versi bot. Untuk bot baru, versinya selalu$LATEST.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum adalah 64.

Pola: \$LATEST|[0-9]+

voiceId

ID suara Amazon Polly yang digunakan Amazon Lex untuk interaksi suara dengan pengguna. Untuk informasi selengkapnya, lihat PutBot.

Tipe: String

Kesalahan

BadRequestException

Permintaan tidak terbentuk dengan baik. Misalnya, nilai tidak valid atau bidang wajib hilang. Periksa nilai bidang, dan coba lagi.

Kode Status HTTP: 400

ConflictException

Ada konflik memproses permintaan. Coba permintaan Anda lagi.

Kode Status HTTP: 409

InternalFailureException

Terjadi kesalahan internal Amazon Lex. Coba permintaan Anda lagi.

Kode Status HTTP: 500

LimitExceededException

Permintaan melebihi batas. Coba permintaan Anda lagi.

Kode Status HTTP: 429

PreconditionFailedException

Checksum sumber daya yang Anda coba ubah tidak cocok dengan checksum dalam permintaan. Periksa checksum sumber daya dan coba lagi.

Kode Status HTTP: 412

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

PutBotAlias

Layanan: Amazon Lex Model Building Service

Membuat alias untuk versi bot yang ditentukan atau menggantikan alias untuk bot yang ditentukan. Untuk mengubah versi bot yang ditunjuk alias, ganti alias. Untuk informasi selengkapnya tentang alias, lihatPembuatan Versi dan Alias.

Operasi ini memerlukan izin utuk tindakan `lex:PutBotAlias`.

Minta Sintaks

```
PUT /bots/botName/aliases/name HTTP/1.1
Content-type: application/json

{
   "botVersion": "string",
   "checksum": "string",
   "conversationLogs": {
      "iamRoleArn": "string",
      "logSettings": [
         {
            "destination": "string",
            "kmsKeyArn": "string",
            "logType": "string",
            "resourceArn": "string"
         }
      ]
   },
   "description": "string",
   "tags": [
      {
         "key": "string",
         "value": "string"
      }
   ]
}
```

Parameter Permintaan URI

Permintaan menggunakan parameter URI berikut.

botName

Nama bot.

Kendala Panjang: Panjang minimum 2. Panjang maksimum 50.

Pola: `^([A-Za-z]_?)+$`

Diperlukan: Ya

name

Nama alias. Namanya tidak peka huruf besar/kecil.

Batasan Panjang: Panjang minimum 1. Panjang maksimum 100.

Pola: `^([A-Za-z]_?)+$`

Wajib: Ya

Isi Permintaan

Permintaan menerima data berikut dalam format JSON.

botVersion

Versi bot.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum adalah 64.

Pola: `\$LATEST|[0-9]+`

Diperlukan: Ya

checksum

Mengidentifikasi revisi spesifik dari versi. `$LATEST`

Saat Anda membuat alias bot baru, biarkan `checksum` bidang kosong. Jika Anda menentukan checksum, Anda mendapatkan `BadRequestException` pengecualian.

Saat Anda ingin memperbarui alias bot, atur `checksum` bidang ke checksum dari revisi versi terbaru. `$LATEST` Jika Anda tidak menentukan `checksum` bidang, atau jika checksum tidak

cocok dengan $LATEST versi, Anda mendapatkan PreconditionFailedException pengecualian.

Tipe: String

Wajib: Tidak

conversationLogs

Pengaturan untuk log percakapan untuk alias.

Tipe: Objek ConversationLogsRequest

Diperlukan: Tidak

description

Deskripsi alias.

Tipe: String

Batasan Panjang: Panjang minimum 0. Panjang maksimum 200.

Diperlukan: Tidak

tags

Daftar tag untuk ditambahkan ke alias bot. Anda hanya dapat menambahkan tag ketika Anda membuat alias, Anda tidak dapat menggunakan PutBotAlias operasi untuk memperbarui tag pada alias bot. Untuk memperbarui tag, gunakan TagResource operasi.

Tipe: Array objek Tag

Anggota Array: Jumlah minimum 0 item. Jumlah maksimum 200 item.

Diperlukan: Tidak

Sintaksis Respons

```
HTTP/1.1 200
Content-type: application/json

{
   "botName": "string",
   "botVersion": "string",
```

```
    "checksum": "string",
    "conversationLogs": {
       "iamRoleArn": "string",
       "logSettings": [
          {
             "destination": "string",
             "kmsKeyArn": "string",
             "logType": "string",
             "resourceArn": "string",
             "resourcePrefix": "string"
          }
       ]
    },
    "createdDate": number,
    "description": "string",
    "lastUpdatedDate": number,
    "name": "string",
    "tags": [
       {
          "key": "string",
          "value": "string"
       }
    ]
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Layanan mengembalikan data berikut dalam format JSON.

botName

Nama bot yang ditunjuk alias.

Jenis: String

Kendala Panjang: Panjang minimum 2. Panjang maksimum 50.

Pola: `^([A-Za-z]_?)+$`

botVersion

Versi bot yang ditunjuk alias.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum adalah 64.

Pola: `\$LATEST|[0-9]+`

checksum

Checksum untuk versi alias saat ini.

Jenis: String

conversationLogs

Pengaturan yang menentukan bagaimana Amazon Lex menggunakan log percakapan untuk alias.

Tipe: Objek ConversationLogsResponse

createdDate

Tanggal alias bot dibuat.

Tipe: Timestamp

description

Deskripsi alias.

Tipe: String

Batasan Panjang: Panjang minimum 0. Panjang maksimum 200.

lastUpdatedDate

Tanggal alias bot diperbarui. Saat Anda membuat sumber daya, tanggal pembuatan dan tanggal pembaruan terakhir adalah sama.

Tipe: Timestamp

name

Nama alias.

Tipe: String

Panjang Batasan: Panjang minimum 1. Panjang maksimum 100.

Pola: `^([A-Za-z]_?)+$`

tags

Daftar tag yang terkait dengan bot.

Tipe: Array objek Tag

Anggota Array: Jumlah minimum 0 item. Jumlah maksimum 200 item.

Kesalahan

BadRequestException

Permintaan tidak terbentuk dengan baik. Misalnya, nilai tidak valid atau bidang wajib hilang. Periksa nilai bidang, dan coba lagi.

Kode Status HTTP: 400

ConflictException

Ada konflik yang memproses permintaan tersebut. Coba permintaan Anda lagi.

Kode Status HTTP: 409

InternalFailureException

Terjadi kesalahan internal Amazon Lex. Coba permintaan Anda lagi.

Kode Status HTTP: 500

LimitExceededException

Permintaan melebihi batas. Coba permintaan Anda lagi.

Kode Status HTTP: 429

PreconditionFailedException

Checksum sumber daya yang Anda coba ubah tidak cocok dengan checksum dalam permintaan. Periksa checksum sumber daya dan coba lagi.

Kode Status HTTP: 412

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

PutIntent

Layanan: Amazon Lex Model Building Service

Menciptakan maksud atau menggantikan maksud yang ada.

Untuk menentukan interaksi antara pengguna dan bot Anda, Anda menggunakan satu atau lebih maksud. Untuk bot pemesanan pizza, misalnya, Anda akan membuat `OrderPizza` maksud.

Untuk membuat intent atau mengganti intent yang ada, Anda harus memberikan yang berikut:

- Nama niat. Misalnya, `OrderPizza`.
- Sampel ucapan. Misalnya, “Bisakah saya memesan pizza, tolong.” dan “Saya ingin memesan pizza.”
- Informasi yang akan dikumpulkan. Anda menentukan jenis slot untuk informasi yang akan diminta bot Anda dari pengguna. Anda dapat menentukan jenis slot standar, seperti tanggal atau waktu, atau jenis slot khusus seperti ukuran dan kerak pizza.
- Bagaimana niatnya akan terpenuhi. Anda dapat menyediakan fungsi Lambda atau mengonfigurasi intent untuk mengembalikan informasi maksud ke aplikasi klien. Jika Anda menggunakan fungsi Lambda, ketika semua informasi maksud tersedia, Amazon Lex memanggil fungsi Lambda Anda. Jika Anda mengonfigurasi maksud Anda untuk mengembalikan informasi maksud ke aplikasi klien.

Anda dapat menentukan informasi opsional lainnya dalam permintaan, seperti:

- Permintaan konfirmasi untuk meminta pengguna mengonfirmasi maksud. Misalnya, “Haruskah saya memesan pizza Anda?”
- Pernyataan kesimpulan untuk dikirim ke pengguna setelah maksud terpenuhi. Misalnya, “Saya memesan pizza Anda.”
- Prompt tindak lanjut yang meminta pengguna untuk aktivitas tambahan. Misalnya, bertanya “Apakah Anda ingin memesan minuman dengan pizza Anda?”

Jika Anda menentukan nama intent yang ada untuk memperbarui intent, Amazon Lex mengganti nilai dalam `$LATEST` versi intent dengan nilai dalam permintaan. Amazon Lex menghapus bidang yang tidak Anda berikan dalam permintaan. Jika Anda tidak menentukan bidang yang diperlukan, Amazon Lex melempar pengecualian. Saat Anda memperbarui `$LATEST` versi intent, `status` bidang bot apa pun yang menggunakan `$LATEST` versi intent akan disetel ke. `NOT_BUILT`

Untuk informasi selengkapnya, lihat Amazon Lex: Cara Kerjanya.

Operasi ini memerlukan izin utuk tindakan `lex:PutIntent`.

Minta Sintaks

```
PUT /intents/name/versions/$LATEST HTTP/1.1
Content-type: application/json

{
   "checksum": "string",
   "conclusionStatement": {
      "messages": [
         {
            "content": "string",
            "contentType": "string",
            "groupNumber": number
         }
      ],
      "responseCard": "string"
   },
   "confirmationPrompt": {
      "maxAttempts": number,
      "messages": [
         {
            "content": "string",
            "contentType": "string",
            "groupNumber": number
         }
      ],
      "responseCard": "string"
   },
   "createVersion": boolean,
   "description": "string",
   "dialogCodeHook": {
      "messageVersion": "string",
      "uri": "string"
   },
   "followUpPrompt": {
      "prompt": {
         "maxAttempts": number,
         "messages": [
            {
               "content": "string",
               "contentType": "string",
               "groupNumber": number
```

```
            }
         ],
         "responseCard": "string"
      },
      "rejectionStatement": {
         "messages": [
            {
               "content": "string",
               "contentType": "string",
               "groupNumber": number
            }
         ],
         "responseCard": "string"
      }
   },
   "fulfillmentActivity": {
      "codeHook": {
         "messageVersion": "string",
         "uri": "string"
      },
      "type": "string"
   },
   "inputContexts": [
      {
         "name": "string"
      }
   ],
   "kendraConfiguration": {
      "kendraIndex": "string",
      "queryFilterString": "string",
      "role": "string"
   },
   "outputContexts": [
      {
         "name": "string",
         "timeToLiveInSeconds": number,
         "turnsToLive": number
      }
   ],
   "parentIntentSignature": "string",
   "rejectionStatement": {
      "messages": [
         {
            "content": "string",
```

```
            "contentType": "string",
            "groupNumber": number
         }
      ],
      "responseCard": "string"
   },
   "sampleUtterances": [ "string" ],
   "slots": [
      {
         "defaultValueSpec": {
            "defaultValueList": [
               {
                  "defaultValue": "string"
               }
            ]
         },
         "description": "string",
         "name": "string",
         "obfuscationSetting": "string",
         "priority": number,
         "responseCard": "string",
         "sampleUtterances": [ "string" ],
         "slotConstraint": "string",
         "slotType": "string",
         "slotTypeVersion": "string",
         "valueElicitationPrompt": {
            "maxAttempts": number,
            "messages": [
               {
                  "content": "string",
                  "contentType": "string",
                  "groupNumber": number
               }
            ],
            "responseCard": "string"
         }
      }
   ]
}
```

Parameter Permintaan URI

Permintaan menggunakan parameter URI berikut.

name

Nama niat. Namanya tidak peka huruf besar/kecil.

Nama tidak dapat cocok dengan nama intent bawaan, atau nama intent bawaan dengan "AMAZON." dihapus. Misalnya, karena ada intent bawaan yang dipanggil`AMAZON.HelpIntent`, Anda tidak dapat membuat maksud khusus yang dipanggil. `HelpIntent`

Untuk daftar intent bawaan, lihat Intent Bawaan Standar di Alexa Skills Kit.

Batasan Panjang: Panjang minimum 1. Panjang maksimum 100.

Pola: `^([A-Za-z]_?)+$`

Wajib: Ya

Isi Permintaan

Permintaan menerima data berikut dalam format JSON.

checksum

Mengidentifikasi revisi spesifik dari versi. `$LATEST`

Saat Anda membuat maksud baru, biarkan `checksum` bidang kosong. Jika Anda menentukan checksum Anda mendapatkan `BadRequestException` pengecualian.

Saat Anda ingin memperbarui maksud, setel `checksum` bidang ke checksum revisi terbaru versi. `$LATEST` Jika Anda tidak menentukan `checksum` bidang, atau jika checksum tidak cocok dengan `$LATEST` versi, Anda mendapatkan `PreconditionFailedException` pengecualian.

Tipe: String

Wajib: Tidak

conclusionStatement

Pernyataan yang Anda ingin Amazon Lex sampaikan kepada pengguna setelah maksud berhasil dipenuhi oleh fungsi Lambda.

Elemen ini relevan hanya jika Anda menyediakan fungsi Lambda di. `fulfillmentActivity` Jika Anda mengembalikan intent ke aplikasi klien, Anda tidak dapat menentukan elemen ini.

> **Note**
>
> Itu `followUpPrompt` dan `conclusionStatement` saling eksklusif. Anda hanya dapat menentukan satu.

Tipe: Objek Statement

Diperlukan: Tidak

confirmationPrompt

Meminta pengguna untuk mengonfirmasi maksud. Pertanyaan ini harus memiliki jawaban ya atau tidak.

Amazon Lex menggunakan prompt ini untuk memastikan bahwa pengguna mengakui bahwa intent siap untuk dipenuhi. Misalnya, dengan `OrderPizza` intent, Anda mungkin ingin mengonfirmasi bahwa pesanan sudah benar sebelum menempatkannya. Untuk maksud lain, seperti maksud yang hanya menanggapi pertanyaan pengguna, Anda mungkin tidak perlu meminta konfirmasi kepada pengguna sebelum memberikan informasi.

> **Note**
>
> Anda harus menyediakan keduanya `rejectionStatement` dan`confirmationPrompt`, atau tidak.

Tipe: Objek Prompt

Diperlukan: Tidak

createVersion

Ketika disetel ke `true` versi bernomor baru dari intent dibuat. Ini sama dengan memanggil `CreateIntentVersion` operasi. Jika Anda tidak menentukan`createVersion`, defaultnya adalah`false`.

Tipe: Boolean

Wajib: Tidak

description

Deskripsi niat.

Jenis: String

Batasan Panjang: Panjang minimum 0. Panjang maksimum 200.

Diperlukan: Tidak

dialogCodeHook

Menentukan fungsi Lambda untuk memanggil untuk setiap input pengguna. Anda dapat memanggil fungsi Lambda ini untuk mempersonalisasi interaksi pengguna.

Misalnya, bot Anda menentukan bahwa penggunanya adalah John. Fungsi Lambda Anda mungkin mengambil informasi John dari basis data backend dan mengisi beberapa nilai. Misalnya, jika Anda menemukan bahwa John tidak toleran terhadap gluten, Anda dapat mengatur slot maksud yang sesuai,`GlutenIntolerant`, ke true. Anda mungkin menemukan nomor telepon John dan mengatur atribut sesi yang sesuai.

Tipe: Objek CodeHook

Diperlukan: Tidak

followUpPrompt

Amazon Lex menggunakan prompt ini untuk meminta aktivitas tambahan setelah memenuhi maksud. Misalnya, setelah `OrderPizza` intent terpenuhi, Anda dapat meminta pengguna untuk memesan minuman.

Tindakan yang dilakukan Amazon Lex bergantung pada respons pengguna, sebagai berikut:

- Jika pengguna mengatakan “Ya” itu merespons dengan prompt klarifikasi yang dikonfigurasi untuk bot.
- jika pengguna mengatakan “Ya” dan melanjutkan dengan ucapan yang memicu maksud, ia memulai percakapan untuk maksud tersebut.
- Jika pengguna mengatakan “Tidak” itu merespons dengan pernyataan penolakan yang dikonfigurasi untuk prompt tindak lanjut.
- Jika tidak mengenali ucapannya, ia mengulangi prompt tindak lanjut lagi.

`followUpPrompt`Bidang dan `conclusionStatement` lapangan saling eksklusif. Anda hanya dapat menentukan satu.

Tipe: Objek FollowUpPrompt

Diperlukan: Tidak

fulfillmentActivity

Wajib. Menjelaskan bagaimana niat terpenuhi. Misalnya, setelah pengguna memberikan semua informasi untuk pesanan pizza, `fulfillmentActivity` mendefinisikan bagaimana bot melakukan pemesanan dengan toko pizza lokal.

Anda dapat mengonfigurasi Amazon Lex untuk mengembalikan semua informasi intent ke aplikasi klien, atau mengarahkannya untuk memanggil fungsi Lambda yang dapat memproses intent (misalnya, melakukan pemesanan dengan restoran pizza).

Tipe: Objek FulfillmentActivity

Diperlukan: Tidak

inputContexts

Larik `InputContext` objek yang mencantumkan konteks yang harus aktif untuk Amazon Lex untuk memilih maksud dalam percakapan dengan pengguna.

Tipe: Array objek InputContext

Anggota Array: Jumlah minimum 0 item. Jumlah maksimum 5 item.

Diperlukan: Tidak

kendraConfiguration

Informasi konfigurasi diperlukan untuk menggunakan `AMAZON.KendraSearchIntent` intent untuk menyambung ke indeks Amazon Kendra. Untuk informasi selengkapnya, lihat AMAZON.KendraSearchIntent.

Tipe: Objek KendraConfiguration

Diperlukan: Tidak

outputContexts

Array `OutputContext` objek yang mencantumkan konteks yang mengaktifkan intent saat intent terpenuhi.

Tipe: Array objek OutputContext

Anggota Array: Jumlah minimum 0 item. Jumlah maksimum 10 item.

Diperlukan: Tidak

parentIntentSignature

Pengidentifikasi unik untuk maksud bawaan yang menjadi dasar maksud ini. Untuk menemukan tanda tangan untuk intent, lihat Intent Bawaan Standar di Alexa Skills Kit.

Tipe: String

Wajib: Tidak

rejectionStatement

Ketika pengguna menjawab “tidak” untuk pertanyaan yang didefinisikan dalam`confirmationPrompt`, Amazon Lex merespons dengan pernyataan ini untuk mengakui bahwa maksud tersebut dibatalkan.

> **Note**
>
> Anda harus menyediakan keduanya `rejectionStatement` dan`confirmationPrompt`, atau tidak.

Tipe: Objek Statement

Diperlukan: Tidak

sampleUtterances

Array ucapan (string) yang mungkin dikatakan pengguna untuk memberi sinyal maksud. Misalnya, “Saya ingin {PizzaSize} pizza”, “Pesan {Kuantitas} {PizzaSize} pizza”.

Dalam setiap ucapan, nama slot dilampirkan dalam kurung kurawal.

Tipe: Array string.

Anggota Array: Jumlah minimum 0 item. Jumlah maksimum 1500 item.

Batasan Panjang: Panjang minimum 1. Panjang maksimum 200.

Diperlukan: Tidak

slots

Sebuah array slot niat. Saat runtime, Amazon Lex memperoleh nilai slot yang diperlukan dari pengguna menggunakan petunjuk yang ditentukan dalam slot. Untuk informasi selengkapnya, lihat Amazon Lex: Cara Kerjanya.

Tipe: Array objek Slot

Anggota Array: Jumlah minimum 0 item. Jumlah maksimum 100 item.

Diperlukan: Tidak

Sintaksis Respons

```
HTTP/1.1 200
Content-type: application/json

{
   "checksum": "string",
   "conclusionStatement": {
      "messages": [
         {
            "content": "string",
            "contentType": "string",
            "groupNumber": number
         }
      ],
      "responseCard": "string"
   },
   "confirmationPrompt": {
      "maxAttempts": number,
      "messages": [
         {
            "content": "string",
            "contentType": "string",
            "groupNumber": number
         }
      ],
      "responseCard": "string"
   },
   "createdDate": number,
   "createVersion": boolean,
   "description": "string",
```

```
    "dialogCodeHook": {
       "messageVersion": "string",
       "uri": "string"
    },
    "followUpPrompt": {
       "prompt": {
          "maxAttempts": number,
          "messages": [
             {
                "content": "string",
                "contentType": "string",
                "groupNumber": number
             }
          ],
          "responseCard": "string"
       },
       "rejectionStatement": {
          "messages": [
             {
                "content": "string",
                "contentType": "string",
                "groupNumber": number
             }
          ],
          "responseCard": "string"
       }
    },
    "fulfillmentActivity": {
       "codeHook": {
          "messageVersion": "string",
          "uri": "string"
       },
       "type": "string"
    },
    "inputContexts": [
       {
          "name": "string"
       }
    ],
    "kendraConfiguration": {
       "kendraIndex": "string",
       "queryFilterString": "string",
       "role": "string"
    },
```

```
    "lastUpdatedDate": number,
    "name": "string",
    "outputContexts": [
       {
          "name": "string",
          "timeToLiveInSeconds": number,
          "turnsToLive": number
       }
    ],
    "parentIntentSignature": "string",
    "rejectionStatement": {
       "messages": [
          {
             "content": "string",
             "contentType": "string",
             "groupNumber": number
          }
       ],
       "responseCard": "string"
    },
    "sampleUtterances": [ "string" ],
    "slots": [
       {
          "defaultValueSpec": {
             "defaultValueList": [
                {
                   "defaultValue": "string"
                }
             ]
          },
          "description": "string",
          "name": "string",
          "obfuscationSetting": "string",
          "priority": number,
          "responseCard": "string",
          "sampleUtterances": [ "string" ],
          "slotConstraint": "string",
          "slotType": "string",
          "slotTypeVersion": "string",
          "valueElicitationPrompt": {
             "maxAttempts": number,
             "messages": [
                {
                   "content": "string",
```

```
                "contentType": "string",
                "groupNumber": number
            }
        ],
        "responseCard": "string"
      }
    }
  ],
  "version": "string"
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Layanan mengembalikan data berikut dalam format JSON.

checksum

Checksum `$LATEST` versi maksud yang dibuat atau diperbarui.

Jenis: String

conclusionStatement

Setelah fungsi Lambda yang ditentukan dalam `fulfillmentActivity` intent memenuhi intent, Amazon Lex menyampaikan pernyataan ini kepada pengguna.

Tipe: Objek Statement

confirmationPrompt

Jika didefinisikan dalam maksud, Amazon Lex meminta pengguna untuk mengonfirmasi maksud sebelum memenuhinya.

Tipe: Objek Prompt

createdDate

Tanggal dimana niat itu dibuat.

Tipe: Timestamp

createVersion

`True`jika versi baru dari intent telah dibuat. Jika `createVersion` bidang tidak ditentukan dalam permintaan, `createVersion` bidang diatur ke false dalam respons.

Jenis: Boolean

description

Deskripsi niat.

Jenis: String

Batasan Panjang: Panjang minimum 0. Panjang maksimum 200.

dialogCodeHook

Jika didefinisikan dalam intent, Amazon Lex memanggil fungsi Lambda ini untuk setiap input pengguna.

Tipe: Objek CodeHook

followUpPrompt

Jika didefinisikan dalam intent, Amazon Lex menggunakan prompt ini untuk meminta aktivitas pengguna tambahan setelah intent terpenuhi.

Tipe: Objek FollowUpPrompt

fulfillmentActivity

Jika didefinisikan dalam intent, Amazon Lex memanggil fungsi Lambda ini untuk memenuhi intent setelah pengguna memberikan semua informasi yang diperlukan oleh intent.

Tipe: Objek FulfillmentActivity

inputContexts

Larik `InputContext` objek yang mencantumkan konteks yang harus aktif untuk Amazon Lex untuk memilih maksud dalam percakapan dengan pengguna.

Tipe: Array objek InputContext

Anggota Array: Jumlah minimum 0 item. Jumlah maksimum 5 item.

kendraConfiguration

Informasi konfigurasi, jika ada, diperlukan untuk terhubung ke indeks Amazon Kendra dan menggunakan intent. `AMAZON.KendraSearchIntent`

Tipe: Objek KendraConfiguration

lastUpdatedDate

Tanggal dimana intent diperbarui. Saat Anda membuat sumber daya, tanggal pembuatan dan tanggal pembaruan terakhir adalah sama.

Tipe: Timestamp

name

Nama niat.

Jenis: String

Panjang Batasan: Panjang minimum 1. Panjang maksimum 100.

Pola: `^([A-Za-z]_?)+$`

outputContexts

Array `OutputContext` objek yang mencantumkan konteks yang mengaktifkan intent saat intent terpenuhi.

Tipe: Array objek OutputContext

Anggota Array: Jumlah minimum 0 item. Jumlah maksimum 10 item.

parentIntentSignature

Pengidentifikasi unik untuk maksud bawaan yang menjadi dasar maksud ini.

Jenis: String

rejectionStatement

Jika pengguna menjawab “tidak” untuk pertanyaan yang didefinisikan di `confirmationPrompt` Amazon Lex merespons dengan pernyataan ini untuk mengakui bahwa maksud tersebut dibatalkan.

Tipe: Objek Statement

sampleUtterances

Array contoh ucapan yang dikonfigurasi untuk maksud.

Tipe: Array string.

Anggota Array: Jumlah minimum 0 item. Jumlah maksimum 1500 item.

Batasan Panjang: Panjang minimum 1. Panjang maksimum 200.

slots

Sebuah array slot intent yang dikonfigurasi untuk maksud.

Tipe: Array objek Slot

Anggota Array: Jumlah minimum 0 item. Jumlah maksimum 100 item.

version

Versi maksudnya. Untuk maksud baru, versinya selalu`$LATEST`.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum adalah 64.

Pola: `\$LATEST|[0-9]+`

Kesalahan

BadRequestException

Permintaan tidak terbentuk dengan baik. Misalnya, nilai tidak valid atau bidang wajib hilang. Periksa nilai bidang, dan coba lagi.

Kode Status HTTP: 400

ConflictException

Ada konflik yang memproses permintaan tersebut. Coba permintaan Anda lagi.

Kode Status HTTP: 409

InternalFailureException

Terjadi kesalahan internal Amazon Lex. Coba permintaan Anda lagi.

Kode Status HTTP: 500

LimitExceededException

Permintaan melebihi batas. Coba permintaan Anda lagi.

Kode Status HTTP: 429

PreconditionFailedException

Checksum sumber daya yang Anda coba ubah tidak cocok dengan checksum dalam permintaan. Periksa checksum sumber daya dan coba lagi.

Kode Status HTTP: 412

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

PutSlotType

Layanan: Amazon Lex Model Building Service

Membuat jenis slot khusus atau menggantikan jenis slot kustom yang ada.

Untuk membuat jenis slot khusus, tentukan nama untuk jenis slot dan satu set nilai enumerasi, yang merupakan nilai yang dapat diasumsikan oleh slot jenis ini. Untuk informasi selengkapnya, lihat Amazon Lex: Cara Kerjanya.

Jika Anda menentukan nama jenis slot yang ada, bidang dalam permintaan menggantikan nilai yang ada dalam `$LATEST` versi jenis slot. Amazon Lex menghapus bidang yang tidak Anda berikan dalam permintaan. Jika Anda tidak menentukan bidang wajib, Amazon Lex melempar pengecualian. Saat Anda memperbarui `$LATEST` versi jenis slot, jika bot menggunakan `$LATEST` versi intent yang berisi jenis slot, `status` bidang bot disetel ke`NOT_BUILT`.

Operasi ini memerlukan izin utuk tindakan `lex:PutSlotType`.

Minta Sintaks

```
PUT /slottypes/name/versions/$LATEST HTTP/1.1
Content-type: application/json

{
   "checksum": "string",
   "createVersion": boolean,
   "description": "string",
   "enumerationValues": [
      {
         "synonyms": [ "string" ],
         "value": "string"
      }
   ],
   "parentSlotTypeSignature": "string",
   "slotTypeConfigurations": [
      {
         "regexConfiguration": {
            "pattern": "string"
         }
      }
   ],
   "valueSelectionStrategy": "string"
}
```

Parameter Permintaan URI

Permintaan menggunakan parameter URI berikut.

name

Nama jenis slot. Namanya tidak peka huruf besar/kecil.

Nama tidak dapat cocok dengan nama tipe slot bawaan, atau nama tipe slot bawaan dengan "AMAZON." dihapus. Misalnya, karena ada jenis slot bawaan yang disebut`AMAZON.DATE`, Anda tidak dapat membuat jenis slot khusus yang disebut`DATE`.

Untuk daftar jenis slot bawaan, lihat Referensi Jenis Slot di Alexa Skills Kit.

Batasan Panjang: Panjang minimum 1. Panjang maksimum 100.

Pola: `^([A-Za-z]_?)+$`

Wajib: Ya

Isi Permintaan

Permintaan menerima data berikut dalam format JSON.

checksum

Mengidentifikasi revisi spesifik dari versi. `$LATEST`

Saat Anda membuat jenis slot baru, biarkan `checksum` bidang kosong. Jika Anda menentukan checksum, Anda mendapatkan `BadRequestException` pengecualian.

Saat Anda ingin memperbarui jenis slot, atur `checksum` bidang ke checksum dari revisi versi terbaru. `$LATEST` Jika Anda tidak menentukan `checksum` bidang, atau jika checksum tidak cocok dengan `$LATEST` versi, Anda mendapatkan `PreconditionFailedException` pengecualian.

Tipe: String

Wajib: Tidak

createVersion

Ketika diatur ke `true` versi bernomor baru dari jenis slot dibuat. Ini sama dengan memanggil `CreateSlotTypeVersion` operasi. Jika Anda tidak menentukan`createVersion`, defaultnya adalah`false`.

Tipe: Boolean

Wajib: Tidak

description

Deskripsi jenis slot.

Jenis: String

Batasan Panjang: Panjang minimum 0. Panjang maksimum 200.

Diperlukan: Tidak

enumerationValues

Daftar `EnumerationValue` objek yang mendefinisikan nilai yang dapat diambil oleh jenis slot. Setiap nilai dapat memiliki daftar`synonyms`, yang merupakan nilai tambahan yang membantu melatih model pembelajaran mesin tentang nilai-nilai yang diselesaikan untuk slot.

Jenis slot ekspresi reguler tidak memerlukan nilai enumerasi. Semua jenis slot lainnya memerlukan daftar nilai enumerasi.

Ketika Amazon Lex menyelesaikan nilai slot, itu menghasilkan daftar resolusi yang berisi hingga lima nilai yang mungkin untuk slot. Jika Anda menggunakan fungsi Lambda, daftar resolusi ini diteruskan ke fungsi. Jika Anda tidak menggunakan fungsi Lambda, Anda dapat memilih untuk mengembalikan nilai yang dimasukkan pengguna atau nilai pertama dalam daftar resolusi sebagai nilai slot. `valueSelectionStrategy`Bidang menunjukkan opsi untuk digunakan.

Tipe: Array objek EnumerationValue

Anggota Array: Jumlah minimum 0 item. Jumlah maksimum 10000 item.

Diperlukan: Tidak

parentSlotTypeSignature

Tipe slot built-in digunakan sebagai induk dari jenis slot. Saat Anda menentukan jenis slot induk, jenis slot baru memiliki semua konfigurasi yang sama dengan induknya.

Hanya `AMAZON.AlphaNumeric` didukung.

Jenis: String

Panjang Batasan: Panjang minimum 1. Panjang maksimum 100.

Pola: `^((AMAZON\.)_?|[A-Za-z]_?)+`

Diperlukan: Tidak

slotTypeConfigurations

Informasi konfigurasi yang memperluas tipe slot bawaan induk. Konfigurasi ditambahkan ke pengaturan untuk jenis slot induk.

Tipe: Array objek SlotTypeConfiguration

Anggota Array: Jumlah minimum 0 item. Jumlah maksimum 10 item.

Diperlukan: Tidak

valueSelectionStrategy

Menentukan strategi resolusi slot yang digunakan Amazon Lex untuk mengembalikan nilai jenis slot. Bidang dapat diatur ke salah satu nilai berikut:

- `ORIGINAL_VALUE`- Mengembalikan nilai yang dimasukkan oleh pengguna, jika nilai pengguna mirip dengan nilai slot.
- `TOP_RESOLUTION`- Jika ada daftar resolusi untuk slot, kembalikan nilai pertama dalam daftar resolusi sebagai nilai jenis slot. Jika tidak ada daftar resolusi, null dikembalikan.

Jika Anda tidak menentukan`valueSelectionStrategy`, defaultnya adalah`ORIGINAL_VALUE`.

Jenis: String

Nilai yang Valid: `ORIGINAL_VALUE | TOP_RESOLUTION`

Diperlukan: Tidak

Sintaksis Respons

```
HTTP/1.1 200
```

```
Content-type: application/json

{
   "checksum": "string",
   "createdDate": number,
   "createVersion": boolean,
   "description": "string",
   "enumerationValues": [
      {
         "synonyms": [ "string" ],
         "value": "string"
      }
   ],
   "lastUpdatedDate": number,
   "name": "string",
   "parentSlotTypeSignature": "string",
   "slotTypeConfigurations": [
      {
         "regexConfiguration": {
            "pattern": "string"
         }
      }
   ],
   "valueSelectionStrategy": "string",
   "version": "string"
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Layanan mengembalikan data berikut dalam format JSON.

checksum

Checksum dari $LATEST versi jenis slot.

Jenis: String

createdDate

Tanggal jenis slot dibuat.

Tipe: Timestamp

createVersion

`True`jika versi baru dari jenis slot dibuat. Jika `createVersion` bidang tidak ditentukan dalam permintaan, `createVersion` bidang diatur ke false dalam respons.

Jenis: Boolean

description

Deskripsi jenis slot.

Jenis: String

Batasan Panjang: Panjang minimum 0. Panjang maksimum 200.

enumerationValues

Daftar `EnumerationValue` objek yang mendefinisikan nilai yang dapat diambil oleh jenis slot.

Tipe: Array objek EnumerationValue

Anggota Array: Jumlah minimum 0 item. Jumlah maksimum 10000 item.

lastUpdatedDate

Tanggal bahwa jenis slot diperbarui. Saat Anda membuat jenis slot, tanggal pembuatan dan tanggal pembaruan terakhir adalah sama.

Tipe: Timestamp

name

Nama jenis slot.

Jenis: String

Panjang Batasan: Panjang minimum 1. Panjang maksimum 100.

Pola: `^([A-Za-z]_?)+$`

parentSlotTypeSignature

Tipe slot built-in digunakan sebagai induk dari jenis slot.

Jenis: String

Panjang Batasan: Panjang minimum 1. Panjang maksimum 100.

Pola: `^((AMAZON\.)_?|[A-Za-z]_?)+`

slotTypeConfigurations

Informasi konfigurasi yang memperluas tipe slot bawaan induk.

Tipe: Array objek SlotTypeConfiguration

Anggota Array: Jumlah minimum 0 item. Jumlah maksimum 10 item.

valueSelectionStrategy

Strategi resolusi slot yang digunakan Amazon Lex untuk menentukan nilai slot. Untuk informasi selengkapnya, lihat PutSlotType.

Tipe: String

Nilai yang Valid: `ORIGINAL_VALUE | TOP_RESOLUTION`

version

Versi dari jenis slot. Untuk jenis slot baru, versinya selalu`$LATEST`.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum adalah 64.

Pola: `\$LATEST|[0-9]+`

Kesalahan

BadRequestException

Permintaan tidak terbentuk dengan baik. Misalnya, nilai tidak valid atau bidang wajib hilang. Periksa nilai bidang, dan coba lagi.

Kode Status HTTP: 400

ConflictException

Ada konflik yang memproses permintaan tersebut. Coba permintaan Anda lagi.

Kode Status HTTP: 409

InternalFailureException

Terjadi kesalahan internal Amazon Lex. Coba permintaan Anda lagi.

Kode Status HTTP: 500

LimitExceededException

Permintaan melebihi batas. Coba permintaan Anda lagi.

Kode Status HTTP: 429

PreconditionFailedException

Checksum sumber daya yang Anda coba ubah tidak cocok dengan checksum dalam permintaan. Periksa checksum sumber daya dan coba lagi.

Kode Status HTTP: 412

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

StartImport

Layanan: Amazon Lex Model Building Service

Memulai pekerjaan untuk mengimpor sumber daya ke Amazon Lex.

Minta Sintaks

```
POST /imports/ HTTP/1.1
Content-type: application/json

{
   "mergeStrategy": "string",
   "payload": blob,
   "resourceType": "string",
   "tags": [
      {
         "key": "string",
         "value": "string"
      }
   ]
}
```

Parameter Permintaan URI

Permintaan tidak menggunakan parameter URI apa pun.

Isi Permintaan

Permintaan menerima data berikut dalam format JSON.

mergeStrategy

Menentukan tindakan yang harus dilakukan `StartImport` operasi ketika ada sumber daya yang ada dengan nama yang sama.

- FAIL_ON_CONFLICT - Operasi impor dihentikan pada konflik pertama antara sumber daya dalam file impor dan sumber daya yang ada. Nama sumber daya yang menyebabkan konflik ada di `failureReason` bidang respons terhadap `GetImport` operasi.

  OVERWRITE_LATEST - Operasi impor berlangsung bahkan jika ada konflik dengan sumber daya yang ada. Versi $LASTEST dari sumber daya yang ada ditimpa dengan data dari file impor.

Tipe: String

Nilai yang Valid: `OVERWRITE_LATEST | FAIL_ON_CONFLICT`

Diperlukan: Ya

payload

Arsip zip dalam format biner. Arsip harus berisi satu file, file JSON yang berisi sumber daya untuk diimpor. Sumber daya harus cocok dengan jenis yang ditentukan di `resourceType` bidang.

Tipe: Objek data biner dienkode Base64

Diperlukan: Ya

resourceType

Menentukan jenis sumber daya untuk mengekspor. Setiap sumber daya juga mengekspor sumber daya apa pun yang bergantung padanya.

- Bot mengekspor maksud yang bergantung.
- Maksud mengekspor jenis slot yang bergantung.

Tipe: String

Nilai yang Valid: `BOT | INTENT | SLOT_TYPE`

Diperlukan: Ya

tags

Daftar tag untuk ditambahkan ke bot yang diimpor. Anda hanya dapat menambahkan tag saat mengimpor bot, Anda tidak dapat menambahkan tag ke maksud atau jenis slot.

Tipe: Array objek Tag

Anggota Array: Jumlah minimum 0 item. Jumlah maksimum 200 item.

Diperlukan: Tidak

Sintaksis Respons

```
HTTP/1.1 201
Content-type: application/json
```

```
{
   "createdDate": number,
   "importId": "string",
   "importStatus": "string",
   "mergeStrategy": "string",
   "name": "string",
   "resourceType": "string",
   "tags": [
      {
         "key": "string",
         "value": "string"
      }
   ]
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respon HTTP 201.

Layanan mengembalikan data berikut dalam format JSON.

createdDate

Stempel waktu untuk tanggal dan waktu pekerjaan impor diminta.

Tipe: Timestamp

importId

Pengidentifikasi untuk pekerjaan impor tertentu.

Tipe: String

importStatus

Status pekerjaan impor. Jika statusnya`FAILED`, Anda bisa mendapatkan alasan kegagalan menggunakan `GetImport` operasi.

Tipe: String

Nilai yang Valid: `IN_PROGRESS | COMPLETE | FAILED`

mergeStrategy

Tindakan yang harus diambil ketika ada konflik penggabungan.

Tipe: String

Nilai yang Valid: `OVERWRITE_LATEST | FAIL_ON_CONFLICT`

name

Nama yang diberikan untuk pekerjaan impor.

Tipe: String

Panjang Batasan: Panjang minimum 1. Panjang maksimum 100.

Pola: `[a-zA-Z_]+`

resourceType

Jenis sumber daya untuk diimpor.

Tipe: String

Nilai yang Valid: `BOT | INTENT | SLOT_TYPE`

tags

Daftar tag yang ditambahkan ke bot yang diimpor.

Tipe: Array objek Tag

Anggota Array: Jumlah minimum 0 item. Jumlah maksimum 200 item.

Galat

BadRequestException

Permintaan tidak terbentuk dengan baik. Misalnya, nilai tidak valid atau bidang wajib hilang. Periksa nilai bidang, dan coba lagi.

Kode Status HTTP: 400

InternalFailureException

Terjadi kesalahan internal Amazon Lex. Coba permintaan Anda lagi.

Kode Status HTTP: 500

LimitExceededException

Permintaan melebihi batas. Coba permintaan Anda lagi.

Kode Status HTTP: 429

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWSAntarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go.
- AWSSDK for Java V2
- AWSSDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

StartMigration

Layanan: Amazon Lex Model Building Service

Mulai memigrasikan bot dari Amazon Lex V1 ke Amazon Lex V2. Migrasikan bot Anda saat Anda ingin memanfaatkan fitur-fitur baru Amazon Lex V2.

Untuk informasi selengkapnya, lihat Memigrasi bot di panduan pengembang Amazon Lex.

Minta Sintaks

```
POST /migrations HTTP/1.1
Content-type: application/json

{
   "migrationStrategy": "string",
   "v1BotName": "string",
   "v1BotVersion": "string",
   "v2BotName": "string",
   "v2BotRole": "string"
}
```

Parameter Permintaan URI

Permintaan tidak menggunakan parameter URI apa pun.

Isi Permintaan

Permintaan menerima data berikut dalam format JSON.

migrationStrategy

Strategi yang digunakan untuk melakukan migrasi.

- `CREATE_NEW`- Membuat bot Amazon Lex V2 baru dan memigrasikan bot Amazon Lex V1 ke bot baru.
- `UPDATE_EXISTING`- Menimpa metadata bot Amazon Lex V2 yang ada dan lokal yang sedang dimigrasikan. Itu tidak mengubah lokal lain di bot Amazon Lex V2. Jika lokal tidak ada, lokal baru dibuat di bot Amazon Lex V2.

Tipe: String

Nilai yang Valid: `CREATE_NEW | UPDATE_EXISTING`

Diperlukan: Ya

v1BotName

Nama bot Amazon Lex V1 yang Anda migrasikan ke Amazon Lex V2.

Tipe: String

Kendala Panjang: Panjang minimum 2. Panjang maksimum 50.

Pola: `^([A-Za-z]_?)+$`

Diperlukan: Ya

v1BotVersion

Versi bot untuk bermigrasi ke Amazon Lex V2. Anda dapat memigrasikan `$LATEST` versi serta versi bernomor apa pun.

Tipe: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum adalah 64.

Pola: `\$LATEST|[0-9]+`

Diperlukan: Ya

v2BotName

Nama bot Amazon Lex V2 tempat Anda memigrasikan bot Amazon Lex V1.

- Jika bot Amazon Lex V2 tidak ada, Anda harus menggunakan strategi `CREATE_NEW` migrasi.
- Jika bot Amazon Lex V2 ada, Anda harus menggunakan strategi `UPDATE_EXISTING` migrasi untuk mengubah konten bot Amazon Lex V2.

Tipe: String

Panjang Batasan: Panjang minimum 1. Panjang maksimum 100.

Pola: `^([0-9a-zA-Z][_-]?)+$`

Diperlukan: Ya

v2BotRole

Peran IAM yang digunakan Amazon Lex untuk menjalankan bot Amazon Lex V2.

Tipe: String

Batasan Panjang: Panjang minimum 20. Panjang maksimum 2048.

Pola: `^arn:[\w\-]+:iam::[\d]{12}:role/.+$`

Diperlukan: Ya

Sintaksis Respons

```
HTTP/1.1 202
Content-type: application/json

{
   "migrationId": "string",
   "migrationStrategy": "string",
   "migrationTimestamp": number,
   "v1BotLocale": "string",
   "v1BotName": "string",
   "v1BotVersion": "string",
   "v2BotId": "string",
   "v2BotRole": "string"
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 202.

Layanan mengembalikan data berikut dalam format JSON.

migrationId

Pengenal unik yang ditetapkan Amazon Lex untuk migrasi.

Tipe: String

Kendala Panjang: Panjang tetap 10.

Pola: `^[0-9a-zA-Z]+$`

migrationStrategy

Strategi yang digunakan untuk melakukan migrasi.

Tipe: String

Nilai yang Valid: `CREATE_NEW | UPDATE_EXISTING`

migrationTimestamp

Tanggal dan waktu migrasi dimulai.

Tipe: Timestamp

v1BotLocale

Lokal yang digunakan untuk bot Amazon Lex V1.

Tipe: String

Nilai yang Valid: `de-DE | en-AU | en-GB | en-IN | en-US | es-419 | es-ES | es-US | fr-FR | fr-CA | it-IT | ja-JP | ko-KR`

v1BotName

Nama bot Amazon Lex V1 yang Anda migrasikan ke Amazon Lex V2.

Tipe: String

Kendala Panjang: Panjang minimum 2. Panjang maksimum 50.

Pola: `^([A-Za-z]_?)+$`

v1BotVersion

Versi bot untuk bermigrasi ke Amazon Lex V2.

Tipe: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum adalah 64.

Pola: `\$LATEST|[0-9]+`

v2BotId

Pengidentifikasi unik untuk bot Amazon Lex V2.

Tipe: String

Kendala Panjang: Panjang tetap 10.

Pola: `^[0-9a-zA-Z]+$`

v2BotRole

Peran IAM yang digunakan Amazon Lex untuk menjalankan bot Amazon Lex V2.

Tipe: String

Batasan Panjang: Panjang minimum 20. Panjang maksimum 2048.

Pola: `^arn:[\w\-]+:iam::[\d]{12}:role/.+$`

Galat

AccessDeniedException

Pengguna atau peran IAM Anda tidak memiliki izin untuk memanggil Amazon Lex V2 API yang diperlukan untuk memigrasi bot Anda.

Kode Status HTTP: 403

BadRequestException

Permintaan tidak terbentuk dengan baik. Misalnya, nilai tidak valid atau bidang wajib hilang. Periksa nilai bidang, dan coba lagi.

Kode Status HTTP: 400

InternalFailureException

Terjadi kesalahan internal Amazon Lex. Coba permintaan Anda lagi.

Kode Status HTTP: 500

LimitExceededException

Permintaan melebihi batas. Coba permintaan Anda lagi.

Kode Status HTTP: 429

NotFoundException

Sumber daya yang ditentukan dalam permintaan tidak ditemukan. Periksa sumber daya dan coba lagi.

Kode Status HTTP: 404

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWSAntarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go.
- AWSSDK for Java V2
- AWSSDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK for Python
- AWS SDK for Ruby V3

TagResource

Layanan: Amazon Lex Model Building Service

Menambahkan tag yang ditentukan ke sumber daya yang ditentukan. Jika kunci tag sudah ada, nilai yang ada diganti dengan nilai baru.

Minta Sintaks

```
POST /tags/resourceArn HTTP/1.1
Content-type: application/json

{
   "tags": [
      {
         "key": "string",
         "value": "string"
      }
   ]
}
```

Parameter Permintaan URI

Permintaan menggunakan parameter URI berikut.

resourceArn

Nama Sumber Daya Amazon (ARN) dari bot, alias bot, atau saluran bot untuk diberi tag.

Batasan Panjang: Panjang minimum 1. Panjang maksimum 1011.

Diperlukan: Ya

Isi Permintaan

Permintaan menerima data berikut dalam format JSON.

tags

Daftar kunci tag untuk ditambahkan ke sumber daya. Jika kunci tag sudah ada, nilai yang ada diganti dengan nilai baru.

Tipe: Array objek Tag

Anggota Array: Jumlah minimum 0 item. Jumlah maksimum 200 item.

Diperlukan: Ya

Sintaksis Respons

```
HTTP/1.1 204
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 204 dengan isi HTTP kosong.

Kesalahan

BadRequestException

Permintaan tidak terbentuk dengan baik. Misalnya, nilai tidak valid atau bidang wajib hilang. Periksa nilai bidang, dan coba lagi.

Kode Status HTTP: 400

ConflictException

Ada konflik memproses permintaan. Coba permintaan Anda lagi.

Kode Status HTTP: 409

InternalFailureException

Terjadi kesalahan internal Amazon Lex. Coba permintaan Anda lagi.

Kode Status HTTP: 500

LimitExceededException

Permintaan melebihi batas. Coba permintaan Anda lagi.

Kode Status HTTP: 429

NotFoundException

Sumber daya yang ditentukan dalam permintaan tidak ditemukan. Periksa sumber daya dan coba lagi.

Kode Status HTTP: 404

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

UntagResource

Layanan: Amazon Lex Model Building Service

Menghapus tag dari bot, bot alias atau saluran bot.

Minta Sintaks

```
DELETE /tags/resourceArn?tagKeys=tagKeys HTTP/1.1
```

Parameter Permintaan URI

Permintaan menggunakan parameter URI berikut.

resourceArn

Nama Sumber Daya Amazon (ARN) dari sumber daya untuk menghapus tag dari.

Batasan Panjang: Panjang minimum 1. Panjang maksimum 1011.

Diperlukan: Ya

tagKeys

Daftar kunci tag untuk dihapus dari sumber daya. Jika tidak ada pada sumber daya, kunci tanda akan diabaikan.

Anggota Array: Jumlah minimum 0 item. Jumlah maksimum 200 item.

Batasan Panjang: Panjang minimum 1. Panjang maksimum 128.

Diperlukan: Ya

Isi Permintaan

Permintaan tidak memiliki isi permintaan.

Sintaks Respons

```
HTTP/1.1 204
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 204 dengan isi HTTP kosong.

Kesalahan

BadRequestException

Permintaan tidak terbentuk dengan baik. Misalnya, nilai tidak valid atau bidang wajib hilang. Periksa nilai bidang, dan coba lagi.

Kode Status HTTP: 400

ConflictException

Ada konflik memproses permintaan. Coba permintaan Anda lagi.

Kode Status HTTP: 409

InternalFailureException

Terjadi kesalahan internal Amazon Lex. Coba permintaan Anda lagi.

Kode Status HTTP: 500

LimitExceededException

Permintaan melebihi batas. Coba permintaan Anda lagi.

Kode Status HTTP: 429

NotFoundException

Sumber daya yang ditentukan dalam permintaan tidak ditemukan. Periksa sumber daya dan coba lagi.

Kode Status HTTP: 404

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go

- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

Amazon Lex Runtime

Tindakan berikut didukung oleh Amazon Lex Runtime Service:

- DeleteSession
- GetSession
- PostContent
- PostText
- PutSession

DeleteSession

Layanan: Amazon Lex Runtime Service

Menghapus informasi sesi untuk bot, alias, dan ID pengguna tertentu.

Minta Sintaks

```
DELETE /bot/botName/alias/botAlias/user/userId/session HTTP/1.1
```

Parameter Permintaan URI

Permintaan menggunakan parameter URI berikut.

botAlias

Alias yang digunakan untuk bot yang berisi data sesi.

Diperlukan: Ya

botName

Nama bot yang berisi data sesi.

Diperlukan: Ya

userId

Pengidentifikasi pengguna yang terkait dengan data sesi.

Kendala Panjang: Panjang minimum 2. Panjang maksimum 100.

Pola: `[0-9a-zA-Z._:-]+`

Wajib: Ya

Isi Permintaan

Permintaan tidak memiliki isi permintaan.

Sintaks Respons

```
HTTP/1.1 200
Content-type: application/json

{
```

```
    "botAlias": "string",
    "botName": "string",
    "sessionId": "string",
    "userId": "string"
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Layanan mengembalikan data berikut dalam format JSON.

botAlias

Alias yang digunakan untuk bot yang terkait dengan data sesi.

Jenis: String

botName

Nama bot yang terkait dengan data sesi.

Jenis: String

sessionId

Pengenal unik untuk sesi tersebut.

Jenis: String

userId

ID pengguna aplikasi klien.

Jenis: String

Kendala Panjang: Panjang minimum 2. Panjang maksimum 100.

Pola: `[0-9a-zA-Z._:-]+`

Kesalahan

BadRequestException

Validasi permintaan gagal, tidak ada pesan yang dapat digunakan dalam konteksnya, atau pembuatan bot gagal, masih dalam proses, atau berisi perubahan yang belum dibangun.

Kode Status HTTP: 400

ConflictException

Dua klien menggunakan akun AWS yang sama, bot Amazon Lex, dan ID pengguna.

Kode Status HTTP: 409

InternalFailureException

Kesalahan layanan internal. Coba lagi panggilannya.

Kode Status HTTP: 500

LimitExceededException

Melebihi batas.

Kode Status HTTP: 429

NotFoundException

Sumber daya (seperti bot Amazon Lex atau alias) yang disebut tidak ditemukan.

Kode Status HTTP: 404

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

GetSession

Layanan: Amazon Lex Runtime Service

Mengembalikan informasi sesi untuk bot tertentu, alias, dan ID pengguna.

Minta Sintaks

```
GET /bot/botName/alias/botAlias/user/userId/session/?
checkpointLabelFilter=checkpointLabelFilter HTTP/1.1
```

Parameter Permintaan URI

Permintaan menggunakan parameter URI berikut.

botAlias

Alias yang digunakan untuk bot yang berisi data sesi.

Diperlukan: Ya

botName

Nama bot yang berisi data sesi.

Diperlukan: Ya

checkpointLabelFilter

String yang digunakan untuk memfilter maksud yang dikembalikan dalam `recentIntentSummaryView` struktur.

Saat Anda menentukan filter, hanya maksud dengan `checkpointLabel` bidangnya disetel ke string itu yang dikembalikan.

Batasan Panjang: Panjang minimum 1. Panjang maksimum 255.

Pola: `[a-zA-Z0-9-]+`

userId

ID pengguna aplikasi klien. Amazon Lex menggunakan ini untuk mengidentifikasi percakapan pengguna dengan bot Anda.

Kendala Panjang: Panjang minimum 2. Panjang maksimum 100.

Pola: `[0-9a-zA-Z._:-]+`

Wajib: Ya

Isi Permintaan

Permintaan tidak memiliki isi permintaan.

Sintaks Respons

```
HTTP/1.1 200
Content-type: application/json

{
   "activeContexts": [
      {
         "name": "string",
         "parameters": {
            "string" : "string"
         },
         "timeToLive": {
            "timeToLiveInSeconds": number,
            "turnsToLive": number
         }
      }
   ],
   "dialogAction": {
      "fulfillmentState": "string",
      "intentName": "string",
      "message": "string",
      "messageFormat": "string",
      "slots": {
         "string" : "string"
      },
      "slotToElicit": "string",
      "type": "string"
   },
   "recentIntentSummaryView": [
      {
         "checkpointLabel": "string",
         "confirmationStatus": "string",
```

```
        "dialogActionType": "string",
        "fulfillmentState": "string",
        "intentName": "string",
        "slots": {
           "string" : "string"
        },
        "slotToElicit": "string"
     }
  ],
  "sessionAttributes": {
     "string" : "string"
  },
  "sessionId": "string"
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Layanan mengembalikan data berikut dalam format JSON.

activeContexts

Daftar konteks aktif untuk sesi tersebut. Konteks dapat diatur ketika maksud terpenuhi atau dengan memanggil`PostContent`,`PostText`, atau `PutSession` operasi.

Anda dapat menggunakan konteks untuk mengontrol maksud yang dapat menindaklanjuti intent, atau untuk memodifikasi operasi aplikasi Anda.

Tipe: Array objek ActiveContext

Anggota Array: Jumlah minimum 0 item. Jumlah maksimum 20 item.

dialogAction

Menjelaskan keadaan bot saat ini.

Tipe: Objek DialogAction

recentIntentSummaryView

Array informasi tentang maksud yang digunakan dalam sesi. Array dapat berisi maksimal tiga ringkasan. Jika lebih dari tiga maksud digunakan dalam sesi, `recentIntentSummaryView` operasi berisi informasi tentang tiga maksud terakhir yang digunakan.

Jika Anda mengatur `checkpointLabelFilter` parameter dalam permintaan, array hanya berisi maksud dengan label yang ditentukan.

Tipe: Array objek IntentSummary

Anggota Array: Jumlah minimum 0 item. Jumlah maksimum 3 item.

sessionAttributes

Peta pasangan kunci/nilai yang mewakili informasi konteks khusus sesi. Ini berisi informasi aplikasi yang diteruskan antara Amazon Lex dan aplikasi klien.

Tipe: Peta string ke string

sessionId

Pengenal unik untuk sesi tersebut.

Jenis: String

Kesalahan

BadRequestException

Validasi permintaan gagal, tidak ada pesan yang dapat digunakan dalam konteksnya, atau pembuatan bot gagal, masih dalam proses, atau berisi perubahan yang belum dibangun.

Kode Status HTTP: 400

InternalFailureException

Kesalahan layanan internal. Coba lagi panggilannya.

Kode Status HTTP: 500

LimitExceededException

Melebihi batas.

Kode Status HTTP: 429

NotFoundException

Sumber daya (seperti bot Amazon Lex atau alias) yang disebut tidak ditemukan.

Kode Status HTTP: 404

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

PostContent

Layanan: Amazon Lex Runtime Service

Mengirim input pengguna (teks atau ucapan) ke Amazon Lex. Klien menggunakan API ini untuk mengirim permintaan teks dan audio ke Amazon Lex saat runtime. Amazon Lex menafsirkan input pengguna menggunakan model pembelajaran mesin yang dibuatnya untuk bot.

`PostContent`Operasi ini mendukung input audio pada 8kHz dan 16kHz. Anda dapat menggunakan audio 8kHz untuk mencapai akurasi pengenalan suara yang lebih tinggi dalam aplikasi audio telepon.

Sebagai tanggapan, Amazon Lex mengembalikan pesan berikutnya untuk disampaikan kepada pengguna. Perhatikan contoh pesan berikut:

- Untuk masukan pengguna “Saya ingin pizza,” Amazon Lex mungkin mengembalikan respons dengan pesan yang memunculkan data slot (misalnya,`PizzaSize`): “Pizza ukuran apa yang Anda inginkan?”.
- Setelah pengguna memberikan semua informasi pesanan pizza, Amazon Lex mungkin membalas tanggapan dengan pesan untuk mendapatkan konfirmasi pengguna: “Pesan pizza?”.
- Setelah pengguna menjawab “Ya” ke prompt konfirmasi, Amazon Lex mungkin mengembalikan pernyataan kesimpulan: “Terima kasih, pizza keju Anda telah dipesan.”.

Tidak semua pesan Amazon Lex memerlukan respons dari pengguna. Misalnya, pernyataan kesimpulan tidak memerlukan tanggapan. Beberapa pesan hanya membutuhkan jawaban ya atau tidak. Selain itu`message`, Amazon Lex menyediakan konteks tambahan tentang pesan dalam respons yang dapat Anda gunakan untuk meningkatkan perilaku klien, seperti menampilkan antarmuka pengguna klien yang sesuai. Pertimbangkan contoh berikut:

- Jika pesannya adalah untuk mendapatkan data slot, Amazon Lex mengembalikan informasi konteks berikut:
  - `x-amz-lex-dialog-state`header diatur ke `ElicitSlot`
  - `x-amz-lex-intent-name`header disetel ke nama maksud dalam konteks saat ini
  - `x-amz-lex-slot-to-elicit`header diatur ke nama slot yang `message` memunculkan informasi
  - `x-amz-lex-slots`header disetel ke peta slot yang dikonfigurasi untuk maksud dengan nilainya saat ini
- Jika pesan adalah prompt konfirmasi, `x-amz-lex-dialog-state` header diatur ke `Confirmation` dan `x-amz-lex-slot-to-elicit` header dihilangkan.

- Jika pesan adalah prompt klarifikasi yang dikonfigurasi untuk maksud, yang menunjukkan bahwa maksud pengguna tidak dipahami, `x-amz-dialog-state` header disetel ke `ElicitIntent` dan `x-amz-slot-to-elicit` header dihilangkan.

Selain itu, Amazon Lex juga mengembalikan aplikasi khusus `sessionAttributes` Anda. Untuk informasi selengkapnya, lihat Mengelola Konteks Percakapan.

Minta Sintaks

```
POST /bot/botName/alias/botAlias/user/userId/content HTTP/1.1
x-amz-lex-session-attributes: sessionAttributes
x-amz-lex-request-attributes: requestAttributes
Content-Type: contentType
Accept: accept
x-amz-lex-active-contexts: activeContexts

inputStream
```

Parameter Permintaan URI

Permintaan menggunakan parameter URI berikut.

accept

Anda meneruskan nilai ini sebagai header `Accept` HTTP.

Pesan yang dikembalikan Amazon Lex dalam respons dapat berupa teks atau ucapan berdasarkan nilai header `Accept` HTTP dalam permintaan.

- Jika nilainya`text/plain; charset=utf-8`, Amazon Lex mengembalikan teks dalam respons.
- Jika nilai dimulai dengan`audio/`, Amazon Lex mengembalikan pidato dalam respons. Amazon Lex menggunakan Amazon Polly untuk menghasilkan pidato (menggunakan konfigurasi yang Anda tentukan di `Accept` header). Misalnya, jika Anda menentukan `audio/mpeg` sebagai nilai, Amazon Lex mengembalikan ucapan dalam format MPEG.
- Jika nilainya`audio/pcm`, pidato yang dikembalikan `audio/pcm` dalam 16-bit, format endian kecil.
- Berikut ini adalah nilai yang diterima:
  - audio/mpeg
  - audio/ogg

- audio/pcm
- teks/polos; charset = utf-8
- audio/* (default ke mpeg)

activeContexts

Daftar konteks yang aktif untuk permintaan. Konteks dapat diaktifkan ketika maksud sebelumnya terpenuhi, atau dengan memasukkan konteks dalam permintaan,

Jika Anda tidak menentukan daftar konteks, Amazon Lex akan menggunakan daftar konteks saat ini untuk sesi tersebut. Jika Anda menentukan daftar kosong, semua konteks untuk sesi akan dihapus.

botAlias

Alias bot Amazon Lex.

Diperlukan: Ya

botName

Nama bot Amazon Lex.

Diperlukan: Ya

contentType

Anda meneruskan nilai ini sebagai header `Content-Type` HTTP.

Menunjukkan format audio atau teks. Nilai header harus dimulai dengan salah satu awalan berikut:

- Format PCM, data audio harus dalam urutan byte endian kecil.
  - audio/l16; tingkat = 16000; saluran = 1
  - audio/x-l16; tingkat sampel = 16000; jumlah saluran = 1
  - audio/lpcm; laju sampel = 8000; = 16; jumlah saluran = 1; = salah sample-size-bits is-big-endian
- Format karya
  - audio/ x-cbr-opus-with -pembukaan; ukuran pembukaan = 0; bit-rate = 256000; = 4 frame-size-milliseconds
- Format teks
  - teks/polos; charset = utf-8

Diperlukan: Ya

requestAttributes

Anda meneruskan nilai ini sebagai header `x-amz-lex-request-attributes` HTTP.

Informasi khusus permintaan diteruskan antara Amazon Lex dan aplikasi klien. Nilai harus berupa JSON serialized dan base64 dikodekan peta dengan kunci string dan nilai-nilai. Ukuran total `requestAttributes` dan `sessionAttributes` header dibatasi hingga 12 KB.

Namespace dicadangkan `x-amz-lex:` untuk atribut khusus. Jangan membuat atribut permintaan apa pun dengan awalan`x-amz-lex:`.

Untuk informasi selengkapnya, lihat Menyetel Atribut Permintaan.

sessionAttributes

Anda meneruskan nilai ini sebagai header `x-amz-lex-session-attributes` HTTP.

Informasi khusus aplikasi diteruskan antara Amazon Lex dan aplikasi klien. Nilai harus berupa JSON serialized dan base64 dikodekan peta dengan kunci string dan nilai-nilai. Ukuran total `sessionAttributes` dan `requestAttributes` header dibatasi hingga 12 KB.

Untuk informasi selengkapnya, lihat Menyetel Atribut Sesi.

userId

ID pengguna aplikasi klien. Amazon Lex menggunakan ini untuk mengidentifikasi percakapan pengguna dengan bot Anda. Saat runtime, setiap permintaan harus berisi `userID` bidang.

Untuk memutuskan ID pengguna yang akan digunakan untuk aplikasi Anda, pertimbangkan faktor-faktor berikut.

- `userID`Bidang tidak boleh berisi informasi pribadi pengguna, misalnya, nama, nomor identifikasi pribadi, atau informasi pribadi pengguna akhir lainnya.
- Jika Anda ingin pengguna memulai percakapan di satu perangkat dan melanjutkan di perangkat lain, gunakan pengenal khusus pengguna.
- Jika Anda ingin pengguna yang sama dapat melakukan dua percakapan independen di dua perangkat yang berbeda, pilih pengenal khusus perangkat.
- Pengguna tidak dapat melakukan dua percakapan independen dengan dua versi berbeda dari bot yang sama. Misalnya, pengguna tidak dapat melakukan percakapan dengan versi PROD dan BETA dari bot yang sama. Jika Anda mengantisipasi bahwa pengguna perlu melakukan

percakapan dengan dua versi yang berbeda, misalnya, saat menguji, sertakan alias bot di ID pengguna untuk memisahkan dua percakapan.

Kendala Panjang: Panjang minimum 2. Panjang maksimum 100.

Pola: `[0-9a-zA-Z._:-]+`

Wajib: Ya

Isi Permintaan

Permintaan menerima data biner berikut.

inputStream

Masukan pengguna dalam format audio PCM atau Opus atau format teks seperti yang dijelaskan dalam header `Content-Type` HTTP.

Anda dapat melakukan streaming data audio ke Amazon Lex atau Anda dapat membuat buffer lokal yang menangkap semua data audio sebelum mengirim. Secara umum, Anda mendapatkan kinerja yang lebih baik jika Anda melakukan streaming data audio daripada buffering data secara lokal.

Diperlukan: Ya

Sintaksis Respons

```
HTTP/1.1 200
Content-Type: contentType
x-amz-lex-intent-name: intentName
x-amz-lex-nlu-intent-confidence: nluIntentConfidence
x-amz-lex-alternative-intents: alternativeIntents
x-amz-lex-slots: slots
x-amz-lex-session-attributes: sessionAttributes
x-amz-lex-sentiment: sentimentResponse
x-amz-lex-message: message
x-amz-lex-encoded-message: encodedMessage
x-amz-lex-message-format: messageFormat
x-amz-lex-dialog-state: dialogState
x-amz-lex-slot-to-elicit: slotToElicit
x-amz-lex-input-transcript: inputTranscript
x-amz-lex-encoded-input-transcript: encodedInputTranscript
```

```
x-amz-lex-bot-version: botVersion
x-amz-lex-session-id: sessionId
x-amz-lex-active-contexts: activeContexts

audioStream
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Respons mengembalikan header HTTP berikut.

activeContexts

Daftar konteks aktif untuk sesi tersebut. Konteks dapat diatur ketika maksud terpenuhi atau dengan memanggil`PostContent`,`PostText`, atau `PutSession` operasi.

Anda dapat menggunakan konteks untuk mengontrol maksud yang dapat menindaklanjuti intent, atau untuk memodifikasi operasi aplikasi Anda.

alternativeIntents

Satu hingga empat maksud alternatif yang mungkin berlaku untuk maksud pengguna.

Setiap alternatif menyertakan skor yang menunjukkan seberapa yakin Amazon Lex bahwa maksud tersebut cocok dengan maksud pengguna. Maksud diurutkan berdasarkan skor kepercayaan.

botVersion

Versi bot yang menanggapi percakapan. Anda dapat menggunakan informasi ini untuk membantu menentukan apakah satu versi bot berkinerja lebih baik daripada versi lain.

Batasan Panjang: Panjang minimum 1. Panjang maksimum adalah 64.

Pola: `[0-9]+|\$LATEST`

contentType

Jenis konten seperti yang ditentukan dalam header `Accept` HTTP dalam permintaan.

dialogState

Mengidentifikasi keadaan interaksi pengguna saat ini. Amazon Lex mengembalikan salah satu nilai berikut sebagai`dialogState`. Klien secara opsional dapat menggunakan informasi ini untuk menyesuaikan antarmuka pengguna.

- `ElicitIntent`- Amazon Lex ingin mendapatkan maksud pengguna. Pertimbangkan contoh berikut:

  Misalnya, pengguna mungkin mengucapkan maksud ("Saya ingin memesan pizza"). Jika Amazon Lex tidak dapat menyimpulkan maksud pengguna dari ucapan ini, Amazon Lex akan mengembalikan status dialog ini.

- `ConfirmIntent`- Amazon Lex mengharapkan respons "ya" atau "tidak".

  Misalnya, Amazon Lex menginginkan konfirmasi pengguna sebelum memenuhi maksud. Alih-alih respons "ya" atau "tidak" sederhana, pengguna mungkin merespons dengan informasi tambahan. Misalnya, "ya, tapi buatlah pizza kerak tebal" atau "tidak, saya ingin memesan minuman." Amazon Lex dapat memproses informasi tambahan tersebut (dalam contoh ini, perbarui slot tipe kerak atau ubah maksud dari OrderPizza ke OrderDrink).

- `ElicitSlot`- Amazon Lex mengharapkan nilai slot untuk maksud saat ini.

  Misalnya, anggaplah dalam tanggapannya Amazon Lex mengirim pesan ini: "Berapa ukuran pizza yang Anda inginkan?". Seorang pengguna mungkin membalas dengan nilai slot (misalnya, "medium"). Pengguna mungkin juga memberikan informasi tambahan dalam tanggapan (misalnya, "pizza kerak tebal sedang"). Amazon Lex dapat memproses informasi tambahan tersebut dengan tepat.

- `Fulfilled`- Menyampaikan bahwa fungsi Lambda telah berhasil memenuhi maksud.
- `ReadyForFulfillment`- Menyampaikan bahwa klien harus memenuhi permintaan.
- `Failed`- Menyampaikan bahwa percakapan dengan pengguna gagal.

  Hal ini dapat terjadi karena berbagai alasan, termasuk bahwa pengguna tidak memberikan respons yang sesuai terhadap permintaan dari layanan (Anda dapat mengonfigurasi berapa kali Amazon Lex dapat meminta pengguna untuk informasi tertentu), atau jika fungsi Lambda gagal memenuhi intent.

Nilai yang Valid: `ElicitIntent | ConfirmIntent | ElicitSlot | Fulfilled | ReadyForFulfillment | Failed`

encodedInputTranscript

Teks yang digunakan untuk memproses permintaan.

Jika input adalah aliran audio, `encodedInputTranscript` bidang berisi teks yang diekstrak dari aliran audio. Ini adalah teks yang sebenarnya diproses untuk mengenali maksud dan nilai slot.

Anda dapat menggunakan informasi ini untuk menentukan apakah Amazon Lex memproses audio yang Anda kirim dengan benar.

`encodedInputTranscript`Bidang ini dikodekan basis-64. Anda harus memecahkan kode bidang sebelum Anda dapat menggunakan nilai.

encodedMessage

Pesan untuk disampaikan kepada pengguna. Pesan dapat berasal dari konfigurasi bot atau dari fungsi Lambda.

Jika intent tidak dikonfigurasi dengan fungsi Lambda, atau jika fungsi Lambda `Delegate` ditampilkan sebagai `dialogAction.type` respons, Amazon Lex memutuskan tindakan berikutnya dan memilih pesan yang sesuai dari konfigurasi bot berdasarkan konteks interaksi saat ini. Misalnya, jika Amazon Lex tidak dapat memahami masukan pengguna, Amazon Lex menggunakan pesan prompt klarifikasi.

Saat membuat maksud, Anda dapat menetapkan pesan ke grup. Ketika pesan ditetapkan ke grup Amazon Lex mengembalikan satu pesan dari setiap grup dalam respons. Bidang pesan adalah string JSON yang lolos yang berisi pesan. Untuk informasi lebih lanjut tentang struktur string JSON yang dikembalikan, lihatFormat Pesan yang Didukung.

Jika fungsi Lambda mengembalikan pesan, Amazon Lex meneruskannya ke klien dalam responsnya.

`encodedMessage`Bidang ini dikodekan basis-64. Anda harus memecahkan kode bidang sebelum Anda dapat menggunakan nilai.

Batasan Panjang: Panjang minimum 1. Panjang maksimum 1366.

inputTranscript

Header ini sudah tidak digunakan lagi.

Teks yang digunakan untuk memproses permintaan.

Anda dapat menggunakan bidang ini hanya di lokal De-de, en-AU, en-GB, en-US, es-419, es-ES, es-US, fr-Ca, fr-FR, dan IT-it. Di semua lokal lainnya, `inputTranscript` bidangnya nol. Anda harus menggunakan `encodedInputTranscript` bidang sebagai gantinya.

Jika input adalah aliran audio, `inputTranscript` bidang berisi teks yang diekstrak dari aliran audio. Ini adalah teks yang sebenarnya diproses untuk mengenali maksud dan nilai slot. Anda

dapat menggunakan informasi ini untuk menentukan apakah Amazon Lex memproses audio yang Anda kirim dengan benar.

intentName

Maksud pengguna saat ini yang diketahui Amazon Lex.

message

Header ini sudah tidak digunakan lagi.

Anda hanya dapat menggunakan bidang ini di lokal De-de, en-AU, en-GB, en-US, es-419, es-ES, es-US, fr-Ca, fr-FR, dan IT-it. Di semua lokal lainnya, `message` bidangnya nol. Anda harus menggunakan `encodedMessage` bidang sebagai gantinya.

Pesan untuk disampaikan kepada pengguna. Pesan dapat berasal dari konfigurasi bot atau dari fungsi Lambda.

Jika intent tidak dikonfigurasi dengan fungsi Lambda, atau jika fungsi Lambda `Delegate` ditampilkan sebagai `dialogAction.type` respons, Amazon Lex memutuskan tindakan berikutnya dan memilih pesan yang sesuai dari konfigurasi bot berdasarkan konteks interaksi saat ini. Misalnya, jika Amazon Lex tidak dapat memahami masukan pengguna, Amazon Lex menggunakan pesan prompt klarifikasi.

Saat membuat maksud, Anda dapat menetapkan pesan ke grup. Ketika pesan ditetapkan ke grup Amazon Lex mengembalikan satu pesan dari setiap grup dalam respons. Bidang pesan adalah string JSON yang lolos yang berisi pesan. Untuk informasi lebih lanjut tentang struktur string JSON yang dikembalikan, lihatFormat Pesan yang Didukung.

Jika fungsi Lambda mengembalikan pesan, Amazon Lex meneruskannya ke klien dalam responsnya.

Batasan Panjang: Panjang minimum 1. Panjang maksimum 1024.

messageFormat

Format pesan respons. Salah satu nilai berikut:

- `PlainText`- Pesan berisi teks UTF-8 biasa.
- `CustomPayload`- Pesan adalah format khusus untuk klien.
- `SSML`- Pesan berisi teks yang diformat untuk output suara.
- `Composite`- Pesan berisi objek JSON yang lolos yang berisi satu atau beberapa pesan dari grup tempat pesan ditetapkan saat maksud dibuat.

Nilai yang Valid: `PlainText | CustomPayload | SSML | Composite`

nluIntentConfidence

Memberikan skor yang menunjukkan seberapa yakin Amazon Lex bahwa intent yang dikembalikan adalah yang cocok dengan maksud pengguna. Skornya antara 0,0 dan 1,0.

Skor adalah skor relatif, bukan skor absolut. Skor dapat berubah berdasarkan peningkatan Amazon Lex.

sentimentResponse

Sentimen yang diungkapkan dalam sebuah ucapan.

Ketika bot dikonfigurasi untuk mengirim ucapan ke Amazon Comprehend untuk analisis sentimen, bidang ini berisi hasil analisis.

sessionAttributes

Peta pasangan kunci/nilai yang mewakili informasi konteks khusus sesi.

sessionId

Pengenal unik untuk sesi tersebut.

slots

Peta slot maksud nol atau lebih (pasangan nama/nilai) Amazon Lex terdeteksi dari input pengguna selama percakapan. Bidang ini dikodekan basis-64.

Amazon Lex membuat daftar resolusi yang berisi kemungkinan nilai untuk slot. Nilai yang dikembalikan ditentukan oleh yang `valueSelectionStrategy` dipilih ketika jenis slot dibuat atau diperbarui. Jika `valueSelectionStrategy` diatur ke`ORIGINAL_VALUE`, nilai yang diberikan oleh pengguna dikembalikan, jika nilai pengguna mirip dengan nilai slot. Jika `valueSelectionStrategy` diatur ke `TOP_RESOLUTION` Amazon Lex mengembalikan nilai pertama dalam daftar resolusi atau, jika tidak ada daftar resolusi, null. Jika Anda tidak menentukan `avalueSelectionStrategy`, defaultnya adalah`ORIGINAL_VALUE`.

slotToElicit

Jika `dialogState` nilainya`ElicitSlot`, mengembalikan nama slot yang Amazon Lex memunculkan nilai.

Respons mengembalikan yang berikut sebagai isi HTTP.

audioStream

Prompt (atau pernyataan) untuk disampaikan kepada pengguna. Ini didasarkan pada konfigurasi dan konteks bot. Misalnya, jika Amazon Lex tidak memahami maksud pengguna, Amazon Lex mengirimkan `clarificationPrompt` konfigurasi untuk bot. Jika maksud memerlukan konfirmasi sebelum mengambil tindakan pemenuhan, ia mengirimkan. `confirmationPrompt` Contoh lain: Misalkan fungsi Lambda berhasil memenuhi intent, dan mengirim pesan untuk disampaikan kepada pengguna. Kemudian Amazon Lex mengirimkan pesan itu sebagai tanggapan.

Kesalahan

BadGatewayException

Entah bot Amazon Lex masih dibangun, atau salah satu layanan dependen (Amazon Polly, AWS Lambda) gagal dengan kesalahan layanan internal.

Kode Status HTTP: 502

BadRequestException

Validasi permintaan gagal, tidak ada pesan yang dapat digunakan dalam konteks, atau pembuatan bot gagal, masih dalam proses, atau berisi perubahan yang belum dibangun.

Kode Status HTTP: 400

ConflictException

Dua klien menggunakan akun AWS yang sama, bot Amazon Lex, dan ID pengguna.

Kode Status HTTP: 409

DependencyFailedException

Salah satu dependensi, seperti AWS Lambda atau Amazon Polly, memberikan pengecualian. Misalnya,

- Jika Amazon Lex tidak memiliki izin yang cukup untuk memanggil fungsi Lambda.
- Jika fungsi Lambda membutuhkan waktu lebih dari 30 detik untuk dijalankan.
- Jika fungsi Lambda pemenuhan mengembalikan `Delegate` tindakan dialog tanpa menghapus nilai slot apa pun.

Kode Status HTTP: 424

InternalFailureException

Kesalahan layanan internal. Coba lagi panggilannya.

Kode Status HTTP: 500

LimitExceededException

Melebihi batas.

Kode Status HTTP: 429

LoopDetectedException

Pengecualian ini tidak digunakan.

Kode Status HTTP: 508

NotAcceptableException

Header terima dalam permintaan tidak memiliki nilai yang valid.

Kode Status HTTP: 406

NotFoundException

Sumber daya (seperti bot Amazon Lex atau alias) yang disebut tidak ditemukan.

Kode Status HTTP: 404

RequestTimeoutException

Pidato masukan terlalu panjang.

Kode Status HTTP: 408

UnsupportedMediaTypeException

Header Content-Type (`PostContent`API) memiliki nilai yang tidak valid.

Kode Status HTTP: 415

Contoh-contoh

Contoh 1

Dalam permintaan ini, URI mengidentifikasi bot (Lalu Lintas), versi bot ($LATEST), dan nama pengguna akhir (someuser). `Content-Type`Header mengidentifikasi format audio dalam tubuh.

Amazon Lex juga mendukung format lain. Untuk mengonversi audio dari satu format ke format lain, jika perlu, Anda dapat menggunakan perangkat lunak open source SoX. Anda menentukan format di mana Anda ingin mendapatkan respons dengan menambahkan header `Accept` HTTP.

Sebagai tanggapan, `x-amz-lex-message` header menunjukkan respons yang dikembalikan Amazon Lex. Klien kemudian dapat mengirimkan respons ini kepada pengguna. Pesan yang sama dikirim dalam format Audio/MPEG melalui pengkodean chunked (seperti yang diminta).

Permintaan Sampel

```
"POST /bot/Traffic/alias/$LATEST/user/someuser/content HTTP/1.1[\r][\n]"
"x-amz-lex-session-attributes: eyJ1c2VyTmFtZSI6IkJvYiJ9[\r][\n]"
"Content-Type: audio/x-l16; channel-count=1; sample-rate=16000f[\r][\n]"
"Accept: audio/mpeg[\r][\n]"
"Host: runtime.lex.us-east-1.amazonaws.com[\r][\n]"
"Authorization: AWS4-HMAC-SHA256 Credential=BLANKED_OUT/20161230/us-east-1/lex/
aws4_request,
SignedHeaders=accept;content-type;host;x-amz-content-sha256;x-amz-date;x-amz-lex-
session-attributes,
 Signature=78ca5b54ea3f64a17ff7522de02cd90a9acd2365b45a9ce9b96ea105bb1c7ec2[\r][\n]"
"X-Amz-Date: 20161230T181426Z[\r][\n]"
"X-Amz-Content-Sha256:
 e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855[\r][\n]"
"Transfer-Encoding: chunked[\r][\n]"
"Connection: Keep-Alive[\r][\n]"
"User-Agent: Apache-HttpClient/4.5.x (Java/1.8.0_112)[\r][\n]"
"Accept-Encoding: gzip,deflate[\r][\n]"
"[\r][\n]"
"1000[\r][\n]"
"[0x7][0x0][0x7][0x0][\n]"
"[0x0][0x7][0x0][0xfc][0xff][\n]"
"[0x0][\n]"
…
```

Contoh Respons

```
"HTTP/1.1 200 OK[\r][\n]"
"x-amzn-RequestId: cc8b34af-cebb-11e6-a35c-55f3a992f28d[\r][\n]"
"x-amz-lex-message: Sorry, can you repeat that?[\r][\n]"
"x-amz-lex-dialog-state: ElicitIntent[\r][\n]"
"x-amz-lex-session-attributes: eyJ1c2VyTmFtZSI6IkJvYiJ9[\r][\n]"
"Content-Type: audio/mpeg[\r][\n]"
```

```
"Transfer-Encoding: chunked[\r][\n]"
"Date: Fri, 30 Dec 2016 18:14:28 GMT[\r][\n]"
"[\r][\n]"
"2000[\r][\n]"
"ID3[0x4][0x0][0x0][0x0][0x0][0x0]#TSSE[0x0][0x0][0x0][0xf][0x0][0x0]
[0x3]Lavf57.41.100[0x0][0x0][0x0][0x0][0x0][0x0][0x0][0x0][0x0][0x0][0xff]
[0xf3]`[0xc4][0x0][0x1b]{[0x8d][0xe8][0x1]C[0x18][0x1][0x0]J[0xe0]`b[0xdd][0xd1]
[0xb][0xfd][0x11][0xdf][0xfe]";[0xbb][0xbb][0x9f][0xee][0xee][0xee][0xee]|DDD/[0xff]
[0xff][0xff][0xff]www?D[0xf7]w^?[0xff][0xfa]h[0x88][0x85][0xfe][0x88][0x88][0x88]
[[0xa2]'[0xff][0xfa]"{[0x9f][0xe8][0x88]]D[0xeb][0xbb][0xbb][0xa2]!u[0xfd][0xdd][0xdf]
[0x88][0x94][0x0]F[0xef][0xa1]8[0x0][0x82]w[0x88]N[0x0][0x0][0x9b][0xbb][0xe8][0xe
…
```

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

PostText

Layanan: Amazon Lex Runtime Service

Mengirim masukan pengguna ke Amazon Lex. Aplikasi klien dapat menggunakan API ini untuk mengirim permintaan ke Amazon Lex saat runtime. Amazon Lex kemudian menafsirkan input pengguna menggunakan model pembelajaran mesin yang dibuatnya untuk bot.

Sebagai tanggapan, Amazon Lex mengembalikan yang berikutnya `message` untuk menyampaikan kepada pengguna opsional `responseCard` untuk ditampilkan. Perhatikan contoh pesan berikut:

- Untuk masukan pengguna “Saya ingin pizza”, Amazon Lex mungkin mengembalikan respons dengan pesan yang memunculkan data slot (misalnya, PizzaSize): “Pizza ukuran apa yang Anda inginkan?”
- Setelah pengguna memberikan semua informasi pesanan pizza, Amazon Lex mungkin mengembalikan tanggapan dengan pesan untuk mendapatkan konfirmasi pengguna “Lanjutkan dengan pesanan pizza?”.
- Setelah pengguna membalas prompt konfirmasi dengan “ya”, Amazon Lex mungkin mengembalikan pernyataan kesimpulan: “Terima kasih, pizza keju Anda telah dipesan.”

Tidak semua pesan Amazon Lex memerlukan respons pengguna. Misalnya, pernyataan kesimpulan tidak memerlukan tanggapan. Beberapa pesan hanya memerlukan respons pengguna “ya” atau “tidak”. Selain itu`message`, Amazon Lex menyediakan konteks tambahan tentang pesan dalam respons yang mungkin Anda gunakan untuk meningkatkan perilaku klien, misalnya, untuk menampilkan antarmuka pengguna klien yang sesuai. Ini adalah`slotToElicit`,`dialogState`,`intentName`, dan `slots` bidang dalam tanggapan. Pertimbangkan contoh berikut:

- Jika pesannya adalah untuk mendapatkan data slot, Amazon Lex mengembalikan informasi konteks berikut:
  - `dialogState`disetel ke ElicitSlot
  - `intentName`disetel ke nama maksud dalam konteks saat ini
  - `slotToElicit`atur ke nama slot `message` yang mendapatkan informasi
  - `slots`disetel ke peta slot, dikonfigurasi untuk maksud, dengan nilai yang diketahui saat ini
- Jika pesan adalah prompt konfirmasi, `dialogState` diatur ke ConfirmIntent dan `SlotToElicit` diatur ke null.

- Jika pesan adalah prompt klarifikasi (dikonfigurasi untuk maksud) yang menunjukkan bahwa maksud pengguna tidak dipahami, maka akan disetel ke ElicitIntent dan `dialogState` `slotToElicit` disetel ke null.

Selain itu, Amazon Lex juga mengembalikan aplikasi khusus `sessionAttributes` Anda. Untuk informasi selengkapnya, lihat Mengelola Konteks Percakapan.

Minta Sintaks

```
POST /bot/botName/alias/botAlias/user/userId/text HTTP/1.1
Content-type: application/json

{
   "activeContexts": [
      {
         "name": "string",
         "parameters": {
            "string" : "string"
         },
         "timeToLive": {
            "timeToLiveInSeconds": number,
            "turnsToLive": number
         }
      }
   ],
   "inputText": "string",
   "requestAttributes": {
      "string" : "string"
   },
   "sessionAttributes": {
      "string" : "string"
   }
}
```

Parameter Permintaan URI

Permintaan menggunakan parameter URI berikut.

botAlias

Alias bot Amazon Lex.

Diperlukan: Ya

botName

Nama bot Amazon Lex.

Diperlukan: Ya

userId

ID pengguna aplikasi klien. Amazon Lex menggunakan ini untuk mengidentifikasi percakapan pengguna dengan bot Anda. Saat runtime, setiap permintaan harus berisi `userID` bidang.

Untuk memutuskan ID pengguna yang akan digunakan untuk aplikasi Anda, pertimbangkan faktor-faktor berikut.

- `userID`Bidang tidak boleh berisi informasi pribadi pengguna, misalnya, nama, nomor identifikasi pribadi, atau informasi pribadi pengguna akhir lainnya.
- Jika Anda ingin pengguna memulai percakapan di satu perangkat dan melanjutkan di perangkat lain, gunakan pengenal khusus pengguna.
- Jika Anda ingin pengguna yang sama dapat melakukan dua percakapan independen di dua perangkat yang berbeda, pilih pengenal khusus perangkat.
- Pengguna tidak dapat melakukan dua percakapan independen dengan dua versi berbeda dari bot yang sama. Misalnya, pengguna tidak dapat melakukan percakapan dengan versi PROD dan BETA dari bot yang sama. Jika Anda mengantisipasi bahwa pengguna perlu melakukan percakapan dengan dua versi yang berbeda, misalnya, saat menguji, sertakan alias bot di ID pengguna untuk memisahkan dua percakapan.

Kendala Panjang: Panjang minimum 2. Panjang maksimum 100.

Pola: `[0-9a-zA-Z._:-]+`

Wajib: Ya

Isi Permintaan

Permintaan menerima data berikut dalam format JSON.

activeContexts

Daftar konteks yang aktif untuk permintaan. Konteks dapat diaktifkan ketika maksud sebelumnya terpenuhi, atau dengan memasukkan konteks dalam permintaan,

Jika Anda tidak menentukan daftar konteks, Amazon Lex akan menggunakan daftar konteks saat ini untuk sesi tersebut. Jika Anda menentukan daftar kosong, semua konteks untuk sesi akan dihapus.

Tipe: Array objek ActiveContext

Anggota Array: Jumlah minimum 0 item. Jumlah maksimum 20 item.

Diperlukan: Tidak

inputText

Teks yang dimasukkan pengguna (Amazon Lex menafsirkan teks ini).

Saat Anda menggunakan AWS CLI, Anda tidak dapat meneruskan URL dalam parameter. `--input-text` Lulus URL menggunakan `--cli-input-json` parameter sebagai gantinya.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 1024.

Diperlukan: Ya

requestAttributes

Informasi khusus permintaan diteruskan antara Amazon Lex dan aplikasi klien.

Namespace dicadangkan `x-amz-lex:` untuk atribut khusus. Jangan membuat atribut permintaan apa pun dengan awalan`x-amz-lex:`.

Untuk informasi selengkapnya, lihat Menyetel Atribut Permintaan.

Tipe: Peta antar string

Diperlukan: Tidak

sessionAttributes

Informasi khusus aplikasi diteruskan antara Amazon Lex dan aplikasi klien.

Untuk informasi selengkapnya, lihat Menyetel Atribut Sesi.

Tipe: Peta antar string

Diperlukan: Tidak

Sintaksis Respons

```
HTTP/1.1 200
Content-type: application/json

{
   "activeContexts": [
      {
         "name": "string",
         "parameters": {
            "string" : "string"
         },
         "timeToLive": {
            "timeToLiveInSeconds": number,
            "turnsToLive": number
         }
      }
   ],
   "alternativeIntents": [
      {
         "intentName": "string",
         "nluIntentConfidence": {
            "score": number
         },
         "slots": {
            "string" : "string"
         }
      }
   ],
   "botVersion": "string",
   "dialogState": "string",
   "intentName": "string",
   "message": "string",
   "messageFormat": "string",
   "nluIntentConfidence": {
      "score": number
   },
   "responseCard": {
      "contentType": "string",
      "genericAttachments": [
         {
            "attachmentLinkUrl": "string",
            "buttons": [
               {
```

```
                "text": "string",
                "value": "string"
            }
        ],
        "imageUrl": "string",
        "subTitle": "string",
        "title": "string"
      }
    ],
    "version": "string"
  },
  "sentimentResponse": {
    "sentimentLabel": "string",
    "sentimentScore": "string"
  },
  "sessionAttributes": {
    "string" : "string"
  },
  "sessionId": "string",
  "slots": {
    "string" : "string"
  },
  "slotToElicit": "string"
}
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Layanan mengembalikan data berikut dalam format JSON.

activeContexts

Daftar konteks aktif untuk sesi tersebut. Konteks dapat diatur ketika maksud terpenuhi atau dengan memanggil`PostContent`,`PostText`, atau `PutSession` operasi.

Anda dapat menggunakan konteks untuk mengontrol maksud yang dapat menindaklanjuti intent, atau untuk memodifikasi operasi aplikasi Anda.

Tipe: Array objek ActiveContext

Anggota Array: Jumlah minimum 0 item. Jumlah maksimum 20 item.

alternativeIntents

Satu hingga empat maksud alternatif yang mungkin berlaku untuk maksud pengguna.

Setiap alternatif menyertakan skor yang menunjukkan seberapa yakin Amazon Lex bahwa maksud tersebut cocok dengan maksud pengguna. Maksudnya diurutkan berdasarkan skor kepercayaan.

Tipe: Array objek PredictedIntent

Anggota Array: Jumlah maksimum 4 item.

botVersion

Versi bot yang menanggapi percakapan. Anda dapat menggunakan informasi ini untuk membantu menentukan apakah satu versi bot berkinerja lebih baik daripada versi lain.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum adalah 64.

Pola: `[0-9]+|\$LATEST`

dialogState

Mengidentifikasi keadaan interaksi pengguna saat ini. Amazon Lex mengembalikan salah satu nilai berikut sebagai`dialogState`. Klien secara opsional dapat menggunakan informasi ini untuk menyesuaikan antarmuka pengguna.

- `ElicitIntent`- Amazon Lex ingin mendapatkan niat pengguna.

  Misalnya, pengguna mungkin mengucapkan maksud (“Saya ingin memesan pizza”). Jika Amazon Lex tidak dapat menyimpulkan maksud pengguna dari ucapan ini, Amazon Lex akan mengembalikan DialogState ini.
- `ConfirmIntent`- Amazon Lex mengharapkan respons “ya” atau “tidak”.

  Misalnya, Amazon Lex menginginkan konfirmasi pengguna sebelum memenuhi maksud.

  Alih-alih “ya” atau “tidak” sederhana, pengguna mungkin merespons dengan informasi tambahan. Misalnya, “ya, tapi buatlah pizza kerak kental” atau “tidak, saya ingin memesan minuman”. Amazon Lex dapat memproses informasi tambahan tersebut (dalam contoh ini, memperbarui nilai slot tipe kerak, atau mengubah maksud dari OrderPizza ke OrderDrink).
- `ElicitSlot`- Amazon Lex mengharapkan nilai slot untuk maksud saat ini.

Misalnya, anggaplah dalam tanggapannya Amazon Lex mengirim pesan ini: “Berapa ukuran pizza yang Anda inginkan?”. Seorang pengguna mungkin membalas dengan nilai slot (misalnya, “medium”). Pengguna mungkin juga memberikan informasi tambahan dalam tanggapan (misalnya, “pizza kerak tebal sedang”). Amazon Lex dapat memproses informasi tambahan tersebut dengan tepat.

- `Fulfilled`- Menyampaikan bahwa fungsi Lambda yang dikonfigurasi untuk maksud telah berhasil memenuhi intent.
- `ReadyForFulfillment`- Menyampaikan bahwa klien harus memenuhi niat.
- `Failed`- Menyampaikan bahwa percakapan dengan pengguna gagal.

  Hal ini dapat terjadi karena berbagai alasan termasuk bahwa pengguna tidak memberikan respons yang sesuai terhadap permintaan dari layanan (Anda dapat mengonfigurasi berapa kali Amazon Lex dapat meminta pengguna untuk informasi tertentu), atau fungsi Lambda gagal memenuhi intent.

Jenis: String

Nilai yang Valid: `ElicitIntent | ConfirmIntent | ElicitSlot | Fulfilled | ReadyForFulfillment | Failed`

intentName

Maksud pengguna saat ini yang diketahui Amazon Lex.

Jenis: String

message

Pesan untuk disampaikan kepada pengguna. Pesan dapat berasal dari konfigurasi bot atau dari fungsi Lambda.

Jika intent tidak dikonfigurasi dengan fungsi Lambda, atau jika fungsi Lambda `Delegate` ditampilkan sebagai `dialogAction.type` responsnya, Amazon Lex memutuskan tindakan berikutnya dan memilih pesan yang sesuai dari konfigurasi bot berdasarkan konteks interaksi saat ini. Misalnya, jika Amazon Lex tidak dapat memahami masukan pengguna, Amazon Lex menggunakan pesan prompt klarifikasi.

Saat membuat intent, Anda dapat menetapkan pesan ke grup. Ketika pesan ditetapkan ke grup Amazon Lex mengembalikan satu pesan dari setiap grup dalam respons. Bidang pesan adalah

string JSON yang lolos yang berisi pesan. Untuk informasi lebih lanjut tentang struktur string JSON yang dikembalikan, lihatFormat Pesan yang Didukung.

Jika fungsi Lambda mengembalikan pesan, Amazon Lex meneruskannya ke klien dalam responsnya.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 1024.

messageFormat

Format pesan respons. Salah satu nilai berikut:

- `PlainText`- Pesan berisi teks UTF-8 biasa.
- `CustomPayload`- Pesan adalah format khusus yang ditentukan oleh fungsi Lambda.
- `SSML`- Pesan berisi teks yang diformat untuk output suara.
- `Composite`- Pesan berisi objek JSON yang lolos yang berisi satu atau beberapa pesan dari grup tempat pesan ditetapkan saat maksud dibuat.

Jenis: String

Nilai yang Valid: `PlainText | CustomPayload | SSML | Composite`

nluIntentConfidence

Memberikan skor yang menunjukkan seberapa yakin Amazon Lex bahwa intent yang dikembalikan adalah yang cocok dengan maksud pengguna. Skornya antara 0,0 dan 1,0. Untuk informasi lebih lanjut, lihat Skor Keyakinan.

Skor adalah skor relatif, bukan skor absolut. Skor dapat berubah berdasarkan peningkatan Amazon Lex.

Tipe: Objek IntentConfidence

responseCard

Merupakan opsi yang harus ditanggapi pengguna terhadap prompt saat ini. Kartu Respons dapat berasal dari konfigurasi bot (di konsol Amazon Lex, pilih tombol pengaturan di sebelah slot) atau dari kait kode (fungsi Lambda).

Tipe: Objek ResponseCard

sentimentResponse

Sentimen yang diungkapkan dan diucapkan.

Ketika bot dikonfigurasi untuk mengirim ucapan ke Amazon Comprehend untuk analisis sentimen, bidang ini berisi hasil analisis.

Tipe: Objek SentimentResponse

sessionAttributes

Peta pasangan nilai kunci yang mewakili informasi konteks khusus sesi.

Tipe: Peta string ke string

sessionId

Pengenal unik untuk sesi tersebut.

Jenis: String

slots

Slot maksud yang dideteksi Amazon Lex dari input pengguna dalam percakapan.

Amazon Lex membuat daftar resolusi yang berisi kemungkinan nilai untuk slot. Nilai yang dikembalikan ditentukan oleh yang `valueSelectionStrategy` dipilih ketika jenis slot dibuat atau diperbarui. Jika `valueSelectionStrategy` diatur ke`ORIGINAL_VALUE`, nilai yang diberikan oleh pengguna dikembalikan, jika nilai pengguna mirip dengan nilai slot. Jika `valueSelectionStrategy` diatur ke `TOP_RESOLUTION` Amazon Lex mengembalikan nilai pertama dalam daftar resolusi atau, jika tidak ada daftar resolusi, null. Jika Anda tidak menentukan a`valueSelectionStrategy`, defaultnya adalah`ORIGINAL_VALUE`.

Tipe: Peta string ke string

slotToElicit

Jika `dialogState` nilainya`ElicitSlot`, mengembalikan nama slot yang Amazon Lex memunculkan nilai.

Jenis: String

Kesalahan

BadGatewayException

Entah bot Amazon Lex masih dibangun, atau salah satu layanan dependen (Amazon Polly, AWS Lambda) gagal dengan kesalahan layanan internal.

Kode Status HTTP: 502

BadRequestException

Validasi permintaan gagal, tidak ada pesan yang dapat digunakan dalam konteksnya, atau pembuatan bot gagal, masih dalam proses, atau berisi perubahan yang belum dibangun.

Kode Status HTTP: 400

ConflictException

Dua klien menggunakan akun AWS yang sama, bot Amazon Lex, dan ID pengguna.

Kode Status HTTP: 409

DependencyFailedException

Salah satu dependensi, seperti AWS Lambda atau Amazon Polly, memberikan pengecualian. Misalnya,

- Jika Amazon Lex tidak memiliki izin yang cukup untuk memanggil fungsi Lambda.
- Jika fungsi Lambda membutuhkan waktu lebih dari 30 detik untuk dijalankan.
- Jika fungsi Lambda pemenuhan mengembalikan `Delegate` tindakan dialog tanpa menghapus nilai slot apa pun.

Kode Status HTTP: 424

InternalFailureException

Kesalahan layanan internal. Coba lagi panggilannya.

Kode Status HTTP: 500

LimitExceededException

Melebihi batas.

Kode Status HTTP: 429

LoopDetectedException

Pengecualian ini tidak digunakan.

Kode Status HTTP: 508

NotFoundException

Sumber daya (seperti bot Amazon Lex atau alias) yang disebut tidak ditemukan.

Kode Status HTTP: 404

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript
- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

PutSession

Layanan: Amazon Lex Runtime Service

Membuat sesi baru atau memodifikasi sesi yang ada dengan bot Amazon Lex. Gunakan operasi ini untuk mengaktifkan aplikasi Anda untuk mengatur status bot.

Untuk informasi selengkapnya, lihat Mengelola Sesi.

Minta Sintaks

```
POST /bot/botName/alias/botAlias/user/userId/session HTTP/1.1
Accept: accept
Content-type: application/json

{
   "activeContexts": [
      {
         "name": "string",
         "parameters": {
            "string" : "string"
         },
         "timeToLive": {
            "timeToLiveInSeconds": number,
            "turnsToLive": number
         }
      }
   ],
   "dialogAction": {
      "fulfillmentState": "string",
      "intentName": "string",
      "message": "string",
      "messageFormat": "string",
      "slots": {
         "string" : "string"
      },
      "slotToElicit": "string",
      "type": "string"
   },
   "recentIntentSummaryView": [
      {
         "checkpointLabel": "string",
         "confirmationStatus": "string",
         "dialogActionType": "string",
```

```
        "fulfillmentState": "string",
        "intentName": "string",
        "slots": {
           "string" : "string"
        },
        "slotToElicit": "string"
     }
  ],
  "sessionAttributes": {
     "string" : "string"
  }
}
```

Parameter Permintaan URI

Permintaan menggunakan parameter URI berikut.

accept

Pesan yang dikembalikan Amazon Lex dalam respons dapat berupa teks atau ucapan berdasarkan tergantung pada nilai bidang ini.

- Jika nilainya`text/plain; charset=utf-8`, Amazon Lex mengembalikan teks dalam respons.
- Jika nilai dimulai dengan`audio/`, Amazon Lex mengembalikan pidato dalam respons. Amazon Lex menggunakan Amazon Polly untuk menghasilkan pidato dalam konfigurasi yang Anda tentukan. Misalnya, jika Anda menentukan `audio/mpeg` sebagai nilai, Amazon Lex mengembalikan ucapan dalam format MPEG.
- Jika nilainya`audio/pcm`, pidato dikembalikan seperti `audio/pcm` dalam 16-bit, format endian kecil.
- Berikut ini adalah nilai yang diterima:
  - `audio/mpeg`
  - `audio/ogg`
  - `audio/pcm`
  - `audio/*`(default ke mpeg)
  - `text/plain; charset=utf-8`

botAlias

Alias yang digunakan untuk bot yang berisi data sesi.

Diperlukan: Ya

botName

Nama bot yang berisi data sesi.

Diperlukan: Ya

userId

ID pengguna aplikasi klien. Amazon Lex menggunakan ini untuk mengidentifikasi percakapan pengguna dengan bot Anda.

Kendala Panjang: Panjang minimum 2. Panjang maksimum 100.

Pola: `[0-9a-zA-Z._:-]+`

Wajib: Ya

Isi Permintaan

Permintaan menerima data berikut dalam format JSON.

activeContexts

Daftar konteks yang aktif untuk permintaan. Konteks dapat diaktifkan ketika maksud sebelumnya terpenuhi, atau dengan memasukkan konteks dalam permintaan,

Jika Anda tidak menentukan daftar konteks, Amazon Lex akan menggunakan daftar konteks saat ini untuk sesi tersebut. Jika Anda menentukan daftar kosong, semua konteks untuk sesi akan dihapus.

Tipe: Array objek ActiveContext

Anggota Array: Jumlah minimum 0 item. Jumlah maksimum 20 item.

Diperlukan: Tidak

dialogAction

Menetapkan tindakan selanjutnya yang harus diambil bot untuk memenuhi percakapan.

Tipe: Objek DialogAction

Diperlukan: Tidak

recentIntentSummaryView

Ringkasan maksud terbaru untuk bot. Anda dapat menggunakan tampilan ringkasan intent untuk menyetel label pos pemeriksaan pada intent dan memodifikasi atribut intent. Anda juga dapat menggunakannya untuk menghapus atau menambahkan objek ringkasan maksud ke daftar.

Maksud yang Anda ubah atau tambahkan ke daftar harus masuk akal untuk bot. Misalnya, nama maksud harus valid untuk bot. Anda harus memberikan nilai yang valid untuk:

- `intentName`
- nama slot
- `slotToElict`

Jika Anda mengirim `recentIntentSummaryView` parameter dalam `PutSession` permintaan, isi tampilan ringkasan baru menggantikan tampilan ringkasan lama. Misalnya, jika `GetSession` permintaan mengembalikan tiga maksud dalam tampilan ringkasan dan Anda memanggil `PutSession` dengan satu maksud dalam tampilan ringkasan, panggilan berikutnya hanya `GetSession` akan menampilkan satu maksud.

Tipe: Array objek IntentSummary

Anggota Array: Jumlah minimum 0 item. Jumlah maksimum 3 item.

Diperlukan: Tidak

sessionAttributes

Peta pasangan kunci/nilai yang mewakili informasi konteks khusus sesi. Ini berisi informasi aplikasi yang diteruskan antara Amazon Lex dan aplikasi klien.

Tipe: Peta antar string

Diperlukan: Tidak

Sintaksis Respons

```
HTTP/1.1 200
Content-Type: contentType
x-amz-lex-intent-name: intentName
x-amz-lex-slots: slots
x-amz-lex-session-attributes: sessionAttributes
```

```
x-amz-lex-message: message
x-amz-lex-encoded-message: encodedMessage
x-amz-lex-message-format: messageFormat
x-amz-lex-dialog-state: dialogState
x-amz-lex-slot-to-elicit: slotToElicit
x-amz-lex-session-id: sessionId
x-amz-lex-active-contexts: activeContexts

audioStream
```

Elemen Respons

Jika tindakan berhasil, layanan mengirimkan kembali respons HTTP 200.

Respons mengembalikan header HTTP berikut.

activeContexts

Daftar konteks aktif untuk sesi tersebut.

contentType

Jenis konten seperti yang ditentukan dalam header `Accept` HTTP dalam permintaan.

dialogState

- `ConfirmIntent`- Amazon Lex mengharapkan respons “ya” atau “tidak” untuk mengonfirmasi maksud sebelum memenuhi maksud.
- `ElicitIntent`- Amazon Lex ingin mendapatkan maksud pengguna.
- `ElicitSlot`- Amazon Lex mengharapkan nilai slot untuk maksud saat ini.
- `Failed`- Menyampaikan bahwa percakapan dengan pengguna telah gagal. Hal ini dapat terjadi karena berbagai alasan, termasuk pengguna tidak memberikan respons yang sesuai terhadap permintaan dari layanan, atau jika fungsi Lambda gagal memenuhi intent.
- `Fulfilled`- Menyampaikan bahwa fungsi Lambda telah berhasil memenuhi niat.
- `ReadyForFulfillment`- Menyampaikan bahwa klien harus memenuhi niat.

Nilai yang Valid: `ElicitIntent | ConfirmIntent | ElicitSlot | Fulfilled | ReadyForFulfillment | Failed`

encodedMessage

Pesan berikutnya yang harus disajikan kepada pengguna.

`encodedMessage`Bidang ini dikodekan basis-64. Anda harus memecahkan kode bidang sebelum Anda dapat menggunakan nilai.

Batasan Panjang: Panjang minimum 1. Panjang maksimum 1366.

intentName

Nama maksud saat ini.

message

Header ini sudah tidak digunakan lagi.

Pesan berikutnya yang harus disajikan kepada pengguna.

Anda hanya dapat menggunakan bidang ini di lokal De-de, en-AU, en-GB, en-US, es-419, es-ES, es-US, fr-Ca, fr-FR, dan IT-it. Di semua lokal lainnya, `message` bidangnya nol. Anda harus menggunakan `encodedMessage` bidang sebagai gantinya.

Batasan Panjang: Panjang minimum 1. Panjang maksimum 1024.

messageFormat

Format pesan respons. Salah satu nilai berikut:

- `PlainText`- Pesan berisi teks UTF-8 biasa.
- `CustomPayload`- Pesan adalah format khusus untuk klien.
- `SSML`- Pesan berisi teks yang diformat untuk output suara.
- `Composite`- Pesan berisi objek JSON yang lolos yang berisi satu atau beberapa pesan dari grup tempat pesan ditetapkan saat maksud dibuat.

Nilai yang Valid: `PlainText | CustomPayload | SSML | Composite`

sessionAttributes

Peta pasangan kunci/nilai yang mewakili informasi konteks khusus sesi.

sessionId

Pengenal unik untuk sesi tersebut.

slots

Peta slot maksud nol atau lebih Amazon Lex terdeteksi dari input pengguna selama percakapan.

Amazon Lex membuat daftar resolusi yang berisi kemungkinan nilai untuk slot. Nilai yang dikembalikan ditentukan oleh yang `valueSelectionStrategy` dipilih ketika jenis slot

dibuat atau diperbarui. Jika `valueSelectionStrategy` diatur ke`ORIGINAL_VALUE`, nilai yang diberikan oleh pengguna dikembalikan, jika nilai pengguna mirip dengan nilai slot. Jika `valueSelectionStrategy` diatur ke `TOP_RESOLUTION` Amazon Lex mengembalikan nilai pertama dalam daftar resolusi atau, jika tidak ada daftar resolusi, null. Jika Anda tidak menentukan default adalah`ORIGINAL_VALUE`. `valueSelectionStrategy`

slotToElicit

Jika `dialogState` ya`ElicitSlot`, mengembalikan nama slot yang Amazon Lex memunculkan nilai.

Respons mengembalikan yang berikut sebagai isi HTTP.

audioStream

Versi audio dari pesan untuk disampaikan kepada pengguna.

Kesalahan

BadGatewayException

Entah bot Amazon Lex masih dibangun, atau salah satu layanan dependen (Amazon Polly, AWS Lambda) gagal dengan kesalahan layanan internal.

Kode Status HTTP: 502

BadRequestException

Validasi permintaan gagal, tidak ada pesan yang dapat digunakan dalam konteksnya, atau pembuatan bot gagal, masih dalam proses, atau berisi perubahan yang belum dibangun.

Kode Status HTTP: 400

ConflictException

Dua klien menggunakan akun AWS yang sama, bot Amazon Lex, dan ID pengguna.

Kode Status HTTP: 409

DependencyFailedException

Salah satu dependensi, seperti AWS Lambda atau Amazon Polly, memberikan pengecualian. Misalnya,

- Jika Amazon Lex tidak memiliki izin yang cukup untuk memanggil fungsi Lambda.
- Jika fungsi Lambda membutuhkan waktu lebih dari 30 detik untuk dijalankan.
- Jika fungsi Lambda pemenuhan mengembalikan `Delegate` tindakan dialog tanpa menghapus nilai slot apa pun.

Kode Status HTTP: 424

InternalFailureException

Kesalahan layanan internal. Coba lagi panggilannya.

Kode Status HTTP: 500

LimitExceededException

Melebihi batas.

Kode Status HTTP: 429

NotAcceptableException

Header terima dalam permintaan tidak memiliki nilai yang valid.

Kode Status HTTP: 406

NotFoundException

Sumber daya (seperti bot Amazon Lex atau alias) yang disebut tidak ditemukan.

Kode Status HTTP: 404

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat berikut ini:

- AWS Antarmuka Baris Perintah
- AWS SDK for .NET
- AWS SDK for C++
- AWS SDK for Go
- AWS SDK for Java V2
- AWS SDK untuk V3 JavaScript

- AWS SDK for PHP V3
- AWS SDK untuk Python
- AWS SDK for Ruby V3

Tipe Data

tipe data berikut didukung oleh Amazon Lex Model Building Service:

- BotAliasMetadata
- BotChannelAssociation
- BotMetadata
- BuiltinIntentMetadata
- BuiltinIntentSlot
- BuiltinSlotTypeMetadata
- CodeHook
- ConversationLogsRequest
- ConversationLogsResponse
- EnumerationValue
- FollowUpPrompt
- FulfillmentActivity
- InputContext
- Intent
- IntentMetadata
- KendraConfiguration
- LogSettingsRequest
- LogSettingsResponse
- Message
- MigrationAlert
- MigrationSummary
- OutputContext
- Prompt

- ResourceReference
- Slot
- SlotDefaultValue
- SlotDefaultValueSpec
- SlotTypeConfiguration
- SlotTypeMetadata
- SlotTypeRegexConfiguration
- Statement
- Tag
- UtteranceData
- UtteranceList

tipe data berikut didukung oleh Amazon Lex Runtime Service:

- ActiveContext
- ActiveContextTimeToLive
- Button
- DialogAction
- GenericAttachment
- IntentConfidence
- IntentSummary
- PredictedIntent
- ResponseCard
- SentimentResponse

Amazon Lex Model Bangunan Layanan

tipe data berikut didukung oleh Amazon Lex Model Building Service:

- BotAliasMetadata
- BotChannelAssociation
- BotMetadata

- BuiltinIntentMetadata
- BuiltinIntentSlot
- BuiltinSlotTypeMetadata
- CodeHook
- ConversationLogsRequest
- ConversationLogsResponse
- EnumerationValue
- FollowUpPrompt
- FulfillmentActivity
- InputContext
- Intent
- IntentMetadata
- KendraConfiguration
- LogSettingsRequest
- LogSettingsResponse
- Message
- MigrationAlert
- MigrationSummary
- OutputContext
- Prompt
- ResourceReference
- Slot
- SlotDefaultValue
- SlotDefaultValueSpec
- SlotTypeConfiguration
- SlotTypeMetadata
- SlotTypeRegexConfiguration
- Statement
- Tag
- UtteranceData

- UtteranceList

BotAliasMetadata

Layanan:Amazon Lex Model Building Service

Menyediakan informasi tentang alias bot.

Daftar Isi

botName

Nama bot yang menunjuk alias.

Jenis: String

Batasan Panjang: Panjang minimum 2. Panjang maksimum 50.

Pola: `^([A-Za-z]_?)+$`

Diperlukan: Tidak

botVersion

Versi bot Amazon Lex yang menunjuk alias.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum adalah 64.

Pola: `\$LATEST|[0-9]+`

Diperlukan: Tidak

checksum

Checksum dari alias bot.

Tipe: String

Wajib: Tidak

conversationLogs

Pengaturan yang menentukan cara Amazon Lex menggunakan log percakapan untuk alias.

Tipe: Objek ConversationLogsResponse

Wajib: Tidak

createdDate

Tanggal bahwa alias bot dibuat.

Tipe: Timestamp

Wajib: Tidak

description

Deskripsi alias bot.

Jenis: String

Batasan Panjang: Panjang minimum 0. Panjang maksimum 200.

Wajib: Tidak

lastUpdatedDate

Tanggal bahwa alias bot diperbarui. Saat Anda membuat sumber daya, tanggal diperbarui terakhir.

Tipe: Timestamp

Wajib: Tidak

name

Nama alias.

Jenis: String

Panjang Batasan: Panjang minimum 1. Panjang maksimum 100.

Pola: `^([A-Za-z]_?)+$`

Diperlukan: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API di salah satuAWS SDK khusus bahasa, lihat yang berikut ini:

- AWSSDK for C++

- AWSSDK for Go
- AWSSDK for Java V2
- AWSSDK for Ruby V3

BotChannelAssociation

Layanan: Amazon Lex Model Building Service

Merupakan hubungan antara bot Amazon Lex dan platform perpesanan eksternal.

Daftar Isi

botAlias

Alias yang menunjuk ke versi spesifik bot Amazon Lex tempat asosiasi ini dibuat.

Jenis: String

Panjang Batasan: Panjang minimum 1. Panjang maksimum 100.

Pola: `^([A-Za-z]_?)+$`

Wajib: Tidak

botConfiguration

Menyediakan informasi yang diperlukan untuk berkomunikasi dengan platform perpesanan.

Tipe: Peta string ke string

Entri Peta: Jumlah maksimum 10 item.

Wajib: Tidak

botName

Nama bot Amazon Lex tempat asosiasi ini dibuat.

> **Note**
>
> Saat ini, Amazon Lex mendukung asosiasi dengan Facebook dan Slack, dan Twilio.

Jenis: String

Panjang Batasan: Panjang minimum 2. Panjang maksimum 50.

Pola: `^([A-Za-z]_?)+$`

Wajib: Tidak

createdDate

Tanggal hubungan antara bot Amazon Lex dan saluran dibuat.

Tipe: Timestamp

Wajib: Tidak

description

Deskripsi teks terkait yang Anda buat.

Jenis: String

Batasan Panjang: Panjang minimum 0. Panjang maksimum 200.

Wajib: Tidak

failureReason

Jika `status` ya`FAILED`, Amazon Lex memberikan alasan bahwa ia gagal untuk membuat asosiasi.

Tipe: String

Wajib: Tidak

name

Nama pengaitan antara bot dan saluran.

Jenis: String

Panjang Batasan: Panjang minimum 1. Panjang maksimum 100.

Pola: `^([A-Za-z]_?)+$`

Wajib: Tidak

status

Status saluran bot.

- `CREATED`- Saluran telah dibuat dan siap untuk digunakan.
- `IN_PROGRESS`- Pembuatan saluran sedang berlangsung.

- `FAILED`- Ada kesalahan saat membuat saluran. Untuk informasi tentang alasan kegagalan, lihat `failureReason` bidang.

Jenis: String

Nilai yang Valid: `IN_PROGRESS | CREATED | FAILED`

Wajib: Tidak

type

Menentukan jenis asosiasi dengan menunjukkan jenis saluran yang dibuat antara bot Amazon Lex dan platform perpesanan eksternal.

Jenis: String

Nilai yang Valid: `Facebook | Slack | Twilio-Sms | Kik`

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API di salah satu AWS SDK khusus bahasa, lihat yang berikut ini:

- AWSSDK for C++
- AWSSDK for Go
- AWSSDK for Java V2
- AWSSDK for Ruby V3

BotMetadata

Layanan: Amazon Lex Model Building Service

Menyediakan informasi tentang bot.

Daftar Isi

createdDate

Tanggal bot dibuat.

Tipe: Timestamp

Wajib: Tidak

description

Deskripsi bot.

Jenis: String

Batasan Panjang: Panjang minimum 0. Panjang maksimum 200.

Wajib: Tidak

lastUpdatedDate

Tanggal bahwa bot diperbarui. Saat Anda membuat bot, tanggal pembuatan dan tanggal diperbarui terakhir adalah sama.

Tipe: Timestamp

Wajib: Tidak

name

Nama bot.

Jenis: String

Batas Panjang: Panjang minimum 2. Panjang maksimum 50.

Pola: `^([A-Za-z]_?)+$`

Wajib: Tidak

status

Status bot.

Jenis: String

Nilai yang Valid: `BUILDING | READY | READY_BASIC_TESTING | FAILED | NOT_BUILT`

Wajib: Tidak

version

Versi bot. Untuk bot baru, versinya selalu`$LATEST`.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum adalah 64.

Pola: `\$LATEST|[0-9]+`

Diperlukan: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API di salah satu AWS SDK khusus bahasa, lihat yang berikut ini:

- AWSSDK for C++
- AWSSDK for Go
- AWSSDK for Java V2
- AWSSDK for Ruby V3

BuiltinIntentMetadata

Layanan: Amazon Lex Model Building Service

Menyediakan metadata untuk intent bawaan.

Daftar Isi

signature

Pengenal unik untuk intent unik untuk intent yang ada. Untuk menemukan tanda tangan untuk maksud, lihat Intent Bawaan Standar dalam Kit Keterampilan Alexa.

Tipe: String

Wajib: Tidak

supportedLocales

Daftar pengenal untuk lokal yang didukung oleh pengguna.

Tipe: Array string

Nilai yang Valid: `de-DE | en-AU | en-GB | en-IN | en-US | es-419 | es-ES | es-US | fr-FR | fr-CA | it-IT | ja-JP | ko-KR`

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API di salah satu AWS SDK khusus bahasa, lihat yang berikut ini:

- AWSSDK for C++
- AWSSDK for Go
- AWSSDK for Java V2
- AWSSDK for Ruby V3

BuiltinIntentSlot

Layanan:Amazon Lex Model Building Service

Memberikan informasi tentang slot yang digunakan dalam maksud bawaan.

Daftar Isi

name

Daftar slot didefinisikan untuk maksud.

Tipe: String

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API di salah satu SDK di salah satu SDK di salah satu SDK di salah satu SDK di salah satuAWS SDK khusus bahasa, lihat yang berikut ini:

- AWSSDK for C++
- AWSSDK for Go
- AWSSDK for Java V2
- AWSSDK for Ruby V3

BuiltinSlotTypeMetadata

Layanan: Amazon Lex Model Building Service

Menyediakan informasi tentang dibangun di jenis Slot.

Daftar Isi

signature

Pengidentifikasi unik untuk jenis slot built-in. Untuk menemukan tanda tangan untuk jenis slot, lihat Referensi Jenis Slot di Alexa Skills Kit.

Tipe: String

Wajib: Tidak

supportedLocales

Daftar lokal target untuk slot.

Tipe: Array string

Nilai yang Valid: `de-DE | en-AU | en-GB | en-IN | en-US | es-419 | es-ES | es-US | fr-FR | fr-CA | it-IT | ja-JP | ko-KR`

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API di salah satu AWS SDK SDK khusus bahasa, lihat yang berikut ini:

- AWSSDK for C++
- AWSSDK for Go
- AWSSDK for Java V2
- AWSSDK for Ruby V3

CodeHook

Layanan:Amazon Lex Model Building Service

Menentukan fungsi Lambda yang memverifikasi permintaan ke bot atau memenuhi permintaan pengguna ke bot..

Daftar Isi

messageVersion

Versi respons permintaan yang Anda inginkan Amazon Lex gunakan untuk menjalankan fungsi Lambda Anda. Untuk informasi selengkapnya, lihat Menggunakan Fungsi Lambda.

Tipe: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 5.

Wajib: Ya

uri

Amazon Resource Name (ARN) fungsi Lambda.

Jenis: String

Batasan Panjang: Panjang minimum 20. Panjang maksimum 2048.

Pola: `arn:aws[a-zA-Z-]*:lambda:[a-z]+-[a-z]+(-[a-z]+)*-[0-9]:[0-9]{12}:function:[a-zA-Z0-9-_]+(\/[0-9a-f]{8}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{4}-[0-9a-f]{12})?(:[a-zA-Z0-9-_]+)?`

Diperlukan: Ya

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API di salah satuAWS SDK khusus bahasa, lihat yang berikut ini:

- AWSSDK for C++
- AWSSDK for Go
- AWSSDK for Java V2

- AWSSDK for Ruby V3

ConversationLogsRequest

Layanan:Amazon Lex Model Building Service

Menyediakan pengaturan yang diperlukan untuk log percakapan.

Daftar Isi

iamRoleArn

Amazon Resource Name (ARN) dari IAM role yang memiliki izin untuk menulis ke Amazon Resource Name (ARN) dari IAM role yang memiliki izin untuk menulis ke CloudWatch Logs dan bucket S3 Anda untuk IAM role yang memiliki izin untuk menulis ke catatan teks dan bucket S3 Anda untuk IAM Jika enkripsi audio diaktifkan, peran ini juga memberikan izin akses untuk kunci AWS KMS yang digunakan untuk mengenkripsi log audio. Untuk informasi selengkapnya, lihat Membuat Peran dan Kebijakan IAM untuk Log Percakapan.

Jenis: String

Batasan Panjang: Panjang minimum 20. Panjang maksimum 2048.

Pola: `^arn:[\w\-]+:iam::[\d]{12}:role/.+$`

Diperlukan: Ya

logSettings

Pengaturan untuk log percakapan Anda. Anda dapat mencatat teks percakapan, audio percakapan, atau keduanya.

Tipe: Array objek LogSettingsRequest

Wajib: Ya

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satuAWS SDK khusus bahasa, lihat yang berikut ini:

- AWSSDK for C++ C++
- AWSSDK for Go
- AWSSDK for Java V2 2 for Java V2

- AWSSDK for Ruby V3 Ruby V3 for Ruby Ruby V

ConversationLogsResponse

Layanan:Amazon Lex Model Building Service

Berisi informasi tentang pengaturan log percakapan.

Daftar Isi

iamRoleArn

Amazon Resource Name (ARN) IAM role yang digunakan untuk menulis CloudWatch Logs atau bucket S3.

Jenis: String

Batasan Panjang: Panjang minimum 20. Panjang maksimum 2048.

Pola: `^arn:[\w\-]+:iam::[\d]{12}:role/.+$`

Diperlukan: Tidak

logSettings

Pengaturan untuk log percakapan Anda. Anda dapat mencatat teks, audio, atau keduanya.

Tipe: Array objek LogSettingsResponse

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API di salah satuAWS SDK khusus bahasa, lihat berikut ini:

- AWSSDK for C++
- AWSSDK for Go
- AWSSDK for Java V2
- AWSSDK for Ruby V3

EnumerationValue

Layanan: Amazon Lex Model Building Service

Setiap jenis slot dapat memiliki satu set nilai. Setiap nilai pencacahan merupakan nilai jenis slot dapat mengambil.

Misalnya, bot pemesanan pizza bisa memiliki jenis slot yang menentukan jenis kerak yang seharusnya dimiliki pizza. Jenis slot dapat mencakup nilai-nilai

- tebal
- tipis
- penuh sesak

Daftar Isi

value

Nilai dari jenis slot.

Jenis: String

Panjang Batasan: Panjang minimum 1. Panjang maksimum 140.

Wajib: Ya

synonyms

Nilai tambahan yang terkait dengan nilai jenis slot.

Tipe: Array string

Panjang Batasan: Panjang minimum 1. Panjang maksimum 140.

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat yang berikut ini:

- AWSSDK untuk C++

- AWSSDK untuk Go
- AWSSDK untuk Java V2
- AWSSDK para Ruby V3

FollowUpPrompt

Layanan:Amazon Lex Model Building Service

Permintaan untuk aktivitas tambahan setelah intent terpenuhi. Misalnya, setelah`OrderPizza` intent terpenuhi, Anda mungkin meminta pengguna untuk mengetahui apakah pengguna ingin memesan minuman.

Daftar Isi

prompt

Meminta informasi dari pengguna.

Tipe: Objek Prompt

Wajib: Ya

rejectionStatement

Jika pengguna menjawab “tidak” untuk pertanyaan yang didefinisikan di`prompt` lapangan, Amazon Lex merespons dengan pernyataan ini untuk mengakui bahwa maksud tersebut dibatalkan.

Tipe: Objek Statement

Wajib: Ya

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API di salah satuAWS SDK khusus bahasa, lihat yang berikut ini:

- AWSSDK for C++
- AWSSDK for Go
- AWSSDK for Java V2
- AWSSDK for Ruby V3

FulfillmentActivity

Layanan: Amazon Lex Model Building Service

Menjelaskan bagaimana maksud terpenuhi setelah pengguna menyediakan semua informasi yang diperlukan untuk maksud tersebut. Anda dapat menyediakan fungsi Lambda untuk memproses maksud, atau Anda dapat mengembalikan informasi maksud ke aplikasi klien. Kami menyarankan Anda menggunakan fungsi Lambda sehingga logika yang relevan berada di Cloud dan membatasi kode sisi klien terutama untuk presentasi. Jika Anda perlu memperbarui logika, Anda hanya memperbarui fungsi Lambda; Anda tidak perlu meng-upgrade aplikasi klien Anda.

Pertimbangkan contoh berikut:

- Dalam aplikasi pemesanan pizza, setelah pengguna memberikan semua informasi untuk melakukan pemesanan, Anda menggunakan fungsi Lambda untuk memesan dengan restoran pizza.
- Dalam aplikasi game, ketika pengguna mengatakan “mengambil batu,” informasi ini harus kembali ke aplikasi klien sehingga dapat melakukan operasi dan memperbarui grafik. Dalam hal ini, Anda ingin Amazon Lex mengembalikan data maksud ke klien.

Daftar Isi

type

Bagaimana maksud harus dipenuhi, baik dengan menjalankan fungsi Lambda atau dengan mengembalikan data slot ke aplikasi klien.

Jenis: String

Nilai yang Valid: `ReturnIntent | CodeHook`

Wajib: Ya

codeHook

Deskripsi fungsi Lambda yang dijalankan untuk memenuhi maksud.

Tipe: Objek CodeHook

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat yang berikut ini:

- AWSSDK untuk C++
- AWSSDK untuk Go
- AWSSDK untuk Java V2
- AWSSDK para Ruby V3

InputContext

Layanan:Amazon Lex Model Building Service

Nama konteks yang harus aktif agar maksud dipilih oleh Amazon Lex.

Daftar Isi

name

Nama konteksnya.

Jenis: String

Panjang Batasan: Panjang minimum 1. Panjang maksimum 100.

Pola: `^([A-Za-z]_?)+$`

Diperlukan: Ya

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API di salah satuAWS SDK khusus bahasa, lihat yang berikut ini:

- AWSSDK for C++
- AWSSDK for Go
- AWSSDK for Java V2
- AWSSDK for Ruby V3

Intent

Layanan:Amazon Lex Model Building Service

Mengidentifikasi versi intent tertentu.

Daftar Isi

intentName

Nama dari maksud.

Jenis: String

Panjang Batasan: Panjang minimum 1. Panjang maksimum 100.

Pola: `^([A-Za-z]_?)+$`

Diperlukan: Ya

intentVersion

Versi maksud.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum adalah 64.

Pola: `\$LATEST|[0-9]+`

Diperlukan: Ya

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API di salah satuAWS SDK khusus bahasa, lihat yang berikut ini:

- AWSSDK for C++
- AWSSDK for Go
- AWSSDK for Java V2
- AWSSDK for Ruby V3

IntentMetadata

Layanan:Amazon Lex Model Building Service

Memberikan informasi tentang maksud.

Daftar Isi

createdDate

Tanggal maksud dibuat.

Tipe: Timestamp

Wajib: Tidak

description

Deskripsi maksud.

Jenis: String

Batasan Panjang: Panjang minimum 0. Panjang maksimum 200.

Wajib: Tidak

lastUpdatedDate

Tanggal bahwa maksud diperbarui. Saat Anda membuat maksud, tanggal pembuatan dan tanggal diperbarui terakhir.

Tipe: Timestamp

Wajib: Tidak

name

Nama maksud.

Jenis: String

Panjang Batasan: Panjang minimum 1. Panjang maksimum 100.

Pola: `^([A-Za-z]_?)+$`

Diperlukan: Tidak

version

Versi maksud.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum adalah 64.

Pola: `\$LATEST|[0-9]+`

Diperlukan: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API di salah satuAWS SDK khusus bahasa, lihat yang berikut ini:

- AWSSDK for C++
- AWSSDK for Go
- AWSSDK for Java V2
- AWSSDK for Ruby V3

KendraConfiguration

Layanan: Amazon Lex Model Building Service

Menyediakan informasi konfigurasi untuk AMAZON. KendraSearchIntentniat. Saat Anda menggunakan maksud ini, Amazon Lex akan menelusuri indeks Amazon Kendra yang ditentukan dan mengembalikan dokumen dari indeks yang cocok dengan ucapan pengguna. Untuk informasi selengkapnya, lihat AMAZON. KendraSearchIntent.

Daftar Isi

kendraIndex

Amazon Resource Name (ARN) dari indeks Amazon Kendra yang Anda inginkan AMAZON. KendraSearchIntentmaksud untuk mencari. Indeks harus berada di akun dan Wilayah yang sama dengan bot Amazon Lex. Jika indeks Amazon Kendra tidak ada, Anda mendapatkan pengecualian saat Anda memanggil `PutIntent` operasi.

Jenis: String

Batasan Panjang: Panjang minimum 20. Panjang maksimum 2048.

Pola: `arn:aws:kendra:[a-z]+-[a-z]+-[0-9]:[0-9]{12}:index\/[a-zA-Z0-9][a-zA-Z0-9_-]*`

Diperlukan: Ya

role

Amazon Resource Name (ARN) dari peran IAM yang memiliki izin untuk mencari indeks Amazon Kendra. Peran harus berada di akun dan Wilayah yang sama dengan bot Amazon Lex. Jika peran tidak ada, Anda mendapatkan pengecualian saat Anda memanggil `PutIntent` operasi.

Jenis: String

Batasan Panjang: Panjang minimum 20. Panjang maksimum 2048.

Pola: `arn:aws:iam::[0-9]{12}:role/.*`

Diperlukan: Ya

queryFilterString

Filter kueri yang dikirim Amazon Lex ke Amazon Kendra untuk memfilter respons dari kueri. Filter dalam format yang ditentukan oleh Amazon Kendra. Untuk informasi selengkapnya, lihat Memfilter kueri.

Anda dapat mengganti string filter ini dengan string filter baru saat runtime.

Jenis: String

Batasan Panjang: Panjang minimum 0.

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat yang berikut ini:

- AWSSDK untuk C++
- AWSSDK untuk Go
- AWSSDK untuk Java V2
- AWSSDK para Ruby V3

LogSettingsRequest

Layanan: Amazon Lex Model Building Service

Pengaturan yang digunakan untuk mengkonfigurasi mode pengiriman dan tujuan untuk log percakapan.

Daftar Isi

destination

Dimana log akan dikirimkan. Log teks dikirim ke grup CloudWatch log Log. Log audio dikirim ke bucket S3.

Jenis: String

Nilai yang Valid: `CLOUDWATCH_LOGS | S3`

Wajib: Ya

logType

Jenis logging untuk mengaktifkan. Log teks dikirim ke grup CloudWatch log Log. Log audio dikirim ke bucket S3.

Jenis: String

Nilai yang Valid: `AUDIO | TEXT`

Wajib: Ya

resourceArn

Amazon Resource Name (ARN) dari grup CloudWatch log Log atau bucket S3 tempat log harus dikirimkan.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Pola: `^arn:[\w\-]+:(?:logs:[\w\-]+:[\d]{12}:log-group:[\.\-_/#A-Za-z0-9]{1,512}(?::\*)?|s3:::[a-z0-9][\.\-a-z0-9]{1,61}[a-z0-9])$`

Diperlukan: Ya

kmsKeyArn

Amazon Resource Name (ARN) dari kunci yang dikelola pelanggan AWS KMS untuk mengenkripsi log audio yang dikirim ke bucket S3. Kuncinya tidak berlaku untuk CloudWatch Log dan opsional untuk bucket S3.

Jenis: String

Batasan Panjang: Panjang minimum 20. Panjang maksimum 2048.

Pola: `^arn:[\w\-]+:kms:[\w\-]+:[\d]{12}:(?:key\/[\w\-]+|alias\/[a-zA-Z0-9:\/_\-]{1,256})$`

Diperlukan: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat yang berikut ini:

- AWSSDK untuk C++
- AWSSDK untuk Go
- AWSSDK untuk Java V2
- AWSSDK para Ruby V3

LogSettingsResponse

Layanan: Amazon Lex Model Building Service

Pengaturan untuk log percakapan.

Daftar Isi

destination

Tujuan di mana log dikirimkan.

Jenis: String

Nilai yang Valid: `CLOUDWATCH_LOGS | S3`

Wajib: Tidak

kmsKeyArn

Amazon Resource Name (ARN) dari kunci yang digunakan untuk mengenkripsi log audio di Bucket S3.

Jenis: String

Batasan Panjang: Panjang minimum 20. Panjang maksimum 2048.

Pola: `^arn:[\w\-]+:kms:[\w\-]+:[\d]{12}:(?:key\/[\w\-]+|alias\/[a-zA-Z0-9:\/_\-]{1,256})$`

Wajib: Tidak

logType

Jenis logging yang diaktifkan.

Jenis: String

Nilai yang Valid: `AUDIO | TEXT`

Wajib: Tidak

resourceArn

Amazon Resource Name (ARN) dari grup CloudWatch log atau Bucket S3 tempat log dikirimkan.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Pola: `^arn:[\w\-]+:(?:logs:[\w\-]+:[\d]{12}:log-group:[\.\-_/#A-Za-z0-9]{1,512}(?::\*)?|s3:::[a-z0-9][\.\-a-z0-9]{1,61}[a-z0-9])$`

Wajib: Tidak

resourcePrefix

Awalan sumber daya adalah bagian pertama dari kunci objek S3 dalam bucket S3 yang Anda tentukan untuk berisi log audio. Untuk CloudWatch Log, ini adalah awalan nama aliran log dalam grup log yang Anda tentukan.

Jenis: String

Batasan Panjang: Panjang maksimum 1024.

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API di salah satu AWS SDK, lihat berikut ini:

- AWSSDK for C++
- AWSSDK for Go
- AWSSDK for Java
- AWSSDK for Ruby V3

Message

Layanan: Amazon Lex Model Building Service

Objek pesan yang menyediakan teks pesan dan jenisnya.

Daftar Isi

content

Teks pesan.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 1000.

Wajib: Ya

contentType

Jenis konten dari string pesan.

Jenis: String

Nilai yang Valid: `PlainText | SSML | CustomPayload`

Wajib: Ya

groupNumber

Mengidentifikasi grup pesan yang dimiliki pesan. Ketika grup ditugaskan ke pesan, Amazon Lex mengembalikan satu pesan dari setiap grup dalam respons.

Tipe: Integer

Rentang yang Valid: Nilai minimum 1. Nilai maksimum 5.

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API di salah satu AWS SDK khusus bahasa, lihat yang berikut ini:

- AWSSDK for C++

- AWSSDK for Go
- AWSSDK for Java V2
- AWSSDK for Ruby V3

MigrationAlert

Layanan: Amazon Lex Model Building Service

Menyediakan informasi tentang peringatan dan peringatan yang dikirim Amazon Lex selama migrasi. Peringatan tersebut mencakup informasi tentang cara mengatasi masalah.

Daftar Isi

details

Detail tambahan tentang peringatan.

Tipe: Array string

Wajib: Tidak

message

Pesan yang menjelaskan mengapa peringatan dikeluarkan.

Tipe: String

Wajib: Tidak

referenceURLs

Tautan ke dokumentasi Amazon Lex yang menjelaskan cara menyelesaikan peringatan.

Tipe: Array string

Wajib: Tidak

type

Jenis peringatan. Ada dua jenis peringatan:

- `ERROR`- Ada masalah dengan migrasi yang tidak dapat diselesaikan. Migrasi berhenti.
- `WARN`- Ada masalah dengan migrasi yang memerlukan perubahan manual ke bot Amazon Lex V2 baru. Migrasi berlanjut.

Jenis: String

Nilai yang Valid: `ERROR | WARN`

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API di salah satu AWS SDK khusus bahasa, lihat yang berikut ini:

- AWSSDK for C++
- AWSSDK for Go
- AWSSDK for Java V2
- AWSSDK for Ruby V3

MigrationSummary

Layanan: Amazon Lex Model Building Service

Memberikan informasi tentang memigrasi bot dari Amazon Lex V1 ke Amazon Lex V2.

Daftar Isi

migrationId

Pengenal unik yang ditetapkan Amazon Lex ke migrasi.

Jenis: String

Batas Panjang: Panjang: Panjang: Panjang: Panjang Batasan: Panjang: Panjang: Panjang: Panjang: Panjang

Pola: `^[0-9a-zA-Z]+$`

Wajib: Tidak

migrationStatus

Status operasi. Ketika status bot tersedia di Amazon Lex V2. `COMPLETE` Mungkin ada peringatan dan peringatan yang perlu diselesaikan untuk menyelesaikan migrasi.

Jenis: String

Nilai yang Valid: `IN_PROGRESS | COMPLETED | FAILED`

Wajib: Tidak

migrationStrategy

Strategi yang digunakan untuk melakukan migrasi.

Jenis: String

Nilai yang Valid: `CREATE_NEW | UPDATE_EXISTING`

Wajib: Tidak

migrationTimestamp

Tanggal dan waktu ketika migrasi dimulai.

Tipe: Timestamp

Wajib: Tidak

v1BotLocale

Lokal bot Amazon Lex V1 yang merupakan sumber migrasi.

Jenis: String

Nilai yang Valid: `de-DE | en-AU | en-GB | en-IN | en-US | es-419 | es-ES | es-US | fr-FR | fr-CA | it-IT | ja-JP | ko-KR`

Wajib: Tidak

v1BotName

Nama bot Amazon Lex V1 yang merupakan sumber migrasi.

Jenis: String

Batas Panjang: Panjang: Panjang: Panjang: Panjang: Panjang: Panjang: Panjang: Panjang: Panjang: Panjang Panjang maksimum 50.

Pola: `^([A-Za-z]_?)+$`

Wajib: Tidak

v1BotVersion

Versi bot Amazon Lex V1 yang merupakan sumber migrasi.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum adalah 64.

Pola: `\$LATEST|[0-9]+`

Wajib: Tidak

v2BotId

Pengenal unik Amazon Lex V2 yang merupakan tujuan migrasi.

Jenis: String

Batas Panjang: Panjang: Panjang: Panjang: Panjang Batasan: Panjang: Panjang: Panjang: Panjang: Panjang

Pola: `^[0-9a-zA-Z]+$`

Wajib: Tidak

v2BotRole

Peran IAM yang digunakan Amazon Lex untuk menjalankan bot Amazon Lex V2.

Jenis: String

Batasan Panjang: Panjang minimum 20. Panjang maksimum 2048.

Pola: `^arn:[\w\-]+:iam::[\d]{12}:role/.+$`

Diperlukan: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API di salah satu AWS SDK khusus bahasa, lihat yang berikut ini:

- AWSSDK for C++
- AWSSDK for Go
- AWSSDK for Java V2
- AWSSDK for Ruby V3

OutputContext

Layanan:Amazon Lex Model Building Service

Spesifikasi konteks keluaran yang disetel saat maksud terpenuhi.

Daftar Isi

name

Nama konteksnya.

Jenis: String

Panjang Batasan: Panjang minimum 1. Panjang maksimum 100.

Pola: `^([A-Za-z]_?)+$`

Diperlukan: Ya

timeToLiveInSeconds

Jumlah detik bahwa konteks harus aktif setelah pertama kali dikirim dalam`PostContent` atau`PostText` respon. Anda dapat mengatur nilai antara 5 dan 86.400 detik (24 jam).

Tipe: Bilangan Bulat

Rentang yang Valid: Nilai minimum 5. Nilai maksimum 86400.

Wajib: Ya

turnsToLive

Jumlah percakapan ternyata konteksnya harus aktif. Giliran percakapan adalah satu`PostContent` atau`PostText` permintaan dan respons yang sesuai dari Amazon Lex.

Tipe: Bilangan Bulat

Rentang yang Valid: Nilai minimum 1. Nilai maksimum 20.

Wajib: Ya

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API di salah satuAWS SDK khusus bahasa, lihat yang berikut ini:

- AWSSDK for C++
- AWSSDK for Go
- AWSSDK for Java V2
- AWSSDK for Ruby V3

Prompt

Layanan:Amazon Lex Model Building Service

Memperoleh informasi dari pengguna. Untuk menentukan prompt, berikan satu atau lebih pesan dan tentukan jumlah upaya untuk mendapatkan informasi dari pengguna. Jika Anda memberikan lebih dari satu pesan, Amazon Lex memilih salah satu pesan yang akan digunakan untuk meminta pengguna. Untuk informasi selengkapnya, lihat Amazon Lex: Cara Kerjanya.

Daftar Isi

maxAttempts

Berapa kali untuk meminta pengguna untuk informasi.

Tipe: Bilangan Bulat

Rentang yang Valid: Nilai minimum 1. Nilai maksimum 5.

Wajib: Ya

messages

Array objek, yang masing-masing menyediakan string pesan dan jenisnya. Anda dapat menentukan string pesan dalam teks biasa atau di Bahasa Markup Sintesis Ucapan (SSKL).

Tipe: Array objek Message

Anggota Array: Jumlah minimum 1 item. Jumlah maksimum 15 item.

Wajib: Ya

responseCard

Kartu respons. Amazon Lex menggunakan prompt ini saat runtime, dalam respons`PostText` API. Ini menggantikan atribut sesi dan nilai slot untuk placeholder di kartu respon. Untuk informasi selengkapnya, lihat Menggunakan Kartu Respons.

Tipe: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 50000.

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API di salah satuAWS SDK khusus bahasa, lihat yang berikut ini:

- AWSSDK for C++
- AWSSDK for Go
- AWSSDK for Java V2
- AWSSDK for Ruby V3

ResourceReference

Layanan:Amazon Lex Model Building Service

Menjelaskan sumber daya yang merujuk ke sumber daya yang Anda coba hapus. Objek ini dikembalikan sebagai bagian dari`ResourceInUseException` pengecualian.

Daftar Isi

name

Nama sumber daya yang menggunakan sumber daya yang Anda coba hapus.

Jenis: String

Panjang Batasan: Panjang minimum 1. Panjang maksimum 100.

Pola: `[a-zA-Z_]+`

Diperlukan: Tidak

version

Versi sumber daya yang menggunakan sumber daya yang Anda coba hapus.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum adalah 64.

Pola: `\$LATEST|[0-9]+`

Diperlukan: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API di salah satuAWS SDK khusus bahasa, lihat yang berikut ini:

- AWSSDK for C++
- AWSSDK for Go
- AWSSDK for Java V2
- AWSSDK for Ruby V3

Slot

Layanan: Amazon Lex Model Building Service

Mengidentifikasi versi slot tertentu.

Daftar Isi

name

Nama slot.

Jenis: String

Panjang Batasan: Panjang minimum 1. Panjang maksimum 100.

Pola: `^([A-Za-z](-|_|.)?)+$`

Diperlukan: Ya

slotConstraint

Menentukan apakah slot diperlukan atau opsional.

Jenis: String

Nilai yang Valid: `Required | Optional`

Wajib: Ya

defaultValueSpec

Daftar nilai default untuk slot. Nilai default digunakan saat Amazon Lex belum menentukan nilai untuk slot. Anda dapat menentukan nilai default dari variabel konteks, atribut sesi, dan nilai yang ditentukan.

Tipe: Objek SlotDefaultValueSpec

Wajib: Tidak

description

Sebuah deskripsi slot.

Jenis: String

Batasan Panjang: Panjang minimum 0. Panjang maksimum 200.

Wajib: Tidak

obfuscationSetting

Menentukan apakah slot dikaburkan dalam log percakapan dan ucapan yang disimpan. Saat Anda mengaburkan slot, nilainya diganti dengan nama slot dalam kurung kurawal ({}). Misalnya, jika nama slot adalah “full_name”, nilai yang dikaburkan diganti dengan “{full_name}”. Untuk informasi lebih lanjut, lihat Slot Obfuscation.

Jenis: String

Nilai yang Valid: `NONE | DEFAULT_OBFUSCATION`

Wajib: Tidak

priority

Mengarahkan Amazon Lex urutan di mana untuk mendapatkan nilai Slot ini dari pengguna. Misalnya, jika intent memiliki dua slot dengan prioritas 1 dan 2, AWS Amazon Lex pertama-tama mendapatkan nilai untuk slot dengan prioritas 1.

Jika beberapa slot memiliki prioritas yang sama, urutan nilai Amazon Lex menghasilkan sewenang-wenang.

Tipe: Bilangan Bulat

Rentang Valid: Nilai minimum 0. Nilai maksimum 100.

Diperlukan: Tidak

responseCard

Satu set tanggapan yang mungkin untuk jenis slot yang digunakan oleh klien berbasis teks. Pengguna memilih opsi dari kartu respons, alih-alih menggunakan teks untuk membalas.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 50000.

Wajib: Tidak

sampleUtterances

Jika Anda mengetahui pola tertentu yang dapat digunakan pengguna untuk menanggapi permintaan Amazon Lex untuk nilai slot, Anda dapat memberikan ucapan tersebut untuk

meningkatkan akurasi. Ini bersifat opsional. Dalam kebanyakan kasus, Amazon Lex mampu memahami ucapan pengguna.

Tipe: Array string

Anggota Array: Jumlah minimum 0 item. Jumlah maksimum 10 item.

Batasan Panjang: Panjang minimum 1. Panjang maksimum 200.

Wajib: Tidak

slotType

Jenis slot, baik jenis slot kustom yang Anda didefinisikan atau salah satu jenis slot built-in.

Jenis: String

Panjang Batasan: Panjang minimum 1. Panjang maksimum 100.

Pola: `^((AMAZON\.)_?|[A-Za-z]_?)+`

Wajib: Tidak

slotTypeVersion

Versi dari jenis slot.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum adalah 64.

Pola: `\$LATEST|[0-9]+`

Wajib: Tidak

valueElicitationPrompt

Prompt yang digunakan Amazon Lex untuk mendapatkan nilai slot dari pengguna.

Tipe: Objek Prompt

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat yang berikut ini:

- AWSSDK untuk C++
- AWSSDK untuk Go
- AWSSDK untuk Java V2
- AWSSDK para Ruby V3

SlotDefaultValue

Layanan:Amazon Lex Model Building Service

Sebuah nilai default untuk slot.

Daftar Isi

defaultValue

Nilai default untuk slot. Anda dapat menentukan salah satu hal berikut:

- `#context-name.slot-name`- Nilai slot “slot-nama” dalam konteks “konteks-nama.”
- `{attribute}`- Nilai slot dari atribut sesi “atribut.”
- `'value'`- Nilai diskrit “nilai.”

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 202.

Wajib: Ya

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API di salah satuAWS SDK khusus bahasa, lihat yang berikut ini:

- AWSSDK for C++
- AWSSDK for Go
- AWSSDK for Java V2
- AWSSDK for Ruby V3

SlotDefaultValueSpec

Layanan:Amazon Lex Model Building Service

Berisi nilai default untuk slot. Nilai default digunakan saat Amazon Lex belum menentukan nilai untuk slot.

Daftar Isi

defaultValueList

Nilai default untuk slot. Anda dapat menentukan lebih dari satu default. Misalnya, Anda dapat menentukan nilai default untuk digunakan dari variabel konteks yang cocok, atribut sesi, atau nilai tetap.

Nilai default yang dipilih dipilih berdasarkan urutan yang Anda tentukan dalam daftar. Misalnya, jika Anda menentukan variabel konteks dan nilai tetap dalam urutan itu, Amazon Lex menggunakan variabel konteks jika tersedia, jika tidak menggunakan nilai tetap.

Tipe: Array objek SlotDefaultValue

Anggota Array: Jumlah minimum 0 item. Jumlah maksimum 10 item.

Wajib: Ya

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API di salah satuAWS SDK for bahasa, lihat yang berikut ini:

- AWSSDK for C++
- AWSSDK for Go
- AWSSDK for Java V2
- AWSSDK for Ruby V3

SlotTypeConfiguration

Layanan:Amazon Lex Model Building Service

Menyediakan informasi konfigurasi untuk jenis slot.

Daftar Isi

regexConfiguration

Ekspresi reguler yang digunakan untuk memvalidasi nilai slot.

Tipe: Objek SlotTypeRegexConfiguration

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API di salah satuAWS SDK khusus bahasa, lihat yang berikut ini:

- AWSSDK for C++
- AWSSDK for Go
- AWSSDK for Java V2
- AWSSDK for Ruby V3

SlotTypeMetadata

Layanan:Amazon Lex Model Building Service

Menyediakan informasi tentang jenis slot yang

Daftar Isi

createdDate

Tanggal jenis slot dibuat.

Tipe: Timestamp

Wajib: Tidak

description

Sebuah deskripsi dari jenis slot.

Jenis: String

Batasan Panjang: Panjang minimum 0. Panjang maksimum 200.

Wajib: Tidak

lastUpdatedDate

Tanggal bahwa jenis slot diperbarui. Saat Anda membuat sumber daya, tanggal pembuatan dan tanggal diperbarui terakhir untuk setiap.

Tipe: Timestamp

Wajib: Tidak

name

Nama jenis slot.

Jenis: String

Panjang Batasan: Panjang minimum 1. Panjang maksimum 100.

Pola: `^([A-Za-z]_?)+$`

Diperlukan: Tidak

version

Versi dari jenis slot.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum adalah 64.

Pola: `\$LATEST|[0-9]+`

Diperlukan: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API di salah satuAWS SDK khusus bahasa, lihat yang berikut ini:

- AWSSDK for C++
- AWSSDK for Go
- AWSSDK for Java V2
- AWSSDK for Ruby V3

SlotTypeRegexConfiguration

Layanan:Amazon Lex Model Building Service

Memberikan ekspresi reguler yang digunakan untuk memvalidasi nilai slot.

Daftar Isi

pattern

Ekspresi reguler yang digunakan untuk memvalidasi nilai slot.

Gunakan ekspresi reguler yang berikut ini. Amazon Lex mendukung karakter berikut dalam ekspresi reguler:

- A-Z, a-z
- 0-9
- Karakter Unicode (“\ u <Unicode>“)

Mewakili karakter Unicode dengan empat digit, misalnya “\ u0041" atau “\ u005a”.

Operator ekspresi reguler berikut tidak didukung:

- Repeater tak terbatas: *, +, atau {x,} tanpa batas atas.
- Kartu liar (.)

Jenis: String

Panjang Batasan: Panjang minimum 1. Panjang maksimum 100.

Wajib: Ya

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API di salah satuAWS SDK khusus bahasa, lihat yang berikut ini:

- AWSSDK for C++
- AWSSDK for Go
- AWSSDK for Java V2
- AWSSDK for Ruby V3

Statement

Layanan:Amazon Lex Model Building Service

Kumpulan pesan yang menyampaikan informasi kepada pengguna. Saat runtime, Amazon Lex memilih pesan yang akan disampaikan.

Daftar Isi

messages

Kumpulan objek pesan.

Tipe: Array objek Message

Anggota Array: Jumlah minimum 1 item. Jumlah maksimum 15 item.

Wajib: Ya

responseCard

Saat runtime, jika klien menggunakan PostTextAPI, Amazon Lex menyertakan kartu respons dalam respons. Ini menggantikan semua atribut sesi dan nilai slot untuk placeholder di kartu respon.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 50000.

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API di salah satuAWS SDK khusus bahasa, lihat yang berikut ini:

- AWSSDK for C++
- AWSSDK for Go
- AWSSDK for Java V2
- AWSSDK for Ruby V3

Tag

Layanan:Amazon Lex Model Building Service

Daftar pasangan kunci/nilai yang mengidentifikasi bot, atau saluran bot. Kunci dan nilai tanda dapat terdiri dari huruf Unicode, angka, white space, dan salah satu simbol berikut: _ . : / = + - @.

Daftar Isi

key

Kunci untuk tanda. Kunci tidak peka dengan huruf besar/kecil.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 128.

Wajib: Ya

value

Nilai yang terkait dengan kunci. Nilai dapat berupa string kosong, tetapi tidak boleh null.

Jenis: String

Batasan Panjang: Panjang minimum 0. Panjang maksimum 256.

Wajib: Ya

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API di salah satuAWS SDK khusus bahasa, lihat yang berikut ini:

- AWSSDK for C++
- AWSSDK for Go
- AWSSDK for Java
- AWSSDK for Ruby

UtteranceData

Layanan:Amazon Lex Model Building Service

Memberikan informasi tentang satu ucapan yang dibuat untuk bot Anda.

Daftar Isi

count

Berapa kali ucapan tersebut diproses.

Tipe: Integer

Wajib: Tidak

distinctUsers

Jumlah total individu yang menggunakan ucapan tersebut.

Tipe: Integer

Wajib: Tidak

firstUtteredDate

Tanggal bahwa ucapan itu pertama kali dicatat.

Tipe: Timestamp

Wajib: Tidak

lastUtteredDate

Tanggal bahwa ucapan itu terakhir dicatat.

Tipe: Timestamp

Wajib: Tidak

utteranceString

Teks yang dimasukkan oleh pengguna atau representasi teks klip audio.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2000.

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API di salah satuAWS SDK khusus bahasa, lihat yang berikut ini:

- AWSSDK for C++
- AWSSDK for Go
- AWSSDK for Java V2
- AWSSDK for Ruby V3

UtteranceList

Layanan:Amazon Lex Model Building Service

Menyediakan daftar ucapan yang telah dibuat ke versi spesifik bot Anda. Daftar ini berisi maksimal 100 ucapan.

Daftar Isi

botVersion

Versi bot yang memproses daftar.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum adalah 64.

Pola: `\$LATEST|[0-9]+`

Diperlukan: Tidak

utterances

Satu atau lebihUtteranceData objek yang berisi informasi tentang ucapan yang telah dibuat ke bot. Jumlah maksimum objek adalah 100.

Tipe: Array objek UtteranceData

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API di salah satuAWS SDK khusus bahasa, lihat yang berikut ini:

- AWSSDK for C++
- AWSSDK for Go
- AWSSDK for Java V2
- AWSSDK for Ruby V3

Amazon Lex Runtime

tipe data berikut didukung oleh Amazon Lex Runtime Service:

- ActiveContext
- ActiveContextTimeToLive
- Button
- DialogAction
- GenericAttachment
- IntentConfidence
- IntentSummary
- PredictedIntent
- ResponseCard
- SentimentResponse

ActiveContext

Layanan:Amazon Lex Runtime Service

Konteks adalah variabel yang berisi informasi tentang keadaan percakapan saat ini antara pengguna dan Amazon Lex. Konteks dapat diatur secara otomatis oleh Amazon Lex ketika maksud terpenuhi, atau dapat diatur pada waktu proses menggunakan`PutContent`,`PutText`, atau`PutSession` operasi.

Daftar Isi

name

Nama konteksnya.

Jenis: String

Panjang Batasan: Panjang minimum 1. Panjang maksimum 100.

Pola: `^([A-Za-z]_?)+$`

Diperlukan: Ya

parameters

variabel negara untuk konteks saat ini. Anda dapat menggunakan nilai-nilai ini sebagai nilai default untuk slot di acara berikutnya.

Tipe: Peta string ke string

Entri Peta: Jumlah minimum 0 item. Jumlah maksimum 10 item.

Batasan Panjang Kunci: Panjang minimum 1. Panjang maksimum 100.

Batasan Panjang Nilai: Panjang minimum 1. Panjang maksimum 1024.

Wajib: Ya

timeToLive

Lamanya waktu atau jumlah belokan yang konteks tetap aktif.

Tipe: Objek ActiveContextTimeToLive

Wajib: Ya

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API di salah satuAWS SDK khusus bahasa, lihat yang berikut ini:

- AWSSDK for C++
- AWSSDK for Go
- AWSSDK for Java V2
- AWSSDK for Ruby V3

ActiveContextTimeToLive

Layanan:Amazon Lex Runtime Service

Lamanya waktu atau jumlah belokan yang konteks tetap aktif.

Daftar Isi

timeToLiveInSeconds

Jumlah detik bahwa konteks harus aktif setelah pertama kali dikirim dalam`PostContent` atau`PostText` respon. Anda dapat mengatur nilai antara 5 dan 86.400 detik (24 jam).

Tipe: Bilangan Bulat

Rentang yang Valid: Nilai minimum 5. Nilai maksimum 86400.

Wajib: Tidak

turnsToLive

Jumlah percakapan ternyata konteksnya harus aktif. Giliran percakapan adalah satu`PostContent` atau`PostText` permintaan dan respons yang sesuai dari Amazon Lex.

Tipe: Bilangan Bulat

Rentang yang Valid: Nilai minimum 1. Nilai maksimum 20.

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API di salah satuAWS SDK khusus bahasa, lihat yang berikut ini:

- AWSSDK for C++
- AWSSDK for Go
- AWSSDK for Java V2
- AWSSDK for Ruby V3

Button

Layanan:Amazon Lex Runtime Service

Merupakan opsi yang akan ditampilkan pada platform klien (Facebook, Slack, dll.)

Daftar Isi

text

Teks yang terlihat oleh pengguna pada tombol.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 15.

Wajib: Ya

value

Nilai yang dikirim ke Amazon Lex saat pengguna memilih tombol. Misalnya, pertimbangkan tombol teks “NYC.” Ketika pengguna memilih tombol, nilai yang dikirim dapat “New York City.”

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 1000.

Wajib: Ya

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API di salah satuAWS SDK for bahasa, lihat yang berikut ini:

- AWSSDK for C++
- AWSSDK for Go
- AWSSDK for Java V2
- AWSSDK for Ruby V3

DialogAction

Layanan: Amazon Lex Runtime Service

Menjelaskan tindakan selanjutnya yang harus diambil bot dalam interaksinya dengan pengguna dan memberikan informasi tentang konteks di mana tindakan tersebut terjadi. Gunakan tipe `DialogAction` data untuk mengatur interaksi ke keadaan tertentu, atau untuk mengembalikan interaksi ke keadaan sebelumnya.

Daftar Isi

type

Tindakan selanjutnya yang harus diambil bot dalam interaksinya dengan pengguna. Nilai yang mungkin adalah:

- `ConfirmIntent`- Tindakan selanjutnya adalah meminta pengguna apakah maksud selesai dan siap untuk dipenuhi. Ini adalah pertanyaan ya/tidak seperti “Tempatkan pesanan?”
- `Close`- Menunjukkan bahwa tidak akan ada respons dari pengguna. Misalnya, pernyataan “Pesanan Anda telah ditempatkan” tidak memerlukan tanggapan.
- `Delegate`- Tindakan selanjutnya ditentukan oleh Amazon Lex.
- `ElicitIntent`- Tindakan selanjutnya adalah menentukan maksud yang ingin dipenuhi pengguna.
- `ElicitSlot`- Tindakan selanjutnya adalah untuk mendapatkan nilai slot dari pengguna.

Jenis: String

Nilai yang Valid: `ElicitIntent | ConfirmIntent | ElicitSlot | Close | Delegate`

Wajib: Ya

fulfillmentState

Keadaan pemenuhan maksud. Nilai yang mungkin adalah:

- `Failed`- Fungsi Lambda yang terkait dengan maksud gagal memenuhi maksud tersebut.
- `Fulfilled`- Maksud telah dipenuhi oleh fungsi Lambda yang terkait dengan maksud.
- `ReadyForFulfillment`- Semua informasi yang diperlukan untuk maksud hadir dan maksud siap untuk dipenuhi oleh aplikasi klien.

Jenis: String

Nilai yang Valid: `Fulfilled | Failed | ReadyForFulfillment`

Wajib: Tidak

intentName

Nama maksud.

Tipe: String

Wajib: Tidak

message

Pesan yang harus ditampilkan kepada pengguna. Jika Anda tidak menentukan pesan, Amazon Lex akan menggunakan pesan yang dikonfigurasi untuk maksud tersebut.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 1024.

Wajib: Tidak

messageFormat

- `PlainText`- Pesan berisi teks UTF-8 biasa.
- `CustomPayload`- Pesan adalah format khusus untuk klien.
- `SSML`- Pesan berisi teks yang diformat untuk output suara.
- `Composite`- Pesan berisi objek JSON yang lolos yang berisi satu atau lebih pesan. Untuk selengkapnya, lihat Grup Pesan.

Jenis: String

Nilai yang Valid: `PlainText | CustomPayload | SSML | Composite`

Wajib: Tidak

slots

Peta slot yang telah dikumpulkan dan nilai-nilai mereka.

Tipe: Peta antar string

Wajib: Tidak

slotToElicit

Nama slot yang harus ditimbulkan dari pengguna.

Tipe: String

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat yang berikut ini:

- AWSSDK untuk C++
- AWSSDK untuk Go
- AWSSDK untuk Java V2
- AWSSDK para Ruby V3

GenericAttachment

Layanan:Amazon Lex Runtime Service

Merupakan opsi yang diberikan kepada pengguna ketika prompt ditampilkan. Bisa berupa gambar, tombol, tautan, atau teks.

Daftar Isi

attachmentLinkUrl

URL lampiran ke kartu respon.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Wajib: Tidak

buttons

Daftar opsi untuk ditampilkan kepada pengguna.

Tipe: Array objek Button

Anggota Array: Jumlah minimum 0 item. Jumlah maksimum 5 item.

Wajib: Tidak

imageUrl

URL citra yang ditampilkan kepada pengguna.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 2048.

Wajib: Tidak

subTitle

Subtitle yang ditunjukkan di bawah judul.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 80.

Wajib: Tidak

title

Judul opsi.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 80.

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API di salah satuAWS SDK khusus bahasa, lihat yang berikut ini:

- AWSSDK for C++
- AWSSDK for Go
- AWSSDK for Java V2
- AWSSDK for Ruby V3

IntentConfidence

Layanan:Amazon Lex Runtime Service

Memberikan skor yang menunjukkan keyakinan bahwa Amazon Lex memiliki bahwa maksud adalah salah satu yang memenuhi maksud pengguna.

Daftar Isi

score

Skor yang menunjukkan seberapa percaya diri Amazon Lex bahwa maksud memenuhi maksud pengguna. Rentang antara 0,00 dan 1,00. Skor yang lebih tinggi menunjukkan kepercayaan diri yang lebih tinggi.

Jenis: Ganda

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API di salah satuAWS SDK khusus bahasa, lihat yang berikut ini:

- AWSSDK for C++
- AWSSDK for Go
- AWSSDK for Java V2
- AWSSDK for Ruby V3

IntentSummary

Layanan: Amazon Lex Runtime Service

Menyediakan informasi tentang keadaan maksud. Anda dapat menggunakan informasi ini untuk mendapatkan status maksud saat ini sehingga Anda dapat memproses intent, atau agar Anda dapat mengembalikan intent ke status sebelumnya.

Daftar Isi

dialogActionType

Tindakan selanjutnya yang harus diambil bot dalam interaksinya dengan pengguna. Nilai yang mungkin adalah:

- `ConfirmIntent`- Tindakan selanjutnya adalah meminta pengguna apakah maksud selesai dan siap untuk dipenuhi. Ini adalah pertanyaan ya/tidak seperti “Tempatkan pesanan?”
- `Close`- Menunjukkan bahwa tidak akan ada respons dari pengguna. Misalnya, pernyataan “Pesanan Anda telah ditempatkan” tidak memerlukan tanggapan.
- `ElicitIntent`- Tindakan selanjutnya adalah menentukan maksud yang ingin dipenuhi pengguna.
- `ElicitSlot`- Tindakan selanjutnya adalah untuk mendapatkan nilai slot dari pengguna.

Jenis: String

Nilai yang Valid: `ElicitIntent | ConfirmIntent | ElicitSlot | Close | Delegate`

Wajib: Ya

checkpointLabel

Label yang ditentukan pengguna yang mengidentifikasi maksud tertentu. Anda dapat menggunakan label ini untuk kembali ke maksud sebelumnya.

Gunakan `checkpointLabelFilter` parameter `GetSessionRequest` operasi untuk menyaring maksud yang dikembalikan oleh operasi kepada mereka yang hanya memiliki label yang ditentukan.

Jenis: String

Batasan Panjang: Panjang minimum 1. Panjang maksimum 255.

Pola: `[a-zA-Z0-9-]+`

Wajib: Tidak

confirmationStatus

Status maksud setelah pengguna merespons prompt konfirmasi. Jika pengguna mengonfirmasi maksud, Amazon Lex menyetel bidang ini ke. `Confirmed` Jika pengguna menolak maksud, Amazon Lex menetapkan nilai ini ke. `Denied` Nilai yang mungkin adalah:

- `Confirmed`- Pengguna telah menanggapi “Ya” pada prompt konfirmasi, mengonfirmasi bahwa maksud sudah lengkap dan siap untuk dipenuhi.
- `Denied`- Pengguna telah menanggapi “Tidak” pada prompt konfirmasi.
- `None`- Pengguna tidak pernah diminta untuk konfirmasi; atau, pengguna diminta tetapi tidak mengkonfirmasi atau menolak prompt.

Jenis: String

Nilai yang Valid: `None | Confirmed | Denied`

Wajib: Tidak

fulfillmentState

Keadaan pemenuhan maksud. Nilai yang mungkin adalah:

- `Failed`- Fungsi Lambda yang terkait dengan maksud gagal memenuhi maksud.
- `Fulfilled`- Maksud telah dipenuhi oleh fungsi Lambda yang terkait dengan maksud.
- `ReadyForFulfillment`- Semua informasi yang diperlukan untuk maksud hadir dan maksud siap untuk dipenuhi oleh aplikasi klien.

Jenis: String

Nilai yang Valid: `Fulfilled | Failed | ReadyForFulfillment`

Wajib: Tidak

intentName

Nama maksud.

Tipe: String

Wajib: Tidak

slots

Peta slot yang telah dikumpulkan dan nilai-nilai mereka.

Tipe: Peta antar string

Wajib: Tidak

slotToElicit

Slot berikutnya untuk mendapatkan dari pengguna. Jika tidak ada slot untuk mendapatkan, bidang kosong.

Tipe: String

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API ini di salah satu AWS SDK khusus bahasa, lihat yang berikut ini:

- AWSSDK untuk C++
- AWSSDK untuk Go
- AWSSDK untuk Java V2
- AWSSDK para Ruby V3

PredictedIntent

Layanan:Amazon Lex Runtime Service

Maksud yang disarankan Amazon Lex memenuhi maksud pengguna. Termasuk nama maksud, keyakinan bahwa Amazon Lex memiliki bahwa niat pengguna puas, dan slot didefinisikan untuk maksud.

Daftar Isi

intentName

Nama maksud yang disarankan Amazon Lex memenuhi maksud pengguna.

Tipe: String

Wajib: Tidak

nluIntentConfidence

Menunjukkan seberapa percaya diri Amazon Lex bahwa maksud memenuhi maksud pengguna.

Tipe: Objek IntentConfidence

Wajib: Tidak

slots

Slot dan slot nilai-nilai yang terkait dengan maksud diprediksi.

Tipe: Peta antar string

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API di salah satuAWS SDK khusus bahasa, lihat yang berikut ini:

- AWSSDK for C++
- AWSSDK for Go
- AWSSDK for Java V2
- AWSSDK for Ruby V3

ResponseCard

Layanan: Amazon Lex Runtime Service

Jika Anda mengonfigurasi kartu respons saat membuat bot, Amazon Lex mengganti atribut sesi dan nilai slot yang tersedia, lalu mengembalikannya. Kartu respons juga dapat berasal dari fungsi Lambda (`dialogCodeHook`dan `fulfillmentActivity` pada maksud).

Daftar Isi

contentType

Jenis konten respon.

Jenis: String

Nilai yang Valid: `application/vnd.amazonaws.card.generic`

Wajib: Tidak

genericAttachments

Array objek lampiran yang mewakili pilihan.

Tipe: Array objek GenericAttachment

Anggota Array: Jumlah minimum 0 item. Jumlah maksimum 10 item.

Wajib: Tidak

version

Versi format kartu respon.

Tipe: String

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API di salah satu AWS SDK khusus bahasa, lihat yang berikut ini:

- AWSSDK for C++

- AWSSDK for Go
- AWSSDK for Java V2
- AWSSDK for Ruby V3

SentimentResponse

Layanan:Amazon Lex Runtime Service

Sentimen diungkapkan dalam sebuah ucapan.

Ketika bot dikonfigurasi untuk mengirim ucapan ke Amazon Comprehend untuk analisis sentimen, struktur bidang ini berisi hasil analisis.

Daftar Isi

sentimentLabel

Sentimen yang disimpulkan bahwa Amazon Comprehend memiliki kepercayaan tertinggi.

Tipe: String

Wajib: Tidak

sentimentScore

Kemungkinan sentimen itu disimpulkan dengan benar.

Tipe: String

Wajib: Tidak

Lihat Juga

Untuk informasi selengkapnya tentang penggunaan API di salah satuAWS SDK khusus bahasa, lihat yang berikut ini:

- AWSSDK for C++
- AWSSDK for Go
- AWSSDK for Java V2
- AWSSDK for Ruby V3

Riwayat Dokumen untuk Amazon Lex

- Pembaruan dokumentasi terakhir: 9 September 2021

Tabel berikut menjelaskan perubahan penting dalam setiap rilis Amazon Lex. Untuk notifikasi tentang pembaruan dokumentasi ini, Anda dapat berlangganan ke umpan RSS.

Perubahan	Deskripsi	Tanggal
Fitur baru	Amazon Lex sekarang mendukung lokal Korea (KO-KR). Untuk informasi selengkapnya, lihat Bahasa-bahasa yang didukung oleh Amazon Lex.	9 September 2021
Fitur baru	Amazon Lex sekarang mendukung bahasa Inggris (India) lokal. Untuk informasi selengkapnya, lihat Bahasa-bahasa yang didukung di Amazon Lex.	15 Juli 2021
Fitur baru	Amazon Lex sekarang menyediakan alat untuk memigrasi bot ke API Amazon Lex V2. Untuk informasi selengkapnya, lihat Memigrasi bot.	13 Juli 2021
Fitur baru	Amazon Lex sekarang mendukung lokal Jepang (Jepang). Untuk informasi selengkapnya, lihat Bahasa-bahasa yang didukung oleh Amazon Lex.	1 April 2021

Fitur baru	Amazon Lex sekarang mendukung lokal Jerman (Jerman) (de-DE) dan Spanyol (Amerika Latin) (es-419). Untuk informasi selengkapnya, lihat Bahasa-bahasa yang didukung oleh Amazon Lex.	23 November 2020
Fitur baru	Amazon Lex sekarang mendukung penggunaan konteks untuk mengelola intent pengaktifan. Untuk informasi selengkapnya, lihat Mengatur Konteks Intent.	19 November 2020
Fitur baru	Amazon Lex sekarang mendukung Perancis (fr-FR), Perancis Kanada (fr-CA), Italia (IT-IT) dan Spanyol (es-ES) lokal. Untuk daftar lengkap terkait lokal yang didukung, lihat Bahasa yang didukung oleh Amazon Lex.	11 November 2020
Fitur baru	Amazon Lex sekarang mendukung lokal Spanyol (AS) (es-AS). Untuk informasi selengkapnya, lihat Bahasa-bahasa yang didukung oleh Amazon Lex.	22 September 2020
Fitur baru	Amazon Lex sekarang mendukung bahasa Inggris (Inggris) (en-GB) lokal. Untuk informasi selengkapnya, lihat Bahasa-bahasa yang didukung oleh Amazon Lex.	15 September 2020

Fitur baru	Amazon Lex sekarang mendukung bahasa Inggris (Australia) (en-AU) lokal. Untuk informasi selengkap nya, lihat Bahasa-bahasa yang didukung oleh Amazon Lex.	8 September 2020
Fitur baru	Amazon Lex sekarang memiliki 7 intent bawaan baru dan 9 jenis slot built-in baru. Untuk informasi lebih lanjut, lihat Intent Built-in dan Jenis Slot.	8 September 2020
Contoh baru	Pelajari cara membuat bot Amazon Lex yang dapat digunakan agen dukungan pelanggan untuk menjawab pertanyaan pelanggan dengan mencari jawaban dengan Amazon Kendra. Untuk informasi selengkapnya, lihat Contoh: Call Center Agent Assistant.	10 Agustus 2020
Fitur baru	Amazon Lex sekarang dapat mengembalikan hingga empat maksud alternatif berdasark an skor kepercayaan. Untuk informasi selengkapnya, lihat Menggunakan Skor Keyakinan .	6 Agustus 2020
Ekspansi wilayah	Amazon Lex sekarang tersedia di Asia Pacific (Tokyo) (ap-northeast-1).	30 Juni 2020

Fitur baru	Amazon Lex sekarang mendukung pencarian indeks Amazon Kendra untuk jawaban atas pertanyaan yang sering diajukan. Untuk informasi selengkapnya, lihat AMAZON. KendraSearchIntent .	11 Juni 2020
Fitur baru	Amazon Lex sekarang mengembalikan lebih banyak informasi dalam log percakapan. Untuk informasi selengkapnya, lihat Melihat Log Teks di CloudWatch Log Amazon.	9 Juni 2020
Ekspansi wilayah	Amazon Lex kini tersedia di Asia Pacific (Singapura) (ap-southeast-1), Europe (Frankfurt) (eu-central-1), dan Europe (London) (eu-west-2).	23 April 2020
Fitur baru	Amazon Lex sekarang mendukung penandaan . Anda dapat menggunakan penandaan untuk mengidentifikasi sumber daya, mengalokasikan biaya, dan mengontrol akses. Untuk informasi selengkapnya, lihat Menandai Amazon Lex Resources Anda.	12 Maret 2020

Fitur baru	Amazon Lex sekarang mendukung ekspresi reguler untuk AMAZON. AlphaNumeric built-in jenis slot. Untuk informasi selengkapnya, lihat AMAZON. AlphaNumeric.	6 Februari 2020
Fitur baru	Amazon Lex sekarang dapat mencatat informasi percakapan dan mengaburkan nilai slot di log tersebut. Untuk informasi selengkapnya, lihat Membuat Log Percakapan dan Slot Obfuscation.	19 Desember 2019
Ekspansi wilayah	Amazon Lex sekarang tersedia di Asia Pacific (Sydney) (ap-southeast-2).	17 Desember 2019
Fitur baru	Amazon Lex sekarang sesuai dengan HIPAA. Untuk informasi selengkapnya, lihat Validasi Kepatuhan untuk Amazon Lex.	10 Desember 2019
Fitur baru	Amazon Lex sekarang dapat mengirim ucapan pengguna ke Amazon Comprehend untuk menganalisis sentimen ucapan tersebut. Untuk informasi lebih lanjut, lihat Analisis Sentimen.	21 November 2019
Fitur baru	Amazon Lex kini mematuhi SOC. Untuk informasi selengkapnya, lihat Validasi Kepatuhan untuk Amazon Lex.	19 November 2019

Fitur baru	Amazon Lex sekarang sesuai dengan PCI. Untuk informasi selengkapnya, lihat Validasi Kepatuhan untuk Amazon Lex.	17 Oktober 2019
Fitur baru	Menambahkan dukungan untuk menambahkan pos pemeriksaan ke intent sehingga Anda dapat dengan mudah kembali ke maksud selama percakapan. Untuk informasi selengkapnya, lihat Mengelola Sesi.	10 Oktober 2019
Fitur baru	Ditambahkan dukungan untuk`AMAZON.FallbackIntent` sehingga bot Anda dapat menangani situasi ketika input pengguna tidak seperti yang diharapkan. Untuk informasi selengkapnya, lihat AMAZON. FallbackIntent.	3 Oktober 2019
Fitur baru	Amazon Lex memungkinkan Anda mengelola informasi sesi untuk bot Anda. Untuk informasi selengkapnya, lihat Mengelola Sesi Dengan Amazon Lex API.	8 Agustus 2019
Ekspansi wilayah	Amazon Lex sekarang tersedia di US West (Oregon) (us-west-2).	8 Mei 2018

Fitur baru	Menambahkan dukungan untuk mengekspor dan mengimpor dalam format Amazon Lex. Untuk informasi selengkapnya, lihat Mengimpor dan Mengekspor Amazon Lex Bots, Intent, dan Jenis Slot.	13 Februari 2018
Fitur baru	Amazon Lex sekarang mendukung pesan respons tambahan untuk bot. Untuk informasi selengkapnya, lihat Tanggapan.	8 Februari 2018
Ekspansi wilayah	Amazon Lex sekarang tersedia di Eropa (Irlandia).	21 November 2017
Fitur baru	Menambahkan dukungan untuk menerapkan bot Amazon Lex di Kik. Untuk informasi selengkapnya, lihat Mengintegrasikan Amazon Lex Bot dengan Kik.	20 November 2017
Fitur baru	Menambahkan dukungan untuk jenis slot built-in baru dan atribut permintaan. Untuk informasi selengkapnya, lihat Jenis Slot Bawaan dan Atribut Permintaan Pengaturan.	3 November 2017
Fitur baru	Ditambahkan ekspor ke fitur Alexa Skills Kit. Untuk informasi lebih lanjut, lihat Mengekspor ke Keterampilan Alexa.	7 September 2017

Fitur baru	Ditambahkan dukungan sinonim untuk nilai jenis slot. Untuk informasi lebih lanjut, lihat Jenis Slot Kustom.	31 Agustus 2017
Fitur baru	DitambahkanAWS CloudTrail integrasi. Untuk informasi selengkapnya, lihat Memantau Panggilan API Amazon Lex denganAWS CloudTrail Log.	15 Agustus 2017
Dokumentasi yang diperluas	Ditambahkan Memulai contoh untukAWS CLI. Untuk informasi selengkapnya, lihat https://docs.aws.amazon.com/lex/latest/dg/gs-cli.html	22 Mei 2017
Panduan baru	Ini adalah rilis pertama dari Panduan Pengguna Amazon Lex.	19 April 2017

AWSGlosarium

Untuk AWS terminologi terbaru, lihat AWSglosarium di Referensi. Glosarium AWS