aws

開發人員指南

Amazon Lex V1

Amazon Lex V1: 開發人員指南

Table of Contents

如果您使用的是 Amazon Lex V2，請參閱 Amazon Lex V2 指南。

如果您使用的是 Amazon Lex V1，我們建議您將機器人升級到 Amazon Lex V2。我們不再向 V1 添加新功能，強烈建議所有新機器人使用 V2。

本文為英文版的機器翻譯版本，如內容有任何歧義或不一致之處，概以英文版為準。

什麼是 Amazon Lex？

Amazon Lex 是一項 AWS 服務，可利用語音和文字為應用程式建置對話式介面。透過 Amazon Lex，如今任何開發人員都能利用支援 Amazon Alexa 的同一套對話引擎，進而能夠在新應用程式和現有應用程式中建置精密複雜的自然語言聊天機器人。Amazon Lex 具備自然語言理解 (NLU) 和自動語音辨識 (ASR) 的深度功能性與靈活性，令您得以透過生動的對話互動打造高參與度的使用者體驗和建立全新類別的產品。

Amazon Lex 令任何開發人員都能快速建置對話型聊天機器人。藉助 Amazon Lex，無需具備深度學習的專業知識，只要在 Amazon Lex 主控台指定基本對話流程即可建立機器人。Amazon Lex 將管理對話並動態調整對話過程的回應。利用主控台，您可以建置、測試和發佈您的文字或語音聊天機器人。而後，您可將對話式介面加入到行動裝置、Web 應用程式和聊天平台 (例如 Facebook Messenger) 上的機器人。

Amazon Lex 提供了與AWS Lambda，您可以輕鬆地整合 AWS 平台上的眾多其他服務，包括 Amazon Cognito、AWS Mobile Hub、Amazon CloudWatch 和 Amazon DynamoDB。透過與 Lambda 的整合，機器人將能存取預先建置的無伺服器企業連接器，以連結至 Salesforce、HubSpot 或 Marketo 等 SaaS 應用程式中的資料。

使用 Amazon Lex 的一些優點包括：

- 簡單— Amazon Lex 將引導您使用主控台在幾分鐘內建立您自己的聊天機器人。您只需提供幾個範例短句，Amazon Lex 便會建置一套完整的自然語言模型，透過此模型，機器人即可利用語音和文字進行互動來提問、回答和完成複雜的任務。

- 民主化深度學習技術— 由支援 Alexa 的相同技術所支援，Amazon Lex 提供了用以建立語音語言理解 (SLU) 系統的 ASR 和 NLU 技術。藉助 SLU，Amazon Lex 接受自然語言語音和文字輸入、理解輸入背後的意圖，並透過叫用相應的業務函數以滿足使用者意圖。

  語音辨識和自然語言理解是電腦科學領域最具挑戰性的幾個待解問題，需要用到大量資料和基礎設施來訓練複雜的深度學習演算法。Amazon Lex 由支援 Alexa 的相同技術所支援，讓所有開發人員都能運用深度學習技術。Amazon Lex 聊天機器人將傳入的語音轉換成文字並理解使用者意圖以產生智慧型回應，使您能夠專注為客戶建置與眾不同的加值化機器人，進而透過對話式介面創造出全新類別的產品。

- 無縫部署和擴展— 藉助 Amazon Lex，您可以直接從 Amazon Lex 主控台建置、測試和部署聊天機器人。Amazon Lex 令您能夠輕鬆地發佈語音或文字聊天機器人，以供予行動裝置、Web 應用程式和聊天服務 (例如 Facebook Messenger) 使用。Amazon Lex 將自動擴展，所以您不必擔心如何佈建硬體和管理基礎設施來強化您的機器人體驗。

- 內建整合 AWS 平台— Amazon Lex 與其他 AWS 服務如 Amazon Cognito、AWS Lambda、Amazon CloudWatch 和AWS Mobile Hub。您可以藉助 AWS 平台來實施安全性、監控、使用者身分驗證、商業邏輯、儲存及行動應用程式開發。

- 經濟實惠— 藉助 Amazon Lex，無需前期成本或繳交最低費用。您只需就發出的文字或語音請求付費。依請求按用量付費的定價和低成本使本服務成為符合經濟效益建置對話式介面的方式。藉助 Amazon Lex 免費方案，您可以輕鬆地試用 Amazon Lex，無需任何初期投資。

您第一次使用 Amazon Lex 嗎？

如果您是第一次使用 Amazon Lex，建議您依序讀以下章節：

1. 開始使用 Amazon Lex— 本節說明如何設定您的帳户和測試 Amazon Lex。
2. API 參考 — 本節提供額外的範例讓您可用以探索 Amazon Lex。

Amazon Lex 運作方式

Amazon Lex 可讓您使用與 Amazon Alexa 相同技術支援的語音或文字界面來建置應用程式。以下是您在使用 Amazon Lex 時執行的典型步驟：

1. 建立機器人並使用您想要支援的一或多個意圖進行設定。設定機器人讓它可以了解使用者的目標 (意圖)，與使用者進行對話以引出資訊，並滿足使用者的意圖。
2. 測試機器人。您可以使用 Amazon Lex 主控台提供的測試視窗用戶端。
3. 發佈版本並建立別名。
4. 部署機器人。您可以將機器人部署在如行動應用程式等平台或簡訊平台上，例如 Facebook Messenger。

在開始使用之前，請熟悉以下 Amazon Lex 核心概念和術語：

- 機器人 — 機器人執行自動化任務，例如訂購比薩餅，預訂酒店，訂購鮮花等。Amazon Lex 機器人由自動語音辨識 (ASR) 和自然語言理解 (NLU) 功能提供支援。每個機器人在您的帳戶內都必須具有唯一名稱。

  Amazon Lex 機器人可以理解以文字或語音提供的使用者輸入，並以自然語言進行交談。您可以建立 Lambda 函數，並將其新增為意圖組態中的程式碼掛接，以執行使用者資料驗證和履行任務。

- 意圖代表使用者想要執行的動作。您建立機器人來支援一或多個相關的意圖。例如，您可以建立一個訂購比薩和飲料的機器人。對於每個意圖，您提供以下必要的資訊：

  - 意圖名稱 — 意圖的描述性名稱。例如：**`OrderPizza`**。意圖名稱在您的帳戶中必須是唯一名稱。
  - 語音範例 — 使用者可能如何傳達意圖。例如，使用者可能會說「我能否訂購比薩」或「我想要訂購比薩」。
  - 如何實現意圖 — 在用戶提供必要的信息（例如，在當地的比薩店下訂單）後，您希望如何實現意圖。建議您建立 Lambda 函數來實現意圖。

您可以選擇性地設定意圖，讓 Amazon Lex 只要將資訊傳回用戶端應用程式，即可完成必要的履行作業。

除了訂購披薩等自訂意圖外，Amazon Lex 還提供內建意圖，可快速設定您的機器人。如需詳細資訊，請參閱內建意圖和槽類型。

- 槽-意圖可以需要零個或多個槽或參數。您將槽新增為意圖組態的一部分。在執行階段，Amazon Lex 會提示使用者輸入特定插槽值。使用者必須提供所有必要插槽的值，Amazon Lex 才能達成意圖。

  例如，`OrderPizza` 意圖需要如比薩大小、餅皮種類和數量等槽。您在意圖組態中新增這些槽。針對每個插槽，您都會提供插槽類型，並提示 Amazon Lex 傳送給用戶端，以從使用者取得資料。用戶可以使用包含其他單詞的插槽值進行回復，例如「請大披薩」或「讓我們堅持小」。Amazon Lex 仍然可以了解預期的插槽值。

- 插槽類型 — 每個插槽都有一個類型。您可以建立自訂槽類型或使用內建槽類型。每種插槽類型在您的帳戶內都必須具有唯一名稱。例如，您可以建立和使用以下 `OrderPizza` 意圖的槽類型：

  - 大小 – 使用列舉值 `Small`、`Medium` 以及 `Large`。
  - 餅皮 – 使用列舉值 `Thick` 和 `Thin`。

  Amazon Lex 也提供內建插槽類型。例如，`AMAZON.NUMBER` 是您可以用於訂購的比薩數量的內建槽類型。如需詳細資訊，請參閱內建意圖和槽類型。

如需可用 Amazon Lex 的清單，請參閱 Amazon Web Services 一般參考》中的 AWS 區域和端點。

下列主題提供額外的資訊。我們建議您依序檢閱，然後探索開始使用 Amazon Lex 練習。

主題

- Amazon Lex 支持的語言

- 程式設計模型
- 管理訊息
- 管理對話內容
- 使用可信度分數
- 對話日誌
- 使用 Amazon Lex API 管理工作階段
- 機器人部署選項
- 內建意圖和槽類型
- 自訂槽類型
- 槽混淆
- 情緒分析
- 標記您的 Amazon Lex 資源

Amazon Lex 支持的語言

Amazon Lex V1 支援各種語言和語言環境。下表列出支援的語言及支援這些語言的功能。

Amazon Lex V2 支援其他語言，請參閱 Amazon Lex V2 支援的語言

支援的語言和語言環境

Amazon Lex V1 支援下列語言和地區設定。

代碼	語言與區域設定
de-DE	德語（德語）
en-AU	英文 (澳洲)
en-GB	英文 (英國)
en-IN	英文 (印度)
zh-TW	英文 (美國)
ES-419	西班牙文 (拉丁美洲)

代碼	語言與區域設定
es-ES	西班牙文 (西班牙)
es-US	西班牙文 (美國)
fr-CA	法文 (加拿大)
fr-FR	法文 (法國)
it-IT	義大利文 (義大利)
ja-JP	日文 (日本)
ko-KR	韓文 (韓國)

Amazon Lex 功能支援的語言和語言環境

除了本表所列之外，所有語言和地區設定都支援所有 Amazon Lex 功能。

功能	支援的語言和地區設定
設定意圖上下文	英文 (美國) (en-US)

程式設計模型

機器人是 Amazon Lex 中的主要資源類型。Amazon Lex 中的其他資源類型為意圖、位置類型、別名和機器人通道關聯。

您可以使用 Amazon Lex 主控台或模型建置 API 來建立機器人。主控台提供圖形化使用者界面，您可用來為應用程式建立生產就緒的機器人。如果您偏好，也可以透過 AWS CLI 使用模型建置 API 或您自己的自訂程式來建立機器人。

建立機器人之後，您將其部署在其中一個支援的平台，或將它整合到您自己的應用程式。當使用者與機器人互動時，用戶端應用程式會使用 Amazon Lex 執行階段 API 將請求傳送至機器人。例如，當使用者說「我想訂購披薩」時，您的用戶端會使用其中一個執行階段 API 操作，將此輸入傳送至 Amazon Lex。使用者可以語音或文字的形式提供輸入。

您也可以建立 Lambda 函數，並在意圖中使用它們。使用這些 Lambda 函數程式碼掛接來執行執行階段活動，例如初始化、驗證使用者輸入和意圖履行。下列各節提供了額外的資訊。

主題

- 模型建置 API 操作
- 執行時間 API 操作
- Lambda 函數作為代碼掛鉤

模型建置 API 操作

要透過程式設計方式建立機器人、意圖和槽類型，請使用模型建置 API 操作。您也可以使用模型建置 API 來管理、更新和刪除機器人的資源。模型建置 API 操作包括：

- PutBot、PutBotAlias、PutIntent 和 PutSlotType 分別會建立和更新機器人、機器人別名、意圖和槽類型。
- CreateBotVersion、CreateIntentVersion 和 CreateSlotTypeVersion 分別會建立和發佈機器人版本、意圖和槽類型。
- GetBot 和 GetBots 分別會取得您已建立的特定機器人或機器人清單。
- GetIntent 和 GetIntents 分別會取得您已建立的特定意圖或意圖清單。
- GetSlotType 和 GetSlotTypes 分別會取得您已建立的特定槽類型或槽類型清單。
- GetBuiltinIntentGetBuiltinIntents、以及取GetBuiltinSlotTypes得 Amazon Lex 內建意圖、Amazon Lex 內建意圖清單，或是您可以在機器人中使用的內建插槽類型清單。
- GetBotChannelAssociation 和 GetBotChannelAssociations 分別會取得機器人與簡訊平台之間的關聯，或機器人與簡訊平台之間的關聯清單。
- DeleteBot、DeleteBotAlias、DeleteBotChannelAssociation、DeleteIntent 和 DeleteSlotType 會移除您帳戶中不需要的資源。

您可以使用模型建置 API 建立自訂工具來管理 Amazon Lex 資源。舉例來說，槽、意圖和槽類型各有 100 個版本的限制。您可以使用模型建置 API 來建置工具，在機器人接近限制時自動刪除舊版本。

為確保一次只有一個作業更新資源，Amazon Lex 使用總和檢查碼。當您使用Put API 作業—PutBotPutBotAliasPutIntent、或PutSlotType — 來更新資源時，您必須在要求中傳遞資源的目前總和檢查碼。如果同時有兩個工具嘗試更新資源，兩個都會提供相同的目前檢查總和。第一個到達 Amazon Lex 的請求與資源目前的總和檢查碼相符。等到第二個請求抵達時，檢查總和已不同。第二個工具會收到 `PreconditionFailedException` 例外狀況，並且更新會終止。

Get操作 —GetBotGetIntent、和GetSlotType — 最終是一致的。如果您在使用其中一個 Get 操作建立或修改資源之後立即使用 Put 操作，可能不會傳回變更。在 Get 操作傳回最新的更新之後，在資源再度經過修改之前，它一律會傳回更新過的資源。您可以查看檢查總和來判斷傳回的是否為更新的資源。

執行時間 API 操作

用戶端應用程式會使用下列執行階段 API 操作與 Amazon Lex 進行通訊：

- PostContent— 進行語音或文字輸入，並傳回意圖資訊以及要傳達給使用者的文字或語音訊息。Amazon Lex 目前支援以下音訊格式：

  輸入音訊格式 – LPCM 和 Opus

  輸出音訊格式 – MPEG、OGG 和 PCM

  PostContent 操作支援 8 kHz 和 16 kHz 的音訊輸入。最終使用者透過電話與 Amazon Lex 交談的應用程式 (例如自動呼叫中心) 可以直接傳遞 8 kHz 音訊。

- PostText – 接受文字輸入並傳回意圖資訊和文字訊息，以傳達給使用者。

您的用戶端應用程式使用執行階段 API 呼叫特定 Amazon Lex 機器人來處理語音 (使用者文字或語音輸入)。例如，假設使用者說「我要訂購比薩」。用戶端會使用其中一個 Amazon Lex 執行階段 API 操作，將此使用者輸入傳送至機器人。Amazon Lex 會從使用者輸入中識別出使用者請求是針對機器人中定義的OrderPizza意圖。Amazon Lex 讓使用者參與對話，以收集必要的資訊或插槽資料，例如披薩大小、配料和比薩餅數量。使用者提供所有必要的插槽資料之後，Amazon Lex 會叫用 Lambda 函數程式碼勾點來達成意圖，或將意圖資料傳回給用戶端，視意圖的設定方式而定。

當機器人使用語音輸入時，使用 PostContent 操作。例如，自動呼叫中心應用程式可以將語音傳送至 Amazon Lex 機器人，而不是傳送代理程式來處理客戶查詢。您可以使用 8 kHz 音頻格式將音頻直接從電話發送到 Amazon Lex 斯。

Amazon Lex 主控台中的測試視窗會使用PostContent API 將文字和語音請求傳送到 Amazon Lex。您可以在開始使用 Amazon Lex 練習中使用此測試視窗。

Lambda 函數作為代碼掛鉤

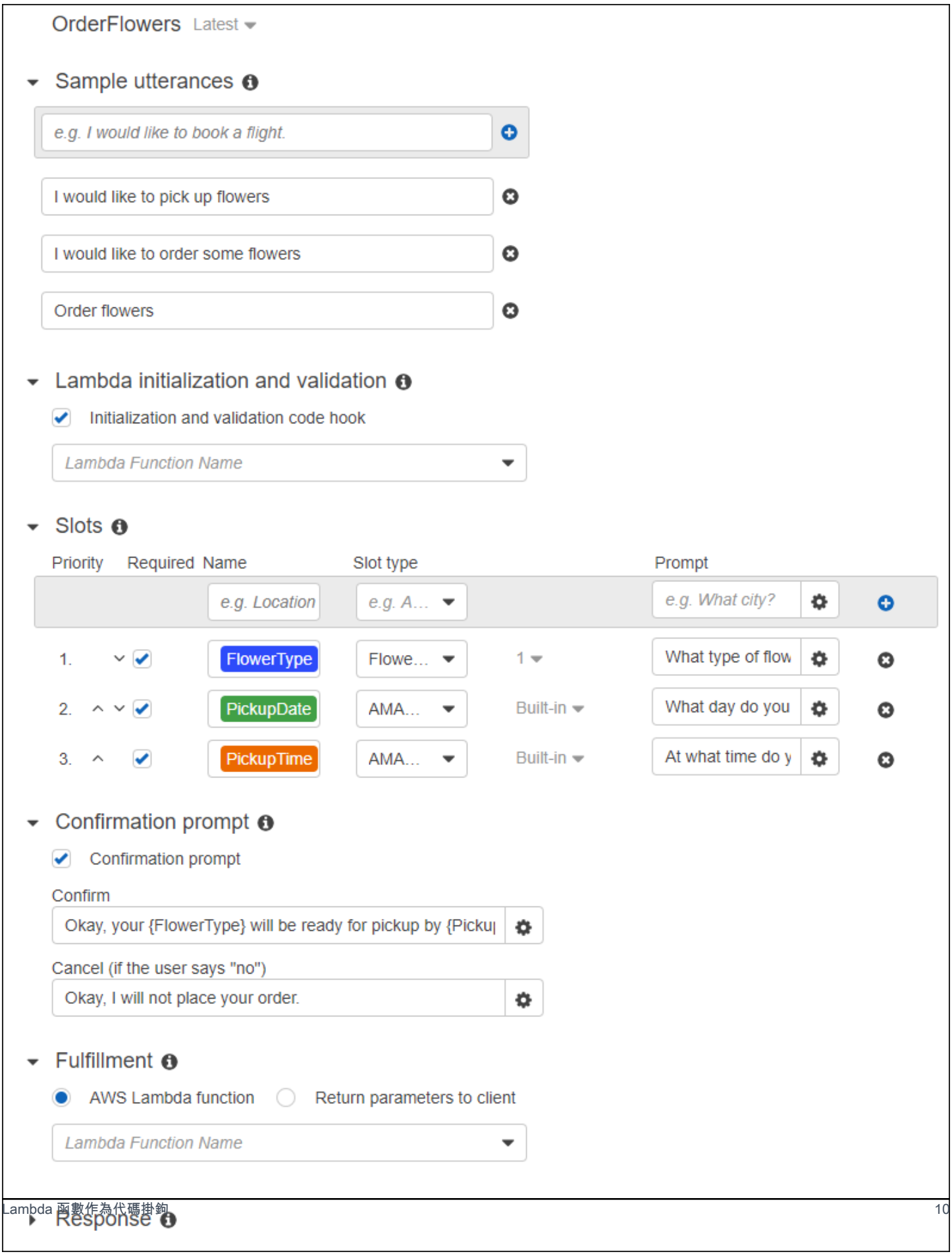
您可以將 Amazon Lex 機器人設定為叫用 Lambda 函數做為程式碼掛接。程式碼掛勾有多個用途：

- 自訂使用者互動 — 例如，當 Joe 要求提供可用的披薩配料時，您可以使用 Joe 選擇的先進知識來顯示配料的子集。
- 驗證使用者的輸入 — 假設 Jen 想要在幾個小時後取花。您可以驗證 Jen 輸入的時間並傳送適當的回應。
- 滿足使用者的意圖 — 在 Joe 提供披薩訂單的所有資訊之後，Amazon Lex 可以叫用 Lambda 函數向當地的比薩店下訂單。

設定意圖時，請在下列位置將 Lambda 函數指定為程式碼掛接：

- 初始化和驗證的對話方塊程式碼掛鉤 — 假設 Amazon Lex 瞭解使用者意圖，則會在每個使用者輸入時叫用此 Lambda 函數。
- 履行代碼掛鉤-在使用者提供完成意圖所需的所有插槽資料之後，便會叫用此 Lambda 函數。

您可以在 Amazon Lex 主控台中選擇意圖並設定程式碼掛接，如下列螢幕擷取畫面所示：


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

您也可以在 dialogCodeHook 操作中使用 fulfillmentActivity 和 PutIntent 欄位來設定程式碼掛勾。

一個 Lambda 函數可以執行初始化、驗證和履行。Lambda 函數收到的事件資料有一個欄位，可將呼叫者識別為對話方塊或履行程式碼掛接。您可以使用此資訊來執行代碼的適當部分。

您可以使用 Lambda 函數來建置可以瀏覽複雜對話方塊的機器人。您可以使用 Lambda 函數回應中的dialogAction欄位來指示 Amazon Lex 採取特定動作。例如，您可以使用對ElicitSlot話方塊動作告訴 Amazon Lex 向使用者詢問不需要的插槽值。如果您已定義釐清提示，則可以在使用者完成前一個意圖時，使用 ElicitIntent 對話方塊動作來引出新的意圖。

如需詳細資訊，請參閱使用 Lambda 函數。

管理訊息

主題

- 訊息的類型
- 用於設定訊息的內容
- 支援的訊息格式
- 訊息群組
- 回應卡

當建立機器人時，您可以設定要它傳送給用戶端的釐清或資訊訊息。請考量下列範例：

- 您可以使用以下釐清提示設定機器人：

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

  如果 Amazon Lex 不了解使用者的意圖，就會將此訊息傳送給用戶端。

- 假設您建立機器人來支援稱為 OrderPizza 的意圖。對於比薩訂單，您需要使用者提供如比薩大小、配料和餅皮種類等資訊。您可以設定以下提示：

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

Amazon Lex 確定使用者訂購披薩的意圖後，就會將這些訊息傳送給用戶端，以取得使用者的資訊。

本節說明在機器人組態中設計使用者互動。

訊息的類型

訊息可以是提示或陳述。

- 提示通常是問題並且預期使用者回應。
- 陳述是提供資訊。它不預期回應。

訊息可以包括槽、工作階段屬性和請求屬性的參考。在執行階段，Amazon Lex 會以實際值取代這些參考。

若要參考已設定的槽值，請使用下列語法：

```
{SlotName}
```

若要參考工作階段屬性，請使用下列語法：

```
[SessionAttributeName]
```

若要參考請求屬性，請使用下列語法：

```
((RequestAttributeName))
```

訊息可以同時包括槽值、工作階段屬性和請求屬性。

例如，假設您在機器人 OrderPizza 意圖中設定以下訊息：

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

此訊息會同時參考槽 (PizzaTopping) 和工作階段屬性 (FirstName 和 DeliveryTime)。在執行階段，Amazon Lex 會以值取代這些預留位置，並將下列訊息傳回給用戶端：

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

若要在訊息中包含方括號 ([]) 或括號 ({})，請使用反斜線 (\) 逸出字元。例如，以下訊息包含大括號和方括號：

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

傳回給用戶端應用程式的文字看起來如下：

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

如需有關會話屬性的資訊，請參閱執行時間 API 操作 PostText 和 PostContent。如需範例，請參閱 預訂行程。

Lambda 函數也可以產生訊息，並將其傳回給 Amazon Lex 以傳送給使用者。如果您在設定意圖時新增 Lambda 函數，則可以動態建立訊息。透過在設定機器人時提供訊息，您就不需要在 Lambda 函數中建構提示。

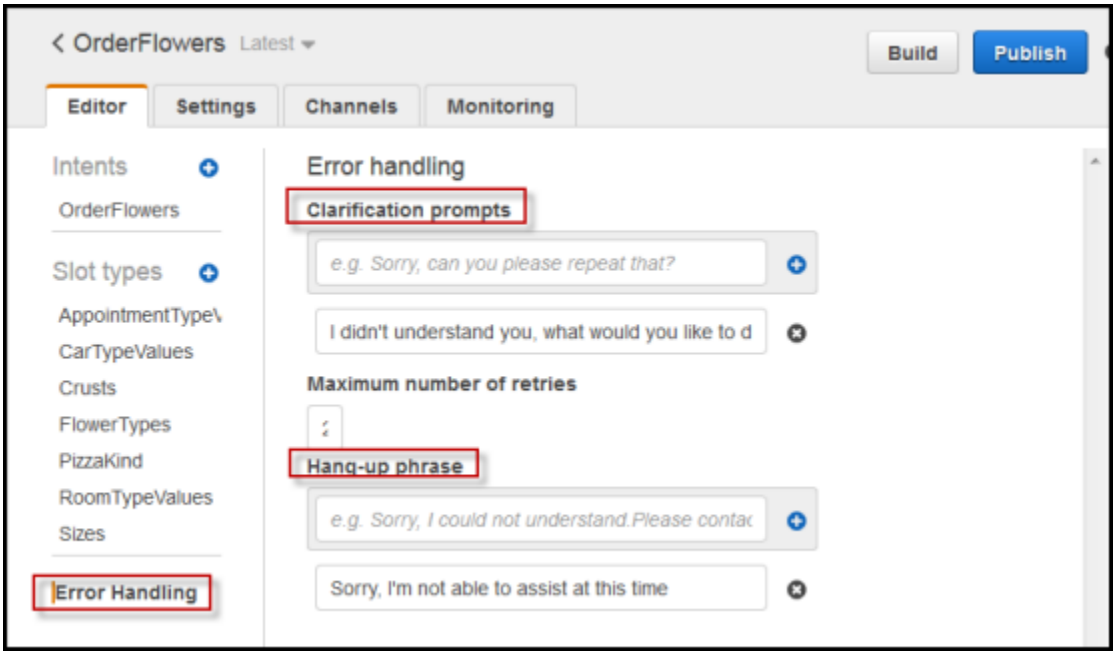
用於設定訊息的內容

當您建立機器人時，您可以在不同的前後關聯中建立訊息，例如機器人中的澄清提示、插槽值提示，以及來自意圖的訊息。Amazon Lex 會在每個環境中選擇適當的訊息以傳回給使用者。您可以針對每個內容提供一組訊息。如果這樣做，Amazon Lex 會隨機從群組中選擇一則訊息。您也可以指定訊息格式或將訊息群組在一起。如需詳細資訊，請參閱支援的訊息格式。

如果您有與意圖相關聯的 Lambda 函數，則可以覆寫在建置階段設定的任何訊息。但是，Lambda 函數不需要使用任何這些消息。

機器人訊息

您可以使用澄清提示和工作階段結束訊息來設定您的機器人。在執行階段，如果 Amazon Lex 不瞭解使用者的意圖，就會使用澄清提示。您可以設定 Amazon Lex 在傳送工作階段結束訊息之前要求澄清的次數。您可以在 Amazon Lex 主控台的「錯誤處理」區段中設定機器人層級訊息，如下圖所示：



使用 API 時，您透過設定 `clarificationPrompt` 操作中的 `abortStatement` 和 PutBot 欄位來設定訊息。

如果您使用含意圖的 Lambda 函數，Lambda 函數可能會傳回回應，指示 Amazon Lex 詢問使用者意圖。如果 Lambda 函數未提供此類訊息，Amazon Lex 會使用澄清提示。

槽提示

您必須為意圖中每個必要的槽指定至少一個提示訊息。在執行階段，Amazon Lex 會使用這些訊息之一來提示使用者提供插槽的值。例如，對於 `cityName` 槽，以下是有效的提示：

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

您可以使用主控台為每個槽設定一或多個提示。您也可以使用 PutIntent 操作建立提示群組。如需詳細資訊，請參閱訊息群組。

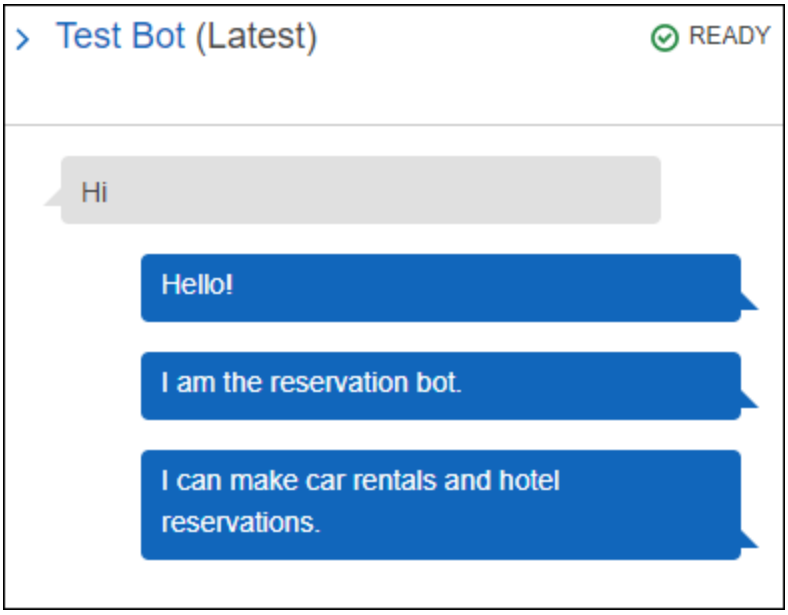
回應

在主控台中，使用 Responses (回應) 區段為您的機器人建立動態、互動的對話。您可以針對一個回應建立一或多個訊息群組。在執行階段，Amazon Lex 透過從每個訊息群組中選取一則訊息來建立回應。如需有關訊息群組的詳細資訊，請參閱訊息群組。

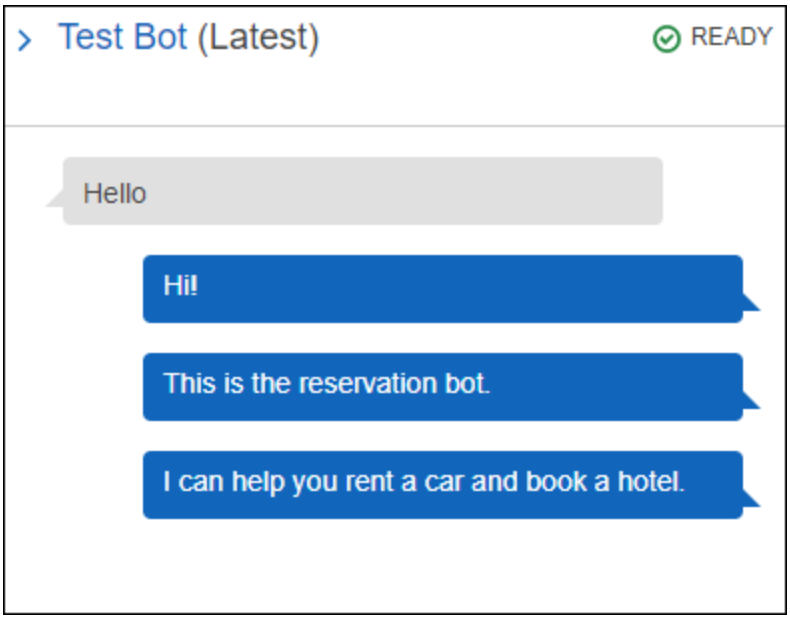
例如，您的第一個訊息群組可以包含不同的問候語：「哈囉」、「嗨」和「您好」。第二個訊息群組可以包含不同形式的簡介：「我是預約機器人」和「這是預約機器人。」 第三個訊息群組可以溝通機器

人的功能：「我可以協助租車和飯店預訂」、「您可以租車與飯店預訂」和「我可以幫您租車和預訂飯店」。

Lex 會從每個訊息群組使用一則訊息，以動態方式在對談中建立回應。例如，一個互動可能是以下內容：

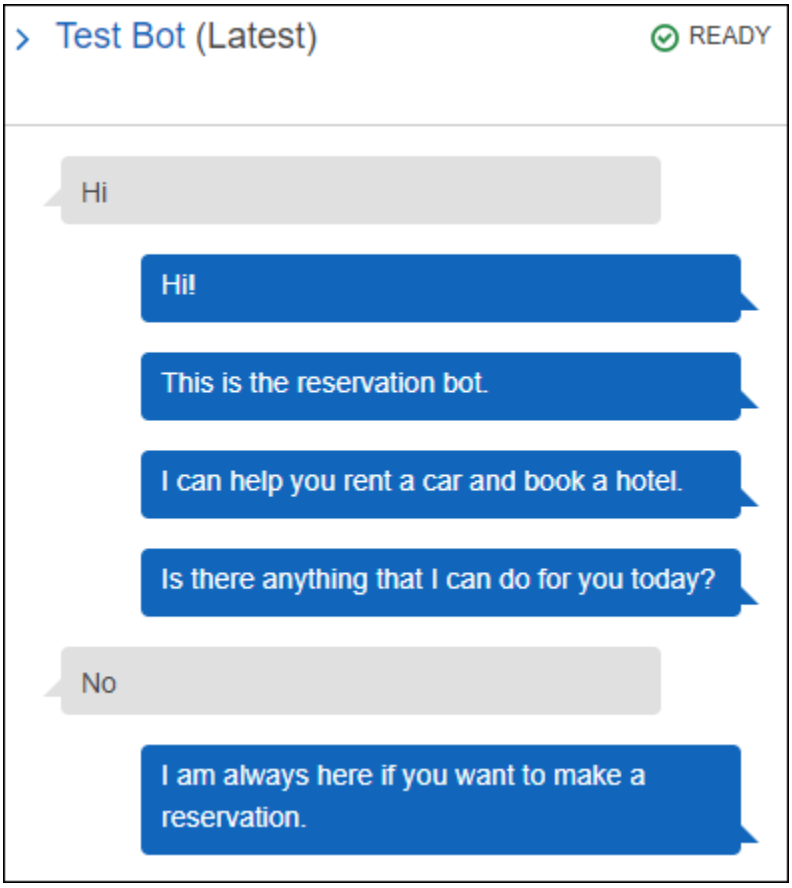


另一個可能是以下內容：



在這兩種情況下，使用者可以新意圖加以回應，例如 `BookCar` 或 `BookHotel` 意圖。

您可以設定機器人在回應中詢問後續問題。例如，對於上述互動，您可以建立使用下列問題第四個訊息群組：「我可以協助租車或預訂飯店？」、「您想要現在預訂嗎？」和「有什麼我可以幫忙的地方嗎？」。對於包括「否」做為回應的訊息，您可以建立後續追蹤提示。以下是範例：



若要建立後續追蹤提示，請選擇 Wait for user reply (等待使用者回覆)。然後輸入當使用者說「否」時，您要傳送的訊息。當建立回應用作為後續追蹤提示時，您還必須在對陳述的回答為「否」時，指定適當的陳述。如需範例，請參閱下列影像：

若要使用 API 新增對意圖的回應，請使用 PutIntent 操作。若要指定回應，請在 conclusionStatement 請求中設定 PutIntent 欄位。若要設定後續追蹤提示，請設定 followUpPrompt 欄位，並包含當使用者表示「否」時要傳送的陳述。您無法同時在相同的意圖上設定 conclusionStatement 欄位和 followUpPrompt 欄位。

支援的訊息格式

當您使用PostText操作時，或在Accept標頭設為的情況下使用PostContent操作時text/plain;charset=utf8，Amazon Lex 支援以下格式的訊息：

- PlainText郵件包含純 UTF-8 文字。
- SSML訊息包含為語音輸出格式化的文字。
- CustomPayload訊息包含您為用戶端建立的自訂格式。您可以定義承載，以符合應用程式的需求。
- Composite訊息是訊息的集合，來自每個訊息群組。如需有關訊息群組的詳細資訊，請參閱訊息群組。

根據預設，Amazon Lex 會傳回針對特定提示定義的任何一則訊息。例如，如果您定義五則訊息來引出插槽值，Amazon Lex 會隨機選擇其中一則訊息並將其傳回給用戶端。

如果您希望 Amazon Lex 在執行階段請求中將特定類型的訊息傳回給用戶端，請設定x-amzn-lex:accept-content-types請求參數。回應僅限於所請求的類型。如果有多個指定類型的 Amazon Lex 訊息，Amazon Lex 會隨機傳回一則訊息。如需有關 x-amz-lex:accept-content-types 標頭的詳細資訊，請參閱設定回應類型。

訊息群組

訊息群組 是對特定提示的一組適當回應。當您希望機器人在對話中以動態方式建立回應時，請使用訊息群組。Amazon Lex 傳回對用戶端應用程式的回應時，會從每個群組隨機選擇一則訊息。您可以為每個回應建立最多 5 個訊息群組。每個群組最多可包含 5 個訊息。如需在主控台中建立訊息群組的範例，請參閱回應。

若要建立訊息群組，您可以使用主控台或使用 PutBot、PutIntent 或 PutSlotType 操作為訊息指派群組號碼。如果您不建立訊息群組，或只建立一個訊息群組，Amazon Lex 會在`Message`欄位中傳送單一訊息。用戶端應用程式只會在主控台中已建立多個訊息群組，或是當您使用 PutIntent 操作建立或更新意圖時建立多個訊息群組時，才會在回應中獲得多個訊息。

Amazon Lex 從群組傳送訊息時，回應的`Message`欄位會包含含有訊息的逸出 JSON 物件。下例顯示當包含多個訊息時，`Message` 欄位的內容。

> **Note**
>
> 範例已經過格式化以利閱讀。回應不包含換行字元 (CR)。

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

您可以設定訊息的格式。格式可為下列其中之一：

- PlainText訊息以 UTF-8 純文字格式顯示。
- SSML — 訊息為語音合成標記語言 (SSML)。
- CustomPayload訊息採用您指定的自訂格式。

若要控制在 `PostContent` 欄位中 `PostText` 和 `Message` 操作所傳回的訊息格式，請設定 `x-amz-lex:accept-content-types` 請求屬性。例如，如果將標頭設定如下，您只會在回應中收到純文字和 SSML 訊息：

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

如果您要求特定的訊息格式，而訊息群組不包含具備該格式的訊息，您會收到 NoUsableMessageException 例外狀況。當使用訊息群組依類型將訊息分組時，請勿使用 x-amz-lex:accept-content-types 標頭。

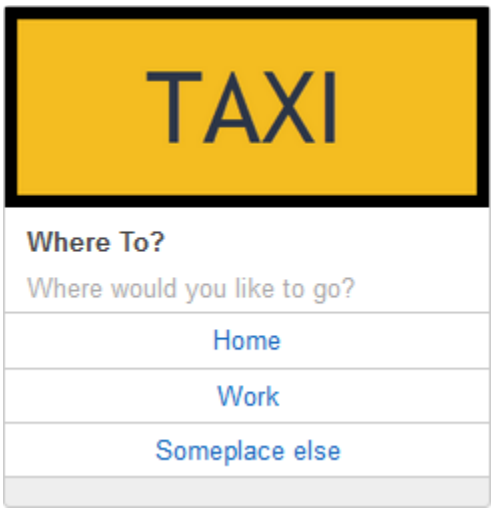
如需有關 x-amz-lex:accept-content-types 標頭的詳細資訊，請參閱設定回應類型。

回應卡

> **Note**
>
> 回應卡不適用於 Amazon Connect 聊天。不過，有關類似功能，請參閱新增互動式訊息至聊天室。

回應卡包含一組對提示適當的回應。使用回應卡透過減少文字互動中的輸入錯誤，可簡化使用者的互動，並提高機器人的準確性。您可以針對 Amazon Lex 傳送到用戶端應用程式的每個提示傳送回應卡。您可以搭配 Facebook、Twilio、Messenger、Slack 和自己的用戶端應用程式使用回應卡。

例如，在計程車應用程式中，您可以在回應卡中設定「家」的選項，並將值設定為使用者的住家地址。當使用者選取此選項時，Amazon Lex 會接收整個地址做為輸入文字。請參閱下圖：



您可以定義回應卡用於以下提示：

- 結論陳述
- 確認提示
- 後續追蹤提示
- 拒絕陳述

- 槽類型表達用語

您只能為每個提示定義一個回應卡。

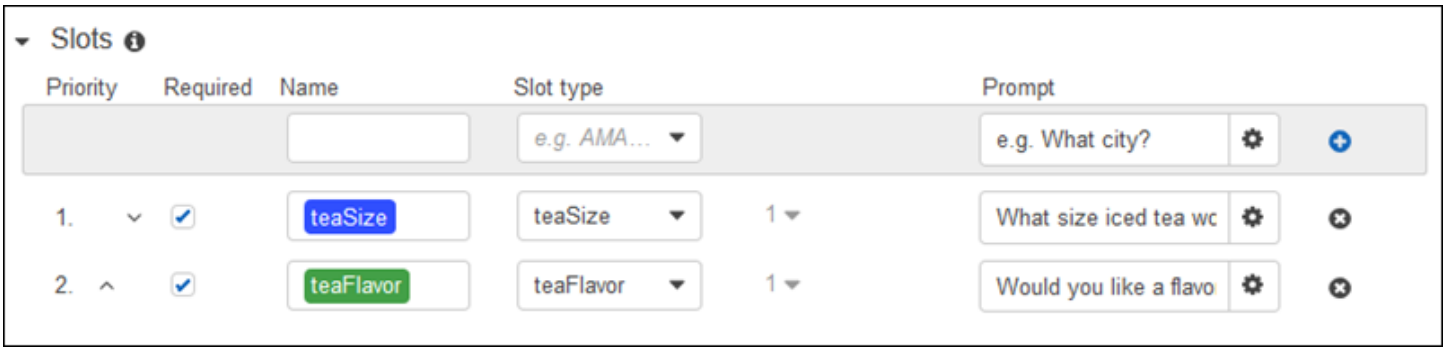
您是在設定意圖時建立回應卡。您可以在建置時間使用主控台或 PutIntent 操作定義靜態回應卡。或者，您可以在 Lambda 函數中在執行階段定義動態回應卡。如果您同時定義靜態和動態回應卡，會以動態回應卡為優先。

Amazon Lex 會以客戶瞭解的格式傳送回應卡。它會針對 Facebook Messenger、Slack 和 Twilio 轉換回應卡。對於其他用戶端，Amazon Lex 會在PostText回應中傳送 JSON 結構。例如，如果客戶端是 Facebook 信使，Amazon Lex 將響應卡轉換為通用模板。如需有關 Facebook Messenger 一般範本的詳細資訊，請參閱 Facebook 網站上的一般範本。如需使用 JSON 結構的範例，請參閱動態產生回應卡。

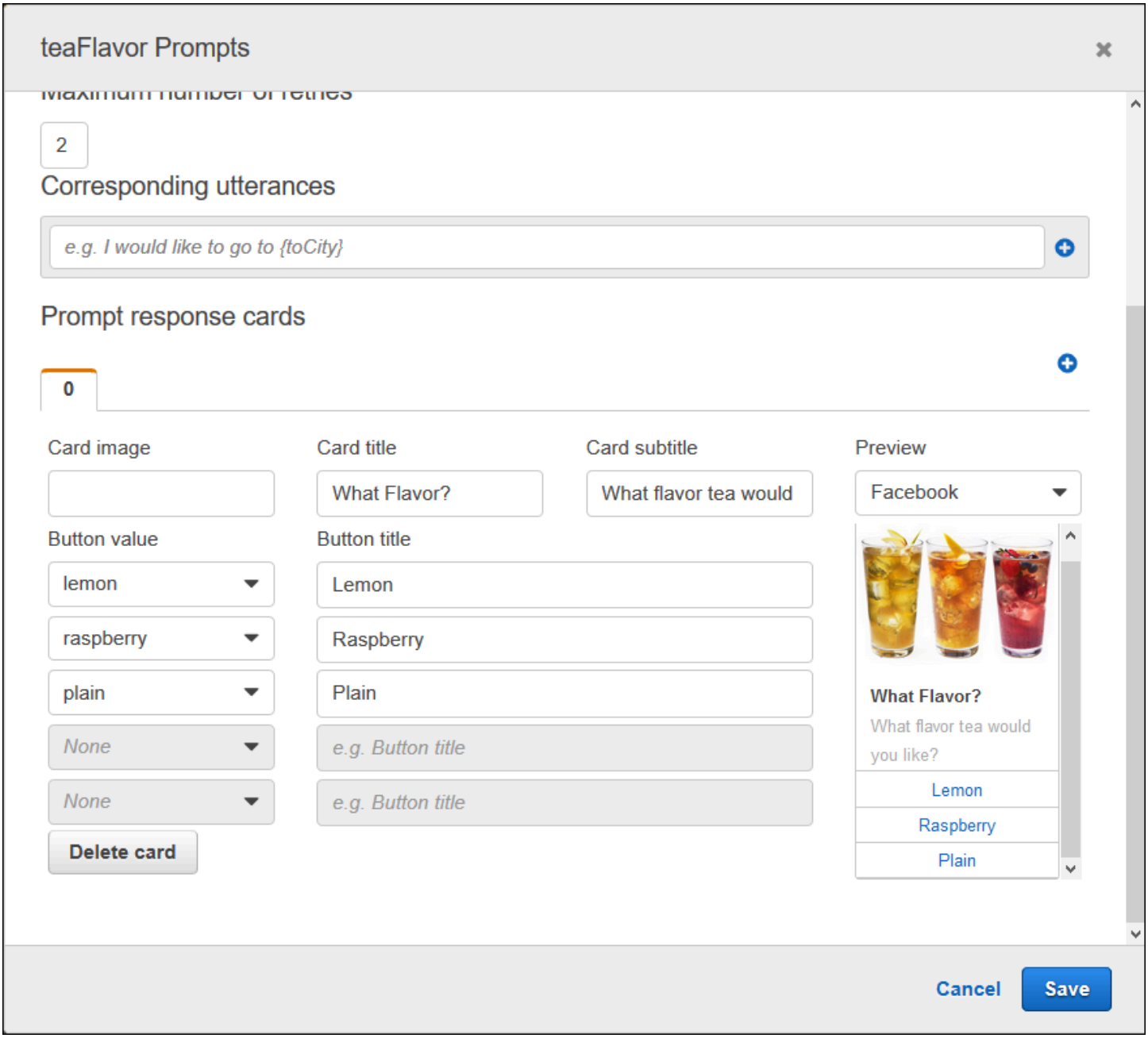
您只能搭配 PostText 操作使用回應卡。您無法搭配 PostContent 操作使用回應卡。

定義靜態回應卡

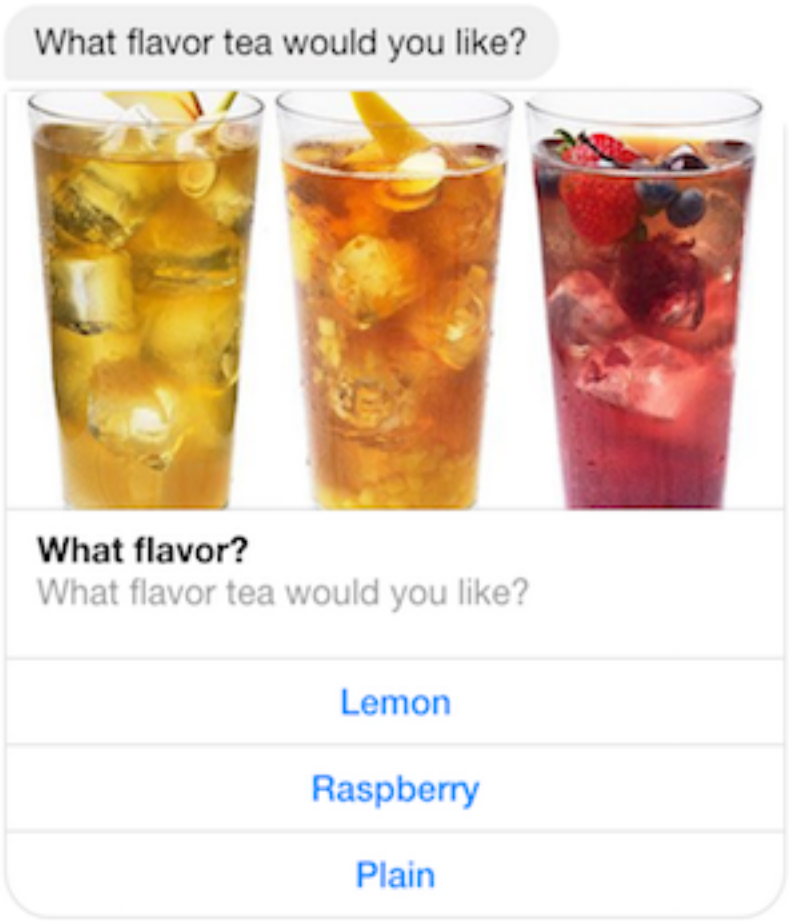
在建立意圖時，使用PutBot作業或 Amazon Lex 主控台定義靜態回應卡。靜態回應卡是與意圖同時定義。請在回應為固定時使用靜態回應卡。假設您要建立具有一個意圖的機器人，當中有個口味的槽。您在定義口味槽時指定提示，如以下主控台螢幕擷取畫面所示：



定義提示時，您可以選擇將回應卡關聯並定義詳細資料與PutBot作業，或者在 Amazon Lex 主控台中定義詳細資訊，如下列範例所示：



現在，假設您已將機器人與 Facebook Messenger 整合。使用者可以按一下按鈕來選擇口味，如下圖所示：



若要自訂回應卡的內容，您可以參考工作階段屬性。在執行階段，Amazon Lex 會以工作階段屬性中的適當值取代這些參考。如需詳細資訊，請參閱設定工作階段屬性。如需範例，請參閱 使用回應卡。

動態產生回應卡

若要在執行階段動態產生回應卡，請針對意圖使用初始化和驗證 Lambda 函數。在 Lambda 函數中執行時決定回應時，請使用動態回應卡。為了回應使用者的輸入，Lambda 函數會產生回應卡，並在回應`dialogAction`區段中傳回回應卡。如需詳細資訊，請參閱回應格式。

以下是來自顯示`responseCard`元素的 Lambda 函數的部分回應。它產生與前一節所示的使用者體驗類似。

```
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"
        }
      ]
    }
  ]
}
```

如需範例，請參閱 安排預約。

管理對話內容

交談內容是使用者、您的應用程式或 Lambda 函數提供給 Amazon Lex 機器人以達成意圖的資訊。交談內容包括使用者提供的插槽資料、要求用戶端應用程式設定的屬性，以及用戶端應用程式和 Lambda 函數建立的工作階段屬性。

主題

- 設定意圖上下文
- 使用預設槽值
- 設定工作階段屬性
- 設定請求屬性
- 設定工作階段逾時
- 在意圖之間共享資訊
- 設定複雜屬性

設定意圖上下文

您可以根據上下文有 Amazon Lex 觸發意圖。上下文是一種狀態變數，可在您定義機器人時與意圖相關聯。

當您使用主控台或使用PutIntent作業建立意圖時，可以配置意圖的前後關聯。您只能在英文 (US) (en-US) 地區設定中使用前後關聯，且僅當您使用PutBot作業建立機器人`true`時將`enableModelImprovements`參數設定為時，才能使用前後關聯。

前後關聯有兩種類型的關係：輸出前後關聯和輸入前後關聯。當一個關聯的意圖被滿足輸出上下文變為活動。輸出內容會在PostText或PostContent作業的回應中傳回至您的應用程式，並針對目前工作階段進行設定。啟動前後關聯後，它會在定義前後關聯時所配置的回合次數或時間限制內保持使用中狀態。

輸入上下文指定在其下的意圖可以被識別的條件。只有當其所有輸入上下文都處於活動狀態時，才能在交談期間識別意圖。沒有輸入上下文的意圖始終符合識別資格。

Amazon Lex 透過滿足輸出內容的意圖，自動管理啟動的環境生命週期。您也可以在呼叫`PostContent`或`PostText`作業時設定使用中前後關聯。

您也可以針對意圖使用 Lambda 函數來設定對話的上下文。Amazon Lex 的輸出內容會傳送至 Lambda 函數輸入事件。Lambda 函數可以在其回應中傳送內容。如需詳細資訊，請參閱Lambda 函數輸入事件和回應格式。

例如，假設您有意圖預訂租車，該租車配置為返回名為「book_car_履約」的輸出內容。當意圖實現時，Amazon Lex 設置輸出上下文變量「book_car_履行」。由於「book_car_fulfill」是一個活動上下文，因此只要將用戶的話語識別為引出該意圖的嘗試，設置為輸入上下文的「book_car_fuled」上下文的意圖被視為識別。您可以將其用於僅在預訂汽車後才有意義的意圖，例如通過電子郵件發送收據或修改預訂。

輸出上下文

當意圖達成時，Amazon Lex 會使意圖的輸出內容處於作用中狀態。您可以使用輸出內容來控制有資格跟進目前意圖的意圖。

每個前後關聯都有工作階段中維護的參數清單。參數是履行意圖的槽值。您可以使用這些參數為其他意圖預先填入插槽值。如需詳細資訊，請參閱 使用預設槽值。

當您使用主控台或使用PutIntent作業建立意圖時，可以設定輸出內容。您可以使用多個輸出內容配置意圖。滿足意圖時，會啟動所有輸出前後關聯，並在PostText或回應中傳PostContent回。

下面顯示了使用控制台將輸出上下文分配給意圖。

當您定義輸出內容時，您也會定義其存留時間、內容包含在 Amazon Lex 回應中的時間長度或圈數。轉向是從您的應用程序向 Amazon Lex 提出的一個請求。一旦圈數或時間過期，上下文就不再處於活動狀態。

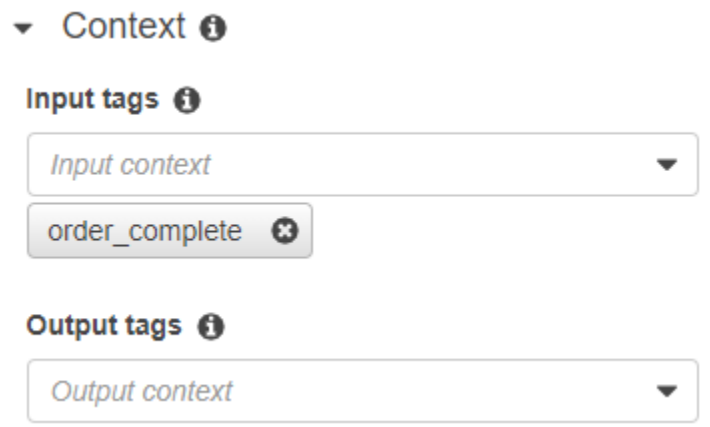
您的應用程式可以視需要使用輸出內容。例如，您的應用程式可以使用輸出內容來：

- 根據上下文更改應用程序的行為。例如，旅遊應用程式對於上下文「已完成的書籍」可能會有不同的動作，而不是「租賃酒店」。
- 將輸出內容傳回 Amazon Lex 做為下一個語音的輸入內容。如果 Amazon Lex 將語音識別為嘗試引出意圖，它會使用上下文來限制可傳回給具有指定上下文的意圖。

輸入上下文

您可以設定輸入內容，以限制識別意圖的交談中的點。沒有輸入上下文的意圖總是有資格被識別。

您可以使用控制台或`PutIntent`操作設置意圖響應的輸入上下文。一個意圖可以有多個輸入上下文。下面顯示了使用控制台將輸入上下文分配給意圖。



對於具有多個輸入上下文的意圖，所有前後關聯都必須處於活動狀態才能觸發意圖。您可以在呼叫PostText、PostContent或PutSession作業時設定輸入內容。

您可以在意圖中配置狹槽，以從目前使用中的前後關聯中取得預設值。Amazon Lex 辨識新意圖但未收到插槽值時，會使用預設值。定義槽時，您可以在塑形中指定前#context-name.parameter-name後關聯名稱和槽名稱。如需詳細資訊，請參閱使用預設槽值。

使用預設槽值

當您使用預設值時，您可以為使用者的輸入未提供插槽時，為新意圖填入插槽值指定來源。此來源可以是先前的對話方塊、要求或工作階段屬性，或是您在建置時設定的固定值。

您可以將下列內容用作預設值的來源。

- 上一個對話方塊 (前後關聯) — #context-name. 參數名稱
- 會話屬性-[屬性名稱]
- 請求屬性 — <attribute-name>
- 固定值-不匹配以前的任何值

當您使用此PutIntent操作將槽加入至意圖時，您可以加入預設值的清單。系統會以列出的順序使用預設值。例如，假設您有一個意圖，其中包含下列定義的槽：

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

辨識出意圖時，名為 "reservation-start-date" 的插槽的值會設定為下列其中一項。

1. 如果 "book-car-fulfilled" 內容處於作用中狀態，則會使用「startDate」參數的值作為預設值。
2. 如果 "book-car-fulfilled" 前後關聯未處於作用中狀態，或未設定「startDate」參數，則會使用 "reservationStartDate" 階段作業屬性的值作為預設值。
3. 如果沒有使用前兩個預設值，則插槽沒有預設值，Amazon Lex 會如常引出值。

如果插槽使用預設值，即使是必要的，也不會產生插槽。

設定工作階段屬性

工作階段屬性包含在工作階段期間在機器人與用戶端應用程式之間傳遞的應用程式特定資訊。Amazon Lex 會將工作階段屬性傳遞給為機器人設定的所有 Lambda 函數。如果 Lambda 函數新增或更新工作階段屬性，Amazon Lex 會將新資訊傳回給用戶端應用程式。例如：

- 在練習 1：使用藍圖 (主控台) 建立 Amazon Lex 機器人 中，範例機器人使用 `price` 工作階段屬性，來維持花朵的價格。Lambda 函數根據訂購的花的類型設置此屬性。如需詳細資訊，請參閱步驟 5 (選用)：檢閱資訊流程的詳細資訊 (主控台)。
- 在預訂行程 中，範例機器人使用 `currentReservation` 工作階段屬性在預訂飯店或預訂租車的對話期間，維持槽類型資料的副本。如需詳細資訊，請參閱資訊流程的詳細資訊。

在 Lambda 函數中使用工作階段屬性來初始化機器人，並自訂提示和回應卡。例如：

- 初始化 — 在薄餅訂購機器人中，用戶端應用程式會在第一次呼叫PostContent或PostText作業時，將使用者的位置做為工作階段屬性傳遞。例如：`"Location": "111 Maple Street"`。Lambda 函數使用此信息來查找最接近的比薩店下訂單。
- Personalize 提示 — 配置提示和回應卡以參考工作階段屬性。例如，「嘿 [FirstName]，你想要什麼澆頭？」 如果您將使用者的名字作為工作階段屬性 (`{"FirstName": "Jo"}`) 傳遞，Amazon Lex 會取代預留位置的名稱。它接著會傳送個人化提示給使用者、「Jo 你好，你想要什麼配料？」

工作階段屬性會在工作階段期間內持續存在。Amazon Lex 將它們存放在加密的資料存放區中，直到工作階段結束為止 用戶端可以透過呼叫 PostContent 或 PostText 操作並將 `sessionAttributes` 欄位設定為值，在請求中建立工作階段屬性。Lambda 函數可以在響應中創建會話屬性。在用戶端或 Lambda 函數建立工作階段屬性之後，只要用戶端應用程式未在 Amazon Lex 的請求中包含`sessionAttribute`欄位，就會使用儲存的屬性值。

例如，假設您有兩個工作階段屬性，`{"x": "1", "y": "2"}`。如果用戶端在未指定`sessionAttributes`欄位的情況下呼叫`PostContent`或`PostText`作業，Amazon Lex 會呼叫具

有已存工作階段屬性的 Lambda 函數 (`{"x": 1, "y": 2}`)。如果 Lambda 函數未傳回工作階段屬性，Amazon Lex 會將儲存的工作階段屬性傳回給用戶端應用程式。

如果用戶端應用程式或 Lambda 函數傳遞工作階段屬性，Amazon Lex 會更新儲存的工作階段屬性。傳遞現有的值，例如 `{"x": 2}`，會更新儲存的值。如果您傳送一組新的工作階段屬性，例如 `{"z": 3}`，現有的值會被移除，只保留新值。當傳遞空白對應 `{}` 時，會清除儲存的值。

若要將工作階段屬性傳送至 Amazon Lex，您必須建 string-to-string立屬性對應。以下說明如何對應工作階段屬性：

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

對於 `PostText` 操作，您使用 `sessionAttributes` 欄位將對應插入請求的本文，如下所示：

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

對於 `PostContent` 操作，您用 base64 來編碼對應，然後將其做為 `x-amz-lex-session-attributes` 標頭傳送。

如果您在工作階段屬性中傳送二進位或結構化資料，必須先將資料轉換為簡單的字串。如需詳細資訊，請參閱設定複雜屬性。

設定請求屬性

請求屬性包含請求特定的資訊並且僅適用於目前的請求。用戶端應用程式會將此資訊傳送至 Amazon Lex。使用請求屬性來傳遞不需要在整個工作階段內保留的資訊。您可以建立自己的請求屬性，也可以使用預先定義的屬性。若要傳送請求屬性，請在 `x-amz-lex-request-attributes` 中使用 the section called “PostContent” 標頭，或在 `requestAttributes` 請求中使用 the section called “PostText” 欄位。請求屬性並不像工作階段屬性會在請求之間保留，因此 `PostContent` 或 `PostText` 回應中不會傳回請求屬性。

> **Note**
>
> 若要傳送在請求之間保留的資訊，請使用工作階段屬性。

命名空間 x-amz-lex: 是預留給預先定義的請求屬性。請勿建立以 x-amz-lex: 為字首的請求屬性。

設定預先定義的請求屬性

Amazon Lex 提供預先定義的請求屬性，以管理其處理傳送至機器人之資訊的方式。該屬性不會在整個工作階段內保留，因此您必須在每個請求中傳送預先定義的屬性。預先定義的屬性全都位於 x-amz-lex: 命名空間中。

除了下列預先定義的屬性之外，Amazon Lex 還為簡訊平台提供預先定義的屬性。如需該些屬性的清單，請參閱在簡訊平台上部署 Amazon Lex 機器人。

設定回應類型

如果您有兩個具有不同功能的用戶端應用程式，可能需要限制回應中的訊息格式。例如，您可能想要將傳送到 Web 用戶端的訊息限制為純文字，但是讓行動用戶端可以同時使用純文字和語音合成標記語言 (SSML)。若要設定 PostContent 和 PostText 操作傳回的訊息格式，請使用 x-amz-lex:accept-content-types" 請求屬性。

您可以將屬性設定為以下訊息類型的任何組合：

- PlainText郵件包含純 UTF-8 文字。
- SSML訊息包含為語音輸出格式化的文字。
- CustomPayload訊息包含您為用戶端建立的自訂格式。您可以定義承載，以符合應用程式的需求。

Amazon Lex 只會傳回回應Message欄位中具有指定類型的訊息。您可以用逗號分隔各值來設定多個值。如果您使用訊息群組，那麼每個訊息群組都必須至少包含一個指定類型的訊息。否則，您會收到 NoUsableMessageException 錯誤。如需詳細資訊，請參閱訊息群組。

> **Note**
>
> x-amz-lex:accept-content-types 請求屬性不會影響 HTML 本文的內容。PostText 操作回應的內容一律是 UTF-8 純文字。PostContent 操作回應的本文在請求中包含的資料具備 Accept 標頭中所設定的格式。

設定偏好的時區

若要設定用於解析日期的時區，以便與使用者的時區相關，請使用 x-amz-lex:time-zone 請求屬性。如果您在 x-amz-lex:time-zone 屬性中沒有指定時區，預設值會依您使用機器人的區域而定。

區域	預設時區
美國東部 (維吉尼亞北部)	America/New_York
美國西部 (奧勒岡)	America/Los_Angeles
亞太區域 (新加坡)	Asia/Singapore
亞太區域 (雪梨)	Australia/Sydney
亞太區域 (東京)	Asia/Tokyo
歐洲 (法蘭克福)	Europe/Berlin
歐洲 (愛爾蘭)	Europe/Dublin
歐洲 (倫敦)	Europe/London

例如，如果用戶響應提示「您希望包裹tomorrow在哪一天交付？」 封裝遞送的實際日期取決於使用者的時區。例如，紐約的 9 月 16 日 01:00 是洛杉磯 9 月 15 日的 22:00。如果您的服務在美國東部 (維吉尼亞北部) 區域執行，而洛杉磯的人員使用預設時區「明天」訂購要遞送的包裹，則包裹將在 17 日遞送，而非 16 日遞送。不過，如果您將 x-amz-lex:time-zone 請求屬性設定為 America/Los_Angeles，則包裹會在 16 日送達。

您可以將屬性設定為任何網際網路號碼分配局 (IANA) 時區名稱。如需時區名稱的清單，請參閱 Wikipedia 上的 tz 資料庫時區清單。

設定使用者定義的請求屬性

使用者定義的請求屬性是您在每個請求中傳送給機器人的資料。您在 amz-lex-request-attributes 請求中的 PostContent 標頭，或在 requestAttributes 請求中 PostText 欄位傳送資訊。

若要將請求屬性傳送至 Amazon Lex，您必須建立屬性 string-to-string 對應。以下說明如何對應請求屬性：

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

對於 PostText 操作，您使用 requestAttributes 欄位將對應插入請求的本文，如下所示：

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

對於 PostContent 操作，您用 base64 來編碼對應，然後將其做為 x-amz-lex-request-attributes 標頭傳送。

如果您在請求屬性中傳送二進位或結構化資料，必須先將資料轉換為簡單的字串。如需詳細資訊，請參閱設定複雜屬性。

設定工作階段逾時

Amazon Lex 會保留內容資訊 (插槽資料和工作階段屬性)，直到交談工作階段結束為止。若要控制機器人的工作階段可持續多久的時間，請設定工作階段逾時。在預設情況下，工作階段持續時間為 5 分鐘，但是您可以指定介於 0 到 1,440 分鐘 (24 小時) 的任何持續時間。

例如，假設您建立一個 ShoeOrdering 機器人來支援如 OrderShoes 和 GetOrderStatus 的意圖。當 Amazon Lex 偵測到使用者的意圖是訂購鞋子時，它會要求提供插槽資料。例如，它會要求鞋子尺寸、顏色、品牌等。如果使用者提供部分插槽資料，但未完成鞋子購買，Amazon Lex 會記住整個工作階段的所有插槽資料和工作階段屬性。如果使用者在工作階段過期之前返回工作階段，可提供其餘的槽資料，並完成購買。

在 Amazon Lex 主控台中，您可以在建立機器人時設定工作階段逾時。使用 AWS 命令列界面 (AWS CLI) 或 API，您可以在建立或更新具有PutBot操作的機器人時，透過設定 IdlesSsionttlInSeconds 欄位來設定逾時。

在意圖之間共享資訊

Amazon Lex 支援在意圖之間共用資訊。若要在意圖之間共享，請使用工作階段屬性。

例如，ShoeOrdering 機器人的使用者開始訂購鞋子。機器人會與使用者進行對話，例如，收集鞋子尺寸、顏色和品牌等槽資料。當使用者下訂單時，履行訂單的 Lambda 函數會設定orderNumber工作階段屬性，其中包含訂單編號。為了取得訂單狀態，使用者使用 GetOrderStatus 意圖。機器人可以要求使用者提供槽資料，例如訂單號碼和訂單日期。當機器人擁有所需的資訊時，便會傳回訂單狀態。

如果您認為使用者可能會在同一個工作階段期間切換意圖，可以設計機器人傳回最近的訂單狀態。與其再次要求使用者提供訂單資訊，您可以使用 orderNumber 工作階段屬性跨意圖共享資訊，並滿足 GetOrderStatus 意圖。機器人可傳回使用者最後一個訂單的狀態來達到此目的。

如需跨意圖資訊共享的範例，請參閱預訂行程。

設定複雜屬性

會話和請求屬性是屬性和值的 string-to-string 映射。在許多情況下，您可以使用字串對應在用戶端應用程式與機器人之間傳輸屬性值。不過，在某些情況下，您可能需要傳輸無法輕易轉換為字串對應的二進位資料或複雜架構。例如，以下 JSON 物件代表美國三個最熱門的城市陣列：

```
{
   "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"
         }
      }
   ]
}
```

這個數據數組不能很好地轉換為 string-to-string地圖。在這種情況下，您可以將物件轉換成簡單的字串，讓您可以透過 PostContent 和 PostText 操作將其傳送到機器人。

例如，如果您正在使用 JavaScript，則可以使用`JSON.stringify`操作將對象轉換為 JSON，並將 JSON 文本轉換為 JavaScript 對象的`JSON.parse`操作：

```
// 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);
```

若要傳送作業的工PostContent作階段屬性，您必須在將屬性新增至要求標頭之前，base64 對屬性進行編碼，如下列 JavaScript程式碼所示：

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

您可以先將資料轉換成以 base64 編碼的字串，然後將該字串當做值在工作階段屬性中傳送，藉此將二進位資料傳送到 PostContent 和 PostText 操作：

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

使用可信度分數

當用户發言時，Amazon Lex 使用自然語言理解 (NLU) 來理解用户的請求並返回正確的意圖。默認情況下，Amazon Lex 會返回您的機器人定義的最有可能的意圖。

在某些情況下，Amazon Lex 可能難以確定最可能的意圖。例如，用户可能會發表模糊的言論，或者可能有兩種相似的意圖。為了幫助確定正確的意圖，您可以將您的域知識與可信度分數替代意圖清單。置信度分數是 Amazon Lex 提供的評級，表明意圖是正確意圖是否具有多大的信心。

要確定兩種備選方法之間的差異，您可以比較它們的置信度分數。例如，如果一個意圖的置信度得分為 0.95，而另一個意圖的得分為 0.65，則第一個意圖可能是正確的。但是，如果一個意圖的得分為 0.75，而另一個意圖的分數為 0.72，則兩種意圖之間存在模糊性，即您可能能夠在應用程序中使用域知識進行區分。

您還可以使用置信度分數創建測試應用程序，以確定對意圖語音的更改是否會對機器人的行為產生影響。例如，您可以使用一組語音獲取機器人意圖的置信度分數，然後用新的語音更新意圖。然後，您可以檢查置信度分數，以查看是否有改善。

Amazon Lex 返回的置信度分數是比較值。你不應該依靠他們作為一個絕對分數。這些值可能會根據 Amazon Lex 的改進而發生變化。

當您使用置信度分數時，Amazon Lex 會在每個響應中返回最有可能的意圖和最多 4 個備選意圖及其相關分數。如果所有置信度得分都小於閾值，則 Amazon Lex 將包

含AMAZON.FallbackIntent，AMAZON.KendraSearchIntent，或者兩者（如果您已配置它們）。您可以使用默認閾值，或設定您自己的閾值。

下列程式碼顯示alternativeIntents區段中的PostTextoperation.

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

在創建或更新自動程序時設置閾值。您可以使用 API 或 Amazon Lex 控台。對於下面列出的區域，您需要選擇加入才能提高準確度和置信度分數。在控制台中，選擇進階選項區段。使用 API，將enableModelImprovements參數時調用PutBotoperation. ：

- 美國東部 (維吉尼亞北部) (us-east-1)
- 美國西部 (奧勒岡) (us-west-2)
- 亞太區域 (雪梨) (ap-southeast-2)
- 歐洲 (愛爾蘭) (eu-west-1)

在所有其他區域，默認情況下可用精度改進和置信度分數支持。

要更改置信閾值，請在控制台中設置該閾值，或使用PutBotoperation. 閾值必須是 1.00 和 0.00 之間的數字。

要使用控制台，請在創建或更新自動程序時設置置信度閾值。

在創建機器人時設置置信閾值（控制台）

- 在 上建立機器人中，在可信度分數閾值欄位。

更新置信閾值（控制台）

1. 從機器人清單中，選擇要更新的機器人。

2. 選擇 Settings (設定) 索引標籤。
3. 在左側導覽中，選擇將軍。
4. 更新可信度分數閾值欄位。

設置或更新置信閾值 (SDK)

- 將`nluIntentConfidenceThreshold`參數PutBotoperation. 以下 JSON 代碼顯示了正在設置的參數。

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

工作期管理

要更改 Amazon Lex 在與用户對話中使用的意圖，您可以使用對話框代碼鈎子 Lambda 函數中的響應，也可以在自定義應用程序中使用會話管理 API。

使用 Lambda 函數

當您使用 Lambda 函數時，Amazon Lex 會使用包含函數輸入的 JSON 結構調用它。JSON 結構包含一個名為`currentIntent`，其中包含 Amazon Lex 確定為用户發言的最可能意圖的意圖。JSON 結構還包括`alternativeIntents`字段，該字段包含最多四個可能滿足用户意圖的附加意圖。每個意圖都包含一個名為`nluIntentConfidenceScore`，其中包含 Amazon Lex 分配給意圖的置信度。

要使用替代意圖，您可以在`ConfirmIntent`或`ElicitSlot`對話框操作。

如需詳細資訊，請參閱 使用 Lambda 函數。

使用工作區段管理 API

要使用與當前意圖不同的意圖，請使用PutSessionoperation. 例如，如果您決定第一個替代方案比 Amazon Lex 選擇的意圖更可取，則可以使用`PutSession`操作來更改意圖，以便用户與之交互的下一個意圖是您選擇的目的。

如需詳細資訊，請參閱使用 Amazon Lex API 管理工作階段。

對話日誌

您可以啟用「對話日誌」來存放機器人互動。您可以使用這些日誌來檢閱機器人的效能，以及疑難排解對話問題。您可以記錄 PostText 操作的文字。您可以同時記錄 PostContent 操作的文字和音訊。透過啟用對話日誌，您可以詳細檢視使用者與您機器人的對話。

例如，具有您機器人的工作階段有一個工作階段 ID。您可以使用此 ID 來取得對話的記錄，包括使用者表達用語和對應的機器回應。您還可以取得中繼資料，例如表達用語的意圖名稱和槽值。

> **Note**
>
> 由於受到兒童線上隱私保護法案 (COPPA) 的限制，您無法搭配機器人使用對話日誌。

對話日誌是針對別名設定的。每個別名都可以對其文字和音訊日誌具有不同的設定。您可以為每個別名啟用文字日誌、音訊日誌或兩者。文字日誌會將文字輸入、音訊輸入的文字、音訊輸入的文字輸入、音訊輸入 CloudWatch 的文字輸 音訊日誌會將音訊輸入存放在 Amazon S3 中。您可以使用 AWS KMS 客戶管理的 CMK 來啟用文字和音訊日誌的加密。

若要設定記錄，請使用主控台或 PutBotAlias 操作。您無法記錄機器人$LATEST別名或 Amazon Lex 主控台中可用的測試機器人的對話。啟用別名的交談記錄PostContent或該別名的PostText作業後，會在已設定的日誌記錄群組或 S3 儲存貯體中 CloudWatch 記錄文字或音訊語音。

主題

交談記錄的 IAM 政策

視您選取的記錄類型而定，Amazon Lex 需要使用 Amazon CloudWatch 日誌和 Amazon Simple Storage Service (S3) 貯體來存放日誌的許可。您必須建立AWS Identity and Access Management角色和許可，才能讓 Amazon Lex 存取這些資源。

建立對話日誌的 IAM 角色和政策

若要啟用交談日誌，您必須授與 CloudWatch 日誌和 Amazon S3 的寫入權限。如果您為 S3 物件啟用物件加密，則需要將存取許可授與用來加密物件的 AWS KMS 金鑰。

您可以使用 IAMAWS Management Console、IAM API 或建AWS Command Line Interface立角色和政策。這些指示會使用 AWS CLI 來建立角色和政策。如需有關使用者指南中的在 AWS Identity Management 使用者指南中[的在 AWS Identity and Access Management 用者指南中的在 JSON 標籤上建立政策](#)。

> **Note**
>
> 下列程式碼是針對 Linux 和 MacOS 格式化的。若為 Windows，請將接續字元 (\) 取代為插入符號 (^)。

建立對話日誌的 IAM 角色

1. 在稱為 **LexConversationLogsAssumeRolePolicyDocument.json** 的目前目錄中建立一個文件、將下列程式碼新增至其中，然後儲存它。本政策文件將 Amazon Lex 新增為該角色的受信任實體。這可讓 Lex 擔任將日誌傳遞至專為對話日誌設定的資源角色。

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

2. 在中AWS CLI，執行下列命令以建立對話日誌的 IAM 角色。

```
aws iam create-role \
    --role-name role-name \
```

```
    --assume-role-policy-document file://
LexConversationLogsAssumeRolePolicyDocument.json
```

接下來，建立政策並將其附加到可讓 Amazon Lex 寫入 CloudWatch 日誌的角色。

建立 IAM 政策以將對話文字日誌記錄至 Lo CloudWatch gs

1. 在目前名為的目錄中建立文件**LexConversationLogsCloudWatchLogsPolicy.json**，將下列 IAM 政策新增至該文件，然後儲存。

```
{
  "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. 在中AWS CLI，建立將寫入權限授與記 CloudWatch 錄日誌群組的 IAM 政策。

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

3. 將政策連接至您為對話日誌建立的 IAM 角色。

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

如果您要將音訊記錄到 S3 儲存貯體，請建立可讓 Amazon Lex 寫入儲存貯體的政策。

建立 IAM 政策以用於音訊日誌傳送給 S3 儲存貯體的音訊日誌

1. 在稱為 **LexConversationLogsS3Policy.json** 的目前目錄中建立一個文件、將下列政策新增至其中，然後儲存它。

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

2. 在中AWS CLI，建立可授與 S3 儲存貯體寫入權限的 IAM 政策。

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

3. 將此政策附加到您為對話日誌建立的角色。

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

授與傳遞 IAM 角色的權限

當您使用主控台AWS Command Line Interface、或AWS SDK 指定用於交談日誌的 IAM 角色時，指定交談日誌 IAM 角色的使用者必須具有將角色傳遞給 Amazon Lex 的權限。為了讓使用者將角色傳遞至 Amazon Lex，您必須將`PassRole`許可授予使用者、角色或群組。

下列政策會定義授予使用者、角色或群組的許可。您可以使用 `iam:AssociatedResourceArn` 和 `iam:PassedToService` 條件金鑰來限制許可的範圍。有關詳情，請參閱授予使用者權限以將角色傳遞給AWS服務和AWS Identity and Access Management使用者指南中的 IAM 和AWS STS條件上下文金鑰。

```
{
    "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"
                }
            }
        }
    ]
}
```

設定對話日誌

您可以使用主控台或 PutBotAlias 操作的 conversationLogs 欄位，啟用和停用對話日誌。您可以開啟或關閉音訊日誌、文字日誌或兩者。記錄會在新的機器人工作階段開始。對於作用中的工作階段，不會反映日誌設定的變更。

若要存放文字日誌，請在您的AWS帳戶中使用 Amazon CloudWatch 日誌日誌群組。您可以使用任何有效的日誌群組。日誌群組必須和 Amazon Lex 機器人位於相同的區域中。如需有關建立 CloudWatch 日誌群組的詳細資訊，請參閱《Amazon Lo CloudWatch gs 使用者指南》中的使用者指南中的使用者指南中的使用者指南中

若要存放音訊日誌，請在您的AWS帳戶中使用 Amazon S3 儲存貯體。您可以使用任何有效的 S3 儲存貯體。儲存貯體必須和 Amazon Lex 機器人位於相同的區域中。如需有關建立 S3 儲存貯體的詳細資訊，請參閱《Amazon Simple Storage Service 入門指南》中的建立儲存貯體。

您必須提供 IAM 角色，其中包含允許 Amazon Lex 寫入已設定的日誌群組或儲存貯體的政策。如需詳細資訊，請參閱建立對話日誌的 IAM 角色和政策。

如果您使用建立服務連結角色AWS Command Line Interface，則必須使用下列custom-suffix選項將自訂尾碼新增至角色：

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

您用來啟用交談記錄的 IAM 角色必須具有iam:PassRole權限。下列政策應連接至角色。

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

啟用對話日誌

使用主控台開啟日誌

1. 打開 Amazon Lex 控制台 https://console.aws.amazon.com/lex。
2. 從清單中選擇一個機器人。
3. 選擇 Settings (設定) 標籤，然後從左側功能表中選擇 Conversation logs (對話日誌)。
4. 在別名清單中，為您要設定對話日誌的別名選擇設定圖示。
5. 選取要記錄文字、音訊或兩者。
6. 對於文字記錄，請輸入 Amazon CloudWatch 日誌記錄群組名稱。
7. 若要記錄音訊，請輸入 S3 儲存貯體資訊。
8. 選用。若要加密音訊日誌，請選擇要用於加密的 AWS KMS 金鑰。
9. 選擇具有必要許可的 IAM 角色。
10. 選擇 Save (儲存) 以開始記錄對話。

使用 API 開啟文字日誌

1. 使用 conversationLogs 欄位的 logSettings 成員中的項目呼叫 PutBotAlias 操作
   - 將 destination 成員設定為 CLOUDWATCH_LOGS

- 將 logType 成員設定為 TEXT
- 將resourceArn成員設定為日誌目標之日誌群組 (ARN) 的日誌群組的 CloudWatch 日誌群組的日誌群組的 Amazon Resource Name (ARN)

2. 將iamRoleArn IAM 角色的 Amazon Resource Name (ARN) 設定為在指定 Resource Name (ARN)。conversationLogs

使用 API 開啟音訊日誌

1. 使用 conversationLogs 欄位的 logSettings 成員中的項目呼叫 PutBotAlias 操作
   - 將 destination 成員設定為 S3
   - 將 logType 成員設定為 AUDIO
   - 將 resourceArn 成員設定為音訊日誌儲存所在之 Amazon S3 儲存貯體的 ARN
   - 選用。若要使用特定 AWS KMS 金鑰加密音訊日誌，請設定金鑰 ARN 的 kmsKeyArn 成員，而此金鑰用於加密。
2. 將iamRoleArn IAM 角色的 Amazon Resource Name (ARN) 設定為在指定 Resource Name (ARN)。conversationLogs

停用對話日誌

使用主控台關閉日誌

1. 打開 Amazon Lex 控制台 https://console.aws.amazon.com/lex。
2. 從清單中選擇一個機器人。
3. 選擇 Settings (設定) 標籤，然後從左側功能表中選擇 Conversation logs (對話日誌)。
4. 在別名清單中，為您要設定對話日誌的別名選擇設定圖示。
5. 清除文字、音訊或兩者的核取方塊以關閉記錄。
6. 選擇 Save (儲存) 以停止記錄對話。

使用 API 關閉日誌

- 呼叫沒有 conversationLogs 欄位的 PutBotAlias 操作。

使用 API 關閉文字日誌

- 如果您是記錄音訊
  - 呼叫只對 AUDIO 具有 logSettings 項目的 PutBotAlias 操作。
  - 對 PutBotAlias 操作的呼叫必須沒有 TEXT 的 logSettings 項目。
- 如果您不是記錄音訊
  - 呼叫沒有 conversationLogs 欄位的 PutBotAlias 操作。

使用 API 關閉音訊日誌

- 如果您是記錄文字
  - 呼叫只對 TEXT 具有 logSettings 項目的 PutBotAlias 操作。
  - 對 PutBotAlias 操作的呼叫必須沒有 AUDIO 的 logSettings 項目。
- 如果您不是記錄文字
  - 呼叫沒有 conversationLogs 欄位的 PutBotAlias 操作。

加密對話日誌

您可以使用加密來協助保護對話日誌的內容。對於文字和音訊日誌，您可以使用AWS KMS客戶受管 CMK 來加密 CloudWatch 日誌日誌群組和 S3 儲存貯體中的資料。

> **Note**
>
> Amazon Lex 僅支援對稱 CMK。請不要使用非對稱 CMK 來加密資料。

您可以使用 Amazon Lex 用於文字 CloudWatch 日誌的日誌記錄群組上的AWS KMS金鑰來啟用加密。您無法在日誌設定中提供 AWS KMS 金鑰，來啟用日誌群組的 AWS KMS 加密。如需詳細資訊，請參閱《Amazon CloudWatch CloudWatch Logs Logs 使用者指南》AWS KMS中的使用者指南中的使用者指南中

對於音訊日誌，您可以在 S3 儲存貯體上使用預設加密，或指定 AWS KMS 金鑰來加密音訊物件。即使您的 S3 儲存貯體使用預設加密，您仍然可以指定不同的 AWS KMS 金鑰來加密音訊物品。如需詳細資訊，請參閱《Amazon S3 Simple Stor age Ser vice 開發人員指南》中的，請參閱《Amazon Simple Storage Service

如果您選擇加密音訊日誌，Amazon Lex 需要AWS KMS許可。您必須將其他政策連接至用來對話日誌的 IAM 角色。如果您在 S3 儲存貯體上使用預設加密，則您的政策必須授與 AWS KMS 金鑰的存取權，而此金鑰是針對該儲存貯體設定的。如果您在音訊日誌設定中指定 AWS KMS 金鑰，則必須授與該金鑰的存取權。

如果您未建立對話日誌的角色，請參閱交談記錄的 IAM 政策。

建立使用AWS KMS金鑰加密音訊記錄的 IAM 政策

1. 在稱為 **LexConversationLogsKMSPolicy.json** 的目前目錄中建立一個文件、將下列政策新增至其中，然後儲存它。

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

2. 在中AWS CLI，建立 IAM 政策，以授與使用AWS KMS金鑰來加密音訊記錄的權限。

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

3. 將此政策附加到您為對話日誌建立的角色。

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

檢視亞馬遜日誌中的文字 CloudWatch 日誌

Amazon Lex 會在亞馬遜日誌中儲存您對話的文字 CloudWatch 日誌。若要檢視記錄，您可以使用記 CloudWatch 錄主控台或 API。如需詳細資訊，請參閱 Amazon Logs 使用者指南中的使用篩選器模式和CloudWatch 日誌見解查詢語法搜尋 CloudWatch 日誌資料。

若要使用 Amazon Lex 主控台檢視日誌

1. 打開 Amazon Lex 控制台 https://console.aws.amazon.com/lex。
2. 從清單中選擇一個機器人。
3. 選擇 Settings (設定) 標籤，然後從左側功能表中選擇 Conversation logs (對話日誌)。
4. 選擇 [文字記錄] 下方的連結，以在 CloudWatch 主控台中檢視別名的記錄檔。

您也可以使用主 CloudWatch 控台或 API 來檢視您的記錄項目。若要尋找日誌項目，請導覽至您針對別名設定的日誌群組。您可以在 Amazon Lex 主控台或使用GetBotAlias操作找到日誌的日誌串流前置詞。

使用者表達用語的日誌項目位於多個日誌串流中。對話中的表達用語在其中一個日誌串流中具有指定前綴的項目。日誌串流中的項目包含下列資訊。

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

```
    }
}
```

日誌項目的內容取決於交易的結果以及機器人和請求的組態。

- 若 `missedUtterance` 欄位是 `true`，則 `intent`、`slots` 和 `slotToElicit` 不會顯示在輸入中。
- 如果音訊日誌已停用或 `inputDialogMode` 欄位是 `Text`，則 `s3PathForAudio` 欄位不會出現。
- 只有在您為機器人定義回應卡片時，才會顯示 `responseCard` 欄位。
- 只有在請求中指定了請求屬性時，才會顯示 `requestAttributes` 對映。
- 只有當提出搜索 Amazon Kendra 索引的請求時，該`kendraResponseAMAZON.KendraSearchIntent`字段才存在。
- 當在機器人的 Lambda 函數中指定替代意圖時，此`developerOverride`欄位為真。
- 只有在請求中指定了工作階段屬性時，才會顯示 `sessionAttributes` 映射。
- 只有在您設定機器人傳回情緒值時，才會顯示 `sentimentResponse` 映射。

> **Note**
>
> 即使 `messageVersion` 中沒有對應的變更，輸入格式也可能變更。如果出現新欄位，您的程式碼不應擲出錯誤。

您必須設定角色和政策，才能讓 Amazon Lex 寫入 CloudWatch 日誌。如需詳細資訊，請參閱 交談記錄的 IAM 政策。

訪問 Amazon S3 中的音頻日誌

Amazon Lex 會將您對話的音訊日誌存放於 S3 儲存貯體中。

使用主控台存取音訊日誌

1. 打開 Amazon Lex 控制台 https://console.aws.amazon.com/lex。
2. 從清單中選擇一個機器人。
3. 選擇 Settings (設定) 標籤，然後從左側功能表中選擇 Conversation logs (對話日誌)。
4. 選擇音訊日誌下的連結，以在 Amazon S3 主控台中存取別名的日誌。

您也可以使用 Amazon S3 主控台或 API 存取得音訊日誌。您可以在 Amazon Lex 主控台或`GetBotAlias`作業回應的`resourcePrefix`欄位中看到音訊檔案的 S3 物件 key prefix。

以測 CloudWatch 量結果監督交談記錄狀態

使用 Amazon CloudWatch 監控交談日誌的交付指標。您可以在指標上設定警示，以便在記錄時注意到應該發生的問題。

Amazon Lex 在`AWS/Lex`命名空間中為交談日誌提供四個指標：

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

如需詳細資訊，請參閱CloudWatch 交談記錄的度量。

成功指標顯示 Amazon Lex 已成功將您的音訊或文字日誌寫入目的地。

失敗指標顯示 Amazon Lex 無法將音訊或文字日誌傳遞到指定的目的地。通常，這是組態錯誤。當您的失敗指標高於零時，請檢查下列情況：

- 請確定 Amazon Lex 是 IAM 角色的受信任實體。
- 對於文字記錄，請確定記錄 CloudWatch 檔記錄群組存在。對於音訊記錄，確定 S3 儲存貯體存在。
- 確保 Amazon Lex 用於存取 CloudWatch 日誌日誌群組或 S3 儲存貯體的 IAM 角色具有日誌群組或儲存貯體的寫入權限。
- 確定 S3 儲存貯體與 Amazon Lex 機器人位於相同的區域，且屬於您的帳戶。
- 如果您使用 S3 加密AWS KMS金鑰，請確保沒有任何政策阻止 Amazon Lex 使用您的金鑰，並確保您提供的 IAM 角色具有必要的AWS KMS許可。如需詳細資訊，請參閱 交談記錄的 IAM 政策。

使用 Amazon Lex API 管理工作階段

使用者開始與您的機器人對話時，Amazon Lex 會建立會議。您的應用程式與 Amazon Lex 之間交換的資訊構成對話的工作階段狀態。您發出請求時，會以您指定的機器人名稱與使用者識別符組合識別此工作階段。如需使用者識別符的詳細資訊，請參閱 PostContent 或 PostText 操作中的 `userId` 欄位。

工作階段操作的回應包括唯一的工作階段識別符。此識別符用以識別使用者的特定工作階段。您可以在測試時使用此識別符，或協助進行機器人的疑難排解。

您可以修改在應用程式與機器人之間傳送的工作階段。例如，您可以修改包含工作階段自訂資訊的工作階段屬性，也可以設定解釋下一個表達用語的對話方塊內容，來變更對話流程。

有兩種方式可以更新工作階段狀態。第一種方法是使用 Lambda 函數與PostContent或者PostText操作，該操作將在對話的每一輪後調用。如需詳細資訊，請參閱 使用 Lambda 函數。另一種方式則是在您的應用程式中使用 Amazon Lex 執行時間 API，來變更工作階段狀態。

Amazon Lex 運行時間 API 提供可讓您管理與機器人對話的工作階段資訊。這些操作為 PutSession 操作、GetSession 操作及 DeleteSession 操作。您使用這些操作取得您使用者的機器人工作階段資訊，並對狀態進行細微控制。

當您想要取得工作階段目前的狀態時，請使用 GetSession 操作。此操作會傳回工作階段目前的狀態，包括您使用者的對話方塊狀態、已設定的任何工作階段狀態，以及使用者最近三次互動之意圖的槽值。

此 PutSession 操作可讓您直接運用目前的工作階段狀態。您可以設定機器人接下來要執行的對話方塊類型。如此一來，您可以控制與機器人的對話流程。設置對話框操作type欄位Delegate，讓 Amazon Lex 決定機器人的下一個動作。

您可以使用 PutSession 操作來建立包含機器人的新工作階段，並設定機器人開始時的意圖。您也可以使用 PutSession 操作，來將一種意圖變更成另一種意圖。建立工作階段或變更意圖時，您也可以設定工作階段狀態，例如槽值和工作階段屬性。完成新的意圖時，您可以選擇重新啟動先前的意圖。您可以使用GetSession操作從 Amazon Lex 取得先前意圖的對話方塊狀態，並使用該資訊設定意圖的對話方塊狀態。

來自 PutSession 操作的回應包含如 PostContent 操作的相同資訊。正如您會對 PostContent 操作的回應一樣，您可以使用此項資訊向使用者提示下一筆資訊。

使用 DeleteSession 操作移除現有工作階段，並重新啟動新的工作階段。例如，當您正在測試機器人時，您可以使用 DeleteSession 操作從機器人移除測試工作階段。

工作階段操作可搭配您的履行 Lambda 函數使用。例如，如果 Lambda 函數返回Failed作為配送狀態，您可以使用PutSession操作，將對話方塊動作類型設為close和fulfillmentState至ReadyForFulfillment以重試完成步驟。

以下是您可以使用工作階段操作進行的一些動作：

- 讓機器人開始對話，而非等候使用者。

- 在對話期間切換意圖。
- 回到先前的意圖。
- 在互動的過程中間開始或重新開始對話。
- 驗證槽值並讓機器人針對無效的值重新提示。

以下進一步說明上述各個動作。

切換意圖

您可以使用 PutSession 操作，來將一種意圖切換成另一種意圖。您也可以使用它切回上一個意圖。您可以使用 PutSession 操作來設定工作階段屬性或新意圖的槽值。

- 呼叫 PutSession 操作。將意圖名稱設為新意圖的名稱，然後將對話方塊動作設為 Delegate。您也可以設定新意圖所需的任何槽值或工作階段屬性。
- Amazon Lex 將在新意圖期間開始與使用者對話。

繼續先前的意圖

若要繼續先前的意圖，請使用 GetSession 操作來取得意圖的摘要，然後使用 PutSession 操作來將意圖設為其之前的對話方塊狀態。

- 呼叫 GetSession 操作。來自操作的回應包括使用者所互動之最後三次意圖的對話方塊狀態摘要。
- 使用意圖摘要的資訊呼叫 PutSession 操作。這將把在對話中同一處的先前意圖傳回給使用者。

在某些情況下，可能需要繼續您使用者與機器人的對話。例如，假設您已建立客服機器人。您的應用程式會判斷使用者是否需要與客服代表談話。與使用者交談後，客服代表可以連同收集到的資訊，將對話轉回機器人。

若要繼續工作階段，請使用與下列相似的步驟：

- 您的應用程式會判斷使用者是否需要與客服代表交談。
- 使用 GetSession 操作來取得意圖目前的對話方塊狀態。
- 客服代表與使用者交談，並解決問題。
- 使用 PutSession 操作來設定意圖的對話方塊狀態。這可能包括設定槽值、設定工作階段屬性或變更意圖。

- 機器人繼續與使用者對話。

您可以使用 PutSession 操作 checkpointLabel 參數來標示意圖，以便於稍後可以找到意圖。例如，詢問客戶相關資訊的機器人可能會進入 Waiting 意圖，同時客戶蒐集資訊。機器人會為目前意圖建立檢查點標籤，然後啟動 Waiting 單元。客戶返回時，機器人可以使用檢查點標籤尋找上一個意圖並切換回來。

意圖必須存在於 GetSession 操作傳回的 recentIntentSummaryView 結構中。如果您在 GetSession 操作請求中指定檢查點標籤，則它最多會傳回包含該檢查點標籤的三個意圖。

- 使用 GetSession 操作來取得工作階段目前的狀態。
- 使用 PutSession 操作來將檢查點標籤新增至上一個意圖。必要時，您可以使用此 PutSession 呼叫，切換至不同的意圖。
- 該是切換回標示的意圖時，請呼叫 GetSession 操作來傳回最近的意圖清單。您可以使用checkpointLabelFilter參數，以使 Amazon Lex 僅返回具有指定檢查點標籤的意圖。

開啟新的工作階段

如果您想要讓機器人開始與您的使用者對話，則可以使用 PutSession 操作。

- 建立無槽的歡迎意圖及提示使用者的結論訊息，以陳述意圖。例如，「您想要訂購什麼？ 您可以說「訂購飲料」或「訂購披薩」。
- 呼叫 PutSession 操作。將意圖名稱設為歡迎意圖的名稱，然後將對話方塊動作設為 Delegate。
- Amazon Lex 將以歡迎意圖的提示回應，以開始與您的使用者對話。

驗證槽值

您可以使用用戶端應用程式驗證對您機器人所做的回應。如果回應無效，您可以使用 PutSession 操作來從您的使用者取得新的回應。例如，假設您的訂花機器人只能賣鬱金香、玫瑰花及水仙花。如果使用者訂購康乃馨，您的應用程式可以執行下列動作：

- 檢查從 PostText 或 PostContent 回應傳回的槽值。
- 如果槽值無效，則呼叫 PutSession 操作。您的應用程式應清除槽值，請設定 slotToElicit 欄位，然後將 dialogAction.type 值設為 elicitSlot。或者，您可以設置message和messageFormat欄位，如果您想要使用來引出槽值，請執行此資訊。

機器人部署選項

目前，Amazon Lex 提供下列機器人部署選項：

- AWS 行動開發套件 — 您可以建置使用 AWS 行動開發套件與 Amazon Lex 通訊的行動應用程式。
- Facebook 的信使-你可以集成你的 Facebook 信使頁面與您的 Amazon Lex 機器人，以便最終用戶在 Facebook 上可以與機器人進行通信。在目前的實作中，這項整合僅支援文字輸入訊息。
- 鬆弛 — 您可以將 Amazon Lex 機器人與 Slack 簡訊應用程式整合。
- Twilio — 您可以將您的 Amazon Lex 機器人與 Twilio 簡單訊息服務 (SMS) 整合。

如需範例，請參閱部署 Amazon Lex 機器人。

內建意圖和槽類型

為了更輕鬆地建立機器人，Amazon Lex 允許您使用標準的內建對應方式和插槽類型。

主題

- 內建意圖
- 內建槽類型

內建意圖

對於常見操作，您可以使用標準的內置意圖庫。若要從內建意圖建立意圖，請在主控台中選擇內建意圖，並為其指定新名稱。新意圖具有基本意圖的組態，例如範例語彙。

您無法在目前的實作中執行以下操作：

- 從基本意圖中新增或移除範例語彙
- 設定內建意圖的槽

若要將內建意圖新增至機器人

1. 登錄到AWS Management Console並打開 Amazon Lex 控制台 https://console.aws.amazon.com/lex/.
2. 選擇要新增內建意圖的機器人。

3. 在導覽窗格中，選擇 Intents (意圖) 旁邊的加號 (+)。
4. 針對 Add intent (新增意圖)，選擇 Search existing intents (搜尋現有的意圖)。
5. 在 [搜尋意圖] 方塊中，輸入要新增至機器人的內建意圖名稱。
6. 針對「複製內建的色彩比對方式」，為意圖命名，然後選擇「新增」。
7. 視需要設定機器人的意圖。

主題

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

> **Note**
>
> 對於英文 (美國) (en-US) 地區設定，Amazon Lex 支援 Alexa 標準內建意圖的意圖。如需內建意圖清單，請參閱 Alexa Skills Kit 中的標準內建意圖。
> Amazon Lex 不支持以下意圖：
>
> - `AMAZON.YesIntent`
> - `AMAZON.NoIntent`
> - Alexa Skills Kit 中 內建意圖程式庫內的意圖

AMAZON.CancelIntent

回應指出使用者想要取消目前互動的字詞和片語。您的應用程式可以使用此意圖，在結束與使用者互動之前，移除插槽類型值和其他屬性。

常見的話語：

- 取消
- 沒關係
- 忘記它

AMAZON.FallbackIntent

當使用者對意圖的輸入不是機器人預期的內容時，您可以設定 Amazon Lex 叫用後援意圖。例如，如果使用者輸入的「我想訂購糖果」與`OrderFlowers`機器人中的意圖不符，Amazon Lex 會叫用後援意圖來處理回應。

透過將內建 `AMAZON.FallbackIntent` 意圖類型新增到機器人來新增備用意圖。您可以使用 PutBot 操作，或從主控台的內建意圖清單中選擇意圖以指定意圖。

呼叫備用意圖需要兩個步驟。在第一個步驟中，備用意圖係根據使用者的輸入進行比對。當備用意圖相符時，機器人的行為方式則取決於為提示設定的重試次數。例如，如果嘗試判斷意圖的最大次數是 2，則機器人會在呼叫備用意圖之前，傳回機器人的釐清提示兩次。

在下列情況下，Amazon Lex 符合後援意圖：

- 使用者對意圖的輸入不符合機器人預期的輸入
- 音訊輸入為雜訊，或文字輸入無法辨識為文字。
- 使用者的輸入不明確，Amazon Lex 無法判斷要叫用的意圖。

以下情況會叫用備用意圖：

- 在對話開始時、嘗試釐清的設定次數之後，機器人不會將使用者輸入識別為意圖。
- 在設定的嘗試次數之後，意圖不會將使用者輸入識別為槽值。
- 在設定的嘗試次數之後，意圖不會將使用者輸入視為確認提示的回應。

您可以搭配備用意圖使用下列項目：

- 履行 Lambda 數
- 一條結論陳述
- 一條後續追蹤提示

您無法將以下內容新增至備用意圖：

- 表達用語
- 槽
- 初始化和驗證 Lambda 函數
- 一條確認提示

如果您已為機器人設定取消陳述式和後援意圖，Amazon Lex 會使用後援意圖。如果您需要您的機器人具有取消語句，則可以將履行功能用於後備意圖，以提供與取消語句相同的行為。如需詳細資訊，請參閱 PutBot 操作的 `abortStatement` 參數。

使用釐清提示

如果您向機器人提供釐清提示，則該提示將用於向使用者請求有效的意圖。釐清提示會依您設定的次數進行重複。在此之後才會呼叫備用意圖。

如果您在建立機器人時未設定澄清提示，且使用者沒有以有效意圖開始對話，Amazon Lex 會立即呼叫您的後援意圖。

當您在沒有說明提示的情況下使用後援意圖時，Amazon Lex 在下列情況下不會呼叫後援：

- 在使用者回應後續提示，但不提供意圖時。例如，回應後續提示：「您今天還想要什麼嗎？」，使用者說「是」。Amazon Lex 會傳回 400 錯誤請求例外狀況，因為它沒有要傳送給使用者以取得意圖的澄清提示。
- 在使用 AWS Lambda 函數時，您會傳回 `ElicitIntent` 對話方塊類型。因為 Amazon Lex 沒有取得使用者意圖的澄清提示，所以會傳回 400 個錯誤請求例外狀況。
- 在使用 `PutSession` 操作時，您會傳送 `ElicitIntent` 對話方塊類型。因為 Amazon Lex 沒有取得使用者意圖的澄清提示，所以會傳回 400 個錯誤請求例外狀況。

使用具有後援意圖的 Lambda 函數

在呼叫備用意圖時，其回應取決於對 PutIntent 操作的 `fulfillmentActivity` 參數設定。機器人會執行下列其中一項操作：

- 將意圖資訊傳回給用戶端應用程式。
- 呼叫履行 Lambda 函數。它會使用為工作階段設定的工作階段變數來呼叫函數。

如需在呼叫備用意圖時設定回應的詳細資訊，請參閱 PutIntent 操作的 `fulfillmentActivity` 參數。

如果您在後援意圖中使用履行 Lambda 函數，則可以使用此函數呼叫其他意圖或與使用者執行某種形式的通訊，例如收集回呼號碼或與客戶服務代表開啟工作階段。

您可以在後援意圖 Lambda 函數中執行任何動作，您可以針對任何其他意圖在履行函數中執行這些動作。如需使用 AWS Lambda 建立履行函數的詳細資訊，請參閱 使用 Lambda 函數。

您可以在相同工作階段中多次叫用備用意圖。例如，假設您的 Lambda 函數使用對`ElicitIntent`話方塊動作來提示使用者不同的意圖。如果 Amazon Lex 在設定的嘗試次數後無法推斷使用者的意圖，則會再次叫用後援意圖。當使用者在設定的嘗試次數之後仍未回應有效的槽值，它也會叫用備用意圖。

您可以設定 Lambda 函數，以追蹤使用工作階段變數呼叫後援意圖的次數。如果呼叫 Lambda 函數的次數超過您在 Lambda 函數中設定的閾值的次數，則 Lambda 函數可能會採取不同的動作。如需工作階段變數的詳細資訊，請參閱 設定工作階段屬性。

AMAZON.HelpIntent

回應表示使用者在與機器人互動時需要協助的字詞或片語。呼叫此意圖時，您可以設定 Lambda 函數或應用程式，以提供機器人功能的相關資訊、詢問有關說明領域的後續問題，或將互動交給人工代理。

常見的話語：

- 說明
- 幫助我
- 你能幫我嗎

AMAZON.KendraSearchIntent

若要搜尋已使用 Amazon Kendra 編製索引的文件，請使用`AMAZON.KendraSearchIntent`意圖。當 Amazon Lex 無法判斷與使用者交談中的下一個動作時，會觸發搜尋意圖。

`AMAZON.KendraSearchIntent`僅在英文 (美國) (en-US) 地區以及美國東部 (維吉尼亞北部)、美國西部 (奧勒岡) 和歐洲 (愛爾蘭) 區域提供。

Amazon Kendra 是一種 machine-learning-based 搜索服務，用於索引自然語言文檔，如 PDF 文檔或 Microsoft Word 文件。它可以搜尋已編製索引的文件，並傳回下列類型的問題回覆：

- 解答
- 可能回答問題的常見問題項目
- 與問題相關的文件

如需使用 AMAZON.KendraSearchIntent 的範例，請參閱範例：為 Amazon Kendra 索引建立常見問題集機器人。

如果您為機器人設定AMAZON.KendraSearchIntent意圖，Amazon Lex 會在無法判斷插槽或意圖的使用者說話時呼叫意圖。例如，如果您的機器人引發了一種稱為「比薩餅餡料」的插槽類型的響應，並且用戶說「什麼是比薩餅？ ，」Amazon Lex 打電話AMAZON.KendraSearchIntent來處理這個問題。如果沒有來自 Amazon Kendra 的回應，交談會依照機器人中的設定繼續進行。

當您在同一個機器人AMAZON.FallbackIntent中同時使用AMAZON.KendraSearchIntent和時，Amazon Lex 使用意圖如下：

1. Amazon Lex 調用AMAZON.KendraSearchIntent. 意圖呼叫 Amazon Kendra Query 操作。
2. 如果 Amazon Kendra 傳回回應，Amazon Lex 會向使用者顯示結果。
3. 如果沒有來自 Amazon Kendra 的回應，Amazon Lex 重新提示用戶。下一個動作取決於來自使用者的回應。
   - 如果使用者的回應包含 Amazon Lex 可辨識的話語 (例如填入插槽值或確認意圖)，則與使用者的對話會按照機器人的設定繼續進行。
   - 如果使用者的回應不包含 Amazon Lex 所辨識的話語，Amazon Lex 會再次呼叫該作業。Query
4. 如果設定的重試次數後沒有回應，Amazon Lex 會呼叫AMAZON.FallbackIntent並結束與使用者的對話。

有三種方法可以使用AMAZON.KendraSearchIntent向 Amazon Kendra 發出請求：

- 讓搜索意圖為您提出請求。Amazon Lex 調用 Amazon Kendra，並將用戶的話語作為搜索字符串。建立意圖時，您可以定義查詢篩選字串，以限制 Amazon Kendra 傳回的回應數量。Amazon Lex 在查詢請求中使用篩選器。
- 使用對話方塊 Lambda 函數，將其他查詢參數新增至請求，以縮小搜尋結果範圍。您可以將包含 Amazon Kendra 查詢參數的kendraQueryFilterString欄位新增至delegate對話方塊動作。當您使用 Lambda 函數將查詢參數新增至請求時，它們的優先順序會高於您建立意圖時定義的查詢篩選器。
- 使用對話方塊 Lambda 函數建立新的查詢。您可以建立 Amazon Lex 傳送的完整亞馬遜肯德拉查詢請求。您可以在 delegate 對話方塊動作的 kendraQueryRequestPayload 欄位中指定查詢。kendraQueryRequestPayload 欄位的優先順序高於 kendraQueryFilterString 欄位。

若要在建立機器人時指定queryFilterString參數，或在對話方塊 Lambda 函數中呼叫delegate動作時指定kendraQueryFilterString欄位，請指定用作 Amazon Kendra

查詢屬性篩選器的字串。如果字串不是有效的屬性篩選條件，則您會在執行時間取得 InvalidBotConfigException 例外狀況。如需有關屬性篩選器的詳細資訊，請參閱 Amazon Kendra 開發人員指南中的使用文件屬性篩選查詢。

若要控制 Amazon Lex 傳送至 Amazon Kendra 的查詢，您可以在對話方塊 Lambda 函數的kendraQueryRequestPayload欄位中指定查詢。如果查詢無效，Amazon Lex 會傳回InvalidLambdaResponseException例外狀況。如需詳細資訊，請參閱 Amazon Kendra 開發人員指南中的查詢操作。

如需使用 AMAZON.KendraSearchIntent 的範例，請參閱範例：為 Amazon Kendra 索引建立常見問題集機器人。

Amazon Kendra 搜索的 IAM 政策

若要使用AMAZON.KendraSearchIntent意圖，您必須使用提供 AWS Identity and Access Management (IAM) 政策的角色，讓 Amazon Lex 承擔具有呼叫 Amazon Kendra Query 意圖之權限的執行時期角色。您使用的 IAM 設定取決於您是AMAZON.KendraSearchIntent使用 Amazon Lex 主控台建立，還是使用 AWS 開發套件或 AWS Command Line Interface (AWS CLI) 來建立。使用主控台時，您可以選擇新增將 Amazon Kendra 呼叫給 Amazon Lex 服務連結角色的權限，或使用專門用於呼叫 Amazon Kendra Query 操作的角色。當您使用 AWS CLI 或開發套件建立意圖時，您必須使用專門用於呼叫 Query 操作的角色。

連接許可

您可以使用主控台連接許可，將 Amazon Kendra Query 操作存取到預設的 Amazon Lex 服務連結角色。將許可附加到服務連結角色時，您不需要建立和管理專門用於連線至 Amazon Kendra 索引的執行時期角色。

您用來存取 Amazon Lex 主控台的使用者、角色或群組必須具有管理角色政策的許可。將下列 IAM 政策附加到主控台存取角色。當您授與這些許可時，角色具有變更現有的服務連結角色政策的許可。

```
{
    "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": "*"
        }
    ]
}
```

指定角色

您可以使用主控台AWS CLI、或 API 來指定呼叫 Amazon Kendra Query 作業時要使用的執行階段角色。

您用來指定執行階段角色的使用者、角色或群組必須具有iam:PassRole權限。下列政策會定義許可。您可以使用 iam:AssociatedResourceArn 和 iam:PassedToService 條件內容鍵來進一步限制許可的範圍。如需詳細資訊，請參閱AWS Identity and Access Management使用指南中的 IAM 和 AWS STS條件內容金鑰。

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

Amazon Lex 用來呼叫 Amazon Kendra 所需的執行階段角色必須具有kendra:Query許可。當您使用現有的 IAM 角色獲得呼叫 Amazon Kendra Query 操作的許可時，該角色必須附加以下政策。

您可以使用 IAM 主控台、IAM API 或建AWS CLI立政策並將其附加到角色。這些指示會使用 AWS CLI 來建立角色和政策。

> **Note**
>
> 下列程式碼是針對 Linux 和 MacOS 格式化的。若為 Windows，請將接續字元 (\) 取代為插入符號 (^)。

將查詢操作許可新增至角色

1. 在目前目錄中建立一個稱為 **KendraQueryPolicy.json** 的文件、將下列程式碼新增至其中，然後儲存它

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

2. 在中AWS CLI，執行下列命令以建立用於執行 Amazon Kendra Query 作業的 IAM 政策。

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

3. 將政策附加到您用來呼叫Query作業的 IAM 角色。

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

您可以選擇更新 Amazon Lex 服務連結角色，或使用為機器人建立時建立AMAZON.KendraSearchIntent的角色。下列程序顯示如何選擇要使用的 IAM 角色。

若要指定 AMAZON 的執行階段角色。 KendraSearchIntent

1. 登錄到AWS Management Console並打開 Amazon Lex 控制台 https://console.aws.amazon.com/lex/.
2. 選擇您要新增 `AMAZON.KendraSearchIntent` 的機器人。
3. 選擇 Intents (意圖) 旁邊的加號 (+)。
4. 在 Add intent (新增意圖) 中，選擇 Search existing intents (搜尋現有的意圖)。
5. 在 Search intents (搜尋意圖) 中，輸入 **`AMAZON.KendraSearchIntent`** 然後選擇 Add (新增)。
6. 在 Copy built-in intent (複製內建意圖) 中，輸入意圖的名稱，例如 **`KendraSearchIntent`**，然後選擇 Add (新增)。
7. 開啟 Amazon Kendra query (Amazon Kendra 查詢) 部分。
8. 在 IAM role (IAM 角色) 下，選擇以下其中一個選項：
   - 若要更新 Amazon Lex 服務連結角色，讓您的機器人能夠查詢 Amazon Kendra 索引，請選擇新增 Amazon Kendra 許可。
   - 若要使用具有呼叫 Amazon Kendra `Query` 作業之權限的角色，請選擇「使用現有角色」。

使用請求和工作階段屬性作為篩選條件

若要篩選 Amazon Kendra 對與目前對話相關項目的回應，請在建立機器人時新增`queryFilterString`參數，使用工作階段和請求屬性做為篩選器。您可以在建立意圖時指定屬性的預留位置，然後 Amazon Lex V2 會在呼叫 Amazon Kendra 之前替換值。如需請求屬性的詳細資訊，請參閱設定請求屬性。如需工作階段屬性的詳細資訊，請參閱 設定工作階段屬性。

以下是使用字串篩選 Amazon Kendra 查詢的`queryFilterString`參數範例。

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

以下是使用名`"SourceURI"`為篩選 Amazon Kendra 查詢的工作階段屬性的`queryFilterString`參數範例。

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

以下是使用名為篩選 Amazon Kendra 查詢的請求`"DepartmentName"`屬性的`queryFilterString`參數範例。

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

這些AMAZON.KendraSearchInteng篩選器使用與 Amazon Kendra 搜尋篩選器相同的格式。如需詳細資訊，請參閱 Amazon Kendra 開發人員指南中的使用文件屬性篩選搜尋結果。

搭配使用的查詢篩選字串AMAZON.KendraSearchIntent必須為每個篩選器的第一個字母使用小寫字母。例如，以下是有效的查詢篩選器AMAZON.KendraSearchIntent。

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

使用搜尋回應

Amazon Kendra 會在意圖conclusion陳述式中傳回搜尋的回應。除非履行 Lambda 函數產生結論訊息，否則意圖必須有conclusion陳述式。

Amazon Kendra 有四種類型的回應。

- x-amz-lex:kendra-search-response-question_answer-question-<N>— 來自與搜索匹配的常見問題解答的問題。
- x-amz-lex:kendra-search-response-question_answer-answer-<N>— 來自與搜索匹配的常見問題解答的答案。

- x-amz-lex:kendra-search-response-document-<N>— 摘錄自索引中的文件，該文件與語音文字相關。
- x-amz-lex:kendra-search-response-document-link-<N>— 索引中與語音文字相關的文件 URL。
- x-amz-lex:kendra-search-response-answer-<N>— 索引中文件摘錄，可回答問題。

回應會以 request 屬性傳回。每個屬性最多可有五個回應，編號為 1 到 5。如需有關回應的詳細資訊，請參閱 Amazon Kendra 開發人員指南中的回應類型。

conclusion 陳述式必須有一或多個訊息群組。每個訊息群組都包含一或多個訊息。每個訊息都可以包含一或多個預留位置變數，這些變數會在 Amazon Kendra 回應中由請求屬性取代。訊息群組中必須至少有一個訊息，而訊息中的所有變數都會由執行時間回應中的請求屬性值取代，或是群組中必須具有無預留位置變數的訊息。請求屬性會以雙括號 ("((" "))") 括起來。下列訊息群組訊息符合來自 Amazon Kendra 的任何回應：

- 「我為您找到了一個常見問題解答問題x-amz-lex：（（（：kendra-search-response-question_ 答案-問題 -1））），答案是（（x-amz-lex：答案 -1））」kendra-search-response-question
- 「我從一個有用的文檔中找到了摘錄 :((x-amz-lex：kendra-search-response-document-1））」
- 「我認為你的問題的答案是（（x-amz-lex：kendra-search-response-answer-1））」

使用 Lambda 函數來管理請求和回應

AMAZON.KendraSearchIntent意圖可以使用您的對話方塊程式碼掛接和履行代碼勾點來管理傳送給 Amazon Kendra 的請求和回應。當您想要修改傳送至 Amazon Kendra 的查詢時，請使用對話方塊程式碼掛接 Lambda 函數，而當您要修改回應時，履行程式碼會掛接 Lambda 函數。

使用對話方塊程式碼掛勾建立查詢

您可以使用對話方塊程式碼掛接來建立要傳送至 Amazon Kendra 的查詢。使用對話方塊程式碼掛勾是選用的。如果您未指定對話方塊程式碼勾點，Amazon Lex 會根據使用者的話語建構查詢，並使用您在設定queryFilterString意圖時提供的查詢 (如果您提供的話)。

您可以使用對話方塊程式碼勾點回應中的兩個欄位來修改 Amazon Kendra 的請求：

- kendraQueryFilterString— 使用此字串可指定 Amazon Kendra 請求的屬性篩選器。您可以使用索引中定義的任何索引欄位來篩選查詢。如需篩選字串的結構，請參閱 Amazon Kendra 開發人員指南中的使用文件屬性篩選查詢。如果指定的篩選字串無效，您會取得

InvalidLambdaResponseException 例外狀況。kendraQueryFilterString 字串會覆寫為意圖所設定的 queryFilterString 中指定的任何查詢字串。

- kendraQueryRequestPayload— 使用此字串可指定 Amazon Kendra 查詢。您的查詢可以使用 Amazon Kendra 的任何功能。如果您沒有指定有效的查詢，您會取得 InvalidLambdaResponseException 例外狀況。如需詳細資訊，請參閱 Amazon Kendra 開發人員指南中的查詢。

建立篩選器或查詢字串之後，您可以將回應的dialogAction欄位設定為的 Amazon Lex 傳送回應delegate。Amazon Lex 會將查詢傳送至 Amazon Kendra，然後將查詢回應傳回至履行代碼勾點。

針對回應使用履行程式碼掛勾

Amazon Lex 將查詢傳送至 Amazon Kendra 之後，查詢回應就會傳回至AMAZON.KendraSearchIntent履行 Lambda 函數。代碼鉤子的輸入事件包含來自 Amazon Kendra 的完整響應。查詢資料的結構與 Amazon Kendra Query 作業傳回的資料結構相同。如需詳細資訊，請參閱 Amazon Kendra 開發人員指南中的查詢回應語法。

履行程式碼掛勾是選用的。如果不存在，或程式碼掛鉤未在回應中傳回訊息，Amazon Lex 會使用該conclusion陳述式進行回應。

範例：為 Amazon Kendra 索引建立常見問題集機器人

此範例會建立 Amazon Lex 機器人，該機器人使用 Amazon Kendra 索引來提供使用者問題的答案。常見問題機器人會管理使用者的對話方塊。它使用 AMAZON.KendraSearchIntent 意圖來查詢索引，並向使用者呈現回應。若要建立機器人，您需：

1. 建立機器人，讓您的客戶與其互動以從機器人取得答案。
2. 建立自訂意圖。您的機器人需要至少一個意圖，此意圖具有至少一個表達用語。此意圖讓您的機器人可以建置，但不會用於其他用途。
3. 將KendraSearchIntent意圖新增至您的機器人，並將其設定為與 Amazon Kendra 索引搭配使用。
4. 透過詢問由 Amazon Kendra 索引中儲存的文件所回答的問題來測試機器人。

您必須先建立 Amazon Kendra 索引，才能使用此範例。如需詳細資訊，請參閱 Amazon Kendra 開發人員指南中的 S3 儲存貯體入門 (主控台)。

建立常見問題機器人

1. 登錄到AWS Management Console並打開 Amazon Lex 控制台 https://console.aws.amazon.com/lex/.
2. 在導覽窗格中，選擇 Bots (機器人)。
3. 選擇 建立 。
4. 選擇 Custom bot (自訂機器人)。設定機器人，如下所示：
   - 機器人名稱 — 為機器人指定其用途的名稱，例如**KendraTestBot**。
   - 輸出語音 — 選擇無。
   - 工作階段逾時 — 輸入**5**。
   - 情緒分析 — 選擇 [否]。
   - 杯 — 選擇沒有.
   - 使用者話語儲存 — 選擇 [不儲存]。
5. 選擇 建立 。

若要成功建置機器人，您必須至少建立一個意圖，此意圖具有至少一個範例表達用語。建置 Amazon Lex 機器人需要此目的，但不適用於常見問答集回應。意圖的表達用語不得套用至您的客戶所提出的任何問題。

建立必要的意圖

1. 在 Getting started with your bot (開始使用您的機器人) 頁面上，選擇 Create intent (建立意圖)。
2. 針對 Add intent (新增意圖)，選擇 Create intent (建立意圖)。
3. 在 Create intent (建立意圖) 對話方塊中，指定意圖名稱，例如 **RequiredIntent**。
4. 在 Sample utterances (範例表達用語) 中，輸入表達用語，例如 **Required utterance**。
5. 選擇 Save intent (儲存意圖)。

現在，建立搜尋 Amazon Kendra 索引及其應傳回的回應訊息的意圖。

要創建一個亞馬遜。 KendraSearchIntent 意圖和響應消息

1. 在導覽窗格中，選擇 Intents (意圖) 旁邊的加號 (+)。
2. 針對 Add intent (新增意圖)，選擇 Search existing intents (搜尋現有的意圖)。
3. 在「搜尋方式」方塊中輸入**AMAZON.KendraSearchIntent**，然後從清單中選擇。

4. 對於 Copy built-in intent (複製內建意圖)，提供意圖的名稱，例如 **KendraSearchIntent**，然後選擇 Add (新增)。
5. 在意圖編輯器中，選擇 Amazon Kendra query (Amazon Kendra 查詢) 以開啟查詢選項。
6. 從 Amazon Kendra index (Amazon Kendra 索引) 功能表中，選擇您想要搜尋意圖的索引。
7. 在 Response (回應) 區段中，新增下列三則訊息：

```
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. 選擇 Save intent (儲存意圖)，然後選擇 Build (建置) 以建置機器人。

最後，使用主控台測試視窗來測試您的機器人的回應。您的問題應該位於索引支援的網域中。

測試您的常見問題機器人

1. 在主控台測試視窗中，輸入索引的問題。
2. 驗證測試視窗回應部分中的答案。
3. 若要重設其他問題的測試視窗，請選擇 Clear chat history (清除聊天歷史記錄)。

AMAZON.PauseIntent

回應可讓使用者暫停與機器人互動的字詞和片語，以便他們稍後可以返回。您的 Lambda 函數或應用程式需要將意圖資料儲存在工作階段變數中，或者當您繼續目前的意圖時，您需要使用GetSession作業來擷取意圖資料。

常見的話語：

- 暫停
- 暫停

AMAZON.RepeatIntent

回應可讓使用者重複上一封郵件的字詞和片語。您的應用程式需要使用 Lambda 函數，將先前的意圖資訊儲存在工作階段變數中，或者您需要使用該GetSession作業來取得先前的意圖資訊。

常見的話語：

- 重複
- 再說一遍
- 重複這一點

AMAZON.ResumeIntent

響應單詞和短語，使用戶能夠恢復先前暫停的意圖。您 Lambda 函數或應用程式必須管理恢復之前的意圖所需的資訊。

常見的話語：

- 恢復
- 繼續
- 繼續, 去

AMAZON.StartOverIntent

回應使用者停止處理目前意圖並從頭開始重新開始的字詞和片語。您可以使用 Lambda 函數或`PutSession`作業再次引出第一個插槽值。

常見的話語：

- 重新開始
- 重新開始
- 重新開始

AMAZON.StopIntent

回應表示使用者想要停止處理目前意圖並結束與機器人互動的字詞和片語。您的 Lambda 函數或應用程式應該清除任何現有屬性和插槽類型值，然後結束互動。

常見的話語：

- stop
- off
- 閉嘴

內建槽類型

Amazon Lex 支援內建插槽類型，可定義如何辨識和處理插槽中的資料。您可以在意圖中建立這些槽類型。如此您就無須為常用的槽資料 (例如日期、時間和位置) 建立列舉值。內建槽類型並沒有版本。

槽類型	簡短描述	支援的地區設定	
亞馬遜. 機場	識別代表機場的單詞。	所有語言環境	
亞馬遜。 AlphaNumeric	辨識由字母和數字組成的字詞。	除了韓文 (Ko-Kr) 以外的所有語言環境	
亞馬遜城	識別代表城市的單詞。	所有語言環境	
亞馬遜國家	識別代表一個國家的單詞。	所有語言環境	
亞馬遜。日期	識別代表日期的單詞並將其轉換為標準格式。	所有語言環境	
亞馬遜持續時間	識別代表持續時間的單詞並將其轉換為標準格式。	所有語言環境	
亞馬遜。 EmailAddress	識別代表電子郵件地址的單詞，並將其轉換為標準電子郵件地址。	所有語言環境	

槽類型	簡短描述	支援的地區設定
亞馬遜。 FirstName	識別代表名字的單詞。	所有語言環境
亞馬遜。 LastName	識別代表姓氏的單詞。	所有語言環境
AMAZON.NUMBER	識別數字單詞並將其轉換為數字。	所有語言環境
AMAZON.Percentage	辨識代表百分比的字詞，並將其轉換為數字和百分比符號 (%)。	所有語言環境
亞馬遜。 PhoneNumber	識別代表電話號碼的單詞並將其轉換為數字字符串。	所有語言環境
亞馬遜。 SpeedUnit	識別代表速度單位的單詞並將其轉換為標準縮寫。	英文 (美國) (en-US)
亚马逊州	識別代表狀態的單詞。	所有語言環境
亞馬遜。 StreetName	識別代表街道名稱的單詞。	除了英文 (美國) 以外的所有語言環境 (zh-TW)
AMAZON.TIME	識別指示時間的單詞並將其轉換為時間格式。	所有語言環境
亞馬遜。 WeightUnit	識別代表重量單位的單詞並將其轉換為標準縮寫	英文 (美國) (en-US)

> **Note**
>
> 對於英文 (美國) (en-US) 地區設定，Amazon Lex 支援 Alexa 技能套件中的插槽類型。如需可用的內建槽類型清單，請參閱 Alexa Skills Kit 文件中的槽類型參考。
>
> - Amazon Lex 不支援AMAZON.LITERAL或AMAZON.SearchQuery內建插槽類型。

亞馬遜. 機場

提供機場清單。範例包括：

- 约翰·F· 甘迺迪國際機場
- 墨爾本機場

亞馬遜。 AlphaNumeric

辨識由字母和數字組成的字串，例如 **APQ123**。

此插槽類型在韓文 (K-KR) 語言環境中不可用。

您可以為包含下列項目的字串使用 AMAZON.AlphaNumeric 槽類型：

- 字母字元，例如 **ABC**
- 數值字元，例如 **123**
- 英數字元的組合，例如 **ABC123**

您可將規則表達式新增至 AMAZON.AlphaNumeric 槽類型，以驗證為該槽輸入的值。例如，您可以使用規則表達式來驗證：

- 英國或加拿大郵遞區號
- 駕照號碼
- 車輛識別碼

使用標準規則運算式。Amazon Lex 在規則運算式中支援下列字元：

- A-Z、a-z

- 0-9

Amazon Lex 還支持正則表達式中的 Unicode 字符。格式為 \u*Unicode*。使用四位數代表 Unicode 字元。例如，[\u0041-\u005A] 等同於 [A-Z]。

不支援下列規則運算式：

- 無限的重複項：*、+ 或 {x,}，沒有上限。
- 萬用字元 (.)

規則運算式的最大長度為 300 個字元。存儲在 AMAZON 中的字符串的最大長度。 AlphaNumeric 使用規則運算式的槽類型為 30 個字元。

以下是一些規則表達式的範例。

- 英數字串，例如 **APQ123** 或 **APQ1**：[A-Z]{3}[0-9]{1,3} 或限制更多的 [A-DP-T]{3} [1-5]{1,3}
- 「美國郵政服務國際優先郵件」格式，例如 **CP123456789US**：CP[0-9]{9}US
- 銀行匯款路線號碼，例如 **123456789**：[0-9]{9}

若要為槽類型設定規則表達式，請使用主控台或 PutSlotType 操作。當您儲存槽類型時，會驗證規則表達式。如果運算式無效，Amazon Lex 會傳回錯誤訊息。

當您在插槽類型中使用規則運算式時，Amazon Lex 會根據規則運算式檢查對該類型插槽的輸入。如果輸入與表達式相符，則會針對該槽接受值。如果輸入不相符，Amazon Lex 會提示使用者重複輸入。

亞馬遜城

提供本地和世界城市的列表。插槽類型可識別城市名稱的常見變化。Amazon Lex 不會從變體轉換為正式名稱。

範例：

- 紐約
- 雷克雅
- 東京
- 凡爾賽宮

亞馬遜國家

世界各地國家的名稱。範例：

- 澳洲
- 德國
- 日本
- 美國
- 烏拉圭

亞馬遜。日期

將代表日期的單字轉換為日期格式。

日期以 ISO-8601 日期格式提供給您的意圖。您的意圖在插槽中收到的日期可能會根據用戶所說的特定短語而有所不同。

- 映射到特定日期的語音，例如「今天」，「現在」或「11 月 25 日」，轉換為完整日期：。2020-11-25這預設為當前日期或之後的日期。
- 對應至特定週的話語 (例如「本週」或「下週」) 會轉換為一週的第一天。在 ISO-8601 格式中，一周從星期一開始，星期日結束。例如，如果今天是 2020-11-25，則「下週」將轉換為. 2020-11-30
- 對應至月份而非特定日期的語音 (例如「下個月」) 會轉換為月份的最後一天。例如，如果今天是 2020-11-25，則「下個月」將轉換為. 2020-12-31
- 對應至年份而非特定月份或日期的話語 (例如「明年」) 會轉換為次年的最後一天。例如，如果今天是 2020-11-25，則「明年」將轉換為. 2021-12-31

亞馬遜持續時間

將指示持續時間的單字轉換為數字持續時間。

持續時間會根據 ISO-8601 持續時間格式解析為格式PnYnMnWnDTnHnMnS。P表示這是一個持續時間，n是一個數值，後面的大寫字母n是特定的日期或時間元素。例如，P3D意味著 3 天。A 用T於表示剩餘的值表示時間元素，而不是日期元素。

範例：

- 「十分鐘」：PT10M

- 「五個小時」：PT5H
- 「三天」：P3D
- 「四十五秒」：PT45S
- 「八個星期」：P8W
- 「七年」：P7Y
- 「五小時十分鐘」：PT5H10M
- 「二年三小時十分鐘」：P2YT3H10M

亞馬遜。 EmailAddress

識別代表以 username@domain 形式提供之電子郵件地址的字詞。地址在使用者名稱中可以包含下列特殊字元：底線 (_)、連字號 (-)、句號 (.) 和加號 (+)。

亞馬遜。 FirstName

常用的名字。此插槽類型可識別正式名稱和非正式暱稱。發送到您的意圖的名稱是用戶發送的值。Amazon Lex 不會從暱稱轉換為正式名稱。

對於聽起來相似但拼寫不同的名字，Amazon Lex 會將您的意圖傳送為單一通用表單。

在英文 (美國) (EN-US) 語言環境中，請使用插槽名稱亞馬遜 .US_FIRST_NAME。

範例：

- 艾米莉
- John
- 蘇菲

亞馬遜。 LastName

常用的姓氏。對於聽起來相似且拼寫不同的名稱，Amazon Lex 會將您的意圖傳送為單一通用表單。

在英文 (美國) (EN-US) 地區設定中，請使用插槽名稱亞馬遜 .us_Last_Name。

範例：

- 布羅斯基

- 大傻瓜
- 埃弗斯
- 帕雷斯
- 邊痕

AMAZON.NUMBER

將表示數字的文字或數字轉換為數字，包括十進位數字。下表顯示 AMAZON.NUMBER 槽類型如何擷取數字字詞。

Input	回應
一百二十三點四五	123.45
一百二十三點四五	123.45
點四二	0.42
點四十二	0.42
232.998	232.998
50	50

AMAZON.Percentage

將代表百分比的字詞和符號轉換成包含百分比符號 (%) 的數值。

如果使用者輸入的數字沒有百分比符號或「百分比」一字，槽值會設定為數字。下表顯示 AMAZON.Percentage 槽類型如何擷取百分比。

Input	回應
50 百分比	50%
0.4% 百分比	0.4%
23.5%	23.5%

Input	回應
二十五, 百分之	25%

亞馬遜。 PhoneNumber

將代表電話號碼的數字或字詞轉換成不含標點符號的字串格式，如下所示。

類型	描述	Input	結果
含前置加號 (+) 的國際號碼	含前置加號的 11 位數號碼。	+61 7 4445 1061 +1 (509) 555-1212	+61744431061 +15095551212
不含前置加號 (+) 的國際號碼	不含前置加號的 11 位數號碼	1 (509) 555-1212 61 7 4445 1061	15095551212 61744451061
國內號碼	不含國際碼的 10 位數字	(03) 5115 4444 (509) 555-1212	0351154444 5095551212
市內號碼	不含國際碼或區碼的 7 位數電話號碼	555-1212	5551212

亞馬遜。 SpeedUnit

將代表速度單位的字詞轉換成相對應的縮寫。例如，「每小時英里數」轉換成 mph。

此插槽類型僅適用於英文 (美國) (en-US) 地區設定。

下例顯示 AMAZON.SpeedUnit 槽類型如何擷取速度單位。

速度單位	縮寫
每小時英里數、mph、MPH、m/h	mph
每小時公里數、kmph、KMPH、km/h	kmph

速度單位	縮寫
每秒公尺數、mps、MPS、m/s	mps
每小時海哩數、節	節

亚马逊州

國家內的地理和政治區域的名稱。

範例：

- 巴伐利亞
- 福島縣
- 西北太平洋
- 昆士蘭
- 威爾士

亞馬遜。 StreetName

典型街道地址內的街道名稱。這只包括街道名稱，而不是門牌號碼。

此插槽類型在英文 (美國) (en-US) 地區設定中無法使用。

範例：

- 堪培拉大道
- 前街
- 市場路

AMAZON.TIME

將代表時間的字詞轉換成時間值。包括不明確時期的分辨率。當使用者輸入不明確的時間時，Amazon Lex 會使用 Lambda 事件的`slotDetails`屬性，將不明確時間的解決方案傳遞給您的 Lambda 函數。例如，如果您的機器人提示使用者交付時間，使用者可以說「10 點鐘」來回應。但是這個時間並不明確，這可表示早上 10 點或下午 10 點。在這種情況下，`slots`地圖中的值是`null`，`slotDetails`實體包含時間的兩個可能的分辨率。Amazon Lex 將以下內容輸入到 Lambda 函數中：

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

當使用者以明確的時間回應時，Amazon Lex 會將時間傳送至 Lambda 事件`slots`屬性中的 Lambda 函數，且`slotDetails`屬性為空。例如，如果您的使用者以「晚上 10:00」回應交付時間的提示，Amazon Lex 會在 Lambda 函數中輸入下列內容：

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

如需從 Amazon Lex 傳送至 Lambda 函數之資料的詳細資訊，請參閱輸入事件格式。

亞馬遜。 WeightUnit

將代表重量單位的字詞轉換成相對應的縮寫。例如，「公斤」會轉換成 kg。

此插槽類型僅適用於英文 (美國) (en-US) 地區設定。

下例顯示 `AMAZON.WeightUnit` 槽類型如何擷取重量單位。

重量單位	縮寫
公斤、kg、KGS	kg
公克、gms、gm、GMS、g	g
毫克、mg、mgs	mg

重量單位	縮寫
磅、lbs、LBS	lbs
盎司、oz、OZ	oz
公噸、噸、t	t
千噸、kt	kt

自訂槽類型

對於每個意圖，您可以指定參數，指出意圖需要滿足使用者的請求的資訊。這些參數或槽，有一個類型。插槽類型是 Amazon Lex 用來訓練機器學習模型以辨識插槽值的值清單。例如，您可以定義一個稱為「Genres.」的槽類型，在該槽類型中的每個值都是一種流派的名稱，「喜劇」、「探險」、「紀錄片」，以此類推。您可以為槽類型值定義同義詞。例如，您可以為值「喜劇」定義同義詞「滑稽」和「幽默」。

您可以設定槽類型來限制槽值的解析。槽值會用作為列舉，並且只會在與其中一個槽值或同義詞相同時，才會將使用者輸入的值會解析為槽值。同義詞會解析為對應的槽值。例如，如果使用者輸入「滑稽」，它會解析為槽值「喜劇」。

您也可以設定槽類型來擴展該值。槽值會用作為訓練資料，並且只會在槽值和同義詞字類似時，才會將槽解析為使用者提供的值。這是預設行為。

Amazon Lex 會維護插槽的可能解析度清單。清單中的每個項目都提供解析度值，Amazon Lex 認為插槽的其他可能性。解析值是最符合槽值的項目。該清單最多可包含五個值。

當使用者輸入的值是同義詞時，解析值清單中的第一個項目是槽類型值。例如，如果使用者輸入「滑稽」，則 `slots` 欄位會包含「滑稽」而 `slotDetails` 欄位中的第一個項目是「喜劇」。您可以在使用 `valueSelectionStrategy` 操作建立或更新槽類型時設定 PutSlotType，如此一來槽值就會以解析清單中的第一個值填滿。

如果您使用的是 Lambda 函數，則函數的輸入事件會包含一個名為的解析度清單`slotDetails`。下列範例顯示 Lambda 函數輸入的插槽和插槽詳細資訊區段：

```
"slots": {
   "MovieGenre": "funny";
```

```
   },
   "slotDetails": {
      "Movie": {
         "resolutions": [
            "value": "comedy"
         ]
      }
   }
```

對於每個槽類型，您最多可以定義 10,000 個值和同義詞。每個機器人總共可有 50,000 個槽類型值和同義詞。例如，您有 5 個槽類型，每個有 5,000 個值和同義詞，或您有 10 個槽類型，每個有 2,500 個值和同義詞。如果您超過這些限制，您在呼叫 PutBot 操作時會取得 `LimitExceededException`。

槽混淆

使用 Amazon Lex，您可以混淆或隱藏槽內容，以便看不見內容。若要保護捕獲為槽值的敏感資料，您可以啟用槽混淆以遮罩對話日誌中的這些值。

當您選擇混淆槽值時，Amazon Lex 會將槽值取代為對話日誌中的槽名稱。對於稱為 `full_name` 的槽，槽值將被混淆，如下所示：

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

如果表達用語包含括號字元 ({})，則 Amazon Lex 會使用兩條反斜線 (\\\) 逸出括號字元。例如，文字 `{John Stiles}` 會被混淆，如下所示：

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

對話日誌中的槽值會被混淆。插槽值仍然可用於來自`PostContent`和`PostText`操作，而且槽值可供您的驗證和履行 Lambda 函數使用。如果您是在提示或回應中使用槽值，則這些槽值不會在對話日誌中混淆。

在對話的第一回合中，Amazon Lex 果辨識出表達用語中的槽和槽值，則它會混淆槽值。如果未辨識出任何槽值，Amazon Lex 不會混淆表達用語。

在第二回合和後續回合中，Amazon Lex 知道要引出的槽，以及槽值是否應該混淆。如果 Amazon Lex 辨識出槽值，則會混淆該值。如果 Amazon Lex 無法辨識出一個值，則會混淆整個表達用語。遺漏表達用語中的任何槽值都不會被混淆。

也 Amazon Lex 會混淆您存放在請求或工作階段屬性中的槽值。如果您是儲存應該當作屬性混淆的槽值，則必須加密或以其他方式混淆該值。

Amazon Lex 不會混淆音訊中的槽值。它的確會混淆音訊記錄中的槽值。

您不需要混淆機器人中的所有槽。您可以使用主控台或使用 Amazon Lex API 來選擇要混淆哪些槽。在主控台中，於槽設定中選擇 Slot obfuscation (槽混淆)。如果您是使用 API，則在呼叫 PutIntent 操作時，將槽的 `obfuscationSetting` 欄位設定為 `DEFAULT_OBFUSCATION`。

情緒分析

您可以使用情緒分析來判斷使用者表達用語中表達的情緒。使用情緒資訊，您可以管理對話流程或執行通話後分析。例如，如果使用者情緒為負面，您可以建立流程，將對話轉交給人類客服人員。

Amazon Lex 與 Amazon Comprehend 整合，以偵測使用者情緒。Amazon Comprehend 的回應會指出文字的整體情緒是正面、中性、負面還是混合的。回應包含使用者表達用語最有可能的情緒，以及每個情緒類別的分數。分數代表正確偵測到的情緒的可能性。

您可以使用主控台或使用 Amazon Lex API 為機器人啟用情緒分析。在 Amazon Lex 主控台上，選擇機器人的「設定」索引標籤，然後將「情緒分析」選項設定為「是」。如果您正在使用 API，請在 `detectSentiment` 欄位設定為 `true` 的情況下呼叫 PutBot 操作。

當啟用情緒分析時，來自 PostContent 和 PostText 操作的回應會傳回在機器人回應中稱為 `sentimentResponse` 的欄位以及其他中繼資料。`sentimentResponse` 欄位有兩個欄位，`SentimentLabel` 和 `SentimentScore`，其中包含情緒分析的結果。如果您使用的是 Lambda 函數，則`sentimentResponse`欄位會包含在傳送至函數的事件資料中。

以下是 `sentimentResponse` 欄位傳回 `PostText` 或 `PostContent` 回應一部分的範例。`SentimentScore` 欄位是字串，其中包含回應的分數。

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

```
        }",
    "SentimentLabel": "POSITIVE"
}
```

Amazon Lex 會代表您呼叫 Amazon Comprehend，以判斷機器人處理的每個話語中的情緒。啟用情緒分析，即表示您同意 Amazon Comprehend 的服務條款和協議。如需可用 Amazon Comprehend 的詳細資訊，請參閱《Amazon Comprehend》定價。

如需 Amazon Comprehend 情緒分析如何運作的詳細資訊，請參閱 Amazon Comprehend 開發人員指南中的判斷情緒。

標記您的 Amazon Lex 資源

為了協助您管理 Amazon Lex 機器人、機器人別名和機器人頻道，您可以將中繼資料指派給每個資源以做為標籤。標籤是您指派給 AWS 資源的標籤。每個標籤皆包含鍵與值。

標籤可讓您以不同的方式分類您的 AWS 資源，例如依據目的、擁有者或應用程式。標籤可協助您：

- 識別和組織您的 AWS 資源。許多 AWS 資源支援標記，因此您可以對不同服務中的資源指派相同的標籤，指出資源是相關的。例如，您可以標記機器人及其使用相同標籤的 Lambda 函數。
- 配置成本。您可以在 AWS Billing and Cost Management 儀表板上啟用標籤。AWS 會使用標籤分類您的成本，並交付每月成本配置報告給您。對於 Amazon Lex，您可以使用別名特有的標籤為每個別名分配成本，但$LATEST別名。您將成本分配給$LATEST別名使用 Amazon Lex 籤。如需詳細資訊，請參閱「」使用成本分配標籤中的AWS Billing and Cost Management使用者指南。
- 控制對 資源的存取。您可以使用 Amazon Lex 的標籤，以建立政策以控制對 Amazon Lex 資源的存取。這些政策可連接至 IAM 角色或使用者，以啟用標籤型存取控制。如需詳細資訊，請參閱 ABAC 與 Amazon Lex。若要檢視身分型政策範例，以根據該資源上的標籤來限制存取資源，請參閱 使用標籤訪問資源。

您可以使用AWS Management Console，AWS Command Line Interface或 Amazon Lex API。

為您的資源建立標籤

如果您使用 Amazon Lex 主控台，則可以在建立資源時標記資源，或者稍後再新增標籤。您也可以使用主控台來更新或移除現有的標籤。

如果您是使用AWS CLI或 Amazon Lex API，請使用下列操作來管理資源的標籤：

- ListTagsForResource– 查看與資源關聯的標籤。

- PutBot和PutBotAlias– 在建立機器人或機器人別名時使用標籤。
- TagResource– 在現有資源上新增和修改標籤。
- UntagResource– 從資源移除標籤。

Amazon Lex 中的下列資源支援標記：

- 機器人 - 使用 Amazon Resource Name (ARN)，如下所示：
  - `arn:${partition}:lex:${region}:${account}:bot:${bot-name}`
- 機器人別名 - 使用如下所示的 ARN：
  - `arn:${partition}:lex:${region}:${account}:bot:${bot-name}:${bot-alias}`
- 機器人頻道 - 使用如下所示的 ARN：
  - `arn:${partition}:lex:${region}:${account}:bot-channel:${bot-name}:${bot-alias}:${channel-name}`

標籤限制

以下基本限制適用於 Amazon Lex 資源上的標籤：

- 標籤的最大數量 - 50
- 最大金鑰長度 - 128 個字元
- 最大值長度 - 256 個字元
- 金鑰與值的有效字元 — a—z、A—Z、0—9、空格和下列字元：_.:/= +-及 @
- 金鑰和值會區分大小寫。
- 請不要使用 `aws:` 做為金鑰的字首；要預訂給 AWS 使用。

標記資源 (主控台)

您可以使用主控台來管理機器人、機器人別名或機器人頻道資源上的標籤。您可以在建立資源時新增標籤，也可以從現有資源新增、修改或移除標籤。

在建立機器人時新增標籤

1. 登入AWS Management Console打開 Amazon Lex 主控台，請前往https://console.aws.amazon.com/lex/。

2. 選擇 Create (建立) 以建立新的機器人。
3. 在 Create your bot (建立您的機器人) 頁面底部，選擇 Tags (標籤)。
4. 選擇 Add tag (新增標籤)，然後新增一或更多個標籤至機器人。您最多可新增 50 個標籤。

在建立機器人別名時新增標籤

1. 登入AWS Management Console打開 Amazon Lex 主控台，請前往https://console.aws.amazon.com/lex/。
2. 選擇您要新增機器人別名的機器人。
3. 選擇 Settings (設定)。
4. 新增別名名稱、選擇機器人版本，然後選擇 Add tags (新增標籤)。
5. 選擇 Add tag (新增標籤)，然後新增一或多個標籤至機器人別名。您最多可新增 50 個標籤。

在您建立機器人頻道時新增標籤

1. 登入AWS Management Console打開 Amazon Lex 主控台，請前往https://console.aws.amazon.com/lex/。
2. 選擇您要新增機器人頻道的機器人。
3. 選擇 Channels (頻道)，然後選擇您要新增的頻道。
4. 新增機器人頻道的詳細資訊，然後選擇 Tags (標籤)。
5. 選擇 Add tag (新增標籤)，然後新增一或多個標籤至機器人頻道。您最多可新增 50 個標籤。

在匯入機器人時新增標籤

1. 登入AWS Management Console打開 Amazon Lex 主控台，請前往https://console.aws.amazon.com/lex/。
2. 選擇 Actions (動作)，然後選擇 Import (匯入)。
3. 選擇用於匯入機器人的 zip 檔案。
4. 選擇 Tags (標籤)，然後選擇 Add tag (新增標籤) 以新增一或多個標籤至機器人。您最多可新增 50 個標籤。

新增、移除或修改現有機器人上的標籤

1. 登入AWS Management Console打開 Amazon Lex 主控台，請前往https://console.aws.amazon.com/lex/。
2. 從左側選單中選擇 Bots (機器人)，然後選擇要修改的機器人。
3. 選擇 Settings (設定)，然後從左側選單中選擇 General (一般)。
4. 選擇 Tags (標籤)，然後新增、修改或移除機器人的標籤。

新增、移除或修改機器人別名上的標籤

1. 登入AWS Management Console打開 Amazon Lex 主控台，請前往https://console.aws.amazon.com/lex/。
2. 從左側選單中選擇 Bots (機器人)，然後選擇要修改的機器人。
3. 選擇 Settings (設定)，然後從左側選單中選擇 Aliases (別名)。
4. 為您要修改的別名選擇 Manage tags (管理標籤)，然後新增、修改或移除機器人別名的標籤。

新增、移除或修改現有機器人頻道上的標籤

1. 登入AWS Management Console打開 Amazon Lex 主控台，請前往https://console.aws.amazon.com/lex/。
2. 從左側選單中選擇 Bots (機器人)，然後選擇要修改的機器人。
3. 選擇 Channels (管道)。
4. 選擇 Tags (標籤)，然後新增、修改或移除機器人頻道的標籤。

標記資源 (AWS CLI)

您可以使用 AWS CLI 來管理機器人、機器人別名或機器人頻道資源上的標籤。您可以在建立機器人或機器人別名時新增標籤，也可以從機器人、機器人別名或機器人頻道新增、修改或移除標籤。

所有範例都已針對 Linux 和 macOS 進行格式化。若要在 Windows 中使用命令，請以插入符號 (^) 取代 Linux 接續字元 (\)。

在建立機器人時新增標籤

- 下列縮寫的 `put-bot` AWS CLI 命令顯示您在建立機器人時必須用來新增標籤的參數。若要實際建立機器人，您必須提供其他參數。如需詳細資訊，請參閱 步驟 4：入門 (AWS CLI)。

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

在建立機器人別名時新增標籤

- 下列縮寫的 put-bot-alias AWS CLI 命令顯示您在建立機器人別名時必須用來新增標籤的參數。若要實際建立機器人別名，您必須提供其他參數。如需詳細資訊，請參閱 練習 5：建立別名 (AWS CLI)。

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

在資源上列出標籤

- 使用 list-tags-for-resource AWS CLI 指令以顯示與機器人、機器人別名、機器人頻道相關聯的資源。

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

新增或修改資源上的標籤

- 使用 tag-resource AWS CLI 命令來新增或修改機器人、機器人別名或機器人頻道。

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

從資源移除標籤

- 使用 untag-resource AWS CLI 命令從機器人、機器人別名或機器人頻道移除標籤。

```
aws lex-models untag-resource \
```

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

開始使用 Amazon Lex

Amazon Lex 提供的 API 操作可讓您與現有的應用程式整合。如需支援的操作清單，請參閱 API 參考。您可以使用下列任一選項：

- AWS 開發套件 — 使用開發套件時，系統會使用您提供的登入資料自動簽署和驗證 Amazon Lex 的請求。這是建置您的應用程式的建議選擇。
- AWS CLI — 您可以使用訪問任何 Amazon Lex 功能，而無需編寫任何代碼。 AWS CLI
- AWS 主控台 — 主控台是開始測試和使用 Amazon Lex 的最簡單方法

如果您是 Amazon Lex 的新手，我們建議您先閱讀Amazon Lex 運作方式。

主題

步驟 1：設定 AWS 帳戶並建立管理員使用者

第一次使用 Amazon Lex 之前，請先完成下列任務：

1. 註冊成為 AWS
2. 建立使用者

註冊成為 AWS

如果您已經有 AWS 帳戶，請跳過此任務。

當您註冊 Amazon Web Services（AWS）時，您的 AWS 帳戶將自動註冊為中的所有服務 AWS，包括 Amazon Lex。您只需支付實際使用服務的費用。

使用 Amazon Lex 時，您只需為使用的資源付費。如果您是新 AWS 客戶，您可以免費開始使用 Amazon Lex。如需更多詳細資訊，請參閱 AWS 免費用量方案。

如果您已經有 AWS 帳戶，請跳至下一個工作。若您尚未擁有 AWS 帳戶，請使用下列程序建立帳戶。

建立 AWS 帳號

1. 開啟 https://portal.aws.amazon.com/billing/signup。
2. 請遵循線上指示進行。

   部分註冊程序需接收來電，並在電話鍵盤輸入驗證碼。

   當您註冊一個時 AWS 帳戶，將創建AWS 帳戶根使用者一個。根使用者有權存取該帳戶中的所有 AWS 服務 和資源。安全性最佳做法是將管理存取權指派給使用者，並僅使用 root 使用者來執行需要 root 使用者存取權的工作。

記下您的 AWS 帳戶 ID，因為下一個任務將需要它。

建立使用者

中 AWS的服務 (例如 Amazon Lex) 需要您在存取登入資料時提供登入資料，以便服務判斷您是否有權存取該服務所擁有的資源。主控台需要您的密碼。不過，我們不建議您 AWS 使用 AWS 帳戶的認證進行存取。我們建議您改進行下列動作：

- 使用 AWS Identity and Access Management (IAM) 建立使用者
- 將使用者新增至具有管理權限的 IAM 群組
- 將管理許可授予您建立的 使用者。

然後，您可以 AWS 使用特殊的 URL 和用戶的憑據進行訪問。

本指南中的「入門」練習假設您有具備管理員權限的使用者 (adminuser)。請遵循程序在您的帳戶中建立 adminuser。

建立管理員使用者並登入主控台

1. 在您的 AWS 帳戶中建立一個名為 adminuser 的管理員使用者。如需指示，請參閱 IAM 使用者指南中的建立第一個使用者和管理員群組。
2. 身為使用者，您可以 AWS Management Console 使用特殊 URL 登入。如需更多詳細資訊，請參閱《IAM 使用者指南》中的使用者如何登入您的帳戶。

如需 IAM 的詳細資訊，請參閱下列各項：

- AWS Identity and Access Management (IAM)
- 入門
- IAM 使用者指南

後續步驟

步驟 2：設定 AWS Command Line Interface

步驟 2：設定 AWS Command Line Interface

如果您偏好搭配 AWS Command Line Interface (AWS CLI) 使用 Amazon Lex，請下載並進行設定。

> **Important**
>
> 您不需 AWS CLI 要執行入門練習中的步驟。不過，本指南中稍後的一些練習會用到 AWS CLI。如果您想要先從使用主控台開始著手，請略過此步驟並移至步驟 3：開始使用 (主控台)。稍後，當您需要時 AWS CLI，請返回此處進行設置。

若要設定 AWS CLI

1. 下載和設定 AWS CLI。如需說明，請參閱《AWS Command Line Interface 使用者指南》中的下列主題：

   - 取得設定 AWS Command Line Interface
   - 設定 AWS Command Line Interface

2. 將管理員使用者的具名設定檔新增至 AWS CLI 組態檔的結尾。您可以在執行 AWS CLI 指令時使用此設定檔。如需具名描述檔的詳細資訊，請參閱《AWS Command Line Interface 使用者指南》中的具名描述檔。

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

   如需可用 AWS 區域的清單，請參閱中的區域和端點Amazon Web Services 一般參考。

3. 在命令提示字元中輸入 Help 命令以驗證設定：

```
aws help
```

步驟 3：開始使用 (主控台)

步驟 3：開始使用 (主控台)

學習如何使用 Amazon Lex 的最簡單方法是使用主控台。為了協助您開始著手，我們建立了以下練習，全都是使用主控台：

- 練習 1 — 使用藍圖建立 Amazon Lex 機器人，該藍圖是一種預先定義的機器人，可提供所有必要的機器人組態。您只需執行最少的工作來測試 end-to-end 設定。

  此外，您可以使用由 AWS Lambda提供的 Lambda 函數藍圖來建立 Lambda 函數。該函數是一種程式碼掛勾，會使用與您的機器人相容之預先定義的程式碼。
- 練習 2 — 透過手動建立和設定機器人來建立自訂機器人。您也可以建立 Lambda 函數做為程式碼掛接。當中有提供範本程式碼。
- 練習 3 — 發佈機器人，然後建立該機器人的新版本。在本練習中，您還會建立別名，指向該機器人版本。

主題

練習 1：使用藍圖 (主控台) 建立 Amazon Lex 機器人

在本練習中，您會進行以下動作：

- 建立您的第一個 Amazon Lex 機器人，並在 Amazon Lex 主控台中進行測試。

  對於本練習，您將使用OrderFlowers藍圖。如需藍圖的相關資訊，請參閱 Amazon Lex 和AWS LambdaBlueprints (藍圖)。

- 建立AWS Lambda函數並在 Lambda 主控台中進行測試。在處理請求時，您的機器人會呼叫此 Lambda 函數。在本練習中，您可以使用AWS Lambda主控台中提供的 Lambda 藍圖 (lex-order-flowers-python) 來建立 Lambda 函數。藍圖程式碼說明如何使用相同的 Lambda 函數來執行初始化和驗證，以及實現`OrderFlowers`意圖。

- 更新機器人，將 Lambda 函數新增為程式碼掛接，以達成意圖。測試體 end-to-end 驗。

以下各節說明藍圖的作用。

Amazon Lex 機器人：藍圖概述

您可以使用OrderFlowers藍圖建立 Amazon Lex 機器人。如需有關機器人結構的詳細資訊，請參閱Amazon Lex 運作方式。此機器人已預先設定如下：

- 意圖 — OrderFlowers
- 槽類型 – 一個稱為 `FlowerTypes` 的自訂槽類型，具有列舉值：`roses`、`lilies` 和 `tulips`。
- 槽 – 意圖需要以下資訊 (也就是槽)，方能使機器人實現意圖。
  - `PickupTime` (AMAZON.TIME 內建類型)
  - `FlowerType`(FlowerTypes 自訂類型)
  - `PickupDate` (AMAZON.DATE 內建類型)
- 表達用語 – 以下範例表達用語代表使用者的意圖：
  - 「我想要取花。」
  - 「我想要訂花。」
- 提示 – 機器人確定意圖之後，會使用以下提示來填充槽：
  - `FlowerType` 槽的提示 –「您想要訂購哪一種花？」
  - 提示輸入`PickupDate`插槽 —「您希望 {FlowerType} 在哪一天被拾取？」
  - 提示輸入`PickupTime`插槽 —「您希望何時拾取 {FlowerType}？」
  - 確認聲明 —「好的，您的 {FlowerType} 將準備好在 {PickupTime} 上取件。PickupDate這樣可以嗎？」

AWS Lambda 函數：藍圖摘要

本練習中的 Lambda 函數會同時執行初始化和驗證以及履行工作。因此，在建立 Lambda 函數之後，您可以將相同的 Lambda 函數指定為程式碼掛接，以處理初始化和驗證和履行工作，以更新意圖組態。

- 作為初始化和驗證程式碼掛鉤，Lambda 函數會執行基本驗證。例如，如果使用者提供的取件時間不在正常工作時間，Lambda 函數會指示 Amazon Lex 重新提示使用者提示時間。
- 作為履行程式碼掛接的一部分，Lambda 函數會傳回摘要訊息，指出已下花訂單 (也就是說，意圖已達成)。

後續步驟

步驟 1：建立 Amazon Lex 機器人

步驟 1：建立 Amazon Lex 機器人

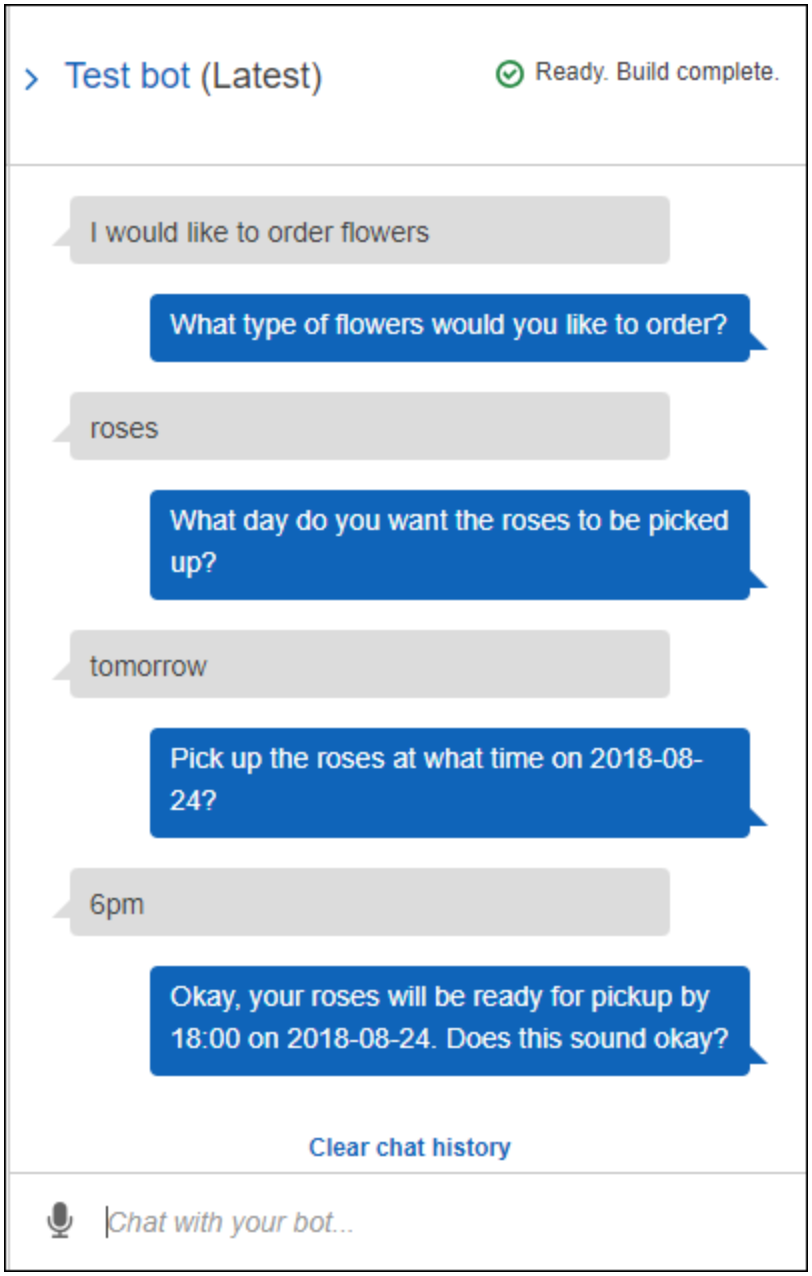
對於這個練習，創建一個機器人訂購鮮花，稱為 OrderFlowersBot。

若要建立 Amazon Lex 機器人 (主控台)

1. 登入，AWS Management Console並在 https://console.aws.amazon.com/lex/ 開啟 Amazon Lex 主控台。
2. 如果這是您第一個機器人，請選擇 Get Started (開始)，否則在機器人頁面，選擇建立。
3. 在建立您的機器人頁面上提供以下資訊，然後選擇建立。
   - 選擇OrderFlowers藍圖。
   - 保留預設機器人名稱 (OrderFlowers)。
   - 對於 COPPA (COPPA)，選擇**No**。
   - 對於使用者話語儲存，請選擇適當的回應。
4. 選擇 Create (建立)。主控台向 Amazon Lex 發出必要的請求以儲存組態。而後，主控台將顯示機器人編輯器視窗。
5. 等待機器人已建置的確認。
6. 測試機器人。

> **Note**
>
> 透過在測試視窗中輸入文字即可測試機器人，或者對於相容的瀏覽器，可由測試視窗中選擇麥克風按鈕並說話。

使用以下範例文字與機器人進行對話來訂花：



機器人憑藉輸入的內容推斷 `OrderFlowers` 意圖並提示提供槽資料。在您提供所有必要的槽資料後，機器人會將所有資訊傳回用戶端應用程式 (本例中即主控台) 以實現意圖 (`OrderFlowers`)。主控台在測試視窗中顯示這類資訊。

具體而言：

- 陳述式「What day do you want the roses to be picked up?」中出現了「roses」一詞，是因為 `pickupDate` 槽的提示已使用替換項 `{FlowerType}` 進行設定。從主控台可確認此項。
- 陳述式「Okay, your roses will be ready...」是您設定的確認提示。
- 最後一個陳述式「`FlowerType:roses...`」是傳回給用戶端 (本例中即測試視窗) 的槽資料。在下一個練習中，您會使用 Lambda 函數來完成意圖，在這種情況下，您會收到一則訊息，指出訂單已完成。

後續步驟

步驟 2 (選用)：檢閱資訊流程的詳細資訊 (主控台)

步驟 2 (選用)：檢閱資訊流程的詳細資訊 (主控台)

本節說明用戶端和 Amazon Lex 之間針對範例對話中每個使用者輸入的資訊流程。

此範例使用主控台測試視窗進行與機器人的對話。

若要開啟 Amazon Lex 測試視窗

1. 登入，AWS Management Console並在 https://console.aws.amazon.com/lex/ 開啟 Amazon Lex 主控台。
2. 選擇要測試的機器人。
3. 在主控台右側，選擇測試聊天機器人。

要查看口語化或輸入型內容的資訊流程，請選擇相應的主題。

主題

步驟 2a (選用)：檢閱口語化資訊流程的詳細資訊 (主控台)

本節說明用戶端使用語音傳送請求時，用戶端和 Amazon Lex 之間的資訊流程。如需詳細資訊，請參閱PostContent。

1. 使用者說：我想要訂花。

   a. 用戶端 (主控台) 傳送以下 PostContent 請求給 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
```

   請求 URI 和本文都向 Amazon Lex 提供信息：

   - 請求 URI — 提供機器人名稱 (OrderFlowers)、bot 別名 ($LATEST) 和使用者名稱 (可識別使用者的隨機字串)。 content表示這是PostContent API 請求 (不是PostText請求) 。
   - 要求標頭
     - x-amz-lex-session-attributes— 以底 64 編碼的值代表「{}」。用戶端發出第一次請求時，沒有工作階段屬性。
     - Content-Type – 反映音訊格式。
   - 請求內文 – 使用者輸入音訊串流「我想要訂花。」

   > **Note**
   >
   > 如果使用者選擇傳送「我想要訂花」的文字給 PostContent API 而非使用語音，請求內文就會是使用者輸入。Content-Type 標頭會相應地進行設定：
   >
   > ```
   > 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. 從輸入串流中，Amazon Lex 偵測到意圖 (OrderFlowers)。而後，其將選擇該意圖的其中一個槽 (本例中為 FlowerType) 和其中一個值引出提示，接著傳送具有以下標頭的回應：

```
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==
```

標頭值提供以下資訊：

- x-amz-lex-input-transcript – 提供來自請求的音訊 (使用者輸入) 的文本
- x-amz-lex-message— 提供在響應中返回的音頻 Amazon Lex 的成績單
- x-amz-lex-slots – base64 編碼版本的槽和值：

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

- x-amz-lex-session-attributes – base64 編碼版本的工作階段屬性 ({})

用戶端播放回應內文中的音訊。

2. 使用者說：玫瑰

a. 用戶端 (主控台) 傳送以下 PostContent 請求給 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")
```

請求內文是使用者輸入音訊串流「玫瑰」。sessionAttributes 仍為空白。

b. Amazon Lex 會根據目前意圖的內容解譯輸入串流 (記得已向此使用者詢問與FlowerType插槽相關的資訊)。Amazon Lex 會先更新目前意圖的插槽值。而後，其將選擇另一個槽 (PickupDate) 以及該槽的其中一則提示訊息 (您想要在什麼時間拿取玫瑰？)，並傳回具有以下標頭的回應：

```
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=
```

標頭值提供以下資訊：

- x-amz-lex-slots – base64 編碼版本的槽和值：

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

- x-amz-lex-session-attributes – base64 編碼版本的工作階段屬性 ({})

用戶端播放回應內文中的音訊。

3. 使用者說：明天

a. 用戶端 (主控台) 傳送以下 PostContent 請求給 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")
```

請求內文是使用者輸入音訊串流「明天」。sessionAttributes 仍為空白。

b. Amazon Lex 會根據目前意圖的內容解譯輸入串流 (記得已向此使用者詢問與PickupDate插槽相關的資訊)。Amazon Lex 會針對目前的意圖更新插槽 (PickupDate) 值。而後，其將選擇另一個槽 (PickupTime) 來引出槽值，並且選擇其中一個值引出提示 (您想要在 2017 年 3 月 18 日什麼時間拿取玫瑰？)，接著傳回具有以下標頭的回應：

```
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
```

標頭值提供以下資訊：

- x-amz-lex-slots – base64 編碼版本的槽和值：

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

- x-amz-lex-session-attributes – base64 編碼版本的工作階段屬性 ({})

用戶端播放回應內文中的音訊。

4. 使用者說：下午 6 點

a. 用戶端 (主控台) 傳送以下 PostContent 請求給 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")
```

請求內文是使用者輸入音訊串流「下午 6 點」。sessionAttributes 仍為空白。

b. Amazon Lex 會根據目前意圖的內容解譯輸入串流 (記得已向此使用者詢問與PickupTime插槽相關的資訊)。其將首先更新目前意圖的槽值。

現在，Amazon Lex 偵測到它具有所有插槽的資訊。不過，OrderFlowers 意圖設定了一則確認訊息。因此，Amazon Lex 需要使用者明確確認，才能繼續履行意圖。訂花之前，其將傳送具有以下要求確認標頭的回應：

```
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
```

標頭值提供以下資訊：

- x-amz-lex-slots – base64 編碼版本的槽和值：

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

- x-amz-lex-session-attributes – base64 編碼版本的工作階段屬性 ({})

用戶端播放回應內文中的音訊。

5. 使用者說：好

a. 用戶端 (主控台) 傳送以下 PostContent 請求給 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")
```

請求內文是使用者輸入音訊串流「好」。sessionAttributes 仍為空白。

b. Amazon Lex 會解譯輸入串流，並瞭解使用者想要繼續執行訂單。OrderFlowers 意圖設定了 ReturnIntent 做為履行活動。這會指示 Amazon Lex 將所有意圖資料傳回給用戶端。Amazon Lex 返回具有以下內容的回應：

```
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
```

x-amz-lex-dialog-state 回應標頭設為 ReadyForFulfillment。隨後用戶端即可實現意圖。

6. 現在，重新測試機器人。要建立新的 (使用者) 內容，請由主控台選擇 Clear (清除) 連結。為 OrderFlowers 意圖提供資料，包括一些無效的資料。例如：

   - 花種為「茉莉」(此花種不受支援)
   - 想要取花的日期為「昨天」

   請注意，機器人會接受這些值，因為您沒有任何程式碼來初始化和驗證使用者資料。在下一節中，您可以新增 Lambda 函數來執行此操作。請注意 Lambda 函數的事項：

   - 函數將於使用者每次輸入後驗證槽資料。其將在結束時實現意圖。也就是說，機器人會處理訂花的下單，然後向使用者傳回一則訊息，而不單只是將槽資料傳回用戶端。如需詳細資訊，請參閱使用 Lambda 函數。
   - 函數還將設定工作階段屬性。如需工作階段屬性的詳細資訊，請參閱 PostText。

     完成入門章節後，您可以接著做其他練習 (其他示例：創建亞馬遜 Lex 機器人)。預訂行程 將利用工作階段屬性，透過跨意圖共享資訊與使用者進行動態對話。

後續步驟

步驟 3：建立 Lambda 函數

步驟 2b (選用)：檢閱輸入型資訊流程的詳細資訊 (主控台)

本節說明用戶端與 Amazon Lex 之間由用戶端使用 PostText API 傳送請求時的資訊流程。如需詳細資訊，請參閱PostText。

1. 使用者輸入：我想要訂花

   a. 用戶端 (主控台) 傳送以下 PostText 請求給 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": {}
}
```

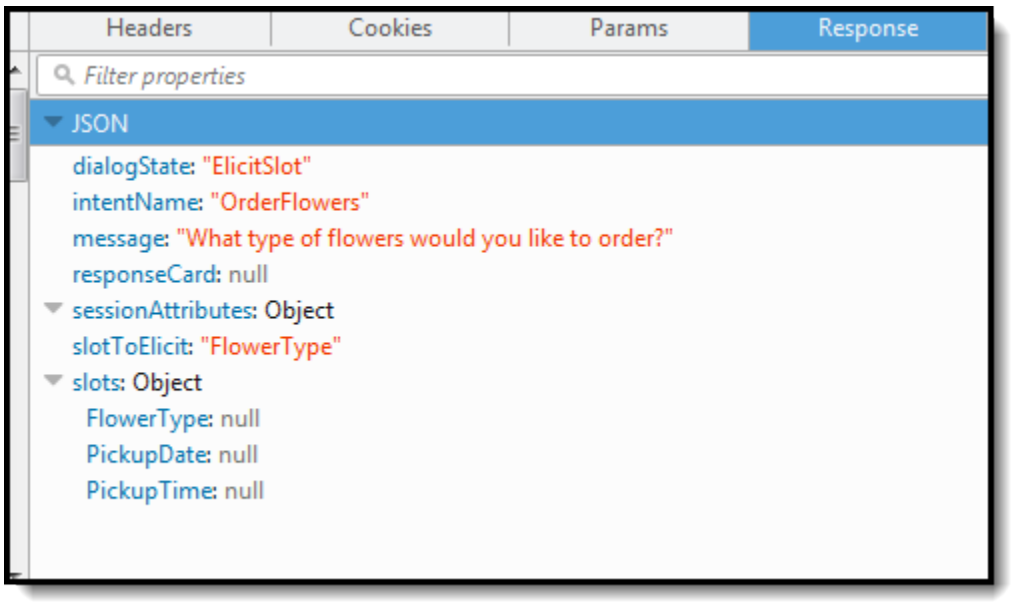
   請求 URI 和本文都向 Amazon Lex 提供信息：

   - 請求 URI — 提供機器人名稱 (OrderFlowers)、bot 別名 ($LATEST) 和使用者名稱 (識別使用者的隨機字串)。末尾的 text 表示其為 PostText API 請求 (而非 PostContent)。

   - 請求本文 – 包含使用者輸入 (inputText) 和空的 sessionAttributes。用戶端發出第一次請求時，沒有工作階段屬性。稍後將由 Lambda 函數起始這些屬性。

   b. 從中inputText，Amazon Lex 檢測到意圖（ OrderFlowers ）。此意圖沒有任何用於初始化和驗證使用者輸入或履行的程式碼掛接 (也就是 Lambda 函數)。

   Amazon Lex 選擇其中一個意圖的插槽（ FlowerType ）來引出價值。其亦將選取槽 (整個意圖組態) 的其中一個值引出提示，然後傳回以下回應給用戶端。主控台向使用者顯示回應中的訊息。



用戶端顯示回應中的訊息。

2. 使用者輸入：玫瑰

   a. 用戶端 (主控台) 傳送以下 PostText 請求給 Amazon Lex：

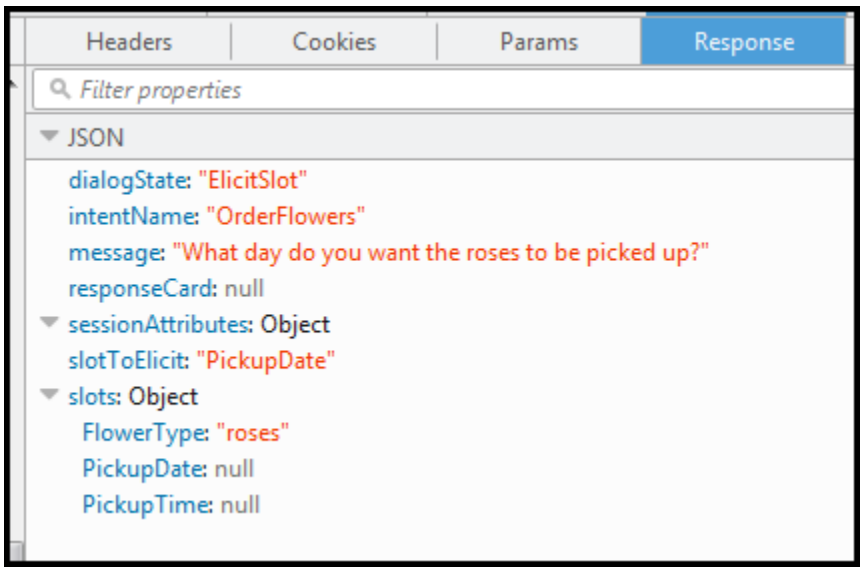
```
POST /bot/OrderFlowers/alias/$LATEST/user/4o9wwdhx6nlheferh6a73fujd3118f5w/text
"Content-Type":"application/json"
"Content-Encoding":"amz-1.0"

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

   請求內文中的 `inputText` 會提供使用者輸入。`sessionAttributes` 仍為空白。

   b. Amazon Lex 首先會`inputText`在目前意圖的內容中解譯，服務會記住已向特定使用者詢問`FlowerType`插槽的相關資訊。Amazon Lex 首先更新目前意圖的插槽值，並選擇另一個位置 (`PickupDate`) 以及其中一個提示訊息 — 您希望什麼日子收到玫瑰花？ — 插槽。

   隨後，Amazon Lex 會傳回：



用戶端顯示回應中的訊息。

3. 使用者輸入：明天

   a. 用戶端 (主控台) 傳送以下 PostText 請求給 Amazon Lex：

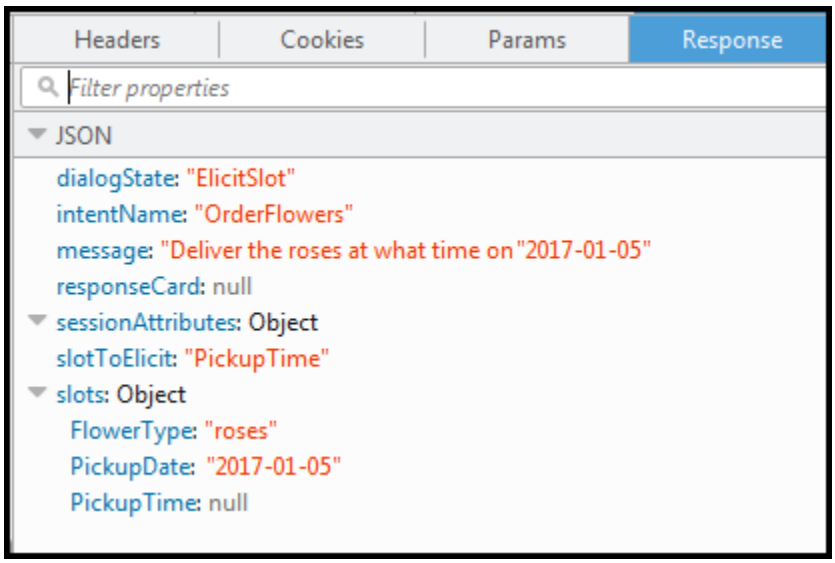
```
POST /bot/OrderFlowers/alias/$LATEST/user/4o9wwdhx6nlheferh6a73fujd3118f5w/text
"Content-Type":"application/json"
"Content-Encoding":"amz-1.0"

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

   請求內文中的 `inputText` 會提供使用者輸入。`sessionAttributes` 仍為空白。

   b. Amazon Lex 首先會`inputText`在目前意圖的內容中解譯，服務會記住已向特定使用者詢問`PickupDate`插槽的相關資訊。Amazon Lex 會針對目前的意圖更新插槽 (`PickupDate`) 值。其將選擇另一個槽 (`PickupTime`) 來引出槽值。它返回了價值引起的提示之一-在 2017-01-05 什麼時候交付玫瑰？ — 給客戶端。

   Amazon Lex 隨後會傳回：



用戶端顯示回應中的訊息。

4. 使用者輸入：下午 6 點

   a. 用戶端 (主控台) 傳送以下 PostText 請求給 Amazon Lex：

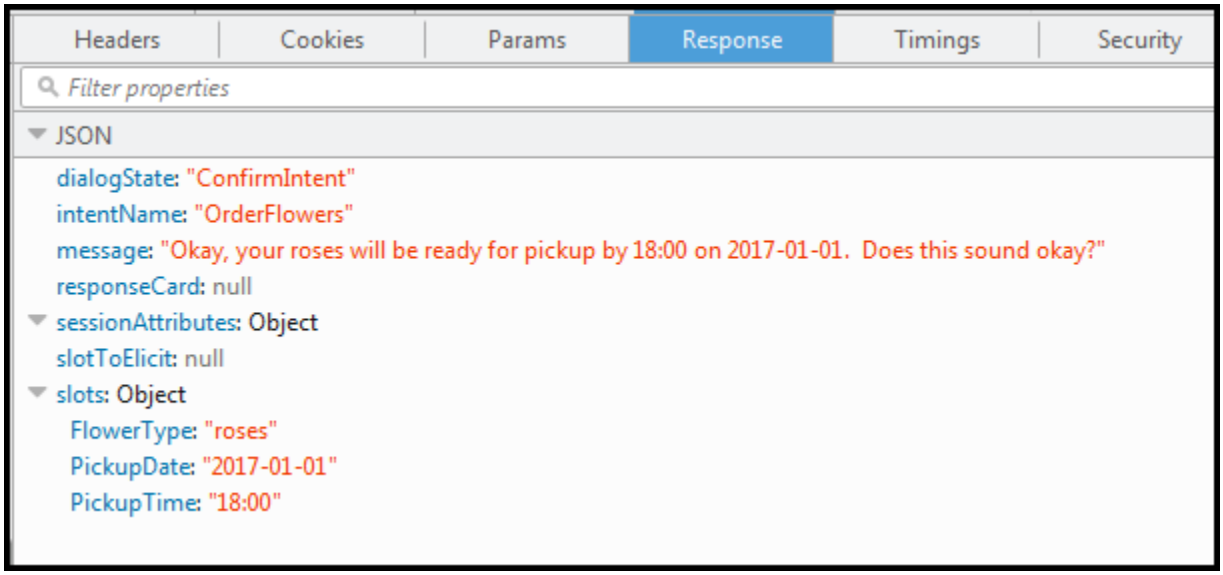
```
POST /bot/OrderFlowers/alias/$LATEST/user/4o9wwdhx6nlheferh6a73fujd3118f5w/text
"Content-Type":"application/json"
"Content-Encoding":"amz-1.0"

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

   請求內文中的 `inputText` 會提供使用者輸入。`sessionAttributes` 仍為空白。

   b. Amazon Lex 首先會`inputText`在目前意圖的內容中解譯，服務會記住已向特定使用者詢問`PickupTime`插槽的相關資訊。Amazon Lex 會先更新目前意圖的插槽值。現在，Amazon Lex 偵測到它具有所有插槽的資訊。

   `OrderFlowers` 意圖設定了一則確認訊息。因此，Amazon Lex 需要使用者明確確認，才能繼續履行意圖。Amazon Lex 在訂購鮮花前，會傳送下列訊息給客戶，要求確認：



用戶端顯示回應中的訊息。

5. 使用者輸入：好

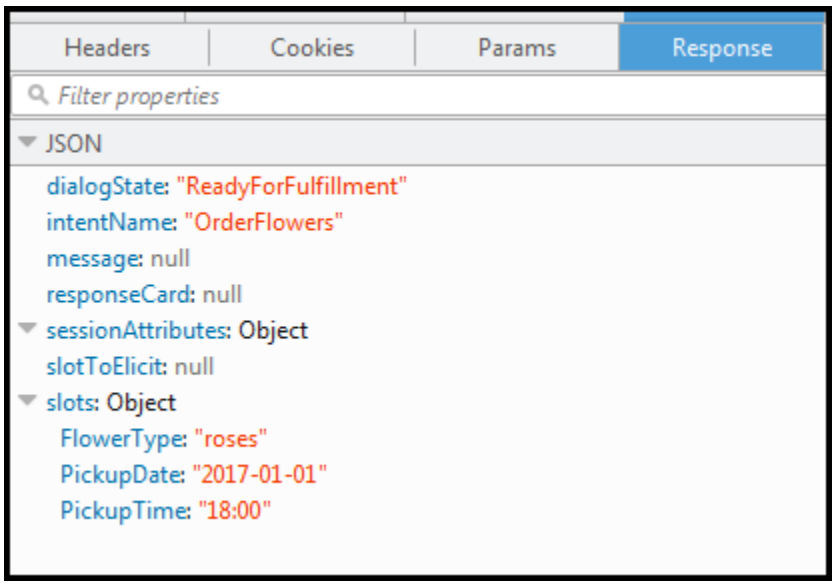
   a. 用戶端 (主控台) 傳送以下 PostText 請求給 Amazon Lex：

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

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

   請求內文中的 `inputText` 會提供使用者輸入。`sessionAttributes` 仍為空白。

   b. Amazon Lex 會`inputText`在確認目前意圖的內容中解譯。其已理解使用者想要完成下單。`OrderFlowers`意圖設定`ReturnIntent`為履行活動 (沒有 Lambda 函數可完成意圖)。因此，Amazon Lex 會將下列插槽資料傳回給用戶端。



Amazon Lex 設置`dialogState`為`ReadyForFulfillment`。隨後用戶端即可實現意圖。

6. 現在，再次測試機器人。為此，您必須由主控台選擇 Clear (清除) 連結以建立新的 (使用者) 內容。接著為訂花意圖提供資料，請嘗試提供無效的資料。例如：

   - 花種為「茉莉」(此花種不受支援)，
   - 想要取花的日期為「昨天」。

   請注意，機器人會接受這些值，因為您沒有任何程式碼來初始化/驗證使用者資料。在下一節中，您可以新增 Lambda 函數來執行此操作。請注意 Lambda 函數的事項：

   - Lambda 函數會在每次使用者輸入之後驗證插槽資料。其將在結束時實現意圖。也就是說，機器人會處理訂花的下單，然後向使用者傳回一則訊息，而不單只是將槽資料傳回用戶端。如需詳細資訊，請參閱使用 Lambda 函數。
   - Lambda 函數也會設定工作階段屬性。如需工作階段屬性的詳細資訊，請參閱 PostText。

     完成入門章節後，您可以接著做其他練習 (其他示例：創建亞馬遜 Lex 機器人)。預訂行程 將利用工作階段屬性，透過跨意圖共享資訊與使用者進行動態對話。

後續步驟

步驟 3：建立 Lambda 函數

步驟 3：建立 Lambda 函數

使用AWS Lambda主控台中的範例事件資料建立 Lambda 函數 (使用lex-order-flowers-python藍圖) 並執行測試叫用。

您可以返回 Amazon Lex 主控台並新增 Lambda 函數做為程式碼掛接，以達成您在`OrderFlowersBot`上一節中建立的`OrderFlowers`意圖。

建立 Lambda 函數 (主控台)

1. 請登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/lambda/ 的 AWS Lambda 主控台。
2. 選擇 Create function (建立函數)。
3. 在 Create function (建立函數) 頁面上，選擇 Use a blueprint (使用藍圖)。在篩選條件的文字方塊中輸入 **`lex-`** 然後按 `Enter` 以尋找藍圖，然後選擇 `lex-order-flowers-python` 藍圖。

   Lambda 函數藍圖在兩個 Node.js 和蟒蛇提供。本練習將使用 Python 提供的藍圖。
4. 在 Basic information (基本資訊) 頁面上，執行以下作業。
   - 輸入 Lambda 函數名稱 (`OrderFlowersCodeHook`)。
   - Execution role (執行角色) 請選擇 Create a new role the basic Lambda permissions (建立具備基本
   - 保留其他預設值。
5. 選擇 Create function (建立函數)。
6. 如果您使用的是英文 (US) (en-US) 以外的地區設定，請依照中所述更新意圖名稱更新特定區域設置的藍圖。
7. 測試 Lambda 函數。
   a. 選擇 Select a test event (選取測試事件)、Configure test events (設定測試事件)。
   b. 從 Event template (事件範本) 清單中選擇 Amazon Lex Order Flowers。這個範例事件與Amazon Lex 請求/回應模型相符 (請參閱使用 Lambda 函數)。為測試事件命名 (`LexOrderFlowersTest`)。
   c. 選擇 Create (建立)。
   d. 選擇 Test (測試) 來測試程式碼掛勾。
   e. 確認 Lambda 函數是否成功執行。在這種情況下，回應與 Amazon Lex 回應模型相符。

後續步驟

步驟 4：將 Lambda 函數新增為程式碼掛接 (主控台)

步驟 4：將 Lambda 函數新增為程式碼掛接 (主控台)

在本節中，您將更新 OrderFlowers意圖的組態以使用 Lambda 函數，如下所示：

- 首先使用 Lambda 函數做為程式碼掛接，以執行OrderFlowers意圖的履行。您可以測試機器人，並確認您已收到來自 Lambda 函數的履行訊息。Amazon Lex 只有在您提供訂購鮮花所需插槽的資料後，才會叫用 Lambda 函數。
- 將相同的 Lambda 函數設定為程式碼掛接，以執行初始化和驗證。您可以測試並驗證 Lambda 函數是否執行驗證 (當您提供位置資料時)。

若要將 Lambda 函數新增為程式碼掛接 (主控台)

1. 在 Amazon Lex 主控台中，選取OrderFlowers機器人。控制台顯示OrderFlowers意圖。確定意圖版本已設為 $LATEST，因為這是唯一能夠修改的版本。
2. 將 Lambda 函數新增為履行程式碼掛鉤並對其進行測試。

   a. 在編輯器中，選擇AWS Lambda函數作為履行，然後選取您在上一個步驟 (OrderFlowersCodeHook) 中建立的 Lambda 函數。選擇 OK 以授與 Amazon Lex Lambda 函數的許可。

      您將此 Lambda 函數設定為程式碼掛接，以實現意圖。Amazon Lex 只有在擁有使用者提供的所有必要插槽資料以達成意圖之後，才會呼叫此函數。

   b. 指定 Goodbye message (再見訊息)。

   c. 選擇 Build (建置)。

   d. 使用之前的對話測試機器人。

   最後一個聲明「謝謝，您的玫瑰訂單...」是您配置為代碼鉤子的 Lambda 函數的響應。在前一節中，沒有 Lambda 函數。現在您正在使用 Lambda 函數來實現OrderFlowers意圖。
3. 將 Lambda 函數新增為初始化和驗證程式碼掛接，然後進行測試。

   您使用的範例 Lambda 函數程式碼可以執行使用者輸入驗證和履行。Lambda 函數收到的輸入事件具有 field (invocationSource)，程式碼會使用此欄位來決定要執行的程式碼部分。如需詳細資訊，請參閱Lambda 函數輸入事件和回應格式。

a. 選取 $LATEST 版本的 `OrderFlowers` 意圖。這是唯一能夠更新的版本。

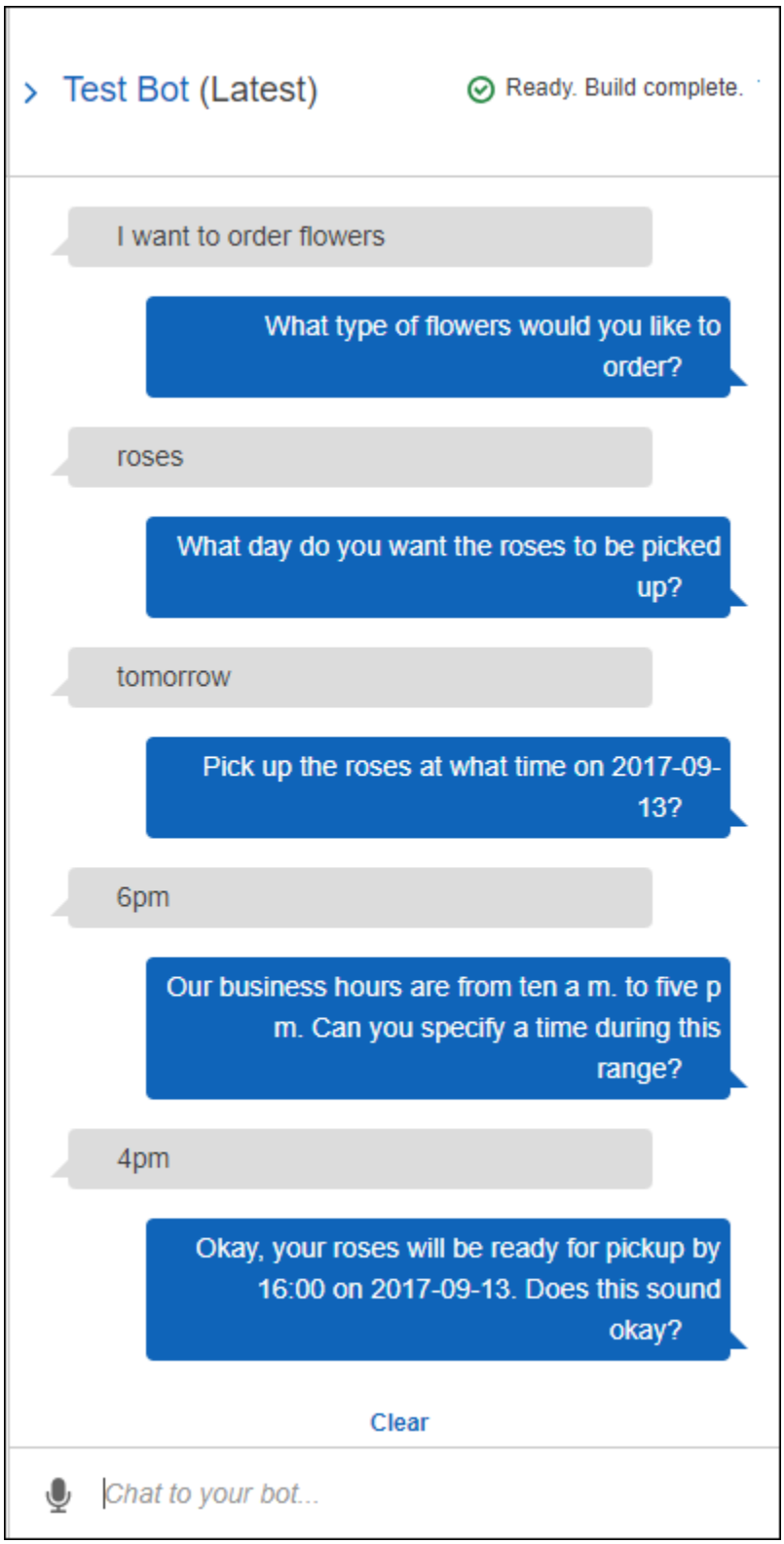
b. 從編輯器的 Options (選項)中選擇 Initialization and validation (初始化和驗證)。

c. 同樣地，選取相同的 Lambda 函數。

d. 選擇 Build (建置)。

e. 測試機器人。

   您現在已準備好與 Amazon Lex 交談，如下圖所示。若要測試驗證部分，請選擇下午 6 點的時間，Lambda 函數會傳回回應 (「我們的工作時間從上午 10 點到下午 5 點」)，然後再次提示您。在您提供所有有效的插槽資料之後，Lambda 函數會完成訂單。


Test Bot (Latest)
Ready. Build complete.
I want to order flowers
What type of flowers would you like to order?
roses
What day do you want the roses to be picked up?
tomorrow
Pick up the roses at what time on 2017-09-13?
6pm
Our business hours are from ten a m. to five p m. Can you specify a time during this range?
4pm
Okay, your roses will be ready for pickup by 16:00 on 2017-09-13. Does this sound okay?
Clear
Chat to your bot...

後續步驟

步驟 5 (選用)：檢閱資訊流程的詳細資訊 (主控台)

步驟 5 (選用)：檢閱資訊流程的詳細資訊 (主控台)

本節說明用戶端和 Amazon Lex 之間針對每個使用者輸入的資訊流程，包括 Lambda 函數的整合。

> **Note**
>
> 本節假設用戶端使用PostText執行階段 API 將請求傳送至 Amazon Lex，並據此顯示請求和回應詳細資訊。如需用戶端與用戶端使用PostContent API 之 Amazon Lex 之間的資訊流程範例，請參閱步驟 2a (選用)：檢閱口語化資訊流程的詳細資訊 (主控台)。

如需 PostText 執行時間 API 的詳細資訊及關於以下步驟所示的請求與回應的更多細節，請參閱 PostText。

1. 使用者：我想要訂花。

   a. 用戶端 (主控台) 傳送以下 PostText 請求給 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": {}
   }
   ```

   請求 URI 和本文都向 Amazon Lex 提供信息：

   - 請求 URI — 提供機器人名稱 (OrderFlowers)、bot 別名 ($LATEST) 和使用者名稱 (識別使用者的隨機字串)。末尾的 text 表示其為 PostText API 請求 (而非 PostContent)。
   - 請求本文 – 包含使用者輸入 (inputText) 和空的 sessionAttributes。用戶端發出第一次請求時，沒有工作階段屬性。稍後將由 Lambda 函數起始這些屬性。

   b. 從中inputText，Amazon Lex 檢測到意圖（OrderFlowers）。此目的是使用 Lambda 函數設定為使用者資料初始化和驗證的程式碼掛接。因此，Amazon Lex 會將下列資訊作為事件資料傳遞，以叫用該 Lambda 函數：

   ```
   {
   ```

```
    "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"
    }
}
```

如需詳細資訊，請參閱輸入事件格式。

除了用戶端傳送的資訊之外，Amazon Lex 還包含下列其他資料：

- `messageVersion`— 目前 Amazon Lex 僅支持 1.0 版本。
- `invocationSource`— 指出 Lambda 函數叫用的目的。在本例中，目的是執行使用者資料初始化和驗證。目前，Amazon Lex 知道使用者尚未提供所有插槽資料來達成意圖。
- `currentIntent` 資訊 – 所有槽值均設定為 null。

c. 此時，所有槽值都是 null。Lambda 函數沒有任何需要驗證的項目。Lambda 函數返回以下響應 Amazon Lex：

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

```
    }
}
```

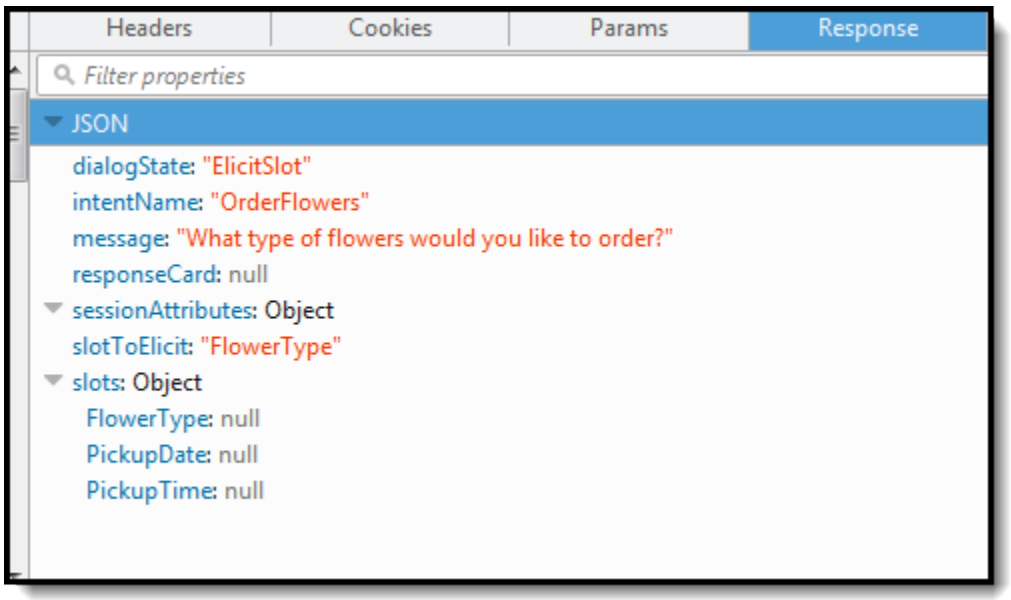
如需回應格式的相關資訊，請參閱回應格式。

注意下列事項：

- `dialogAction.type`— 將此值設定為`Delegate`，Lambda 函數將決定下一個動作方向的責任委派給 Amazon Lex。

> **Note**
>
> 如果 Lambda 函數在使用者資料驗證中偵測到任何內容，它會指示 Amazon Lex 接下來該怎麼做，如接下來的幾個步驟所示。

d. 根據`dialogAction.type`, Amazon Lex 決定行動的下一個過程. 由於沒有任何槽獲得填充，其決定引出 `FlowerType` 槽的值。服務將選取該槽的其中一個值引出提示「您想要訂購哪一種花？」，然後傳回以下回應給用戶端：



用戶端顯示回應中的訊息。

2. 使用者：玫瑰

a. 用戶端會將下列PostText要求傳送至 Amazon Lex：

```
POST /bot/OrderFlowers/alias/$LATEST/user/ignw84y6seypre4xly5rimopuri2xwnd/text
```

```
"Content-Type":"application/json"
"Content-Encoding":"amz-1.0"

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

請求內文中的 inputText 會提供使用者輸入。sessionAttributes 仍為空白。

b. Amazon Lex 首先inputText在當前意圖的上下文中解釋。服務會記住其已向具體使用者詢問過有關 FlowerType 槽的資訊。它會更新目前意圖中的插槽值，並使用下列事件資料叫用 Lambda 函數：

```
{
    "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"
    }
}
```

注意下列事項：

- invocationSource – 仍為 DialogCodeHook (僅驗證使用者資料)。
- currentIntent.slots— Amazon Lex 已經更新了FlowerType插槽玫瑰.

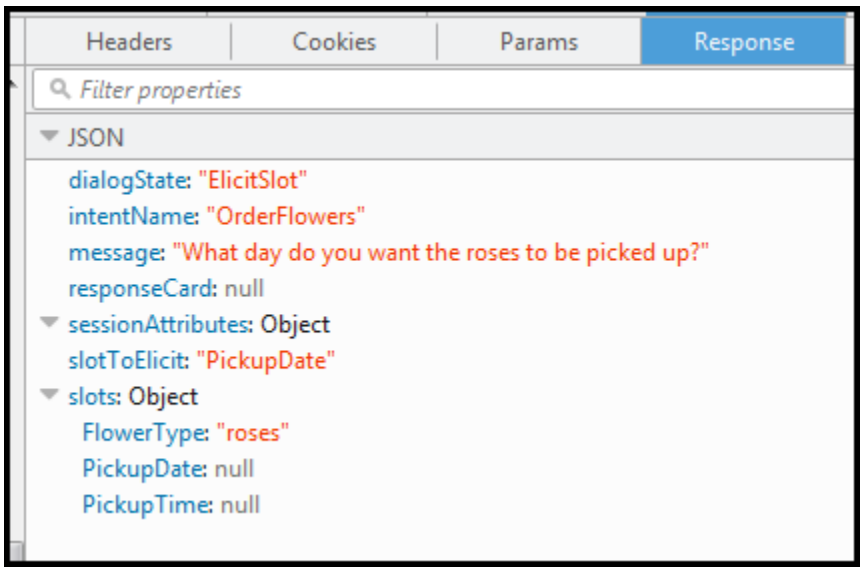
c. 根據的invocationSource值DialogCodeHook，Lambda 函數會執行使用者資料驗證。它會辨識roses為有效的插槽值 (並設定Price為工作階段屬性)，並將下列回應傳回 Amazon Lex。

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

注意下列事項：

- sessionAttributes-Lambda 函數已添加Price（玫瑰）作為會話屬性。
- dialogAction.type – 設定為 Delegate。使用者資料是有效的，因此 Lambda 函數會指示 Amazon Lex 選擇下一個動作方案。

d. 根據dialogAction.type, Amazon Lex 選擇行動的下一個過程. Amazon Lex 知道需要更多插槽資料，因此它會根據意圖組態挑選具有最高優先順序的下一個未填充的插槽 (PickupDate)。Amazon Lex 選擇其中一個價值引起的提示訊息 —「您希望什麼日子收到玫瑰花？」 —for 此插槽根據意圖配置，然後將以下響應發送回客戶端：



用戶端將顯示回應中的訊息 –「您想要在哪一天拿取玫瑰？」

3. 使用者：明天

   a. 用戶端會將下列PostText要求傳送至 Amazon Lex：

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

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

   請求內文中的 inputText 會提供使用者輸入，且用戶端會將工作階段屬性傳回給服務。

   b. Amazon Lex 會記住上下文 — 它正在為PickupDate插槽引出資料。在此情況下，服務知道 inputText 值是用於 PickupDate 槽。然後，Amazon Lex 克斯通過發送以下事件調用 Lambda 函數：

```
{
    "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 已經currentIntent.slots通過設置PickupDate值更新了。另請注意，服務將原sessionAttributes樣傳遞給 Lambda 函數。

c. 根據的invocationSource值DialogCodeHook，Lambda 函數會執行使用者資料驗證。它會識別PickupDate插槽值是有效的，並將下列回應傳回給 Amazon Lex：

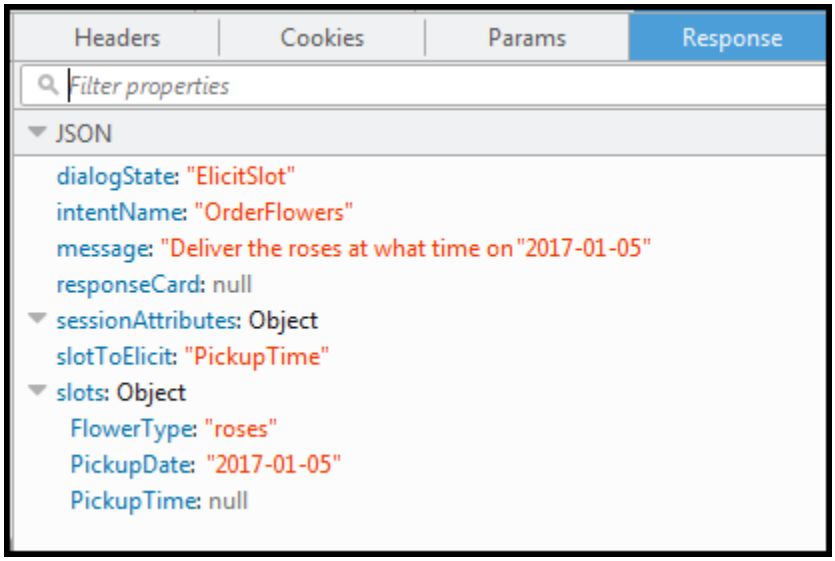
```
{
    "sessionAttributes": {
        "Price": 25
    },
    "dialogAction": {
        "type": "Delegate",
        "slots": {
            "PickupTime": null,
            "FlowerType": "roses",
            "PickupDate": "2017-01-05"
        }
    }
}
```

注意下列事項：

- sessionAttributes – 未改變。

- `dialogAction.type` – 設定為 `Delegate`。使用者資料是有效的，而 Lambda 函數會指示 Amazon Lex 選擇下一個動作方案。

d. 根據`dialogAction.type`, Amazon Lex 選擇行動的下一個過程. Amazon Lex 知道需要更多插槽資料，因此它會根據意圖組態挑選具有最高優先順序的下一個未填充的插槽 (`PickupTime`)。Amazon Lex 選擇提示消息之一 (「在 2017-01-05 什麼時候交付玫瑰？」) 根據意圖配置此插槽，並將以下響應發送回客戶端：



客戶端在響應中顯示消息-「在 2017-01-05 什麼時候交付玫瑰？」

4. 使用者：下午 4 點

a. 用戶端會將下列PostText要求傳送至 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"
    }
}
```

請求內文中的 `inputText` 會提供使用者輸入。用戶端將在請求中傳遞 `sessionAttributes`。

b. Amazon Lex 了解上下文。其明白這是稍早引出的 PickupTime 槽的資料。在這種情況下，它知道該inputText值是用於PickupTime插槽。然後，Amazon Lex 克斯通過發送以下事件調用 Lambda 函數：

```
{
    "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 已經currentIntent.slots通過設置PickupTime值更新了。

c. 根據的invocationSource值DialogCodeHook，Lambda 函數會執行使用者資料驗證。它可以識別PickupDate插槽值是有效的，並將以下響應返回給 Amazon Lex。

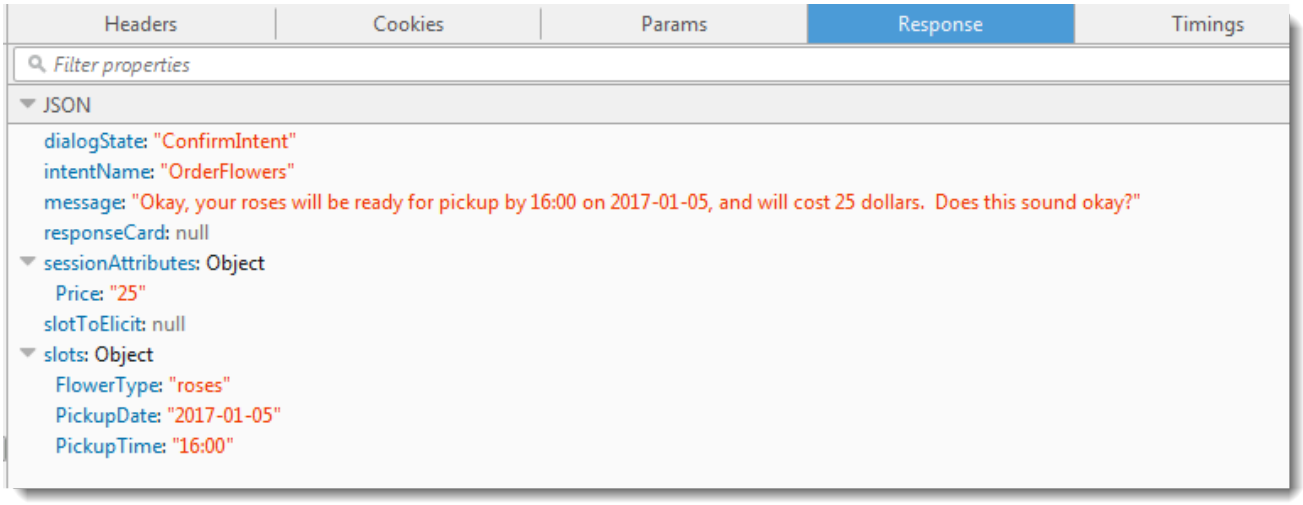
```
{
    "sessionAttributes": {
        "Price": 25
    },
    "dialogAction": {
        "type": "Delegate",
        "slots": {
            "PickupTime": "16:00",
            "FlowerType": "roses",
```

```
            "PickupDate": "2017-01-05"
        }
    }
}
```

注意下列事項：

- `sessionAttributes` – 工作階段屬性未改變。
- `dialogAction.type` – 設定為 `Delegate`。使用者資料是有效的，因此 Lambda 函數會指示 Amazon Lex 選擇下一個動作方案。

d. 目前，Amazon Lex 知道它擁有所有插槽數據。該意圖設定了一則確認提示。因此，Amazon Lex 會先傳送下列回應給要求確認的使用者，再完成意圖：



用戶端將顯示回應中的訊息並等待使用者回應。

5. 使用者：好

a. 用戶端會將下列PostText要求傳送至 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 會inputText在確認目前意圖的內容中解譯。Amazon Lex 瞭解使用者想要繼續進行訂單。這次 Amazon Lex 透過傳送下列事件來叫用 Lambda 函數來完成意圖，該事件會FulfillmentCodeHook在傳送invocationSource至 Lambda 函數的事件中設定為。Amazon Lex 也設置confirmationStatus為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"
    }
}
```

注意下列事項：

- invocationSource— 這次 Amazon Lex 將此值設置為FulfillmentCodeHook，指導 Lambda 函數以實現意圖。
- confirmationStatus – 設定為 Confirmed。

c. 這一次，Lambda 函數會完成OrderFlowers意圖，並傳回下列回應：

```
{
    "sessionAttributes": {
        "Price": "25"
```

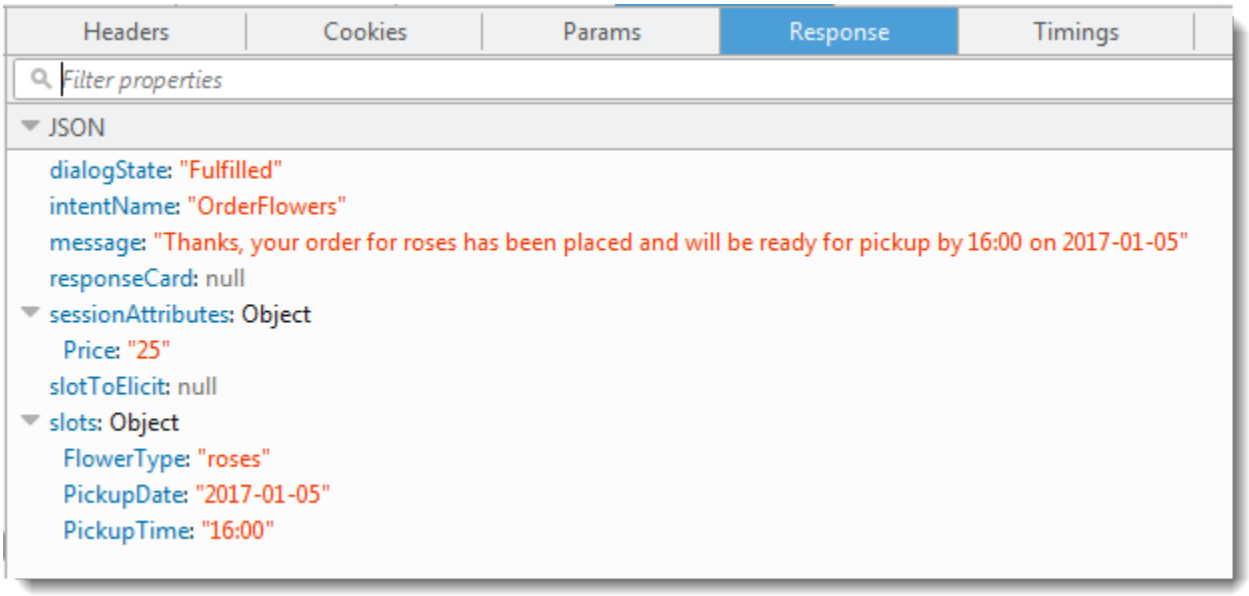
```
    },
    "dialogAction": {
        "type": "Close",
        "fulfillmentState": "Fulfilled",
        "message": {
            "contentType": "PlainText",
            "content": "Thanks, your order for roses has been placed and will
 be ready for pickup by 16:00 on 2017-01-05"
        }
    }
}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

注意下列事項：

- 設定`dialogAction.type` — Lambda 函數會將此值設定為`Close`，指示 Amazon Lex 不會預期使用者回應。
- `dialogAction.fulfillmentState` – 設定為 Fulfilled 且附上相應的 `message` 以傳達給使用者。

d. Amazon Lex 會檢閱，`fulfillmentState`並將下列回應傳回給用戶。

Amazon Lex 接著會將下列項目傳回給用戶端：



請注意：

- `dialogState`— Amazon Lex 將此值設置為`fulfilled`.
- `message`-與提供的 Lambda 函數相同的消息。

用戶端將顯示該訊息。

6. 現在，再次測試機器人。要建立新的 (使用者) 內容，請由測試視窗選擇 Clear (清除) 連結。接著為 OrderFlowers 意圖提供無效的槽資料。這次 Lambda 函數會執行資料驗證、將無效的插槽資料值重設為空值，並要求 Amazon Lex 提示使用者提供有效資料。例如，嘗試以下操作：

   - 花種為「茉莉」(此花種不受支援)，
   - 想要取花的日期為「昨天」。
   - 下單後再輸入其他花種，而不要回覆「好」確認下單。作為回應，Lambda 函數會更新工作階段Price中的屬性，以保持花卉訂單的執行總數。

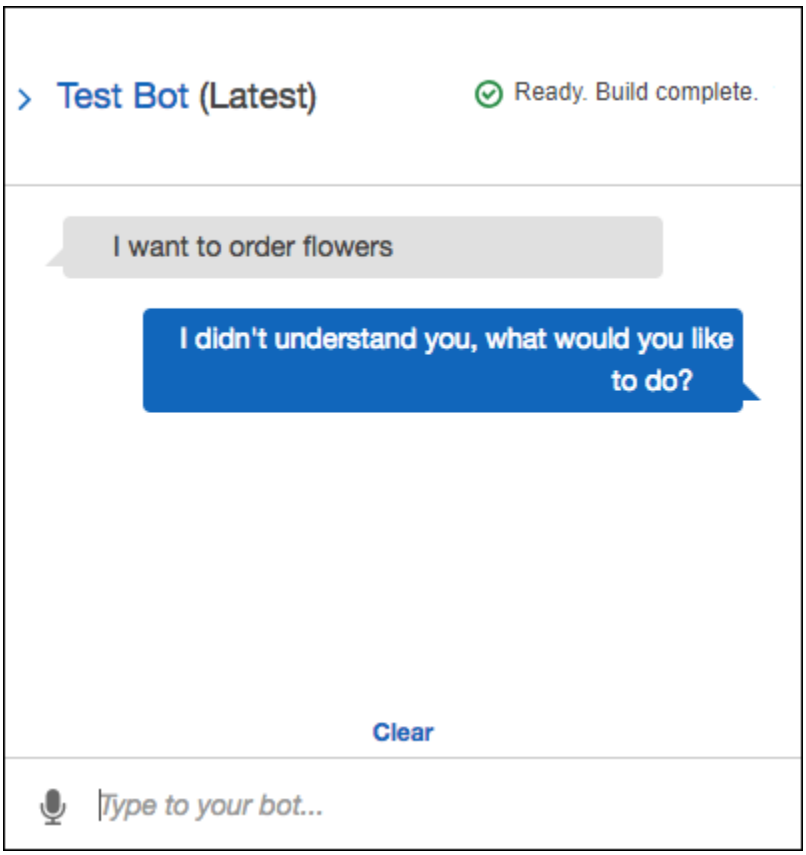
   Lambda 函數也會執行履行活動。

後續步驟

步驟 6：更新意圖組態以加入表達用語 (主控台)

步驟 6：更新意圖組態以加入表達用語 (主控台)

OrderFlowers 機器人只設定了兩個表達用語。這為 Amazon Lex 提供有限的資訊，以建立可辨識並回應使用者意圖的機器學習模型。嘗試輸入「我想訂購鮮花」，如下面的測試窗口。Amazon Lex 無法辨識文字，並回應「我不瞭解您，您想要做什麼？」 您可以透過加入更多表達用語來改善機器學習模型。

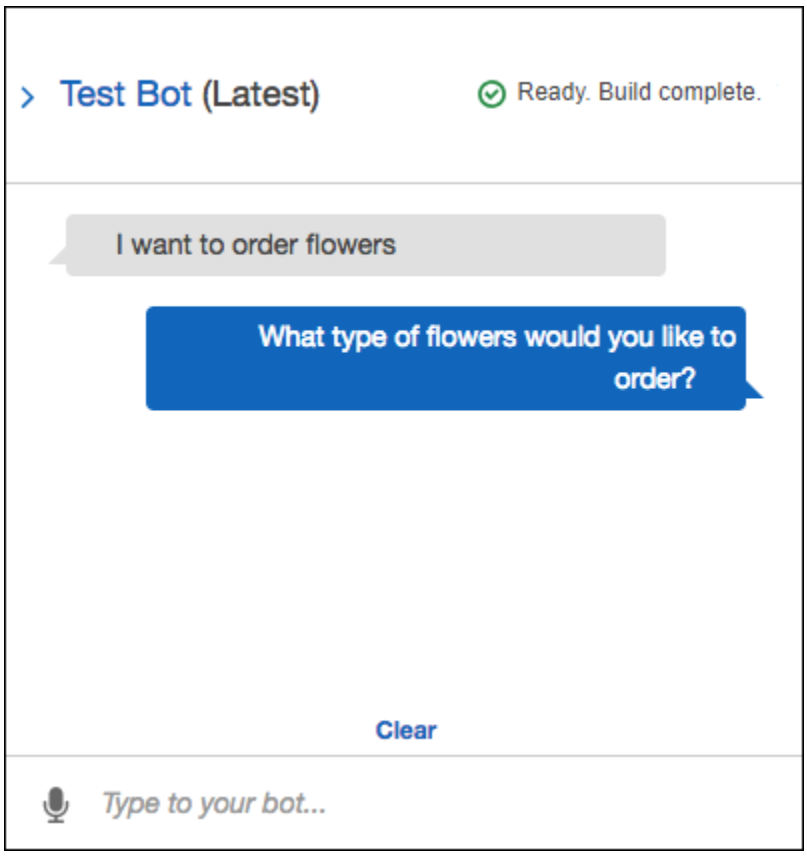


您新增的每個話語都會為 Amazon Lex 提供有關如何回應使用者的詳細資訊。Amazon Lex 不需要新增精確的話語，Amazon Lex 會從您提供的範例進行概括，以識別完全相符項目和類似輸入。

加入表達用語 (主控台)

1. 在意圖編輯器的 [範例語彙] 區段中輸入語音，如下圖所示，將「我想要花朵」新增至意圖，然後按一下新語音旁邊的加號圖示。

2. 建置您的機器人使其接受變更。選擇 Build (建置)，然後再次選擇 Build (建置)。
3. 測試您的機器人以確認其能夠判別新表達用語。在測試視窗中，如下圖所示，輸入「我要訂購鮮花」。Amazon Lex 識別這句話，並以「您想訂購什麼類型的花？」作出回應。



後續步驟

[步驟 7 (選用)：清理 (主控台)]

步驟 7 (選用)：清理 (主控台)

現在，刪除您所建立的資源並清理您的帳戶。

您只能刪除未使用的資源。一般而言，您應該按照以下順序刪除資源：

- 刪除機器人以釋放意圖資源。
- 刪除意圖以釋放槽類型資源。
- 最後刪除槽類型。

清理您的帳戶 (主控台)

1. 登入，AWS Management Console並在 https://console.aws.amazon.com/lex/ 開啟 Amazon Lex 主控台。

2. 從 Bot 清單中，選擇旁的核取方塊OrderFlowers。
3. 要刪除機器人，選擇 Delete (刪除)，然後從確認對話方塊選擇 Continue (繼續)。
4. 從左側窗格選擇 意圖。
5. 在意圖清單中，選擇OrderFlowersIntent。
6. 要刪除意圖，選擇 Delete (刪除)，然後從確認對話方塊選擇 Continue (繼續)。
7. 從左側窗格選擇 Slot types (槽類型)。
8. 從槽類型清單中選擇 Flowers。
9. 要刪除槽類型，選擇 Delete (刪除)，然後從確認對話方塊選擇 Continue (繼續)。

您已移除您建立並清理帳戶的所有 Amazon Lex 資源。如有需要，您可以使用 Lambda 主控台刪除本練習中使用的 Lambda 函數。

練習 2：建立自訂 Amazon Lex 機器人

在本練習中，您會使用 Amazon Lex 主控台建立訂購披薩 (`OrderPizzaBot`) 的自訂機器人。設定此機器人的流程如下：加入自訂意圖 (`OrderPizza`)、定義自訂槽類型，以及定義要完成比薩接單所需的槽 (比薩餅皮、尺寸等)。如需槽類型和槽的詳細資訊，請參閱Amazon Lex 運作方式。

主題

- 步驟 1：建立 Lambda 函數
- 步驟 2：建立機器人
- 步驟 3：建置及測試機器人
- 步驟 4 (選用)：清理

步驟 1：建立 Lambda 函數

首先，創建一個滿足比薩餅訂單的 Lambda 函數。在下一節中，您在所建立的 Amazon Lex 機器人中指定此函數。

若要建立 Lambda 函數

1. 請登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/lambda/ 的 AWS Lambda 主控台。
2. 選擇 Create function (建立函數)。

3. 在 Create function (建立函數) 頁面上，選擇 Author from scratch (從頭開始撰寫)。

   由於您是使用本練習提供的自訂程式碼建立 Lambda 函數，因此您需要選擇從頭開始撰寫該函數。

   請執行下列動作：

   a. 輸入名稱 (`PizzaOrderProcessor`)。

   b. 對於 Runtime (執行時間)，選擇最新版本的 Node.js。

   c. Role (角色) 選擇 Create new role from template(s) (從範本建立新角色)。

   d. 輸入新的角色名稱 (`PizzaOrderProcessorRole`)。

   e. 選擇 Create function (建立函數)。

4. 在函數頁面上，執行以下操作：

   在 Function code (函數程式碼) 區段，選擇 Edit code inline (編輯程式碼內嵌)，然後複製以下 Node.js 函數程式碼並將其貼至視窗中。

```
'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. 選擇 Save (儲存)。

使用範例事件資料測試 Lambda 函數

在主控台中，使用範例事件資料手動叫用 Lambda 函數來測試它。

測試 Lambda 函數：

1. 請登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/lambda/ 的 AWS Lambda 主控台。
2. 在 Lambda 函數頁面上，選擇 Lambda 函數 (`PizzaOrderProcessor`)。
3. 從函數頁面上的測試事件清單中，選擇 Configure test events (設定測試事件)。
4. 在 Configure test event (設定測試事件) 頁面上，執行以下操作：
   a. 選擇 Create new test event (建立新測試事件)。
   b. 在 Event name (事件名稱) 欄位內，輸入事件的名稱 (`PizzaOrderProcessorTest`)。
   c. 將下列 Amazon Lex 事件複製到視窗中。

```
{
  "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. 選擇 Create (建立)。

AWS Lambda 會建立測試，而您將返回函數頁面。選擇測試，然後 Lambda 執行您的 Lambda 函數

在結果方塊中，選擇 Details (詳細資訊)。主控台將在 Execution result (執行結果) 窗格內顯示以下輸出。

```
{
  "sessionAttributes": {},
  "dialogAction": {
    "type": "Close",
    "fulfillmentState": "Fulfilled",
    "message": {
      "contentType": "PlainText",
      "content": "Okay, I have ordered your large meat pizza on thin crust."
    }
}
```

後續步驟

步驟 2：建立機器人

步驟 2：建立機器人

在本步驟中，您將建立一個用於處理比薩訂單的機器人。

主題

- 建立機器人
- 建立意圖
- 建立槽類型
- 設定意圖
- 設定 機器人

建立機器人

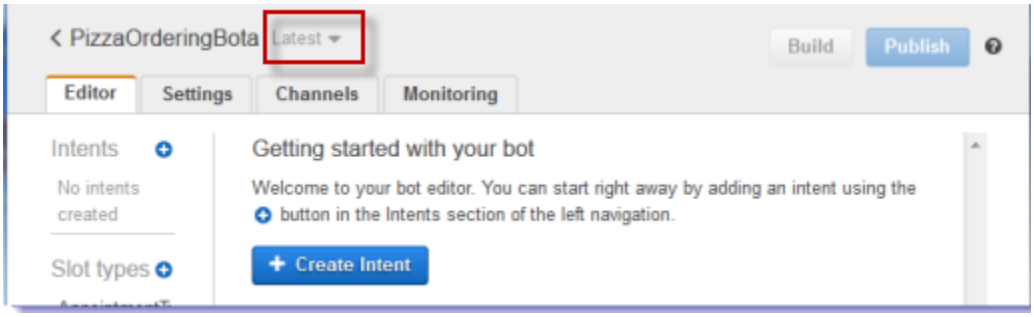
使用最少的必要資訊，建立 PizzaOrderingBot 機器人。稍後再為該機器人加入意圖，也就是使用者想要執行的動作。

建立機器人

1. 登入，AWS Management Console並開啟位於 https://console.aws.amazon.com/lex/ 的 Amazon Lex 主控台https://console.aws.amazon.com/lex/開啟 Amazon Lex 主控台。
2. 建立機器人。
   a. 如果這是您第一次建立機器人，請選擇 Get Started (開始使用)。否則，選擇 機器人，然後選擇 建立。
   b. 在建立您的機器人頁面上，選擇自訂機器人並提供以下資訊：
      - 機器人名稱： PizzaOrderingBot
      - 語言：選擇機器人的語言和地區設定。
      - 輸出語音：Salli
      - 工作階段逾時：5 分鐘
      - 杯：選擇適當的回應。
      - 使用者話語儲存：選擇適當的回應。

c. 選擇 Create (建立)。

主控台會傳送 Amazon Lex 的請求，以建立新的機器人。Amazon Lex 將機器人版本設置為$LATEST。建立機器人之後，Amazon Lex 會顯示機器人編輯器索引標籤，如下圖所示：



- 機器人版本 Latest (最新的) 在主控台內顯示於機器人名稱旁。新的 Amazon Lex 資源有$LATEST作為版本。如需詳細資訊，請參閱版本控制與別名。
- 由於您尚未建立任何意圖或槽類型，故未列出任何內容。
- Build (建置) 和 Publish (發佈) 是機器人層級的活動。在您設定妥整個機器人之後，將會更進一步了解這些活動。

後續步驟

建立意圖

建立意圖

現在，使用最少的必要資訊建立 OrderPizza 意圖，也就是使用者想要執行的動作。稍後再為該意圖加入槽類型，接著設定意圖。

建立意圖

1. 在 Amazon Lex 主控台中，選擇「意圖」旁邊的加號 (+)，然後選擇「建立新意圖」。
2. 在 Create intent (建立意圖) 對話方塊中，輸入意圖的名稱 (OrderPizza)，然後選擇 Add (加入)。

主控台會傳送要求至 Amazon Lex 以建立OrderPizza意圖。在這個範例中，您要先建立槽類型之後再建立槽意圖。

後續步驟

建立槽類型

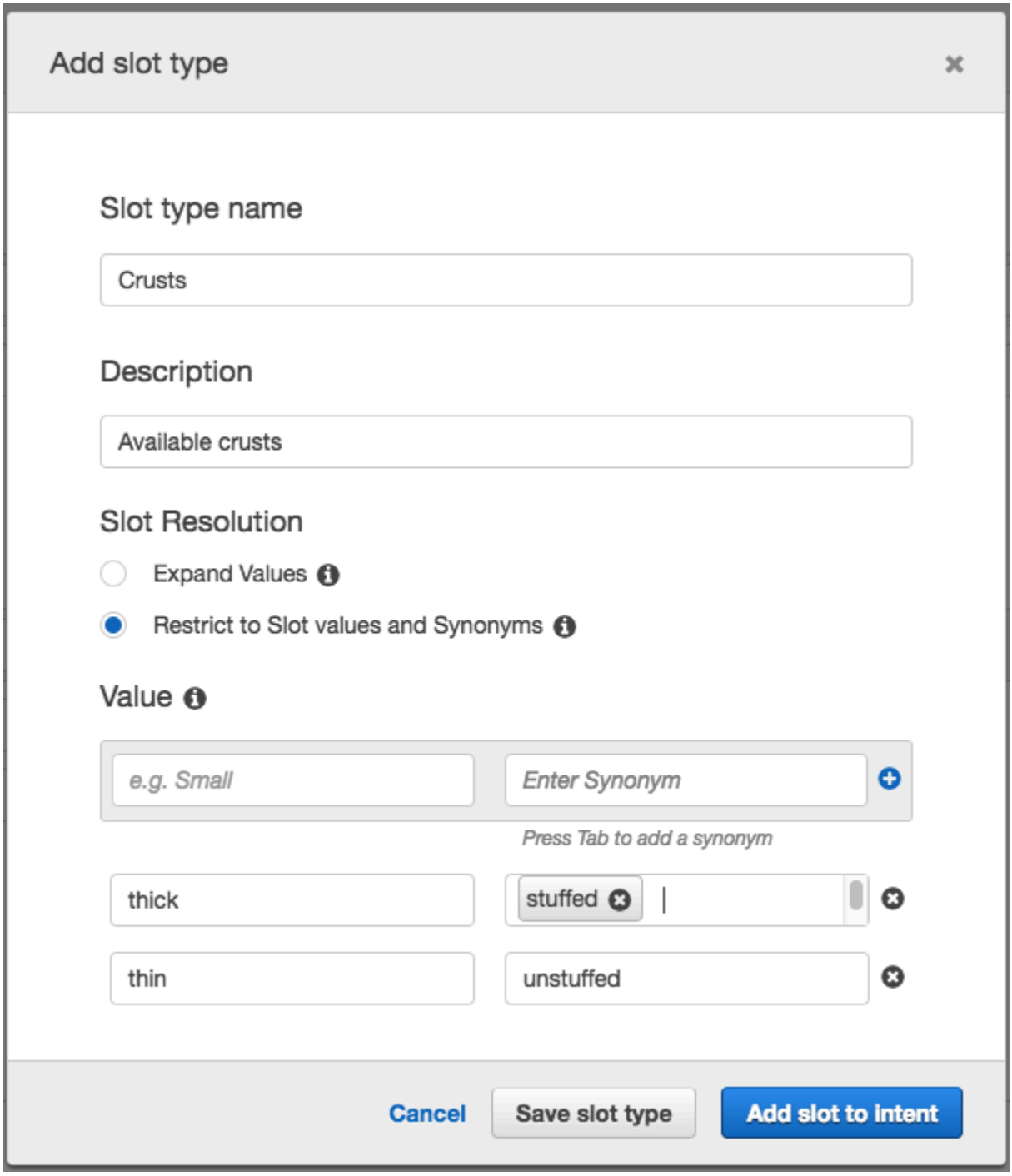
建立槽類型

建立 `OrderPizza` 意圖所使用的槽類型 (參數值)。

建立槽類型

1. 從左側選單中，選擇 Slot types (槽類型) 旁的加號 (+)。
2. 在 Add slot type (加入槽類型) 對話方塊中，加入以下資訊：
   - Slot type name (槽類型名稱) – Crusts
   - Description (說明) – 供應的餅皮
   - 選擇 Restrict to Slot values and Synonyms (限制為槽值和同義詞)
   - 值 — 類型**`thick`**。按下 Tab 鍵並在 Synonym (同義詞) 欄位內輸入 **`stuffed`**。選擇加號 (+)。輸入 **`thin`**，然後再次選擇加號 (+)。

   此對話方塊應如下列影像所示：



3. 選擇 Add slot to intent (加入槽至意圖)。
4. 在 Intent (意圖) 頁面上，選擇 Required (必要)。將槽的名稱由 **slotOne** 變更為 **crust**。切換至 **What kind of crust would you like?** 提示：
5. 使用下表的值重複Step 1 到Step 4：

名稱	描述	值	槽名稱	提示
Sizes	供應的尺寸	小型、中型、大型	size	要多大尺寸的比薩？
PizzaKind	供應的比薩	素食、起司	pizzaKind	您要素食比薩還是起司比薩？

後續步驟

設定意圖

設定意圖

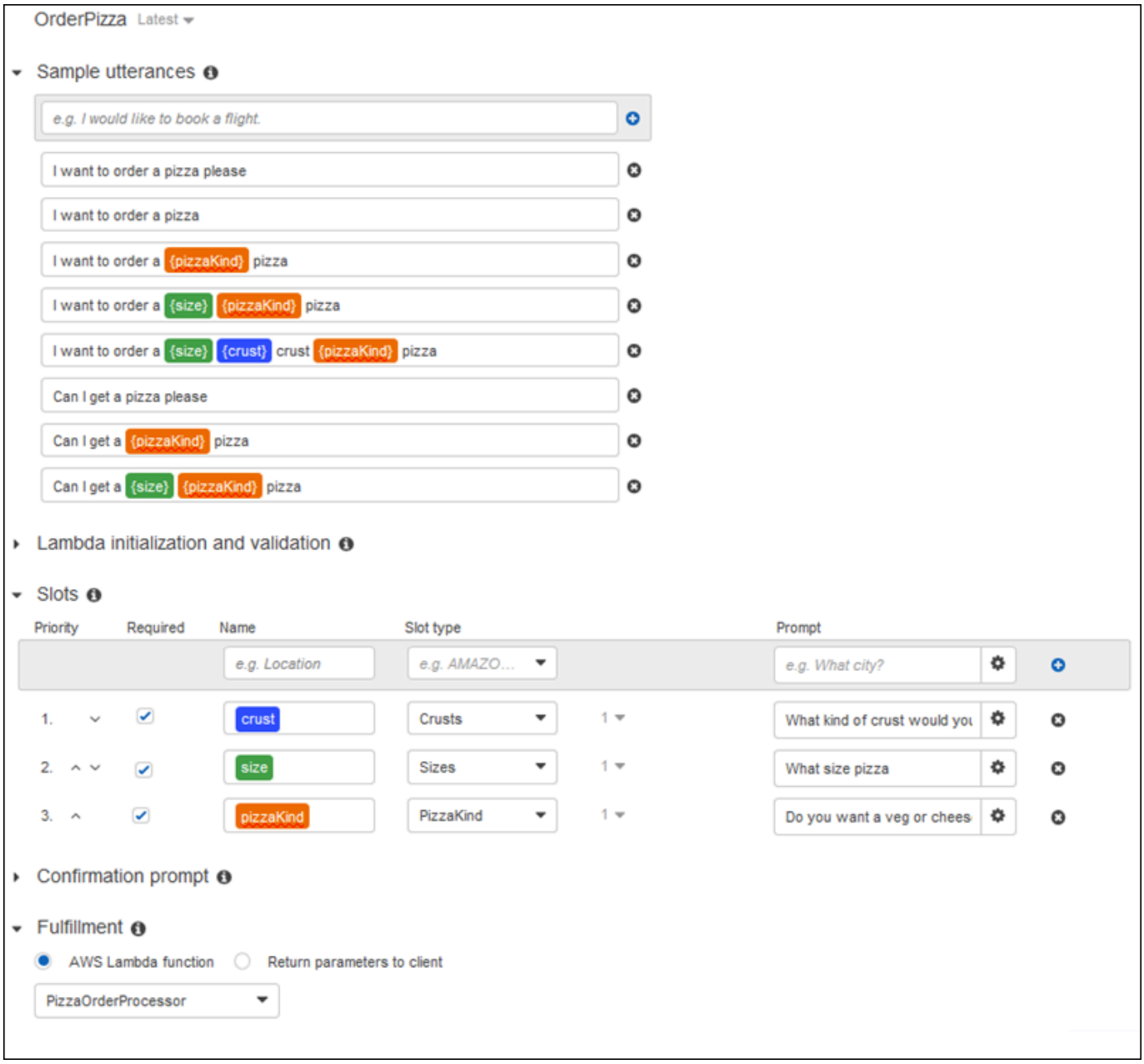
設定 OrderPizza 意圖以便完成使用者訂購比薩的接單工作。

設定意圖

- 在OrderPizza組態頁面上，依照下列方式設定意圖：
  - 範例語音 — 輸入下列字串。大括號 {} 內為槽名稱。
    - 我想要訂比薩
    - 我想要訂一個比薩
    - 我想要訂一個 {pizzaKind} 比薩
    - 我想要訂一個 {size} 尺寸的 {pizzaKind} 比薩
    - 我想要訂一個 {size} 尺寸的 {crust} 餅皮 {pizzaKind} 比薩
    - 我能否點一個比薩
    - 我能否點一個 {pizzaKind} 比薩
    - 我能否點一個 {size} 尺寸的 {pizzaKind} 比薩
  - Lambda initialization and validation (Lambda 初始化和驗證) – 保留預設值。
  - Confirmation prompt (確認提示) – 保留預設值。
  - 履行 — 執行下列任務：
    - 選擇 AWS Lambda 函數。
    - 選擇 **PizzaOrderProcessor**。

- 如果顯示 [新增至 Lambda 函數] 對話方塊，請選擇 [確定] 以授與呼叫`PizzaOrderProcessor` Lambda 函數的`OrderPizza`意圖權限。
- 保持 None (無) 的選取狀態。

意圖應如下所示：



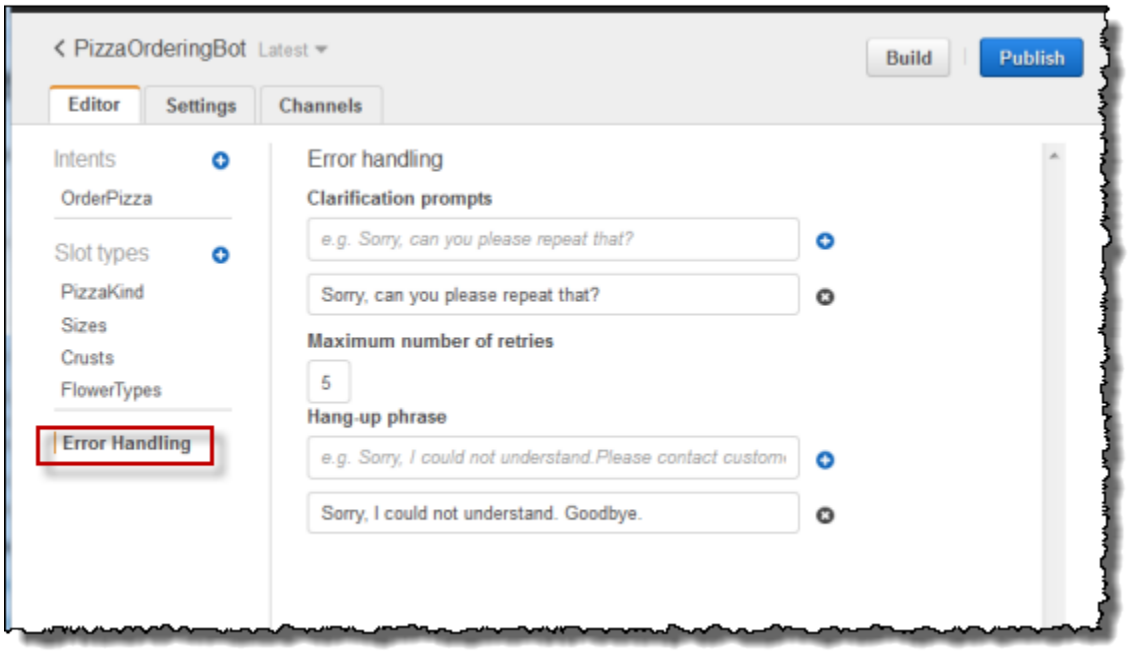
後續步驟

[設定 機器人]

設定 機器人

為 PizzaOrderingBot 機器人設定錯誤處理。

1. 前往 PizzaOrderingBot 機器人。選擇「編輯器」，然後選擇「錯誤處理」，如下圖所示：



2. 使用 Editor (編輯器) 索引標籤設定機器人錯誤處理。

   - 您在 Clarification Prompts (釐清提示) 中提供的資訊將對應到機器人的 clarificationPrompt 組態。

     當 Amazon Lex 無法判斷使用者意圖時，服務會傳回包含此訊息的回應。
   - 您在 Hang-up (擱置) 詞句中提供的資訊將對應到機器人的 abortStatement 組態。

     如果服務在連續多次請求後無法判斷使用者的意圖，Amazon Lex 會傳回含有此訊息的回應。

   保留預設值。

後續步驟

步驟 3：建置及測試機器人

步驟 3：建置及測試機器人

透過建置及測試機器人確保其正常運作。

建置及測試機器人

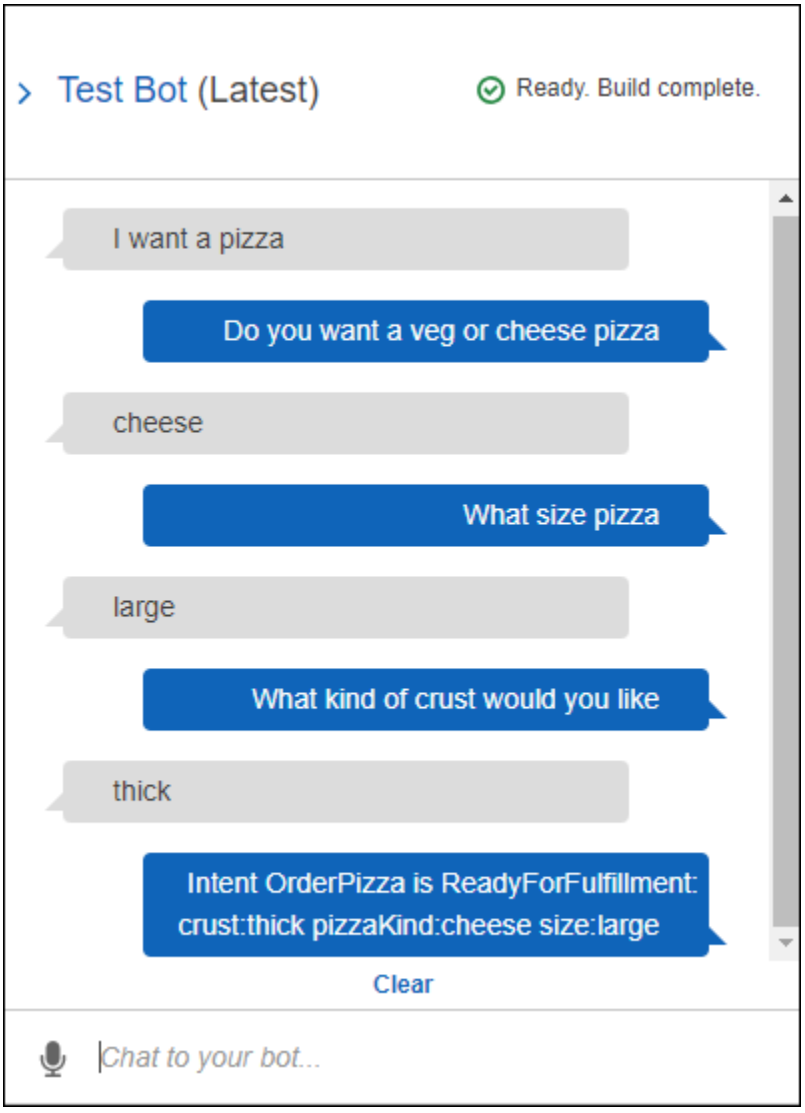
1. 要建置 PizzaOrderingBot 機器人，選擇 Build (建置)。

   Amazon Lex 為機器人建立機器人學習模型。當您測試機器人時，主控台會使用執行階段 API 將使用者輸入傳回 Amazon Lex。然後，Amazon Lex 會使用機器學習模型來解譯使用者輸入。

   完成建置可能需要一些時間。

2. 若要測試機器人，請在「測試機器人」視窗中開始與 Amazon Lex 機器人通訊。

   - 例如，您可能會說出或輸入下列內容：



   - 使用您在 OrderPizza 意圖中設定的範例表達用語來測試機器人。例如，以下是您為 PizzaOrder 意圖設定的一種範例表達用語：

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

要對其進行測試，請輸入如下內容：

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

當您輸入「我想訂購披薩」時，Amazon Lex 會偵測到意圖 (OrderPizza)。然後，Amazon Lex 要求提供插槽信息。

在您提供所有插槽資訊之後，Amazon Lex 會叫用您針對意圖設定的 Lambda 函數。

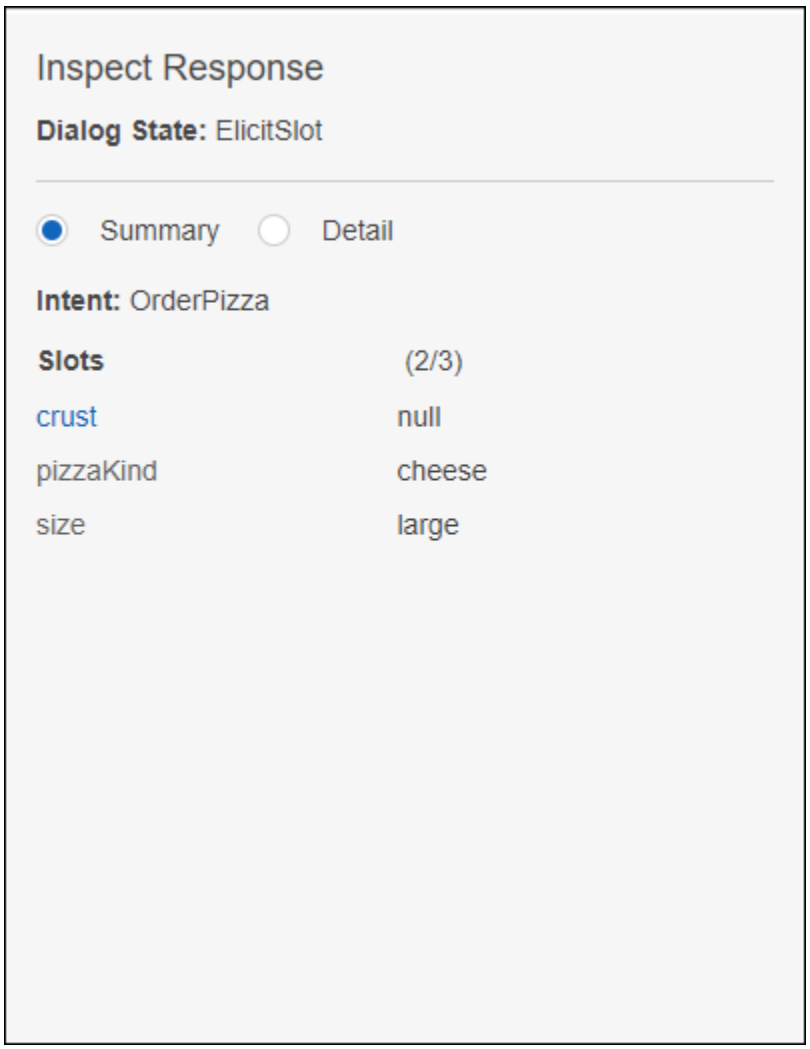
Lambda 函數返回一條消息 (「好吧，我已經訂購了你的...」) Amazon Lex，Amazon Lex 返回給你..

檢測回應

聊天視窗下方有一個窗格，可讓您檢查來自 Amazon Lex 的回應。該窗格提供有關機器人狀態的完整資訊，其內容會隨著您與機器人的互動而變化。窗格的內容會顯示操作的目前狀態。

- 對話狀態 — 與使用者交談的目前狀態。其可能是 ElicitIntent、ElicitSlot、ConfirmIntent 或 Fulfilled。

- 摘要 (Summary) — 顯示對話方塊的簡化檢視，其中會顯示要履行意圖的槽值，以便您可以追蹤資訊流程。其將顯示意圖名稱、槽數和已填充的槽數，以及所有各槽及其關聯值的清單。請參閱下圖：



- 詳細資訊 — 顯示聊天機器人的原始 JSON 回應，讓您在測試和偵錯聊天機器人時，更深入地了解機器人互動和對話方塊的目前狀態。如果您在聊天視窗中輸入，檢測窗格將顯示來自 PostText 操作的 JSON 回應。如果您在聊天視窗中說話，檢測窗格將顯示來自 PostContent 操作的回應標頭。請參閱下圖：

後續步驟

[步驟 4 (選用)：清理](#)

步驟 4 (選用)：清理

刪除您所建立的資源並清理您的帳戶，以免建立的資源產生更多費用。

您只能刪除未使用的資源。例如，您無法刪除由意圖所參照的槽類型。您無法刪除由機器人所參照的意圖。

按照以下順序刪除資源：

- 刪除機器人以釋放意圖資源。
- 刪除意圖以釋放槽類型資源。

- 最後刪除槽類型。

清理您的帳戶

1. 登入，AWS Management Console並開啟位於 https://console.aws.amazon.com/lex/ 的 Amazon Lex 主控台https://console.aws.amazon.com/lex/開啟 Amazon Lex 主控台。
2. 從機器人列表中選擇PizzaOrderingBot。
3. 要刪除該機器人，選擇刪除，然後選擇繼續。
4. 從左側窗格選擇 意圖。
5. 在意圖清單中，選擇OrderPizza。
6. 要刪除該意圖，選擇刪除，然後選擇 繼續)。
7. 從左側選單中選擇槽類型。
8. 從槽類型清單中選擇餅皮)。
9. 要刪除該槽類型，選擇刪除，然後選擇 繼續。
10. 重複Step 8 和Step 9 處置 `Sizes` 和 `PizzaKind` 槽類型。

您已移除了自己建立的所有資源並清理了您的帳戶。

後續步驟

- 發佈版本並建立別名
- 建立 Amazon Lex 機器人AWS Command Line Interface

練習 3：發佈版本和建立別名

您在入門練習 1 和 2 中已建立並已測試機器人。在本練習中，您會進行以下動作：

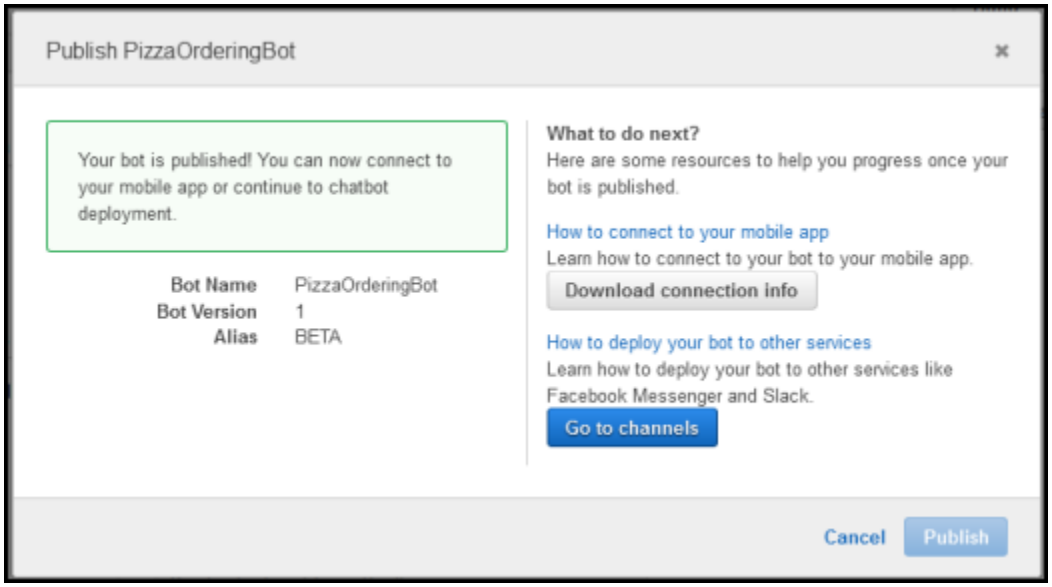
- 發佈機器人的新版本。Amazon Lex 會擷取該$LATEST版本的快照複本以發佈新版本。
- 建立指向新版本的別名。

如需有關版本控制與別名的詳細資訊，請參閱版本控制與別名。

執行以下步驟為您針對本練習所建立的機器人發佈版本：

1. 在 Amazon Lex 主控台中，選擇您建立的其中一個機器人。

   確認主控台在機器人名稱旁邊顯示 $LATEST 為機器人版本。
2. 選擇 Publish (發佈)。
3. 在 Publish (發佈) ***botname (###)*** 名稱精靈上，指定別名 **BETA**，然後選擇 Publish (發佈)。
4. 確認 Amazon Lex 主控台在機器人名稱旁顯示新版本，如下圖所示。



有了已發佈版本和別名的可行機器人後，就可以部署機器人了 (在您的行動應用程式或將機器人 Facebook Messenger 整合)。如需範例，請參閱 整合 Amazon Lex 機器人與臉書信使。

步驟 4：入門 (AWS CLI)

在此步驟中，您使用AWS CLI以建立、測試和修改 Amazon Lex 機器人。您必須熟悉使用 CLI 並且有文字編輯器來完成這些練習。如需詳細資訊，請參閱「」步驟 2：設定 AWS Command Line Interface

- 練習 1 — 建立和測試 Amazon Lex 機器人。本練習提供您建立自訂槽類型、意圖和機器人所需的所有 JSON 物件。如需詳細資訊，請參閱「」Amazon Lex 運作方式
- 練習 2 — 更新您在練習 1 中所建立的機器人來新增額外的範例表達用語。Amazon Lex 會使用範例表達用語為機器人建立機器學習模型。
- 練習 3 — 更新您在練習 1 中所建立的機器人來新增 Lambda 函數，以驗證使用者輸入並滿足意圖。
- 練習 4 — 發佈您在練習 1 中所建立之槽類型、意圖和機器人資源的版本。版本是無法變更之資源的快照。
- 練習 5 — 為您在練習 1 中所建立的機器人建立別名。

- 練習 6 — 透過刪除您在練習 1 忠所建立的槽類型、意圖和機器人，以及在練習 5 中所建立的別名，來清除您的帳戶。

主題

練習 1：建立 Amazon Lex 機器人 (AWS CLI)

一般而言，當建立機器人時，您會：

1. 建立槽類型來定義機器人要處理的資訊。
2. 建立意圖來定義機器人支援的使用者動作。使用您之前建立的自訂槽類型來定義您的意圖所需的槽或參數。
3. 建立機器人來使用您所定義的意圖。

在本練習中，您使用 CLI 建立和測試新的 Amazon Lex 機器人。請使用我們提供的 JSON 結構來建立機器人。若要執行本練習中的命令，您必須知道要執行命令的區域。如需區域的列表，請參閱 模型建置配額 。

主題

步驟 1：建立服務連結角色 (AWS CLI)

Amazon Lex 假設AWS Identity and Access Management要調用的服務連結角色AWS服務代表您的機器人。這些角色位於您的帳戶中，它們已連結到 Amazon Lex 使用案例並且具備預先定義的許可權。如需詳細資訊，請參閱 使用 Amazon Lex 的服務連結角色。

如果您已經使用主控台建立 Amazon Lex 機器人，則會自動建立服務連結的角色。跳至 步驟 2：建立自訂槽類型 (AWS CLI)。

建立服務連結角色 (AWS CLI)

1. 在 AWS CLI 中輸入行以下命令：

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

2. 使用以下命令來檢查政策：

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

回應為：

```
{
    "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"
}
```

後續步驟

步驟 2：建立自訂槽類型 (AWS CLI)

步驟 2：建立自訂槽類型 (AWS CLI)

以可訂購的花朵列舉值來建立自訂槽類型。您會在後續步驟中建立 `OrderFlowers` 意圖時使用此類型。槽類型會定義意圖的槽或參數可能的值。

若要執行本練習中的命令，您必須知道要執行命令的區域。如需區域的列表，請參閱 模型建置配額 。

建立自訂槽類型 (AWS CLI)

1. 建立名為 **`FlowerTypes.json`** 的文字檔案。從 FlowerTypes.json 複製 JSON 程式碼到文字檔案。
2. 使用 AWS CLI 呼叫 PutSlotType 操作來建立槽類型。此範例格式適用於 Unix、Linux 和 macOS。用於 Windows 時，請以插入號 (^) 取代每一行結尾處的 Unix 接續字元斜線 (\)。

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

伺服器的回應為：

```
{
    "enumerationValues": [
        {
            "value": "tulips"
        },
        {
            "value": "lilies"
        },
        {
            "value": "roses"
        }
    ],
    "name": "FlowerTypes",
    "checksum": "checksum",
    "version": "$LATEST",
    "lastUpdatedDate": timestamp,
    "createdDate": timestamp,
```

```
    "description": "Types of flowers to pick up"
}
```

後續步驟

步驟 3：建立意圖 (AWS CLI)

FlowerTypes.json

以下程式碼是建立 FlowerTypes 自訂槽類型所需的 JSON 資料：

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

步驟 3：建立意圖 (AWS CLI)

為 OrderFlowersBot 機器人建立意圖，並提供三個槽或參數。槽允許機器人滿足意圖：

- FlowerType 是自訂槽類型，會指定可以訂購的花種。
- AMAZON.DATE 和 AMAZON.TIME 是內建的槽類型，用於向使用者取得送花的日期和時間。

若要執行本練習中的命令，您必須知道要執行命令的區域。如需區域的列表，請參閱 模型建置配額 。

建立 **OrderFlowers** 意圖 (AWS CLI)

1. 建立名為 **OrderFlowers.json** 的文字檔案。從 OrderFlowers.json 複製 JSON 程式碼到文字檔案。

2. 在 AWS CLI 中呼叫 PutIntent 操作來建立意圖。此範例格式適用於 Unix、Linux 和 macOS。用於 Windows 時，請以插入號 (^) 取代每一行結尾處的 Unix 接續字元斜線 (\)。

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

伺服器會以下列內容回應：

```
{
    "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"
}
```

後續步驟

[步驟 4：建立機器人 (AWS CLI)]

OrderFlowers.json

以下程式碼是建立 `OrderFlowers` 意圖所需的的 JSON 資料：

```
{
    "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"
}
```

步驟 4：建立機器人 (AWS CLI)

OrderFlowersBot 機器人有一個意圖，即您在之前的步驟所建立的 OrderFlowers 意圖。若要執行本練習中的命令，您必須知道要執行命令的區域。如需區域的列表，請參閱 模型建置配額 。

> **Note**
>
> 以下 AWS CLI 範例格式適用於 Unix、Linux 和 macOS。用於 Windows 時，請將 "\$LATEST" 變更為 $LATEST。

建立 **OrderFlowersBot** 機器人 (AWS CLI)

1. 建立名為 **OrderFlowersBot.json** 的文字檔案。從 OrderFlowersBot.json 複製 JSON 程式碼到文字檔案。
2. 在 AWS CLI 中呼叫 PutBot 操作來建立機器人。此範例格式適用於 Unix、Linux 和 macOS。用於 Windows 時，請以插入號 (^) 取代每一行結尾處的 Unix 接續字元斜線 (\)。

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

伺服器會隨之回應。當建立或更新機器人時，status 欄位會設定為 BUILDING。這表示機器人尚未準備就緒。若要判斷機器人何時準備就緒，請使用下一步中的 GetBot 操作。

```
{
    "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. 若要判斷您的新機器人是否已準備就緒，請執行下列命令。重複此命令，直到 `status` 欄位傳回 READY 為止。此範例格式適用於 Unix、Linux 和 macOS。用於 Windows 時，請以插入號 (^) 取代每一行結尾處的 Unix 接續字元斜線 (\)。

```
aws lex-models get-bot \
    --region region \
    --name OrderFlowersBot \
    --version-or-alias "\$LATEST"
```

在回應中尋找 `status` 欄位：

```
{
    "status": "READY",

    ...

}
```

後續步驟

步驟 5：測試機器人 (AWS CLI)

OrderFlowersBot.json

以下程式碼提供建立`OrderFlowers`Amazon Lex 機器人：

```
{
    "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"
}
```

步驟 5：測試機器人 (AWS CLI)

若要測試機器人，您可以使用以文字或語音為基礎的測試。

主題

- 使用文字輸入測試機器人 (AWS CLI)
- 使用語音輸入測試機器人 (AWS CLI)

使用文字輸入測試機器人 (AWS CLI)

若要利用文字輸入確認機器人可正確運作，請使用 PostText 操作。若要執行本練習中的命令，您必須知道要執行命令的區域。如需區域的列表，請參閱執行時間服務配額。

> **Note**
>
> 以下 AWS CLI 範例格式適用於 Unix、Linux 和 macOS。用於 Windows 時，請將 "\$LATEST" 變更為 $LATEST，並以插入號 (^) 取代每一行結尾處的反斜線 (\) 接續字元。

使用文字來測試機器人 (AWS CLI)

1. 在 AWS CLI 中展開與 OrderFlowersBot 機器人的對話。此範例格式適用於 Unix、Linux 和 macOS。用於 Windows 時，請以插入號 (^) 取代每一行結尾處的 Unix 接續字元斜線 (\)。

```
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 會識別使用者的意圖，並傳回以下的回應以開始對話：

```
{
    "slotToElicit": "FlowerType",
    "slots": {
        "PickupDate": null,
        "PickupTime": null,
        "FlowerType": null
    },
    "dialogState": "ElicitSlot",
    "message": "What type of flowers would you like to order?",
    "intentName": "OrderFlowers"
}
```

2. 執行以下命令來完成與機器人的對談。

```
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"
```

在確認訂單後，Amazon Lex 會傳送履行回應以完成對話：

```
{
    "slots": {
        "PickupDate": "2017-05-16",
        "PickupTime": "10:00",
        "FlowerType": "roses"
    },
    "dialogState": "ReadyForFulfillment",
    "intentName": "OrderFlowers"
}
```

後續步驟

[使用語音輸入測試機器人 (AWS CLI)]

使用語音輸入測試機器人 (AWS CLI)

若要使用音訊檔案測試機器人，請使用 PostContent 操作。您使用 Amazon Polly 文字轉語音操作來產生音訊檔。

若要執行本練習中的命令，您必須知道要執行 Amazon Lex 和 Amazon Polly 命令的區域。如需 Amazon Lex 區域的列表，請參。執行時間服務配額。如需 Amazon Polly 區域的列表，請參AWS區域與端點中的Amazon Web Services 一般參考。

> **Note**
>
> 以下 AWS CLI 範例格式適用於 Unix、Linux 和 macOS。用於 Windows 時，請將 "\$LATEST" 變更為 $LATEST，並以插入號 (^) 取代每一行結尾處的反斜線 (\) 接續字元。

使用語音輸入測試機器人 (AWS CLI)

1. 在 中AWS CLI，使用 Amazon Polly 建立音訊檔案。此範例格式適用於 Unix、Linux 和 macOS。用於 Windows 時，請以插入號 (^) 取代每一行結尾處的 Unix 接續字元斜線 (\)。

   ```
   aws polly synthesize-speech \
       --region region \
       --output-format pcm \
       --text "i would like to order flowers" \
       --voice-id "Salli" \
       IntentSpeech.mpg
   ```

2. 若要傳送音訊檔案到 Amazon Lex，請執行以下命令。Amazon Lex 會將回應的音訊儲存在指定的輸出檔中。

   ```
   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 會以對第一個槽的請求進行回應。它會將音訊回應儲存在指定的輸出檔中。

   ```
   {
       "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. 若要訂玫瑰，請建立以下音訊檔案，並將它傳送到 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. 若要設定送花日期，請建立以下音訊檔案，並將它傳送到 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. 若要設定送花時間，請建立以下音訊檔案，並將它傳送到 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. 若要確認送達，請建立以下音訊檔案，並將它傳送到 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
```

在確認送達後，Amazon Lex 會傳送回應，以確認意圖的履行：

```
{
    "contentType": "text/plain;charset=utf-8",
    "dialogState": "ReadyForFulfillment",
    "intentName": "OrderFlowers",
```

```
    "inputTranscript": "yes",
    "slots": {
        "PickupDate": "2017-05-16",
        "PickupTime": "10:00",
        "FlowerType": "roses"
    }
}
```

後續步驟

練習 2：新增表達用語 (AWS CLI)

練習 2：新增表達用語 (AWS CLI)

為了改善 Amazon Lex 用來識別使用者之請求的機器學習模型，請新增另一個範例表達用語到機器人。

新增表達用語包含四個步驟。

1. 使用GetIntent操作從 Amazon Lex 取得意圖。
2. 更新意圖。
3. 使用PutIntent操作將更新的意圖傳回 Amazon Lex。
4. 使用 GetBot 和 PutBot 操作重建使用該意圖的任何機器人。

若要執行本練習中的命令，您必須知道要執行命令的區域。如需區域的列表，請參閱 模型建置配額 。

來自 GetIntent 操作的回應包含一個名為 checksum 的欄位，可識別意圖的特定修訂版本。使用 PutIntent 操作更新意圖時，您必須提供檢查總和值。若不提供，會得到以下錯誤訊息：

```
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**
>
> 以下 AWS CLI 範例格式適用於 Unix、Linux 和 macOS。用於 Windows 時，請將 "\$LATEST" 變更為 $LATEST，並以插入號 (^) 取代每一行結尾處的反斜線 (\) 接續字元。

更新 **OrderFlowers** 意圖 (AWS CLI)

1. 在 中AWS CLI，從 Amazon Lex 取得意圖。Amazon Lex 會將輸出傳送到一個稱為**OrderFlowers-V2.json.**

   ```
   aws lex-models get-intent \
       --region region \
       --name OrderFlowers \
       --intent-version "\$LATEST" > OrderFlowers-V2.json
   ```

2. 在文字編輯器中開啟 **OrderFlowers-V2.json**。

   1. 尋找並刪除 `createdDate`、`lastUpdatedDate` 和 `version` 欄位。
   2. 將以下內容新增到 `sampleUtterances` 欄位：

      ```
      I want to order flowers
      ```

   3. 儲存檔案。

3. 使用下列命令將更新的意圖傳送到 Amazon Lex：

   ```
   aws lex-models put-intent  \
       --region region \
       --name OrderFlowers \
       --cli-input-json file://OrderFlowers-V2.json
   ```

   Amazon Lex 會傳送以下回應：

   ```
   {
       "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"
}
```

更新好意圖之後，重建任何使用它的機器人。

重建 **OrderFlowersBot** 機器人 (AWS CLI)

1. 在 AWS CLI 中取得 OrderFlowersBot 機器人的定義，並使用下列命令將它儲存到檔案：

```
aws lex-models get-bot \
    --region region \
    --name OrderFlowersBot \
    --version-or-alias "\$LATEST" > OrderFlowersBot-V2.json
```

2. 在文字編輯器中開啟 **OrderFlowersBot-V2.json**。移除 createdDate、lastUpdatedDate、status 和 version 欄位。
3. 在文字編輯器中，將下列行新增至機器人定義：

```
"processBehavior": "BUILD",
```

4. 在 AWS CLI 中，透過執行下列命令來建立機器人的新修訂版：

```
aws lex-models put-bot \
    --region region \
    --name OrderFlowersBot \
    --cli-input-json file://OrderFlowersBot-V2.json
```

伺服器的回應為：

```
{
    "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"
}
```

後續步驟

練習 3：新增 Lambda 函數 (AWS CLI)

練習 3：新增 Lambda 函數 (AWS CLI)

新增 Lambda 函數來驗證使用者輸入並滿足使用者對機器人的意圖。

新增 Lambda 表達式包含五個步驟。

1. 使用 LambdaAddPermission函數來啟用`OrderFlowers`意圖調用 Lambda呼叫operation.
2. 使用GetIntent操作從 Amazon Lex 取得意圖。
3. 更新意圖以新增 Lambda 函數。
4. 使用PutIntent操作將更新的意圖傳回 Amazon Lex。
5. 使用 GetBot 和 PutBot 操作重建使用該意圖的任何機器人。

若要執行本練習中的命令，您必須知道要執行命令的區域。如需區域的列表，請參閱 模型建置配額 。

如 Lambda 您在新增`InvokeFunction`權限，您會得到以下錯誤訊息：

```
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.
```

來自 GetIntent 操作的回應包含一個名為 checksum 的欄位，可識別意圖的特定修訂版本。使用 PutIntent 操作更新意圖時，您必須提供檢查總和值。若不提供，會得到以下錯誤訊息：

```
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.
```

本練習使用 Lambda 函數從練習 1：使用藍圖 (主控台) 建立 Amazon Lex 機器人。如需有關建立 Lambda 函數的指示，請參步驟 3：建立 Lambda 函數。

> **Note**
>
> 以下 AWS CLI 範例格式適用於 Unix、Linux 和 macOS。用於 Windows 時，請將 "\$LATEST" 變更為 $LATEST。

新增 Lambda 函數到意圖

1. 在 AWS CLI 中，為 OrderFlowers 意圖新增 InvokeFunction 許可：

   ```
   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 會傳送以下回應：

```
{
    "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. 從 Amazon Lex 取得意圖。Amazon Lex 會將輸出傳送到一個稱為**OrderFlowers-V3.json**。

```
aws lex-models get-intent \
    --region region \
    --name OrderFlowers \
    --intent-version "\$LATEST" > OrderFlowers-V3.json
```

3. 在文字編輯器中，開啟 **OrderFlowers-V3.json**。

   1. 尋找並刪除 `createdDate`、`lastUpdatedDate` 和 `version` 欄位。
   2. 更新 `fulfillmentActivity` 欄位：

      ```
      "fulfillmentActivity": {
          "type": "CodeHook",
          "codeHook": {
              "uri": "arn:aws:lambda:region:account
      ID:function:OrderFlowersCodeHook",
              "messageVersion": "1.0"
          }
      }
      ```

   3. 儲存檔案。

4. 在 中AWS CLI，會將更新的意圖傳送到 Amazon Lex：

```
aws lex-models put-intent \
    --region region \
    --name OrderFlowers \
    --cli-input-json file://OrderFlowers-V3.json
```

更新好意圖之後，重建機器人。

重建 OrderFlowersBot 機器人

1. 在 AWS CLI 中取得 OrderFlowersBot 機器人的定義，並將它儲存到檔案：

```
aws lex-models get-bot \
    --region region \
    --name OrderFlowersBot \
    --version-or-alias "\$LATEST" > OrderFlowersBot-V3.json
```

2. 在文字編輯器中開啟 **OrderFlowersBot-V3.json**。移除 createdDate、lastUpdatedDate、status 和 version 欄位。
3. 在文字編輯器中，將下列行新增至機器人的定義：

```
"processBehavior": "BUILD",
```

4. 在 AWS CLI 中，建立機器人的新修訂版本：

```
aws lex-models put-bot \
    --region region \
    --name OrderFlowersBot \
    --cli-input-json file://OrderFlowersBot-V3.json
```

伺服器的回應為：

```
{
    "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"
}
```

後續步驟

練習 4：發佈版本 (AWS CLI)

練習 4：發佈版本 (AWS CLI)

現在為您在練習 1 中建立的機器人建立一個版本。 版本是機器人的的快照。版本建立後，便無法變更。您唯一可以更新的版本是 $LATEST 版本。如需有關版本的詳細資訊，請參閱版本控制與別名。

您必須先發佈機器人所用的意圖後，才能發佈機器人的版本。同樣地，您必須發佈該些意圖所參考的槽類型。一般而言，發佈機器人的版本需執行下列動作：

1. 使用 CreateSlotTypeVersion 操作發佈槽類型的版本。
2. 使用 CreateIntentVersion 操作發佈意圖的版本。
3. 使用 CreateBotVersion 操作發佈機器人的版本。

若要執行本練習中的命令，您必須知道要執行命令的區域。如需區域的列表，請參閱 模型建置配額 。

主題

- 步驟 1：發佈槽類型 (AWS CLI)
- 步驟 2：發佈意圖 (AWS CLI)
- 步驟 3：發佈機器人 (AWS CLI)

步驟 1：發佈槽類型 (AWS CLI)

您必須先發佈槽類型的版本之後，才能發佈使用槽類型之任何意圖的版本。在這個範例中，您發佈 `FlowerTypes` 槽類型。

> **Note**
>
> 以下 AWS CLI 範例格式適用於 Unix、Linux 和 macOS。用於 Windows 時，請將 "\$LATEST" 變更為 $LATEST，並以插入號 (^) 取代每一行結尾處的反斜線 (\) 接續字元。

發佈槽類型 (AWS CLI)

1. 在 AWS CLI 中，取得最新版本的槽類型：

```
aws lex-models get-slot-type \
    --region region \
    --name FlowerTypes \
    --slot-type-version "\$LATEST"
```

Amazon Lex 會隨之回應。記錄 $LATEST 版本目前修訂版的檢查總和。

```
{
    "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. 發佈槽類型的版本。使用您在前一步驟中記錄的檢查總和。

```
aws lex-models create-slot-type-version \
    --region region \
    --name FlowerTypes \
    --checksum "checksum"
```

Amazon Lex 會隨之回應。記錄版本編號以用於下個步驟。

```
{
    "version": "1",
    "enumerationValues": [
        {
            "value": "tulips"
        },
        {
            "value": "lilies"
        },
        {
            "value": "roses"
        }
    ],
    "name": "FlowerTypes",
    "createdDate": timestamp,
    "lastUpdatedDate": timestamp,
    "description": "Types of flowers to pick up"
}
```

後續步驟

步驟 2：發佈意圖 (AWS CLI)

步驟 2：發佈意圖 (AWS CLI)

您必須先發佈意圖所參考的所有槽類型之後，才能發佈意圖。槽類型必須是已編號的版本，而非 $LATEST 版本。

首先，更新 OrderFlowers 意圖以使用您在前面的步驟中發佈的 FlowerTypes 槽類型版本。然後，發佈 OrderFlowers 意圖的新版本。

> **Note**
>
> 以下 AWS CLI 範例格式適用於 Unix、Linux 和 macOS。用於 Windows 時，請將 "\$LATEST" 變更為 $LATEST，並以插入號 (^) 取代每一行結尾處的反斜線 (\) 接續字元。

發佈意圖的版本 (AWS CLI)

1. 在 AWS CLI 中取得 OrderFlowers 意圖的 $LATEST 版本，並將它儲存到檔案：

```
aws lex-models get-intent \
    --region region \
    --name OrderFlowers \
    --intent-version "\$LATEST" > OrderFlowers_V4.json
```

2. 在文字編輯器中，開啟 **OrderFlowers_V4.json** 檔案。刪除 createdDate、lastUpdatedDate 和 version 欄位。找出 FlowerTypes 槽類型並將版本變更為您在前面的步驟中記錄的版本編號。以下 **OrderFlowers_V4.json** 檔案片段顯示變更的位置：

```
        {
            "slotType": "FlowerTypes",
            "name": "FlowerType",
            "slotConstraint": "Required",
            "valueElicitationPrompt": {
                "maxAttempts": 2,
                "messages": [
                    {
                        "content": "What type of flowers?",
                        "contentType": "PlainText"
                    }
                ]
            },
            "priority": 1,
            "slotTypeVersion": "version",
            "sampleUtterances": []
        },
```

3. 在 AWS CLI 中儲存意圖的修訂版：

```
aws lex-models put-intent \
    --name OrderFlowers \
    --cli-input-json file://OrderFlowers_V4.json
```

4. 取得意圖最新修訂版的檢查總和：

```
aws lex-models get-intent \
    --region region \
    --name OrderFlowers \
    --intent-version "\$LATEST" > OrderFlowers_V4a.json
```

以下回應片段顯示意圖的的檢查總和。請將此記錄下來以用於下個步驟。

```
    "name": "OrderFlowers",
    "checksum": "checksum",
    "version": "$LATEST",
```

5. 發佈意圖的新版本：

```
aws lex-models create-intent-version \
    --region region \
    --name OrderFlowers \
    --checksum "checksum"
```

以下回應片段顯示意圖的新版本。記錄版本編號以用於下個步驟。

```
    "name": "OrderFlowers",
    "checksum": "checksum",
    "version": "version",
```

後續步驟

步驟 3：發佈機器人 (AWS CLI)

步驟 3：發佈機器人 (AWS CLI)

在發佈機器人使用的所有槽類型和意圖之後，您就可以發佈機器人了。

更新 OrderFlowersBot 機器人以使用您在前面的步驟中所更新的 OrderFlowers 意圖。然後，發佈 OrderFlowersBot 機器人的新版本。

> **Note**
>
> 以下 AWS CLI 範例格式適用於 Unix、Linux 和 macOS。用於 Windows 時，請將 "\$LATEST" 變更為 $LATEST，並以插入號 (^) 取代每一行結尾處的反斜線 (\) 接續字元。

發佈機器人的版本 (AWS CLI)

1. 在 AWS CLI 中取得 OrderFlowersBot 機器人的 $LATEST 版本，並將它儲存到檔案：

```
aws lex-models get-bot \
    --region region \
    --name OrderFlowersBot \
    --version-or-alias "\$LATEST" > OrderFlowersBot_V4.json
```

2. 在文字編輯器中，開啟 **OrderFlowersBot_V4.json** 檔案。刪除 createdDate、lastUpdatedDate、status 和 version 欄位。找出 OrderFlowers 意圖並將版本變更為您在前面的步驟中記錄的版本編號。以下 **OrderFlowersBot_V4.json** 片段顯示變更的位置。

```
    "intents": [
        {
            "intentVersion": "version",
            "intentName": "OrderFlowers"
        }
```

3. 在 AWS CLI 中，儲存機器人的新版本。記下呼叫 put-bot 傳回的版本號碼。

```
aws lex-models put-bot \
    --name OrderFlowersBot \
    --cli-input-json file://OrderFlowersBot_V4.json
```

4. 取得機器人最新修訂版的檢查總和。使用步驟 3 傳回的版本號碼。

```
aws lex-models get-bot \
    --region region \
    --version-or-alias version \
    --name OrderFlowersBot > OrderFlowersBot_V4a.json
```

以下回應片段顯示機器人的檢查總和。請將此記錄下來以用於下個步驟。

```
    "name": "OrderFlowersBot",
    "locale": "en-US",
    "checksum": "checksum",
```

5. 發佈機器人的新版本：

```
aws lex-models create-bot-version \
    --region region \
    --name OrderFlowersBot \
    --checksum "checksum"
```

以下回應片段顯示機器人的新版本。

```
    "checksum": "checksum",
    "abortStatement": {
        ...
    },
    "version": "1",
    "lastUpdatedDate": timestamp,
```

後續步驟

練習 5：建立別名 (AWS CLI)

練習 5：建立別名 (AWS CLI)

別名是特定機器人版本的指標。透過別名，您可以輕鬆地更新用戶端應用程式所使用的版本。如需更多詳細資訊，請參閱版本控制與別名。若要執行本練習中的命令，您必須知道要執行命令的區域。如需區域的列表，請參閱 模型建置配額 。

建立別名 (AWS CLI)

1. 在 AWS CLI 中，取得您在 練習 4：發佈版本 (AWS CLI) 中建立的 OrderFlowersBot 機器人版本。

```
aws lex-models get-bot \
    --region region \
    --name OrderFlowersBot \
```

```
    --version-or-alias version > OrderFlowersBot_V5.json
```

2. 在文字編輯器中開啟 **OrderFlowersBot_v5.json**。尋找並記錄版本編號。
3. 在 AWS CLI 中建立機器人別名：

```
aws lex-models put-bot-alias  \
    --region region \
    --name PROD \
    --bot-name OrderFlowersBot \
    --bot-version version
```

以下是伺服器的回應：

```
{
    "name": "PROD",
    "createdDate": timestamp,
    "checksum": "checksum",
    "lastUpdatedDate": timestamp,
    "botName": "OrderFlowersBot",
    "botVersion": "1"
}}
```

後續步驟

練習 6：清除 (AWS CLI)

練習 6：清除 (AWS CLI)

刪除您建立的資源並清除您的帳戶。

您只能刪除未使用的資源。一般而言，您應該依下列順序刪除資源。

1. 刪除別名以釋放機器人資源。
2. 刪除機器人以釋放意圖資源。
3. 刪除意圖以釋放槽類型資源。
4. 刪除槽類型。

若要執行本練習中的命令，您必須知道要執行命令的區域。如需區域的列表，請參閱 模型建置配額 。

清除您的帳戶 (AWS CLI)

1. 在 AWS CLI 命令列中刪除別名：

```
aws lex-models delete-bot-alias \
    --region region \
    --name PROD \
    --bot-name OrderFlowersBot
```

2. 在 AWS CLI 命令列中刪除機器人：

```
aws lex-models delete-bot \
    --region region \
    --name OrderFlowersBot
```

3. 在 AWS CLI 命令列中刪除意圖：

```
aws lex-models delete-intent \
    --region region \
    --name OrderFlowers
```

4. 從 AWS CLI 命令列刪除槽類型：

```
aws lex-models delete-slot-type \
    --region region \
    --name FlowerTypes
```

您已移除了自己建立的所有資源並清理了您的帳戶。

版本控制與別名

Amazon Lex 支援發佈機器人、意圖和槽類型的版本，以讓您能控制用户端應用程式所使用的實作。版本是您的工作具有編號的快照，可供您發佈以用於工作流程的各個不同環節，例如開發、測試部署和生產。

Amazon Lex 機器人也支援別名。別名是特定機器人版本的指標。透過別名，您可以輕鬆地更新用戶端應用程式所使用的版本。例如，您可以將別名指向版本 1 的機器人。當您準備好要更新機器人時，即可發佈版本 2 並將別名變更為指向新的版本。由於您的應用程式是使用別名而非特定版本，所有您的用戶端皆無需進行更新便能獲得新功能。

主題

版本控制

當您對 Amazon Lex 資源進行版本控制時，您將建立該資源的快照，進而能夠以建立其版本時的資源現存狀態使用該資源。版本建立後，在您繼續處理應用程式期間該資源將保持不變。

$LATEST 版本

當您建立 Amazon Lex 機器人、意圖或槽類型時，只會有一個版本，即$LATEST版本。

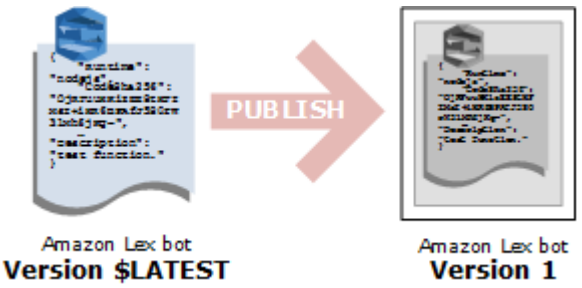
$LATEST 是您的資源的工作複本。您只能更新 $LATEST 版本，而且直到您發佈第一個版本之前，$LATEST 是您所擁有資源的唯一版本。

唯獨 $LATEST 版本的資源能夠使用 $LATEST 版本的另一項資源。例如，$LATEST 版本的機器人可使用 $LATEST 版本的意圖，而 $LATEST 版本的意圖可使用 $LATEST 版本的槽類型。

所以此$LATEST機器人的版本僅應用於手動測試。Amazon Lex 會限制您可以對$LATEST版本的機器人。

發佈亞 Amazon Lex 資源版本

當您發佈資源時，Amazon Lex 會建立$LATEST版本並將其儲存為具有編號的版本。發佈的版本無法變更。



您可以使用 Amazon Lex 主控台或CreateBotVersionoperation. 如需範例，請參閱 練習 3：發佈版本和建立別名。

當您修改 $LATEST 版本的資源後，可發佈新版本以就您的用戶端應用程式進行任何適用的變更。每次發佈版本時，Amazon Lex 都會複製$LATEST版本以建立新版本，並將版本編號遞增 1。版本編號絕不會重複使用。例如，若您移除了版本編號為 10 的資源後又重新建立該資源，則 Amazon Lex 為其指派的下一個版本編號將是版本 11。

發佈機器人之前，您必須先將該機器人指向其所使用之某一編號版本的任何意圖。若您嘗試發佈的新版本機器人是使用 $LATEST 版本的意圖，Amazon Lex 將會傳回 HTTP 400 錯誤的請求例外狀況。發佈某一編號版本的意圖之前，您必須先將該意圖指向其所使用之某一編號版本的任何槽類型。否則，您將收到 HTTP 400 錯誤的請求例外狀況。

> **Note**
>
> 僅當上次發佈的版本不同於$LATEST版本。如果您嘗試發佈$LATEST版本而未對其進行修改，Amazon Lex 便不會建立或發佈新版本。

更新亞 Amazon Lex 資源

您可以僅更新$LATEST版本的 Amazon Lex 機器人、意圖或槽類型。發佈的版本無法變更。更新資源之後，您隨時可透過主控台或使用 CreateBotVersion、CreateIntentVersion 或 CreateSlotTypeVersion 操作以發佈新版本。

刪除 Amazon Lex 資源或版本

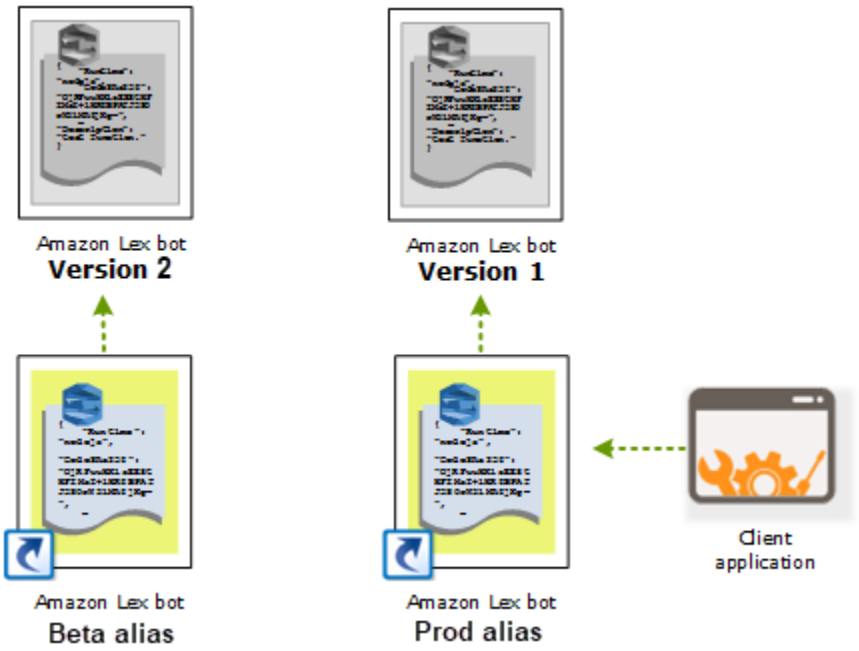
Amazon Lex 支援使用主控台或以下 API 操作之一刪除資源或版本：

- DeleteBot
- DeleteBotVersion
- DeleteBotAlias
- DeleteBotChannelAssociation
- DeleteIntent
- DeleteIntentVersion
- DeleteSlotType
- DeleteSlotTypeVersion

別名

別名是特定版本 Amazon Lex 機器人的指標。利用別名以讓用戶端應用程式能夠使用特定版本的機器人，而無需由應用程式追蹤其為哪個版本。

以下範例顯示了兩個版本的 Amazon Lex 機器人，版本 1 和版本 2。這兩個機器人版本各有其相關聯的別名，分別為 BETA 和 PROD。用戶端應用程式使用 PROD 別名存取機器人。



建立另一版本的機器人之後，您可以使用主控台或 PutBot 操作，將別名更新為指向新版本的機器人。一旦您變更別名，所有您的用戶端應用程式都將使用新版本。如果新版本發生問題，您只需要將別名變更為指向前一個版本即可還原回該版本。

> **Note**
>
> 儘管您可以從主控台測試 $LATEST 版本的機器人，但建議您在將機器人與您的用戶端應用程式整合時，首先發佈一個版本並建立別名以指向該版本。如存在本節所述原因，請在您的用

戶端應用程式中使用別名。當您更新別名時，Amazon Lex 將等到所有目前工作階段的工作階段逾時皆已過期後，才開始使用新版本。如需工作階段逾時的詳細資訊，請參閱the section called “設定工作階段逾時”

使用 Lambda 函數

您可以建立AWS Lambda函數，用作為 Amazon Lex 機器人的程式碼掛勾。您可以識別 Lambda 函數在您的意圖組態中執行初始化和驗證、履行或兩者。

我們建議您使用 Lambda 函數做為機器人的程式碼掛勾。如果沒有 Lambda 函數，您的機器人將意圖資訊傳回給用户端應用程式以履行。

主題

Lambda 函數輸入事件和回應格式

本節說明了 Amazon Lex 提供給 Lambda 函數的事件資料結構。請使用此資訊來剖析 Lambda 程式碼內的輸入內容。文中亦說明了 Amazon Lex 預期您的 Lambda 函數傳回的回應格式。

主題

輸入事件格式

以下顯示傳遞給 Lambda 函數的 Amazon Lex 事件的一般格式。請在寫入 Lambda 函數時使用此資訊。

> **Note**
>
> 即使 `messageVersion` 中沒有對應的變更，輸入格式也可能變更。如果出現新欄位，您的程式碼不應擲出錯誤。

```
{
  "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"
            }
        }
    ]
}
```

請注意以下有關事件欄位的更多資訊：

- currentIntent – 提供意圖 name、slots slotDetails 和 confirmationStatus 欄位。

  nluIntentConfidenceScore是 Amazon Lex 對當前意圖最符合用户當前意圖的信心。

  slots是槽名稱 (為意圖所設定) 與 Amazon Lex 在使用者對話中識別出的槽值的對應。槽值會保持 null，直到使用者提供值為止。

  輸入事件中的槽值可能不符合為槽所設定的其中一個值。例如，如果使用者對提示「您想要什麼顏色的車？」 和「比薩」，」Amazon Lex 將傳回「比薩」做為槽值。您的函數應該驗證值，以確保內容符合邏輯。

  slotDetails 會提供有關槽值的更多資訊。resolutions 陣列包含為槽所識別的其他值清單。每個槽最多可有 5 個值。

originalValue 欄位包含使用者為槽輸入的值。當槽類型設定為傳回最常用的解析值來做為槽值時，originalValue 可能與 slots 欄位中的值不同。

confirmationStatus 提供對確認提示 (如果有的話) 的使用者回應。例如，如果 Amazon Lex 詢問「您是否要訂購大型起士比薩？」，根據使用者的回應，此欄位的值可以是Confirmed或者Denied。否則，此欄位的值會式 None。

如果使用者確認意圖、Amazon Lex 會將此欄位設定為Confirmed。如果使用者拒絕意圖、Amazon Lex 會將此值設定為Denied。

在確認回應中，使用者表達用語可能會提供槽更新。例如，使用者可能說「是，大小改為中型。」在這種情況下，後續 Lambda 事件會有已更新的槽值PizzaSize設定為medium。Amazon Lex 設定confirmationStatus至None，因為使用者修改了一些槽資料，而需要使用 Lambda 函數執行使用者資料驗證。

- 替代用途— 如果啟用置信度分數，Amazon Lex 將返回最多四個備選方法。每個意圖都包含一個分數，表明 Amazon Lex 對意圖是基於用户言論的正確意圖的信心程度。

  備選意圖的內容與currentIntent欄位。如需詳細資訊，請參閱 使用可信度分數。

- bot – 處理請求之機器人的相關資訊。
  - name – 處理請求的機器人名稱。
  - alias – 處理請求的機器人版本別名。
  - version – 處理請求的機器人版本。

- userId— 此值是由用户端應用程式提供。Amazon Lex 傳遞給 Lambda 函數。

- inputTranscript – 用來處理請求的文字。

  如果輸入是文字，`inputTranscript` 欄位會包含使用者輸入的文字。

  如果輸入是音訊串流，`inputTranscript` 欄位會包含從音訊串流擷取的文字。這是實際處理以識別意圖和槽值的文字。

- invocationSource— 指明為什 Amazon Lex 叫 Lambda 函數，它會將此設定為下列其中一個值：
  - `DialogCodeHook`— Amazon Lex 會設定此值以引導 Lambda 函數來初始化函數和驗證使用者的資料輸入。

    當將意圖設定為呼叫 Lambda 函數做為初始化和驗證程式碼掛勾時，在了解意圖後，Amazon Lex 會呼叫每個使用者輸入 (表達用語) 上指定的 Lambda 函數。

    > **Note**
    >
    > 如果意圖不明確、Amazon Lex 便無法呼叫 Lambda 函數。

  - `FulfillmentCodeHook`— Amazon Lex 設定此值以引導 Lambda 函數來滿足意圖。

    如果將意圖設定為呼叫 Lambda 函數做為履行程式碼掛勾，Amazon Lex 會設定`invocationSource`只有在具有所有槽資料以滿足意圖後才會傳回此值。

  在您的意圖組態中，可以有兩個單獨的 Lambda 函數來初始化和驗證使用者資料，以及滿足意圖。您也可以使用一個 Lambda 函數來執行兩者。在這種情況下，Lambda 函數可以使用`invocationSource`值以遵循正確的程式碼路徑。

- outputDialogMode— 對於每個使用者輸入，用户端會使用 Amazon Lex 中一個執行時間 API 操作，PostContent或者PostText。Amazon Lex 使用請求參數來判斷回應用户端的是文字還是語音，並相應設定此欄位。

  Lambda 函數可以使用此資訊來產生適當的訊息。例如，如果用户端預期語音回應，Lambda 函數可以傳回語音合成標記語言 (SSML)，而非文字。

- messageVersion— 事件的訊息，可識別進入 Lambda 函數的格式以及從 Lambda 函數的回應預期的格式。

  > **Note**
  >
  > 您在定義意圖時設定此值。在目前實作中，僅支援訊息版本 1.0。因此，主控台假設 1.0 的預設值，而且不會顯示訊息的版本。

- sessionAttributes— 用户端在請求中傳送的應用程式特定的工作階段屬性。如果您希望 Amazon Lex 將其包含在對用户端的回應中，您的 Lambda 函數應該在回應中把這些傳回 Amazon Lex。如需詳細資訊，請參閱「」設定工作階段屬性

- requestAttributes— 用户端在請求中傳送的請求特定的工作階段屬性。使用請求屬性來傳遞不需要在整個工作階段內保留的資訊。若無請求屬性，則數值將為 null。如需詳細資訊，請參閱「」設定請求屬性

- recentIntentSummaryView— 關於意圖狀態的資訊。您可以查看最近使用之三個意圖的相關資訊。您可以使用此資訊來設定意圖中的值，或返回先前的意圖。如需詳細資訊，請參閱 使用 Amazon Lex API 管理工作階段。

- sentimentResponse— Amazon Comprehend 情緒分析最後表達用語的結果。您可以根據使用者表達的情緒，使用此資訊來管理機器人的對話流程。如需詳細資訊，請參閱 情緒分析。

- kendraResponse— 查詢 Amazon Kendra 索引的結果。只會出現在履行程式碼掛勾的輸入中，以及只有在意圖延伸 AMAZON.KendraSearchIntent 內建意圖時才會出現。欄位包含來自 Amazon Kendra 搜尋的整個回應。如需詳細資訊，請參閱 AMAZON.KendraSearchIntent。

- 活動文本— 一個或多個在與用戶進行對話期間處於活動狀態的上下文。
  - TimeToLive— 上下文保持活動狀態的用戶對話中的時間長度或轉彎次數。
  - 名稱— 上下文的名稱。
  - 參數鍵/值對的列表包含激活上下文的意圖中插槽的名稱和值。

  如需詳細資訊，請參閱 設定意圖上下文。

回應格式

Amazon Lex 預期 Lambda 函數使用以下列格式回應：

```
{
  "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.
  }
}
```

回應包含四個欄位。所以此sessionAttributes、recentIntentSummaryView，和activeContexts欄位是選用的，dialogAction欄位是必要的。dialogAction 欄位的內容取決於 type 欄位的值。如需詳細資訊，請參閱 dialogAction。

sessionAttributes

選用。如果您包含 sessionAttributes 欄位，它可以留空。如果 Lambda 函數沒有傳回工作階段屬性，則最後已知sessionAttributes通過 API 或 Lambda 函數傳遞。如需詳細資訊，請參閱 PostContent 和 PostText 操作。

```
  "sessionAttributes": {
     "key1": "value1",
     "key2": "value2"
  }
```

recentIntentSummaryView

選用。如果包含，則設定一個或更多最近的意圖的值。您最多可以包含三個意圖的資訊。例如，您可以根據目前意圖所蒐集的資訊設定先前意圖的值。摘要中的資訊必須適用於該意圖。例如，意圖名稱必須是機器人中的意圖。如果要在摘要檢視中包含槽值，該槽必須存在於意圖中。如果沒有在回應中包含 recentIntentSummaryView，最近意圖的所有值仍會保持不變。如需詳細資訊，請參閱 PutSession 操作或 IntentSummary 資料類型。

```
"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"
    }
 ]
```

活動文本

選用。如果包含，則會設定一個或多個上下文的值。例如，您可以包含一個上下文，以使一個或多個意圖具有該上下文作為輸入符合對話的下一個回閤中的識別條件。

響應中未包含的任何活動上下文都會減少其生存時間值，並且在下一個請求中仍然處於活動狀態。

如果您為輸入事件中包含的上下文指定的生存時間為 0，則該上下一個請求將處於非活動狀態。

如需詳細資訊，請參閱 設定意圖上下文。

dialogAction

必要。所以此`dialogAction`欄位會引導 Amazon Lex 執行後續動作，並說明在 Amazon Lex 傳回回回回應給用户端之後對使用者的期望。

`type` 欄位會指出後續動作，它也會判斷 Lambda 函數需要在`dialogAction`值。

- `Close`— 通知 Amazon Lex 不預期使用者回應。例如，「您訂購的比薩已下單」不需要回應。

  `fulfillmentState` 欄位是必要的。Amazon Lex 使用此值來設置`dialogState`欄位PostContent或者PostText回應用程式。`message` 和 `responseCard` 欄位是選用的。如果您不指定訊息，Amazon Lex 會使用針對意圖設定的再見訊息或後續追蹤訊息。

```
"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— 通知 Amazon Lex 預期使用者給予是或否的回答，以確認或拒絕目前的意圖。

  您必須包含 intentName 與 slots 欄位。針對指定的意圖所填入的每個槽，slots 欄位都必須包含項目。您不需要為未填入的槽 slots 欄位中包含項目。 如果意圖的 message 欄位為空值，您必須為納入 confirmationPrompt 欄位。的內容message欄 Lambda 會優先於confirmationPrompt在意圖中指定。此 responseCard 欄位為選用。

```
"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— 引導 Amazon Lex 根據機器人組態選擇後續動作。如果回應不包含任何工作階段屬性，Amazon Lex 會保留現有的屬性。如果您希望槽值為空，您就不需在請求中包含槽欄位。如果您的履行函數沒有移除任何槽就傳回 DependencyFailedException 對話方塊動作，您將收到 Delegate 例外狀況。

  kendraQueryRequestPayload 和 kendraQueryFilterString 欄位是選用的，只有當意圖是從 AMAZON.KendraSearchIntent 內建意圖衍生而來時才會使用。如需詳細資訊，請參閱AMAZON.KendraSearchIntent。

```
 "dialogAction": {
  "type": "Delegate",
  "slots": {
```

```
      "slot-name": "value",
      "slot-name": "value",
      "slot-name": "value"
    },
    "kendraQueryRequestPayload": "Amazon Kendra query",
    "kendraQueryFilterString": "Amazon Kendra attribute filters"
  }
```

- `ElicitIntent`— 通知 Amazon Lex 預期使用者以包含意圖的表達用語回應。例如，「我想要大型比薩。」，亦即指出 `OrderPizzaIntent`。「大型」的表達用語在另一方面，並不足以讓 Amazon Lex 推斷使用者的意圖。

  `message` 和 `responseCard` 欄位是選用的。如果您未提供訊息，Amazon Lex 會使用其中一個機器人的澄清提示。如果未定義釐清提示，Amazon Lex 會傳回 400 錯誤請求例外狀況。

```
{
  "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`— 通知 Amazon Lex 預期使用者在回應中提供槽值。

  `intentName`、`slotToElicit` 和 `slots` 欄位是必要的。`message` 和 `responseCard` 欄位是選用的。如果您不指定訊息，Amazon Lex 會使用為槽設定的其中一個槽引出提示。

```
 "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 和AWS LambdaBlueprints (藍圖)

Amazon Lex 主控台提供已預先設定的範例機器人 (稱為機器人藍圖)，讓您可以快速在主控台中建立和測試機器人。對於其中的每個機器人藍圖，同時會提供 Lambda 函數藍圖。這些藍圖提供的範本程式碼可與其與對應的機器人搭配運作。您可以使用這些藍圖，快速建立使用 Lambda 函數做為程式碼掛勾所設定的機器人，並測試端對端設定，而無需撰寫程式碼。

您可以使用以下 Amazon Lex 機器人藍圖和對應的AWS Lambda函數藍圖做為機器人的程式碼掛勾：

- Amazon Lex 藍圖 —`OrderFlowers`
  - AWS Lambda藍圖 —`lex-order-flowers-python`
- Amazon Lex 藍圖 —`ScheduleAppointment`
  - AWS Lambda藍圖 —`lex-make-appointment-python`
- Amazon Lex 藍圖 —`BookTrip`
  - AWS Lambda藍圖 —`lex-book-trip-python`

若要使用藍圖建立機器人，並設定它使用 Lambda 函數做為程式碼掛勾，請參考練習 1：使用藍圖 (主控台) 建立 Amazon Lex 機器人。如需使用其他藍圖的範例，請參閱其他示例：創建亞馬遜 Lex 機器人。

更新特定區域設置的藍圖

如果您在英語 (US) (en-US) 以外的語言環境中使用藍圖，則需要更新任何意圖的名稱以包含該區域設置。例如，如果您使用`OrderFlowers`藍圖，您需要執行下列作業。

- 尋找`dispatch`函數在 Lambda 函數代碼末尾附近。
- 在 中`dispatch`函數中，更新意圖的名稱以包含您正在使用的區域設置。例如，如果您使用的是英語 (澳大利亞) (en-AU) 區域設置，請更改以下行：

  ```
  if intent_name == 'OrderFlowers':
  ```

  至

  ```
  if intent_name == 'OrderFlowers_enAU':
  ```

其他藍圖使用其他意圖名稱，在使用它們之前，應按照上述方式進行更新。

部署 Amazon Lex 機器人

本節提供將 Amazon Lex 機器人部署在各種簡訊平台及行動應用程式的範例。

主題

在簡訊平台上部署 Amazon Lex 機器人

本節說明如何將 Amazon Lex 機器人部署在 Facebook、Slack 和 Twilio 簡訊平台上。

> **Note**
>
> 當存放 Facebook、Slack 或 Twilio 組態時，Amazon Lex 會使用AWS Key Management Service客户管理的密鑰來加密信息。您第一次建立管道到其中一個簡訊平台時，Amazon Lex 會建立預設的客服管密鑰 (aws/lex。您也可以使用AWS KMS。這可給予您更多彈性，包括能夠建立、輪換和停用金鑰。您也可以定義存取控制並稽核用來保護資料的加密金鑰。如需詳細資訊，請參閱《AWS Key Management Service 開發人員指南》。

當簡訊平台傳送請求給 Amazon Lex 時，會包含平台特定的資訊，做為 Lambda 函數的請求屬性。請使用這些屬性來自訂機器人的行為。如需詳細資訊，請參閱 設定請求屬性。

所有屬性都會使用 `x-amz-lex:` 命名空間做為字首。例如，`user-id` 屬性稱為 `x-amz-lex:user-id`。除了特定平台專用的屬性外，還有所有簡訊平台傳送的常見屬性。下表列出簡訊平台傳送到機器人 Lambda 函數的請求屬性。

常見的請求屬性

屬性	Description (描述)
`channel-id`	來自 Amazon Lex 的管道端點識別符。
`channel-name`	來自 Amazon Lex 的管道名稱。
`channel-type`	下列其中一值：

屬性	Description (描述)
	• Facebook • Kik • Slack • Twilio-SMS
webhook-endpoint-url	管道的 Amazon Lex 端點.

Facebook 請求屬性

屬性	Description (描述)
user-id	傳送者的 Facebook 識別符。請參閱 https://developers.facebook.com/docs/messenger-platform/webhook-reference/message-received。
facebook-page-id	接收者的 Facebook 網頁識別符。請參閱 https://developers.facebook.com/docs/messenger-platform/webhook-reference/message-received。

Kik 請求屬性

屬性	Description (描述)
kik-chat-id	有您的機器人加入的對話所使用的識別碼。如需詳細資訊，請參閱 https://dev.kik.com/#/docs/messaging#message-formats。
kik-chat-type	該訊息來源的對話種類。如需詳細資訊，請參閱 https://dev.kik.com/#/docs/messaging#message-formats。
kik-message-id	識別訊息的 UUID。如需詳細資訊，請參閱 https://dev.kik.com/#/docs/messaging#message-formats。
kik-message-type	訊息的類型。如需詳細資訊，請參閱 https://dev.kik.com/#/docs/messaging#message-types。

Twilio 請求屬性

屬性	Description (描述)
`user-id`	傳送者的電話號碼 (「寄件者」)。請參閱 https://www.twilio.com/docs/api/rest/message。
`twilio-target-phone-number`	接收者的電話號碼 (「收件人」)。請參閱 https://www.twilio.com/docs/api/rest/message。

Slack 請求屬性

屬性	Description (描述)
`user-id`	Slack 使用者識別符。請參閱 https://api.slack.com/types/user。
`slack-team-id`	傳送訊息之團隊的識別符。請參閱 https://api.slack.com/methods/team.info。
`slack-bot-token`	提供機器人 Slack API 存取權的機器人符記。請參閱 https://api.slack.com/docs/token-types。

整合 Amazon Lex 機器人與臉書信使

這個練習演示了如何 Facebook 信使與您的 Amazon Lex 機器人集成。您會執行以下步驟：

1. 建立 Amazon Lex 機器人
2. 建立 Facebook 應用程式
3. 集成 Facebook 信使與您的 Amazon Lex 機器人
4. 驗證整合

主題

步驟 1：建立 Amazon Lex 機器人

如果您還沒有 Amazon Lex 機器人，則需要建立一個並部署一個。在本主題中，我們假設您使用的是在入門練習 1 中建立的機器人。不過，您可以使用本指南中提供的任何範例機器人。如需入門練習 1，請參閱練習 1：使用藍圖 (主控台) 建立 Amazon Lex 機器人。

1. 建立 Amazon Lex 機器人。如需相關指示，請參閱練習 1：使用藍圖 (主控台) 建立 Amazon Lex 機器人。
2. 部署機器人並建立別名。如需相關指示，請參閱練習 3：發佈版本和建立別名。

步驟 2：建立 Facebook 應用程式

在 Facebook 開發人員入口網站上，建立 Facebook 應用程式和 Facebook 粉絲專頁。如需相關指示，請參閱 Facebook Messenger 平台文件的快速入門。記下以下資訊：

- Facebook 應用程式的應用程式密鑰
- Facebook 粉絲專頁的粉絲專頁存取權杖

第 3 步：集成 Facebook 信使與 Amazon Lex 機器人

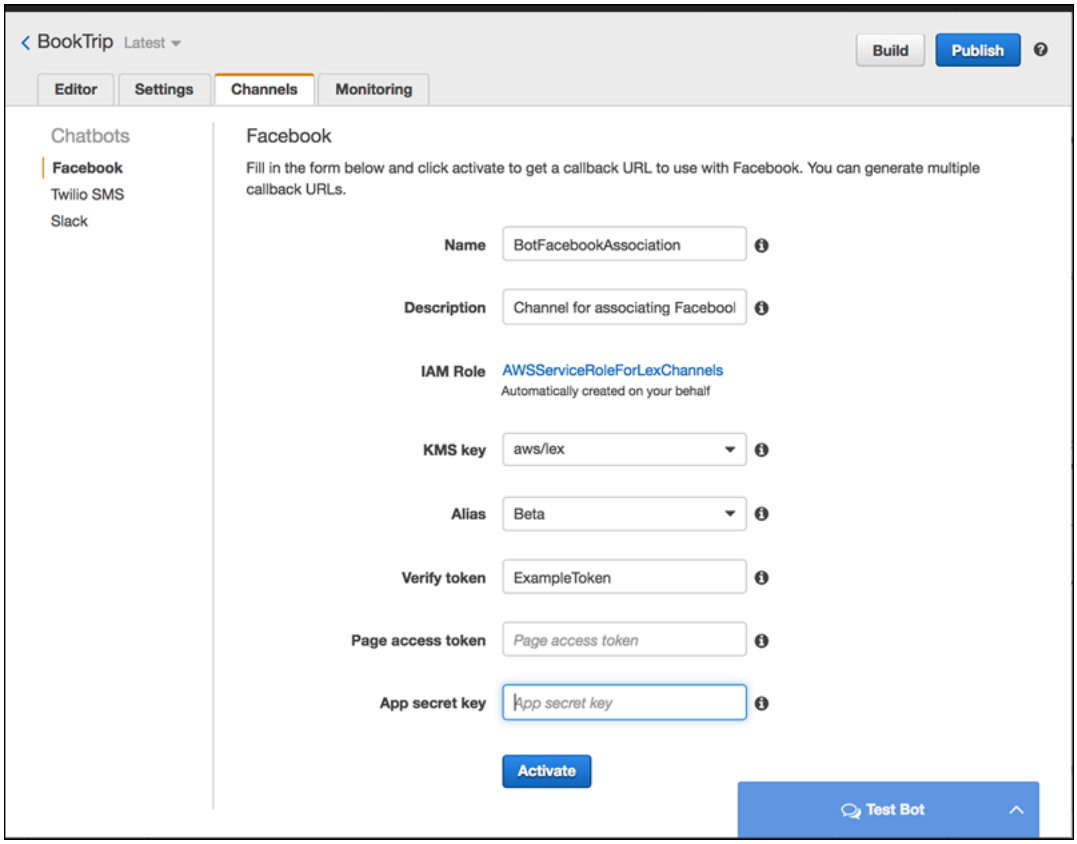
在本節中，您將 Facebook 信使與您的 Amazon Lex 機器人集成。

完成此步驟後，主控台將提供回呼 URL。記下該 URL。

將 Facebook Messenger 與您的機器人整合

1. a. 登入，AWS Management Console並在 https://console.aws.amazon.com/lex/ 開啟 Amazon Lex 主控台。
   b. 選擇您的 Amazon Lex 機器人。
   c. 選擇 Channels (管道)。
   d. 從 Chatbots (聊天機器人) 下方選擇 Facebook。主控台隨即顯示 Facebook 整合頁面。
   e. 在 Facebook 整合頁面上，執行以下操作：
      - 輸入以下名稱：`BotFacebookAssociation`。
      - 對於 KMS key (KMS 金鑰)，選擇 aws/lex。
      - 對於 Alias (別名)，選擇機器人別名。

- 對於 Verify token (驗證權杖)，輸入任意權杖。此權杖可以是您自選的任何字串 (例如 `ExampleToken`)。稍後在 Facebook 開發人員入口網站上設定 Webhook 時將會用到此權杖。
- 對於 Page access token (粉絲專頁存取權杖)，輸入您在步驟 2 從 Facebook 取得的權杖。
- 對於 App secret key (應用程式密鑰)，輸入您在步驟 2 從 Facebook 取得的密鑰。



f. 選擇 Activate (啟用)。

主控台隨即建立機器人管道關聯並傳回回呼 URL。記下該 URL。

2. 在 Facebook 開發人員入口網站上，選擇您的應用程式。
3. 選擇 Messenger 產品，然後從頁面的 Webhooks 區段選擇設定 Webhooks。

如需相關指示，請參閱 Facebook Messenger 平台文件的快速入門。

4. 在訂閱精靈的 Webhook 頁面上，執行以下操作：

- 對於回呼 URL，請輸入程序稍早在 Amazon Lex 主控台中提供的回呼 URL。
- 對於驗證權杖，請輸入您在 Amazon Lex 中使用的相同權杖。

- 選擇訂閱欄位 (messages、messaging_postbacks 和 messaging_optins)。
- 選擇驗證並儲存。這啟動臉書和 Amazon Lex 之間的握手。

5. 啟用 Webhook 整合。選擇您所建立的粉絲專頁，然後選擇訂閱。

> **Note**
>
> 如果您更新或重新建立了 Webhook，請先取消訂閱該粉絲專頁後再重新訂閱。

步驟 4：測試整合

現在，您可以從 Facebook 信使與您的 Amazon Lex 機器人開始對話。

1. 開啟您的 Facebook 粉絲專頁，然後選擇收件匣訊息。
2. 在 Messenger 視窗中，使用步驟 1：建立 Amazon Lex 機器人 所提供測試用的同一組表達用語。

整合亞 Amazon Lex 機器人與 Kik

本練習提供將 Amazon Lex 機器人與 Kik 簡訊應用程式整合的指示。您會執行以下步驟：

1. 建立 Amazon Lex 機器人。
2. 使用 Kik 應用程式和網站建立 Kik 機器人。
3. 使用 Amazon Lex 主控台將您的 Amazon Lex 機器人與 Kik 機器人整合在一起。
4. 使用 Kik 與 Amazon Lex 機器人進行對話，以測試 Amazon Lex 機器人與 Kik 之間的關聯性。

主題

步驟 1：建立 Amazon Lex 機器人

如果您還沒有 Amazon Lex 機器人，則建立一個機器人。在本主題中，我們假設您使用的是在入門練習 1 中建立的機器人。不過，您可以使用本指南中提供的任何範例機器人。如需入門練習 1，請參閱練習 1：使用藍圖 (主控台) 建立 Amazon Lex 機器人

1. 建立 Amazon Lex 機器人。如需相關指示，請參閱練習 1：使用藍圖 (主控台) 建立 Amazon Lex 機器人。
2. 部署機器人並建立別名。如需相關指示，請參閱練習 3：發佈版本和建立別名。

後續步驟

步驟 2：建立 Kik 機器人

步驟 2：建立 Kik 機器人

在此步驟中，您使用 Kik 使用者界面來建立 Kik 機器人。您可以使用建立機器人時產生的資訊，將其連接到 Amazon Lex 機器人。

1. 請下載並安裝 Kik 的應用程式，然後註冊 Kik 帳戶 (如果您尚未這麼做的話)。如果您已經有帳戶，請登入。
2. 開啟 Kik 網站： https://dev.kik.com/。將瀏覽器視窗保持開啟。
3. 在 Kik 應用程式中，選擇齒輪圖示以開啟設定，然後選擇 Your Kik Code (您的 Kik 程式碼)。
4. 在 Kik 網站上掃描 Kik 程式碼以開啟 Botsworth 聊天機器人。選擇 Yes (是) 以開啟機器人儀表板。
5. 在 Kik 應用程式中，選擇 Create a Bot (建立機器人)。依照提示建立 Kik 機器人。
6. 機器人一旦建立之後，在瀏覽器中選擇 Configuration (組態)。確定已選取您的新機器人。
7. 請記下機器人名稱和 API 金鑰以用於下一節。

後續步驟

步驟 3：將 Kik 機器人與 Amazon Lex 機器人集成

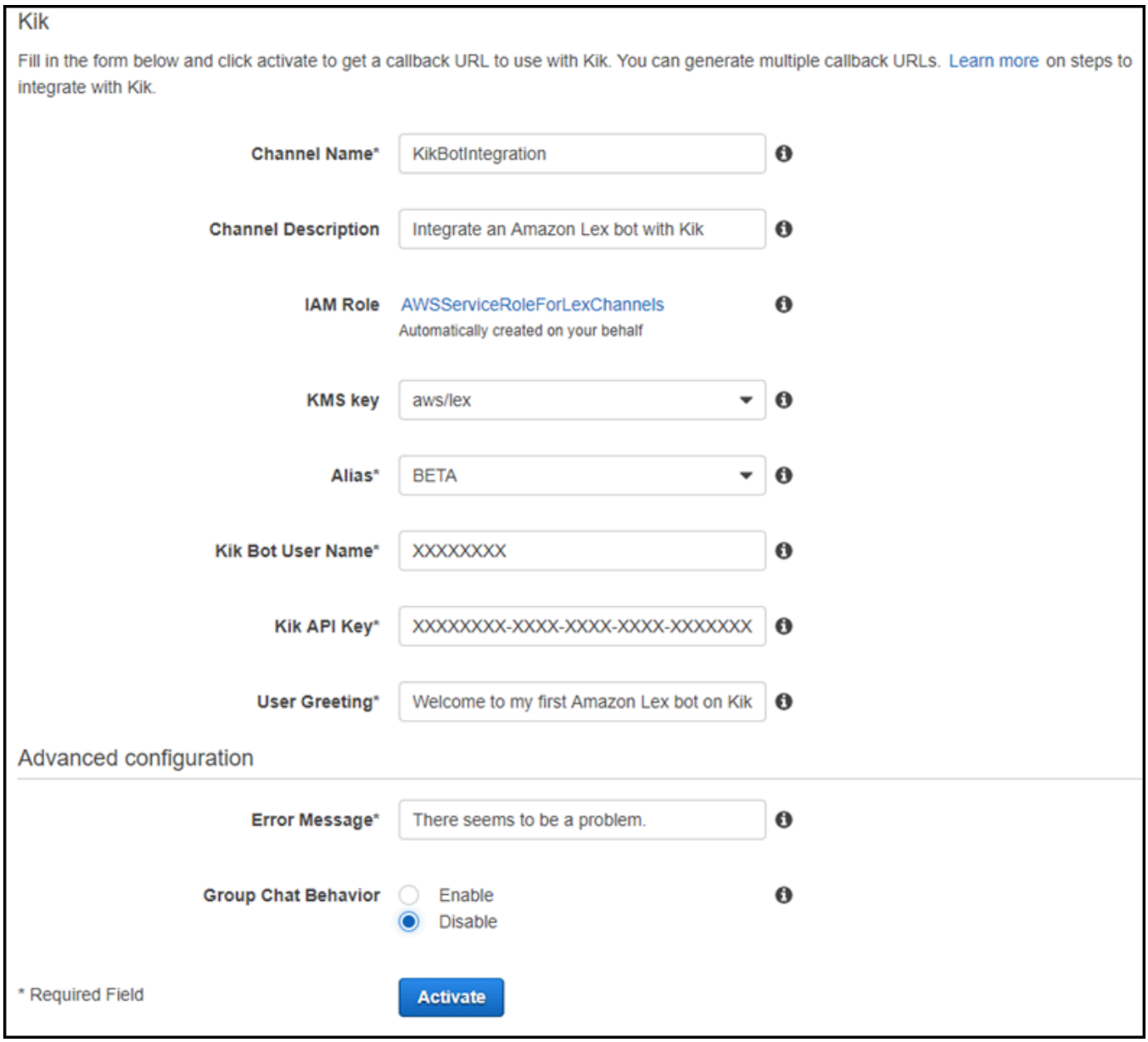
步驟 3：將 Kik 機器人與 Amazon Lex 機器人集成

既然您已經建立了 Amazon Lex 機器人和 Kik 機器人，您就可以在 Amazon Lex 中建立它們之間的通道關聯。當關聯被激活，Amazon Lex 自動與 Kik 設置一個回調網址。

1. 登入 AWS 管理主控台，開啟位於 https://console.aws.amazon.com/lex/ 的 Amazon Lex 主控台。
2. 選擇您在步驟 1 中建立 Amazon Lex 機器人。
3. 選擇 Channels (管道) 索引標籤。
4. 在 Channels (管道) 區段中，選擇 Kik。
5. 在 Kik 頁面上，提供以下資訊：

   - 輸入名稱。例如：`BotKikIntegration`。
   - 輸入描述。
   - 從 KMS key (KMS 金鑰)下拉式清單中選擇「aws/lex」。
   - 對於 Alias (別名)，從下拉式清單選擇別名。
   - 對於 Kik bot user name (Kik 機器人使用者名稱)，輸入您在 Kik 上為機器人取的名稱。
   - 對於 Kik API key (Kik API 金鑰)，輸入在 Kik 上指定給機器人的 API 金鑰。
   - 對於 User greeting (使用者問候)，輸入您希望機器人在使用者第一次與它聊天時傳送的問候語。
   - 對於 Error message (錯誤訊息)，輸入在不了解部分對談時顯示給使用者的錯誤訊息。
   - 對於 Group chat behavior (群組聊天行為)，選擇其中一個選項：
     - Enable (啟用) – 啟用整個聊天群組在單一對談中與您的機器人互動。
     - Disable (停用) - 將對談限制為聊天群組中的一個使用者。

   - 選擇 Activate (啟用) 來建立關聯並將其連結到 Kik 機器人。



後續步驟

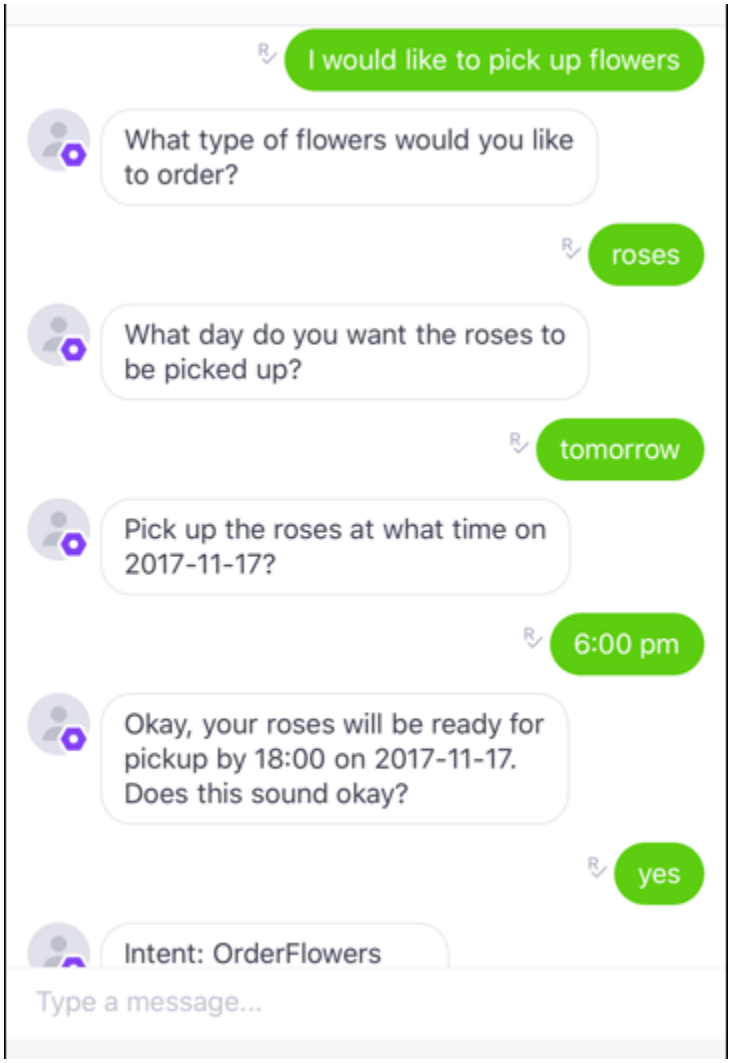
[步驟 4：測試整合](#)

步驟 4：測試整合

現在，您已經在 Amazon Lex 機器人和 Kik 之間建立了關聯，您可以使用 Kik 應用程式來測試關聯性。

1. 啟動 Kik 應用程式並登入。選擇您建立的機器人。

2. 您可以利用以下來測試機器人：



當您輸入每個詞組時，Amazon Lex 機器人會透過 Kik 回應您為每個插槽建立的提示。

整合亞 Amazon Lex 機器人與 Slack

本練習提供將 Amazon Lex 機器人與 Slack 簡訊應用程式整合的指示。您會執行以下步驟：

1. 建立 Amazon Lex 機器人。
2. 建立 Slack 簡訊應用程式。
3. 將 Slack 應用程式與您的機器人 Amazon Lex 整合。
4. 透過與您的 Amazon Lex 機器人進行對話來測試整合。您使用 Slack 應用程式傳送訊息並在瀏覽器視窗中測試。

主題

步驟 1：建立 Amazon Lex 機器人

如果您還沒有 Amazon Lex 機器人，請建立並部署一個機器人。在本主題中，我們假設您使用的是在入門練習 1 中建立的機器人。不過，您可以使用本指南中提供的任何範例機器人。如需入門練習 1，請參閱練習 1：使用藍圖 (主控台) 建立 Amazon Lex 機器人

1. 建立 Amazon Lex 機器人。如需相關指示，請參閱練習 1：使用藍圖 (主控台) 建立 Amazon Lex 機器人。
2. 部署機器人並建立別名。如需相關指示，請參閱練習 3：發佈版本和建立別名。

後續步驟

步驟 2：註冊 Slack 並建立 Slack 團隊

步驟 2：註冊 Slack 並建立 Slack 團隊

註冊 Slack 帳戶並建立 Slack 團隊。如需相關指示，請參閱使用 Slack。在下一節中，您會建立任何 Slack 團隊都可以安裝的 Slack 應用程式。

後續步驟

步驟 3：建立 Slack 簡訊應用程式。

步驟 3：建立 Slack 簡訊應用程式。

請在本節執行以下動作：

1. 在 Slack API 主控台上建立 Slack 應用程式

2. 配置應用程式以將互動式簡訊功能新增至您的機器人：

您在本節最後會取得應用程式登入資料 (用戶端 ID、用戶端密碼和驗證符記)。在下一節中，您會使用此資訊在 Amazon Lex 主控台中設定機器人通道關聯。

1. 在 http://api.slack.com 登入 Slack API 主控台。
2. 建立 應用程式。

   在您成功建立應用程式後、Slack 會顯示應用程式的 Basic Information (基本資訊) 頁面。
3. 如下設定應用程式功能：

   - 在左側選單中，選擇 Intivity (與捷徑)。

     - 選擇切換開關以啟用互動式元件。
     - 在 Request URL (請求 URL) 方塊中，指定任何有效的 URL。例如，您可以使用 **`https://slack.com`**.

       > **Note**
       >
       > 暫時先輸入任何有效的 URL 以取得下一步驟所需的驗證符記。在 Amazon Lex 主控台新增機器人通道關聯之後，您將會更新此 URL。

     - 選擇 Save Changes (儲存變更)。
4. 在左側功能表的 Settings (設定) 中，選擇在 Basic Information (基本資訊)。記錄以下應用程式登入資料：

   - 用戶端 ID
   - 用戶端密碼
   - 驗證符記

後續步驟

步驟 4：整合 Slack 應用程式與 Amazon Lex 機器人

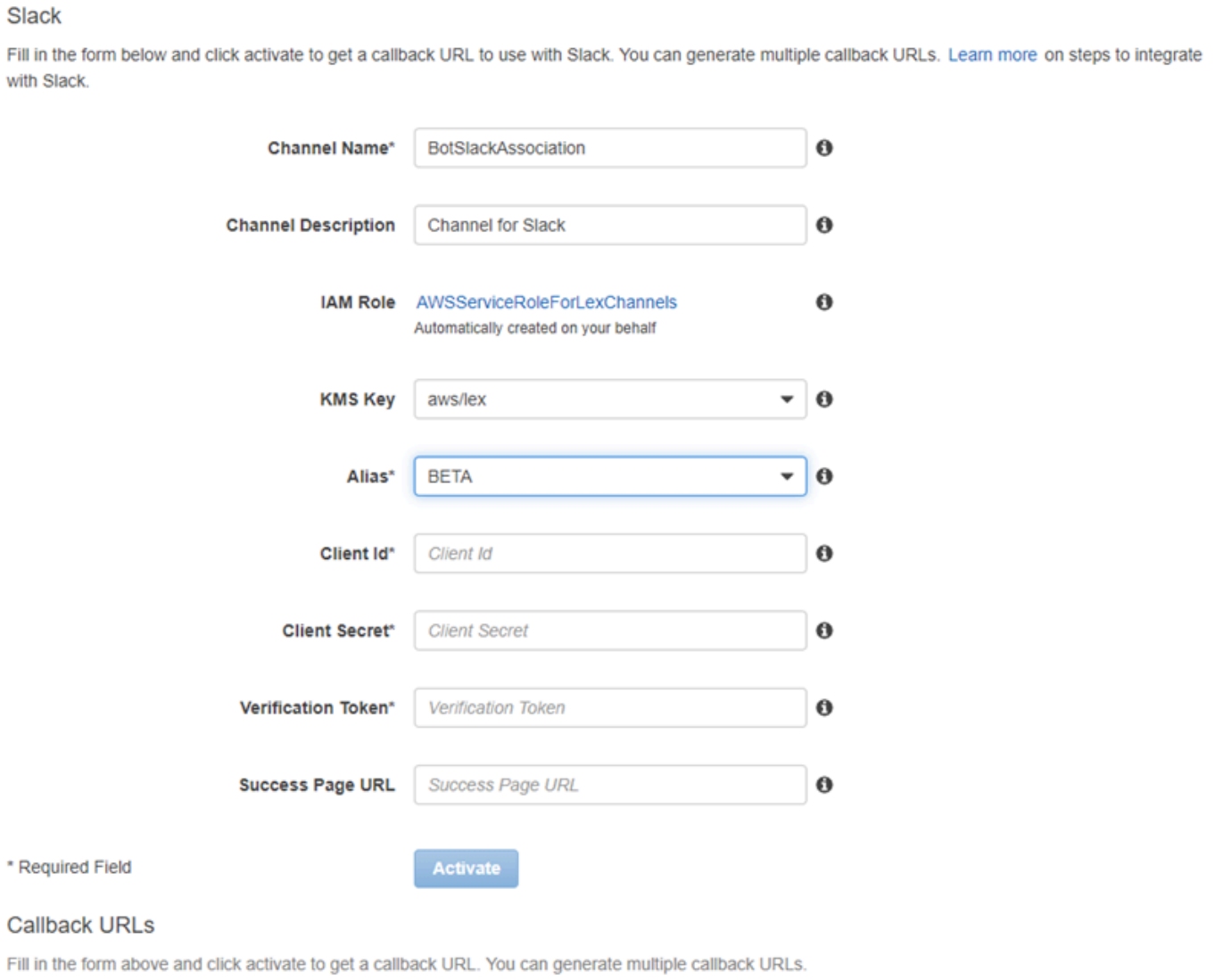
步驟 4：整合 Slack 應用程式與 Amazon Lex 機器人

現在您已擁有 Slack 應用程式登入資料，您可以將應用程式與 Amazon Lex 機器人整合。若要將 Slack 應用程式與您的機器人建立關聯，請在 Amazon Lex 中新增機器人通道關聯。

在 Amazon Lex 主控台中，啟用機器人通道關聯，將機器人與 Slack 應用程式建立關聯。啟動機器人通道關聯後，Amazon Lex 會傳回兩個 URL (回傳網址和 OAuth URL)。記錄這些 URL，因為您稍後需要用到。

將 Slack 應用程式與您的 Amazon Lex 機器人整合

1. 登入 AWS 管理主控台，開啟位於 https://console.aws.amazon.com/lex/ 的 Amazon Lex 主控台。
2. 選擇您在步驟 1 中建立的 Amazon Lex 機器人。
3. 選擇 Channels (管道) 索引標籤。
4. 從左側選單中選擇 Slack。
5. 在 Slack 頁面上，提供以下資訊：
   - 輸入名稱。例如：`BotSlackIntegration`。
   - 從 KMS key (KMS 金鑰)下拉式清單中選擇「aws/lex」。
   - 對於 Alias (別名)，選擇機器人別名。
   - 輸入您在前面的步驟中記錄的 Client Id (用戶端 ID)、Client secret (用戶端秘密) 和 Verification Token (驗證符記)。這些是 Slack 應用程式的登入資料。



6. 選擇 Activate (啟用)。

   主控台會建立機器人管道關聯時，並傳回兩個 URL (回傳 URL 和 OAuth URL)。將它們記錄下來。您會在下一節更新 Slack 應用程式組態來使用這些端點，如下所示：

   - 回傳網址是監聽 Slack 事件的 Amazon Lex 機器人的端點。您可以使用此 URL：
     - 做為 Slack 應用程式的 事件訂閱功能中的請求 URL。
     - 來取代 Slack 應用程式的互動式訊息功能中請求 URL 的預留位置值。
   - OAuth 網址是您的 Amazon Lex 機器人的端點，用於與 Slack 進行 OAuth 交握。

後續步驟

步驟 5：完成 Slack 整合

步驟 5：完成 Slack 整合

在本節中，請使用 Slack API 主控台來完成 Slack 應用程式的整合。

1. 在 http://api.slack.com 登入 Slack API 主控台。選擇您在步驟 3：建立 Slack 簡訊應用程式。中建立的應用程式。
2. 依下列方式更新 OAuth 與許可功能：

   a. 在左側功能表中，選擇 OAuth 與許可。

   b. 在「重新導向網址」區段中，新增 Amazon Lex 在上一個步驟中提供的 OAuth 網址。選擇 Add a new Redirect URL (新增重新導向 URL)，然後選擇 Save URLs (儲存 URL)。

   c. 在「機器人權杖範圍」區段中，使用「新增 OAuth 範圍」按鈕新增兩個權限。以下列文字篩選清單：

      - **`chat:write`**
      - **`team:read`**
3. 將請求 URL 值更新為 Amazon Lex 在上一步中提供的回傳 URL，以更新互動性和捷徑功能。輸入您在步驟 4 中儲存的回傳 URL，然後選擇 Save Changes (儲存變更)。
4. 依下列方式訂閱事件訂閱功能：

   - 選擇 On (開) 選項來啟用事件。
   - 將請求 URL 值設定為 Amazon Lex 在上一個步驟中提供的回傳網址。
   - 在 Subscribe to Bot Events (訂閱機器人事件) 區段中，訂閱 `message.im` 機器人事件，以啟用最終使用者與 Slack 機器人之間的直接簡訊。
   - 儲存變更。
5. 啟用從「訊息」標籤傳送訊息，如下所示：

   - 在左側選單中，選擇 In(App Home)。
   - 在 「顯示分頁」區段中，選擇「允許使用者從訊息」標籤傳送 Slash 命令和訊息。

後續步驟

步驟 6：測試整合

步驟 6：測試整合

現在，使用瀏覽器視窗來測試 Slack 與 Amazon Lex 機器人的整合。

1. 選擇 Settings (設定) 下的 Manage Distribution (管理分佈)。選擇 Add to Slack (新增到 Slack) 以安裝應用程式。授權機器人以回應訊息。
2. 您會被重新導向您的 Slack 團隊。在左側功能表的 Direct Messages (直接訊息) 區段中，選擇您的機器人。如果您沒有看到您的機器人，選擇 Direct Messages (直接訊息) 旁邊的加號圖示 (+) 來搜尋。
3. 與您的 Slack 應用程式進行聊天，該應用程式已連結至 Amazon Lex 機器人。您的機器人現在可回應訊息。

   如果您使用入門練習 1 建立了機器人，可以使用該練習中提供的範例對話。如需詳細資訊，請參閱 步驟 4：將 Lambda 函數新增為程式碼掛接 (主控台)。

整合 Amazon Lex 機器人與 Twilio 可程式化的簡訊

本練習提供將 Amazon Lex 機器人與 Twilio 簡易訊息服務 (SMS) 整合的指示。您會執行以下步驟：

1. 建立 Amazon Lex 機器人
2. 將 Twilio 可編程短信與您的機器人 Amazon Lex 集成
3. 透過使用行動電話上的簡訊服務測試設定，與 Amazon Lex 機器人互動
4. 測試整合

主題

- 步驟 1：建立 Amazon Lex 機器人
- 步驟 2：建立 Twilio SMS 帳戶
- 步驟 3：整合 Twilio 簡訊服務端點與 Amazon Lex 機器人
- 步驟 4：測試整合

步驟 1：建立 Amazon Lex 機器人

如果您還沒有 Amazon Lex 需要建立一個。在本主題中，我們假設您使用的是在入門練習 1 中建立的機器人。不過，您可以使用本指南中提供的任何範例機器人。如需入門練習 1，請參閱練習 1：使用藍圖 (主控台) 建立 Amazon Lex 機器人。

1. 建立 Amazon Lex 機器人。如需相關指示，請參閱練習 1：使用藍圖 (主控台) 建立 Amazon Lex 機器人。
2. 部署機器人並建立別名。如需相關指示，請參閱練習 3：發佈版本和建立別名。

步驟 2：建立 Twilio SMS 帳戶

註冊 Twilio 帳戶，並記錄以下帳戶資訊：

- ACCOUNT SID
- AUTH TOKEN

如需註冊的相關指示，請參閱 https://www.twilio.com/console。

步驟 3：整合 Twilio 簡訊服務端點與 Amazon Lex 機器人

將 Twilio 與您的 Amazon Lex 機器人整合

1. 若要將 Amazon Lex 機器人與您的 Twilio 可程式化 SMS 端點建立關聯，請在 Amazon Lex 主控台啟用機器人通道關聯。啟動機器人通道關聯後，Amazon Lex 會傳回回回 URL。請記錄此回呼 URL，因為稍後將會用到。

   a. 登入，AWS Management Console並在 https://console.aws.amazon.com/lex/ 開啟 Amazon Lex 主控台。

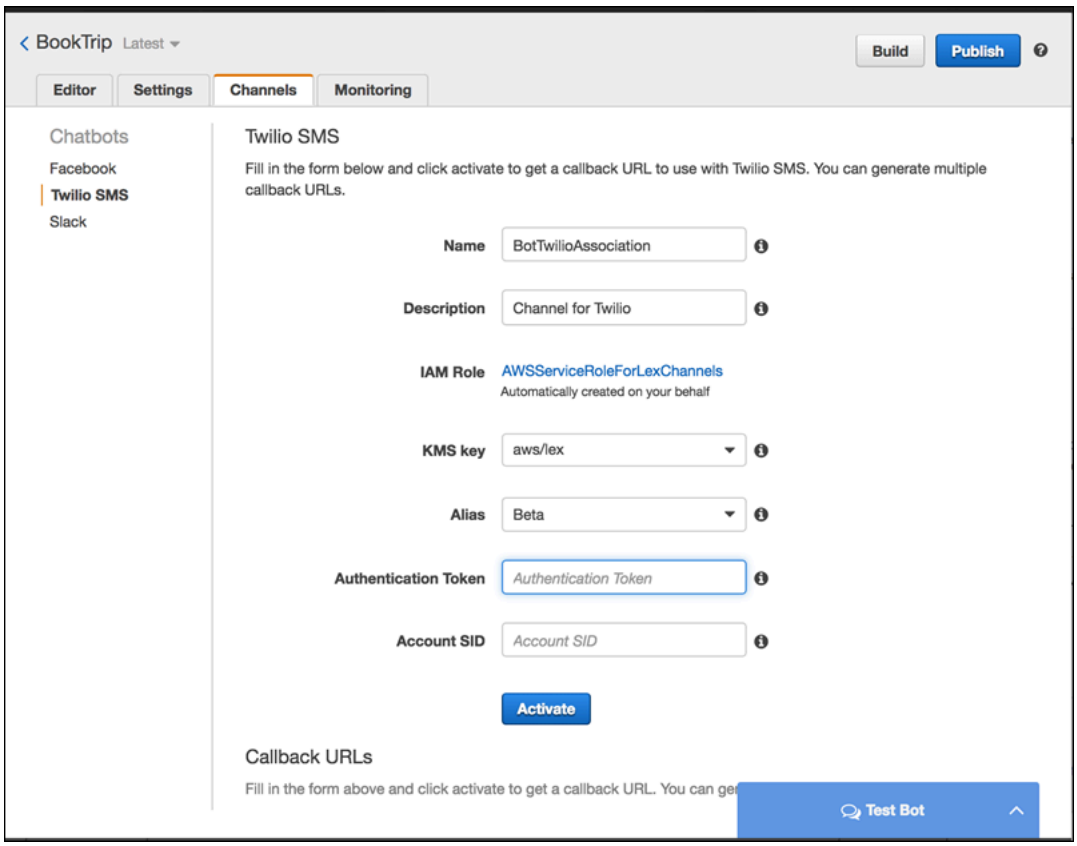
   b. 選擇您在步驟 1 中建立的 Amazon Lex 機器人。

   c. 選擇 Channels (管道) 索引標籤。

   d. 在 Chatbots (聊天機器人) 區段，選擇 Twilio SMS。

   e. 在 Twilio SMS 頁面上，提供以下資訊：

      - 輸入名稱。例如：`BotTwilioAssociation`。
      - 從 KMS key (KMS 金鑰) 中選擇「aws/lex」。
      - 對於 Alias (別名)，選擇機器人別名。
      - 對於 Authentication Token (身分驗證權杖)，輸入您的 Twilio 帳戶的 AUTH TOKEN。
      - 對於 Account SID (帳戶 SID)，輸入您的 Twilio 帳戶的 ACCOUNT SID。



f. 選擇 Activate (啟用)。

主控台隨即建立機器人管道關聯並傳回回呼 URL。記錄此 URL。

2. 在 Twilio 主控台上，將 Twilio 簡訊端點連接到 Amazon Lex 機器人。

a. 從 https://www.twilio.com/console 登入 Twilio 主控台。

b. 如果您沒有 Twilio SMS 端點，請自行建立。

c. 將請求 URL 值設定為 Amazon Lex 在上一步中提供的回呼 URL，以更新簡訊服務的輸入設定組態。

步驟 4：測試整合

使用您的手機測試 Twilio SMS 與您的機器人之間的整合。

測試整合

1. 從 https://www.twilio.com/console 登入 Twilio 主控台，然後執行以下操作：
   a. 在 Manage Numbers (管理號碼) 下方，確認您擁有與簡訊服務相關聯的 Twilio 號碼。

      您可以將訊息傳送到此號碼，並從行動電話與 Amazon Lex 機器人進行簡訊互動。
   b. 確認您的行動電話已列為已驗證來電顯示。

      如果不是，請按照 Twilio 控制台上的說明啟用計劃用於測試的手機。

      現在，您可以使用行動電話將訊息傳送到對應至 Amazon Lex 機器人的 Twilio SMS 端點。
2. 使用您的手機傳送訊息至 Twilio 號碼。

   Amazon Lex 機器人回應。如果您使用入門練習 1 建立了機器人，可以使用該練習中提供的範例對話。如需詳細資訊，請參閱 步驟 4：將 Lambda 函數新增為程式碼掛接 (主控台)。

在行動應用程式中部署 Amazon Lex 機器人

使用AWS Amplify您也可以將 Amazon Lex 機器人與行動或 Web 應用程式集成。如需詳細資訊，請參閱「」入門中的AWS Amplify文件。

匯入及匯出 Amazon Lex 機器人、意圖與參數槽類型

您可以匯入或匯出機器人、意圖或參數槽類型。例如，如果您想要與其他 AWS 帳戶中的同事分享機器人，可以匯出該機器人，然後再將其傳送給該同事。如果您想要將多個表達用語新增到機器人，可以先匯出該機器人，新增表達用語後，再將該機器人匯回至您的帳戶。

您可以出口以 Amazon Lex (用以共用或修改它們) 或 Alexa 技能格式。您可以進口僅在 Amazon Lex 格式。

當您匯出資源時，匯出格式必須與接收匯出之服務相容，也就是 Amazon Lex 或 Alexa Shit。如果您以 Amazon Lex 格式匯出機器人，可以將該機器人重新匯入至您的帳戶，或者另一個帳戶中的 Amazon Lex 使用者也可以將其匯入至自己的帳戶。您也可以使用與 Alexa 技能相容的格式，匯出機器人。然後，可以用 Alexa Skills Kit 匯入機器人，供 Alexa 使用。如需詳細資訊，請參閱 匯出至 Alexa 技能。

當您匯出機器人、意圖或參數槽類型時，其資源會寫入 JSON 檔案。若要匯出機器人、意圖或參數槽類型，可以使用 Amazon Lex 主控台或GetExportoperation. 請使用 StartImport 匯入機器人、意圖或參數槽類型。

主題

- 以 Amazon Lex 格式匯出及匯入
- 匯出至 Alexa 技能

以 Amazon Lex 格式匯出及匯入

若要從 Amazon Lex 匯出機器人、意圖和參數槽類型，且目的是要再重新匯入至 Amazon Lex，請以 Amazon Lex 格式建立 JSON 檔案。您可以在此檔案中編輯您的資源，並將該檔案匯回至 Amazon Lex。例如，您可以將表達用語新增到意圖，再將變更後的意圖匯回至您的帳戶。您也可以使用 JSON 格式來分享資源。例如，您可以從一個 AWS 區域匯出機器人，再將其匯入另一個區域。或是將 JSON 檔案傳送給同事，分享機器人。

主題

- 以 Amazon Lex 格式導出
- 以 Amazon Lex 格式導入
- 匯入及匯出的 JSON 格式

以 Amazon Lex 格式導出

以格式，匯出您的 Amazon Lex 機器人、意圖與參數槽類型，並將該格式匯入AWS帳户。您可匯出下列資源：

- 機器人，包括機器人使用的所有意圖與自訂參數槽類型
- 意圖，包括意圖使用的所有自訂參數槽類型
- 自訂參數槽類型，包括所有參數槽類型的值

您只可匯出有版本編號的資源，而無法匯出資源的 $LATEST 版本。

匯出是一種非同步的程序。匯出完成之後，您會得到由 Amazon S3 預先簽章的 URL。而該 URL 會提供含有 JSON 格式之匯出資源的 .zip 封存檔位置。

您可以使用主控台或 GetExport 操作，匯出機器人、意圖或參數槽類型。

機器人、意圖或參數槽類型的匯出程序皆相同。您只需在下列程序中替換掉機器人的意圖或參數槽類型即可。

匯出機器人

匯出機器人

1. 前往登入 AWS 管理主控台，並開啟 Amazon Lex 主控台，https://console.aws.amazon.com/lex/。
2. 選擇 Bots (機器人)，然後選擇要匯出的機器人。
3. 在 Actions (動作) 功能表上，選擇 Export (匯出)。
4. 在 Export Bot (匯出機器人) 對話方塊中，選擇要匯出的機器人版本。為 Platform (平台) 選擇 Amazon Lex。
5. 選擇 Export (匯出)。
6. 下載並儲存 .zip 封存檔。

Amazon Lex 會將機器人匯出為 JSON 檔案，並將其包含在 .zip 封存檔中。若要更新機器人，請修改 JSON 文字，然後再將其匯回至 Amazon Lex。

下一步驟

以 Amazon Lex 格式導入

以 Amazon Lex 格式導入

當您以 Amazon Lex 格式將資源匯出至 JSON 檔案後，即可將內含該資源的 JSON 檔案，匯入一或多個AWS帳户。例如，您可以匯出機器人，然後再將其匯入另一個 AWS 區域。或者，也可以將該機器人傳送給同事，讓同事自行將其匯入自己的帳戶。

當您匯入機器人、意圖或參數槽類型時，必須決定是否要在匯入期間覆寫資源 (例如意圖或參數槽類型) 的 $LATEST 版本，或若是當希望保留帳戶中的資源時，是否要讓匯入失敗。例如，如果您將資源的編輯版本上傳到您的帳戶，您可以選擇覆寫 $LATEST 版本。如果您要上傳同事傳送給您的資源，則可以選擇若發生資源衝突時，就讓匯入失敗，以免替換掉了自己原先的資源。

在匯入資源時，會套用指派給發出匯入要求之使用者的許可。該使用者必須具有帳戶中匯入所影響之所有資源的許可。該使用者也必須具有下列操作的許可：GetBot、PutBot、GetIntent PutIntent、GetSlotType、PutSlotType。如需許可的詳細資訊，請參閱「亞馬遜萊克斯如何與 IAM 合作」。

匯入會回報處理期間所發生的錯誤。某些錯誤會在匯入開始前回報，而其他錯誤則會在匯入程序期間回報。例如，如果匯入意圖的帳户未具備呼叫意圖使用之 Lambda 函數的許可，則要先對參數槽類型或意圖進行變更，才可進行匯入。如果匯入在匯入程序期間失敗，則在程序失敗前所匯入之 $LATEST 版本的所有意圖或參數槽類型，皆會修改。您無法還原對 $LATEST 版本所進行的變更。

當您匯入資源時，所有相依的資源都會匯入 $LATEST 版本的資源，然後為其提供一個版本編號。例如，如果機器人使用意圖，就會為該意圖提供一個版本編號。如果意圖使用自訂參數槽類型，就會為該參數槽類型提供一個版本編號。

資源只會匯入一次。例如，如果機器人包含 `OrderPizza` 意圖與 `OrderDrink` 意圖，且兩者皆仰賴自訂的參數槽類型 `Size`，則只會匯入一次該 `Size` 參數槽類型，而同時用於這兩項意圖。

> **Note**
>
> 如果您將自動程序導出為`enableModelImprovements`參數`false`，則必須打開包含 bot 定義的 .zip 文件，然後更改`enableModelImprovements`參數`true`以下區域：
>
> - 亞太區域 (新加坡) (ap-southeast-1)
> - 亞太區域 (東京) (ap-northeast-1)
> - 歐洲 (法蘭克福) (eu-central-1)
> - 歐洲 (倫敦) (eu-west-2)

匯入機器人、意圖或參數槽類型的程序皆相同。您只需在下列程序中適當地替換掉意圖或參數槽類型即可。

匯入機器人

匯入機器人

1. 前往登入 AWS 管理主控台，並開啟 Amazon Lex 主控台，https://console.aws.amazon.com/lex/。
2. 選擇 Bots (機器人)，然後選擇要匯入的機器人。若要匯入新的機器人，請跳過此步驟。
3. 為 Actions (動作) 選擇 Import (匯入)。
4. 為 Import Bot (匯入機器人) 選擇.zip 封存檔，其內應有包含要匯入之機器人的 JSON 檔案。如果您要在合併前先查看合併衝突，請選擇 Notify me of merge conflicts (出現合併衝突請通知我)。如果您關閉了衝突檢查，就會覆寫機器人使用之 $LATEST 版本的所有資源。
5. 選擇 Import (匯入)。如果您已選擇在發生合併衝突時通知您，則在發生衝突時，就會出現列有這些衝突的對話方塊。若要覆寫 $LATEST 版本之所有衝突的資源，請選擇 Overwrite and continue (覆寫並繼續)。若要停止匯入，請選擇 Cancel (取消)。

現在即已可在帳戶中測試機器人。

匯入及匯出的 JSON 格式

下列範例示範以 Amazon Lex 格式，匯出及匯入參數槽類型、意圖和機器人的 JSON 結構。

參數槽類型結構

以下為自訂參數槽類型的 JSON 結構。當您匯入或匯出參數槽類型時，以及當您匯出相依於自訂參數槽類型的意圖時，請使用此結構。

```
{
  "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"
  }
}
```

意圖結構

以下為意圖的 JSON 結構。當您匯入或匯出意圖以及相依於意圖的機器人時，請使用此結構。

```
{
  "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 #######.
    ]
  }
}
```

機器人結構

以下為機器人的 JSON 結構。當您匯入或匯出機器人時，請使用此結構。

```
{
  "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 ####.
    ],
    "slotTypes": [
        List of slot type JSON structures.
        For more information, see #######.
    ],
    "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"
        }
      ]
    }
  }
}
```

匯出至 Alexa 技能

您可以使用與 Alexa 技術相容的格式匯出機器人結構描述。當您將機器人匯出成 JSON 檔案後，請使用技能建置器將其上傳至 Alexa。

匯出機器人及其結構描述 (互動模型)

1. 登入AWS Management Console，並開啟 Amazon Lex 主控台，並開啟https://console.aws.amazon.com/lex/。
2. 選擇您想要匯出的機器人。
3. 為 Actions (動作) 選擇 Export (匯出)。
4. 選擇要匯出之機器人的版本。為格式選擇 Alexa Skills Kit，然後選擇 Export (匯出)。
5. 出現下載對話方塊時，請選擇檔案的儲存位置，然後選擇 Save (儲存)。

下載的檔案是 .zip 封存檔，內含一個以匯出機器人命名的檔案。它包含將機器人匯出為 Alexa 技能所需的資訊。

> **Note**
>
> Amazon Lex 與 Alexa Scit 的相異點如下：
>
> - Alexa Skills Kit 不支援工作階段屬性，以方括號 ([]) 表示。您需要更新使用工作階段屬性的提示。
> - Alexa Skills Kit 不支援標點符號。您需要更新使用標點符號的表達用語。

將機器人上傳至 Alexa 技能

1. 在 https://developer.amazon.com/ 登入開發人員入口網站。
2. 在 Alexa Skills (Alexa 技能) 頁面，選擇 Create Skill (建立技能)。
3. 在 Create a new skill (建立新的技能) 頁面，輸入技能名稱和該技能的預設語言。請確定已為技能模型選取 Custom (自訂)，然後選擇 Create skill (建立技能)。
4. 請確定已選取 Start from scratch (從頭開始) 並選擇 Choose (選擇)。
5. 在左側功能表中，選擇 JSON Editor (JSON 編輯器)。將您從 Amazon Lex 匯出的 JSON 檔案拖曳到 JSON 編輯器。
6. 選擇 Save Model (儲存模型) 以儲存您的互動模式。

將結構描述上傳至 Alexa 技能後，使用 Alexa 進行執行技能的必要變更。如需有關建立 Alexa 技能的詳細資訊，請參閱 Alexa Skills Kit 中的使用技能產生器 (Beta 版)。

其他示例：創建亞馬遜 Lex 機器人

以下各節提供其他 Amazon Lex 練習及step-by-step指示。

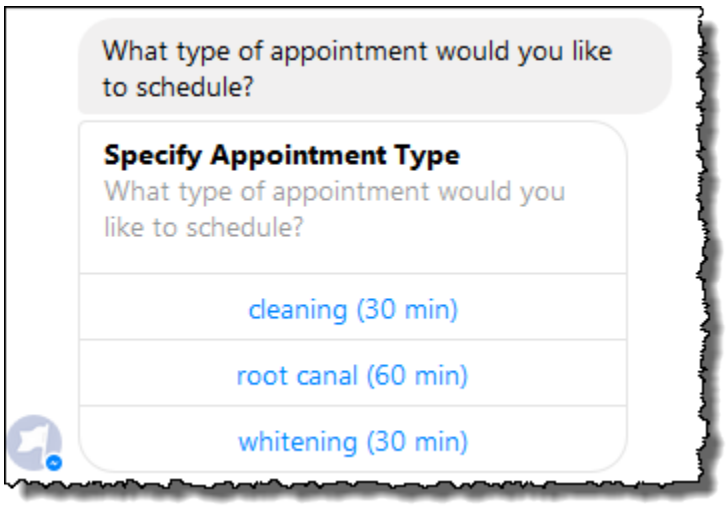
主題

安排預約

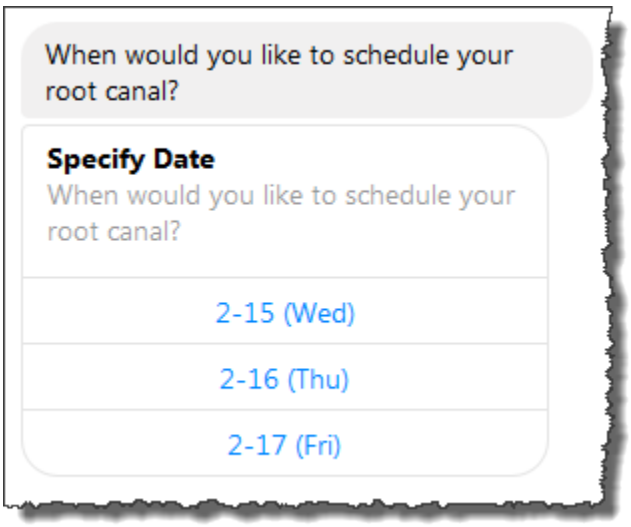
在本練習中的範例機器人會為牙醫診所排定預約。範例同時說明如何使用回應卡，利用按鈕來取得使用者輸入。範例特別說明以動態方式在執行時間產生回應卡。

您可以在建置時間設定回應卡 (又稱靜態回應卡) 或在 AWS Lambda 函數中以動態方式產生。在此範例中，機器人使用以下回應卡：

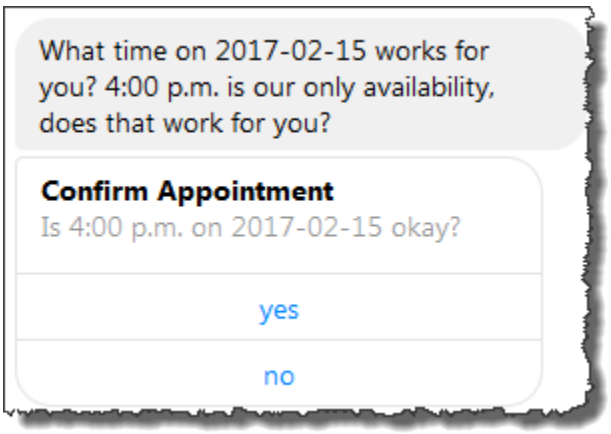
- 列出預約類型按鈕的回應卡。如需範例，請參閱下列影像：



- 列出預約日期按鈕的回應卡。如需範例，請參閱下列影像：



- 列出按鈕以確認建議之預約時間的回應卡。如需範例，請參閱下列影像：



可預約的日期和時間各不相同，因此您需要在執行時間產生回應卡。您要使用 AWS Lambda 函數，以動態方式產生這些回應卡。Lambda 函數在其對亞馬遜萊克斯的響應返回響應卡。Amazon Lex 在對客戶的回應中包含回應卡。

如果用戶端 (例如，Facebook Messenger) 支援回應卡，使用者可以從按鈕清單中選擇，或輸入回應。否則，使用者會直接輸入回應。

除了上例中顯示的按鈕之外，您也可以包含影像、附件和其他有用的資訊以顯示在回應卡上。如需有關回應卡的資訊，請參閱 回應卡。

在本練習中，您會進行以下動作：

- 建立並測試機器人 (使用ScheduleAppointment藍圖)。就此練習，您使用機器人藍圖來快速設定和測試機器人。如需可用的藍圖清單，請參閱Amazon Lex 和AWS LambdaBlueprints (藍圖)。此機器人已針對一個意圖 (MakeAppointment) 進行預先設定。

- 建立和測試 Lambda 函數 (使用 Lambda 提供的lex-make-appointment-python藍圖)。您可以將MakeAppointment意圖設定為使用此 Lambda 函數做為程式碼掛接，以執行初始化、驗證和履行工作。

  > **Note**
  >
  > 提供的 Lambda 函數範例會根據牙醫預約的模擬可用性展示動態對話。在實際的應用程式中，您可以使用實際的行事曆來設定預約時間。

- 更新MakeAppointment意圖組態，以使用 Lambda 函數做為程式碼掛接。然後，測試端對端體驗。
- 將排程約會機器人發佈到 Facebook Messenger，以便您可以查看運作中的回應卡 (Amazon Lex 主控台中的用戶端目前不支援回應卡)。

以下章節提供有關在本練習中所用之藍圖的摘要資訊。

主題

機器人藍圖概觀 (ScheduleAppointment)

您用來為此練習建立機器人的ScheduleAppointment藍圖已預先設定下列項目：

- 槽類型 – 一個稱為 AppointmentTypeValue 的自訂槽類型，包含 root canal、cleaning 和 whitening 的列舉值。

- 意圖 – 一個意圖 (MakeAppointment)，這已預先設定如下：
  - 槽 – 已使用以下槽來設定意圖：
    - 槽 AppointmentType，為 AppointmentTypes 自訂類型。
    - 槽 Date，為 AMAZON.DATE 內建類型。
    - 槽 Time，為 AMAZON.TIME 內建類型。
  - 表達用語 - 意圖已使用以下表達用語進行預先設定：
    - 「我想要預約」
    - 「預約」
    - 「預訂 {AppointmentType}」

    如果使用者說出上述任何內容，Amazon Lex 會判斷這MakeAppointment是意圖，然後使用提示來引出插槽資料。
  - 提示 - 意圖已使用以下提示進行預先設定：
    - AppointmentType 槽的提示 -「您想要排定哪一種預約？」
    - 提示輸入Date插槽 —「我應該何時排定您的 {AppointmentType}？」
    - 提示輸入Time插槽 —「您要在什麼時間排程 {AppointmentType}？」 以及

      「在 {Date} 幾點？」
    - 確認提示 -「可以約 {Time}，要我幫您預約該時間嗎？」
    - 取消訊息 -「好的，我不會排定預約。」

Lambda 函數藍圖概觀 (lex-make-appointment-python)

Lambda 函數藍圖 (lex-make-appointment-python) 是您使用機器人藍圖建立之機 ScheduleAppointment器人的程式碼掛接。

此 Lambda 函數藍圖程式碼可以執行初始化/驗證和履行工作。

- Lambda 函數程式碼會根據牙醫預約的範例可用性來展示動態交談 (在實際應用程式中，您可能會使用行事曆)。對於使用者指定的一天或日期，程式碼設定如下：
  - 如果沒有可用的約會，Lambda 函數會傳回回應，指示 Amazon Lex 提示使用者另一天或日期 (將dialogAction類型設定為ElicitSlot)。如需詳細資訊，請參閱回應格式。

- 如果在指定日期或日期只有一個可用的約會，Lambda 函數會建議回應中的可用時間，並指示 Amazon Lex 透過在回應`dialogAction`中設定，以取得使用者確認`ConfirmIntent`。這裡說明了您可以如何透過主動提出可預約時間，改善使用者體驗。
- 如果有多個可用的約會，Lambda 函數會在回應 Amazon Lex 時傳回可用時間清單。Amazon Lex 會將回應傳回給客戶端，其中包含來自 Lambda 函數的訊息。

- 作為履行程式碼掛鉤，Lambda 函數會傳回摘要訊息，指出已排定約會 (亦即，意圖已完成)。

> **Note**
>
> 在這個範例中，我們示範如何使用回應卡。拉姆達函數構造並返回一個響應卡亞馬遜萊克斯。回應卡會將可預約的日期和時間列為可供選擇的按鈕。使用 Amazon Lex 主控台提供的用戶端測試機器人時，您看不到回應卡。若要查看，必須將機器人與簡訊平台整合，例如 Facebook Messenger。如需相關指示，請參閱整合 Amazon Lex 機器人與臉書信使。如需回應卡的詳細資訊，請參閱管理訊息 。

當 Amazon Lex 叫用 Lambda 函數時，它會將事件資料作為輸入傳遞。其中一個事件欄位是`invocationSource` Lambda 函數用來在輸入驗證和履行活動之間進行選擇。如需詳細資訊，請參閱輸入事件格式。

後續步驟

步驟 1：創建一個亞馬遜萊克斯機器人

步驟 1：創建一個亞馬遜萊克斯機器人

在本節中，您會使用 Amazon Lex 主控台提供的ScheduleAppointment藍圖來建立 Amazon Lex 機器人。

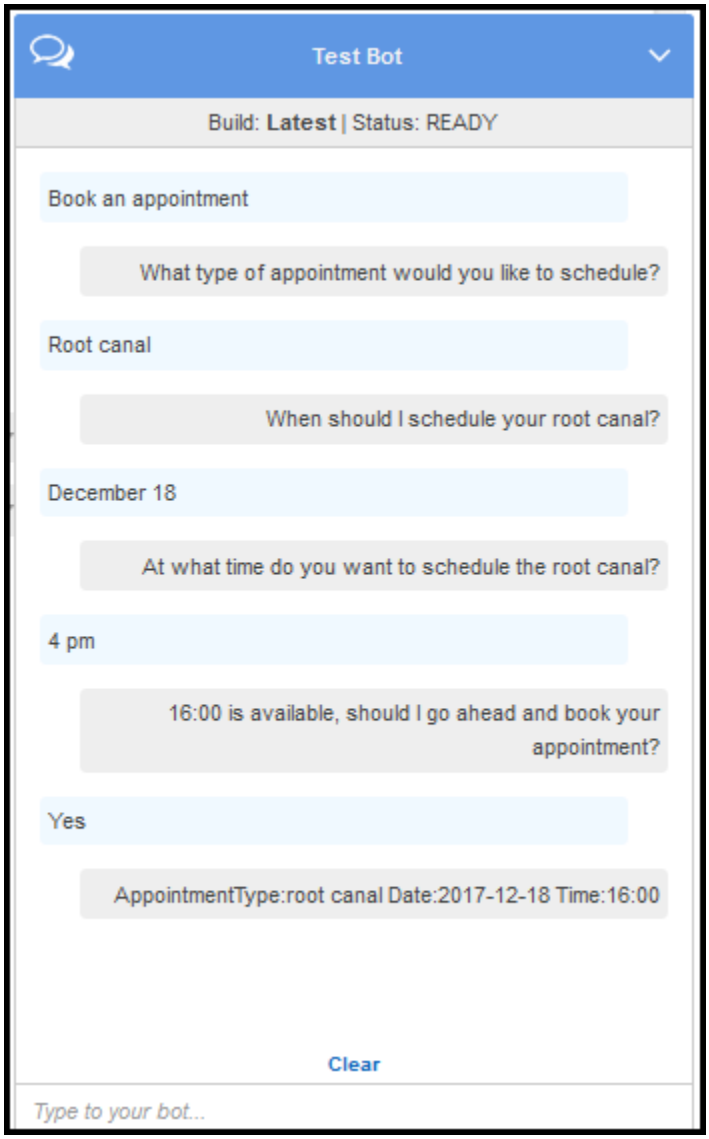
1. 登錄到AWS Management Console並打開亞馬遜萊克斯控制台 https://console.aws.amazon.com/lex/.
2. 在 Bots (機器人) 頁面，選擇 Create (建立)。
3. 在 [Create your Lex bot] 頁面，執行下列動作：
   - 選擇ScheduleAppointment藍圖。
   - 保留預設機器人名稱 (ScheduleAppointment)。
4. 選擇 建立 。

此步驟會儲存並建立機器人。主控台會在建置程序期間將下列請求傳送至 Amazon Lex：

- 建立新版本的槽類型 (從 $LATEST 版本)。如需有關此機器人藍圖中定義之槽類型的資訊，請參閱 機器人藍圖概觀 (ScheduleAppointment)。
- 建立 `MakeAppointment` 意圖的版本 (從 $LATEST 版本)。在某些情況下，主控台會在建立新版本之前傳送 `update` API 操作的請求。
- 更新機器人的 $LATEST 版本。

  目前，Amazon Lex 會為機器人建立機器人學習模型。當您在主控台中測試機器人時，主控台會使用執行階段 API 將使用者輸入傳回 Amazon Lex。然後，Amazon Lex 會使用機器學習模型來解譯使用者輸入。

5. 控制台顯示ScheduleAppointment機器人。在 [Editor] 標籤中，檢閱預先設定的意圖 (`MakeAppointment`) 詳細資訊。
6. 在測試視窗中測試機器人。使用以下螢幕擷取畫面與您的機器人進行測試對談：



注意下列事項：

- 機器人從初始的使用者輸入 (「預約」) 推斷意圖 (MakeAppointment)。
- 機器人之後使用設定的提示向使用者取得槽資料。
- 機器人藍圖已使用以下確認提示設定 MakeAppointment 意圖：

```
{Time} is available, should I go ahead and book your appointment?
```

使用者提供所有插槽資料後，Amazon Lex 會傳回回應給用戶端，並顯示確認提示做為訊息。用戶端會為使用者顯示訊息：

```
16:00 is available, should I go ahead and book your appointment?
```

請注意，機器人會接受任何預約日期和時間值，因為您沒有任何程式碼來初始化或驗證使用者資料。在下一節中，您可以新增 Lambda 函數來執行此操作。

後續步驟

步驟 2：建立 Lambda 函數

步驟 2：建立 Lambda 函數

在本節中，您會使用 Lambda 主控台中提供的藍圖 (lex-make-appointment-python) 來建立 Lambda 函數。您也可以使用主控台提供的範例 Amazon Lex 事件資料來呼叫 Lambda 函數來測試該函數。

1. 請登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/lambda/ 的 AWS Lambda 主控台。
2. 選擇 Create a Lambda function (建立 Lambda 函數)。
3. 對於選取藍圖，鍵**lex**入以尋找藍圖，然後選擇lex-make-appointment-python藍圖。
4. 設定 Lambda 函數，如下所示。
   - 輸入 Lambda 函數名稱 (`MakeAppointmentCodeHook`)。
   - 對於角色，選擇 Create a new role from template(s) (從範本建立新角色)，然後輸入角色名稱。
   - 保留其他預設值。
5. 選擇 Create Function (建立函數)。
6. 如果您使用的是英文 (US) (en-US) 以外的地區設定，請依照中更新特定區域設置的藍圖所述更新意圖名稱。
7. 測試拉姆達函數。
   a. 選擇操作，然後選擇 Configure test event (設定測試事件)。
   b. 從 Sample event template (範例事件範本) 清單中選擇 Lex-Make Appointment (preview) (Lex-Make 預約 (預覽))。此範例事件使用 Amazon Lex 請求/回應模型，其值設定為符合來自 Amazon Lex 機器人的請求。如需 Amazon Lex 請求/回應模型的相關資訊，請參閱。使用 Lambda 函數
   c. 選擇 Save and test (儲存並測試)。

d. 確認 Lambda 函數是否成功執行。在這種情況下，回應與 Amazon Lex 回應模型相符。

後續步驟

步驟 3：更新意圖：設定程式碼掛勾

步驟 3：更新意圖：設定程式碼掛勾

在本節中，您將更新`MakeAppointment`意圖的組態，以使用 Lambda 函數做為驗證和履行活動的程式碼掛接。

1. 在 Amazon Lex 主控台中，選取ScheduleAppointment機器人。控制台顯示MakeAppointment意圖。如下修改意圖組態。

   > **Note**
   >
   > 您只能更新任何 Amazon Lex 資源 (包括意圖) 的 $LATE 版本。確定意圖版本已設為 $LATEST。您尚未發佈機器人版本，所以在主控台中應該仍舊是 $LATEST 版本。

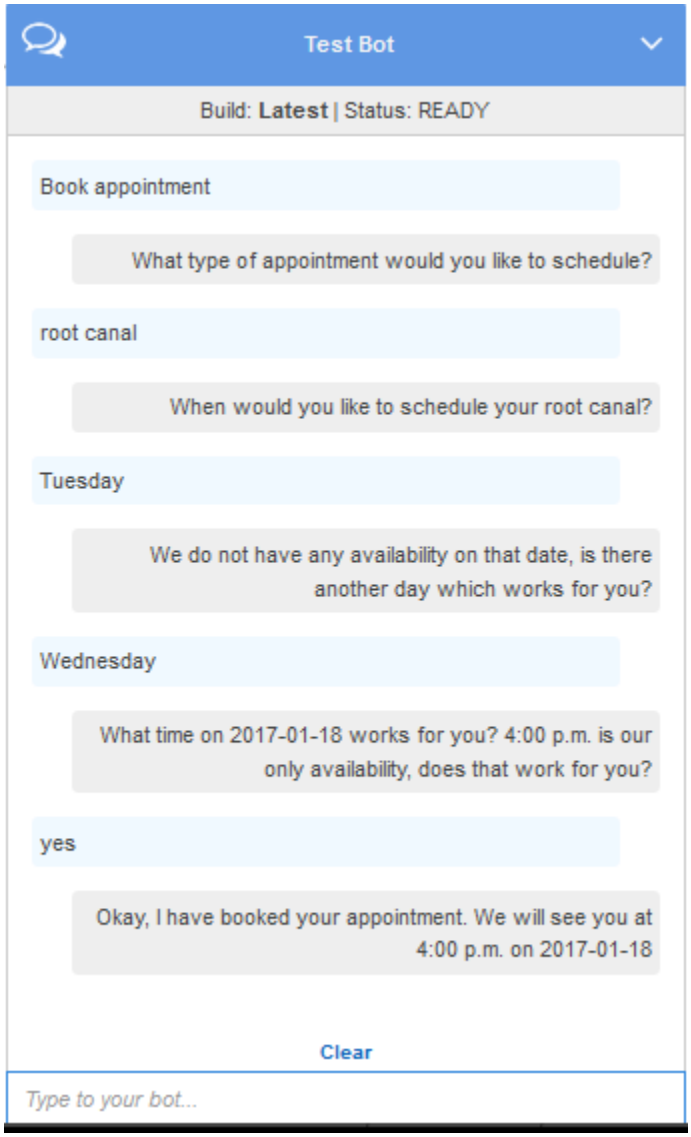
   a. 在 [選項] 區段中，選擇 [初始化和驗證程式碼掛接]，然後從清單中選擇 Lambda 函數。

   b. 在「履行」區段中，選擇 AWS Lambda 函數，然後從清單中選擇 Lambda 函數。

   c. 選擇 Goodbye message (再見訊息) 並輸入訊息。

2. 選擇 Save (儲存)，然後選擇 Build (建置)。

3. 測試機器人，如下圖所示：



後續步驟

步驟 4：將機器人部署在 Facebook Messenger 平台上

步驟 4：將機器人部署在 Facebook Messenger 平台上

在上一節中，您使用 Amazon Lex 主控台中的用戶端測試了ScheduleAppointment機器人。目前 Amazon Lex 主控台不支援回應卡。若要測試機器人所支援之動態產生的回應卡，將機器人部署在 Facebook Messenger 平台上並進行測試。

如需相關指示，請參閱整合 Amazon Lex 機器人與臉書信使。

後續步驟

資訊流程的詳細資訊

資訊流程的詳細資訊

ScheduleAppointment 機器人藍圖主要是展示如何運用動態產生的回應卡。本練習中的 Lambda 函數在其對亞馬遜萊克斯的回應中包含回應卡。Amazon Lex 在回覆客戶端時包含回應卡。本部分說明下列兩項：

- 客戶端和亞馬遜萊克斯之間的數據流。

  本節假設用戶端使用PostText執行階段 API 將請求傳送至 Amazon Lex，並據此顯示請求/回應詳細資訊。如需有關 PostText 執行時間 API 的詳細資訊，請參閱 PostText。

  > **Note**
  >
  > 如需用戶端和用戶端使用 PostContent API 之間的 Amazon Lex 之間的資訊流程範例，請參閱步驟 2a (選用)：檢閱口語化資訊流程的詳細資訊 (主控台) 。

- 亞馬遜萊克斯和拉姆達函數之間的數據流。如需詳細資訊，請參閱Lambda 函數輸入事件和回應格式。

> **Note**
>
> 此範例假設您使用的是 Facebook 信使用戶端，該用戶端不會將要求中的工作階段屬性傳遞給亞馬遜 Lex。因此，本節所示的範例顯示空的 sessionAttributes。如果您使用 Amazon Lex 主控台中提供的用戶端測試機器人，用戶端會包含工作階段屬性。

本節說明在使用者每次輸入之後，會發生什麼情況。

1. 使用者：類型**Book an appointment**。

   a. 用戶端 (主控台) 傳送以下 PostContent 請求給 Amazon Lex：

```
POST /bot/ScheduleAppointment/alias/$LATEST/
user/bijt6rovckwecnzesbthrr1d7lv3ja3n/text
"Content-Type":"application/json"
"Content-Encoding":"amz-1.0"

{
   "inputText":"book appointment",
   "sessionAttributes":{}
}
```

請求 URI 和本文都向亞馬遜 Lex 提供信息：

- 請求 URI — 提供機器人名稱 (ScheduleAppointment)、機器人別名 ($LATEST) 以及使用者名稱 ID。結尾 `text` 表示它是一種 `PostText` (而非 `PostContent`) API 請求。
- 請求本文 – 包含使用者輸入 (`inputText`) 和空的 `sessionAttributes`。

b. 從中`inputText`，亞馬遜萊克斯檢測到意圖（`MakeAppointment`）。此服務會叫用設定為程式碼掛接的 Lambda 函數，藉由傳遞下列事件來執行初始化和驗證。如需詳細資訊，請參閱 輸入事件格式。

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

除了用戶端傳送的資訊之外，Amazon Lex 還包含下列資料：

- currentIntent – 提供目前的意圖資訊。
- invocationSource— 指出 Lambda 函數叫用的目的。在這種情況下，目的是執行用戶數據初始化和驗證。Amazon Lex 知道使用者尚未提供所有插槽資料來達成意圖。)
- messageVersion— 目前亞馬遜萊克斯僅支持 1.0 版本。

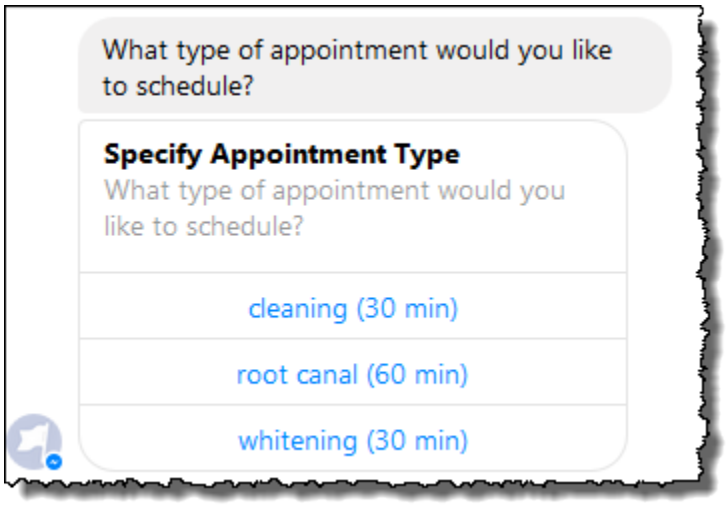
c. 目前，所有槽值都是 null (沒有可驗證的內容)。Lambda 函數會將下列回應傳回給 Amazon Lex，指示服務以引出插槽的AppointmentType資訊。如需回應格式的相關資訊，請參閱回應格式。

```
{
    "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,
```

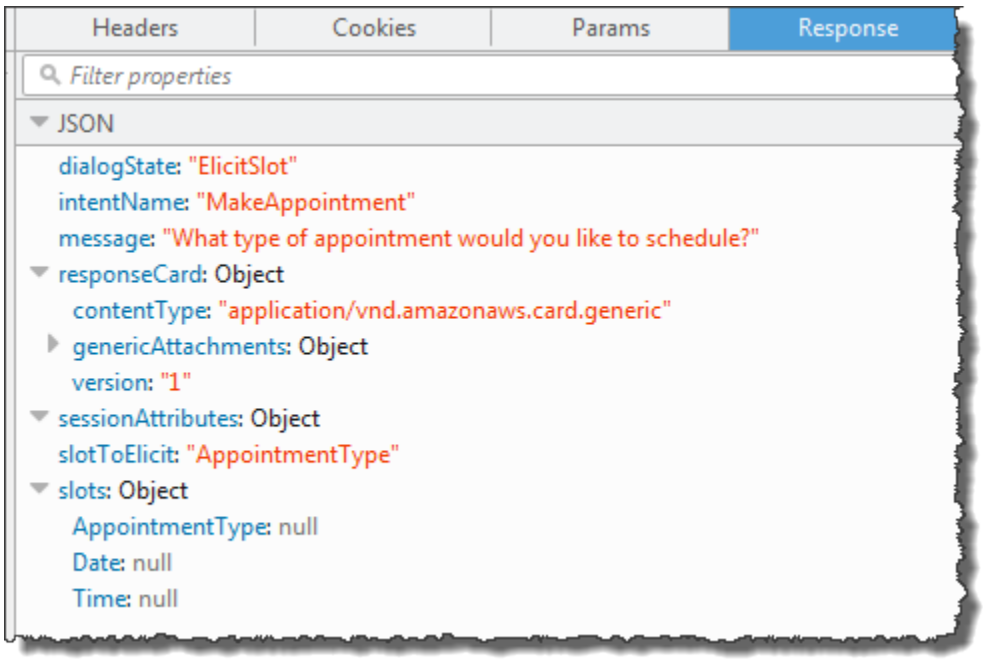
```
            "Date": null,
            "Time": null
        },
        "type": "ElicitSlot",
        "message": {
            "content": "What type of appointment would you like to schedule?",
            "contentType": "PlainText"
        }
    },
    "sessionAttributes": {}
}
```

回應包含 dialogAction 和 sessionAttributes 欄位。此外，dialogAction 欄位會傳回以下欄位：

- type— 透過將此欄位設定為ElicitSlot，Lambda 函數會指示 Amazon Lex 針對欄位中指定的插槽引出slotToElicit值。Lambda 函數還提供了message一個傳達給用戶。
- responseCard— 識別AppointmentType插槽可能值的清單。支援回應卡片的用戶端 (例如 Facebook Messenger) 會顯示回應卡，讓使用者選擇約會類型，如下圖所示：



d. 如 Lambda 函數回應dialogAction.type中所述，Amazon Lex 會將下列回應傳回給用戶端：



客戶端讀取響應，然後顯示以下消息：「您想安排什麼類型的約會？」 以及回應卡 (如果用戶端支援回應卡的話)。

2. 使用者：根據用戶端，使用者有兩個選項：

   - 如果顯示回應卡，請選擇 root canal (60 min) 或輸入 **`root canal`**。
   - 如果用戶端不支援回應卡，輸入 **`root canal`**。

   a. 用戶端會將下列`PostText`要求傳送至 Amazon Lex (已新增換行符號以提高可讀性)：

```
POST /bot/BookTrip/alias/$LATEST/user/bijt6rovckwecnzesbthrr1d7lv3ja3n/text
"Content-Type":"application/json"
"Content-Encoding":"amz-1.0"

{
   "inputText": "root canal",
    "sessionAttributes": {}
}
```

   b. Amazon Lex 透過傳送下列事件做為參數，叫用 Lambda 函數進行使用者資料驗證：

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

在事件資料中，注意下列事項：

- invocationSource 繼續做為 DialogCodeHook。在此步驟中，我們只需驗證使用者資料。
- 亞馬遜萊克斯將currentIntent.slots插槽中的AppointmentType字段設置為root canal。
- 亞馬遜萊克斯只是在用戶端和 Lambda 函數之間傳遞sessionAttributes欄位。

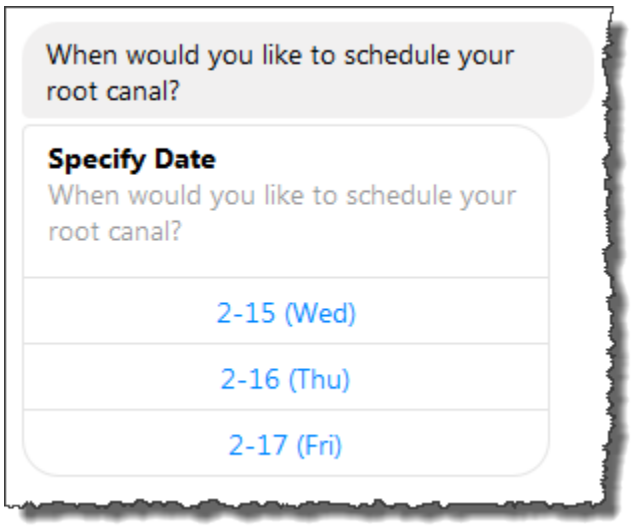
c. Lambda 函數會驗證使用者輸入，並將下列回應傳回給 Amazon Lex，引導服務引出約會日期的值。

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

同樣地，回應也包含 dialogAction 和 sessionAttributes 欄位。此外，dialogAction 欄位會傳回以下欄位：

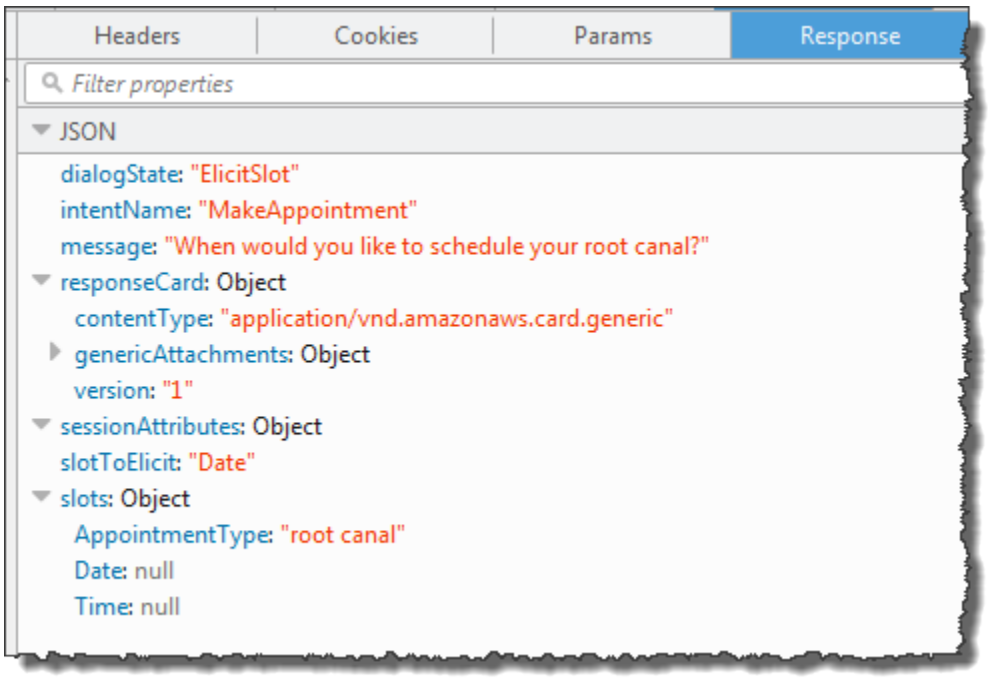
- `type`— 透過將此欄位設定為`ElicitSlot`，Lambda 函數會指示 Amazon Lex 針對欄位中指定的插槽引出`slotToElicit`值。Lambda 函數還提供了`message`一個傳達給用戶。
- `responseCard`— 識別`Date`插槽可能值的清單。支援回應卡 (例如 Facebook Messenger) 的用戶端會顯示回應卡，讓使用者選擇約會日期，如下圖所示：



雖然 Lambda 函數返回五個日期，客戶端（Facebook 信使）有一個響應卡三個按鈕的限制。因此，您在螢幕擷取畫面中只會看到前三個值。

這些日期在 Lambda 函數中進行硬編碼。在生產應用程式中，您可以使用行事曆以即時取得可用的日期。由於日期是動態的，因此您必須在 Lambda 函數中動態產生回應卡。

d. Amazon Lex 會注意到`dialogAction.type`並將下列回應傳回給用戶端，其中包含來自 Lambda 函數回應的資訊。



用戶端會顯示訊息：When would you like to schedule your root canal? 和回應卡 (如果用戶端有支援回應卡)。

3. 使用者：類型**Thursday**。

   a. 用戶端會將下列PostText要求傳送至 Amazon Lex (已新增換行符號以提高可讀性)：

```
POST /bot/BookTrip/alias/$LATEST/user/bijt6rovckwecnzesbthrr1d7lv3ja3n/text
"Content-Type":"application/json"
"Content-Encoding":"amz-1.0"

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

   b. Amazon Lex 會將下列事件做為參數傳送，藉此叫用 Lambda 函數進行使用者資料驗證：

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

在事件資料中，注意下列事項：

- invocationSource 繼續做為 DialogCodeHook。在此步驟中，我們只需驗證使用者資料。
- 亞馬遜萊克斯將currentIntent.slots插槽中的Date字段設置為2017-02-16。
- 亞馬遜萊克斯只是在客戶端和 Lambda 函數sessionAttributes之間傳遞。

c. Lambda 函數會驗證使用者輸入。這次 Lambda 函數會判斷指定日期沒有可用的約會。它會將下列回應傳回給 Amazon Lex，並將服務導向至再次引出預約日期的值。

```
{
    "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"
                        },
                        {
```

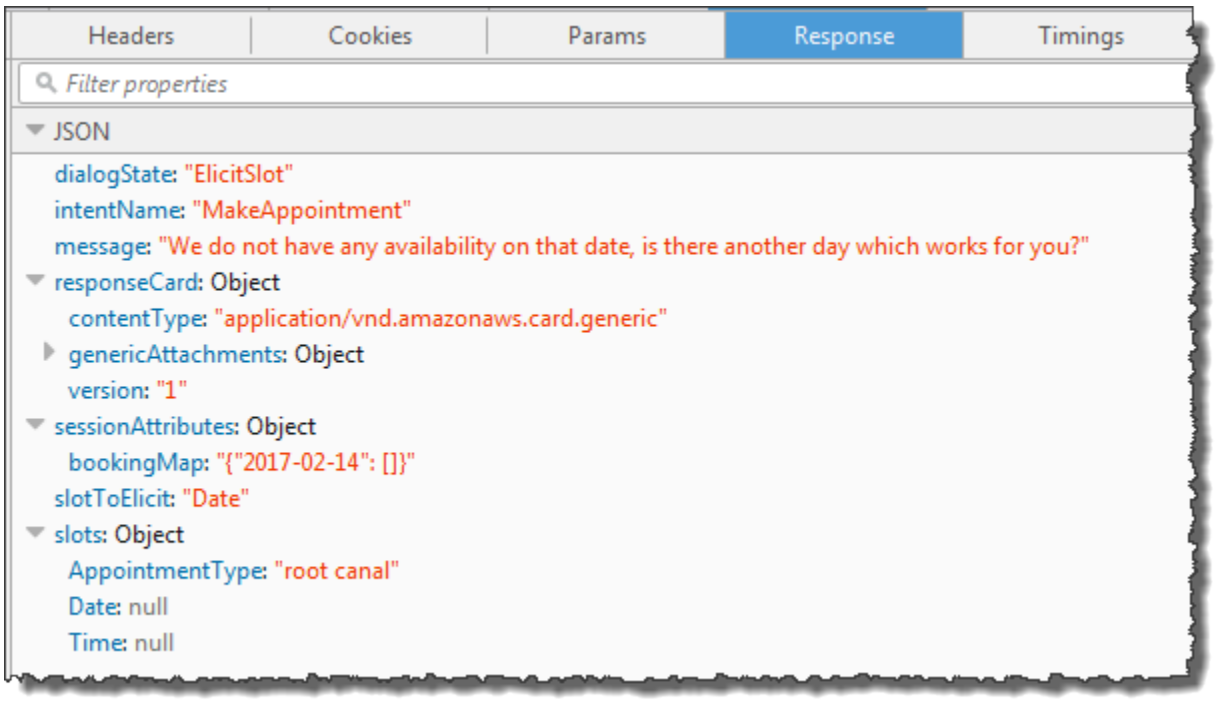
```
                            "text": "2-20 (Mon)",
                            "value": "Monday, February 20, 2017"
                        },
                        {
                            "text": "2-21 (Tue)",
                            "value": "Tuesday, February 21, 2017"
                        }
                    ],
                    "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\": []}"
   }
}
```

同樣地，回應也包含 dialogAction 和 sessionAttributes 欄位。此外，dialogAction 會傳回以下欄位：

- dialogAction 欄位：
  - type— Lambda 函數將此值設定為，ElicitSlot並將slotToElicit欄位重設為Date。Lambda 函數還提供了一個適當message的傳達給用戶。
  - responseCard – 傳回 Date 槽的值清單。

- `sessionAttributes`-這次 Lambda 函數包含`bookingMap`工作階段屬性。其值是要求的預約日期及可預約的時間 (空白物件表示沒有可預約的時間)。

d. Amazon Lex 會注意到`dialogAction.type`並將下列回應傳回給用戶端，其中包含來自 Lambda 函數回應的資訊。



用戶端會顯示訊息：We do not have any availability on that date, is there another day which works for you? 和回應卡 (如果用戶端有支援回應卡)。

4. 使用者：根據用戶端，使用者有兩個選項：

- 如果有顯示回應卡，請選擇 2-15 (Wed)，或輸入 **Wednesday**。
- 如果用戶端不支援回應卡，輸入 **Wednesday**。

a. 用戶端會將下列`PostText`要求傳送至亞馬遜萊克斯：

```
POST /bot/BookTrip/alias/$LATEST/user/bijt6rovckwecnzesbthrr1d7lv3ja3n/text
"Content-Type":"application/json"
"Content-Encoding":"amz-1.0"

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

```
}
```

> **Note**
>
> Facebook Messenger 用戶端不會設定任何工作階段屬性。如果您想要在請求之間維護工作階段狀態，則必須在 Lambda 函數中這樣做。在真實的應用程式中，您可能需要在後端資料庫中維護這些工作階段屬性。

b. Amazon Lex 透過傳送下列事件做為參數，叫用 Lambda 函數進行使用者資料驗證：

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

亞馬遜萊克斯`currentIntent.slots`通過將`Date`插槽設置為更新`2017-02-15`。

c. Lambda 函數會驗證使用者輸入，並將下列回應傳回給 Amazon Lex，並引導該函數引出約會時間的值。

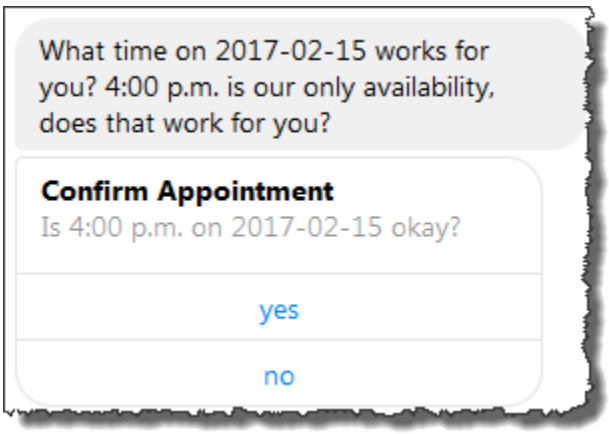
```
{
    "dialogAction": {
        "slots": {
```

```
            "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\"]}"
    }
}
```

同樣地，回應也包含 dialogAction 和 sessionAttributes 欄位。此外，dialogAction 會傳回以下欄位：

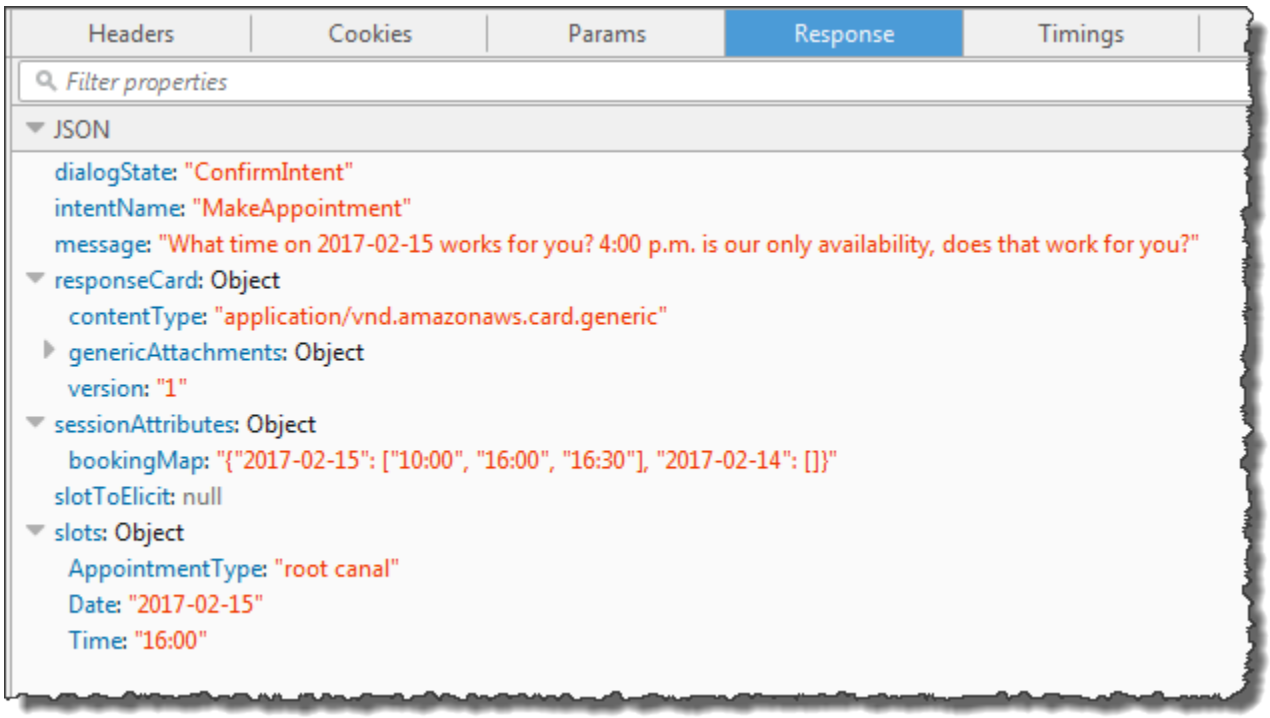
- dialogAction 欄位：

- `type`— Lambda 函數會將此值設定為`ConfirmIntent`，指示 Amazon Lex 取得使用者確認中建議的約會時間`message`。
- `responseCard`— 傳回是/否值清單，供使用者選擇。如果用戶端支援回應卡，它會顯示回應卡，如下例所示：



- `sessionAttributes`-Lambda 函數會設定`bookingMap`工作階段屬性，其值設定為約會日期，並在該日期設定可用的約會。在這個範例中，這些是 30 分鐘的預約。對於需要一個小時的根管治療，只能預訂下午 4 點。

d. 如 Lambda 函數回應`dialogAction.type`中所述，Amazon Lex 會將下列回應傳回給用戶端：



客戶端顯示消息：2017-02-15 上的什麼時間適合您？ 下午 4 點是我們唯一的可用性，這對你有用嗎？

5. 使用者：選擇 **yes**。

亞馬遜 Lex 使用下列事件資料叫用 Lambda 函數。因為使用者回覆了**yes**，Amazon Lex 會`confirmationStatus`將設定為`Confirmed`，並將`Time`欄位設定`currentIntent.slots`為`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": {
   }
}
```

由於`confirmationStatus`已確認，Lambda 函數會處理意圖 (預約牙科預約)，並將下列回應傳回給 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\"]}"
    }
}
```

注意下列事項：

- Lambda 函數已更新 `sessionAttributes`。
- `dialogAction.type`設定為`Close`，這會指示 Amazon Lex 預期不會收到使用者回應。
- `dialogAction.fulfillmentState` 設為 `Fulfilled`，指出意圖已順利達到。

客戶顯示消息：好的，我已經預約了你的約會。我們將在下午 4 點見。

預訂行程

本範例說明如何建立一個機器人，並設定為支援多個意圖。範例亦說明了如何使用工作階段屬性進行跨意圖資訊共享。建立機器人之後，您可以使用 Amazon Lex 主控台中的測試用戶端來測試機器人 (BookTrip)。用戶端會使用PostText執行階段 API 作業，針對每個使用者輸入傳送請求至 Amazon Lex。

此範例中的 BookTrip 機器人設定了兩個意圖 (BookHotel 和 BookCar)。例如，假設使用者先是預訂飯店。在互動期間，使用者提供如入住日期、位置和住宿天數等資訊。滿足意圖後，用戶端便可以使用工作階段屬性來保留此資訊。如需工作階段屬性的詳細資訊，請參閱 PostText。

現在假設使用者繼續預訂租車。使用使用者在先前 BookHotel 意圖中提供的資訊 (也就是目的地城市，以及簽入和結帳日期)，您設定為初始化和驗證 BookCar 意圖的程式碼掛接 (Lambda 函數)，初始化 BookCar 意圖的位置資料 (也就是目的地、接機城市、接機日期和返回日期)。這說明了跨意圖資訊共享如何能夠讓您建立可以與使用者進行動態對話的機器人。

在這個範例中，我們使用以下工作階段屬性。只有用戶端和 Lambda 函數可以設定和更新工作階段屬性。Amazon Lex 只會在用戶端和 Lambda 函式之間傳遞這些資訊。Amazon Lex 不會維護或修改任何工作階段屬性。

- currentReservation— 包含進行中保留的位置資料以及其他相關資訊。例如，以下是從用戶端到 Amazon Lex 的範例請求。它在請求本文中顯示 currentReservation 工作階段屬性。

```
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— 包含先前意圖的類似資訊 (如果有的話)。例如，如果使用者預訂了酒店，然後正在預訂汽車，則此工作階段屬性會儲存先前 BookHotel 意圖的插槽資料。

- confirmationContext— Lambda 函數會根據先前保留區的位置資料 (如果有的話) 預先填入某些插槽資料AutoPopulate時，將此值設定為。如此可跨意圖資訊共享。例如，如果使用者之前預訂了酒店，現在想預訂汽車，Amazon Lex 可以提示使用者確認 (或拒絕) 汽車預訂的城市和酒店預訂日期

在本練習中，您將使用藍圖建立 Amazon Lex 機器人和 Lambda 函數。如需有關藍圖的詳細資訊，請參閱 Amazon Lex 和AWS LambdaBlueprints (藍圖)。

後續步驟

步驟 1：檢閱用於此練習的藍圖

步驟 1：檢閱用於此練習的藍圖

主題

- 機器人藍圖概觀 (BookTrip)
- Lambda 函數藍圖概觀 (lex-book-trip-python)

機器人藍圖概觀 (BookTrip)

您用來建立機器人的藍圖 (BookTrip) 提供下列預先設定：

- 槽類型 – 兩個自訂槽類型：
  - RoomTypes 與列舉值：king、queen 和 deluxe，用於 BookHotel 意圖。
  - CarTypes 與列舉值：economy、standard、midsize、full size、luxury 和 minivan，用於 BookCar 意圖。

- 意圖 1（BookHotel）-它預先配置如下：
  - 預先設定的槽
    - RoomType，為 RoomTypes 自訂槽類型
    - Location，為 AMAZON.US_CITY 內建槽類型
    - CheckInDate，為 AMAZON.DATE 內建槽類型
    - Nights，為 AMAZON.NUMBER 內建槽類型
  - 預先設定的表達用語
    - 「預訂飯店」
    - 「我想預訂飯店」
    - 「在{Location}預訂 {Nights} 晚」

    如果使用者說出上述任何內容，Amazon Lex 會判斷這BookHotel是意圖，然後提示使用者輸入插槽資料。
  - 預先設定的提示
    - Location 槽的提示 –「您要在哪個城市留宿？」
    - CheckInDate 槽的提示 –「您要在哪一天入住？」

    - Nights 槽的提示 –「您要住幾晚？」
    - RoomType 槽的提示 –「您想要哪一種房型，標準雙人房、加大雙人房或豪華房？」
    - 確認聲明-「好吧，我讓你在 {位置} 開始 {} 晚住宿 {CheckInDate}。要我預訂嗎？」
    - 拒絕 –「好的，我已取消您目前的預訂」。

- 意圖 2（BookCar）-它預先配置如下：
  - 預先設定的槽
    - PickUpCity，為 AMAZON.US_CITY 內建類型
    - PickUpDate，為 AMAZON.DATE 內建類型
    - ReturnDate，為 AMAZON.DATE 內建類型
    - DriverAge，為 AMAZON.NUMBER 內建類型
    - CarType，為 CarTypes 自訂類型
  - 預先設定的表達用語
    - 「預訂租車」
    - 「預約租車」
    - 「租車預訂」

    如果使用者說出上述任何內容，Amazon Lex BookCar 就會判斷意圖，然後提示使用者輸入插槽資料。
  - 預先設定的提示
    - PickUpCity 槽的提示 –「您需要在哪個城市租車？」
    - PickUpDate 槽的提示 –「您要在哪一天開始租車？」
    - ReturnDate 槽的提示 –「您要在哪一天還車？」
    - DriverAge 槽的提示 –「此租車的駕駛幾歲？」
    - 提示輸入CarType插槽-「您想租用哪種類型的汽車？ 我們最受歡迎的選項是經濟型、標準型及豪華型」
    - 確認聲明-「好的，我讓你在 {CarType} 中從 {} 到 {PickUpCity} 中的 {PickUpDateReturnDate} 租賃。要我預訂嗎？」
    - 拒絕 –「好的，我已取消您目前的預訂」。

Lambda 函數藍圖概觀 (lex-book-trip-python)

除了機器人藍圖之外，還AWS Lambda提供一個藍圖 (lex-book-trip-python)，您可以將其用作機器人藍圖的程式碼掛接。如需機器人藍圖和對應 Lambda 函數藍圖的清單，請參閱Amazon Lex 和AWS LambdaBlueprints (藍圖)。

當您使用 BookTrip 藍圖建立機器人時，您可以將此 Lambda 函數新增為程式碼掛接，以同時初始化/驗證使用者資料輸入和實現意圖，以更新意圖 (BookCar 和 BookHotel) 的組態。

提供的這個 Lambda 函數程式碼展示了使用先前已知的資訊 (關於使用者初始化意圖的槽值，保留在工作階段屬性中) 進行動態對談。如需詳細資訊，請參閱管理對話內容。

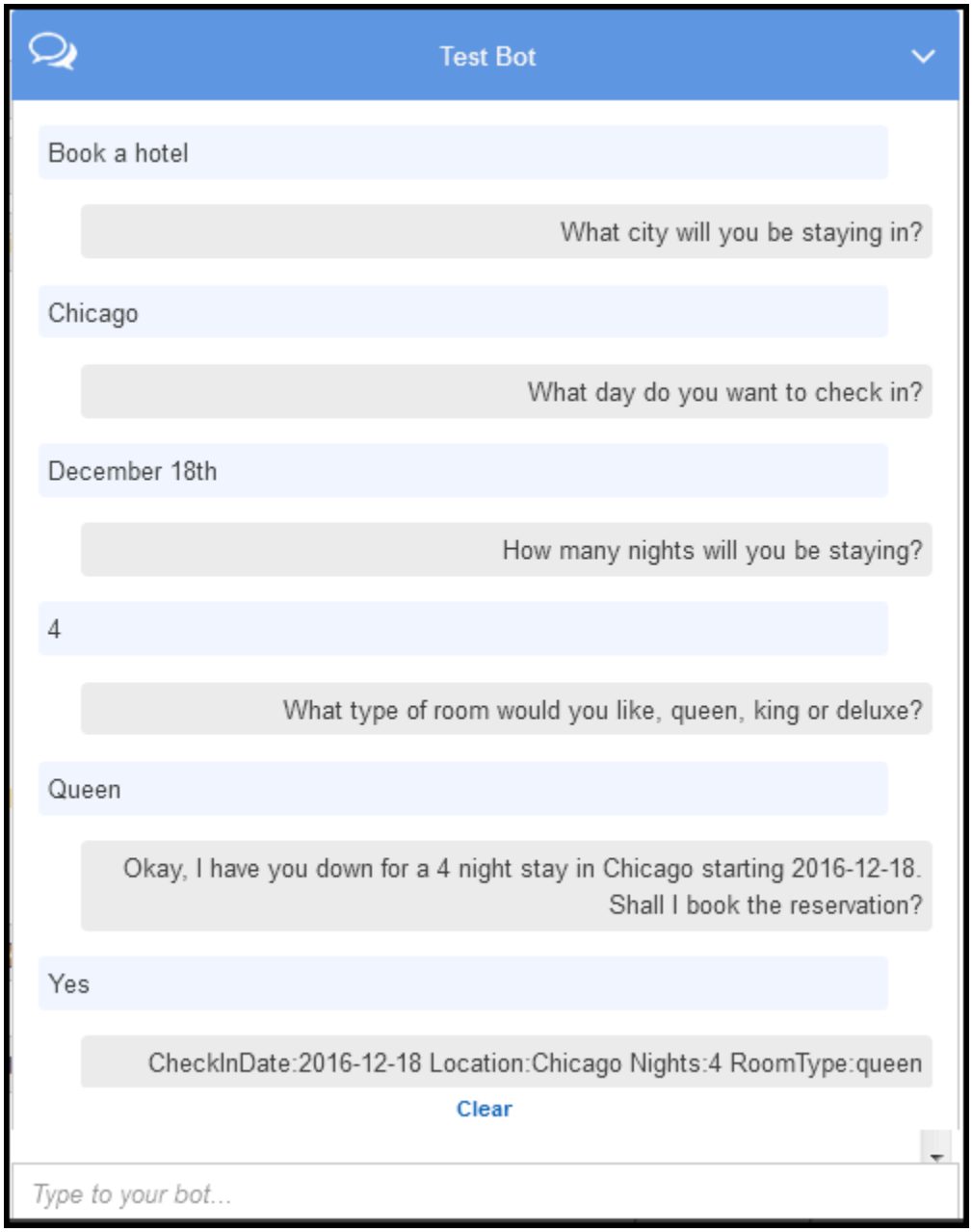
後續步驟

步驟 2：建立 Amazon Lex 機器人

步驟 2：建立 Amazon Lex 機器人

在本節中，您將建立 Amazon Lex 機器人 (BookTrip)。

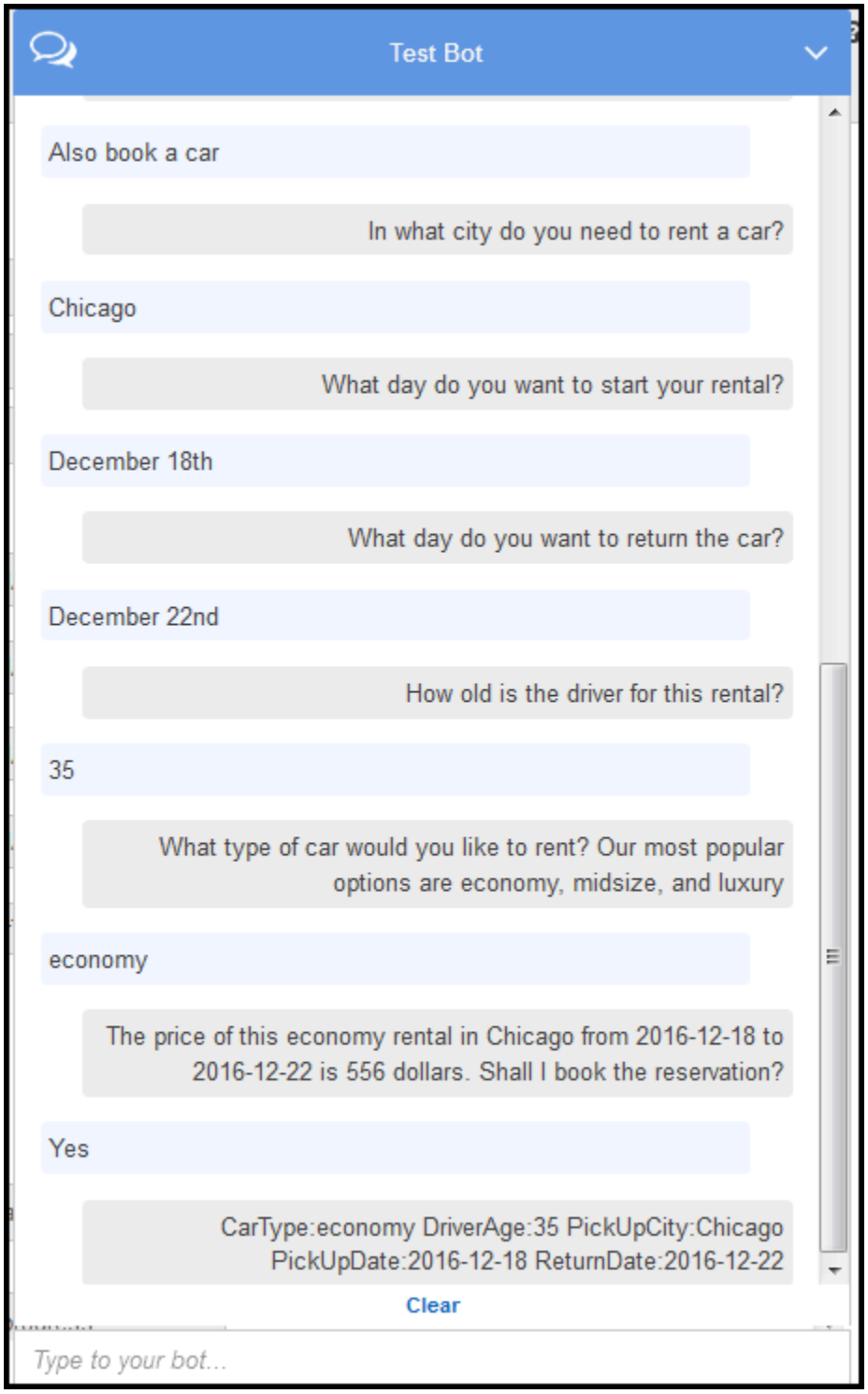
1. 登入，AWS Management Console並在 https://console.aws.amazon.com/lex/ 開啟 Amazon Lex 主控台。
2. 在 Bots (機器人) 頁面，選擇 Create (建立)。
3. 在 Create your Lex bot (建立您的機器人) 頁面，
   - 選擇BookTrip藍圖。
   - 保留預設機器人名稱 (BookTrip)。
4. 選擇 Create (建立)。主控台會傳送一系列請求給 Amazon Lex 以建立機器人。注意下列事項：
5. 控制台顯示 BookTrip 機器人。在編輯器索引標籤上，檢閱預先設定的對應方式 (BookCar 和 BookHotel) 的詳細資料。
6. 在測試視窗中測試機器人。使用下圖與您的機器人進行測試對話：



從最初的用戶輸入（「預訂酒店」），Amazon Lex 推斷意圖（BookHotel）。機器人之後使用在此意圖中預先設定的提示，向使用者引出槽資料。使用者提供所有插槽資料後，Amazon Lex 會將回應傳回給用戶端，其中包含所有使用者輸入作為訊息的訊息。用戶端在回應中顯示訊息，如下所示。

```
CheckInDate:2016-12-18 Location:Chicago Nights:5 RoomType:queen
```

現在，您將繼續對話，並嘗試在接下來的對話中預訂汽車。



請注意，

- 此時並沒有使用者資料驗證。例如，您可以提供任何城市來預訂飯店。

- 您再次提供一些相同的資訊 (目的地、城市、取車城市、取車日期和還車日期) 來預訂租車。在動態對話中，您的機器人應該根據之前使用者預訂飯店所提供的輸入，初始化這當中的部分資訊。

在下節，您可以建立 Lambda 函數透過工作階段屬性，使用跨意圖資訊共享來執行一些使用者資料驗證及初始化。您接著新增 Lambda 函數做為程式碼掛勾，來執行使用者輸入的初始化/驗證及滿足意圖，藉以更新意圖組態。

後續步驟

步驟 3：建立 Lambda 函式

步驟 3：建立 Lambda 函式

在本節中，您將使用AWS Lambda主控台中提供的藍圖 (lex-book-trip-python) 建立 Lambda 函數。您也可以使用主控台提供的範例事件資料叫用 Lambda 函數來測試它。

這個 Lambda 函數是用 Python 編寫的。

1. 請登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/lambda/ 的 AWS Lambda 主控台。
2. 選擇 Create function (建立函數)。
3. 選擇 Use a blueprint (使用藍圖)。輸入 **lex** 來尋找藍圖，選擇 `lex-book-trip-python` 藍圖。
4. 選擇設定 Lambda 函數，如下所示。
   - 輸入 Lambda 函數名稱 (`BookTripCodeHook`)。
   - 對於角色，選擇 Create a new role from template(s) (從範本建立新角色)，然後輸入角色名稱。
   - 保留其他預設值。
5. 選擇 Create function (建立函數)。
6. 如果您使用的是英文 (US) (en-US) 以外的地區設定，請依照中所述更新意圖名稱更新特定區域設置的藍圖。
7. 測試 Lambda 函式。您可以使用範例資料來預訂汽車和預訂酒店兩次叫用 Lambda 函數。
   a. 從 [選取測試事件] 下拉式清單中選擇 [設定測試事件]。
   b. 從範例活動範本清單中選擇 Amazon Lex 書籍飯店。

此範例事件符合 Amazon Lex 的請求/回應模型。如需詳細資訊，請參閱使用 Lambda 函數。

c. 選擇 Save and test (儲存並測試)。

d. 確認 Lambda 函式是否成功執行，在這種情況下，回應與 Amazon Lex 回應模型相符。

e. 重複步驟。這次您從範例活動範本清單中選擇 Amazon Lex 書車。Lambda 函數會處理汽車預約。

後續步驟

步驟 4：將 Lambda 函數新增為程式碼掛接

步驟 4：將 Lambda 函數新增為程式碼掛接

在本節中，您可以將 Lambda 函數新增為初始化/驗證 BookCar 和履行活動的程式碼掛接，以更新和BookHotel意圖的組態。請務必選擇意圖的 $LAST 版本，因為您只能更新 Amazon Lex 資源的 $LAST 版本。

1. 在 Amazon Lex 主控台中，選擇BookTrip機器人。
2. 在編輯器索引標籤上，選擇BookHotel意圖。依以下方式更新意圖組態：

   a. 確定意圖版本 (意圖名稱) 為 $LATEST。

   b. 將 Lambda 函數新增為初始化和驗證程式碼掛接，如下所示：

      - 在 Options (選項) 中，選擇 Initialization and validation code hook (初始化和驗證程式碼掛勾)。
      - 從清單中選擇 Lambda 函式。

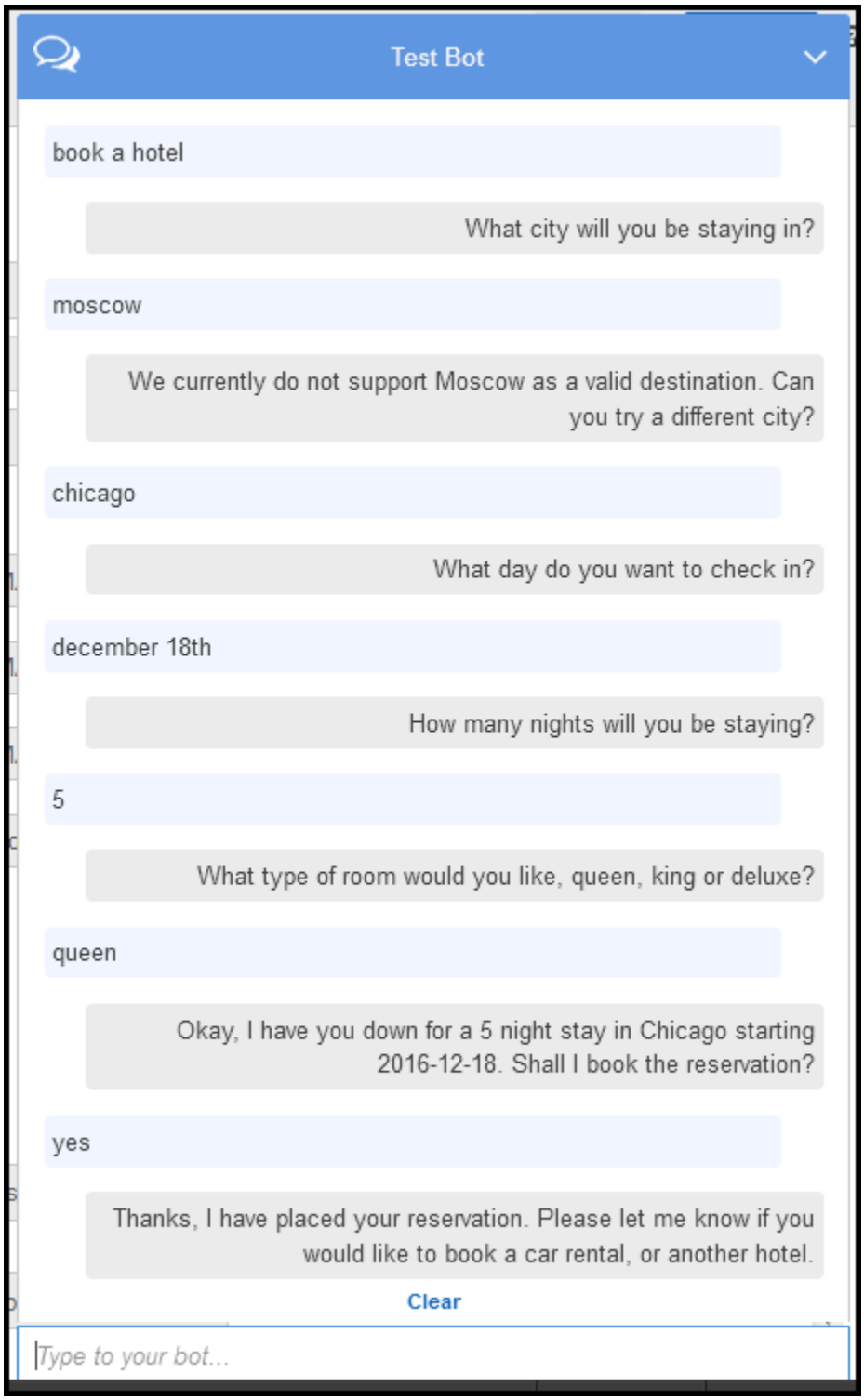
   c. 新增 Lambda 函式作為履行程式碼掛接，如下：

      - 在 Fulfillment (履行)，選擇 AWS Lambda function (AWS Lambda 函數)。
      - 從清單中選擇 Lambda 函式。
      - 選擇 Goodbye message (再見訊息) 並輸入訊息。

   d. 選擇 Save (儲存)。
3. 在編輯器索引標籤上，選擇 BookCar 意圖。按照上述步驟將您的 Lambda 函數新增為驗證和履行程式碼掛勾。

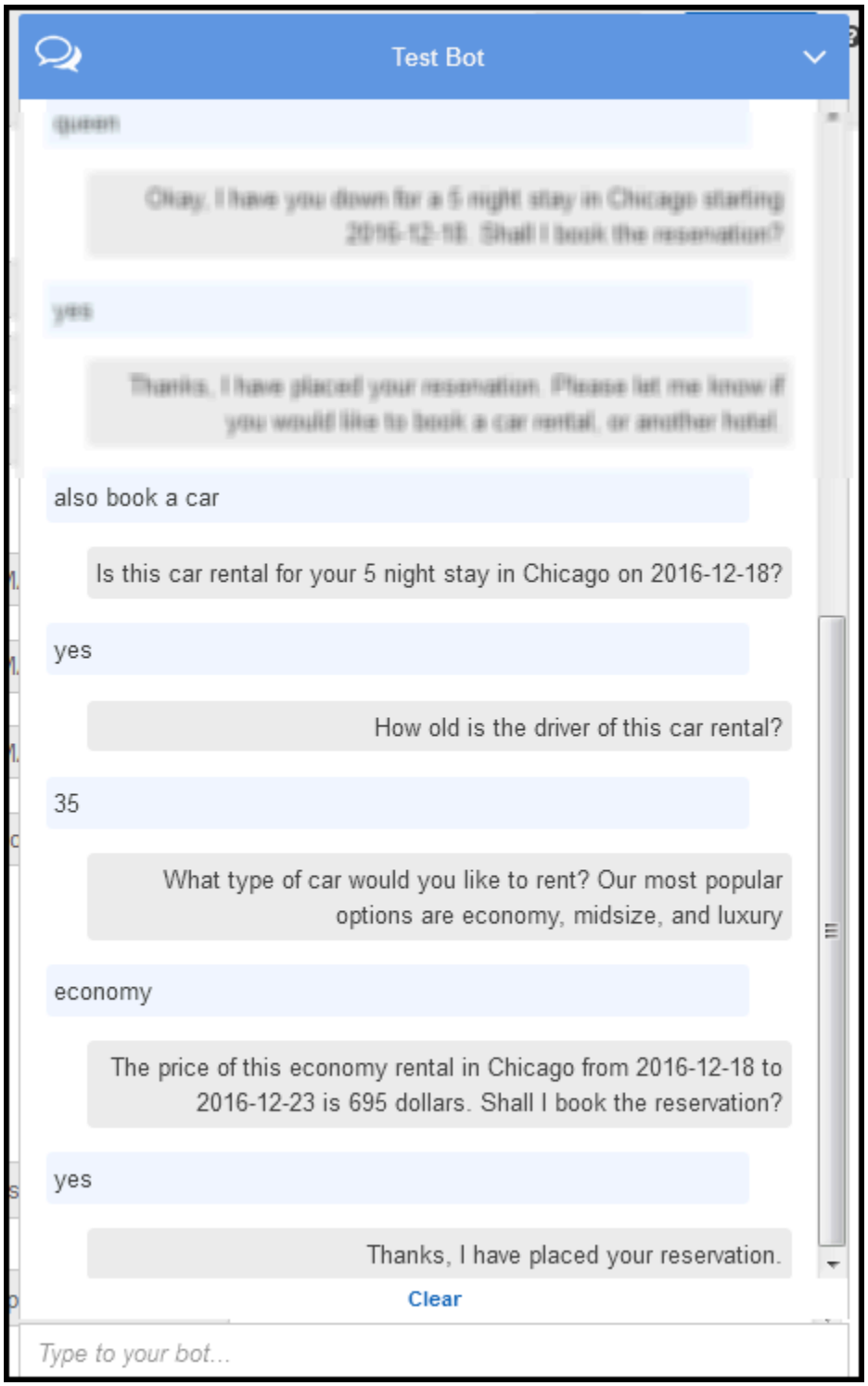
4. 選擇 Build (建置)。主控台會傳送一系列請求至 Amazon Lex 以儲存組態。

5. 測試機器人。現在，您有一個執行初始化、使用者資料驗證和履行的 Lambda 函數，您可以在下列對話中看到使用者互動的差異：



如需從用戶端 (主控台) 到 Amazon Lex 的資料流程，以及從 Amazon Lex 到 Lambda 函數的資料流程的詳細資訊，請參閱資料流程：預訂飯店意圖。

6. 繼續對話，並預訂汽車，如下圖所示：



當您選擇預訂汽車時，用戶端 (主控台) 會向 Amazon Lex 傳送要求，其中包含工作階段屬性 (來自上一個交談 BookHotel)。Amazon Lex 會將此資訊傳遞給 Lambda 函數，然後初始化 (也就是預先填入) 部分 BookCar 插槽資料 (也就是 PickUpDate ReturnDate、和 PickUpCity)。

> **Note**
>
> 這說明了如何利用工作階段屬性跨意圖來保持內容。主控台用戶端在測試視窗中提供 Clear (清除) 連結，使用者可以用此連結來清除任何之前的工作階段屬性。

如需從用戶端 (主控台) 到 Amazon Lex 的資料流程，以及從 Amazon Lex 到 Lambda 函數的資料流程的詳細資訊，請參閱資料流程：預訂租車意圖。

資訊流程的詳細資訊

在本練習中，您使用 Amazon L BookTrip ex 主控台提供的測試視窗用戶端與 Amazon Lex 機器人進行對話。本節說明下列各項：

- 用戶端和 Amazon Lex 之間的資料流。

  本節假設用戶端使用`PostText`執行階段 API 將請求傳送至 Amazon Lex，並據此顯示請求和回應詳細資訊。如需有關 `PostText` 執行時間 API 的詳細資訊，請參閱 PostText。

  > **Note**
  >
  > 如需用戶端和用戶端使用`PostContent` API 的 Amazon Lex 之間的資訊流程範例，請參閱步驟 2a (選用)：檢閱口語化資訊流程的詳細資訊 (主控台) 。

- Amazon Lex 和 Lambda 函數之間的數據流。如需詳細資訊，請參閱Lambda 函數輸入事件和回應格式。

主題

- 資料流程：預訂飯店意圖

- 資料流程：預訂租車意圖

資料流程：預訂飯店意圖

本節說明在使用者每次輸入之後，會發生什麼情況。

1. 使用者：「預訂飯店」

   a. 用戶端 (主控台) 傳送以下 PostText 請求給 Amazon Lex：

   ```
   POST /bot/BookTrip/alias/$LATEST/user/wch89kjqcpkds8seny7dly5x3otq68j3/text
   "Content-Type":"application/json"
   "Content-Encoding":"amz-1.0"

   {
      "inputText":"book a hotel",
      "sessionAttributes":{}
   }
   ```

   請求 URI 和本文都向 Amazon Lex 提供信息：

   - 請求 URI — 提供機器人名稱 (BookTrip)、機器人別名 ($LATEST) 和使用者名稱。末尾的 `text` 表示其為 `PostText` API 請求 (而非 `PostContent`)。
   - 請求本文 – 包含使用者輸入 (`inputText`) 和空的 `sessionAttributes`。最初，這是一個空對象和 Lambda 函數首先設置會話屬性。

   b. 從中`inputText`，Amazon Lex 檢測到意圖（BookHotel）。此目的是使用 Lambda 函數設定為使用者資料初始化/驗證的程式碼掛接。因此，Amazon Lex 會傳遞下列資訊做為事件參數來叫用該 Lambda 函數 (請參閱輸入事件格式)：

   ```
   {
      "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"
   }
}
```

除了用戶端傳送的資訊之外，Amazon Lex 還包含下列其他資料：

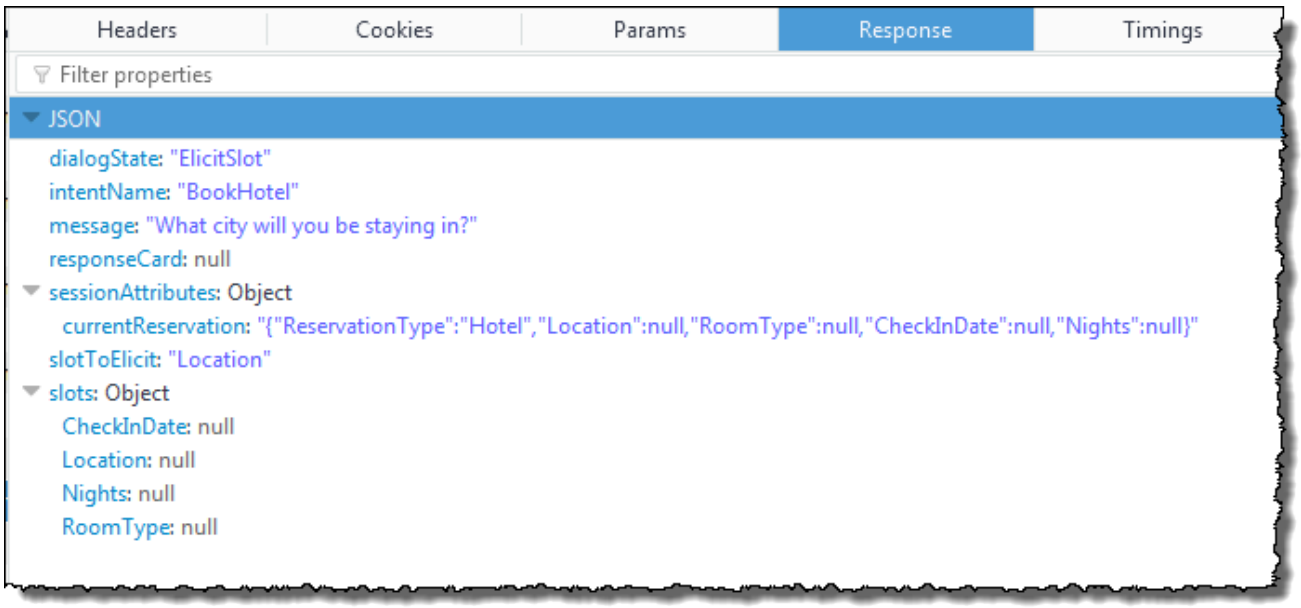
- `messageVersion`— 目前 Amazon Lex 僅支持 1.0 版本。
- `invocationSource`— 指出 Lambda 函數叫用的目的。在這種情況下，它是執行使用者資料初始化和驗證 (目前 Amazon Lex 知道使用者尚未提供所有插槽資料來達成意圖)。
- `currentIntent` – 所有槽值設定為 null。

c. 此時，所有槽值都是 null。Lambda 函數沒有任何需要驗證的項目。Lambda 函數返回以下響應 Amazon Lex。如需有關回應格式的資訊，請參閱回應格式。

```
{
   "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`— Lambda 函數包含此工作階段屬性。其值是目前槽資訊和預訂類型的副本。
>
> 只有 Lambda 函數和用戶端可以更新這些工作階段屬性。Amazon Lex 只是通過這些值。
> - `dialogAction.type`— 將此值設定為`Delegate`，Lambda 函數會將下一個動作過程的責任委派給 Amazon Lex。
>
> 如果 Lambda 函數在使用者資料驗證中偵測到任何內容，它會指示 Amazon Lex 下一步該怎麼做。

d. 根據`dialogAction.type`，Amazon Lex 決定下一個操作過程-從用戶那裡獲取`Location`插槽的數據。它根據意圖組態選擇其中一個提示訊息 (「您會待在哪個城市？」) 給這個槽，然後傳送以下回應使用者：



工作階段屬性傳遞給用戶端。

用戶端讀取回應，然後顯示「您要在哪個城市留宿？」

2. 使用者：「莫斯科」

a. 用戶端會將下列`PostText`要求傳送至 Amazon Lex (新增分行符號以提高可讀性)：

```
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}"
   }
}
```

除了 inputText 之外，用戶端還包含了它所收到的相同 currentReservation 工作階段屬性。

b. Amazon Lex 首先inputText在目前意圖的內容中解譯 (該服務會記得已向特定使用者詢問有關Location插槽的資訊)。它會更新目前意圖的插槽值，並使用下列事件叫用 Lambda 函數：

```
{
    "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` 繼續做為 `DialogCodeHook`。在此步驟中，我們只需驗證使用者資料。
> - Amazon Lex 只是將會話屬性傳遞給 Lambda 函式。
> - 對於`currentIntent.slots`, Amazon Lex 已經更新了`Location`插槽`Moscow`.

c. Lambda 函數會執行使用者資料驗證，並判斷該`Moscow`位置無效。

> **Note**
>
> 本練習中的 Lambda 函數有簡單的有效城市清`Moscow`單，不在清單中。在生產應用程式中，您可以使用後端資料庫來取得此資訊。

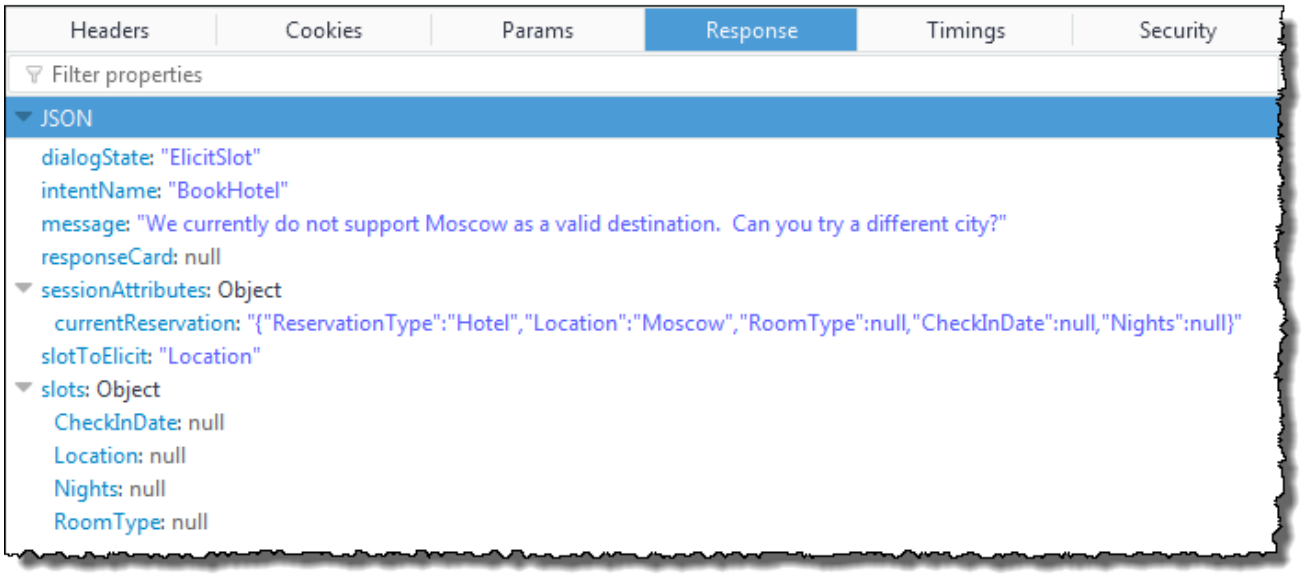
它會將插槽值重設回 null，並透過傳送下列回應，指示 Amazon Lex 再次提示使用者輸入另一個值：

```
{
    "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 重設為 null。
> - dialogAction.type設定為ElicitSlot，可指示 Amazon Lex 透過提供下列資訊再次提示使用者：
>   - dialogAction.slotToElicit – 向使用者引出資料的槽。
>   - dialogAction.message – 傳達給使用者的 message。

d. Amazon Lex 會在下列回應中通知dialogAction.type並將資訊傳遞給用戶端：



用戶端直接顯示訊息：「莫斯科目前不是我們支援的有效目的地。您可以嘗試不同的城市嗎？」

3. 使用者：「芝加哥」

   a. 用戶端會將下列PostText要求傳送至 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 知道上下文，它正在引出Location插槽的數據。在此情況下，服務知道 inputText 值是用於 Location 槽。然後，它會傳送下列事件來叫用 Lambda 函數：

```
{
    "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"
```

```
    }
}
```

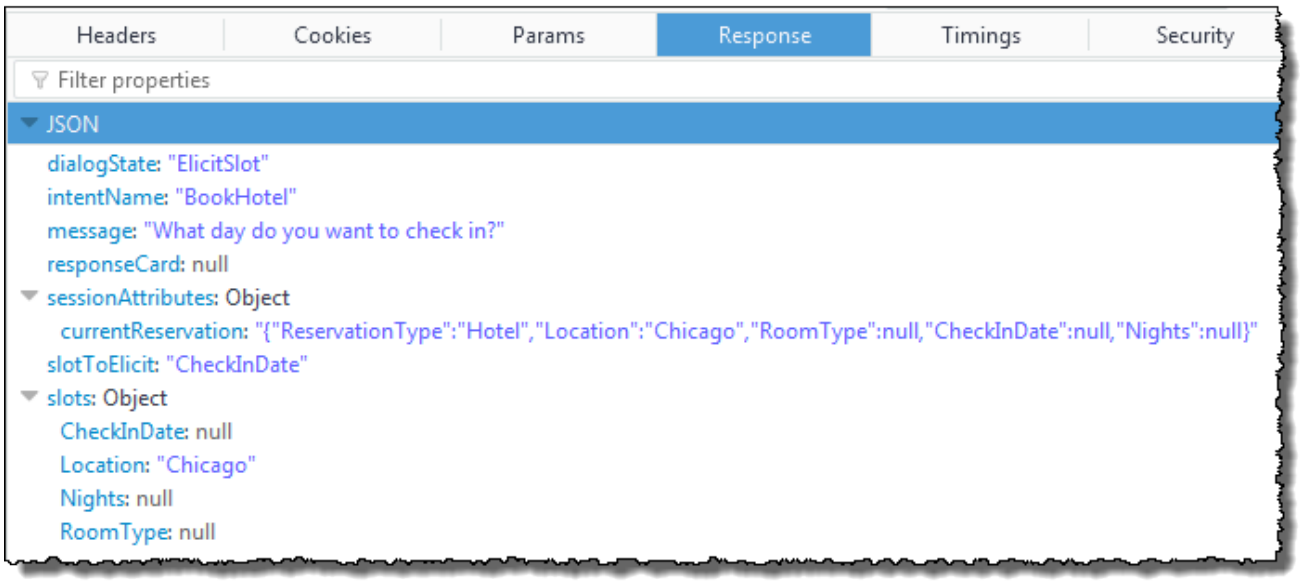
Amazon LexcurrentIntent.slots 通過將Location插槽設置為更新了Chicago。

c. 根據的invocationSource值DialogCodeHook，Lambda 函數會執行使用者資料驗證。它會辨識Chicago為有效的插槽值、相應地更新工作階段屬性，然後將下列回應傳回 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— Lambda 函數會透過Location將設定為來更新此工作階段屬性Chicago。
> - dialogAction.type – 已設定為 Delegate。使用者資料有效，而 Lambda 函數會指示 Amazon Lex 選擇下一個動作方案。

d. 據說dialogAction.type，Amazon Lex 選擇行動的下一個過程。Amazon Lex 知道需要更多插槽資料，並根據意圖組態挑選具有最高優先順序的下一個未填充的插槽 (CheckInDate)。它根據意圖組態選擇其中一個提示訊息 (「您入住的日期是哪一天？」) 給這個槽，然後將以下回應傳回給使用者：

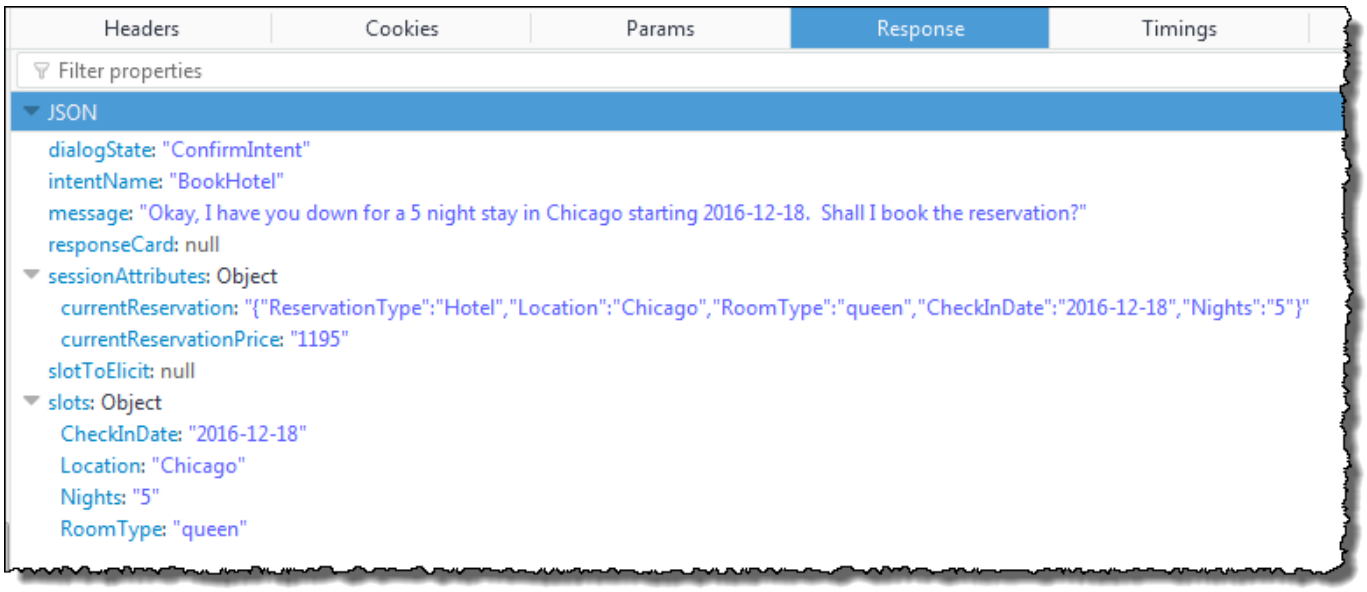


用戶端顯示訊息：「您要在哪一天入住？」

4. 使用者互動會繼續：使用者提供資料、Lambda 函數驗證資料，然後將下一個動作過程委派給 Amazon Lex。最後，使用者會提供所有插槽資料，Lambda 函數會驗證所有使用者輸入，然後 Amazon Lex 會辨識出它具有所有插槽資料。

> **Note**
>
> 在本練習中，在使用者提供所有插槽資料之後，Lambda 函數會計算飯店預訂的價格，並將其傳回為另一個工作階段屬性 (`currentReservationPrice`)。

此時，意圖已準備就緒，但是在 Amazon Lex 達成 BookHotel意圖之前，已設定確認提示需要使用者確認的確認提示。因此，Amazon Lex 會在預訂酒店前向客戶傳送下列訊息，要求確認：



用戶端顯示訊息：「好的，您要在芝加哥預訂 5 晚，從 2016-12-18 開始。要我預訂嗎？」

5. 使用者：「是」

   a. 用戶端會將下列PostText要求傳送至 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 會inputText在確認目前意圖的內容中解譯。Amazon Lex 瞭解使用者想要繼續進行保留。這次 Amazon Lex 會透過傳送下列事件來呼叫 Lambda 函數來完成意圖。透過在事件FulfillmentCodeHook中invocationSource將設定為，它會傳送至 Lambda 函數。Amazon Lex 也設置confirmationStatus為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`— 這次，Amazon Lex 將此值設置為`FulfillmentCodeHook`，指導 Lambda 函數以實現意圖。
> - `confirmationStatus` – 已設定為 `Confirmed`。

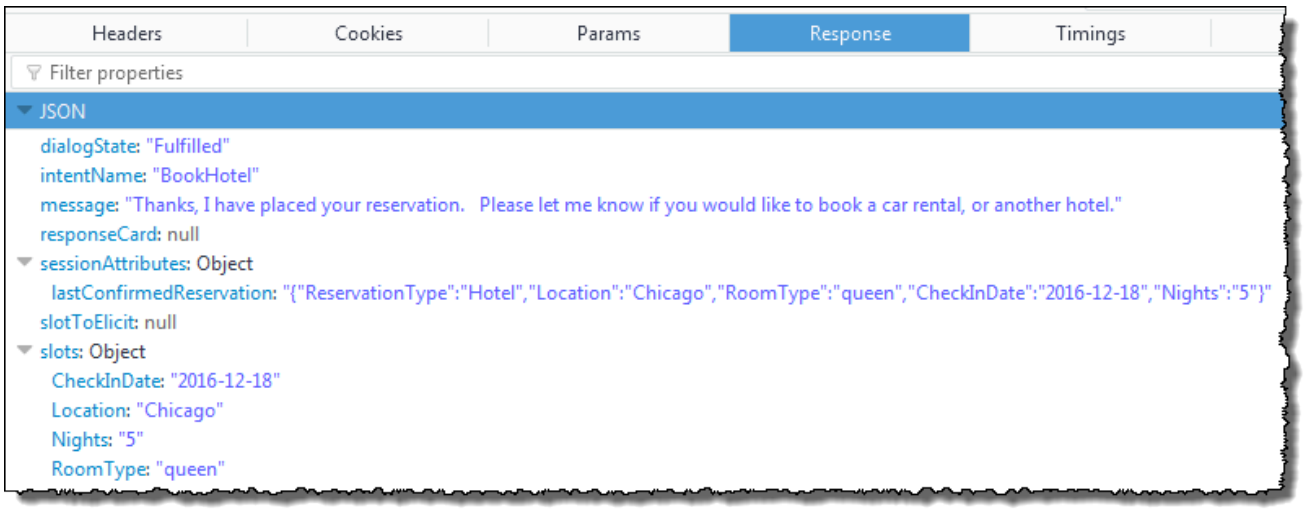
c. 這一次，Lambda 函數完成了這個 BookHotel 意圖，Amazon Lex 完成保留，然後傳回下列回應：

```
{
    "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`— 是 Lambda 函數新增的新工作階段屬性 (而不是`currentReservation`,`currentReservationPrice`)。
> - `dialogAction.type`— Lambda 函數會將此值設定為`Close`，表示 Amazon Lex 預期不會收到使用者回應。
> - `dialogAction.fulfillmentState` – 已設定為 `Fulfilled` 並包含適當的 `message` 以傳達給使用者。

d. Amazon Lex 會檢閱`fulfillmentState`並將下列回應傳送給用戶端：



> **Note**
>
> - `dialogState`— Amazon Lex 將此值設置為`Fulfilled`.
> - `message`-與 Lambda 函數提供的消息相同。

用戶端將顯示該訊息。

資料流程：預訂租車意圖

本練習中的 BookTrip 機器人支援兩種意圖 (BookHotel 和 BookCar)。預訂飯店之後，使用者可以繼續預訂租車的對話。只要工作階段未逾時，用戶端就會在每個後續的請求中繼續傳送工作階段屬性 (在這個範例中，即 `lastConfirmedReservation`)。Lambda 函數可以使用此資訊來初始化 BookCar 意圖的插槽資料。這說明您如何能夠在跨意圖資料分享中使用工作階段屬性。

具體來說，當使用者選擇 BookCar 意圖時，Lambda 函數會使用工作階段屬性中的相關資訊來預先填入 BookCar 意圖的位置 (PickUpDate ReturnDate、和 PickUpCity)。

> **Note**
>
> Amazon Lex 主控台提供清除連結，您可以用來清除任何先前的工作階段屬性。

按照此程序中的步驟，繼續對話。

1. 使用者：「還要預訂租車」

   a. 客戶端將以下`PostText`請求發送到 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\"}"
   }
}
```

用戶端包含 `lastConfirmedReservation` 工作階段屬性。

b. Amazon Lex 從中偵測到意圖 (BookCar)`inputText`。此意圖也會設定為叫用 Lambda 函數來執行使用者資料的初始化和驗證。Amazon Lex 會使用 Lex 使用 Lambda 函式，並發生下圖：

```
{
    "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"
    }
}
```

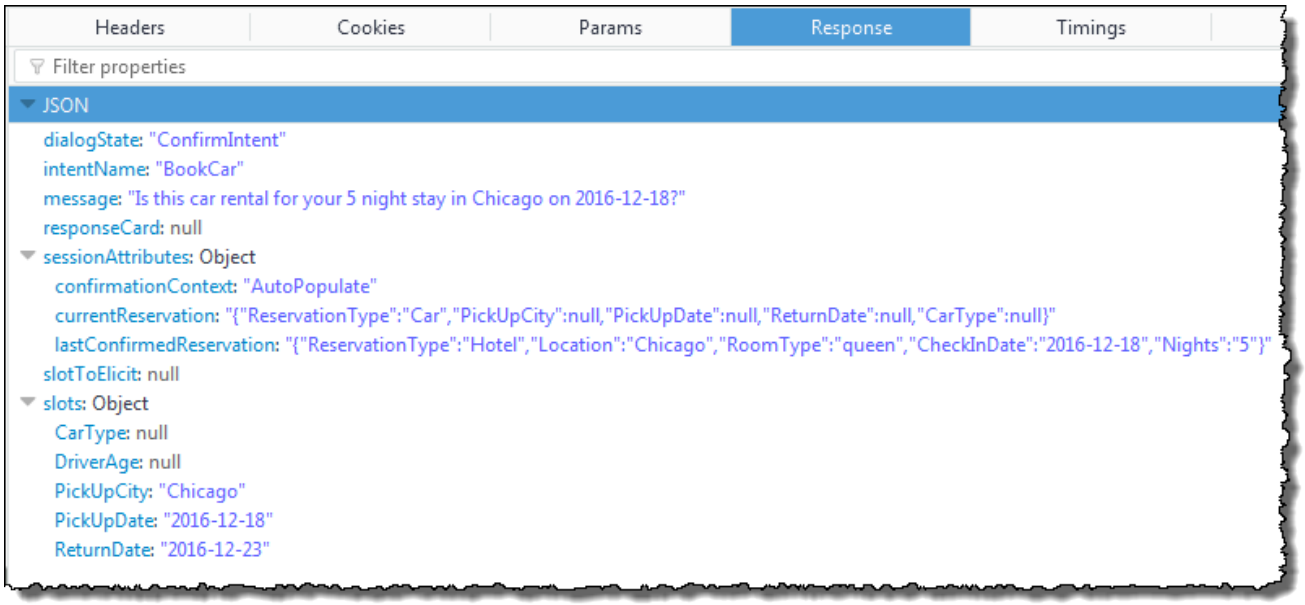
> **Note**
>
> - `messageVersion`— 目前 Amazon Lex 僅支持 1.0 版本。
> - `invocationSource` – 指出呼叫的目的是執行初始化和使用者資料驗證。
> - `currentIntent`— 它包括意圖名稱和插槽。此時，所有插槽值都是 null。

c. Lambda 函數會注意到所有空插槽值，而不需要驗證任何內容。然而，它使用工作階段屬性來初始化一些槽值 (`PickUpDate`、`ReturnDate` 和 `PickUpCity`)，然後傳回以下回應：

```
{
    "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**
>
> - 除此之外`lastConfirmedReservation`，Lambda 函數還包含更多工作階段屬性 (`currentReservation`和`confirmationContext`)。
> - `dialogAction.type`設定為`ConfirmIntent`，這會通知 Amazon Lex 使用者預期會收到「是」、「不」回覆 (confirmationContext 設定為 AutoPopulate，Lambda 函數知道是/否使用者回覆是要取得執行 Lambda 函數初始化的使用者確認 (auto 填入的插槽資料)。
>
> Lambda 函數也會在回應中包含資訊訊息，以`dialogAction.message`便 Amazon Lex 傳回給用戶端。
>
> > **Note**
> >
> > `ConfirmIntent` 一詞 (`dialogAction.type` 的值) 不與任何機器人意圖相關。在此範例中，Lambda 函數使用此術語來指示 Amazon Lex 從使用者取得「是/否」回覆。

d. 根據`dialogAction.type`，Amazon Lex 將以下響應返回給客戶端：



用戶端顯示訊息：「此次租車是用於您在 2016-12-18 於芝加哥留宿 5 晚嗎？」

2. 使用者：「是」

   a. 客戶端將以下PostText請求發送到 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 讀取inputText並且知道上下文 (要求使用者確認 auto 填入)。Amazon Lex 會透過傳送 Lambda 函式，藉由傳送下圖所示。

```
{
    "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"
    }
}
```

因為用戶回答是，Amazon Lex 設置`confirmationStatus`為`Confirmed`。

c. 從中`confirmationStatus`，Lambda 函數知道預先填入的值是正確的。Lambda 函數會執行下列動作：

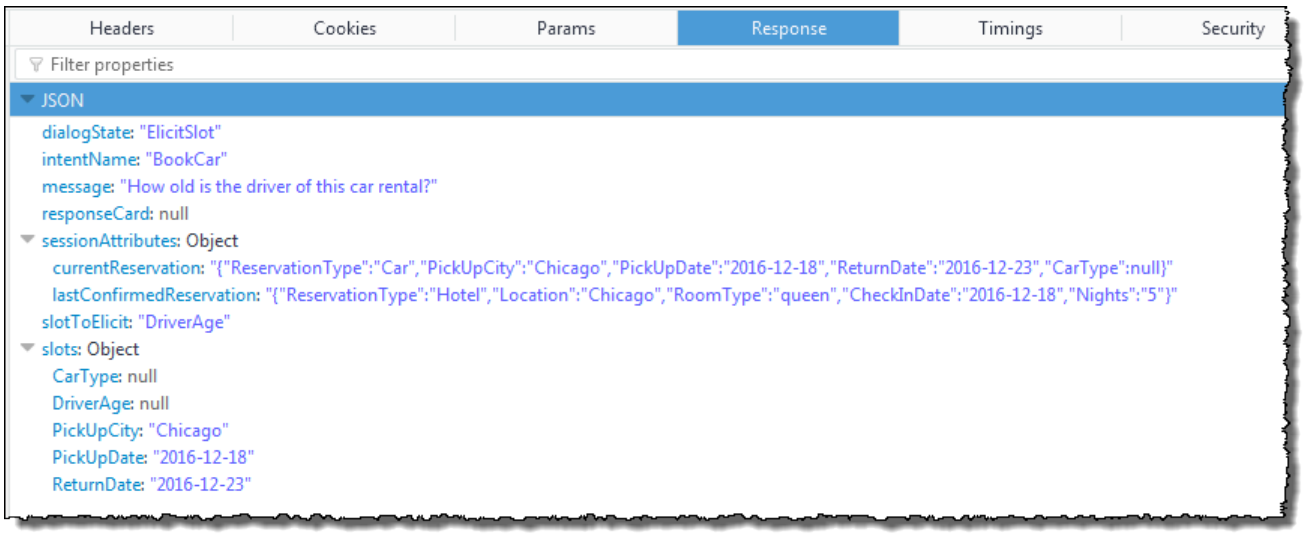
- 以它預先填入的槽值更新 `currentReservation` 工作階段屬性。
- 將 `dialogAction.type` 設定為 `ElicitSlot`
- 將 `slotToElicit` 值設定為 `DriverAge`。

傳送以下回應：

```
{
    "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 返回以下響應：



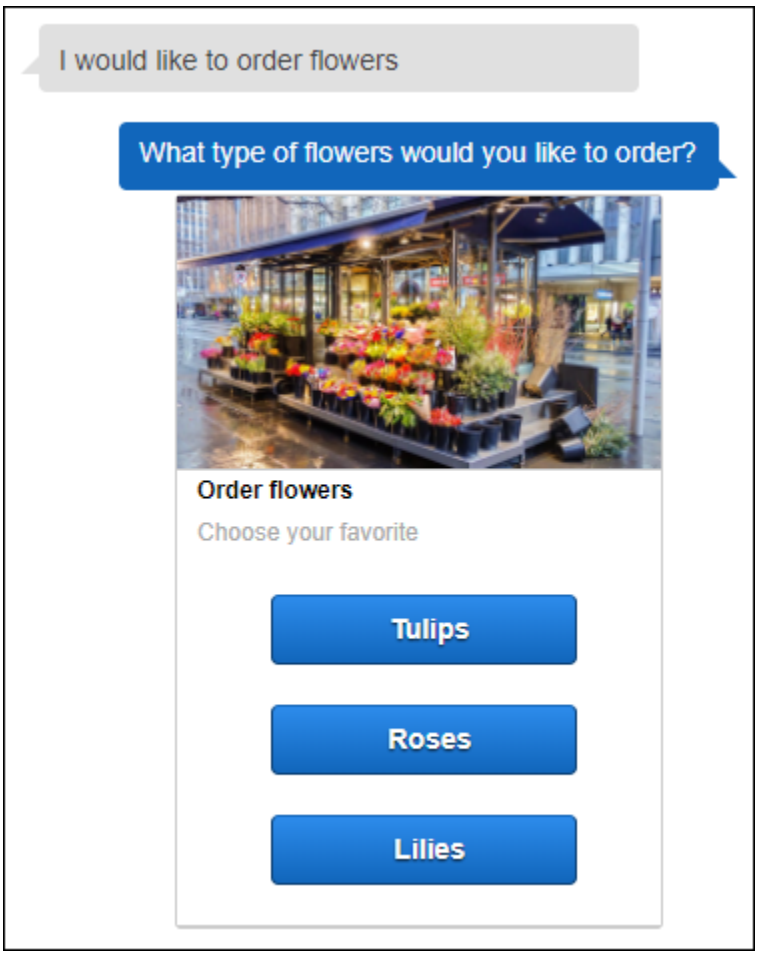
客戶端顯示消息「這輛租車的司機幾歲？」 而對談繼續。

使用回應卡

在本練習中，您將透過加入回應卡擴展入門練習 1。您可以建立支援 OrderFlowers 意圖的機器人，然後為FlowerType插槽新增回應卡來更新意圖。FlowerType 槽除了有以下的提示外，使用者還能從回應卡選擇花種：

```
What type of flowers would you like to order?
```

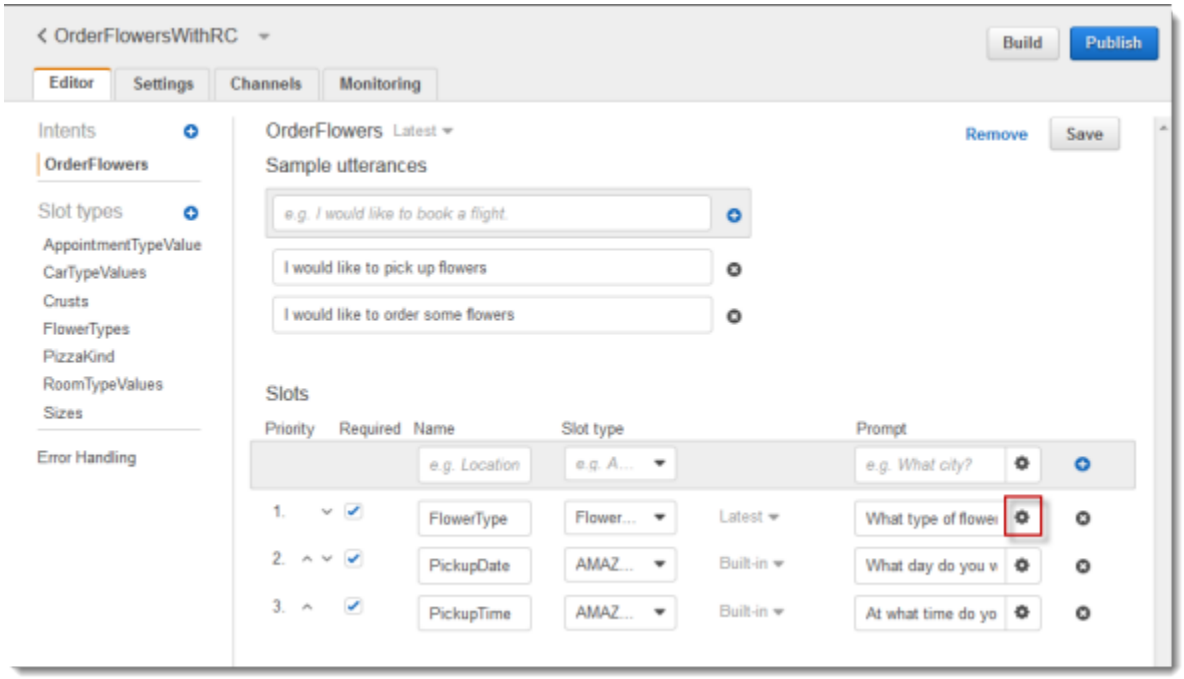
回應卡如下所示：



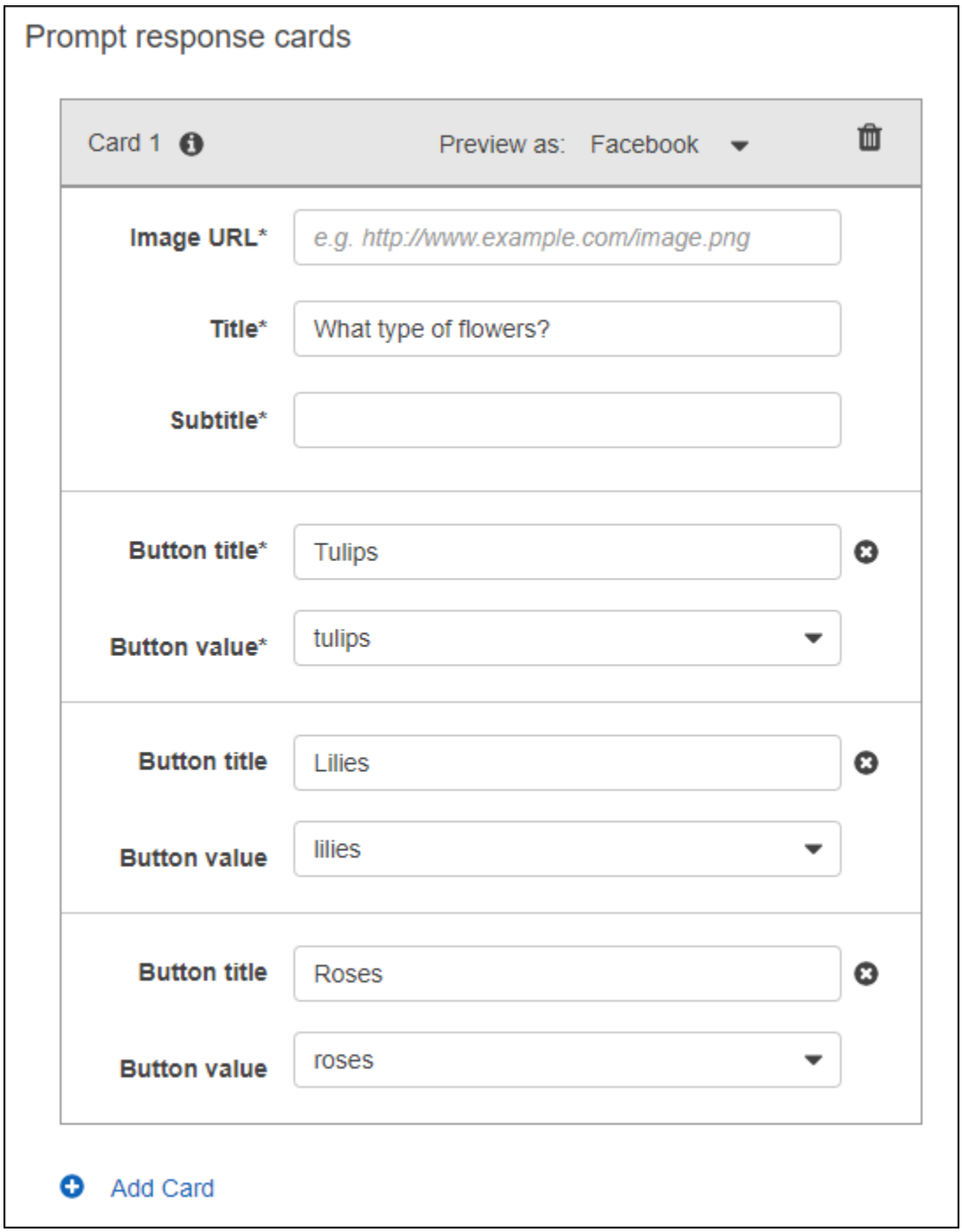
機器人使用者可以輸入文字或從花種清單中選擇。此回應卡有設定圖像，其將如下所示出現在用戶端中。如需回應卡的詳細資訊，請參閱回應卡。

使用回應卡建立及測試機器人：

1. 遵循入門練習 1 來建立和測試 OrderFlowers 機器人。您必須完成步驟 1、2 和 3。您不需要新增 Lambda 函數來測試回應卡。如需相關指示，請參閱練習 1：使用藍圖 (主控台) 建立 Amazon Lex 機器人。
2. 透過加入回應卡更新機器人，然後發佈版本。發佈版本時，指定別名 (BETA) 以指向該版本。
   a. 在 Amazon Lex 主控台中，選擇您的機器人。
   b. 選擇 `OrderFlowers` 意圖。
   c. 選擇「什麼類型的花」提示旁的設置齒輪圖標，以配置的響應卡`FlowerType`，如下圖所示。



d. 為卡片指定一個標題，並依照以下螢幕擷取畫面所示設定三個按鈕。您可以選擇性地加入圖像至回應卡，若您已有圖像 URL。如果您是使用 Twilio SMS 部署機器人，則必須提供圖像 URL。



e. 選擇 Save (儲存) 以儲存回應卡。

f. 選擇 Save intent (儲存意圖) 以儲存意圖組態。

g. 要建置機器人，選擇 Build (建置)。

h. 要發佈機器人版本，選擇 Publish (發佈)。指定 BETA 做為指向機器人版本的別名。如需版本控制的詳細資訊，請參閱「版本控制與別名」。

3. 將機器人部署在簡訊平台上：

- 將機器人部署在 Facebook Messenger 平台上並測試整合。如需相關指示，請參閱整合 Amazon Lex 機器人與臉書信使。在您訂花時，訊息視窗會顯示回應卡，讓您能夠選擇花種。
- 將機器人部署在 Slack 平台上並測試整合。如需相關指示，請參閱整合亞 Amazon Lex 機器人與 Slack 。在您訂花時，訊息視窗會顯示回應卡，讓您能夠選擇花種。
- 將機器人部署在 Twilio SMS 平台上。如需相關指示，請參閱整合 Amazon Lex 機器人與 Twilio 可程式化的簡訊 。在您訂花時，Twilio 發送的訊息會顯示回應卡上的圖像。Twilio SMS 不支援在回應中使用按鈕。

更新語音

在本練習中，您將為入門練習 1 所建立的表達用語加入額外的表達用語。您可以使用 Amazon Lex 主控台中的監控索引標籤來檢視機器人無法辨識的話語。為了改善使用者體驗，您應將這類表達用語加入至機器人。

在以下條件下，並在以下條件下，並開啟位於以下條件下

- 建立機器人時，`childDirected`欄位設定為 true。
- 您正在使用一個或多個插槽的插槽混淆。
- 您選擇不參與改善 Amazon Lex 的活動。

> **Note**
>
> 表達用語統計資料為每天產生一次。您可以查看未能判別的表達用語、該表達用語的聽取次數及上次聽取的日期和時間。缺漏的表達用語最多可能需要 24 個小時才會出現於主控台。

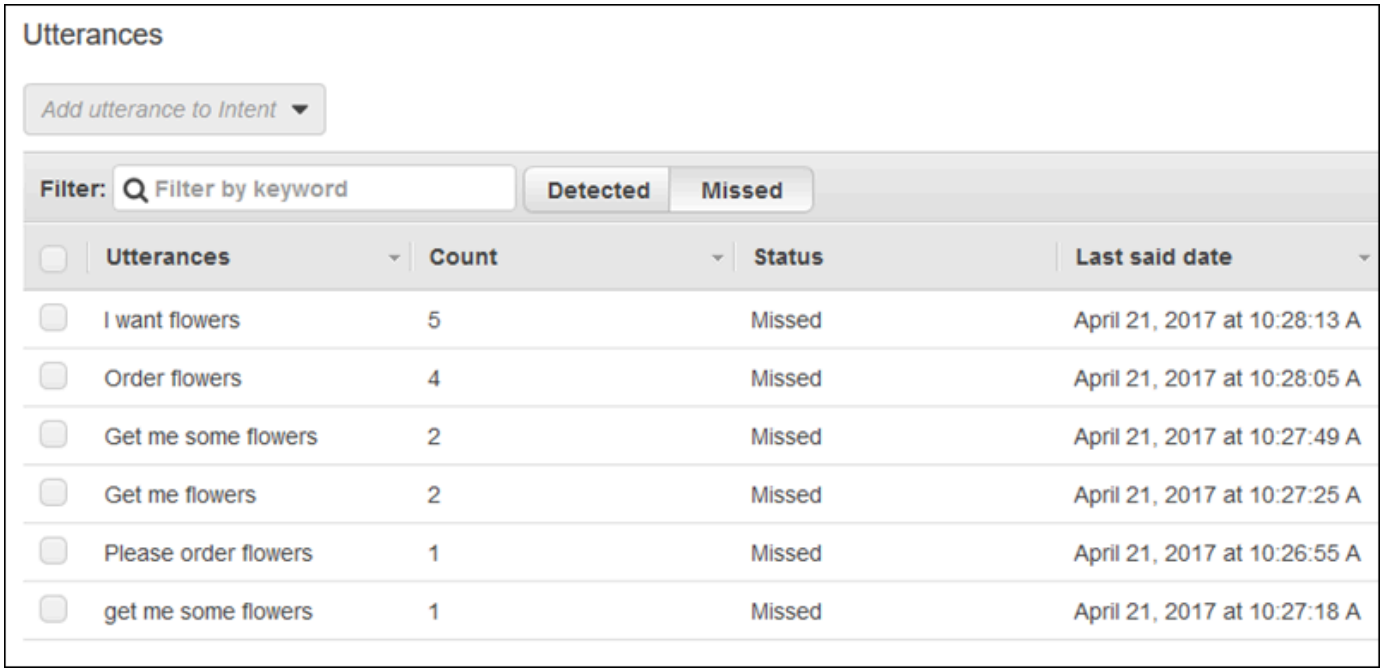
您可以查看不同版本機器人適用的表達用語。若要變更您正在查看表達用語之機器人的版本，請從機器人名稱旁的下拉式清單選擇不同的版本。

查看缺漏的表達用語並將其加入至機器人：

1. 依照入門練習 1 第一個步驟的指示，建立及測試 `OrderFlowers` 機器人。如需相關指示，請參閱練習 1：使用藍圖 (主控台) 建立 Amazon Lex 機器人。
2. 在 Test Bot (測試機器人) 視窗中輸入以下表達用語，對機器人進行測試。每個表達用語各輸入數次。範例機器人將無法判別以下表達用語：

- 訂花
- 為我介紹有什麼花
- 請訂花
- 為我介紹幾種花

3. 等待 Amazon Lex 收集有關遺漏語音的使用情況資料。表達用語資料為每天產生一次，通常在深夜產生。
4. 登入，AWS Management Console並開啟位於 https://console.aws.amazon.com/lex/ 的 Amazon Lex 主控台。
5. 選擇 `OrderFlowers` 機器人。
6. 選擇 Monitoring (監控) 索引標籤，然後從左側選單中選擇 Utterances (表達用語)，再選擇 Missed (缺漏) 按鈕。下列窗格顯示最多 100 個遺漏的話語。



7. 要選擇欲加入至機器人的任何缺漏的表達用語，請選取表達用語旁的核取方塊。要將表達用語加入至 `$LATEST` 版本的意圖，選擇 Add utterance to intent (加入表達用語至意圖) 下拉式清單旁的向下箭頭，然後選擇意圖。
8. 要重建您的機器人，選擇 Build (建置) 後再次選擇 Build (建置) 以重建您的機器人。
9. 要確認您的機器人能夠判別新的表達用語，請使用 Test Bot (測試機器人) 窗格。

與網站整合

在這個範例中，您會將機器人整合到使用文字和語音的網站，使用 JavaScript 和 AWS 服務為網站訪客建置互動式體驗。您可以自 AWS AI 部落格 選擇已記錄的範例：

- 為 Chatbot 部署 Web 用户界面— 展示功能完整的 Web UI，為 Web 用户端提供 Amazon Lex 聊天機器人。您可以藉此了解 Web 用戶端，或做為自己應用程式的建立區塊。

- 「訪客，您好！」 — 利用 Amazon Lex 吸引您的網絡用户— 在您的網站上展示使用 Amazon Lex、瀏覽器中適用於 JavaScript 的 AWS 開發套件和 Amazon Cognito 在您的網站上建立對話式體驗。

- 在瀏覽器中捕獲語音輸入並將其發送到 Amazon Lex— 在網站中使用「瀏覽器中適用於 JavaScript 的開發套件」來示範內嵌以語音為基礎的聊天機器人。應用程式會錄製音訊、傳送音訊至 Amazon Lex，然後播放回應。

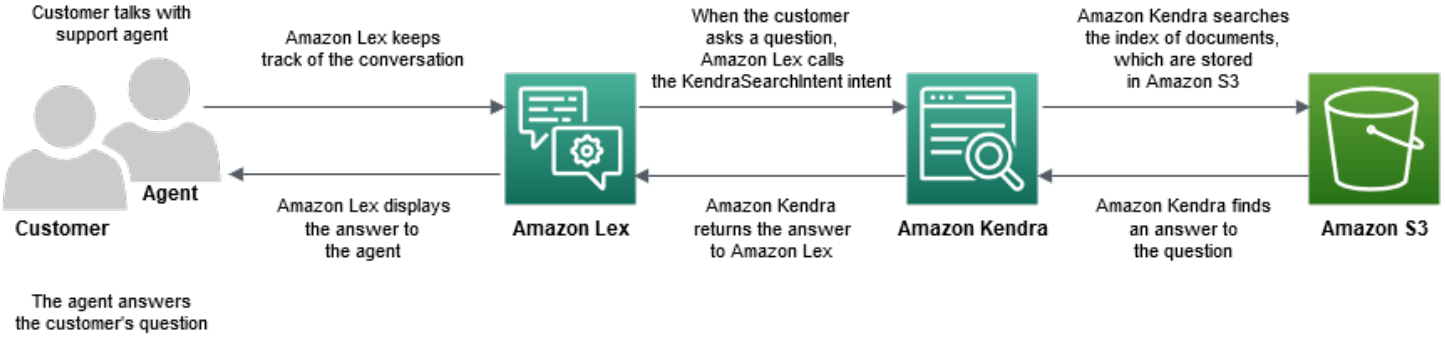
呼叫中心代理助理

在本教學中，您將 Amazon Lex 與 Amazon Kendra 搭配使用來建立代理程式協助機器人，以協助客戶支援專員並將其發佈為 Web 應用程式。Amazon Kendra 是一種企業搜尋服務，使用機器學習來搜尋文件以尋找答案。如需有關亞馬遜 Kendra 的詳細資訊，請參閱 Amazon Kendra 開發人員指南。

Amazon Lex 機器人廣泛應用於客服中心，作為客戶的第一個聯絡點。機器人通常能夠解決客戶問題。當機器人無法回答問題時，它會將對話轉移給客戶支援員工。

在本教學中，我們會建立 Amazon Lex 機器人，以供客服人員即時回答客戶查詢。通過閱讀機器人提供的答案，代理程序無法手動查找答案。

您在本教學課程中建立的機器人和 Web 應用程式可快速提供正確的資源，協助客服人員有效率且準確地回應客戶。下圖顯示 Web 應用程式如何運作。



如圖所示，Amazon Simple Storage Service (Amazon S3) 儲存貯體中。如果您還沒有 S3 儲存 Amazon Kendra。除了 Amazon S3 之外，您還將使用 Amazon Cognito 進行本教程。Amazon Cognito 會管理將機器人部署為 Web 應用程式的許可。

在本教學中，您會建立 Amazon Kendra 索引，以提供客戶問題的答案、建立機器人並新增可讓其根據與客戶的對話建議答案的意圖、設定 Amazon Cognito 以管理存取許可，以及將機器人部署為 Web 應用程式。

預估時間：75 分鐘

估計費用：Amazon Kendra 指數每小時 2.50 美元，Amazon Lex 請求 1000 美元。您的 Amazon Kendra 索引會在您完成此練習後繼續執行。請務必將其刪除以避免不必要的成本。

注意：請務必為本教學中使用的所有服務選擇相同的 AWS 區域。

主題

- 步驟 1：建立 Amazon Kendra 索引。
- 步驟 2：建立 Amazon Lex 機器人區域。
- 步驟 3：新增自訂和內建意圖集區域集區域
- 步驟 4：設定 Amazon Cognito
- 步驟 5：將機器人部署為 Web 應用程式
- 步驟 6：使用機器人集區

步驟 1：建立 Amazon Kendra 索引。

首先建立可回答客戶問題的文件的 Amazon Kendra 索引。索引提供用戶端查詢的搜尋 API。您可以從來源文件建立索引。Amazon Kendra 會將在索引文件中找到的答案傳回給機器人，並將其顯示給代理程式。

Amazon Kendra 建議的回應品質和準確性取決於您編製索引的文件。文件應包含代理程式經常存取的檔案，且必須存放在 S3 儲存貯體中。您可以使用 .html、微軟辦公室 (.doc、.ppt)、PDF 和文字格式編製非結構化和半結構化資料的索引。

若要建立 Amazon Kendra 索引，請參閱 Amazon Kendra 開發人員指南中的 S3 儲存貯體入門 (主控台)。

若要新增有助於回答客戶查詢的問題和答案 (FAQ)，請參閱 Amazon Kendra 開發人員指南中的新增問題和答案。對於本教學課程，請在上使用 ML_FAQ.csv 檔案 GitHub。

下一步驟

步驟 2：建立 Amazon Lex 機器人區域。

步驟 2：建立 Amazon Lex 機器人區域。

Amazon Lex 提供呼叫中心代理程式和 Amazon Kendra 索引之間的介面。它跟踪代理商和客戶之間的對話，並根據客戶要求的問題調用`AMAZON.KendraSearchIntent`意圖。意圖是使用者想要執行的動作。

Amazon Kendra 會搜尋已編製索引的文件，並將其顯示在機器人中的 Amazon Lex 傳回答案。只有代理才能看到此答案。

建立代理程式助理程式機器人

1. 登入，AWS Management Console並在 https://console.aws.amazon.com/lex/ 開啟 Amazon Lex 主控台。
2. 在導覽窗格中，選擇 Bots (機器人)。
3. 選擇 Create (建立)。
4. 選擇自定義機器人並配置機器人。
   a. 機器人名稱 — 輸入表示機器人用途的名稱，例如**`AgentAssistBot`**。
   b. 輸出語音 — 選擇無。
   c. 工作階段逾時 — 輸入**5**。
   d. 杯 — 選擇沒有.
5. 選擇 Create (建立)。建立機器人之後，Amazon Lex 會顯示機器人編輯器索引標籤。

下一步驟

步驟 3：新增自訂和內建意圖集區域集區域

步驟 3：新增自訂和內建意圖集區域集區域

意圖表示呼叫中心代理希望機器人執行的動作。在這種情況下，代理程式會希望機器人根據客服與客戶的對話建議回應和有用的資源。

Amazon Lex 有兩種意圖類型：自訂意圖和內建意圖。 AMAZON.KendraSearchIntent是內建意圖。機器人會使用AMAZON.KendraSearchIntent意圖查詢索引，並顯示 Amazon Kendra 建議的回應。

此範例中的機器人不需要自訂意圖。不過，若要建置機器人，您必須至少建立一個具有至少一個範例語音的自訂意圖。只有在建置您的代理程式助理機器人時，才需要此目的。它不執行任何其他功能。意圖的話語不得回答客戶可能會提出的任何問題。這可確保呼AMAZON.KendraSearchIntent叫以回答客戶查詢。如需詳細資訊，請參閱AMAZON.KendraSearchIntent。

建立所需的自訂意圖

1. 在 Getting started with your bot (開始使用您的機器人) 頁面上，選擇 Create intent (建立意圖)。
2. 針對 Add intent (新增意圖)，選擇 Create intent (建立意圖)。
3. 在「建立意圖」對話方塊中，輸入意圖的描述性名稱，例如**RequiredIntent**。
4. 在「範例語音」中，輸入描述性語音，例如**Required utterance**。
5. 選擇 Save intent (儲存意圖)。

要添加亞馬遜。 KendraSearchIntent 意圖和響應消息

1. 在導覽窗格中，選擇「對應方式」旁邊的加號 (+)。
2. 選擇 [搜尋現有的意圖]。
3. 在「搜尋方式」方塊中輸入**AMAZON.KendraSearchIntent**，然後從清單中選擇。
4. 為意圖指定描述性名稱，例如**AgentAssistSearchIntent**，然後選擇「新增」。
5. 在意圖編輯器中，選擇 Amazon Kendra query (Amazon Kendra 查詢) 以開啟查詢選項。
6. 選擇您想要搜索的索引，
7. 在 [回應] 區段中，將下列三個訊息新增至訊息群組。

```
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. 選擇 Save intent (儲存意圖)。
9. 選擇「建置」以建置機器人。

下一步驟

步驟 4：設定 Amazon Cognito

步驟 4：設定 Amazon Cognito

若要管理 Web 應用程式的許可和使用者，您需要設定 Amazon Cognito。Amazon Cognito 可確保 Web 應用程式的安全性並具有存取控制功能。Amazon Cognito 使用身分集區區區區域提供AWS登入資料集區域。AWS在本教程中，它提供了對 Amazon Lex 的訪問。

建立身分集區域中心，Amazon Cognito 為身分集區已驗證和未驗證使用者提供AWS Identity and Access Management (IAM) 角色。您可以透過新增授與 Amazon Lex 存取權的政策來修改 IAM 角色。

設定 Amazon Cognito

1. 登入，AWS Management Console並在 https://console.aws.amazon.com/cognito/ 開啟 Amazon Cognito 主控台。
2. 選擇 Manage Identity Pools (管理身分集區)。
3. 選擇 Create new identity pool (建立新的身分集區)。
4. 設定身分集區區區區區域

   a. 識別集區名稱 — 輸入指出集區用途的名稱，例如**`BotPool`**。

   b. 在「未驗證身分」區段中，選擇「啟用對未驗證身分的存取」。
5. 選擇 Create Pool (建立集區)。
6. 在 [識別要搭配新身分集區使用的 IAM 角色] 頁面上，選擇 [檢視詳細資料]。
7. 記錄 IAM 角色名稱。稍後您將修改它們。

8. 選擇 Allow (允許)。
9. 在 Amazon Cognito 入門頁面上，對於平台，選擇JavaScript。
10. 在「取得AWS認證」區段中，尋找並記錄身分集區 ID。
11. 若要允許存取 Amazon Lex，請修改已驗證和未驗證的 IAM 角色。

    a. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/iam/ 的 IAM 主控台。

    b. 在功能窗格的 [存取管理] 下，選擇 [角色]。

    c. 在搜尋方塊中，輸入已驗證 IAM 角色的名稱，再選擇旁邊的核取方塊。

        i. 選擇 Attach policies (連接政策)。

        ii. 在搜尋方塊中輸入**AmazonLexRunBotsOnly**並選擇旁邊的核取方塊。

        iii. 選擇 Attach policy (連接政策)。

    d. 在搜尋方塊中輸入未驗證 IAM 角色的名稱，再選擇旁邊的核取方塊。

        i. 選擇 Attach policies (連接政策)。

        ii. 在搜尋方塊中輸入**AmazonLexRunBotsOnly**並選擇旁邊的核取方塊。

        iii. 選擇 Attach policy (連接政策)。

下一步驟

步驟 5：將機器人部署為 Web 應用程式

步驟 5：將機器人部署為 Web 應用程式

將您的機器人部署為 Web 應用程式

1. 將資料庫下載到您的電腦：https://github.com/awsdocs/amazon-lex-developer-guide /blob/ 主程式/應用程式/代理程式支援機器人。
2. 導覽至下載的儲存庫，然後在編輯器中開啟 index.html 檔案。
3. 進行下列變更：

    a. 在區`AWS.config.credentials`段中，輸入您的區域名稱和您的身分集區 ID。

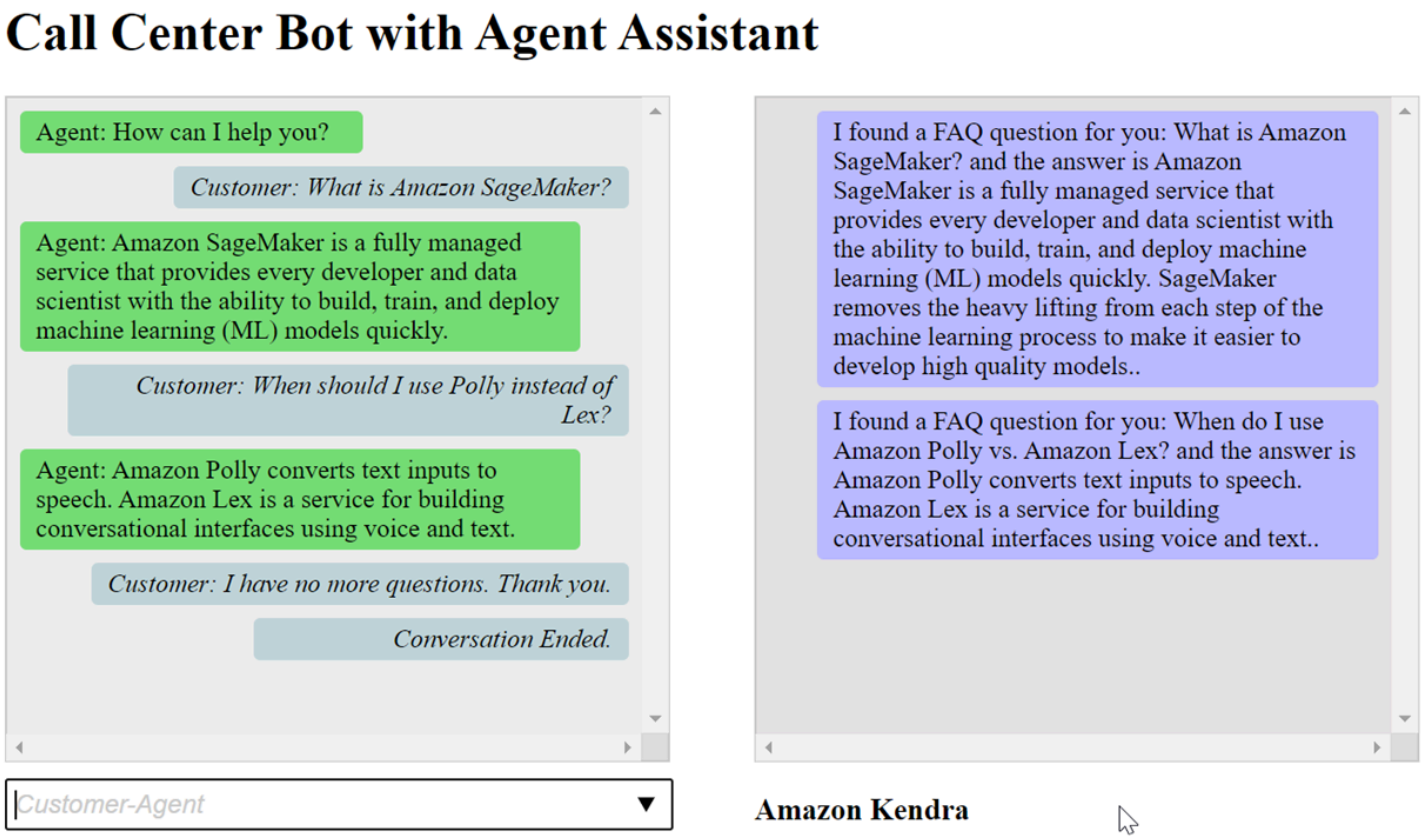
    b. 在 `Amazon Lex runtime parameters`區段中，輸入機器人名稱。

    c. 儲存檔案。

步驟 6：使用機器人集區

為了展示目的，您以客戶和代理的身份向 Bot 提供輸入。為了區分這兩者，客戶提出的問題以「客戶：」開頭，代理提供的答案以「Agent:」開頭。您可以從建議的輸入選單中進行選擇。

透過開啟`index.html`來執行 Web 應用程式，以與您的機器人進行類似下列影像的對話：



在 index.html 文件中的`pushChat()`功能解釋如下。

```
            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;
```

當您以客戶身分提供輸入時，Amazon Lex 執行階段 API 會將其傳送到 Amazon Lex。

該showMessage(daText, senderRequest, displayWindow)功能在聊天窗口中顯示代理和客戶之間的對話。Amazon Kendra 建議的回應會顯示在相鄰的視窗中。客戶說話時，對話結束**“I have no more questions. Thank you.”**

注意：請在不使用 Amazon Kendra 索引時刪除。

移轉機器人

Amazon Lex V2 API 使用更新的資訊架構，可簡化資源版本控制，並支援機器人中的多種語言。如需詳細資訊，請參閱 Amazon Lex V2 開發人員指南中的遷移指南。

要使用這些新功能，您需要遷移您的機器人。當您移轉機器人時，Amazon Lex 會提供下列項目：

- 遷移會將您的自訂意圖和插槽類型複製到 Amazon Lex V2 機器人。
- 您可以在相同的 Amazon Lex V2 機器人中新增多種語言。在 Amazon Lex V1 中，您可以為每種語言建立個別的機器人。您可以將多個 Amazon Lex V1 機器人 (每個機器人使用不同的語言) 遷移到一個 Amazon Lex V2 機器人。
- Amazon Lex 將 Amazon Lex V1 內建插槽類型和意圖對應至 Amazon Lex V2 內建插槽類型和對應方式。如果無法遷移內建項目，Amazon Lex 會傳回一則訊息，告訴您下一步該怎麼做。

遷移程序不會移轉下列項目：

- 別名
- Amazon Kendra 指數
- AWS Lambda 函式
- 交談日誌設定
- 消息傳遞渠道，例如 Slack
- Tags (標籤)

若要移轉機器人，您的使用者或角色必須同時擁有 Amazon Lex 和 Amazon Lex V2 API 操作的 IAM 許可。如需所需許可，請參閱允許使用者將機器人移轉至 Amazon Lex V2 API。

遷移機器人（控制台）

使用 Amazon Lex V1 主控台將機器人的結構遷移到 Amazon Lex V2 機器人。

若要使用主控台將機器人移轉至 Amazon Lex V2 API

1. 登入，AWS Management Console並在 https://console.aws.amazon.com/lex/ 開啟 Amazon Lex 主控台。
2. 在左側選單中，選擇 Sequi retion 工具。

3. 從機器人清單中，選擇您要移轉的機器人，然後選擇 [移轉]。
4. 選擇您要移轉的機器人版本，然後輸入要移轉到的機器人名稱。如果您輸入現有 Amazon Lex V2 機器人的名稱，Amazon Lex V1 機器人會遷移到詳細資料中顯示的語言，並覆寫該語言的草稿版本。
5. 選擇 Next (下一步)。
6. 選擇 Amazon Lex 用來執行機器人的 Amazon Lex V2 API 版本的 IAM 角色。您可以選擇使用執行機器人所需的最低權限建立新角色，也可以選擇現有的 IAM 角色。
7. 選擇 Next (下一步)。
8. 檢閱遷移設定。如果看起來沒問題，請選擇 [開始移轉]。

啟動移轉程序後，您會返回移轉工具開始頁面。您可以在「歷史記錄」表格中監控遷移進度。當 [移轉狀態] 欄顯示 [完成移轉已完成] 時。

亞馬遜萊克斯使用 Amazon Lex V2 API 中的`StartImport`操作來匯入遷移的機器人。您會在每次遷移的 Amazon Lex V2 主控台匯入歷史記錄表格中看到一個項目。

在移轉期間，Amazon Lex 可能會在機器人中找到無法移轉的資源。您會收到無法移轉的每個資源的錯誤或警告訊息。每則訊息都包含說明如何解決問題的說明文件連結。

遷移 Lambda 函式

Amazon Lex V2 會變更為機器人定義 Lambda 函數的方式。它只允許一個機器人中的每種語言在別名中使用一個 Lambda 函數。如需遷移 Lambda 函式的詳細資訊，請參閱將 Lambda 函數從 Amazon Lex V1 遷移到 Amazon Lex V2。

遷移訊息

在遷移期間，Amazon Lex 可能會找到無法遷移到相等 Amazon Lex V2 資源的資源 (例如內建插槽類型)。發生這種情況時，Amazon Lex 會傳回描述發生情況的遷移訊息，並提供說明如何修正遷移問題的文件連結。以下各節說明移轉機器人時可能出現的問題，以及如何解決此問題。

主題

- 內建意圖
- 內建槽型
- 交談日誌
- 訊息群組

- 提示和片語
- 其他 Amazon Lex v1 功能

內建意圖

當您使用 Amazon Lex V2 不支援的內建意圖時，意圖會對應至 Amazon Lex V2 機器人中的自訂意圖。自訂意圖不包含語音。若要繼續使用意圖，請新增範例語音。

內建槽型

任何使用 Amazon Lex V2 不支援的插槽類型的遷移插槽，都不會獲得插槽類型值。若要使用此插槽：

- 建立自訂插槽類型
- 加入槽類型預期的插槽類型值
- 更新插槽以使用新的自訂插槽類型

交談日誌

遷移不會更新 Amazon Lex V2 機器人的交談日誌設定。

設定交談記錄

1. 在 https://console.aws.amazon.com/lexv2 打開 Amazon Lex V2 控制台。
2. 從機器人清單中，選擇您要設定其對話記錄的機器人。
3. 從左側選單中，選擇 Pales (別名)，然後從清單中選擇 Pales (別名)。
4. 在 [交談記錄] 區段中，選擇 [管理交談記錄] 以設定機器人別名的交談記錄。

訊息群組

Amazon Lex V2 僅支援每個訊息群組一則訊息和兩個替代訊息。如果 Amazon Lex V1 機器人中的每個訊息群組有三個以上的訊息，則只會遷移前三個訊息。若要在訊息群組中使用更多訊息，請使用 Lambda 函數輸出各種訊息。

提示和片語

Amazon Lex V2 使用不同的機制來進行跟進、澄清和掛斷提示。

對於後續提示，請在履行後使用上下文轉存切換到不同的意圖。

例如，假設您有意圖預訂汽車租賃，該租車會設定為傳回名為的輸出內容book_car_fulfilled。當意圖達成時，Amazon Lex 會將輸出內容變數設定為book_car_fulfilled。由於book_car_fulfilled是一個活躍的上下文，因此只要用戶的話語被識別為嘗試引起該意圖的嘗試，就會考慮將其作為輸入上下文的意圖進行識別。book_car_fulfilled您可以將其用於僅在預訂汽車後才有意義的意圖，例如通過電子郵件發送收據或修改預訂。

Amazon Lex V2 不支援澄清提示和掛斷詞組 (中止陳述式)。Amazon Lex V2 機器人包含預設的後援意圖，如果沒有符合任何意圖，就會叫用該意圖。若要傳送含重試的澄清提示，請設定 Lambda 函數，並在後援意圖中啟用對話方塊程式碼掛接。Lambda 函數可以輸出澄清提示作為回應，並在工作階段屬性中重試值。如果重試值超過重試次數上限，您可以輸出掛斷片語並關閉交談。

其他 Amazon Lex v1 功能

遷移工具僅支援遷移 Amazon Lex V1 機器人及其基礎意圖、插槽類型和插槽。如需其他功能，請參閱 Amazon Lex V2 文件中的下列主題。

- 機器人別名：別名
- 機器人通道：在簡訊平台上部署 Amazon Lex V2 機器人
- 交談記錄設定：使用交談記錄進行監控
- Amazon Kendra 索引：亞馬遜。 KendraSearchIntent
- Lambda 函數：使用AWS Lambda函數
- 標籤：標記資源

將 Lambda 函數從 Amazon Lex V1 遷移到 Amazon Lex V2

Amazon Lex V2 只允許一個機器人中的每種語言使用一個 Lambda 函數。Lambda 函數及其設定是針對您在執行階段使用的機器人別名進行設定。

如果針對意圖啟用對話方塊和履行程式碼掛接，則會針對該語言的所有意圖叫用 Lambda 函數。

Amazon Lex V2 Lambda 函數具有與 Amazon Lex V1 不同的輸入和輸出消息格式。這些都是 Lambda 函數輸入格式的差異。

- Amazon Lex V2 替換currentIntent與結alternativeIntents構與結interpretations構。每個解譯都包含意圖、意圖的 NLU 信賴度分數，以及選用的情緒分析。

- Amazon Lex V2 移動activeContexts，sessionAttributes在 Amazon Lex V1 到統一的sessionState結構。此結構會提供有關交談的目前狀態的相關資訊，包括原始請求 ID。
- Amazon Lex V2 不返回recentIntentSummaryView. 請改用sessionState結構中的資訊。
- Amazon Lex V2 輸入提供bot屬性localeId中的botId和。
- 輸入結構包含提供輸入類型資訊的inputMode屬性：文字、語音或 DTMF。

以下是 Lambda 函數輸出格式的差異：

- Amazon Lex V1 中的activeContexts和結sessionAttributes構被亞馬 Amazon Lex V2 中的sessionState結構所取代。
- 輸出中recentIntentSummaryView未包含在輸出中。
- Amazon Lex V1dialogAction 結構分為兩個結構，dialogAction這是sessionState結構的一部分，messages這是必要的dialogAction.typeElicitIntent。Amazon Lex 從這個結構中選擇訊息來顯示給使用者。

使用 Amazon Lex V2 API 建立機器人時，每種語言的每個機器人別名只有一個 Lambda 函數，而不是針對每個意圖使用 Lambda 函數。如果你想繼續使用單獨的功能，你可以創建一個路由器功能，為每個意圖激活一個單獨的功能。以下是您可以在應用程式中使用或修改的路由器功能。

```
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
```

更新的欄位清單

下表提供 Amazon Lex V2 Lambda 請求和回應中更新欄位的詳細資訊。您可以使用這些表格來對應版本之間的欄位。

要求

下列欄位已更新為 Lambda 函數請求格式。

使用中內容

結`activeContexts`構現在是結`sessionState`構的一部分。

v1 結構	v1 結構
主動式環境	工作階段狀態. 主動上下文
主動式上下文 [*]。 timeToLive	工作階段狀態。主動式上下文 [*]。 timeToLive
主動式上下文 [*]。 timeToLive。 timeToLiveInSeconds	工作階段狀態。主動式上下文 [*]。 timeToLive。 timeToLiveInSeconds
主動式上下文 [*]。 timeToLive。 turnsToLive	工作階段狀態。主動式上下文 [*]。 timeToLive。 turnsToLive
主動式上下文 [*]. 名稱	工作階段狀態. 主動式上下文 [*]. 名稱
主動式上下文 [*]. 參數	工作階段狀態. 主動式上下文 [*]. 上下文屬性

替代意圖

索引 1 到 N 的解釋清單包含 Amazon Lex V2 預測的替代意圖清單，以及其可信度分數。`recentIntentSummaryView`已從 Amazon Lex V2 中的請求結構中移除。若要查看中的詳細資訊`recentIntentSummaryView`，請使用GetSession作業。

v1 結構	v1 結構
替代意圖	解釋 [1：*]
recentIntentSummary檢視	N/A

機器人

在 Amazon Lex V2 中，機器人和別名具有識別碼。機器人 ID 是程式碼掛接輸入的一部分。別名 ID 會包含在內，但不包含別名名稱。Amazon Lex V2 支援相同機器人的多個語言環境，因此包含地區設定 ID。

v1 結構	v1 結構
機器人	機器人
機器人名	機器人名
N/A	機器人識別碼
機器人別名	N/A
N/A	不良病毒
機器人版	機器人版
N/A	植物学

目前的意圖

結`sessionState.intent`構包含使用中意圖的詳細資訊。Amazon Lex V2 也會傳回`interpretations`結構中所有意圖的清單，包括替代意圖。解釋列表中的第一個元素始終與相同`sessionState.intent`。

v1 結構	v1 結構
currentIntent	會話狀態。意圖或解釋 [0]. 意圖

v1 結構	v1 結構
目前意圖名稱	工作階段狀態。意圖名稱或解釋 [0]。
currentIntent。 nluConfidenceScore	解釋 [0]. 無信心. 得分

對話框動作

`confirmationStatus`欄位現在是`sessionState`結構的一部分。

v1 結構	v1 結構
目前意圖. 確認狀態	會話狀態. 意圖. 確認狀態或解釋 [0]. 意圖.
N/A	會話狀態。意圖。狀態或解釋 [*] .intent.state

Amazon Kendra

`kendraResponse`欄位現在是`sessionState`和`interpretations`結構的一部分。

v1 結構	v1 結構
kendraResponse	會話狀態. 意圖. 肯德拉響應或解釋 [0].

Inmo

結`sentimentResponse`構即會移至新`interpretations`結構。

v1 結構	v1 結構
sentimentResponse 持	詮釋 [0]. Intion
情緒支持. 情緒標籤	解釋 [0]. 情緒自發. 情緒
情緒支持。感情分數	解釋 [0]. 情緒自發. 情緒分數

槽

Amazon Lex V2 在`sessionState.intent`結構內提供單一`slots`物件，其中包含已解析的值、解譯值和使用者所說的原始值。Amazon Lex V2 也透過設定`slotShape`為`List`和設定`values`清單來支援多值插槽。單值槽由`value`字段支持，它們的形狀被假定為`Scalar`。

v1 結構	v1 結構
目前意向插槽	會話狀態. 意圖插槽或解釋 [0].
目前意圖。插槽 [*]. 值	值. 解釋值或解釋 [0]. 意圖插槽 [*]. 值.
N/A	插槽 [*]. 值. 形狀或解釋 [0]. 意圖插槽 [*].
N/A	插槽 [*]. 值或解釋 [0]. 意圖插槽 [*].
目前意圖。插槽詳細資訊	會話狀態. 意圖插槽或解釋 [0].
目前意圖。插槽詳細資訊 [*].	插槽 [*]. 解析值或解釋 [0]. 意圖插槽 [*]. 已解析值
目前意圖。插槽詳細資訊 [*]. 原始值	插槽 [*]. 原始值或解釋 [0]. 意圖插槽 [*]. 原始值

其他

Amazon Lex V2`sessionId` 欄位與亞 Amazon Lex V1 中的`userId`欄位相同。Amazon Lex V2 還發送呼叫者`inputMode`的：文本，DTMF 或語音。

v1 結構	v1 結構
userId	sessionId
inputTranscript	inputTranscript
invocationSource	invocationSource
outputDialogMode	responseContentType
messageVersion	messageVersion

v1 結構	v1 結構
sessionAttributes	階段作業狀態. 工作階段屬性
requestAttributes	requestAttributes
N/A	輸入模式
N/A	originatingRequestId

回應

Lambda 函數回應訊息格式中的下列欄位已變更。

使用中內容

結`activeContexts`構已移至結`sessionState`構。

v1 結構	v1 結構
主動式環境	工作階段狀態. 主動上下文
主動式上下文 [*]。 timeToLive	工作階段狀態。主動式上下文 [*]。 timeToLive
主動式上下文 [*]。 timeToLive。 timeToLiveInSeconds	工作階段狀態。主動式上下文 [*]。 timeToLive。 timeToLiveInSeconds
主動式上下文 [*]。 timeToLive。 turnsToLive	工作階段狀態。主動式上下文 [*]。 timeToLive。 turnsToLive
主動式上下文 [*]. 名稱	工作階段狀態. 主動式上下文 [*]. 名稱
主動式上下文 [*]. 參數	工作階段狀態. 主動式上下文 [*]. 上下文屬性

對話框動作

結`dialogAction`構已移至結`sessionState`構。您現在可以在對話方塊動作中指定多個訊息，而`genericAttachments`結構現在就是結`imageResponseCard`構。

v1 結構	v1 結構
dialogAction	工作階段狀態. 對話
對話動作. 類型	工作階段狀態. 對話. 類型
dialogAction。 slotToElicit	會話狀態。意圖。對話。 slotToElicit
對話. 類型. 履行狀態	工作階段狀態. 意圖. 狀態
對話動作. 訊息	messages
對話動作. 訊息. 內容類型	訊息 [*]. 內容類型
對話動作. 訊息. 內容	消息 [*]. 內容
對話. 回應卡	消息 [*]。 imageResponseCard
對話. 回應卡. 版本	N/A
對話. 回應卡. 內容類型	訊息 [*]. 內容類型
對話框. 回應卡. 一般附件	N/A
對話框. 回應卡. 一般附件 [*]. 標題	消息 [*]。 imageResponseCard. 標題
對話. 回應卡. 一般附件 [*].	消息 [*]。 imageResponseCard. 字幕。
對話框. 回應卡. 一般附件 [* imageUrl	消息 [*]。 imageResponseCard. imageUrl
對話框. 回應卡. 一般附件 [*]. 按鈕	消息 [*]。 imageResponseCard. 按鈕。
一般附件 [*]. 按鈕 [*].	消息 [*]。 imageResponseCard. 按鈕 [*]. 值
通用附件 [*]. 按鈕 [*].	消息 [*]。 imageResponseCard. 按鈕 [*]. 文字
dialogAction。 kendraQueryRequest酬載	dialogAction。 kendraQueryRequest酬載
dialogAction。 kendraQueryFilter字符串	dialogAction。 kendraQueryFilter字符串

意圖和槽

屬於dialogAction結構一部分的意圖和槽欄位現在已成為sessionState結構的一部分。

v1 結構	v1 結構
對話動作. 意圖名稱	工作階段狀態. 意圖名稱
對話. 插槽	工作階段狀態. 意圖槽
對話。插槽 [*]. 鍵	工作階段狀態。意圖插槽 [*] .key
對話框. 插槽 [*]. 值	工作階段狀態。意圖插槽 [*]. 值.
N/A	工作階段狀態。意圖插槽 [*].
N/A	工作階段狀態. 意圖插槽 [*]. 值

其他

結sessionAttributes構現在是結sessionState構的一部分。
結recentIntentSummaryReview構已經移除。

v1 結構	v1 結構
sessionAttributes	階段作業狀態. 工作階段屬性
recentIntentSummary檢視	N/A

Amazon Lex 中的安全

雲安全 AWS 是最高的優先級。身為 AWS 客戶，您可以從資料中心和網路架構中獲益，該架構專為滿足對安全性最敏感的組織的需求而打造。

安全是 AWS 與您之間共同的責任。共同的責任模型 將此描述為雲端 本身 的安全和雲端內部的安全：

- 雲端的安全性 — AWS 負責保護在 AWS 雲端中執行 AWS 服務的基礎架構。 AWS 還為您提供可以安全使用的服務。第三方稽核人員定期檢測及驗證安全的效率也是我們 AWS 合規計劃的一部分。若要了解適用於 Amazon Lex 的合規計劃，請參閱合規計劃適用範圍的AWS 服務。
- 雲端中的安全性 — 您的責任取決於您使用的 AWS 服務。您也必須對資料敏感度、組織要求，以及適用法律和法規等其他因素負責。

本文件將協助您了解如何在使用 Amazon Lex 時套用共同的責任模型。下列主題說明如何設定 Amazon Lex 以符合安全和合規目標。您也將學習如何使用可協助您監控和保護 Amazon Lex 資源的其他 AWS 服務。

主題

- Amazon Lex 的數據保護
- Amazon Lex 的 Identity and Access Management
- 在 Amazon Lex 監控
- Amazon Lex 的合規驗證
- Amazon Lex 的彈性
- Amazon Lex 的基礎設施

Amazon Lex 的數據保護

Amazon Lex 會收集客戶內容以進行疑難排解，並協助改善服務。根據預設，客戶內容會受到保護。您可以使用 Amazon Lex API 刪除個別客戶的內容。

Amazon Lex 存儲四種類型的內容：

- 範例表達用語，用於建置和訓練機器人
- 使用者與機器人互動時的客戶表達用語

- 在使用者與機器人互動期間，提供應用程式特定資訊的會話屬性。
- 請求屬性，該屬性包含將單一請求套用到機器人的資訊

任何專為兒童使用而設計的 Amazon Lex 機器人均受《兒童線上隱私保護法》(COPPA) 管轄。您可以使用主控台或 Amazon Lex API 將childDirected欄位設定為，告訴亞馬遜萊克斯機器人受到 COPPA 的約束。true當 childDirected 欄位設為 true 時，將不會儲存任何使用者表達用語。

主題

靜態加密

Amazon Lex 會加密其存放的使用者話語。

主題

範例表達用語

在您開發機器人時，您可以為每個意圖和槽提供範例表達用語。您也可以為槽提供自訂值和同義詞。這些資訊會在靜態時加密，用來建立機器人並建立使用者體驗。

客戶表達用語

除非childDirected欄位設定為，否則 Amazon Lex 會加密使用者傳送至機器人的話語。true

當 childDirected 欄位設為 true 時，將不會儲存任何使用者表達用語。

當 childDirected 欄位設為 false (預設) 時，使用者表達用語將會加密並存放 15 天，以和 GetUtterancesView 操作搭配使用。若要刪除為特定使用者存放的表達用語用於，請使用 DeleteUtterances 操作。

當您的機器人接受語音輸入時，輸入將無限期存儲。Amazon Lex 使用它來改善機器人回應使用者輸入的能力。

使用 DeleteUtterances 操作以刪除特定使用者存放的表達用語。

工作階段屬性

工作階段屬性包含 Amazon Lex 和用戶端應用程式之間傳遞的應用程式特定資訊。Amazon Lex 會將工作階段屬性傳遞給針對機器人設定的所有 AWS Lambda 功能。如果 Lambda 函數新增或更新工作階段屬性，Amazon Lex 會將新資訊傳回給用戶端應用程式。

會話屬性會在工作階段期間保持加密存放。在最後一個使用者表達用語後，您可以將工作階段設定為保持作用至少 1 分鐘，與最多達 24 小時。預設工作階段持續時間為 5 分鐘。

請求屬性

請求屬性包含請求特定的資訊並且僅適用於目前的請求。用戶端應用程式會使用請求屬性，在執行時間將資訊傳送至 Amazon Lex。

您可使用請求屬性來傳遞不需要在整個工作階段內保留的資訊。由於請求屬性不會跨請求保留，因此不會儲存。

傳輸中加密

Amazon Lex 使用 HTTPS 通訊協定與您的用戶端應用程式進行通訊。它使用 HTTPS 和 AWS 簽名與其他服務 (例如 Amazon Polly) 通訊，並代表您的應用程式進 AWS Lambda 行通訊。

金鑰管理

Amazon Lex 透過內部金鑰保護您的內容免於未經授權的使用。

Amazon Lex 的 Identity and Access Management

AWS Identity and Access Management (IAM) 可協助管理員安全地控制 AWS 資源存取權。 AWS 服務 IAM 管理員控制哪些人可以通過身份驗證 (登入) 和授權 (具有許可) 來使用 Amazon Lex 資源。您可以使用 IAM AWS 服務 ，無需額外付費。

主題

- 物件
- 使用身分驗證

- 使用政策管理存取權
- 亞馬遜萊克斯如何與 IAM 合作
- Amazon Lex 的基於身份的政策示例
- AWS亞馬遜萊克斯的受管政策
- 使用 Amazon Lex 的服務連結角色
- 疑難排解 Amazon Lex 身分和存取

物件

您使用 AWS Identity and Access Management (IAM) 的方式會有所不同，這取決於您在 Amazon Lex 中所做的工作。

服務使用者 — 如果您使用 Amazon Lex 服務執行工作，則管理員會為您提供所需的登入資料和許可。當您使用更多 Amazon Lex 功能完成工作時，可能需要額外的許可。了解存取許可的管理方式可協助您向管理員請求正確的許可。如果您無法在 Amazon Lex 中存取某項功能，請參閱疑難排解 Amazon Lex 身分和存取。

服務管理員 — 如果您負責公司的 Amazon Lex 資源，您可能擁有 Amazon Lex 的完整存取權。判斷服務使用者應存取哪些 Amazon Lex 功能和資源是您的工作。接著，您必須將請求提交給您的 IAM 管理員，來變更您服務使用者的許可。檢閱此頁面上的資訊，了解 IAM 的基本概念。若要進一步了解貴公司如何將 IAM 與 Amazon Lex 搭配使用，請參閱亞馬遜萊克斯如何與 IAM 合作。

IAM 管理員 — 如果您是 IAM 管理員，您可能想要了解如何撰寫政策以管理 Amazon Lex 存取權的詳細資訊。若要檢視可在 IAM 中使用的 Amazon Lex 身分型政策範例，請參閱。Amazon Lex 的基於身份的政策示例

使用身分驗證

驗證是您 AWS 使用身分認證登入的方式。您必須以 IAM 使用者身分或假設 IAM 角色進行驗證 (登入 AWS)。 AWS 帳戶根使用者

您可以使用透過 AWS 身分識別來源提供的認證，以聯合身分識別身分登入。 AWS IAM Identity Center (IAM 身分中心) 使用者、貴公司的單一登入身分驗證，以及您的 Google 或 Facebook 登入資料都是聯合身分識別的範例。您以聯合身分登入時，您的管理員先前已設定使用 IAM 角色的聯合身分。當您使 AWS 用同盟存取時，您會間接擔任角色。

根據您的使用者類型，您可以登入 AWS Management Console 或 AWS 存取入口網站。如需有關登入的詳細資訊 AWS，請參閱《AWS 登入 使用指南》 AWS 帳戶中的如何登入您的。

如果您 AWS 以程式設計方式存取，請 AWS 提供軟體開發套件 (SDK) 和命令列介面 (CLI)，以便使用您的認證以加密方式簽署要求。如果您不使用 AWS 工具，則必須自行簽署要求。如需使用建議的方法自行簽署請求的詳細資訊，請參閱 IAM 使用者指南中的簽署 AWS API 請求。

無論您使用何種身分驗證方法，您可能都需要提供額外的安全性資訊。例如， AWS 建議您使用多重要素驗證 (MFA) 來增加帳戶的安全性。如需更多資訊，請參閱 AWS IAM Identity Center 使用者指南中的多重要素驗證和 IAM 使用者指南中的在 AWS中使用多重要素驗證 (MFA)。

AWS 帳戶 根使用者

當您建立時 AWS 帳戶，您會從一個登入身分開始，該身分可完整存取該帳戶中的所有資源 AWS 服務和資源。此身分稱為 AWS 帳戶 root 使用者，可透過使用您用來建立帳戶的電子郵件地址和密碼登入來存取。強烈建議您不要以根使用者處理日常任務。保護您的根使用者憑證，並將其用來執行只能由根使用者執行的任務。如需這些任務的完整清單，了解需以根使用者登入的任務，請參閱 IAM 使用者指南中的需要根使用者憑證的任務。

聯合身分

最佳作法是要求人類使用者 (包括需要系統管理員存取權的使用者) 使用與身分識別提供者的同盟，才能使用臨時登入資料進行存取 AWS 服務 。

聯合身分識別是來自企業使用者目錄的使用者、Web 身分識別提供者、Identi ty Center 目錄，或使用透過身分識別來源提供的認證進行存取 AWS 服務 的任何使用者。 AWS Directory Service同盟身分存取時 AWS 帳戶，他們會假設角色，而角色則提供臨時認證。

對於集中式存取權管理，我們建議您使用 AWS IAM Identity Center。您可以在 IAM Identity Center 中建立使用者和群組，也可以連線並同步到自己身分識別來源中的一組使用者和群組，以便在所有應用程式 AWS 帳戶 和應用程式中使用。如需 IAM Identity Center 的詳細資訊，請參閱 AWS IAM Identity Center 使用者指南中的什麼是 IAM Identity Center？。

IAM 使用者和群組

IAM 使用者是您內部的身分，具 AWS 帳戶 有單一人員或應用程式的特定許可。建議您盡可能依賴暫時憑證，而不是擁有建立長期憑證 (例如密碼和存取金鑰) 的 IAM 使用者。但是如果特定使用案例需要擁有長期憑證的 IAM 使用者，建議您輪換存取金鑰。如需更多資訊，請參閱 IAM 使用者指南中的為需要長期憑證的使用案例定期輪換存取金鑰。

IAM 群組是一種指定 IAM 使用者集合的身分。您無法以群組身分簽署。您可以使用群組來一次為多名使用者指定許可。群組可讓管理大量使用者許可的程序變得更為容易。例如，您可以擁有一個名為 IAMAdmins 的群組，並給予該群組管理 IAM 資源的許可。

使用者與角色不同。使用者只會與單一人員或應用程式建立關聯，但角色的目的是在由任何需要它的人員取得。使用者擁有永久的長期憑證，但角色僅提供暫時憑證。如需進一步了解，請參閱IAM 使用者指南中的建立 IAM 使用者 (而非角色) 的時機。

IAM 角色

IAM 角色是您 AWS 帳戶 內部具有特定許可的身分。它類似 IAM 使用者，但不與特定的人員相關聯。您可以切換角色，在中暫時擔任 IAM 角色。 AWS Management Console 您可以透過呼叫 AWS CLI 或 AWS API 作業或使用自訂 URL 來擔任角色。如需使用角色的方法詳細資訊，請參閱 IAM 使用者指南中的使用 IAM 角色。

使用暫時憑證的 IAM 角色在下列情況中非常有用：

- 聯合身分使用者存取 — 如需向聯合身分指派許可，請建立角色，並為角色定義許可。當聯合身分進行身分驗證時，該身分會與角色建立關聯，並獲授予由角色定義的許可。如需有關聯合角色的相關資訊，請參閱 IAM 使用者指南中的為第三方身分提供者建立角色。如果您使用 IAM Identity Center，則需要設定許可集。為控制身分驗證後可以存取的內容，IAM Identity Center 將許可集與 IAM 中的角色相關聯。如需有關許可集的資訊，請參閱 AWS IAM Identity Center 使用者指南中的許可集。
- 暫時 IAM 使用者許可 – IAM 使用者或角色可以擔任 IAM 角色來暫時針對特定任務採用不同的許可。
- 跨帳戶存取權：您可以使用 IAM 角色，允許不同帳戶中的某人 (信任的主體) 存取您帳戶的資源。角色是授予跨帳戶存取權的主要方式。但是，對於某些策略 AWS 服務，您可以將策略直接附加到資源（而不是使用角色作為代理）。若要了解跨帳戶存取角色和以資源為基礎的政策之間的差異，請參閱 IAM 使用者指南中的 IAM 中的跨帳戶資源存取。
- 跨服務訪問 — 有些 AWS 服務 使用其他 AWS 服務功能。例如，當您在服務中進行呼叫時，該服務通常會在 Amazon EC2 中執行應用程式或將物件儲存在 Amazon Simple Storage Service (Amazon S3) 中。服務可能會使用呼叫主體的許可、使用服務角色或使用服務連結角色來執行此作業。
  - 轉寄存取工作階段 (FAS) — 當您使用 IAM 使用者或角色在中執行動作時 AWS，您會被視為主體。使用某些服務時，您可能會執行某個動作，進而在不同服務中啟動另一個動作。FAS 會使用主體呼叫的權限 AWS 服務，並結合要求 AWS 服務 向下游服務發出要求。只有當服務收到需要與其 AWS 服務 他資源互動才能完成的請求時，才會發出 FAS 請求。在此情況下，您必須具有執行這兩個動作的許可。如需提出 FAS 請求時的政策詳細資訊，請參閱《轉發存取工作階段》。
  - 服務角色 – 服務角色是服務擔任的 IAM 角色，可代表您執行動作。IAM 管理員可以從 IAM 內建立、修改和刪除服務角色。如需詳細資訊，請參閱 IAM 使用者指南中的建立角色以委派許可給 AWS 服務服務。
  - 服務連結角色 — 服務連結角色是連結至. AWS 服務服務可以擔任代表您執行動作的角色。服務連結角色會顯示在您的中， AWS 帳戶 且屬於服務所有。IAM 管理員可以檢視，但不能編輯服務連結角色的許可。

- 在 Amazon EC2 上執行的應用程式 — 您可以使用 IAM 角色來管理在 EC2 執行個體上執行的應用程式以及發出 AWS CLI 或 AWS API 請求的臨時登入資料。這是在 EC2 執行個體內儲存存取金鑰的較好方式。若要將 AWS 角色指派給 EC2 執行個體並提供給其所有應用程式，請建立連接至執行個體的執行個體設定檔。執行個體設定檔包含該角色，並且可讓 EC2 執行個體上執行的程式取得暫時憑證。如需詳細資訊，請參閱 IAM 使用者指南中的利用 IAM 角色來授予許可給 Amazon EC2 執行個體上執行的應用程式。

如需了解是否要使用 IAM 角色或 IAM 使用者，請參閱 IAM 使用者指南中的建立 IAM 角色 (而非使用者) 的時機。

使用政策管理存取權

您可以透 AWS 過建立原則並將其附加至 AWS 身分識別或資源來控制中的存取。原則是一個物件 AWS ，當與身分識別或資源相關聯時，會定義其權限。 AWS 當主參與者 (使用者、root 使用者或角色工作階段) 提出要求時，評估這些原則。政策中的許可決定是否允許或拒絕請求。大多數原則會 AWS 以 JSON 文件的形式儲存在中。如需 JSON 政策文件結構和內容的詳細資訊，請參閱 IAM 使用者指南中的 JSON 政策概觀。

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

預設情況下，使用者和角色沒有許可。若要授予使用者對其所需資源執行動作的許可，IAM 管理員可以建立 IAM 政策。然後，管理員可以將 IAM 政策新增至角色，使用者便能擔任這些角色。

IAM 政策定義該動作的許可，無論您使用何種方法來執行操作。例如，假設您有一個允許 `iam:GetRole` 動作的政策。具有該原則的使用者可以從 AWS Management Console AWS CLI、或 AWS API 取得角色資訊。

身分型政策

身分型政策是可以附加到身分 (例如 IAM 使用者、使用者群組或角色) 的 JSON 許可政策文件。這些政策可控制身分在何種條件下能對哪些資源執行哪些動作。若要了解如何建立身分類型政策，請參閱 IAM 使用者指南中的建立 IAM 政策。

身分型政策可進一步分類成內嵌政策或受管政策。內嵌政策會直接內嵌到單一使用者、群組或角色。受管理的策略是獨立策略，您可以將其附加到您的 AWS 帳戶. 受管政策包括 AWS 受管政策和客戶管理的策略。如需了解如何在受管政策及內嵌政策間選擇，請參閱 IAM 使用者指南中的在受管政策和內嵌政策間選擇。

資源型政策

資源型政策是連接到資源的 JSON 政策文件。資源型政策的最常見範例是 IAM 角色信任政策和 Amazon S3 儲存貯體政策。在支援資源型政策的服務中，服務管理員可以使用它們來控制對特定資源的存取權限。對於附加政策的資源，政策會定義指定的主體可以對該資源執行的動作以及在何種條件下執行的動作。您必須在資源型政策中指定主體。主參與者可以包括帳戶、使用者、角色、同盟使用者或。 AWS 服務

資源型政策是位於該服務中的內嵌政策。您無法在以資源為基礎的政策中使用 IAM 的 AWS 受管政策。

存取控制清單 (ACL)

存取控制清單 (ACL) 可控制哪些主體 (帳戶成員、使用者或角色) 擁有存取某資源的許可。ACL 類似於資源型政策，但它們不使用 JSON 政策文件格式。

Amazon S3 和 Amazon VPC 是支援 ACL 的服務範例。 AWS WAF如需進一步了解 ACL，請參閱 Amazon Simple Storage Service 開發人員指南中的存取控制清單 (ACL) 概觀。

其他政策類型

AWS 支援其他較不常見的原則類型。這些政策類型可設定較常見政策類型授予您的最大許可。

- 許可界限 – 許可範圍是一種進階功能，可供您設定身分型政策能授予 IAM 實體 (IAM 使用者或角色) 的最大許可。您可以為實體設定許可界限。所產生的許可會是實體的身分型政策和其許可界限的交集。會在 `Principal` 欄位中指定使用者或角色的資源型政策則不會受到許可界限限制。所有這類政策中的明確拒絕都會覆寫該允許。如需許可界限的詳細資訊，請參閱 IAM 使用者指南中的 IAM 實體許可界限。
- 服務控制策略（SCP）— SCP 是 JSON 策略，用於指定中組織或組織單位（OU）的最大權限。AWS Organizations AWS Organizations 是一種用於分組和集中管理您企業擁 AWS 帳戶 有的多個服務。若您啟用組織中的所有功能，您可以將服務控制政策 (SCP) 套用到任何或所有帳戶。SCP 限制成員帳戶中實體的權限，包括每個 AWS 帳戶根使用者帳戶。如需 Organizations 和 SCP 的詳細資訊，請參閱 AWS Organizations 使用者指南中的 SCP 運作方式。
- 工作階段政策 – 工作階段政策是一種進階政策，您可以在透過編寫程式的方式建立角色或聯合使用者的暫時工作階段時，作為參數傳遞。所產生工作階段的許可會是使用者或角色的身分型政策和工作階段政策的交集。許可也可以來自資源型政策。所有這類政策中的明確拒絕都會覆寫該允許。如需詳細資訊，請參閱 IAM 使用者指南中的工作階段政策。

多種政策類型

將多種政策類型套用到請求時，其結果形成的許可會更為複雜、更加難以理解。要了解如何在涉及多個政策類型時 AWS 確定是否允許請求，請參閱《IAM 使用者指南》中的政策評估邏輯。

亞馬遜萊克斯如何與 IAM 合作

在您使用 IAM 管理 Amazon Lex 的存取權限之前，請先了解哪些 IAM 功能可與 Amazon Lex 搭配使用。

您可以搭配 Amazon Lex 使用的 IAM 功能

IAM 功能	Amazon Lex 支持
身分型政策	是
資源型政策	否
政策動作	是
政策資源	是
政策條件索引鍵 (服務特定)	是
ACL	否
ABAC(政策中的標籤)	部分
臨時憑證	是
主體許可	是
服務角色	是
服務連結角色	是

若要深入瞭解 Amazon Lex 和其他 AWS 服務如何搭配大多數 IAM 功能使用，請參閱 IAM 使用者指南中的可與 IAM 搭配使用的AWS 服務。

Amazon Lex 基於身份的政策

支援身分型政策	是

身分型政策是可以連接到身分 (例如 IAM 使用者、使用者群組或角色) 的 JSON 許可政策文件。這些政策可控制身分在何種條件下能對哪些資源執行哪些動作。若要了解如何建立身分類型政策，請參閱《IAM 使用者指南》中的建立 IAM 政策。

使用 IAM 身分型政策，您可以指定允許或拒絕的動作和資源，以及在何種條件下允許或拒絕動作。您無法在身分型政策中指定主體，因為這會套用至連接的使用者或角色。如要了解您在 JSON 政策中使用的所有元素，請參閱《IAM 使用者指南》中的 IAM JSON 政策元素參考。

Amazon Lex 的基於身份的政策示例

若要檢視 Amazon Lex 以身分識別為基礎的政策範例，請參閱。Amazon Lex 的基於身份的政策示例

Amazon Lex 內的資源型政策

支援以資源基礎的政策	否

資源型政策是附加到資源的 JSON 政策文件。資源型政策的最常見範例是 IAM 角色信任政策和 Amazon S3 儲存貯體政策。在支援資源型政策的服務中，服務管理員可以使用它們來控制對特定資源的存取權限。對於附加政策的資源，政策會定義指定的主體可以對該資源執行的動作以及在何種條件下執行的動作。您必須在資源型政策中指定主體。主參與者可以包括帳戶、使用者、角色、同盟使用者或。 AWS 服務

如需啟用跨帳戶存取權，您可以指定在其他帳戶內的所有帳戶或 IAM 實體，作為資源型政策的主體。新增跨帳戶主體至資源型政策，只是建立信任關係的一半。當主體和資源位於不同時 AWS 帳戶，受信任帳戶中的 IAM 管理員也必須授與主體實體 (使用者或角色) 權限，才能存取資源。其透過將身分型政策連接到實體來授與許可。不過，如果資源型政策會為相同帳戶中的主體授予存取，這時就不需要額外的身分型政策。如需詳細資訊，請參閱 IAM 使用者指南中的 IAM 中的跨帳戶資源存取。

Amazon Lex 的政策行動

支援政策動作	是

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼条件下可以對什麼資源執行哪些動作。

JSON 政策的 `Action` 元素描述您可以用來允許或拒絕政策中存取的動作。原則動作通常與關聯的 AWS API 作業具有相同的名稱。有一些例外狀況，例如沒有相符的 API 操作的僅限許可動作。也有一些作業需要政策中的多個動作。這些額外的動作稱為相依動作。

政策會使用動作來授予執行相關聯動作的許可。

若要查看 Amazon Lex 動作清單，請參閱服務授權參考資料中由 Amazon Lex 定義的動作。

Amazon Lex 中的政策動作會在動作前使用下列前置詞：

```
lex
```

如需在單一陳述式中指定多個動作，請用逗號分隔。

```
"Action": [
      "lex:action1",
      "lex:action2"
         ]
```

您也可以使用萬用字元 (*) 來指定多個動作。例如，若要指定開頭是 `Describe` 文字的所有動作，請包含以下動作：

```
"Action": "lex:Describe*"
```

Amazon Lex 政策資源

支援政策資源	是

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

`Resource` JSON 政策元素可指定要套用動作的物件。陳述式必須包含 `Resource` 或 `NotResource` 元素。最佳實務是使用其 Amazon Resource Name (ARN) 來指定資源。您可以針對支援特定資源類型的動作 (稱為資源層級許可) 來這麼做。

對於不支援資源層級許可的動作 (例如列出操作)，請使用萬用字元 (*) 來表示陳述式適用於所有資源。

```
"Resource": "*"
```

亞 Amazon Lex 機器人資源 ARN 具有以下格式。

```
arn:aws:lex:${Region}:${Account}:bot:${Bot-Name}
```

如需 ARN 格式的詳細資訊，請參閱 Amazon 資源名稱 (ARN) 和 AWS 服務命名空間。

例如，若要在陳述式中指定 `OrderFlowers` 機器人，請使用以下 ARN。

```
"Resource": "arn:aws:lex:us-east-2:123456789012:bot:OrderFlowers"
```

若要指定所有屬於特定帳戶的機器人，請使用萬用字元 (*)。

```
"Resource": "arn:aws:lex:us-east-2:123456789012:bot:*"
```

某些 Amazon Lex 動作 (例如用於建立資源的動作) 無法在特定資源上執行。在那些情況下，您必須使用萬用字元 (*)。

```
"Resource": "*"
```

若要查看 Amazon Lex 資源類型及其 ARN 的清單，請參閱服務授權參考資料中由 Amazon Lex 定義的資源。若要了解可以使用哪些動作指定每個資源的 ARN，請參閱 Amazon Lex 定義的動作。

Amazon Lex 的政策條件金鑰

支援服務特定政策條件金鑰	是

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

`Condition` 元素 (或 `Condition` 區塊) 可讓您指定使陳述式生效的條件。`Condition` 元素是選用項目。您可以建立使用條件運算子的條件運算式 (例如等於或小於)，來比對政策中的條件和請求中的值。

若您在陳述式中指定多個 Condition 元素，或是在單一 Condition 元素中指定多個索引鍵， AWS 會使用邏輯 AND 操作評估他們。如果您為單一條件索引鍵指定多個值，請使用邏輯OR運算來 AWS 評估條件。必須符合所有條件，才會授與陳述式的許可。

您也可以在指定條件時使用預留位置變數。例如，您可以只在使用者使用其 IAM 使用者名稱標記時，將存取資源的許可授予該 IAM 使用者。如需更多資訊，請參閱 IAM 使用者指南中的 IAM 政策元素：變數和標籤。

AWS 支援全域條件金鑰和服務特定條件金鑰。若要查看所有 AWS 全域條件金鑰，請參閱《IAM 使用者指南》中的AWS 全域條件內容金鑰。

若要查看 Amazon Lex 條件金鑰清單，請參閱服務授權參考資料中的 Amazon Lex 的條件金鑰。若要了解可以使用條件金鑰的動作和資源，請參閱 Amazon Lex 定義的動作。

下表列出適用於 Amazon Lex 資源的 Amazon Lex 條件金鑰。您可以在 IAM 許可政策中的Condition元素中包含這些金鑰。

在亞馬遜萊克斯 ACL

支援 ACL	否

存取控制清單 (ACL) 可控制哪些主體 (帳戶成員、使用者或角色) 擁有存取某資源的許可。ACL 類似於資源型政策，但它們不使用 JSON 政策文件格式。

ABAC 與 Amazon Lex

支援 ABAC (政策中的標籤)	部分

屬性型存取控制 (ABAC) 是一種授權策略，可根據屬性來定義許可。在中 AWS，這些屬性稱為標籤。您可以將標籤附加到 IAM 實體 (使用者或角色) 和許多 AWS 資源。為實體和資源加上標籤是 ABAC 的第一步。您接著要設計 ABAC 政策，允許在主體的標籤與其嘗試存取的資源標籤相符時操作。

ABAC 在成長快速的環境中相當有幫助，並能在政策管理變得繁瑣時提供協助。

如需根據標籤控制存取，請使用 aws:ResourceTag/*key-name*、aws:RequestTag/*key-name* 或 aws:TagKeys 條件索引鍵，在政策的條件元素中，提供標籤資訊。

如果服務支援每個資源類型的全部三個條件金鑰，則對該服務而言，值為 Yes。如果服務僅支援某些資源類型的全部三個條件金鑰，則值為 Partial。

如需 ABAC 的詳細資訊，請參閱 IAM 使用者指南中的什麼是 ABAC?。如要查看含有設定 ABAC 步驟的教學課程，請參閱 IAM 使用者指南中的使用屬性型存取控制 (ABAC)。

您可以將標籤與特定類型的 Amazon Lex 資源建立關聯以進行授權。若要根據標籤控制存取，請使用 `lex:ResourceTag/${TagKey}`、`aws:RequestTag/${TagKey}` 或 `aws:TagKeys` 條件索引鍵，在政策的條件元素中，提供標籤資訊。

如需標記 Amazon Lex 資源的詳細資訊，請參閱標記您的 Amazon Lex 資源。

若要檢視身分型政策範例，以根據該資源上的標籤來限制存取資源，請參閱 使用標籤訪問資源。

下表列出了以標籤為基礎的存取控制動作和對應的資源類型。每個動作都會根據與對應資源類型相關聯的標籤進行授權。

動作	資源類型	條件索引鍵	備註
CreateBotVersion	機器人	`lex:ResourceTag`	
DeleteBot	機器人	`lex:ResourceTag`	
DeleteBotAlias	別名	`lex:ResourceTag`	
DeleteBotChannelAssociation	通道	`lex:ResourceTag`	
DeleteBotVersion	機器人	`lex:ResourceTag`	
DeleteSession	機器人或別名	`lex:ResourceTag`	當別名設定為 $LATEST 時，使用與機器人相關聯的標籤。與其他別名一起使用時，會使用與指定別名相關聯的標籤。
DeleteUtterances	機器人	`lex:ResourceTag`	
GetBot	機器人或別名	`lex:ResourceTag`	當 `versionOrAlias` 設定為 $LATEST 或數字版本

動作	資源類型	條件索引鍵	備註
			時，則使用與機器人相關聯的標籤。與別名一起使用時，使用與指定別名相關聯的標籤
GetBotAlias	別名	lex:ResourceTag	
GetBotChannelAssociation	頻道	lex:ResourceTag	
GetBotChannelAssociations	頻道	lex:ResourceTag	當別名設為「-」時，使用與機器人相關聯的標籤。當機器人別名已指定時，使用與指定別名相關聯的標籤
GetBotVersions	機器人	lex:ResourceTag	
GetExport	機器人	lex:ResourceTag	
GetSession	機器人或別名	lex:ResourceTag	當別名設定為 $LATEST 時，使用與機器人相關聯的標籤。與其他別名一起使用時，會使用與指定別名相關聯的標籤。
GetUtterancesView	機器人	lex:ResourceTag	
ListTagsForResource	機器人、別名或頻道	lex:ResourceTag	

動作	資源類型	條件索引鍵	備註
PostContent	機器人或別名	lex:ResourceTag	當別名設定為 $LATEST 時，使用與機器人相關聯的標籤。與其他別名一起使用時，會使用與指定別名相關聯的標籤。
PostText	機器人或別名	lex:ResourceTag	當別名設定為 $LATEST 時，使用與機器人相關聯的標籤。與其他別名一起使用時，會使用與指定別名相關聯的標籤。
PutBot	機器人	lex:ResourceTag, aws:RequestTag, aws:TagKeys	
PutBotAlias	別名	lex:ResourceTag, aws:RequestTag, aws:TagKeys	
PutSession	機器人或別名	lex:ResourceTag	當別名設定為 $LATEST 時，使用與機器人相關聯的標籤。與其他別名一起使用時，會使用與指定別名相關聯的標籤。

動作	資源類型	條件索引鍵	備註
StartImport	機器人	lex:ResourceTag	依賴 PutBot 操作的存取政策。該 StartImport 操作的特定標籤和權限將被忽略。
TagResource	機器人、別名或頻道	lex:ResourceTag, aws:RequestTag, aws:TagKeys	
UntagResource	機器人、別名或頻道	lex:ResourceTag, aws:RequestTag, aws:TagKeys	

透過 Amazon Lex 使用臨時登入資料

支援臨時憑證	是

當您使用臨時憑據登錄時，某些 AWS 服務 不起作用。如需其他資訊，包括哪些 AWS 服務 與臨時登入資料搭配AWS 服務 使用，請參閱 IAM 使用者指南中的 IAM。

如果您使用除了使用者名稱和密碼以外的任何方法登入，則您正在 AWS Management Console 使用臨時認證。例如，當您 AWS 使用公司的單一登入 (SSO) 連結存取時，該程序會自動建立暫時認證。當您以使用者身分登入主控台，然後切換角色時，也會自動建立臨時憑證。如需切換角色的詳細資訊，請參閱 IAM 使用者指南中的切換至角色 (主控台)。

您可以使用 AWS CLI 或 AWS API 手動建立臨時登入資料。然後，您可以使用這些臨時登入資料來存取 AWS。 AWS 建議您動態產生臨時登入資料，而非使用長期存取金鑰。如需詳細資訊，請參閱 IAM 中的暫時性安全憑證。

您可以搭配聯合使用暫時憑證、擔任 IAM 角色，或是擔任跨帳戶角色。您可以透過呼叫AssumeRole或GetFederation權杖等 AWS STS API 作業來取得臨時安全登入資料。

Amazon Lex 的跨服務主體許可

支援轉寄存取工作階段 (FAS)	是

當您使用 IAM 使用者或角色在中執行動作時 AWS，您會被視為主體。使用某些服務時，您可能會執行某個動作，進而在不同服務中啟動另一個動作。FAS 會使用主體呼叫的權限 AWS 服務，並結合要求 AWS 服務 向下游服務發出要求。只有當服務收到需要與其 AWS 服務 他資源互動才能完成的請求時，才會發出 FAS 請求。在此情況下，您必須具有執行這兩個動作的許可。如需提出 FAS 請求時的政策詳細資訊，請參閱《轉發存取工作階段》。

Amazon Lex 的服務角色

支援服務角色	是

服務角色是服務擔任的 IAM 角色，可代您執行動作。IAM 管理員可以從 IAM 內建立、修改和刪除服務角色。如需詳細資訊，請參閱 IAM 使用者指南中的建立角色以委派許可給 AWS 服務服務。

> **Warning**
>
> 變更服務角色的許可可可可可能會中斷 Amazon Lex 的功能。只有在 Amazon Lex 提供指導時才編輯服務角色。

在 Amazon Lex 選擇 IAM 角色

Amazon Lex 使用服務鏈接角色來調用 Amazon Comprehend 和 Amazon Polly。它使用您的 AWS Lambda 函數的資源級權限來調用它們。

您必須提供 IAM 角色才能啟用交談標記。如需詳細資訊，請參閱 建立對話日誌的 IAM 角色和政策。

Amazon Lex 的服務連結角色

支援服務連結角色	是

服務連結角色是連結至. AWS 服務服務可以擔任代表您執行動作的角色。服務連結角色會顯示在您的中， AWS 帳戶 且屬於服務所有。IAM 管理員可以檢視，但不能編輯服務連結角色的許可。

如需建立或管理 Amazon Lex 服務連結角色的詳細資訊，請參閱使用 Amazon Lex 的服務連結角色。

Amazon Lex 的基於身份的政策示例

依預設，使用者和角色沒有建立或修改 Amazon Lex 資源的權限。他們也無法使用 AWS Management Console、 AWS Command Line Interface (AWS CLI) 或 AWS API 來執行工作。若要授予使用者對其所需資源執行動作的許可，IAM 管理員可以建立 IAM 政策。然後，管理員可以將 IAM 政策新增至角色，使用者便能擔任這些角色。

若要了解如何使用這些範例 JSON 政策文件建立 IAM 身分型政策，請參閱《IAM 使用者指南》中的建立 IAM 政策。

如需 Amazon Lex 定義的動作和資源類型的詳細資訊，包括每種資源類型的 ARN 格式，請參閱服務授權參考中的 Amazon Lex 的動作、資源和條件金鑰。

主題

- 政策最佳實務
- 使用 Amazon Lex 控制台
- 允許使用者檢視他們自己的許可
- 刪除所有 Amazon Lex 機
- 允許使用者將機器人移轉至 Amazon Lex V2 API
- 使用標籤訪問資源

政策最佳實務

以身分識別為基礎的政策決定某人是否可以在您的帳戶中建立、存取或刪除 Amazon Lex 資源。這些動作可能會讓您的 AWS 帳戶產生費用。當您建立或編輯身分型政策時，請遵循下列準則及建議事項：

- 開始使用 AWS 受管原則並邁向最低權限權限 — 若要開始授與使用者和工作負載的權限，請使用可授與許多常見使用案例權限的AWS 受管理原則。它們在您的 AWS 帳戶. 建議您透過定義特定於您使用案例的 AWS 客戶管理政策，進一步降低使用權限。如需更多資訊，請參閱 IAM 使用者指南中的 AWS 受管政策或任務職能的AWS 受管政策。
- 套用最低權限許可 – 設定 IAM 政策的許可時，請僅授予執行任務所需的許可。為實現此目的，您可以定義在特定條件下可以對特定資源採取的動作，這也稱為最低權限許可。如需使用 IAM 套用許可的更多相關資訊，請參閱 IAM 使用者指南中的 IAM 中的政策和許可。
- 使用 IAM 政策中的條件進一步限制存取權 – 您可以將條件新增至政策，以限制動作和資源的存取。例如，您可以撰寫政策條件，指定必須使用 SSL 傳送所有請求。您也可以使用條件來授與對服務動

作的存取權 (如透過特定) 使用這些動作 AWS 服務，例如 AWS CloudFormation。如需詳細資訊，請參閱 IAM 使用者指南中的 IAM JSON 政策元素：條件。

- 使用 IAM Access Analyzer 驗證 IAM 政策，確保許可安全且可正常運作 – IAM Access Analyzer 驗證新政策和現有政策，確保這些政策遵從 IAM 政策語言 (JSON) 和 IAM 最佳實務。IAM Access Analyzer 提供 100 多項政策檢查及切實可行的建議，可協助您編寫安全且實用的政策。如需更多資訊，請參閱 IAM 使用者指南中的 IAM Access Analyzer 政策驗證。
- 需要多因素身份驗證 (MFA) — 如果您的案例需要 IAM 使用者或根使用者 AWS 帳戶，請開啟 MFA 以獲得額外的安全性。如需在呼叫 API 操作時請求 MFA，請將 MFA 條件新增至您的政策。如需更多資訊，請參閱 IAM 使用者指南中的設定 MFA 保護的 API 存取。

如需 IAM 中最佳實務的相關資訊，請參閱 IAM 使用者指南中的 IAM 安全最佳實務。

使用 Amazon Lex 控制台

若要存取 Amazon Lex 主控台，您必須擁有最少一組許可。這些許可必須允 Amazon Lex 在 AWS 帳戶. 如果您建立比最基本必要許可更嚴格的身分型政策，則對於具有該政策的實體 (使用者或角色) 而言，主控台就無法如預期運作。

您不需要為僅對 AWS CLI 或 AWS API 進行呼叫的使用者允許最低主控台權限。反之，只需允許存取符合他們嘗試執行之 API 操作的動作就可以了。

AWS 透過提供由建立和管理的獨立 IAM 政策來解決許多常見使用案例 AWS。這些政策稱為 AWS 受管政策。相較於必須自己編寫政策，使用 AWS 受管政策可以更輕鬆地將適用的許可，指派給使用者、群組和角色。如需詳細資訊，請參閱《IAM 使用者指南》中的 AWS Managed Policies (AWS 受管政策)。

Amazon Lex 專用的下列 AWS 受管政策 (您可以附加到帳戶中的群組和角色)：

- AmazonLexReadOnly— 授予對 Amazon Lex 資源的唯讀存取權。
- AmazonLexRunBots僅限 — 授予執行 Amazon Lex 交談機器人的存取權。
- AmazonLexFullAccess— 授予建立、讀取、更新、刪除和執行所有 Amazon Lex 資源的完整存取權。此外，還授予將名稱以開頭的 Lambda 函數`AmazonLex`與 Amazon Lex 意圖建立關聯的能力。

> **Note**
>
> 您可以登入 IAM 主控台並搜尋特定政策，以檢閱這些許可政策。

該AmazonLexFullAccess政策不會授予使用者使用KendraSearchIntent意圖查詢 Amazon Kendra 索引的權限。若要查詢索引，您必須將其他權限新增至策略。如要了解必要的許可，請參閱 Amazon Kendra 搜索的 IAM 政策。

您也可以建立自己的自訂 IAM 政策，以允許 Amazon Lex API 動作的許可。您可以將這些自訂政策附加到需要這些權限的 IAM 角色或群組。

如需適用於 Amazon Lex 的 AWS 受管政策的詳細資訊，請參閱AWS亞馬遜萊克斯的受管政策。

允許使用者檢視他們自己的許可

此範例會示範如何建立政策，允許 IAM 使用者檢視附加到他們使用者身分的內嵌及受管政策。此原則包含在主控台上或以程式設計方式使用 AWS CLI 或 AWS API 完成此動作的權限。

```
{
    "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": "*"
```

```
        }
    ]
}
```

刪除所有 Amazon Lex 機

此範例政策授予 AWS 帳戶中的使用者刪除帳戶中任何機器人的權限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "lex:DeleteBot"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

允許使用者將機器人移轉至 Amazon Lex V2 API

下列 IAM 權限政策可讓使用者開始將機器人從 Amazon Lex 遷移到 Amazon Lex V2 API，並查看移轉清單及其進度。

```
{
    "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": "*"
        }
    ]
}
```

使用標籤訪問資源

此範例策略授予 AWS 帳號中的使用者或角色權限，以便將PostText作業與任何標記為索引鍵**Department**和值的資源搭配使用**Support**。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": "lex:PostText",
            "Effect": "Allow",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "lex:ResourceTag/Department": "Support"
                }
            }
        }
    ]
}
```

AWS亞馬遜萊克斯的受管政策

AWS 受管政策是由 AWS 建立和管理的獨立政策。AWS 受管政策的設計在於為許多常見使用案例提供許可，如此您就可以開始將許可指派給使用者、群組和角色。

請記住，AWS 受管政策可能不會授與您特定使用案例的最低權限許可，因為它們可供所有 AWS 客戶使用。我們建議您定義使用案例專屬的客戶管理政策，以便進一步減少許可。

您無法更改 AWS 受管政策中定義的許可。如果 AWS 更新 AWS 受管政策中定義的許可，更新會影響政策連接的所有主體身分 (使用者、群組和角色)。在推出新的 AWS 服務 或有新的 API 操作可供現有服務使用時，AWS 很可能會更新 AWS 受管政策。

如需詳細資訊，請參閱《IAM 使用者指南》https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_managed-vs-inline.html#aws-managed-policies中的 AWS 受管政策。

AWS 受管政策：AmazonLexReadOnly

您可將 `AmazonLexReadOnly` 政策連接到 IAM 身分。

此政策授予唯讀許可，允許使用者檢視 Amazon Lex 和 Amazon Lex V2 模型建置服務中的所有動作。

許可詳細資訊

此政策包含以下許可：

- `lex`— 模型建置服務中的 Amazon Lex 和亞馬遜 Lex V2 資源的唯讀存取權。

```
{
    "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": "*"
        }
    ]
}
```

AWS 受管政策：AmazonLexRunBotsOnly

您可將 AmazonLexRunBotsOnly 政策連接到 IAM 身分。

此政策授予唯讀許可，允許存取執行 Amazon Lex 和 Amazon Lex V2 交談機器人。

許可詳細資訊

此政策包含以下許可：

- lex— 只讀訪問亞馬遜 Lex 和亞馬遜 Lex V2 運行時中的所有操作。

```
{
    "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": "*"
        }
    ]
}
```

AWS 受管政策：AmazonLexFullAccess

您可將 AmazonLexFullAccess 政策連接到 IAM 身分。

此政策授予管理許可，允許使用者建立、讀取、更新和刪除 Amazon Lex 和 Amazon Lex V2 資源，以及執行 Amazon Lex 和 Amazon Lex V2 交談機器人的權限。

許可詳細資訊

此政策包含以下許可：

- lex— 允許主體讀取和寫入 Amazon Lex 和 Amazon Lex V2 模型建置和執行時期服務中的所有動作。
- cloudwatch— 允許校長查看亞馬遜CloudWatch指標和警報。
- iam— 可讓主參與者建立和刪除服務連結角色、傳遞角色，以及將原則附加至角色和中斷連結。權限被限制為「亞馬遜」亞馬遜萊克斯操作和「Lexv2. 亞馬遜 V2」亞馬遜萊克斯 V2 操作.
- kendra— 允許校長列出亞馬遜肯德拉索引。

- kms— 允許主參與者描述AWS KMS金鑰和別名。
- lambda— 允許主參與者列出AWS Lambda函數和管理連接到任何 Lambda 函數的權限。
- polly— 允許校長描述亞馬遜波莉聲音並合成語音。

```
{
    "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"
                    ]
                }
            }
        }
    ]
}
```

亞馬遜萊克斯更新AWS受管理政策

檢視有關更新的詳細資訊AWS自從這項服務開始追蹤這些變更後，Amazon Lex 的受管政策。如需有關此頁面變更的自動警示，請訂閱 Amazon Lex 上的 RSS 摘要Amazon Lex 的文件歷史記錄頁面。

變更	描述	日期
AmazonLexFullAccess – 更新現有政策	Amazon Lex 新增了新的許可，允許對 Amazon Lex V2 模型建置服務作業進行唯讀存取。	2021 年 8 月 18 日
AmazonLexReadOnly – 更新現有政策	Amazon Lex 新增了新的許可，允許對 Amazon Lex V2 模型建置服務作業進行唯讀存取。	2021 年 8 月 18 日
AmazonLexRunBotsOnly – 更新現有政策	Amazon Lex 新增了新的許可，允許對 Amazon Lex V2 執行階段服務作業進行唯讀存取。	2021 年 8 月 18 日

變更	描述	日期
亞馬遜 Lex 開始跟踪更改	亞馬遜萊克斯開始跟踪其更改AWS受管理的策略。	2021 年 8 月 18 日

使用 Amazon Lex 的服務連結角色

Amazon Lex 使用AWS Identity and Access Management (IAM) 服務連結角色。服務連結角色是直接連結至 Amazon Lex 的一種特殊 IAM 角色類型。服務連結角色由 Amazon Lex 預先定義，內含該服務代您呼叫其他服AWS務所需的所有許可。

服務連結角色可讓設定 Amazon Lex 更為簡單，因為您不必手動新增必要的許可。Amazon Lex 定義其服務連結角色的許可，除非另有定義，否則僅有 Amazon Lex 可以擔任其角色。定義的許可包括信任政策和許可政策，並且該許可政策不能連接到任何其他 IAM 實體。

您必須先刪除角色的相關資源，才能刪除服務連結角色。如此可保護您的 Amazon Lex 資源，避免您不小心移除資源的存取許可。

Amazon Lex 的服務連結角色許可

Amazon Lex 使用兩個服務連結角色：

- AWSServiceRoleForLexBots— Amazon Lex 使用此服務連結角色呼叫 Amazon Polly 來為您的機器人合成語音回應、呼叫 Amazon Comprehend 進行情緒分析，並選擇性使用 Amazon Kendra 來搜尋索引。
- AWSServiceRoleForLexChannels— Amazon Lex 使用此服務連結角色在管理通道時將文字張貼到您的機器人。

您必須設定許可，IAM 實體 (如使用者、群組或角色) 才可建立、編輯或刪除服務連結角色。如需詳細資訊，請參閱《IAM 使用者指南》中的服務連結角色許可。

建立 Amazon Lex 的服務連結角色

您不需要手動建立一個服務連結角色。當您在中建立機器人、機器人通道或 Amazon Kendra 搜尋意圖時AWS Management Console，Amazon Lex 會為您建立服務連結角色。

若您刪除此服務連結角色，之後需要再次建立，您可以在帳戶中使用相同程序重新建立角色。當您建立新的機器人、通道關聯或 Amazon Kendra 搜尋意圖時，Amazon Lex 會再次為您建立服務連結角色。

您也可以使用AWS CLI使用案例以建立服務連結角色。AWSServiceRoleForLexBots在AWS CLI建立Amazon Lex 服務名稱的服務連結角色lex.amazonaws.com。如需詳細資訊，請參閱步驟 1：建立服務連結角色 (AWS CLI)。如果您刪除此服務連結角色，您可以使用此相同的程序以再次建立該角色。

編輯 Amazon Lex 的服務連結角色

Amazon Lex 不允許您編輯 Amazon Lex 服務連結角色。因為有各種實體可能會參考服務連結角色，所以您無法在建立角色之後變更角色名稱。然而，您可使用 IAM 來編輯角色描述。如需更多資訊，請參閱 IAM 使用者指南中的編輯服務連結角色。

刪除 Amazon Lex 的服務連結角色

若您不再使用需要服務連結角色的功能或服務，我們建議您刪除該角色。如此一來，您就沒有未主動監控或維護的未使用實體。然而，在手動刪除服務連結角色之前，您必須先清除資源。

> **Note**
>
> 若 Amazon Lex 服務在您試圖刪除資源時正在使用該角色，刪除可能會失敗。若此情況發生，請等待數分鐘後並再次嘗試操作。

若要刪除服務連結角色所使用的 Amazon Lex 資源：

1. 刪除您正在使用的任何機器人管道。
2. 刪除您帳戶中的任何機器人。

使用 IAM 手動刪除服務連結角色

使用 IAM 主控台AWS CLI、或AWS API 刪除 Amazon Lex 服務連結角色。如需詳細資訊，請參閱 IAM 使用者指南中的刪除服務連結角色。

Amazon Lex 服務連結角色的支援區域

Amazon Lex 在所有提供服務的區域中支援使用服務連結角色。如需詳細資訊，請參閱 Amazon Lex 端點和配額。

疑難排解 Amazon Lex 身分和存取

使用下列資訊可協助您診斷和修正使用 Amazon Lex 和 IAM 時可能會遇到的常見問題。

主題

我沒有授權在 Amazon Lex 中執行操作

如果您收到錯誤，告知您未獲授權執行動作，您的政策必須更新，允許您執行動作。

下列範例錯誤會在mateojackson IAM 使用者嘗試使用主控台檢視一個虛構 *my-example-widget* 資源的詳細資訊，但卻無虛構 lex:*GetWidget* 許可時發生。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 lex:GetWidget on resource: my-example-widget
```

在此情況下，必須更新 mateojackson 使用者的政策，允許使用 lex:*GetWidget* 動作存取 *my-example-widget* 資源。

如果您需要協助，請聯絡您的 AWS 管理員。您的管理員提供您的簽署憑證。

我沒有授權執行 iam：PassRole

如果您收到未獲授權執行iam:PassRole動作的錯誤訊息，則必須更新您的政策以允許您將角色傳遞給 Amazon Lex。

有些 AWS 服務 允許您將現有角色傳遞給該服務，而不是建立新的服務角色或服務連結角色。如需執行此作業，您必須擁有將角色傳遞至該服務的許可。

當名為的 IAM 使用者marymajor嘗試使用主控台在 Amazon Lex 中執行動作時，會發生下列範例錯誤。但是，動作請求服務具備服務角色授予的許可。Mary 沒有將角色傳遞至該服務的許可。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

在這種情況下，Mary 的政策必須更新，允許她執行 iam:PassRole 動作。

如果您需要協助，請聯絡您的 AWS 管理員。您的管理員提供您的簽署憑證。

我想允許我以外的人訪 AWS 帳戶 問我的 Amazon Lex 資源

您可以建立一個角色，讓其他帳戶中的使用者或您組織外部的人員存取您的資源。您可以指定要允許哪些信任物件取得該角色。針對支援基於資源的政策或存取控制清單 (ACL) 的服務，您可以使用那些政策來授予人員存取您的資源的許可。

如需進一步了解，請參閱以下內容：

- 若要了解 Amazon Lex 是否支援這些功能，請參閱亞馬遜萊克斯如何與 IAM 合作。
- 若要了解如何提供對您所擁有資源 AWS 帳戶 的存取權，請參閱《IAM 使用者指南》中您擁有的另一 AWS 帳戶 個 IAM 使用者提供存取權限。
- 若要了解如何將資源存取權提供給第三方 AWS 帳戶，請參閱 IAM 使用者指南中的提供第三方 AWS 帳戶 擁有的存取權。
- 如需了解如何透過聯合身分提供存取權，請參閱 IAM 使用者指南中的將存取權提供給在外部進行身分驗證的使用者 (聯合身分)。
- 若要了解跨帳戶存取使用角色和以資源為基礎的政策之間的差異，請參閱 IAM 使用者指南中的 IAM 中的跨帳戶資源存取。

在 Amazon Lex 監控

監控對於維持 Amazon Lex 聊天機器人的可靠性、可用性和效能非常重要。本主題說明如何使用 Amazon CloudWatch 日誌和AWS CloudTrail監控 Amazon Lex，並說明 Amazon Lex 執行階段和通道關聯指標。

主題

- 與亞馬遜監控亞馬遜 Lex CloudWatch
- 使用AWS CloudTrail日誌監控 Amazon Lex API 呼叫

與亞馬遜監控亞馬遜 Lex CloudWatch

若要追蹤 Amazon Lex 機器人的運作狀態，請使用 Amazon CloudWatch。使用此功能 CloudWatch，您可以取得個別 Amazon Lex 作業的指標，或為您的帳戶取得全球 Amazon Lex 操作的指標。您也可以設定 CloudWatch 警示，以便在一或多個量度超過您定義的臨界值時收到通知。例如，您可以監控特定期間內對某機器人所提出的請求數量，檢視成功請求的延遲，或在錯誤超出閾值時發出警示。

CloudWatch Amazon Lex 的指標

若要取得 Amazon Lex 操作的指標，您必須指定下列資訊：

- 指標維度。維度是用來識別量度的一組名稱-值配對。Amazon Lex 有三個維度：
  - `BotAlias, BotName, Operation`
  - `BotAlias, BotName, InputMode, Operation`
  - `BotName, BotVersion, InputMode, Operation`
- 指標名稱，例如 `MissedUtteranceCount` 或 `RuntimeRequestCount`。

您可以使用AWS Management Console、或 CloudWatch API 取得 Amazon Lex 的指標。AWS CLI您可以透過其中 CloudWatch 一個 Amazon AWS 軟體開發套件 (開發套件) 或 CloudWatch API 工具使用 API。Amazon Lex 主控台會根據 CloudWatch API 中的原始資料顯示圖形。

您必須擁有適當的 CloudWatch 許可才能監控 Amazon Lex CloudWatch 。如需詳細資訊，請參閱 Amazon CloudWatch 使用者指南 CloudWatch中的 Amazon 身份驗證和存取控制。

檢視 Amazon Lex 指標

使用 Amazon Lex 主控台或主控台檢視 Amazon Lex 指 CloudWatch標。

若要檢視指標 (Amazon Lex 主控台)

1. 登錄到AWS Management Console並打開 Amazon Lex 控制台 https://console.aws.amazon.com/lex/.
2. 從機器人清單選擇您要查看指標的機器人。
3. 選擇 Monitoring (監控)。指標會顯示在圖形中。

若要檢視量度 (CloudWatch 主控台)

1. 請登入AWS Management Console並開啟 CloudWatch 主控台，網址為 https://console.aws.amazon.com/cloudwatch/。
2. 選擇指標，選擇所有指標，然後選擇 AWS/Lex。
3. 選擇維度、選擇指標名稱，再選擇 Add to graph (新增至圖形)。
4. 選擇日期範圍的值。所選日期範圍的指標計數會顯示在圖形中。

建立警示

CloudWatch 警示會監視指定時段內的單一指標，並執行一或多個動作：傳送通知至 Amazon Simple Notification Service (Amazon SNS) 主題或 Auto Scaling 政策。動作或動作是根據您指定數個期間內，相對於指定臨界值的測量結果值。 CloudWatch也可以在警示狀態變更時傳送 Amazon SNS 訊息給您。

CloudWatch 警示只有在狀態變更且您指定的期間內持續存在時，才會呼叫動作。

設定警示

1. 請登入AWS Management Console並開啟 CloudWatch 主控台，網址為 https://console.aws.amazon.com/cloudwatch/。
2. 選擇 Alarms (警示)，然後選擇 Create Alarm (建立警示)。
3. 選擇 AWS/Lex Metrics指標，然後選擇一個指標。
4. 對於 Time Range (時間範圍)，選擇要監控的時間範圍，然後選擇 Next (下一步)。
5. 輸入 Name (名稱) 和 Description (描述)。
6. 對於 Whenever (每當)，選擇 **>=** 並輸入最大值。
7. 如果您要 CloudWatch 在到達鬧鐘狀態時傳送電子郵件，請在 [動作] 區段中，針對 [每當此警示] 選擇 [狀態為鬧鐘]。對於 Send notification to (傳送通知至)，選擇郵件清單或選擇 New list (新清單) 並建立新的郵件清單。
8. 在 Alarm Preview (警示預覽) 區段中預覽警示。如果警示符合您的要求，選擇 Create Alarm (建立警示)。

CloudWatchAmazon Lex 運行時的指標

下表說明 Amazon Lex 執行階段指標。

指標	描述
`KendraIndexAccessError`	Amazon Lex 無法訪問您的亞 Amazon Kendra 索引的次數。 使用 `Text` 或 `Speech` `InputMode` 進行 `PostContent` 操作的有效維度： • `BotName, BotAlias, Operation, InputMode`

指標	描述
	PostText 操作的有效維度： • BotName, BotAlias, Operation 單位：計數
KendraLatency	Amazon Kendra 回應來自的請求所AMAZON.KendraSearchIntent 花費的時間量。 使用 Text 或 Speech InputMode 進行 PostContent 操作的有效維度： • BotName, BotVersion, Operation, InputMode • BotName, BotAlias, Operation, InputMode PostText 操作的有效維度： • BotName, BotVersion, Operation • BotName, BotAlias, Operation 單位：毫秒

指標	描述
KendraSuccess	從您的 Amazon Kendra 索引發出AMAZON.KendraSearchIntent 的成功請求數目。 使用 Text 或 Speech InputMode 進行 PostContent 操作的有效維度： • BotName, BotVersion, Operation, InputMode • BotName, BotAlias, Operation, InputMode PostText 操作的有效維度： • BotName, BotVersion, Operation • BotName, BotAlias, Operation 單位：計數
KendraSystemErrors	Amazon Lex 無法查詢亞馬 Amazon Kendra 索引的次數。 使用 Text 或 Speech InputMode 進行 PostContent 操作的有效維度： • BotName, BotAlias, Operation, InputMode PostText 操作的有效維度： • BotName, BotAlias, Operation 單位：計數

指標	描述
KendraThrottledEvents	Amazon Kendra 限制請求的次數。AMAZON.KendraSearchIntent 使用 Text 或 Speech InputMode 進行 PostContent 操作的有效維度： • BotName, BotAlias, Operation, InputMode PostText 操作的有效維度： • BotName, BotAlias, Operation 單位：計數
MissedUtteranceCount	指定期間內無法辨識的表達用語數量。 使用 Text 或 Speech InputMode 進行 PostContent 操作的有效維度： • BotName, BotVersion, Operation, InputMode • BotName, BotAlias, Operation, InputMode PostText 操作的有效維度： • BotName, BotVersion, Operation • BotName, BotAlias, Operation

指標	描述
RuntimeConcurrency	指定時間範圍內同時連線的數目。 RuntimeConcurrency 報告為StatisticSet . 使用 Text 或 Speech InputMode 進行 PostContent 操作的有效維度： • 操作， BotName， BotVersion， InputMode • 操作， BotName， BotAlias， InputMode 其他操作的有效尺寸： • 操作， BotName， BotVersion • 操作， BotName， BotAlias 單位：計數
RuntimeInvalidLambdaResponses	指定期間內無效 AWS Lambda (Lambda) 回應的數目。 使用 Text 或 Speech InputMode 進行 PostContent 操作的有效維度： • BotName, BotAlias, Operation, InputMode PostText 操作的有效維度： • BotName, BotAlias, Operation

指標	描述
RuntimeLambdaErrors	指定期間內的 Lambda 執行階段錯誤數目。 使用 Text 或 Speech InputMode 進行 PostContent 操作的有效維度： • BotName, BotAlias, Operation, InputMode PostText 操作的有效維度： • BotName, BotAlias, Operation
RuntimePollyErrors	指定期間內無效的 Amazon Polly 回應數目。 使用 Text 或 Speech InputMode 進行 PostContent 操作的有效維度： • BotName, BotAlias, Operation, InputMode PostText 操作的有效維度： • BotName, BotAlias, Operation
RuntimeRequestCount	指定期間內的執行時間請求數量。 使用 Text 或 Speech InputMode 進行 PostContent 操作的有效維度： • BotName, BotVersion, Operation, InputMode • BotName, BotAlias, Operation, InputMode PostText 操作的有效維度： • BotName, BotVersion, Operation • BotName, BotAlias, Operation 單位：計數

指標	描述
RuntimeSucessfulRequestLatency **Important** 此量度是RuntimeSuccessfulRequestLatency ，而不是RuntimeSuccessfulRequestLatency 。	提出請求與傳回回應期間的成功請求延遲。 使用 Text 或 Speech InputMode 進行 PostContent 操作的有效維度： • BotName, BotVersion, Operation, InputMode • BotName, BotAlias, Operation, InputMode PostText 操作的有效維度： • BotName, BotVersion, Operation • BotName, BotAlias, Operation 單位：毫秒
RuntimeSystemErrors	指定期間內的系統錯誤數量。系統錯誤的回應碼範圍是 500 到 599。 使用 Text 或 Speech InputMode 進行 PostContent 操作的有效維度： • BotName, BotAlias, Operation, InputMode PostText 操作的有效維度： • BotName, BotAlias, Operation 單位：計數

指標	描述
RuntimeThrottledEvents	已調節的請求數目。當 Amazon Lex 收到的請求數量超過您帳戶設定的每秒交易限制時，會調節請求。如果經常超過為您的帳戶所設的限制，您可以請求提高上限。若要請求提高，請參閱 AWS Service Limits。 使用 Text 或 Speech InputMode 進行 PostContent 操作的有效維度： • BotName, BotAlias, 操作, InputMode PostText 操作的有效維度： • BotName, BotAlias, Operation 單位：計數
RuntimeUserErrors	指定期間內的使用者錯誤數量。使用者錯誤的回應碼範圍是 400 到 499。 使用 Text 或 Speech InputMode 進行 PostContent 操作的有效維度： • BotName, BotAlias, Operation, InputMode PostText 操作的有效維度： • BotName, BotAlias, Operation 單位：計數

Amazon Lex 執行階段指標會使用AWS/Lex命名空間，並以下列維度提供指標。您可以在 CloudWatch 主控台中依維度分組量度：

維度	描述
BotName, BotAlias, Operation, InputMode	依照機器人別名、機器人名稱、操作 (PostContent) 及文字或語音輸入，為指標進行分組。
BotName, BotVersion, Operation, InputMode	依照機器人名稱、機器人版本、操作 (PostContent) 及文字或語音輸入，為指標進行分組。
BotName, BotVersion, Operation	依照機器人名稱、機器人版本和操作 (PostText)，為指標進行分組。
BotName, BotAlias, Operation	依照機器人名稱、機器人別名和操作 (PostText)，為指標進行分組。

CloudWatch Amazon Lex 通道協會的指標

通道關聯是 Amazon Lex 和消息傳遞渠道（例如 Facebook）之間的關聯。下表說明 Amazon Lex 通道關聯指標。

指標	描述
BotChannelAuthErrors	簡訊管道在指定期間內傳回的身分驗證錯誤數量。身分驗證錯誤表示在管道建立期間所提供的秘密字符無效或已過期。
BotChannelConfigurationErrors	指定期間內的組態錯誤數量。組態錯誤表示管道有一或多個組態項目無效。
BotChannelInboundThrottledEvents	Amazon Lex 在指定期間內限制簡訊通道傳送訊息的次數。
BotChannelOutboundThrottledEvents	在指定時間段內限制從 Amazon Lex 傳出事件至簡訊通道的次數。
BotChannelRequestCount	指定期間內在管道上提出的請求數量。

指標	描述
BotChannelResponseCardErrors	Amazon Lex 在指定期間內無法張貼回應卡的次數。
BotChannelSystemErrors	指定期間內某個通道在 Amazon Lex 中發生的內部錯誤數目。

Amazon Lex 通道關聯指標會使用AWS/Lex命名空間，並提供下列維度的指標。您可以在 CloudWatch 主控台中依維度分組量度：

維度	描述
BotAlias, BotChannelName, BotName, Source	依照機器人別名、管道名稱、機器人名稱和流量來源，為指標進行分組。

CloudWatch 交談記錄的度量

Amazon Lex 使用下列指標進行交談記錄：

指標	描述
ConversationLogsAudioDeliverySuccess	在指定時段成功遞送至 S3 儲存貯體的音訊日誌數目。 單位：計數
ConversationLogsAudioDeliveryFailure	在指定時段無法遞送至 S3 儲存貯體的音訊日誌數目。遞送失敗表示針對對話日誌設定的資源發生錯誤。錯誤可能包括 IAM 許可不足、無法存取的AWS KMS金鑰或無法存取的 S3 儲存貯體。 單位：計數

指標	描述
ConversationLogsTextDeliverySuccess	在指定期間內成功傳遞至 CloudWatch 記錄檔的文字記錄數目。 單位：計數
ConversationLogsTextDeliveryFailure	在指定期間內無法傳送至 CloudWatch 記錄檔的文字記錄數目。遞送失敗表示針對對話日誌設定的資源發生錯誤。錯誤可能包括 IAM 許可不足、無法存取的AWS KMS金鑰或無法存取的 CloudWatch 記錄日誌群組。 單位：計數

Amazon Lex 交談日誌指標使用AWS/Lex命名空間，並為以下維度提供指標。您可以在 CloudWatch 主控台中依維度分組量度。

維度	描述
BotAlias	依機器人別名將指標分組。
BotName	依機器人名稱將指標分組。
BotVersion	依機器人版本將指標分組。

使用AWS CloudTrail日誌監控 Amazon Lex API 呼叫

Amazon Lex 與此服務整合在一起AWS CloudTrail，可提供 Amazon Lex 中使用者、角色或服務所採取的動作記錄的AWS服務。 CloudTrail 在事件中擷取 Amazon Lex 的 API 呼叫子集，包括來自 Amazon Lex 主控台的呼叫，以及從程式碼呼叫到 Amazon Lex API 的呼叫。如果您建立追蹤，您可以啟用持續交付 CloudTrail 事件到 Amazon S3 儲存貯體，包括 Amazon Lex 的事件。如果您未設定追蹤，您仍然可以在 [事件歷程記錄] 中檢視 CloudTrail 主控台中最近的事件。使用收集的資訊 CloudTrail，您可以判斷向 Amazon Lex 發出的請求、提出請求的 IP 地址、提出請求的人員、提出請求的時間以及其他詳細資訊。

若要進一步了解 CloudTrail，包括如何設定和啟用它，請參閱AWS CloudTrail使用者指南。

Amazon Lex 信息 CloudTrail

CloudTrail 在您創建AWS帳戶時，您的帳戶已啟用。Amazon Lex 中發生受支援的事件活動時，該活動會與事件歷史記錄中的其他AWS服務 CloudTrail 事件一起記錄在事件中。您可以檢視、搜尋和下載 AWS 帳戶的最新事件。如需詳細資訊，請參閱檢視具有事 CloudTrail 件記錄的事件。

如需AWS帳戶中持續記錄事件 (包括 Amazon Lex 的事件)，請建立追蹤。追蹤可 CloudTrail 將日誌檔交付到 Amazon Simple Storage Service (Amazon S3) 儲存貯體。根據預設，當您在主控台建立追蹤記錄時，追蹤記錄會套用到所有 AWS 區域。該追蹤會記錄來自 AWS 分割區中所有區域的事件，並將日誌檔案交付到您指定的 S3 儲存貯體。此外，您還可以設定其他AWS服務，以進一步分析 CloudTrail 記錄中收集的事件資料並採取行動。如需詳細資訊，請參閱：

- 建立追蹤的概觀
- CloudTrail 支援的服務與整合
- 設定 CloudTrail 的 Amazon SNS 通知
- 從多個區域接收 CloudTrail 記錄檔並從多個帳戶接收 CloudTrail 記錄檔

Amazon Lex 支援將下列操作記錄為記 CloudTrail 錄檔中的事件：

- 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

每一筆事件或日誌項目都會包含產生請求者的資訊。此資訊可協助您判斷下列事項：

- 該請求是否使用根或 使用者登入資料提出
- 提出該請求時，是否使用了特定角色或聯合身分使用者的臨時安全憑證
- 該請求是否由另一項 AWS 服務提出

如需詳細資訊，請參閱 CloudTrail 使用者身分元素。

如需 CloudTrail 記錄日誌中之 Amazon Lex 動作的相關資訊，請參閱 Amazon Lex 模型建置服務。例如，呼叫PutBotGetBot、和DeleteBot作業會在 CloudTrail 記錄中產生項目。記錄在 Amazon Lex 執行時間服務、PostContent 和 PostText 中的動作都不會記錄。

範例：Amazon Lex 日誌檔項目

追蹤是一種組態，可讓事件以日誌檔的形式傳遞至您指定的 S3 儲存貯體。 CloudTrail 記錄檔包含一或多個記錄項目。事件代表來自任何來源的單一請求，包括有關請求的操作，動作的日期和時間，請求參數等信息。 CloudTrail 日誌文件不是公共 API 調用的有序堆棧跟踪，因此它們不會以任何特定順序顯示。

下列範例 CloudTrail 記錄項目顯示呼叫作PutBot業的結果。

```
   {
                "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"
        }
    }
```

Amazon Lex 的合規驗證

第三方稽核員會評估 Amazon Lex 的安全性和合規性，做為多個 AWS 合規計劃的一部分。Amazon Lex 是 HIPAA 合格的服務。它符合 PCI、SOC 和 ISO 標準。您可以使用下載第三方稽核報告 AWS Artifact。如需詳細資訊，請參閱在 AWS Artifact 中下載報告。

使用 Amazon Lex 時的合規責任取決於資料的敏感度、組織的合規目標以及適用的法律和法規。如果您使用 Amazon Lex 時必須遵守 PCI 等標準，請 AWS 提供下列資源協助：

- 安全性與合規性快速入門指南 — 部署指南，討論架構考量，並提供步驟，以部署以安全性和法規遵循為重點的基準環境 AWS
- HIPAA 安全與合規架構白皮書 - 本白皮書說明公司可如何運用 AWS 來建立 HIPAA 合規的應用程式。
- AWS 合規資源 — 可能適用於您的產業和地點的工作簿和指南集合
- AWS Config— 評估您的資源配置如何符合內部實踐，行業準則和法規的服務
- AWS Security Hub— 全面檢視您的安全狀態，可協助 AWS 您檢查您是否符合安全性產業標準和最佳做法

如需特定合規計劃範圍內的 AWS 服務清單，請參閱合規計劃的 AWS 服務範圍。如需一般資訊，請參閱 AWS 合規計劃。

Amazon Lex 的彈性

AWS 全球基礎架構是圍繞區 AWS 域和可用區域建立的。 AWS 區域提供多個實體分離和隔離的可用區域，這些區域透過低延遲、高輸送量和高度備援的網路連線。透過可用區域，您所設計與操作的應用程式和資料庫，就能夠在可用區域之間自動容錯移轉，而不會發生中斷。可用區域的可用性、容錯能力和擴充能力，均較單一或多個資料中心的傳統基礎設施還高。

如需區域和可用區域的相關 AWS 資訊，請參閱AWS 全域基礎結構。

除了 AWS 全球基礎設施之外，Amazon Lex 還提供多種功能來協助支援您的資料恢復和備份需求。

Amazon Lex 的基礎設施

作為受管服務，Amazon Lex 受到 Amazon Web Services：安 AWS 全流程概觀白皮書中所述的全球網路安全程序的保護。

您可以使用已發佈的 AWS API 呼叫透過網路存取 Amazon Lex。用戶端必須支援 TLS (Transport Layer Security) 1.0。建議使用 TLS 1.2 或更新版本。用戶端也必須支援具備完美轉送私密 (PFS) 的密碼套件，例如臨時 Diffie-Hellman (DHE) 或橢圓曲線臨時 Diffie-Hellman (ECDHE)。現代系統 (如 Java 7 和更新版本) 大多會支援這些模式。此外，請求必須使用存取金鑰 ID 和與 IAM 主體相關聯的私密存取金鑰來簽署。或者，您可以透過 AWS Security Token Service (AWS STS) 來產生暫時安全憑證來簽署請求。

您可以從任何網路位置呼叫這些 API 操作，但 Amazon Lex 支援資源層級存取政策，其中可能包含以來源 IP 位址為基礎的限制。您也可以使用 Amazon Lex 政策來控制來自特定 Amazon Virtual Private Cloud 端 (Amazon VPC) 端點或特定 VPC 的存取。實際上，這會將對指定 Amazon Lex 資源的網路存取從網路內的特定 VPC 隔離出來 AWS 。

Amazon Lex 的準則和配額

以下各節提供使用 Amazon Lex 時的準則和配額。

主題

支援的區域

如需可用 Amazon Lex 的AWS清單，請參閱《Amazon Web Services 一般參考》中的 AWS 區域和端點。

一般準則

本節說明使用 Amazon Lex 時的一般準則。

- 簽署請求 —API 參考 使用簽名 V4 中的所有 Amazon Lex 模型建置和執行時期 API 操作來驗證請求。如需有關驗證請求的詳細資訊，請參閱《Amazon Web Services 一般參考》中的 Signature 第 4 版簽署程序。

  對於PostContent，Amazon Lex 在 Amazon Simple Storage Service (S3)) API 參考中，針對授權標頭：在單一區塊中傳輸承載 (AWS 簽名版本 4) 中描述的簽名計算中所述的未簽署承載選項。

  當使用未簽署的承載選項時，請勿在正式請求中包含承載的雜湊。您可以改用字串常值「UNSIGNED-PAYLOAD」做為承載的雜湊。另外也請在 `x-amz-content-sha256` 請求中包含名稱 `UNSIGNED-PAYLOAD` 和數值 `PostContent` 的標頭。

- 請注意以下有關 Amazon Lex 如何從使用者話語擷取插槽值的相關資訊：

Amazon Lex 使用您在插槽類型定義中提供的列舉值來訓練其機器學習模型。假設您使用以下範例表達用語，來定義稱為 GetPredictionIntent 的意圖：

```
"Tell me the prediction for {Sign}"
```

當中 {Sign} 是自訂類型為 ZodiacSign 的槽。它有 12 個列舉值，Aries 至 Pisces。從用戶的話語「告訴我預測...」 Amazon Lex 了解以下是一個十二生肖。

當valueSelectionStrategy欄位設定為ORIGINAL_VALUE使用PutSlotType操作，或者在主控台中選取「展開」值時，如果使用者說「告訴我地球的預測」，Amazon Lex 會推斷「earth」是ZodiacSign並將其傳遞給您的用戶端應用程式或 Lambda 函數。您必須先檢查槽值具有有效的值，再將其用於您的履行活動。

如果您使用 PutSlotType 操作將 valueSelectionStrategy 欄位設定為 TOP_RESOLUTION，或是主控台中已選取Restrict to slot values and synonyms (限制為槽值和同義詞)，則傳回的值會受限於您為槽類型定義的值。例如，如果使用者表示「我想知道地球的運勢預測」，將無法辨識該值，因為它不屬於為槽類型定義的值之一。當為槽值定義同義詞時，會將其視同槽值，不過，傳回的是槽值，而不是同義詞。

當 Amazon Lex 呼叫 Lambda 函數或傳回與用戶端應用程式的語音互動結果時，無法保證插槽值的情況。例如，如果您引出 Amazon.Movie 內置插槽類型的值，並且用戶說或輸入「隨風飄逝」，Amazon Lex 可能會返回「隨風而去」，「隨風而去」。在文字互動中，槽值的大小寫會符合輸入的文字或槽值，端視 valueResolutionStrategy 欄位的值而定。

- 定義包含縮寫的槽值時，請使用下列模式：
  - 以句點分隔的大寫字母 (V.D.)
  - 以空格分隔的大寫字母 (D V D)

- Amazon Lex 不支持亞馬遜。Alexa 技能套件支持的文字內置插槽類型。不過，Amazon Lex 支援建立可用來實作此功能的自訂插槽類型。如前一點所述，您可以擷取自訂槽類型定義以外的值。新增更多和多樣化的列舉值來提升自動語音辨識 (ASR) 和自然語言了解 (NLU) 的準確性。

- AMAZON.DATE 和 AMAZON.TIME 內建槽類型會同時擷取絕對和相對的日期和時間。相對日期和時間可在 Amazon Lex 處理請求的區域中解決。

  對於AMAZON.TIME內建插槽類型，如果使用者未指定時間是中午之前或之後，則時間不明確，Amazon Lex 會再次提示使用者。我們建議提示引出絕對的時間。例如，使用如「您希望比薩何時送達？ 您可以說下午 6 點或傍晚 6 點」的提示。

- 在機器人中提供混淆的訓練資料，可減少 Amazon Lex 瞭解使用者輸入的能力。請考量以下範例：

  假設您的機器人中有兩個意圖 (OrderPizza 和 OrderDrink)，而且兩者都是以「我想要訂購」的表達用語來設定。這個話語並不會對應到 Amazon Lex 在建置階段為機器人建立語言模型時所能學到的特定目的。因此，當使用者在執行階段輸入此語音時，Amazon Lex 無法挑選具有高度信心的意圖。

  再來看看另一個範例，您定義了自訂意圖向使用者獲取確認 (例如，MyCustomConfirmationIntent) 並以表達用語「是」和「否」來設定意圖。請注意，Amazon Lex 也有一種語言模型，可用於瞭解使用者確認。這可能會產生衝突的情況。當使用者以「是」回應時，這是表示確認進行中的意圖，還是確認使用者請求您所建立的自訂意圖？

  一般來說，您提供的範例表達用語應該對應到特定的意圖，或是選擇對應到特定槽值。

- 執行時間 API 操作 PostContent 和 PostText 會將使用者 ID 視為必要的參數。開發人員可以將此設定為符合 API 中所述之限制的任何值。我們建議您不要使用此參數來傳送任何機密資訊 (例如使用者登入、電子郵件或身分證號碼。這個 ID 主要是用來唯一識別與機器人的對話 (可能有多個使用者訂購外送的比薩)。

- 如果您的用戶端應用程式使用 Amazon Cognito 進行身份驗證，您可以使用 Amazon Cognito 使用者識別碼做為 Amazon Lex 使用者識別碼。請注意，為您的機器人設定的任何 Lambda 函數都必須具有自己的身份驗證機制，以識別代表 Amazon Lex 叫用 Lambda 函數的使用者。

- 我們鼓勵您定義一個意圖來捕捉使用者停止對話的意圖。例如，您可以使用示例語音`NothingIntent`（「我不想要任何東西」，「退出」，「再見」）定義一個意圖（），沒有插槽，也沒有將 Lambda 函數配置為代碼掛鉤。這可讓使用者從容地關閉對話。

配額

本節說明 Amazon Lex 目前的配額。這些配額依類別分組。

您可以調整或增加服務配額。請聯絡AWS客戶支援以增加配額。增加 Services 配額可能需要幾天。如果您要在較大專案中增加配額，請務必將這段時間加入您的計劃中。

主題

執行時間服務配額

除了 API 參考中所述的配額之外，請注意以下事項：

API 配額

- PostContent 操作的語音輸入最長可達 15 秒。

- 在兩項執行時間 API 操作 PostContent 和 PostText 中，輸入文字大小最多可達 1024 個 Unicode 字元。

- PostContent 標頭的大小上限為 16 KB。請求和工作階段標頭的總大小上限為 12 KB。

- 在文字模式下使用PostContent或PostText作業時，與機器人的並行交談數目上限為 2 個$LATEST別名，所有其他別名為 50 個。配額分別適用於每個 API。

- 在語音模式下使用PostContent操作時，與機器人的並行文字模式交談數目上限為 2，所有其他別名為 125 個。$LATEST配額分別適用於每個 API。

- 機器人$LATEST別名的並行工作階段管理呼叫 (PutSessionGetSession、和DeleteSession) 數目上限為 2，所有其他別名則為 50 個。

- Lambda 函數的最大輸入大小為 12 KB。輸出大小上限為 25 KB，其中 12 KB 可以是工作階段屬性。

使用 **$LATEST** 版本

- 您的機器人$LATEST版本只能用於手動測試。Amazon Lex 限制了您可以對機器人$LATEST版本發出的執行時間請求數量。

- 當您更新機器人的$LATEST版本時，Amazon Lex 會使用該機器人$LATEST版本終止任何用戶端應用程式的任何進行中交談。一般而言，您不應該在生產環境中使用 $LATEST 版本的機器人，因為 $LATEST 版本可能會更新。您應該改發佈一個版本，並使用該版本。

- 當您更新別名時，Amazon Lex 需要幾分鐘的時間來取得變更。當修改 $LATEST 版本的機器人時，該變更會立即生效。

工作階段逾時

- 在建立機器人時所設定的工作階段逾時會決定機器人要保留對談內容多久的時間，例如目前使用者意圖和槽資料。

- 使用者開始與您的機器人交談，直到工作階段到期為止，Amazon Lex 會使用相同的機器人版本，即使您將機器人別名更新為指向其他版本也是如此。

模型建置配額

模型建置是指建立和管理機器人。這包括建立和管理機器人、意圖、槽類型、槽和機器人管道關聯。

主題

- 機器人配額
- 意圖配額
- 槽類型配額

機器人配額

- 您可在整個模型建置 API 間設定提示和陳述。每個提示或陳述最多可有 5 個訊息，且每則訊息可包含 1 到 1000 個 UTF-8 字元。

- 當使用訊息群組時，您可以為每則訊息定義最多五個訊息群組。每個訊息群組可包含最多 5 個訊息，而所有訊息群組最多只能有 15 個訊息。

- 您可以為意圖和槽定義範例表達用語。所有表達用語最多可使用 200,000 個字元。

- 每個槽類型可以定義最多 10,000 個值和同義詞。每個機器人最多可包含 50,000 個槽類型值和同義詞。

- 機器人、別名和機器人管道關聯名稱在建立時不區分大小寫。如果您建立 PizzaBot，然後嘗試建立 pizzaBot，您會收到錯誤。然而，當存取資源時，資源名稱有區分大小寫，您就必須指定 PizzaBot 而非 pizzaBot。這些名稱長度必須介於 2 到 50 個 ASCII 字元之間。

- 您可以發佈的所有資源類型版本上限是 100 個版本。請注意，別名沒有版本控制。

- 在機器人中，意圖名稱和槽名稱必須是唯一的，您不能有同名的意圖和槽。

- 您可以建立機器人並設定為支援多個意圖。如果兩個意圖有同名的槽，則對應的插槽類型也必須相同。

  例如，假設您建立一個機器人來支援兩個意圖 (OrderPizza 和 OrderDrink)。如果這兩個意圖都有 size 槽，那麼槽類型在兩個地方中都必須相同。

  此外，您為其中一個意圖中的槽提供的範例表達用語，也適用於在另一個意圖中同名的槽。

- 您可以把 250 個意圖的上限數量與一個機器人相關聯。

- 您在建立機器人時指定工作階段逾時。工作階段逾時可以介於一分鐘到一天之間。預設值為五分鐘。

- 您可以為機器人建立最多 5 個別名。

- 您最多可以建立 250 個 AWS 帳戶中建立 250 個機器人。

- 您不能建立多個意圖從相同的內建意圖擴展。

意圖配額

- 意圖和槽名稱在建立時不區分大小寫。也就是說，如果您建立 OrderPizza 意圖，然後再次嘗試建立另一個 orderPizza 意圖，您會收到錯誤。然而，當存取這些資源時，資源名稱有區分大小寫，因此要指定 OrderPizza 而非 orderPizza。這些名稱長度必須介於 1 到 100 個 ASCII 字元之間。

- 意圖最多可有 1,500 個範例表達用語。必須至少有一個範例表達用語。每個範例表達用語長度最多可達 200 個 UTF-8 字元。一個機器人中的所有意圖和槽最多可以使用 200,000 個字元。意圖的範例表達用語：
  - 可參考零或多個槽名稱。
  - 僅可參考槽名稱一次。

  例如：

  ```
  I want a pizza
  I want a {pizzaSize} pizza
  I want a {pizzaSize} {pizzaTopping} pizza
  ```

- 雖然每個意圖最多支援 1,500 個語音，但如果您使用較少的話語，Amazon Lex 可能會有更好的辨識您所提供音樂集之外的輸入。

- 您可以在一個意圖中為每個訊息建立最多 5 個訊息群組。一個訊息的所有訊息群組中總共可有 15 個訊息。

- 主控台只能建立 conclusionStatement 和 followUpPrompt 訊息的訊息群組。您可以使用 Amazon Lex API 為任何其他訊息建立訊息群組。

- 每個槽最多可有 10 個範例表達用語。每個範例表達用語必須確實參考槽名稱一次。例如：

  ```
  {pizzaSize} please
  ```

- 每個機器人的意圖和槽最多共可有 200,000 個字元。

- 您不能為從內建意圖擴展的意圖提供表達用語。對於所有其他意圖，您必須至少提供一個範例表達用語。意圖包含槽，但是槽層級的範例表達用語是選用的。

- 內建槽
  - 目前，Amazon Lex 不支援內建意圖的插槽引出功能。您無法建立 Lambda 函數，以便在回應中傳回含有衍生自內建意圖的意圖的`ElicitSlot`指示詞。如需詳細資訊，請參閱回應格式。
  - 此服務不支援新增範例表達用語到內建意圖。同樣地，您無法在內建意圖新增或移除槽。

- 您可以為每個 AWS 帳戶建立最多 1,000 個意圖。您可以在意圖中建立最多 100 個槽。

槽類型配額

- 槽類型名稱在建立時不區分大小寫。如果您建立 `PizzaSize` 槽類型，然後再次嘗試建立另一個 `pizzaSize` 槽類型，您會收到錯誤。然而，當存取這些資源時，資源名稱有區分大小寫，您就必須指定 `PizzaSize` 而非 `pizzaSize`。名稱長度必須介於 1 到 100 個 ASCII 字元之間。

- 您建立的自訂槽類型最多可有 10,000 個列舉值和同義詞。每個值長度最多可達 140 個 UTF-8 字元。列舉值和同義詞不能包含重複值。

- 對於槽類型值，請在適當時，指定大小寫。例如，對於稱為 `Procedure` 的槽類型，如果值為 `MRI`，請指定「MRI」和「mri」的值。

- 內建插槽類型 — 目前 Amazon Lex 不支援為內建插槽類型新增列舉值或同義字。

API 參考

本節提供 Amazon Lex API 操作的說明文件。如需提供 Amazon Lex 的 AWS 區域清單，請參閱 Amazon Web Services 一般參考中的 AWS 區域和端點。

主題

- 動作
- 資料類型

動作

Amazon Lex 模型構建服務支援下列動作：

- 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

Amazon Lex 運行時服務支援下列動作：

- DeleteSession
- GetSession
- PostContent
- PostText

- PutSession

Amazon Lex 模型構建服務

Amazon Lex 模型構建服務支援下列動作：

- 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

服務：Amazon Lex Model Building Service

根據版本創建一個新版$LATEST本的機器人。如果自建立上一個$LATEST版本後，此資源的版本並未變更，Amazon Lex 不會建立新版本。它返回上次創建的版本。

> **Note**
>
> 您只能更新機器人的$LATEST版本。您無法更新使用CreateBotVersion作業建立的編號版本。

當您建立機器人的第一個版本時，Amazon Lex 會將版本設定為 1。後續版本會累加 1。如需詳細資訊，請參閱 版本控制。

這項操作需要 lex:CreateBotVersion 動作的許可。

請求語法

```
POST /bots/name/versions HTTP/1.1
Content-type: application/json

{
   "checksum": "string"
}
```

URI 請求參數

請求會使用下列 URI 參數。

name

您要建立新版本的機器人名稱。名稱區分大小寫。

長度約束：最小長度為 2。長度上限為 50。

模式：^([A-Za-z]_?)+$

必要：是

請求主體

請求接受採用 JSON 格式的下列資料。

checksum

識別機器人$LATEST版本的特定修訂版本。如果您指定總和檢查碼，且機器人的$LATEST版本具有不同的總和檢查碼，則會傳回PreconditionFailedException例外狀況，而 Amazon Lex 不會發佈新版本。如果您未指定總和檢查碼，Amazon Lex 會發佈該$LATEST版本。

類型：字串

必要：否

回應語法

```
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"
}
```

回應元素

如果動作成功，則服務傳回 HTTP 201 回應。

服務會傳回下列 JSON 格式的資料。

abortStatement

Amazon Lex 用來取消對話的訊息。如需詳細資訊，請參閱 PutBot。

類型：Statement 物件

checksum

識別所建立機器人版本的總和檢查碼。

類型：字串

childDirected

對於使用 Amazon Lex 模型建置服務建立的每個 Amazon Lex 機器人，您必須在`childDirected`欄位中指定或全部或部分針對 13 歲以下兒童並遵守《兒童線上隱私保護法》(COPPA) 的網站、程式或其他應用程式是否與全部或部分導向或鎖定目標的網站、程式`true`或其他應用程`false`式相關。透過**`true`**在**`childDirected`**欄位中指定，即表示您確認您對 Amazon Lex 的使用與針對 13 歲以下兒童且受 COPPA 規範的全部或部分針對網站、程式或其他應用程式

有關。透過`false`在`childDirected`欄位中指定，即表示您確認您對 Amazon Lex 的使用與針對 13 歲以下兒童且受 COPPA 規範的全部或部分針對網站、程式或其他應用程式無關。您不得為欄位指定預設值，該`childDirected`欄位無法準確反映您對 Amazon Lex 的使用情況是否與針對 13 歲以下兒童的網站、程式或其他應用程式全部或部分針對 13 歲以下且受 COPPA 規限的網站、程式或其他應用程式有關。

如果您對 Amazon Lex 的使用涉及全部或部分針對 13 歲以下兒童的網站、程式或其他應用程式，您必須根據 COPPA 取得任何必要的可驗證父母同意。如需針對全部或部分針對 13 歲以下兒童的網站、程式或其他應用程式使用 Amazon Lex 的相關資訊，請參閱 Amazon Lex 常見問題集。

類型：布林值

clarificationPrompt

Amazon Lex 在不了解使用者要求時所使用的訊息。如需詳細資訊，請參閱 PutBot。

類型：Prompt 物件

createdDate

建立機器人版本的日期。

類型：Timestamp

description

機器人的描述。

類型：字串

長度限制：長度下限為 0。長度上限為 200。

detectSentiment

指出是否應將使用者輸入的語音傳送至 Amazon Comprehend 進行情緒分析。

類型：布林值

enableModelImprovements

指出機器人是否使用精確度改進。 `true`表示機器人正在使用改進，否則，`false`。

類型：布林值

failureReason

如果`status`是`FAILED`，Amazon Lex 會提供無法建置機器人的原因。

類型：字串

idleSessionTTLInSeconds

Amazon Lex 保留交談中收集之資料的時間上限 (以秒為單位)。如需詳細資訊，請參閱 PutBot。

類型：整數

有效範圍：最小值為 60。最大值為 86400。

intents

Intent 物件的陣列。如需詳細資訊，請參閱 PutBot。

類型：Intent 物件陣列

lastUpdatedDate

更新此機器人$LATEST版本的日期。

類型：Timestamp

locale

指定機器人的目標地區設定。

類型：字串

有效值: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

機器人的名稱。

類型：字串

長度約束：最小長度為 2。長度上限為 50。

模式：^([A-Za-z]_?)+$

status

當您傳送建立或更新機器人的請求時，Amazon Lex 會將status回應元素設定為BUILDING。在 Amazon Lex 建置機器人之後，它就會設定status為READY。如果 Amazon Lex 無法建置機器人，它就會設定status為FAILED。Amazon Lex 返回failureReason響應元素失敗的原因。

類型：字串

有效值:BUILDING | READY | READY_BASIC_TESTING | FAILED | NOT_BUILT

version

機器人的版本。

類型：字串

長度限制：長度下限為 1。長度上限為 64。

模式：\$LATEST|[0-9]+

voiceId

亞馬遜 Lex 用於與用戶進行語音交互的 Amazon Polly 語音 ID。

類型：字串

錯誤

BadRequestException

請求的格式不正確。例如，值無效或缺少必填欄位。請檢查欄位值，然後再試一次。

HTTP 狀態碼：400

ConflictException

處理要求時發生衝突。請再次嘗試您的請求。

HTTP 狀態碼：409

InternalFailureException

發生內部 Amazon Lex 錯誤。請再次嘗試您的請求。

HTTP 狀態碼：500

LimitExceededException

請求超過限制。請再次嘗試您的請求。

HTTP 狀態碼：429

NotFoundException

找不到要求中指定的資源。請檢查資源，然後再試一次。

HTTP 狀態碼：404

PreconditionFailedException

您嘗試變更之資源的總和檢查碼與要求中的總和檢查碼不符。檢查資源的總和檢查碼，然後再試一次。

HTTP 狀態碼：412

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面
- AWS SDK for .NET
- AWS SDK for C++
- AWS 適用於轉到 V2 的 SDK
- AWS SDK for Java V2 的軟件
- AWS 適用於 JavaScript V3 的 SDK
- AWS SDK for PHP
- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

CreateIntentVersion

服務：Amazon Lex Model Building Service

根據意圖的版本建立意圖的新$LATEST版本。如果此意圖的$LATEST版本在您上次更新之後沒有變更，Amazon Lex 不會建立新版本。它返回您創建的最後一個版本。

> **Note**
>
> 您只能更新意圖的$LATEST版本。您無法更新使用CreateIntentVersion作業建立的編號版本。

當您建立意圖版本時，Amazon Lex 會將版本設定為 1。後續版本會累加 1。如需詳細資訊，請參閱 版本控制。

這項操作需要許可來執行 lex:CreateIntentVersion 動作。

請求語法

```
POST /intents/name/versions HTTP/1.1
Content-type: application/json

{
   "checksum": "string"
}
```

URI 請求參數

請求會使用下列 URI 參數。

name

您要建立新版本的意圖名稱。名稱區分大小寫。

長度限制：長度下限為 1。長度上限為 100。

模式：^([A-Za-z]_?)+$

必要：是

請求主體

請求接受採用 JSON 格式的下列資料。

checksum

應該用來建立新$LATEST版本之意圖版本的總和檢查碼。如果您指定總和檢查碼，且意圖的$LATEST版本具有不同的總和檢查碼，Amazon Lex 會傳回PreconditionFailedException例外狀況，而不會發佈新版本。如果您未指定總和檢查碼，Amazon Lex 會發佈該$LATEST版本。

類型：字串

必要：否

回應語法

```
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"
}
```

回應元素

如果動作成功，則服務傳回 HTTP 201 回應。

服務會傳回下列 JSON 格式的資料。

checksum

建立之意圖版本的總和檢查碼。

類型：字串

conclusionStatement

在`fulfillmentActivity`欄位中指定的 Lambda 函數達成意圖之後，Amazon Lex 就會將此陳述式傳達給使用者。

類型：Statement 物件

confirmationPrompt

如果已定義，Amazon Lex 在完成使用者意圖之前使用的提示確認。

類型：Prompt 物件

createdDate

建立意圖的日期。

類型：Timestamp

description

意圖的描述。

類型：字串

長度限制：長度下限為 0。長度上限為 200。

dialogCodeHook

如果已定義，Amazon Lex 會針對每個使用者輸入叫用此 Lambda 函數。

類型：CodeHook 物件

followUpPrompt

如果已定義，Amazon Lex 會在完成意圖後使用此提示來徵求其他使用者活動。

類型：FollowUpPrompt 物件

fulfillmentActivity

描述如何實現意圖。

類型：FulfillmentActivity 物件

inputContexts

`InputContext`物件陣列，列出 Amazon Lex 必須處於作用中狀態的內容，才能在與使用者交談中選擇意圖。

類型：InputContext 物件陣列

陣列成員：項目數下限為 0。項目數上限為 5。

kendraConfiguration

用於將 Amazon Kendra 索引與`AMAZON.KendraSearchIntent`意圖連接的組態資訊 (如果有的話)。

類型：KendraConfiguration 物件

lastUpdatedDate

更新意圖的日期。

類型：Timestamp

name

意圖的名稱。

類型：字串

長度限制：長度下限為 1。長度上限為 100。

模式：^([A-Za-z]_?)+$

outputContexts

OutputContext物件陣列，列出實現意圖時意圖啟動的前後關聯。

類型：OutputContext 物件陣列

陣列成員：項目數下限為 0。項目數上限為 10。

parentIntentSignature

內建意圖的唯一識別碼。

類型：字串

rejectionStatement

如果使用者對中定義的問題回答「否」confirmationPrompt，Amazon Lex 會以此聲明回應，確認意圖已取消。

類型：Statement 物件

sampleUtterances

針對意圖設定的範例語音陣列。

類型：字串陣列

陣列成員：項目數下限為 0。最多可容納 1500 件物品數量。

長度限制：長度下限為 1。長度上限為 200。

slots

插槽類型陣列，用於定義實現意圖所需的資訊。

類型：Slot 物件陣列

陣列成員：項目數下限為 0。項目數上限為 100。

version

指派給意圖新版本的版本號碼。

類型：字串

長度限制：長度下限為 1。長度上限為 64。

模式：\$LATEST|[0-9]+

錯誤

BadRequestException

請求的格式不正確。例如，值無效或缺少必填欄位。請檢查欄位值，然後再試一次。

HTTP 狀態碼：400

ConflictException

處理要求時發生衝突。請再次嘗試您的請求。

HTTP 狀態碼：409

InternalFailureException

發生內部 Amazon Lex 錯誤。請再次嘗試您的請求。

HTTP 狀態碼：500

LimitExceededException

請求超過限制。請再次嘗試您的請求。

HTTP 狀態碼：429

NotFoundException

找不到要求中指定的資源。請檢查資源，然後再試一次。

HTTP 狀態碼：404

PreconditionFailedException

您嘗試變更之資源的總和檢查碼與要求中的總和檢查碼不符。檢查資源的總和檢查碼，然後再試一次。

HTTP 狀態碼：412

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面
- AWS SDK for .NET
- AWS SDK for C++
- AWS 適用於轉到 V2 的 SDK
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 JavaScript V3 的 SDK
- AWS SDK for PHP
- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

CreateSlotTypeVersion

服務：Amazon Lex Model Building Service

根據指定插槽類型的$LATEST版本建立插槽類型的新版本。如果自上次建立$LATEST版本以來，此資源的版本並未變更，Amazon Lex 不會建立新版本。它返回您創建的最後一個版本。

> **Note**
>
> 您只能更新插槽類型的$LATEST版本。您無法更新使用CreateSlotTypeVersion作業建立的編號版本。

當您建立插槽類型的版本時，Amazon Lex 會將版本設定為 1。後續版本會累加 1。如需詳細資訊，請參閱 版本控制。

這項操作需要 lex:CreateSlotTypeVersion 動作的許可。

請求語法

```
POST /slottypes/name/versions HTTP/1.1
Content-type: application/json

{
   "checksum": "string"
}
```

URI 請求參數

請求會使用下列 URI 參數。

name

您要為其建立新版本之插槽類型的名稱。名稱區分大小寫。

長度限制：長度下限為 1。長度上限為 100。

模式：^([A-Za-z]_?)+$

必要：是

請求主體

請求接受採用 JSON 格式的下列資料。

checksum

您要發佈之插槽類型$LATEST版本的總和檢查碼。如果您指定總和檢查碼，且插槽類型的$LATEST版本具有不同的總和檢查碼，Amazon Lex 會傳回PreconditionFailedException例外狀況且不會發佈新版本。如果您未指定總和檢查碼，Amazon Lex 會發佈該$LATEST版本。

類型：字串

必要：否

回應語法

```
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"
```

```
}
```

回應元素

如果動作成功，則服務傳回 HTTP 201 回應。

服務會傳回下列 JSON 格式的資料。

checksum

插槽類型$LATEST版本的總和檢查碼。

類型：字串

createdDate

插槽類型的建立日期。

類型：Timestamp

description

位置類型的描述。

類型：字串

長度限制：長度下限為 0。長度上限為 200。

enumerationValues

定義插槽類型可採用的值的EnumerationValue物件清單。

類型：EnumerationValue 物件陣列

陣列成員：項目數下限為 0。最多可包含 1 萬個項目。

lastUpdatedDate

更新插槽類型的日期。建立資源時，建立日期和上次更新日期相同。

類型：Timestamp

name

位置類型的名稱。

類型：字串

長度限制：長度下限為 1。長度上限為 100。

模式：^([A-Za-z]_?)+$

parentSlotTypeSignature

內建插槽類型使用插槽類型的父系。

類型：字串

長度限制：長度下限為 1。長度上限為 100。

模式：^((AMAZON\.)_?|[A-Za-z]_?)+

slotTypeConfigurations

擴充上層內建插槽類型的組態資訊。

類型：SlotTypeConfiguration 物件陣列

陣列成員：項目數下限為 0。項目數上限為 10。

valueSelectionStrategy

Amazon Lex 用來判斷插槽價值的策略。如需詳細資訊，請參閱 PutSlotType。

類型：字串

有效值:ORIGINAL_VALUE | TOP_RESOLUTION

version

指派給新插槽類型版本的版本。

類型：字串

長度限制：長度下限為 1。長度上限為 64。

模式：\$LATEST|[0-9]+

錯誤

BadRequestException

請求的格式不正確。例如，值無效或缺少必填欄位。請檢查欄位值，然後再試一次。

HTTP 狀態碼：400

ConflictException

處理要求時發生衝突。請再次嘗試您的請求。

HTTP 狀態碼：409

InternalFailureException

發生內部 Amazon Lex 錯誤。請再次嘗試您的請求。

HTTP 狀態碼：500

LimitExceededException

請求超過限制。請再次嘗試您的請求。

HTTP 狀態碼：429

NotFoundException

找不到要求中指定的資源。請檢查資源，然後再試一次。

HTTP 狀態碼：404

PreconditionFailedException

您嘗試變更之資源的總和檢查碼與要求中的總和檢查碼不符。檢查資源的總和檢查碼，然後再試一次。

HTTP 狀態碼：412

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面
- AWS SDK for .NET
- AWS SDK for C++
- AWS 適用於轉到 V2 的 SDK
- AWS SDK for Java V2 的軟件
- AWS 適用於 JavaScript V3 的 SDK

- AWS SDK for PHP
- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

DeleteBot

服務：Amazon Lex Model Building Service

刪除機器人的所有版本，包括$LATEST版本。要刪除機器人的特定版本，請使用DeleteBotVersion操作。作DeleteBot業不會立即移除機器人結構描述。相反，它被標記為刪除，稍後將其刪除。

Amazon Lex 無限期存放語音，以改善機器人回應使用者輸入的能力。刪除機器人時，這些語音不會被刪除。若要移除語音，請使用此作DeleteUtterances業。

如果機器人有別名，您就無法刪除它。此DeleteBot作業會傳回ResourceInUseException例外狀況，其中包含參照機器人之別名的參考。若要移除機器人的參照，請刪除別名。如果您再次收到相同的例外狀況，請刪除反向連結別名，直到DeleteBot作業成功為止。

這項操作需要 lex:DeleteBot 動作的許可。

請求語法

```
DELETE /bots/name HTTP/1.1
```

URI 請求參數

請求會使用下列 URI 參數。

name

機器人的名稱。名稱區分大小寫。

長度約束：最小長度為 2。長度上限為 50。

模式：^([A-Za-z]_?)+$

必要：是

請求主體

請求沒有請求主體。

回應語法

```
HTTP/1.1 204
```

回應元素

如果動作成功，則服務會送回具有空 HTTP 主體的 HTTP 204 回應。

錯誤

BadRequestException

請求的格式不正確。例如，值無效或缺少必填欄位。請檢查欄位值，然後再試一次。

HTTP 狀態碼：400

ConflictException

處理要求時發生衝突。請再次嘗試您的請求。

HTTP 狀態碼：409

InternalFailureException

發生內部 Amazon Lex 錯誤。請再次嘗試您的請求。

HTTP 狀態碼：500

LimitExceededException

請求超過限制。請再次嘗試您的請求。

HTTP 狀態碼：429

NotFoundException

找不到要求中指定的資源。請檢查資源，然後再試一次。

HTTP 狀態碼：404

ResourceInUseException

您嘗試刪除的資源會由其他資源參照。使用此資訊可移除您嘗試刪除之資源的參照。

例外的主體包含描述資源的 JSON 物件。

```
{ "resourceType": BOT | BOTALIAS | BOTCHANNEL | INTENT,

"resourceReference": {

"name": string, "version": string } }
```

HTTP 狀態碼：400

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面
- AWS SDK for .NET
- AWS SDK for C++
- AWS 適用於轉到 V2 的 SDK
- AWS SDK for Java V2 的軟件
- AWS 適用於 JavaScript V3 的 SDK
- AWS SDK for PHP
- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

DeleteBotAlias

服務：Amazon Lex Model Building Service

刪除指定機器人的別名。

您無法刪除機器人與訊息通道之間關聯中使用的別名。如果在通道關聯中使用別名，則DeleteBot作業會傳回ResourceInUseException例外狀況，其中包含參照機器人之通道關聯的參照。您可以刪除通道關聯來移除別名的參照。如果您再次收到相同的例外狀況，請刪除反向連結，直到DeleteBotAlias作業成功為止。

請求語法

```
DELETE /bots/botName/aliases/name HTTP/1.1
```

URI 請求參數

請求會使用下列 URI 參數。

botName

別名所指向的機器人的名稱。

長度約束：最小長度為 2。長度上限為 50。

模式：^([A-Za-z]_?)+$

必要：是

name

要刪除的別名的名稱。名稱區分大小寫。

長度限制：長度下限為 1。長度上限為 100。

模式：^([A-Za-z]_?)+$

必要：是

請求主體

請求沒有請求主體。

回應語法

```
HTTP/1.1 204
```

回應元素

如果動作成功，則服務會送回具有空 HTTP 主體的 HTTP 204 回應。

錯誤

BadRequestException

請求的格式不正確。例如，值無效或缺少必填欄位。請檢查欄位值，然後再試一次。

HTTP 狀態碼：400

ConflictException

處理要求時發生衝突。請再次嘗試您的請求。

HTTP 狀態碼：409

InternalFailureException

發生內部 Amazon Lex 錯誤。請再次嘗試您的請求。

HTTP 狀態碼：500

LimitExceededException

請求超過限制。請再次嘗試您的請求。

HTTP 狀態碼：429

NotFoundException

找不到要求中指定的資源。請檢查資源，然後再試一次。

HTTP 狀態碼：404

ResourceInUseException

您嘗試刪除的資源會由其他資源參照。使用此資訊可移除您嘗試刪除之資源的參照。

例外的主體包含描述資源的 JSON 物件。

```
{ "resourceType": BOT | BOTALIAS | BOTCHANNEL | INTENT,
```

```
"resourceReference": {
"name": string, "version": string } }
```

HTTP 狀態碼：400

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面
- AWS SDK for .NET
- AWS SDK for C++
- AWS 適用於轉到 V2 的 SDK
- AWS SDK for Java V2 的軟件
- AWS 適用於 JavaScript V3 的 SDK
- AWS SDK for PHP
- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

DeleteBotChannelAssociation

服務：Amazon Lex Model Building Service

刪除 Amazon Lex 機器人與簡訊平台之間的關聯。

這項操作需要 `lex:DeleteBotChannelAssociation` 動作的許可。

請求語法

```
DELETE /bots/botName/aliases/aliasName/channels/name HTTP/1.1
```

URI 請求參數

請求會使用下列 URI 參數。

aliasName

指向正在建立此關聯之 Amazon Lex 機器人特定版本的別名。

長度限制：長度下限為 1。長度上限為 100。

模式：`^([A-Za-z]_?)+$`

必要：是

botName

Amazon Lex 機器人的名稱。

長度約束：最小長度為 2。長度上限為 50。

模式：`^([A-Za-z]_?)+$`

必要：是

name

關聯的名稱。名稱區分大小寫。

長度限制：長度下限為 1。長度上限為 100。

模式：`^([A-Za-z]_?)+$`

必要：是

請求主體

請求沒有請求主體。

回應語法

```
HTTP/1.1 204
```

回應元素

如果動作成功，則服務會送回具有空 HTTP 主體的 HTTP 204 回應。

錯誤

BadRequestException

請求的格式不正確。例如，值無效或缺少必填欄位。請檢查欄位值，然後再試一次。

HTTP 狀態碼：400

ConflictException

處理要求時發生衝突。請再次嘗試您的請求。

HTTP 狀態碼：409

InternalFailureException

發生內部 Amazon Lex 錯誤。請再次嘗試您的請求。

HTTP 狀態碼：500

LimitExceededException

請求超過限制。請再次嘗試您的請求。

HTTP 狀態碼：429

NotFoundException

找不到要求中指定的資源。請檢查資源，然後再試一次。

HTTP 狀態碼：404

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面
- AWS SDK for .NET
- AWS SDK for C++
- AWS 適用於轉到 V2 的 SDK
- AWS SDK for Java V2 的軟件
- AWS 適用於 JavaScript V3 的 SDK
- AWS SDK for PHP
- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

DeleteBotVersion

服務：Amazon Lex Model Building Service

刪除機器人的特定版本。要刪除機器人的所有版本，請使用該DeleteBot操作。

這項操作需要 `lex:DeleteBotVersion` 動作的許可。

請求語法

```
DELETE /bots/name/versions/version HTTP/1.1
```

URI 請求參數

請求會使用下列 URI 參數。

name

機器人的名稱。

長度約束：最小長度為 2。長度上限為 50。

模式：`^([A-Za-z]_?)+$`

必要：是

version

要刪除的機器人版本。您無法刪除機器人的$LATEST版本。若要刪除$LATEST版本，請使用此DeleteBot作業。

長度限制：長度下限為 1。長度上限為 64。

模式：`[0-9]+`

必要：是

請求主體

請求沒有請求主體。

回應語法

```
HTTP/1.1 204
```

回應元素

如果動作成功，則服務會送回具有空 HTTP 主體的 HTTP 204 回應。

錯誤

BadRequestException

請求的格式不正確。例如，值無效或缺少必填欄位。請檢查欄位值，然後再試一次。

HTTP 狀態碼：400

ConflictException

處理要求時發生衝突。請再次嘗試您的請求。

HTTP 狀態碼：409

InternalFailureException

發生內部 Amazon Lex 錯誤。請再次嘗試您的請求。

HTTP 狀態碼：500

LimitExceededException

請求超過限制。請再次嘗試您的請求。

HTTP 狀態碼：429

NotFoundException

找不到要求中指定的資源。請檢查資源，然後再試一次。

HTTP 狀態碼：404

ResourceInUseException

您嘗試刪除的資源會由其他資源參照。使用此資訊可移除您嘗試刪除之資源的參照。

例外的主體包含描述資源的 JSON 物件。

```
{ "resourceType": BOT | BOTALIAS | BOTCHANNEL | INTENT,

"resourceReference": {

"name": string, "version": string } }
```

HTTP 狀態碼：400

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面
- AWS SDK for .NET
- AWS SDK for C++
- AWS 適用於轉到 V2 的 SDK
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 JavaScript V3 的 SDK
- AWS SDK for PHP
- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

DeleteIntent

服務：Amazon Lex Model Building Service

刪除意圖的所有版本，包括$LATEST版本。若要刪除意圖的特定版本，請使用此DeleteIntentVersion作業。

只有在未參照意圖的情況下，您才能刪除意圖版本。若要刪除在一或多個機器人中參照的意圖 (請參閱Amazon Lex 運作方式)，您必須先移除這些參照。

> **Note**
>
> 如果你得到ResourceInUseException異常，它提供了一個示例參考，顯示意圖被引用的位置。若要移除對意圖的參照，請更新機器人或將其刪除。如果您嘗試再次刪除意圖時出現相同的例外狀況，請重複直到意圖沒有參照且呼叫成功DeleteIntent為止。

這項操作需要 lex:DeleteIntent 動作的許可。

請求語法

```
DELETE /intents/name HTTP/1.1
```

URI 請求參數

請求會使用下列 URI 參數。

name

意圖的名稱。名稱區分大小寫。

長度限制：長度下限為 1。長度上限為 100。

模式：^([A-Za-z]_?)+$

必要：是

請求主體

請求沒有請求主體。

回應語法

```
HTTP/1.1 204
```

回應元素

如果動作成功，則服務會送回具有空 HTTP 主體的 HTTP 204 回應。

錯誤

BadRequestException

請求的格式不正確。例如，值無效或缺少必填欄位。請檢查欄位值，然後再試一次。

HTTP 狀態碼：400

ConflictException

處理要求時發生衝突。請再次嘗試您的請求。

HTTP 狀態碼：409

InternalFailureException

發生內部 Amazon Lex 錯誤。請再次嘗試您的請求。

HTTP 狀態碼：500

LimitExceededException

請求超過限制。請再次嘗試您的請求。

HTTP 狀態碼：429

NotFoundException

找不到要求中指定的資源。請檢查資源，然後再試一次。

HTTP 狀態碼：404

ResourceInUseException

您嘗試刪除的資源會由其他資源參照。使用此資訊可移除您嘗試刪除之資源的參照。

例外的主體包含描述資源的 JSON 物件。

```
{ "resourceType": BOT | BOTALIAS | BOTCHANNEL | INTENT,
```

```
"resourceReference": {
"name": string, "version": string } }
```

HTTP 狀態碼：400

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面
- AWS SDK for .NET
- AWS SDK for C++
- AWS 適用於轉到 V2 的 SDK
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 JavaScript V3 的 SDK
- AWS SDK for PHP
- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

DeleteIntentVersion

服務：Amazon Lex Model Building Service

刪除特定版本的意圖。若要刪除意圖的所有版本，請使用此DeleteIntent作業。

這項操作需要 `lex:DeleteIntentVersion` 動作的許可。

請求語法

```
DELETE /intents/name/versions/version HTTP/1.1
```

URI 請求參數

請求會使用下列 URI 參數。

name

意圖的名稱。

長度限制：長度下限為 1。長度上限為 100。

模式：`^([A-Za-z]_?)+$`

必要：是

version

要刪除之意圖的版本。您無法刪除意圖的$LATEST版本。若要刪除$LATEST版本，請使用此DeleteIntent作業。

長度限制：長度下限為 1。長度上限為 64。

模式：`[0-9]+`

必要：是

請求主體

請求沒有請求主體。

回應語法

```
HTTP/1.1 204
```

回應元素

如果動作成功，則服務會送回具有空 HTTP 主體的 HTTP 204 回應。

錯誤

BadRequestException

請求的格式不正確。例如，值無效或缺少必填欄位。請檢查欄位值，然後再試一次。

HTTP 狀態碼：400

ConflictException

處理要求時發生衝突。請再次嘗試您的請求。

HTTP 狀態碼：409

InternalFailureException

發生內部 Amazon Lex 錯誤。請再次嘗試您的請求。

HTTP 狀態碼：500

LimitExceededException

請求超過限制。請再次嘗試您的請求。

HTTP 狀態碼：429

NotFoundException

找不到要求中指定的資源。請檢查資源，然後再試一次。

HTTP 狀態碼：404

ResourceInUseException

您嘗試刪除的資源會由其他資源參照。使用此資訊可移除您嘗試刪除之資源的參照。

例外的主體包含描述資源的 JSON 物件。

```
{ "resourceType": BOT | BOTALIAS | BOTCHANNEL | INTENT,

"resourceReference": {

"name": string, "version": string } }
```

HTTP 狀態碼：400

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面
- AWS SDK for .NET
- AWS SDK for C++
- AWS 適用於轉到 V2 的 SDK
- AWS SDK for Java V2 的軟件
- AWS 適用於 JavaScript V3 的 SDK
- AWS SDK for PHP
- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

DeleteSlotType

服務：Amazon Lex Model Building Service

刪除插槽類型的所有版本，包括$LATEST版本。若要刪除插槽類型的特定版本，請使用此DeleteSlotTypeVersion作業。

只有在沒有參考插槽類型的情況下，才能刪除該版本。若要刪除一或多個意圖中參照的槽類型，您必須先移除這些參照。

> **Note**
>
> 如果出現例ResourceInUseException外狀況，例外狀況會提供範例參照，其中顯示參考插槽類型的意圖。若要移除槽類型的參照，請更新意圖或將其刪除。如果您嘗試再次刪除插槽類型時出現相同的例外狀況，請重複此步驟，直到槽類型沒有參照並且DeleteSlotType呼叫成功為止。

這項操作需要 `lex:DeleteSlotType` 動作的許可。

請求語法

```
DELETE /slottypes/name HTTP/1.1
```

URI 請求參數

請求會使用下列 URI 參數。

name

位置類型的名稱。名稱區分大小寫。

長度限制：長度下限為 1。長度上限為 100。

模式：^([A-Za-z]_?)+$

必要：是

請求主體

請求沒有請求主體。

回應語法

```
HTTP/1.1 204
```

回應元素

如果動作成功，則服務會送回具有空 HTTP 主體的 HTTP 204 回應。

錯誤

BadRequestException

請求的格式不正確。例如，值無效或缺少必填欄位。請檢查欄位值，然後再試一次。

HTTP 狀態碼：400

ConflictException

處理要求時發生衝突。請再次嘗試您的請求。

HTTP 狀態碼：409

InternalFailureException

發生內部 Amazon Lex 錯誤。請再次嘗試您的請求。

HTTP 狀態碼：500

LimitExceededException

請求超過限制。請再次嘗試您的請求。

HTTP 狀態碼：429

NotFoundException

找不到要求中指定的資源。請檢查資源，然後再試一次。

HTTP 狀態碼：404

ResourceInUseException

您嘗試刪除的資源會由其他資源參照。使用此資訊可移除您嘗試刪除之資源的參照。

例外的主體包含描述資源的 JSON 物件。

```
{ "resourceType": BOT | BOTALIAS | BOTCHANNEL | INTENT,
```

```
"resourceReference": {

"name": string, "version": string } }
```

HTTP 狀態碼：400

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面
- AWS SDK for .NET
- AWS SDK for C++
- AWS 適用於轉到 V2 的 SDK
- AWS SDK for Java V2 的軟件
- AWS 適用於 JavaScript V3 的 SDK
- AWS SDK for PHP
- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

DeleteSlotTypeVersion

服務：Amazon Lex Model Building Service

刪除特定版本的插槽類型。若要刪除插槽類型的所有版本，請使用此DeleteSlotType作業。

這項操作需要 `lex:DeleteSlotTypeVersion` 動作的許可。

請求語法

```
DELETE /slottypes/name/version/version HTTP/1.1
```

URI 請求參數

請求會使用下列 URI 參數。

name

位置類型的名稱。

長度限制：長度下限為 1。長度上限為 100。

模式：`^([A-Za-z]_?)+$`

必要：是

version

要刪除之插槽類型的版本。您無法刪除插槽類型的$LATEST版本。若要刪除$LATEST版本，請使用此DeleteSlotType作業。

長度限制：長度下限為 1。長度上限為 64。

模式：`[0-9]+`

必要：是

請求主體

請求沒有請求主體。

回應語法

```
HTTP/1.1 204
```

回應元素

如果動作成功，則服務會送回具有空 HTTP 主體的 HTTP 204 回應。

錯誤

BadRequestException

請求的格式不正確。例如，值無效或缺少必填欄位。請檢查欄位值，然後再試一次。

HTTP 狀態碼：400

ConflictException

處理要求時發生衝突。請再次嘗試您的請求。

HTTP 狀態碼：409

InternalFailureException

發生內部 Amazon Lex 錯誤。請再次嘗試您的請求。

HTTP 狀態碼：500

LimitExceededException

請求超過限制。請再次嘗試您的請求。

HTTP 狀態碼：429

NotFoundException

找不到要求中指定的資源。請檢查資源，然後再試一次。

HTTP 狀態碼：404

ResourceInUseException

您嘗試刪除的資源會由其他資源參照。使用此資訊可移除您嘗試刪除之資源的參照。

例外的主體包含描述資源的 JSON 物件。

```
{ "resourceType": BOT | BOTALIAS | BOTCHANNEL | INTENT,

"resourceReference": {

"name": string, "version": string } }
```

HTTP 狀態碼：400

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面
- AWS SDK for .NET
- AWS SDK for C++
- AWS 適用於轉到 V2 的 SDK
- AWS SDK for Java V2 的軟件
- AWS 適用於 JavaScript V3 的 SDK
- AWS SDK for PHP
- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

DeleteUtterances

服務：Amazon Lex Model Building Service

刪除儲存的話語。

Amazon Lex 會儲存使用者傳送至您的機器人的語音內容。話語會儲存 15 天以供GetUtterancesView作業使用，然後無限期儲存，以便提升機器人回應使用者輸入的能力。

使用此DeleteUtterances作業可手動刪除特定使用者的已儲存語音。當您使用該DeleteUtterances操作時，為了提高機器人響應用戶輸入的能力而存儲的話語將立即刪除。儲存以配合GetUtterancesView作業使用的語音會在 15 天後刪除。

這項操作需要 `lex:DeleteUtterances` 動作的許可。

請求語法

```
DELETE /bots/botName/utterances/userId HTTP/1.1
```

URI 請求參數

請求會使用下列 URI 參數。

botName

存儲語音的機器人的名稱。

長度約束：最小長度為 2。長度上限為 50。

模式：`^([A-Za-z]_?)+$`

必要：是

userId

進行語音的使用者的唯一識別碼。這是在包含語音的PostContent或PostText作業要求中傳送的使用者識別碼。

長度約束：最小長度為 2。長度上限為 100。

必要：是

請求主體

請求沒有請求主體。

回應語法

```
HTTP/1.1 204
```

回應元素

如果動作成功，則服務會送回具有空 HTTP 主體的 HTTP 204 回應。

錯誤

BadRequestException

請求的格式不正確。例如，值無效或缺少必填欄位。請檢查欄位值，然後再試一次。

HTTP 狀態碼：400

InternalFailureException

發生內部 Amazon Lex 錯誤。請再次嘗試您的請求。

HTTP 狀態碼：500

LimitExceededException

請求超過限制。請再次嘗試您的請求。

HTTP 狀態碼：429

NotFoundException

找不到要求中指定的資源。請檢查資源，然後再試一次。

HTTP 狀態碼：404

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面
- AWS SDK for .NET
- AWS SDK for C++
- AWS 適用於轉到 V2 的 SDK
- AWS SDK for Java V2 的軟件

- AWS 適用於 JavaScript V3 的 SDK
- AWS SDK for PHP
- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

GetBot

服務：Amazon Lex Model Building Service

傳回特定機器人的中繼資料資訊。您必須提供機器人名稱以及機器人版本或別名。

這項操作需要 `lex:GetBot` 動作的許可。

請求語法

```
GET /bots/name/versions/versionoralias HTTP/1.1
```

URI 請求參數

請求會使用下列 URI 參數。

name

機器人的名稱。名稱區分大小寫。

長度約束：最小長度為 2。長度上限為 50。

模式：`^([A-Za-z]_?)+$`

必要：是

versionoralias

機器人的版本或別名。

必要：是

請求主體

請求沒有請求主體。

回應語法

```
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"
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

abortStatement

當使用者選擇結束對話而不完成對話時，Amazon Lex 傳回的訊息。如需詳細資訊，請參閱 PutBot。

類型：Statement 物件

checksum

用於識別機器人版本的特定修訂$LATEST版本的機器人的總和檢查碼。

類型：字串

childDirected

對於使用 Amazon Lex 模型建置服務建立的每個 Amazon Lex 機器人，您必須在childDirected欄位中指定或全部或部分針對 13 歲以下兒童並遵守《兒童線上隱私保護法》(COPPA) 的網站、程式或其他應用程式是否與全部或部分導向或鎖定目標的網站、程式true或其他應用程false式相關。透過**true**在**childDirected**欄位中指定，即表示您確認您對 Amazon Lex 的使用與針對 13 歲以下兒童且受 COPPA 規範的全部或部分針對網站、程式或其他應用程式有關。在childDirected欄位false中指定，即表示您確認您對 Amazon Lex 的使用與針對 13 歲以下兒童且受 COPPA 規範的全部或部分針對網站、程式或其他應用程式無關。您不得為欄位指定預設值，該childDirected欄位無法準確反映您對 Amazon Lex 的使用情況是否與針對 13 歲以下兒童的網站、程式或其他應用程式全部或部分針對 13 歲以下且受 COPPA 規限的網站、程式或其他應用程式有關。

如果您對 Amazon Lex 的使用涉及全部或部分針對 13 歲以下兒童的網站、程式或其他應用程式，您必須根據 COPPA 取得任何必要的可驗證父母同意。如需針對全部或部分針對 13 歲以下兒童的網站、程式或其他應用程式使用 Amazon Lex 的相關資訊，請參閱 Amazon Lex 常見問題集。

類型：布林值

clarificationPrompt

Amazon Lex 在不了解使用者要求時所使用的訊息。如需詳細資訊，請參閱 PutBot。

類型：Prompt 物件

createdDate

建立機器人的日期。

類型：Timestamp

description

機器人的描述。

類型：字串

長度限制：長度下限為 0。長度上限為 200。

detectSentiment

指出是否應將使用者語音傳送至 Amazon Comprehend 進行情緒分析。

類型：布林值

enableModelImprovements

指出機器人是否使用精確度改進。 `true`表示機器人正在使用改進，否則，`false`。

類型：布林值

failureReason

如果`status`是的話`FAILED`，Amazon Lex 會解釋為什麼無法建置機器人。

類型：字串

idleSessionTTLInSeconds

Amazon Lex 保留交談中收集之資料的時間上限 (以秒為單位)。如需詳細資訊，請參閱 PutBot。

類型：整數

有效範圍：最小值為 60。最大值為 86400。

intents

`intent` 物件的陣列。如需詳細資訊，請參閱 PutBot。

類型：Intent 物件陣列

lastUpdatedDate

機器人更新的日期。建立資源時，建立日期和上次更新日期相同。

類型：Timestamp

locale

機器人的目標地區設定。

類型：字串

有效值: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

機器人的名稱。

類型：字串

長度約束：最小長度為 2。長度上限為 50。

模式：^([A-Za-z]_?)+$

nluIntentConfidenceThreshold

決定 Amazon Lex 在或回PostText應中傳回替代意圖時插入AMAZON.FallbackIntentAMAZON.KendraSearchIntent、或兩者的位置的PostContent分數。 AMAZON.FallbackIntent如果所有意圖的可信度分數低於此值，則會插入。 AMAZON.KendraSearchIntent只有在為機器人配置時才會插入。

類型：Double

有效範圍：最小值為 0。最大值為 1。

status

機器人的狀態。

當狀態為 BUILDING Amazon Lex 正在建置機器人以供測試和使用時。

如果機器人的狀態是READY_BASIC_TESTING，您可以使用機器人意圖中指定的確切話語來測試機器人。當機器人已準備好進行完整測試或執行時，狀態為READY。

如果構建機器人時出現問題，則狀態為，FAILED並且該failureReason字段解釋了機器人未構建的原因。

如果機器人已儲存但未建置，則狀態為NOT_BUILT。

類型：字串

有效值:BUILDING | READY | READY_BASIC_TESTING | FAILED | NOT_BUILT

version

機器人的版本。對於新的機器人，版本始終是$LATEST。

類型：字串

長度限制：長度下限為 1。長度上限為 64。

模式：\$LATEST|[0-9]+

voiceId

亞馬遜 Lex 用於與用戶進行語音交互的 Amazon Polly 語音 ID。如需詳細資訊，請參閱 PutBot。

類型：字串

錯誤

BadRequestException

請求的格式不正確。例如，值無效或缺少必填欄位。請檢查欄位值，然後再試一次。

HTTP 狀態碼：400

InternalFailureException

發生內部 Amazon Lex 錯誤。請再次嘗試您的請求。

HTTP 狀態碼：500

LimitExceededException

請求超過限制。請再次嘗試您的請求。

HTTP 狀態碼：429

NotFoundException

找不到要求中指定的資源。請檢查資源，然後再試一次。

HTTP 狀態碼：404

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面
- AWS SDK for .NET
- AWS SDK for C++
- AWS 適用於轉到 V2 的 SDK
- AWS SDK for Java V2 的開發
- AWS 適用於 JavaScript V3 的 SDK
- AWS SDK for PHP
- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

GetBotAlias

服務：Amazon Lex Model Building Service

傳回 Amazon Lex 機器人別名的相關資訊。如需關於別名的詳細資訊，請參閱版本控制與別名。

這項操作需要 `lex:GetBotAlias` 動作的許可。

請求語法

```
GET /bots/botName/aliases/name HTTP/1.1
```

URI 請求參數

請求會使用下列 URI 參數。

botName

機器人的名稱。

長度約束：最小長度為 2。長度上限為 50。

模式：`^([A-Za-z]_?)+$`

必要：是

name

機器人別名的名稱。名稱區分大小寫。

長度限制：長度下限為 1。長度上限為 100。

模式：`^([A-Za-z]_?)+$`

必要：是

請求主體

請求沒有請求主體。

回應語法

```
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"
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

botName

別名所指向的機器人的名稱。

類型：字串

長度約束：最小長度為 2。長度上限為 50。

模式：^([A-Za-z]_?)+$

botVersion

別名指向的機器人版本。

類型：字串

長度限制：長度下限為 1。長度上限為 64。

模式：\$LATEST|[0-9]+

checksum

機器人別名的總和檢查碼。

類型：字串

conversationLogs

決定 Amazon Lex 如何將交談日誌用於別名的設定。

類型：ConversationLogsResponse 物件

createdDate

建立機器人別名的日期。

類型：Timestamp

description

機器人別名的說明。

類型：字串

長度限制：長度下限為 0。長度上限為 200。

lastUpdatedDate

更新機器人別名的日期。建立資源時，建立日期和上次更新日期相同。

類型：Timestamp

name

機器人別名的名稱。

類型：字串

長度限制：長度下限為 1。長度上限為 100。

模式：^([A-Za-z]_?)+$

錯誤

BadRequestException

請求的格式不正確。例如，值無效或缺少必填欄位。請檢查欄位值，然後再試一次。

HTTP 狀態碼：400

InternalFailureException

發生內部 Amazon Lex 錯誤。請再次嘗試您的請求。

HTTP 狀態碼：500

LimitExceededException

請求超過限制。請再次嘗試您的請求。

HTTP 狀態碼：429

NotFoundException

找不到要求中指定的資源。請檢查資源，然後再試一次。

HTTP 狀態碼：404

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面
- AWS SDK for .NET
- AWS SDK for C++
- AWS 適用於轉到 V2 的 SDK
- AWS SDK for Java V2 的軟件
- AWS 適用於 JavaScript V3 的 SDK
- AWS SDK for PHP
- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

GetBotAliases

服務：Amazon Lex Model Building Service

傳回指定 Amazon Lex 機器人的別名清單。

這項操作需要 `lex:GetBotAliases` 動作的許可。

請求語法

```
GET /bots/botName/aliases/?
maxResults=maxResults&nameContains=nameContains&nextToken=nextToken HTTP/1.1
```

URI 請求參數

請求會使用下列 URI 參數。

[botName](#)

機器人的名稱。

長度約束：最小長度為 2。長度上限為 50。

模式：`^([A-Za-z]_?)+$`

必要：是

[maxResults](#)

要在回應中傳回的最大別名數目。預設值為 50。

有效範圍：最小值為 1。最大值為 50。

[nameContains](#)

要在機器人別名中符合的子字串。如果其名稱的任何部分匹配的子字符串的別名將被返回。例如，「xyz」符合「XYZ」和「阿布扎克」兩者。

長度限制：長度下限為 1。長度上限為 100。

模式：`^([A-Za-z]_?)+$`

[nextToken](#)

用於獲取別名下一頁的分頁令牌。如果截斷對此呼叫的回應，Amazon Lex 會在回應中傳回分頁權杖。要獲取別名的下一頁，請在下一個請求中指定分頁令牌。

請求主體

請求沒有請求主體。

回應語法

```
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"
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

BotAliases

物件陣列，每個BotAliasMetadata物件都描述一個機器人別名。

類型：BotAliasMetadata 物件陣列

nextToken

用於獲取別名下一頁的分頁令牌。如果截斷對此呼叫的回應，Amazon Lex 會在回應中傳回分頁權杖。要獲取別名的下一頁，請在下一個請求中指定分頁令牌。

類型：字串

錯誤

BadRequestException

請求的格式不正確。例如，值無效或缺少必填欄位。請檢查欄位值，然後再試一次。

HTTP 狀態碼：400

InternalFailureException

發生內部 Amazon Lex 錯誤。請再次嘗試您的請求。

HTTP 狀態碼：500

LimitExceededException

請求超過限制。請再次嘗試您的請求。

HTTP 狀態碼：429

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面
- AWS SDK for .NET
- AWS SDK for C++
- AWS 適用於轉到 V2 的 SDK
- AWS SDK for Java V2 的軟件

- AWS 適用於 JavaScript V3 的 SDK
- AWS SDK for PHP
- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

GetBotChannelAssociation

服務：Amazon Lex Model Building Service

傳回 Amazon Lex 機器人與簡訊平台之間關聯的相關資訊。

這項操作需要 `lex:GetBotChannelAssociation` 動作的許可。

請求語法

```
GET /bots/botName/aliases/aliasName/channels/name HTTP/1.1
```

URI 請求參數

請求會使用下列 URI 參數。

aliasName

指向建立此關聯之特定 Amazon Lex 機器人版本的別名。

長度限制：長度下限為 1。長度上限為 100。

模式：`^([A-Za-z]_?)+$`

必要：是

botName

Amazon Lex 機器人的名稱。

長度約束：最小長度為 2。長度上限為 50。

模式：`^([A-Za-z]_?)+$`

必要：是

name

機器人與通道之間的關聯名稱。名稱區分大小寫。

長度限制：長度下限為 1。長度上限為 100。

模式：`^([A-Za-z]_?)+$`

必要：是

請求主體

請求沒有請求主體。

回應語法

```
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"
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

botAlias

指向建立此關聯之特定 Amazon Lex 機器人版本的別名。

類型：字串

長度限制：長度下限為 1。長度上限為 100。

模式：^([A-Za-z]_?)+$

botConfiguration

提供簡訊平台與 Amazon Lex 機器人通訊所需的資訊。

類型：字串到字串映射

地圖項目：最多 10 個項目。

botName

Amazon Lex 機器人的名稱。

類型：字串

長度約束：最小長度為 2。長度上限為 50。

模式：^([A-Za-z]_?)+$

createdDate

建立機器人與通道之間關聯的日期。

類型：Timestamp

description

機器人與通道之間關聯的說明。

類型：字串

長度限制：長度下限為 0。長度上限為 200。

failureReason

如果status是FAILED，Amazon Lex 會提供無法建立關聯的原因。

類型：字串

name

機器人與通道之間的關聯名稱。

類型：字串

長度限制：長度下限為 1。長度上限為 100。

模式：^([A-Za-z]_?)+$

status

機器人通道的狀態。

- CREATED-該頻道已創建並準備使用。
- IN_PROGRESS-頻道創建正在進行中。

- FAILED-建立頻道時發生錯誤。如需失敗原因的相關資訊，請參閱failureReason欄位。

類型：字串

有效值:IN_PROGRESS | CREATED | FAILED

type

訊息平台的類型。

類型：字串

有效值:Facebook | Slack | Twilio-Sms | Kik

錯誤

BadRequestException

請求的格式不正確。例如，值無效或缺少必填欄位。請檢查欄位值，然後再試一次。

HTTP 狀態碼：400

InternalFailureException

發生內部 Amazon Lex 錯誤。請再次嘗試您的請求。

HTTP 狀態碼：500

LimitExceededException

請求超過限制。請再次嘗試您的請求。

HTTP 狀態碼：429

NotFoundException

找不到要求中指定的資源。請檢查資源，然後再試一次。

HTTP 狀態碼：404

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面

- AWS SDK for .NET
- AWS SDK for C++
- AWS 適用於轉到 V2 的 SDK
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 JavaScript V3 的 SDK
- AWS SDK for PHP
- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

GetBotChannelAssociations

服務：Amazon Lex Model Building Service

傳回與指定機器人相關聯的所有通道清單。

作GetBotChannelAssociations業需要lex:GetBotChannelAssociations動作的權限。

請求語法

```
GET /bots/botName/aliases/aliasName/channels/?
maxResults=maxResults&nameContains=nameContains&nextToken=nextToken HTTP/1.1
```

URI 請求參數

請求會使用下列 URI 參數。

aliasName

指向正在建立此關聯之特定 Amazon Lex 機器人版本的別名。

長度限制：長度下限為 1。長度上限為 100。

模式：^(-|^([A-Za-z]_?)+$)$

必要：是

botName

關聯中 Amazon Lex 機器人的名稱。

長度約束：最小長度為 2。長度上限為 50。

模式：^([A-Za-z]_?)+$

必要：是

maxResults

要在回應中傳回的關聯數目上限。預設值為 50。

有效範圍：最小值為 1。最大值為 50。

nameContains

要在通道關聯名稱中匹配的子字符串。如果其名稱的任何部分與子字符串匹配，則將返回關聯。例如，「xyz」符合「XYZ」和「阿布扎克」兩者。若要傳回所有機器人通道關聯，請使用連字號 (「-」) 作為nameContains參數。

長度限制：長度下限為 1。長度上限為 100。

模式：^([A-Za-z]_?)+$

nextToken

用於獲取關聯下一頁的分頁令牌。如果截斷對此呼叫的回應，Amazon Lex 會在回應中傳回分頁權杖。要獲取關聯的下一頁，請在下一個請求中指定分頁令牌。

請求主體

請求沒有請求主體。

回應語法

```
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"
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

botChannelAssociations

物件陣列 (每個關聯各一個)，提供 Amazon Lex 機器人及其與通道關聯的相關資訊。

類型：BotChannelAssociation 物件陣列

nextToken

一個分頁令牌，用於獲取關聯的下一頁。如果截斷對此呼叫的回應，Amazon Lex 會在回應中傳回分頁權杖。要獲取關聯的下一頁，請在下一個請求中指定分頁令牌。

類型：字串

錯誤

BadRequestException

請求的格式不正確。例如，值無效或缺少必填欄位。請檢查欄位值，然後再試一次。

HTTP 狀態碼：400

InternalFailureException

發生內部 Amazon Lex 錯誤。請再次嘗試您的請求。

HTTP 狀態碼：500

LimitExceededException

請求超過限制。請再次嘗試您的請求。

HTTP 狀態碼：429

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面

- AWS SDK for .NET
- AWS SDK for C++
- AWS 適用於轉到 V2 的 SDK
- AWS SDK for Java V2 的軟件
- AWS 適用於 JavaScript V3 的 SDK
- AWS SDK for PHP
- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

GetBots

服務：Amazon Lex Model Building Service

傳回機器人資訊，如下所示：

- 如果您提供nameContains欄位，回應會包含名稱包含指定字串之所有機器人$LATEST版本的資訊。
- 如果您未指定nameContains欄位，作業會傳回所有機器人$LATEST版本的相關資訊。

這項操作需要 `lex:GetBots` 動作的許可。

請求語法

```
GET /bots/?maxResults=maxResults&nameContains=nameContains&nextToken=nextToken HTTP/1.1
```

URI 請求參數

請求會使用下列 URI 參數。

maxResults

要求傳回的回應中要傳回的最大機器人數目。預設為 10。

有效範圍：最小值為 1。最大值為 50。

nameContains

要在機器人名稱中符合的子字串。如果機器人名稱的任何部分與子字串相符，則會傳回機器人。例如，「xyz」符合「XYZ」和「阿布扎克」兩者。

長度約束：最小長度為 2。長度上限為 50。

模式：`^([A-Za-z]_?)+$`

nextToken

一個分頁令牌，用於獲取機器人的下一頁。如果截斷對此呼叫的回應，Amazon Lex 會在回應中傳回分頁權杖。要獲取機器人的下一頁，請在下一個請求中指定分頁令牌。

請求主體

請求沒有請求主體。

回應語法

```
HTTP/1.1 200
Content-type: application/json

{
   "bots": [
      {
         "createdDate": number,
         "description": "string",
         "lastUpdatedDate": number,
         "name": "string",
         "status": "string",
         "version": "string"
      }
   ],
   "nextToken": "string"
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

bots

botMetadata物件陣列，每個機器人都有一個項目。

類型：BotMetadata 物件陣列

nextToken

如果響應被截斷，則它包含一個分頁令牌，您可以在下一個請求中指定該標記以獲取機器人的下一頁。

類型：字串

錯誤

BadRequestException

請求的格式不正確。例如，值無效或缺少必填欄位。請檢查欄位值，然後再試一次。

HTTP 狀態碼：400

InternalFailureException

發生內部 Amazon Lex 錯誤。請再次嘗試您的請求。

HTTP 狀態碼：500

LimitExceededException

請求超過限制。請再次嘗試您的請求。

HTTP 狀態碼：429

NotFoundException

找不到要求中指定的資源。請檢查資源，然後再試一次。

HTTP 狀態碼：404

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面
- AWS SDK for .NET
- AWS SDK for C++
- AWS 適用於轉到 V2 的 SDK
- AWS SDK for Java V2 的开发
- AWS 適用於 JavaScript V3 的 SDK
- AWS SDK for PHP
- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

GetBotVersions

服務：Amazon Lex Model Building Service

取得有關機器人所有版本的資訊。

該GetBotVersions操作會為每個機器人版本返回一個BotMetadata對象。例如，如果一個機器人有三個編號的版本，則GetBotVersions作業會在回應中傳回四個BotMetadata物件，每個編號版本各傳回一個物件，另一個用於$LATEST版本。

該GetBotVersions操作始終返回至少一個版本，即$LATEST版本。

這項操作需要 `lex:GetBotVersions` 動作的許可。

請求語法

```
GET /bots/name/versions/?maxResults=maxResults&nextToken=nextToken HTTP/1.1
```

URI 請求參數

請求會使用下列 URI 參數。

maxResults

回應中要傳回的 Bot 版本數目上限。預設為 10。

有效範圍：最小值為 1。最大值為 50。

name

應傳回版本的機器人的名稱。

長度約束：最小長度為 2。長度上限為 50。

模式：`^([A-Za-z]_?)+$`

必要：是

nextToken

用於獲取機器人版本下一頁的分頁令牌。如果截斷對此呼叫的回應，Amazon Lex 會在回應中傳回分頁權杖。要獲取版本的下一頁，請在下一個請求中指定分頁令牌。

請求主體

請求沒有請求主體。

回應語法

```
HTTP/1.1 200
Content-type: application/json

{
   "bots": [
      {
         "createdDate": number,
         "description": "string",
         "lastUpdatedDate": number,
         "name": "string",
         "status": "string",
         "version": "string"
      }
   ],
   "nextToken": "string"
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

bots

BotMetadata物件陣列，機器人的每個編號版本各一個，另外一個用於該$LATEST版本。

類型：BotMetadata 物件陣列

nextToken

用於獲取機器人版本下一頁的分頁令牌。如果截斷對此呼叫的回應，Amazon Lex 會在回應中傳回分頁權杖。要獲取版本的下一頁，請在下一個請求中指定分頁令牌。

類型：字串

錯誤

BadRequestException

請求的格式不正確。例如，值無效或缺少必填欄位。請檢查欄位值，然後再試一次。

HTTP 狀態碼：400

InternalFailureException

發生內部 Amazon Lex 錯誤。請再次嘗試您的請求。

HTTP 狀態碼：500

LimitExceededException

請求超過限制。請再次嘗試您的請求。

HTTP 狀態碼：429

NotFoundException

找不到要求中指定的資源。請檢查資源，然後再試一次。

HTTP 狀態碼：404

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面
- AWS SDK for .NET
- AWS SDK for C++
- AWS 適用於轉到 V2 的 SDK
- AWS SDK for Java V2 的軟件
- AWS 適用於 JavaScript V3 的 SDK
- AWS SDK for PHP
- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

GetBuiltinIntent

服務：Amazon Lex Model Building Service

傳回關於內建意圖的資訊。

這項操作需要 `lex:GetBuiltinIntent` 動作的許可。

請求語法

```
GET /builtins/intents/signature HTTP/1.1
```

URI 請求參數

請求會使用下列 URI 參數。

signature

內置意圖的唯一標識符。若要尋找意圖的簽名，請參閱 Alexa 技能套件中的標準內建意圖。

必要：是

請求主體

請求沒有請求主體。

回應語法

```
HTTP/1.1 200
Content-type: application/json

{
   "signature": "string",
   "slots": [
      {
         "name": "string"
      }
   ],
   "supportedLocales": [ "string" ]
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

signature

內置意圖的唯一標識符。

類型：字串

slots

BuiltinIntentSlot物件陣列，意圖中每個插槽類型都有一個項目。

類型：BuiltinIntentSlot 物件陣列

supportedLocales

意圖支援的語言環境清單。

類型：字串陣列

有效值: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

錯誤

BadRequestException

請求的格式不正確。例如，值無效或缺少必填欄位。請檢查欄位值，然後再試一次。

HTTP 狀態碼：400

InternalFailureException

發生內部 Amazon Lex 錯誤。請再次嘗試您的請求。

HTTP 狀態碼：500

LimitExceededException

請求超過限制。請再次嘗試您的請求。

HTTP 狀態碼：429

NotFoundException

找不到要求中指定的資源。請檢查資源，然後再試一次。

HTTP 狀態碼：404

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面
- AWS SDK for .NET
- AWS SDK for C++
- AWS 適用於轉到 V2 的 SDK
- AWS SDK for Java V2 的开发
- AWS 適用於 JavaScript V3 的 SDK
- AWS SDK for PHP
- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

GetBuiltinIntents

服務：Amazon Lex Model Building Service

取得符合指定條件的內建意圖。

這項操作需要 `lex:GetBuiltinIntents` 動作的許可。

請求語法

```
GET /builtins/intents/?
locale=locale&maxResults=maxResults&nextToken=nextToken&signatureContains=signatureContains
 HTTP/1.1
```

URI 請求參數

請求會使用下列 URI 參數。

locale

意圖支援的語言環境清單。

有效值:`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

回應中要傳回的意圖數目上限。預設為 10。

有效範圍：最小值為 1。最大值為 50。

nextToken

一個分頁令牌，用於獲取意圖的下一頁。如果截斷此 API 呼叫，Amazon Lex 會在回應中傳回分頁權杖。要獲取意圖的下一頁，請在下一個請求中使用分頁令牌。

signatureContains

要在內建意圖簽名中比對的子字串。如果其簽名的任何部分匹配的子字符串將被返回意圖。例如，「xyz」符合「XYZ」和「阿卡茲」。若要尋找意圖的簽名，請參閱 Alexa 技能套件中的標準內建意圖。

請求主體

請求沒有請求主體。

回應語法

```
HTTP/1.1 200
Content-type: application/json

{
   "intents": [
      {
         "signature": "string",
         "supportedLocales": [ "string" ]
      }
   ],
   "nextToken": "string"
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

intents

builtinIntentMetadata對象的數組，一個響應中的每個意圖。

類型：BuiltinIntentMetadata 物件陣列

nextToken

一個分頁令牌，用於獲取意圖的下一頁。如果對此 API 呼叫的回應遭到截斷，Amazon Lex 會在回應中傳回分頁權杖。要獲取意圖的下一頁，請在下一個請求中指定分頁令牌。

類型：字串

錯誤

BadRequestException

請求的格式不正確。例如，值無效或缺少必填欄位。請檢查欄位值，然後再試一次。

HTTP 狀態碼：400

InternalFailureException

發生內部 Amazon Lex 錯誤。請再次嘗試您的請求。

HTTP 狀態碼：500

LimitExceededException

請求超過限制。請再次嘗試您的請求。

HTTP 狀態碼：429

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面
- AWS SDK for .NET
- AWS SDK for C++
- AWS 適用於轉到 V2 的 SDK
- AWS SDK for Java V2 的軟件
- AWS 適用於 JavaScript V3 的 SDK
- AWS SDK for PHP
- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

GetBuiltinSlotTypes

服務：Amazon Lex Model Building Service

取得符合指定條件的內建插槽類型。

如需內建插槽類型的清單，請參閱 Alexa 技能套件中的插槽類型參考。

這項操作需要 `lex:GetBuiltInSlotTypes` 動作的許可。

請求語法

```
GET /builtins/slottypes/?
locale=locale&maxResults=maxResults&nextToken=nextToken&signatureContains=signatureContains
 HTTP/1.1
```

URI 請求參數

請求會使用下列 URI 參數。

locale

插槽類型支援的語言環境清單。

有效值:`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

要在回應中傳回的插槽類型的最大數目。預設為 10。

有效範圍：最小值為 1。最大值為 50。

nextToken

一個分頁令牌，用於獲取插槽類型的下一頁。如果對此 API 呼叫的回應遭到截斷，Amazon Lex 會在回應中傳回分頁權杖。要獲取插槽類型的下一頁，請在下一個請求中指定分頁令牌。

signatureContains

要在內建插槽類型簽章中相符的子字串。如果其簽名的任何部分匹配的子字符串的插槽類型將被返回。例如，「xyz」符合「XYZ」和「阿卡茲」。

請求主體

請求沒有請求主體。

回應語法

```
HTTP/1.1 200
Content-type: application/json

{
   "nextToken": "string",
   "slotTypes": [
      {
         "signature": "string",
         "supportedLocales": [ "string" ]
      }
   ]
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

nextToken

如果響應被截斷，則響應包括一個分頁令牌，您可以在下一個請求中使用它來獲取插槽類型的下一頁。

類型：字串

slotTypes

`BuiltInSlotTypeMetadata`對象的數組，每個插槽類型返回一個條目。

類型：BuiltinSlotTypeMetadata 物件陣列

錯誤

BadRequestException

請求的格式不正確。例如，值無效或缺少必填欄位。請檢查欄位值，然後再試一次。

HTTP 狀態碼：400

InternalFailureException

發生內部 Amazon Lex 錯誤。請再次嘗試您的請求。

HTTP 狀態碼：500

LimitExceededException

請求超過限制。請再次嘗試您的請求。

HTTP 狀態碼：429

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面
- AWS SDK for .NET
- AWS SDK for C++
- AWS 適用於轉到 V2 的 SDK
- AWS SDK for Java V2 的开发
- AWS 適用於 JavaScript V3 的 SDK
- AWS SDK for PHP
- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

GetExport

服務：Amazon Lex Model Building Service

以指定的格式匯出 Amazon Lex 資源的內容。

請求語法

```
GET /exports/?exportType=exportType&name=name&resourceType=resourceType&version=version
 HTTP/1.1
```

URI 請求參數

請求會使用下列 URI 參數。

exportType

匯出資料的格式。

有效值:ALEXA_SKILLS_KIT | LEX

必要：是

name

要匯出的機器人名稱。

長度限制：長度下限為 1。長度上限為 100。

模式：[a-zA-Z_]+

必要：是

resourceType

要匯出的資源類型。

有效值:BOT | INTENT | SLOT_TYPE

必要：是

version

要匯出的機器人版本。

長度限制：長度下限為 1。長度上限為 64。

模式：[0-9]+

必要：是

請求主體

請求沒有請求主體。

回應語法

```
HTTP/1.1 200
Content-type: application/json

{
   "exportStatus": "string",
   "exportType": "string",
   "failureReason": "string",
   "name": "string",
   "resourceType": "string",
   "url": "string",
   "version": "string"
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

exportStatus

匯出的狀態。

- IN_PROGRESS-匯出正在進行中。
- READY-匯出完成。
- FAILED-匯出無法完成。

類型：字串

有效值:IN_PROGRESS | READY | FAILED

exportType

匯出資料的格式。

類型：字串

有效值:ALEXA_SKILLS_KIT | LEX

failureReason

如果status是FAILED，Amazon Lex 會提供無法匯出資源的原因。

類型：字串

name

要匯出的機器人名稱。

類型：字串

長度限制：長度下限為 1。長度上限為 100。

模式：[a-zA-Z_]+

resourceType

匯出資源的類型。

類型：字串

有效值:BOT | INTENT | SLOT_TYPE

url

S3 預先簽署的 URL，提供匯出資源的位置。匯出的資源是 ZIP 封存檔，其中包含以 JSON 格式匯出的資源。歸檔的結構可能會改變。您的代碼不應該依賴歸檔結構。

類型：字串

version

要匯出的機器人版本。

類型：字串

長度限制：長度下限為 1。長度上限為 64。

模式：[0-9]+

錯誤

BadRequestException

請求的格式不正確。例如，值無效或缺少必填欄位。請檢查欄位值，然後再試一次。

HTTP 狀態碼：400

InternalFailureException

發生內部 Amazon Lex 錯誤。請再次嘗試您的請求。

HTTP 狀態碼：500

LimitExceededException

請求超過限制。請再次嘗試您的請求。

HTTP 狀態碼：429

NotFoundException

找不到要求中指定的資源。請檢查資源，然後再試一次。

HTTP 狀態碼：404

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面
- AWS SDK for .NET
- AWS SDK for C++
- AWS 適用於轉到 V2 的 SDK
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 JavaScript V3 的 SDK
- AWS SDK for PHP
- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

GetImport

服務：Amazon Lex Model Building Service

取得有關從作業開始之匯入工`StartImport`作的資訊。

請求語法

```
GET /imports/importId HTTP/1.1
```

URI 請求參數

請求會使用下列 URI 參數。

importId

要傳回之匯入工作資訊的識別碼。

必要：是

請求主體

請求沒有請求主體。

回應語法

```
HTTP/1.1 200
Content-type: application/json

{
   "createdDate": number,
   "failureReason": [ "string" ],
   "importId": "string",
   "importStatus": "string",
   "mergeStrategy": "string",
   "name": "string",
   "resourceType": "string"
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

createdDate

建立匯入工作的日期和時間的時間戳記。

類型：Timestamp

failureReason

字串，說明匯入工作無法完成的原因。

類型：字串陣列

importId

特定匯入工作的識別碼。

類型：字串

importStatus

匯入工作的狀態。如果狀態為FAILED，您可以從failureReason欄位中取得失敗的原因。

類型：字串

有效值:IN_PROGRESS | COMPLETE | FAILED

mergeStrategy

匯入檔案中的現有資源與資源發生衝突時所採取的動作。

類型：字串

有效值:OVERWRITE_LATEST | FAIL_ON_CONFLICT

name

指定給匯入工作的名稱。

類型：字串

長度限制：長度下限為 1。長度上限為 100。

模式：[a-zA-Z_]+

resourceType

匯入的資源類型。

類型：字串

有效值:BOT | INTENT | SLOT_TYPE

錯誤

BadRequestException

請求的格式不正確。例如，值無效或缺少必填欄位。請檢查欄位值，然後再試一次。

HTTP 狀態碼：400

InternalFailureException

發生內部 Amazon Lex 錯誤。請再次嘗試您的請求。

HTTP 狀態碼：500

LimitExceededException

請求超過限制。請再次嘗試您的請求。

HTTP 狀態碼：429

NotFoundException

找不到要求中指定的資源。請檢查資源，然後再試一次。

HTTP 狀態碼：404

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面
- AWS SDK for .NET
- AWS SDK for C++
- AWS 適用於轉到 V2 的 SDK
- AWS SDK for Java V2 的軟件
- AWS 適用於 JavaScript V3 的 SDK
- AWS SDK for PHP

- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

GetIntent

服務：Amazon Lex Model Building Service

返回有關意圖的信息。除了意圖名稱之外，您還必須指定意圖版本。

這項操作需要許可來執行 `lex:GetIntent` 動作。

請求語法

```
GET /intents/name/versions/version HTTP/1.1
```

URI 請求參數

請求會使用下列 URI 參數。

name

意圖的名稱。名稱區分大小寫。

長度限制：長度下限為 1。長度上限為 100。

模式：`^([A-Za-z]_?)+$`

必要：是

version

意圖的版本。

長度限制：長度下限為 1。長度上限為 64。

模式：`\$LATEST|[0-9]+`

必要：是

請求主體

請求沒有請求主體。

回應語法

```
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"
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

checksum

意圖的總和檢查碼。

類型：字串

conclusionStatement

在fulfillmentActivity元素中指定的 Lambda 函數完成意圖之後，Amazon Lex 就會將此陳述式傳達給使用者。

類型：Statement 物件

confirmationPrompt

如果在機器人中定義，Amazon Lex 會在完成使用者要求之前使用提示來確認意圖。如需詳細資訊，請參閱 PutIntent。

類型：Prompt 物件

createdDate

建立意圖的日期。

類型：Timestamp

description

意圖的描述。

類型：字串

長度限制：長度下限為 0。長度上限為 200。

dialogCodeHook

如果在機器人中定義，Amazon Amazon Lex 會針對每個使用者輸入叫用此 Lambda 函數。如需詳細資訊，請參閱 PutIntent。

類型：CodeHook 物件

followUpPrompt

如果在機器人中定義，Amazon Lex 會在完成意圖後使用此提示來請求其他使用者活動。如需詳細資訊，請參閱 PutIntent。

類型：FollowUpPrompt 物件

fulfillmentActivity

描述如何實現意圖。如需詳細資訊，請參閱 PutIntent。

類型：FulfillmentActivity 物件

inputContexts

InputContext物件陣列，列出 Amazon Lex 必須處於作用中狀態的內容，才能在與使用者交談中選擇意圖。

類型：InputContext 物件陣列

陣列成員：項目數下限為 0。項目數上限為 5。

kendraConfiguration

設定資訊 (如果有的話)，以便依據AMAZON.KendraSearchIntent意圖連接到 Amazon Kendra 索引。

類型：KendraConfiguration 物件

lastUpdatedDate

更新意圖的日期。建立資源時，建立日期和上次更新日期相同。

類型：Timestamp

name

意圖的名稱。

類型：字串

長度限制：長度下限為 1。長度上限為 100。

模式：^([A-Za-z]_?)+$

outputContexts

OutputContext物件陣列，列出實現意圖時意圖啟動的前後關聯。

類型：OutputContext 物件陣列

陣列成員：項目數下限為 0。項目數上限為 10。

parentIntentSignature

內建意圖的唯一識別碼。

類型：字串

rejectionStatement

如果使用者對中定義的問題回答「否」confirmationPrompt，Amazon Lex 會以此聲明回應，確認意圖已取消。

類型：Statement 物件

sampleUtterances

針對意圖設定的範例語音陣列。

類型：字串陣列

陣列成員：項目數下限為 0。最多可容納 1500 個物品數量。

長度限制：長度下限為 1。長度上限為 200。

slots

針對意圖配置的意圖槽陣列。

類型：Slot 物件陣列

陣列成員：項目數下限為 0。項目數上限為 100。

version

意圖的版本。

類型：字串

長度限制：長度下限為 1。長度上限為 64。

模式：\$LATEST|[0-9]+

錯誤

BadRequestException

請求的格式不正確。例如，值無效或缺少必填欄位。請檢查欄位值，然後再試一次。

HTTP 狀態碼：400

InternalFailureException

發生內部 Amazon Lex 錯誤。請再次嘗試您的請求。

HTTP 狀態碼：500

LimitExceededException

請求超過限制。請再次嘗試您的請求。

HTTP 狀態碼：429

NotFoundException

找不到要求中指定的資源。請檢查資源，然後再試一次。

HTTP 狀態碼：404

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面
- AWS SDK for .NET
- AWS SDK for C++
- AWS 適用於轉到 V2 的 SDK
- AWS SDK for Java V2 的軟件
- AWS 適用於 JavaScript V3 的 SDK
- AWS SDK for PHP
- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

GetIntents

服務：Amazon Lex Model Building Service

返回意圖信息，如下所示：

- 如果您指定nameContains欄位，會傳回包含指定字串之所有意圖的$LATEST版本。
- 如果您未指定nameContains欄位，會傳回所有意圖$LATEST版本的相關資訊。

該操作需要該操lex:GetIntents作的權限。

請求語法

```
GET /intents/?maxResults=maxResults&nameContains=nameContains&nextToken=nextToken
 HTTP/1.1
```

URI 請求參數

請求會使用下列 URI 參數。

maxResults

回應中要傳回的意圖數目上限。預設為 10。

有效範圍：最小值為 1。最大值為 50。

nameContains

要在意圖名稱中匹配的子字符串。如果其名稱的任何部分匹配的子字符串的意圖將被返回。例如，「xyz」符合「XYZ」和「阿卡茲」。

長度限制：長度下限為 1。長度上限為 100。

模式：^([A-Za-z]_?)+$

nextToken

一個分頁令牌，用於獲取意圖的下一頁。如果對此 API 呼叫的回應遭到截斷，Amazon Lex 會在回應中傳回分頁權杖。要獲取意圖的下一頁，請在下一個請求中指定分頁令牌。

請求主體

請求沒有請求主體。

回應語法

```
HTTP/1.1 200
Content-type: application/json

{
   "intents": [
      {
         "createdDate": number,
         "description": "string",
         "lastUpdatedDate": number,
         "name": "string",
         "version": "string"
      }
   ],
   "nextToken": "string"
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

intents

Intent 物件的陣列。如需詳細資訊，請參閱 PutBot。

類型：IntentMetadata 物件陣列

nextToken

如果回應遭到截斷，則回應會包含一個分頁 Token，您可以在下一個要求中指定，以擷取下一頁的意圖。

類型：字串

錯誤

BadRequestException

請求的格式不正確。例如，值無效或缺少必填欄位。請檢查欄位值，然後再試一次。

HTTP 狀態碼：400

InternalFailureException

發生內部 Amazon Lex 錯誤。請再次嘗試您的請求。

HTTP 狀態碼：500

LimitExceededException

請求超過限制。請再次嘗試您的請求。

HTTP 狀態碼：429

NotFoundException

找不到要求中指定的資源。請檢查資源，然後再試一次。

HTTP 狀態碼：404

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面
- AWS SDK for .NET
- AWS SDK for C++
- AWS 適用於轉到 V2 的 SDK
- AWS SDK for Java V2 的軟件
- AWS 適用於 JavaScript V3 的 SDK
- AWS SDK for PHP
- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

GetIntentVersions

服務：Amazon Lex Model Building Service

獲取有關意圖的所有版本的信息。

此GetIntentVersions作業會針IntentMetadata對每個意圖版本傳回物件。例如，如果意圖有三個編號版本，則GetIntentVersions作業會在回應中傳回四個IntentMetadata物件，每個物件會傳回一個編號版本，另一個用於$LATEST版本。

該GetIntentVersions操作始終返回至少一個版本，即$LATEST版本。

這項操作需要 `lex:GetIntentVersions` 動作的許可。

請求語法

```
GET /intents/name/versions/?maxResults=maxResults&nextToken=nextToken HTTP/1.1
```

URI 請求參數

請求會使用下列 URI 參數。

maxResults

回應中要傳回的意圖版本數目上限。預設為 10。

有效範圍：最小值為 1。最大值為 50。

name

應該返回哪些版本的意圖的名稱。

長度限制：長度下限為 1。長度上限為 100。

模式：`^([A-Za-z]_?)+$`

必要：是

nextToken

用於獲取意圖版本的下一頁的分頁令牌。如果截斷對此呼叫的回應，Amazon Lex 會在回應中傳回分頁權杖。要獲取版本的下一頁，請在下一個請求中指定分頁令牌。

請求主體

請求沒有請求主體。

回應語法

```
HTTP/1.1 200
Content-type: application/json

{
   "intents": [
      {
         "createdDate": number,
         "description": "string",
         "lastUpdatedDate": number,
         "name": "string",
         "version": "string"
      }
   ],
   "nextToken": "string"
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

intents

IntentMetadata物件陣列，意圖的每個編號版本各有一個，另外一個用於$LATEST版本。

類型：IntentMetadata 物件陣列

nextToken

用於獲取意圖版本的下一頁的分頁令牌。如果截斷對此呼叫的回應，Amazon Lex 會在回應中傳回分頁權杖。要獲取版本的下一頁，請在下一個請求中指定分頁令牌。

類型：字串

錯誤

BadRequestException

請求的格式不正確。例如，值無效或缺少必填欄位。請檢查欄位值，然後再試一次。

HTTP 狀態碼：400

InternalFailureException

發生內部 Amazon Lex 錯誤。請再次嘗試您的請求。

HTTP 狀態碼：500

LimitExceededException

請求超過限制。請再次嘗試您的請求。

HTTP 狀態碼：429

NotFoundException

找不到要求中指定的資源。請檢查資源，然後再試一次。

HTTP 狀態碼：404

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面
- AWS SDK for .NET
- AWS SDK for C++
- AWS 適用於轉到 V2 的 SDK
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 JavaScript V3 的 SDK
- AWS SDK for PHP
- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

GetMigration

服務：Amazon Lex Model Building Service

提供有關從 Amazon Lex V1 機器人進行中或完整遷移至 Amazon Lex V2 機器人的詳細資訊。使用此作業可檢視與移轉相關的移轉警示和警告。

請求語法

```
GET /migrations/migrationId HTTP/1.1
```

URI 請求參數

請求會使用下列 URI 參數。

migrationId

要檢視之移轉的唯一識別碼。由StartMigration作業傳回。migrationID

長度約束：固定長度為 10。

模式：^[0-9a-zA-Z]+$

必要：是

請求主體

請求沒有請求主體。

回應語法

```
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"
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

alerts

指出將 Amazon Lex V1 機器人移轉至 Amazon Lex V2 時發生問題的警示和警告清單。當 Amazon Lex V1 功能在 Amazon Lex V2 中具有不同的實作時，您會收到警告。

如需詳細資訊，請參閱 Amazon Lex V2 開發人員指南中的遷移機器人。

類型：MigrationAlert 物件陣列

migrationId

遷移的唯一識別碼。這與調用GetMigration操作時使用的標識符相同。

類型：字串

長度約束：固定長度為 10。

模式：^[0-9a-zA-Z]+$

migrationStatus

指出移轉的狀態。當狀態為COMPLETE移轉完成且機器人可在 Amazon Lex V2 中使用時。可能有警示和警告需要解決才能完成移轉。

類型：字串

有效值:IN_PROGRESS | COMPLETED | FAILED

migrationStrategy

用於執行移轉的策略。

- CREATE_NEW-建立新的 Amazon Lex V2 機器人，並將 Amazon Lex V1 機器人移轉至新的機器人。
- UPDATE_EXISTING-覆寫現有的 Amazon Lex V2 機器人中繼資料和要遷移的地區設定。它不會變更 Amazon Lex V2 機器人中的任何其他語言環境。如果語言環境不存在，則會在 Amazon Lex V2 機器人中建立新的地區設定。

類型：字串

有效值:CREATE_NEW | UPDATE_EXISTING

migrationTimestamp

移轉開始的日期和時間。

類型：Timestamp

v1BotLocale

Amazon Lex V1 機器人的地區設定已遷移到 Amazon Lex V2。

類型：字串

有效值: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

Amazon Lex V1 機器人的名稱遷移到 Amazon Lex V2。

類型：字串

長度約束：最小長度為 2。長度上限為 50。

模式：^([A-Za-z]_?)+$

v1BotVersion

Amazon Lex V1 機器人的版本遷移到 Amazon Lex V2。

類型：字串

長度限制：長度下限為 1。長度上限為 64。

模式：\$LATEST|[0-9]+

v2BotId

Amazon Lex V1 正在遷移到的亞 Amazon Lex V2 機器人的唯一識別碼。

類型：字串

長度約束：固定長度為 10。

模式：^[0-9a-zA-Z]+$

v2BotRole

亞馬遜萊克斯用來執行亞馬遜萊克 Amazon Lex V2 機器人的 IAM 角色。

類型：字串

長度限制：長度下限為 20。長度上限為 2048。

模式：^arn:[\w\-]+:iam::[\d]{12}:role/.+$

錯誤

BadRequestException

請求的格式不正確。例如，值無效或缺少必填欄位。請檢查欄位值，然後再試一次。

HTTP 狀態碼：400

InternalFailureException

發生內部 Amazon Lex 錯誤。請再次嘗試您的請求。

HTTP 狀態碼：500

LimitExceededException

請求超過限制。請再次嘗試您的請求。

HTTP 狀態碼：429

NotFoundException

找不到要求中指定的資源。請檢查資源，然後再試一次。

HTTP 狀態碼：404

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面
- AWS SDK for .NET
- AWS SDK for C++
- AWS 適用於轉到 V2 的 SDK
- AWS SDK for Java V2 的开发
- AWS 適用於 JavaScript V3 的 SDK
- AWS SDK for PHP
- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

GetMigrations

服務：Amazon Lex Model Building Service

取得亞馬遜 Lex V1 和亞馬遜萊克斯 V2 之間的遷移清 Amazon Lex。

請求語法

```
GET /migrations?
maxResults=maxResults&migrationStatusEquals=migrationStatusEquals&nextToken=nextToken&sortByAtt
 HTTP/1.1
```

URI 請求參數

請求會使用下列 URI 參數。

maxResults

遷移的最大數量在響應中返回。預設為 10。

有效範圍：最小值為 1。最大值為 50。

migrationStatusEquals

篩選清單以僅包含指定狀態下的移轉。

有效值:IN_PROGRESS | COMPLETED | FAILED

nextToken

一個分頁令牌，用於獲取遷移的下一頁。如果截斷對此作業的回應，Amazon Lex 會在回應中傳回分頁權杖。要獲取遷移的下一頁，請在請求中指定分頁令牌。

sortByAttribute

要排序移轉清單所依據的欄位。您可以依 Amazon Lex V1 機器人名稱或開始遷移的日期和時間來排序。

有效值:V1_BOT_NAME | MIGRATION_DATE_TIME

sortByOrder

順序排序列表。

有效值:ASCENDING | DESCENDING

v1BotNameContains

篩選清單，使其只包含名稱包含指定字串的機器人。該字符串匹配機器人名稱中的任何位置。

長度約束：最小長度為 2。長度上限為 50。

模式：^([A-Za-z]_?)+$

請求主體

請求沒有請求主體。

回應語法

```
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"
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

migrationSummaries

從亞馬遜 Lex V1 遷移到亞馬遜萊克斯 V2 的摘要陣列。若要查看遷移的詳細資料，請在呼叫GetMigration作業時使用摘要。migrationId

類型：MigrationSummary 物件陣列

nextToken

如果響應被截斷，它包括一個分頁令牌，您可以在下一個請求中指定以獲取遷移的下一頁。

類型：字串

錯誤

BadRequestException

請求的格式不正確。例如，值無效或缺少必填欄位。請檢查欄位值，然後再試一次。

HTTP 狀態碼：400

InternalFailureException

發生內部 Amazon Lex 錯誤。請再次嘗試您的請求。

HTTP 狀態碼：500

LimitExceededException

請求超過限制。請再次嘗試您的請求。

HTTP 狀態碼：429

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面
- AWS SDK for .NET
- AWS SDK for C++
- AWS 適用於轉到 V2 的 SDK
- AWS SDK for Java V2 的軟件

- AWS 適用於 JavaScript V3 的 SDK
- AWS SDK for PHP
- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

GetSlotType

服務：Amazon Lex Model Building Service

傳回特定版本的插槽類型的相關資訊。除了指定插槽類型名稱之外，您還必須指定插槽類型版本。

這項操作需要 `lex:GetSlotType` 動作的許可。

請求語法

```
GET /slottypes/name/versions/version HTTP/1.1
```

URI 請求參數

請求會使用下列 URI 參數。

name

位置類型的名稱。名稱區分大小寫。

長度限制：長度下限為 1。長度上限為 100。

模式：`^([A-Za-z]_?)+$`

必要：是

version

插槽類型的版本。

長度限制：長度下限為 1。長度上限為 64。

模式：`\$LATEST|[0-9]+`

必要：是

請求主體

請求沒有請求主體。

回應語法

```
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"
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

checksum

插槽類型$LATEST版本的總和檢查碼。

類型：字串

createdDate

插槽類型的建立日期。

類型：Timestamp

description

位置類型的描述。

類型：字串

長度限制：長度下限為 0。長度上限為 200。

enumerationValues

定義插槽類型可採用的值的EnumerationValue物件清單。

類型：EnumerationValue 物件陣列

陣列成員：項目數下限為 0。最多可包含 1 萬個項目。

lastUpdatedDate

更新插槽類型的日期。建立資源時，建立日期和上次更新日期相同。

類型：Timestamp

name

位置類型的名稱。

類型：字串

長度限制：長度下限為 1。長度上限為 100。

模式：^([A-Za-z]_?)+$

parentSlotTypeSignature

用作插槽類型的父插槽類型的內建插槽類型。

類型：字串

長度限制：長度下限為 1。長度上限為 100。

模式：^((AMAZON\.)_?|[A-Za-z]_?)+

slotTypeConfigurations

擴充上層內建插槽類型的組態資訊。

類型：SlotTypeConfiguration 物件陣列

陣列成員：項目數下限為 0。項目數上限為 10。

valueSelectionStrategy

Amazon Lex 用來判斷插槽價值的策略。如需詳細資訊，請參閱 PutSlotType。

類型：字串

有效值:ORIGINAL_VALUE | TOP_RESOLUTION

version

插槽類型的版本。

類型：字串

長度限制：長度下限為 1。長度上限為 64。

模式：\$LATEST|[0-9]+

錯誤

BadRequestException

請求的格式不正確。例如，值無效或缺少必填欄位。請檢查欄位值，然後再試一次。

HTTP 狀態碼：400

InternalFailureException

發生內部 Amazon Lex 錯誤。請再次嘗試您的請求。

HTTP 狀態碼：500

LimitExceededException

請求超過限制。請再次嘗試您的請求。

HTTP 狀態碼：429

NotFoundException

找不到要求中指定的資源。請檢查資源，然後再試一次。

HTTP 狀態碼：404

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面

- AWS SDK for .NET
- AWS SDK for C++
- AWS 適用於轉到 V2 的 SDK
- AWS SDK for Java V2 的軟件
- AWS 適用於 JavaScript V3 的 SDK
- AWS SDK for PHP
- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

GetSlotTypes

服務：Amazon Lex Model Building Service

返回槽類型的信息，如下所示：

- 如果您指定nameContains欄位，會傳回包含指定字串的所有插槽類型的$LATEST版本。
- 如果您未指定nameContains欄位，會傳回所有插槽類型$LATEST版本的相關資訊。

該操作需要該操lex:GetSlotTypes作的權限。

請求語法

```
GET /slottypes/?maxResults=maxResults&nameContains=nameContains&nextToken=nextToken
 HTTP/1.1
```

URI 請求參數

請求會使用下列 URI 參數。

maxResults

要在回應中傳回的插槽類型的最大數目。預設為 10。

有效範圍：最小值為 1。最大值為 50。

nameContains

要在插槽類型名稱中相符的子字串。如果其名稱的任何部分匹配的子字符串的槽類型將被返回。例如，「xyz」符合「XYZ」和「阿布扎克」兩者。

長度限制：長度下限為 1。長度上限為 100。

模式：^([A-Za-z]_?)+$

nextToken

一個分頁令牌，用於獲取插槽類型的下一頁。如果對此 API 呼叫的回應遭到截斷，Amazon Lex 會在回應中傳回分頁權杖。要獲取插槽類型的下一頁，請在下一個請求中指定分頁令牌。

請求主體

請求沒有請求主體。

回應語法

```
HTTP/1.1 200
Content-type: application/json

{
   "nextToken": "string",
   "slotTypes": [
      {
         "createdDate": number,
         "description": "string",
         "lastUpdatedDate": number,
         "name": "string",
         "version": "string"
      }
   ]
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

nextToken

如果響應被截斷，它包含一個分頁令牌，您可以在下一個請求中指定該標記以獲取插槽類型的下一頁。

類型：字串

slotTypes

物件陣列，每個插槽類型各一個，可提供插槽類型名稱、版本和說明等資訊。

類型：SlotTypeMetadata 物件陣列

錯誤

BadRequestException

請求的格式不正確。例如，值無效或缺少必填欄位。請檢查欄位值，然後再試一次。

HTTP 狀態碼：400

InternalFailureException

發生內部 Amazon Lex 錯誤。請再次嘗試您的請求。

HTTP 狀態碼：500

LimitExceededException

請求超過限制。請再次嘗試您的請求。

HTTP 狀態碼：429

NotFoundException

找不到要求中指定的資源。請檢查資源，然後再試一次。

HTTP 狀態碼：404

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面
- AWS SDK for .NET
- AWS SDK for C++
- AWS 適用於轉到 V2 的 SDK
- AWS SDK for Java V2 的軟件
- AWS 適用於 JavaScript V3 的 SDK
- AWS SDK for PHP
- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

GetSlotTypeVersions

服務：Amazon Lex Model Building Service

取得插槽類型之所有版本的相關資訊。

此GetSlotTypeVersions作業會針SlotTypeMetadata對插槽類型的每個版本傳回一個物件。例如，如果插槽類型有三個編號版本，則GetSlotTypeVersions作業會在回應中傳回四個SlotTypeMetadata物件，每個編號版本各傳回一個物件，另一個用於$LATEST版本。

該GetSlotTypeVersions操作始終返回至少一個版本，即$LATEST版本。

這項操作需要 `lex:GetSlotTypeVersions` 動作的許可。

請求語法

```
GET /slottypes/name/versions/?maxResults=maxResults&nextToken=nextToken HTTP/1.1
```

URI 請求參數

請求會使用下列 URI 參數。

maxResults

要在回應中傳回的插槽類型版本數目上限。預設為 10。

有效範圍：最小值為 1。最大值為 50。

name

應該返回哪些版本插槽類型的名稱。

長度限制：長度下限為 1。長度上限為 100。

模式：`^([A-Za-z]_?)+$`

必要：是

nextToken

用於獲取插槽類型版本的下一頁的分頁令牌。如果截斷對此呼叫的回應，Amazon Lex 會在回應中傳回分頁權杖。要獲取版本的下一頁，請在下一個請求中指定分頁令牌。

請求主體

請求沒有請求主體。

回應語法

```
HTTP/1.1 200
Content-type: application/json

{
   "nextToken": "string",
   "slotTypes": [
      {
         "createdDate": number,
         "description": "string",
         "lastUpdatedDate": number,
         "name": "string",
         "version": "string"
      }
   ]
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

nextToken

用於獲取插槽類型版本的下一頁的分頁令牌。如果截斷對此呼叫的回應，Amazon Lex 會在回應中傳回分頁權杖。要獲取版本的下一頁，請在下一個請求中指定分頁令牌。

類型：字串

slotTypes

SlotTypeMetadata物件陣列，插槽類型的每個編號版本各一個，另外一個用於$LATEST版本。

類型：SlotTypeMetadata 物件陣列

錯誤

BadRequestException

請求的格式不正確。例如，值無效或缺少必填欄位。請檢查欄位值，然後再試一次。

HTTP 狀態碼：400

InternalFailureException

發生內部 Amazon Lex 錯誤。請再次嘗試您的請求。

HTTP 狀態碼：500

LimitExceededException

請求超過限制。請再次嘗試您的請求。

HTTP 狀態碼：429

NotFoundException

找不到要求中指定的資源。請檢查資源，然後再試一次。

HTTP 狀態碼：404

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面
- AWS SDK for .NET
- AWS SDK for C++
- AWS 適用於轉到 V2 的 SDK
- AWS SDK for Java V2 的軟件
- AWS 適用於 JavaScript V3 的 SDK
- AWS SDK for PHP
- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

GetUtterancesView

服務：Amazon Lex Model Building Service

使用該GetUtterancesView操作可獲取有關用戶對您的機器人進行的語音的信息。您可以使用此列表來調整機器人響應的話語。

例如，假設您已經創建了一個機器人來訂購鮮花。使用者使用您的機器人一段時間後，請使用該GetUtterancesView操作來查看他們提出的請求以及他們是否成功。您可能會發現「我要花」的話語沒有被認可。您可以將此話語添加到OrderFlowers意圖中，以便您的機器人識別該話語。

發布新版本的機器人後，您可以獲取有關舊版本和新版本的信息，以便您可以比較兩個版本的性能。

表達用語統計資料為每天產生一次。過去 15 天的資料可用。您可以在每個請求中要求最多 5 個機器人版本的資訊。Amazon Lex 會傳回機器人在過去 15 天內收到的最常用語音。回應包含每個版本最多 100 個語音的相關資訊。

在下列情況下，不會產生話語統計資料：

- 建立機器人時，childDirected欄位設定為 true。
- 您正在使用一個或多個插槽的插槽混淆。
- 您選擇不參與改善亞 Amazon Lex。

這項操作需要 lex:GetUtterancesView 動作的許可。

請求語法

```
GET /bots/botname/utterances?
view=aggregation&bot_versions=botVersions&status_type=statusType HTTP/1.1
```

URI 請求參數

請求會使用下列 URI 參數。

botname

應傳回其語音資訊的機器人名稱。

長度約束：最小長度為 2。長度上限為 50。

模式：^([A-Za-z]_?)+$

必要：是

botVersions

應傳回語音資訊的機器人版本陣列。限制為每個請求 5 個版本。

陣列成員：項目數下限為 1。項目數上限為 5。

長度限制：長度下限為 1。長度上限為 64。

模式：\$LATEST|[0-9]+

必要：是

statusType

要返回已識別和處理的語音，請使用。Detected要返回未識別的語音，請使用。Missed

有效值:Detected | Missed

必要：是

請求主體

請求沒有請求主體。

回應語法

```
HTTP/1.1 200
Content-type: application/json

{
   "botName": "string",
   "utterances": [
      {
         "botVersion": "string",
         "utterances": [
            {
               "count": number,
               "distinctUsers": number,
               "firstUtteredDate": number,
               "lastUtteredDate": number,
               "utteranceString": "string"
            }
         ]
```

```
      }
   ]
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

botName

傳回語音資訊的機器人名稱。

類型：字串

長度約束：最小長度為 2。長度上限為 50。

模式：`^([A-Za-z]_?)+$`

utterances

一個UtteranceList對象數組，每個對象都包含描述由您的機器人處理的話語的UtteranceData對象列表。每個版本的回應最多包含 100 個`UtteranceData`物件。Amazon Lex 會傳回機器人在過去 15 天內收到的最常用語音。

類型：UtteranceList 物件陣列

錯誤

BadRequestException

請求的格式不正確。例如，值無效或缺少必填欄位。請檢查欄位值，然後再試一次。

HTTP 狀態碼：400

InternalFailureException

發生內部 Amazon Lex 錯誤。請再次嘗試您的請求。

HTTP 狀態碼：500

LimitExceededException

請求超過限制。請再次嘗試您的請求。

HTTP 狀態碼：429

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面
- AWS SDK for .NET
- AWS SDK for C++
- AWS 適用於轉到 V2 的 SDK
- AWS SDK for Java V2 的开发
- AWS 適用於 JavaScript V3 的 SDK
- AWS SDK for PHP
- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

ListTagsForResource

服務：Amazon Lex Model Building Service

獲取與指定資源關聯的標籤列表。只有機器人、機器人別名和機器人通道可以有關聯的標籤。

請求語法

```
GET /tags/resourceArn HTTP/1.1
```

URI 請求參數

請求會使用下列 URI 參數。

resourceArn

用於獲取標籤列表的資源的 Amazon 資源名稱（ARN）。

長度限制：長度下限為 1。最大長度為 1011。

必要：是

請求主體

請求沒有請求主體。

回應語法

```
HTTP/1.1 200
Content-type: application/json

{
   "tags": [
      {
         "key": "string",
         "value": "string"
      }
   ]
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

tags

與資源相關聯的標籤。

類型：Tag 物件陣列

陣列成員：項目數下限為 0。項目數上限為 200。

錯誤

BadRequestException

請求的格式不正確。例如，值無效或缺少必填欄位。請檢查欄位值，然後再試一次。

HTTP 狀態碼：400

InternalFailureException

發生內部 Amazon Lex 錯誤。請再次嘗試您的請求。

HTTP 狀態碼：500

LimitExceededException

請求超過限制。請再次嘗試您的請求。

HTTP 狀態碼：429

NotFoundException

找不到要求中指定的資源。請檢查資源，然後再試一次。

HTTP 狀態碼：404

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面
- AWS SDK for .NET
- AWS SDK for C++

- AWS 適用於轉到 V2 的 SDK
- AWS SDK for Java V2 的軟件
- AWS 適用於 JavaScript V3 的 SDK
- AWS SDK for PHP
- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

PutBot

服務：Amazon Lex Model Building Service

建立 Amazon Lex 交談機器人或取代現有的機器人。當您建立或更新機器人時，您只需要指定名稱、地區設定，以及是否將機器人導向 13 歲以下的兒童。您可以使用此功能稍後新增意圖，或從現有機器人中移除意圖。當您使用最少的資訊建立機器人時，系統會建立或更新機器人，但 Amazon Lex 會傳回回應FAILED。您可以在新增一或多個意圖之後建立機器人。如需 Amazon Lex 機器人的詳細資訊，請參閱Amazon Lex 運作方式。

如果您指定現有機器人的名稱，請求中的欄位會取代機器人$LATEST版本中的現有值。Amazon Lex 會移除您在請求中未提供值的任何欄位，但idleTTLInSeconds和privacySettings欄位設定為其預設值除外。如果您未指定必要欄位的值，Amazon Lex 會擲回例外狀況。

這項操作需要 lex:PutBot 動作的許可。如需詳細資訊，請參閱 Amazon Lex 的 Identity and Access Management。

請求語法

```
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"
}
```

URI 請求參數

請求會使用下列 URI 參數。

name

機器人的名稱。名稱不區分大小寫。

長度約束：最小長度為 2。長度上限為 50。

模式：^([A-Za-z]_?)+$

必要：是

請求主體

請求接受採用 JSON 格式的下列資料。

abortStatement

當 Amazon Lex 無法瞭解使用者在內容中的輸入時，它會嘗試幾次引出資訊。之後，Amazon Lex 會將中abortStatement定義的訊息傳送給使用者，然後取消對話。若要設定重試次數，請使用插槽類型的valueElicitationPrompt欄位。

例如，在披薩訂購機器人中，Amazon Lex 可能會詢問使用者「您想要什麼類型的外殼？」 如果使用者的回應不是預期的回應之一 (例如，「薄殼」、「深盤」等)，Amazon Lex 會嘗試再次引出正確的回應。

例如，在比薩餅訂購應用程序中，OrderPizza可能是意圖之一。此意圖可能需要CrustType插槽。您可以在建立CrustType插槽時指定valueElicitationPrompt欄位。

如果您已定義後援意圖，則不會將 cancel 陳述式傳送給使用者，則會改用後援意圖。如需詳細資訊，請參閱 AMAZON。 FallbackIntent。

類型：Statement 物件

必要：否

checksum

識別版本的特定修訂$LATEST版本。

當您建立新的機器人時，請將checksum欄位留空。如果你指定一個校驗和，你會得到一個BadRequestException異常。

當您想要更新機器人時，請將checksum欄位設定為該版本最新修訂$LATEST版的總和檢查碼。如果您未指定 checksum欄位，或者總和檢查碼與$LATEST版本不相符，則會出現PreconditionFailedException例外狀況。

類型：字串

必要：否

childDirected

對於使用 Amazon Lex 模型建置服務建立的每個 Amazon Lex 機器人，您必須在childDirected欄位中指定或全部或部分針對 13 歲以下兒童並遵守《兒童線上隱私保護法》(COPPA) 的網站、程式或其他應用程式是否與全部或部分導向或鎖定目標的網站、程式true或其他應用程false式相關。透過**true**在**childDirected**欄位中指定，即表示您確認您對 Amazon Lex 的使用與針對 13 歲以下兒童且受 COPPA 規範的全部或部分針對網站、程式或其他應用程式

有關。透過`false`在`childDirected`欄位中指定，即表示您確認您對 Amazon Lex 的使用與針對 13 歲以下兒童且受 COPPA 規範的全部或部分針對網站、程式或其他應用程式無關。您不得為欄位指定預設值，該`childDirected`欄位無法準確反映您對 Amazon Lex 的使用情況是否與針對 13 歲以下兒童的網站、程式或其他應用程式全部或部分針對 13 歲以下且受 COPPA 規限的網站、程式或其他應用程式有關。

如果您對 Amazon Lex 的使用涉及全部或部分針對 13 歲以下兒童的網站、程式或其他應用程式，您必須根據 COPPA 取得任何必要的可驗證父母同意。如需針對全部或部分針對 13 歲以下兒童的網站、程式或其他應用程式使用 Amazon Lex 的相關資訊，請參閱 Amazon Lex 常見問題集。

類型：布林值

必要：是

clarificationPrompt

當 Amazon Lex 不瞭解使用者的意圖時，會使用此訊息來取得說明。若要指定 Amazon Lex 應重複說明提示的次數，請使用此`maxAttempts`欄位。如果 Amazon Lex 仍然無法理解，它會在`abortStatement`欄位中傳送訊息。

當您創建澄清提示時，請確保它建議用戶的正確響應。例如，對於訂購比薩餅和飲料的機器人，您可能會創建此澄清提示：「您想要做什麼？ 您可以說「點一份比薩餅」或「點一杯飲料」。

如果您已經定義了一個後備意圖，如果澄清提示重複了`maxAttempts`字段中定義的次數，則會調用它。如需詳細資訊，請參閱 AMAZON。 FallbackIntent。

如果您未定義澄清提示，在執行階段 Amazon Lex 會在下列三種情況下傳回 400 錯誤請求例外狀況：

- 後續提示-當用戶響應後續提示但不提供意圖時。例如，回應後續提示：「您今天還想要什麼嗎？」 用戶說「是」。Amazon Lex 將傳回 400 錯誤請求例外狀況，因為它沒有要傳送給使用者以取得意圖的澄清提示。
- Lambda 函數-使用 Lambda 函數時，您會返回一個`ElicitIntent`對話框類型。由於 Amazon Lex 沒有向使用者取得意圖的澄清提示，因此會傳回 400 個錯誤請求例外狀況。
- PutSession 操作-使用`PutSession`操作時，您會發送對`ElicitIntent`話框類型。由於 Amazon Lex 沒有向使用者取得意圖的澄清提示，因此會傳回 400 個錯誤請求例外狀況。

類型：Prompt 物件

必要：否

createVersion

當設置為true一個新的編號版本的機器人被創建。這與呼叫作CreateBotVersion業相同。如果未指定createVersion，則預設值為false。

類型：布林值

必要：否

description

機器人的描述。

類型：字串

長度限制：長度下限為 0。長度上限為 200。

必要：否

detectSentiment

設定為true使用者的話語時，會傳送至 Amazon Comprehend 進行情緒分析。如果未指定detectSentiment，則預設值為false。

類型：布林值

必要：否

enableModelImprovements

設定為true以啟用存取自然語言理解改進功能。

將enableModelImprovements參數設定為時，true您可以使用nluIntentConfidenceThreshold參數來設定可信度分數。如需詳細資訊，請參閱可信度分數。

您只能在特定區域中設定enableModelImprovements參數。如果您將參數設定為true，您的機器人可以存取精確度改進。

您可以將 en-US 地區設定enableModelImprovements參數設定false為的區域包括：

- 美國東部 (維吉尼亞北部) (us-east-1)
- 美國西部 (奧勒岡) (us-west-2)
- 亞太區域 (雪梨) (ap-southeast-2)

- 歐洲 (愛爾蘭) (eu-west-1)

在其他區域和地區設定中，enableModelImprovements參數預設會設true定為。在這些區域和語言環境中設置參數false拋出ValidationException異常。

類型：布林值

必要：否

idleSessionTTLInSeconds

Amazon Lex 保留交談中收集之資料的時間上限 (以秒為單位)。

使用者互動工作階段會在指定的時間內保持作用中狀態。若在此期間沒有發生任何對話，則工作階段會過期，且 Amazon Lex 會刪除逾時之前提供的任何資料。

例如，假設用戶選擇了 OrderPizza 意圖，但通過下訂單而被側面跟踪。如果使用者未在指定的時間內完成訂單，Amazon Lex 會捨棄收集的插槽資訊，使用者必須重新開始。

如果您沒有在PutBot操作請求中包含idleSessionTTLInSeconds元素，Amazon Lex 會使用預設值。如果要求取代現有的機器人，這也是如此。

預設值為 300 秒 (5 分鐘)。

類型：整數

有效範圍：最小值為 60。最大值為 86400。

必要：否

intents

Intent 物件的陣列。每個意圖代表一個用戶可以表達的命令。例如，披薩訂購機器人可能支持 OrderPizza 意圖。如需詳細資訊，請參閱 Amazon Lex 運作方式。

類型：Intent 物件陣列

必要：否

locale

指定機器人的目標地區設定。機器人中使用的任何意圖都必須與機器人的語言環境相容。

預設值為 en-US。

類型：字串

有效值: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

必要：是

nluIntentConfidenceThreshold

決定 Amazon Lex 在或回PostText應中傳回替代意圖時插入AMAZON.KendraSearchIntent、PostContent或兩者的閾值。AMAZON.FallbackIntent AMAZON.FallbackIntent並且僅AMAZON.KendraSearchIntent在為機器人配置了它們時才會插入。

您必須將enableModelImprovements參數設定為，才true能在下列區域中使用可信度分數。

- 美國東部 (維吉尼亞北部) (us-east-1)
- 美國西部 (奧勒岡) (us-west-2)
- 亞太區域 (雪梨) (ap-southeast-2)
- 歐洲 (愛爾蘭) (eu-west-1)

在其他區域中，true依預設會將enableModelImprovements參數設定為。

例如，假設機器人的信賴度閾值設定為 0.80 和. AMAZON.FallbackIntent Amazon Lex 返回三個具有以下置信度分數的替代意圖：意圖（0.70），意圖 TB（0.60），意圖等（0.50）。來自PostText操作的響應將是：

- 亞馬遜。 FallbackIntent
- 意圖
- IntenTB
- IntenTC

類型：Double

有效範圍：最小值為 0。最大值為 1。

必要：否

processBehavior

如果將processBehavior元素設定為BUILD，Amazon Lex 會建立機器人以便執行該機器人。如果將元素設置為 SAVE Amazon Lex，則可以保存機器人，但不構建它。

如果未指定此值，則預設值為BUILD。

類型：字串

有效值:SAVE | BUILD

必要：否

tags

要新增至機器人的標籤清單。您只能在創建機器人時添加標籤，無法使用該PutBot操作更新機器人上的標籤。若要更新標籤，請使用 TagResource 操作。

類型：Tag 物件陣列

陣列成員：項目數下限為 0。項目數上限為 200。

必要：否

voiceId

您希望 Amazon Lex 用於與使用者進行語音互動的 Amazon Polly 語音識別碼。為語音設定的語言環境必須與機器人的地區設定相符。如需詳細資訊，請參閱 Amazon Polly 開發人員指南中的 Amazon Polly 中的聲音。

類型：字串

必要：否

回應語法

```
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"
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

abortStatement

Amazon Lex 用來取消對話的訊息。如需詳細資訊，請參閱 PutBot。

類型：Statement 物件

checksum

您建立的機器人的總和檢查碼。

類型：字串

childDirected

對於使用 Amazon Lex 模型建置服務建立的每個 Amazon Lex 機器人，您必須在`childDirected`欄位中指定或全部或部分針對 13 歲以下兒童並遵守《兒童線上隱私保護法》(COPPA) 的網站、程式或其他應用程式是否與全部或部分導向或鎖定目標的網站、程式`true`或其他應用程`false`式相關。透過**`true`**在**`childDirected`**欄位中指定，即表示您確認您對 Amazon Lex 的使用與針對 13 歲以下兒童且受 COPPA 規範的全部或部分針對網站、程式或其他應用程式有關。透過`false`在`childDirected`欄位中指定，即表示您確認您對 Amazon Lex 的使用與針對 13 歲以下兒童且受 COPPA 規範的全部或部分針對網站、程式或其他應用程式無關。您不得為欄位指定預設值，該`childDirected`欄位無法準確反映您對 Amazon Lex 的使用情況是否與針對 13 歲以下兒童的網站、程式或其他應用程式全部或部分針對 13 歲以下且受 COPPA 規限的網站、程式或其他應用程式有關。

如果您對 Amazon Lex 的使用涉及全部或部分針對 13 歲以下兒童的網站、程式或其他應用程式，您必須根據 COPPA 取得任何必要的可驗證父母同意。如需針對全部或部分針對 13 歲以下兒童的網站、程式或其他應用程式使用 Amazon Lex 的相關資訊，請參閱 Amazon Lex 常見問題集。

類型：布林值

clarificationPrompt

Amazon Lex 在不了解使用者意圖時所使用的提示。如需詳細資訊，請參閱 PutBot。

類型：Prompt 物件

createdDate

建立機器人的日期。

類型：Timestamp

createVersion

`True`如果創建了新版本的機器人。如果未在請求中指定該字`createVersion`段，則該`createVersion`字段在響應中設置為 false。

類型：布林值

description

機器人的描述。

類型：字串

長度限制：長度下限為 0。長度上限為 200。

detectSentiment

`true`如果機器人設定為將使用者話語傳送至 Amazon Comprehend 進行情緒分析。如果未在請求中指定該字`detectSentiment`段，則該`detectSentiment`字段`false`在響應中。

類型：布林值

enableModelImprovements

指出機器人是否使用精確度改進。 `true`表示機器人正在使用改進，否則，`false`。

類型：布林值

failureReason

如果`status`是`FAILED`，Amazon Lex 會提供無法建置機器人的原因。

類型：字串

idleSessionTTLInSeconds

Amazon Lex 保留交談中收集之資料的最長時間長度。如需詳細資訊，請參閱 PutBot。

類型：整數

有效範圍：最小值為 60。最大值為 86400。

intents

`Intent` 物件的陣列。如需詳細資訊，請參閱 PutBot。

類型：Intent 物件陣列

lastUpdatedDate

機器人更新的日期。建立資源時，建立日期和上次更新日期相同。

類型：Timestamp

locale

機器人的目標地區設定。

類型：字串

有效值: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

機器人的名稱。

類型：字串

長度約束：最小長度為 2。長度上限為 50。

模式：^([A-Za-z]_?)+$

nluIntentConfidenceThreshold

決定 Amazon Lex 在或回PostText應中傳回替代意圖時插入AMAZON.FallbackIntentAMAZON.KendraSearchIntent、或兩者的位置的PostContent分數。 AMAZON.FallbackIntent如果所有意圖的可信度分數低於此值，則會插入。AMAZON.KendraSearchIntent只有在為機器人配置時才會插入。

類型：Double

有效範圍：最小值為 0。最大值為 1。

status

當您傳送建立機器人的請求時BUILD，Amazon Lex 會將status回應元素processBehavior設定為BUILDING。

在該READY_BASIC_TESTING狀態下，您可以使用與插槽類型中機器人意圖和值配置的話語完全匹配的用戶輸入來測試機器人。

如果 Amazon Lex 無法構建機器人，Amazon Lex 設置status為FAILED。Amazon Lex 返回failureReason響應元素失敗的原因。

當您設定processBehavior為時SAVE，Amazon Lex 會將狀態碼設定為NOT BUILT。

當機器人處於READY狀態時，您可以測試並發佈機器人。

類型：字串

有效值:BUILDING | READY | READY_BASIC_TESTING | FAILED | NOT_BUILT

tags

與機器人相關聯的標籤列表。

類型：Tag 物件陣列

陣列成員：項目數下限為 0。項目數上限為 200。

version

機器人的版本。對於新的機器人，版本始終是$LATEST。

類型：字串

長度限制：長度下限為 1。長度上限為 64。

模式：\$LATEST|[0-9]+

voiceId

亞馬遜 Lex 用於與用戶進行語音交互的 Amazon Polly 語音 ID。如需詳細資訊，請參閱 PutBot。

類型：字串

錯誤

BadRequestException

請求的格式不正確。例如，值無效或缺少必填欄位。請檢查欄位值，然後再試一次。

HTTP 狀態碼：400

ConflictException

處理要求時發生衝突。請再次嘗試您的請求。

HTTP 狀態碼：409

InternalFailureException

發生內部 Amazon Lex 錯誤。請再次嘗試您的請求。

HTTP 狀態碼：500

LimitExceededException

請求超過限制。請再次嘗試您的請求。

HTTP 狀態碼：429

PreconditionFailedException

您嘗試變更之資源的總和檢查碼與要求中的總和檢查碼不符。檢查資源的總和檢查碼，然後再試一次。

HTTP 狀態碼：412

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面
- AWS SDK for .NET
- AWS SDK for C++
- AWS 適用於轉到 V2 的 SDK
- AWS SDK for Java V2 的軟件
- AWS 適用於 JavaScript V3 的 SDK
- AWS SDK for PHP
- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

PutBotAlias

服務：Amazon Lex Model Building Service

為指定的機器人版本建立別名，或取代指定機器人的別名。若要變更別名指向的機器人版本，請取代別名。如需關於別名的詳細資訊，請參閱版本控制與別名。

這項操作需要 `lex:PutBotAlias` 動作的許可。

請求語法

```
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"
      }
   ]
}
```

URI 請求參數

請求會使用下列 URI 參數。

botName

機器人的名稱。

長度約束：最小長度為 2。長度上限為 50。

模式：^([A-Za-z]_?)+$

必要：是

name

別名的名稱。名稱不區分大小寫。

長度限制：長度下限為 1。長度上限為 100。

模式：^([A-Za-z]_?)+$

必要：是

請求主體

請求接受採用 JSON 格式的下列資料。

botVersion

機器人的版本。

類型：字串

長度限制：長度下限為 1。長度上限為 64。

模式：\$LATEST|[0-9]+

必要：是

checksum

識別版本的特定修訂$LATEST版本。

當您建立新的機器人別名時，請將checksum欄位留空。如果你指定一個校驗和，你會得到一個BadRequestException異常。

當您想要更新機器人別名時，請將checksum欄位設定為該版本最新修訂$LATEST版本的總和檢查碼。如果您未指定 checksum欄位，或者總和檢查碼與$LATEST版本不相符，則會出現PreconditionFailedException例外狀況。

類型：字串

必要：否

conversationLogs

別名的交談記錄設定。

類型：ConversationLogsRequest 物件

必要：否

description

別名的描述。

類型：字串

長度限制：長度下限為 0。長度上限為 200。

必要：否

tags

要新增至機器人別名的標籤清單。您只能在建立別名時新增標籤，無法使用此PutBotAlias操作來更新機器人別名上的標籤。若要更新標籤，請使用 TagResource 操作。

類型：Tag 物件陣列

陣列成員：項目數下限為 0。項目數上限為 200。

必要：否

回應語法

```
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"
      }
   ]
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

botName

別名所指向的機器人的名稱。

類型：字串

長度約束：最小長度為 2。長度上限為 50。

模式：^([A-Za-z]_?)+$

botVersion

別名指向的機器人版本。

類型：字串

長度限制：長度下限為 1。長度上限為 64。

模式：\$LATEST|[0-9]+

checksum

目前版本別名的總和檢查碼。

類型：字串

conversationLogs

決定 Amazon Lex 如何將交談日誌用於別名的設定。

類型：ConversationLogsResponse 物件

createdDate

建立機器人別名的日期。

類型：Timestamp

description

別名的描述。

類型：字串

長度限制：長度下限為 0。長度上限為 200。

lastUpdatedDate

更新機器人別名的日期。建立資源時，建立日期和上次更新日期相同。

類型：Timestamp

name

別名的名稱。

類型：字串

長度限制：長度下限為 1。長度上限為 100。

模式：`^([A-Za-z]_?)+$`

tags

與機器人相關聯的標籤列表。

類型：Tag 物件陣列

陣列成員：項目數下限為 0。項目數上限為 200。

錯誤

BadRequestException

請求的格式不正確。例如，值無效或缺少必填欄位。請檢查欄位值，然後再試一次。

HTTP 狀態碼：400

ConflictException

處理要求時發生衝突。請再次嘗試您的請求。

HTTP 狀態碼：409

InternalFailureException

發生內部 Amazon Lex 錯誤。請再次嘗試您的請求。

HTTP 狀態碼：500

LimitExceededException

請求超過限制。請再次嘗試您的請求。

HTTP 狀態碼：429

PreconditionFailedException

您嘗試變更之資源的總和檢查碼與要求中的總和檢查碼不符。檢查資源的總和檢查碼，然後再試一次。

HTTP 狀態碼：412

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面
- AWS SDK for .NET
- AWS SDK for C++
- AWS 適用於轉到 V2 的 SDK

- AWS SDK for Java V2 的軟件
- AWS 適用於 JavaScript V3 的 SDK
- AWS SDK for PHP
- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

PutIntent

服務：Amazon Lex Model Building Service

建立意圖或取代現有的意圖。

要定義用戶和您的機器人之間的交互，您可以使用一個或多個意圖。例如，對於比薩餅訂購機器人，您將創建一個`OrderPizza`意圖。

若要建立意圖或取代現有的意圖，您必須提供下列資訊：

- 意圖名稱。例如 `OrderPizza`。
- 語音樣本。例如，「我可以點披薩嗎？」 和「我想點披薩。」
- 要收集的信息。您可以為機器人將向使用者請求的資訊指定位置類型。您可以指定標準插槽類型，例如日期或時間，或自訂插槽類型，例如比薩餅的大小和外殼。
- 意圖將如何實現。您可以提供 Lambda 函數或設定意圖，以將意圖資訊傳回給用戶端應用程式。如果您使用 Lambda 函數，當所有意圖資訊都可用時，Amazon Lex 會叫用您的 Lambda 函數。如果您將意圖配置為將意圖信息返回給客戶端應用程序。

您可以在請求中指定其他可選信息，例如：

- 要求使用者確認意圖的確認提示。例如，「我可以點你的披薩嗎？」
- 在意圖完成後發送給用戶的結論聲明。例如，「我下了你的披薩訂單。」
- 後續提示，要求使用者進行其他活動。例如，詢問「您想用披薩點飲料嗎？」

如果您指定現有的意圖名稱來更新意圖，Amazon Lex 會以請求中的值取代意圖`$LATEST`版本中的值。Amazon Lex 會移除您在請求中未提供的欄位。如果您未指定必要欄位，Amazon Lex 就會擲回例外狀況。當您更新意圖的`$LATEST`版本時，使用意圖`$LATEST`版本的任何機器人的`status`欄位都會設定為`NOT_BUILT`。

如需詳細資訊，請參閱 Amazon Lex 運作方式。

這項操作需要 `lex:PutIntent` 動作的許可。

請求語法

```
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"
         }
      }
   ]
}
```

URI 請求參數

請求會使用下列 URI 參數。

name

意圖的名稱。名稱不區分大小寫。

該名稱不能匹配內置的意圖名稱，或帶有「AMAZON」的內置意圖名稱。刪除。例如，因為有一個名為的內置意圖AMAZON.HelpIntent，因此您無法創建名為的自定義意圖HelpIntent。

如需內建意圖清單，請參閱 Alexa Skills Kit 中的標準內建意圖。

長度限制：長度下限為 1。長度上限為 100。

模式：^([A-Za-z]_?)+$

必要：是

請求主體

請求接受採用 JSON 格式的下列資料。

checksum

識別版本的特定修訂$LATEST版本。

當您建立新意圖時，請將checksum欄位保留空白。如果你指定一個校驗和，你會得到一個BadRequestException異常。

當您想要更新意圖時，請將checksum欄位設定為該版本最新修訂$LATEST版本的總和檢查碼。如果您未指定 checksum欄位，或者總和檢查碼與$LATEST版本不相符，則會出現PreconditionFailedException例外狀況。

類型：字串

必要：否

conclusionStatement

您希望 Amazon Lex 在 Lambda 函數成功完成意圖後傳送給使用者的陳述式。

僅當您在中提供 Lambda 函數時，此元素才相關fulfillmentActivity。如果將意圖返回到客戶端應用程序，則無法指定此元素。

> **Note**
>
> followUpPrompt和conclusionStatement是相互排斥的。您只能指定一個。

類型：Statement 物件

必要：否

confirmationPrompt

提示使用者確認意圖。這個問題應該有「是」或「否」的答案。

Amazon Lex 使用此提示來確保使用者確認意圖已準備好履行。例如，出於OrderPizza意圖，您可能需要在下訂單之前確認訂單是否正確。對於其他意圖 (例如只回應使用者問題的意圖)，您可能不需要在提供資訊之前詢問使用者確認。

> **Note**
>
> 您必須同時提供rejectionStatement和confirmationPrompt，或兩者都不提供。

類型：Prompt 物件

必要：否

createVersion

當設定為意圖true的新編號版本時，會建立意圖。這與呼叫作CreateIntentVersion業相同。如果未指定createVersion，預設值為false。

類型：布林值

必要：否

description

意圖的描述。

類型：字串

長度限制：長度下限為 0。長度上限為 200。

必要：否

dialogCodeHook

指定要為每個使用者輸入呼叫的 Lambda 函數。您可以叫用此 Lambda 函數來個人化使用者互動。

例如，假設您的機器人判斷使用者是 John。您的 Lambda 函數可能會從後端資料庫擷取 John 的資訊，並預先填入某些值。例如，如果您發現 John 不耐受麩質，您可能會將對應的意圖槽設定為 true。GlutenIntolerant您可能會找到 John 的電話號碼，並設定對應的工作階段屬性。

類型：CodeHook 物件

必要：否

followUpPrompt

Amazon Lex 在完成意圖後使用此提示來徵求其他活動。例如，在OrderPizza意圖達成之後，您可能會提示使用者訂購飲料。

Amazon Lex 採取的動作取決於使用者的回應，如下所示：

- 如果使用者說「是」，則會以針對機器人設定的說明提示進行回應。
- 如果用戶說「是」並繼續觸發意圖的話語，它會為意圖開始對話。
- 如果使用者說「否」，則會回應為後續提示設定的拒絕陳述式。
- 如果它不識別話語，它會再次重複後續提示。

followUpPrompt欄位和conclusionStatement欄位是互斥的。您只能指定一個。

類型：FollowUpPrompt 物件

必要：否

fulfillmentActivity

必要。描述如何實現意圖。例如，在使用者提供披薩訂單的所有資訊之後，會fulfillmentActivity定義機器人如何向當地披薩商店下訂單。

您可以設定 Amazon Lex 將所有意圖資訊傳回至用戶端應用程式，或指示它叫用可處理意圖的 Lambda 函數 (例如，向披薩店下訂單)。

類型：FulfillmentActivity 物件

必要：否

inputContexts

InputContext物件陣列，列出 Amazon Lex 必須處於作用中狀態的內容，才能在與使用者交談中選擇意圖。

類型：InputContext 物件陣列

陣列成員：項目數下限為 0。項目數上限為 5。

必要：否

kendraConfiguration

使用AMAZON.KendraSearchIntent意圖連線至 Amazon Kendra 索引所需的組態資訊。如需詳細資訊，請參閱 AMAZON。 KendraSearchIntent。

類型：KendraConfiguration 物件

必要：否

outputContexts

OutputContext物件陣列，列出實現意圖時意圖啟動的前後關聯。

類型：OutputContext 物件陣列

陣列成員：項目數下限為 0。項目數上限為 10。

必要：否

parentIntentSignature

要作為此意圖基礎的內建意圖的唯一識別碼。若要尋找意圖的簽名，請參閱 Alexa 技能套件中的標準內建意圖。

類型：字串

必要：否

rejectionStatement

當使用者對於中定義的問題回答「否」時confirmationPrompt，Amazon Lex 會回應此陳述式，確認意圖已取消。

> **Note**
>
> 您必須同時提供rejectionStatement和confirmationPrompt，或兩者都不提供。

類型：Statement 物件

必要：否

sampleUtterances

用戶可能會說用來表示意圖的語音（字符串）的數組。例如，「我想要 {PizzaSize} 比薩餅」，「訂購 {數量} {PizzaSize} 比薩餅」。

在每個話語中，插槽名稱用大括號括起來。

類型：字串陣列

陣列成員：項目數下限為 0。最多可容納 1500 件物品數量。

長度限制：長度下限為 1。長度上限為 200。

必要：否

slots

意圖槽的陣列。在執行階段，Amazon Lex 會使用插槽中定義的提示，向使用者提出所需的插槽值。如需詳細資訊，請參閱 Amazon Lex 運作方式。

類型：Slot 物件陣列

陣列成員：項目數下限為 0。項目數上限為 100。

必要：否

回應語法

```
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"
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

checksum

建立或更新之意圖$LATEST版本的總和檢查碼。

類型：字串

conclusionStatement

在意圖中指定的 Lambda 函數達成fulfillmentActivity意圖之後，Amazon Lex 就會將此陳述式傳達給使用者。

類型：Statement 物件

confirmationPrompt

如果意圖中有定義，Amazon Lex 會提示使用者在完成意圖之前確認意圖。

類型：Prompt 物件

createdDate

建立意圖的日期。

類型：Timestamp

createVersion

True如果創建了意圖的新版本。如果未在請求中指定該字createVersion段，則該createVersion字段在響應中設置為 false。

類型：布林值

description

意圖的描述。

類型：字串

長度限制：長度下限為 0。長度上限為 200。

dialogCodeHook

如果意圖中已定義，Amazon Lex 會針對每個使用者輸入叫用此 Lambda 函數。

類型：CodeHook 物件

followUpPrompt

如果意圖中有定義，Amazon Lex 會在完成意圖後使用此提示來徵求其他使用者活動。

類型：FollowUpPrompt 物件

fulfillmentActivity

如果意圖中定義，Amazon Lex 會在使用者提供意圖所需的所有資訊之後，叫用此 Lambda 函數來完成意圖。

類型：FulfillmentActivity 物件

inputContexts

InputContext物件陣列，列出 Amazon Lex 必須處於作用中狀態的內容，才能在與使用者交談中選擇意圖。

類型：InputContext 物件陣列

陣列成員：項目數下限為 0。項目數上限為 5。

kendraConfiguration

連接至 Amazon Kendra 索引並使用AMAZON.KendraSearchIntent意圖所需的組態資訊 (如果有的話)。

類型：KendraConfiguration 物件

lastUpdatedDate

更新意圖的日期。建立資源時，建立日期與上次更新日期相同。

類型：Timestamp

name

意圖的名稱。

類型：字串

長度限制：長度下限為 1。長度上限為 100。

模式：^([A-Za-z]_?)+$

outputContexts

OutputContext物件陣列，列出實現意圖時意圖啟動的前後關聯。

類型：OutputContext 物件陣列

陣列成員：項目數下限為 0。項目數上限為 10。

parentIntentSignature

此意圖所依據之內建意圖的唯一識別碼。

類型：字串

rejectionStatement

如果使用者對 confirmationPrompt Amazon Lex 中定義的問題回答「否」，則會以此陳述式回應，以確認意圖已取消。

類型：Statement 物件

sampleUtterances

針對意圖設定的範例語音陣列。

類型：字串陣列

陣列成員：項目數下限為 0。最多可容納 1500 件物品數量。

長度限制：長度下限為 1。長度上限為 200。

slots

針對意圖設定的意圖槽陣列。

類型：Slot 物件陣列

陣列成員：項目數下限為 0。項目數上限為 100。

version

意圖的版本。對於一個新的意圖，版本總是$LATEST。

類型：字串

長度限制：長度下限為 1。長度上限為 64。

模式：\$LATEST|[0-9]+

錯誤

BadRequestException

請求的格式不正確。例如，值無效或缺少必填欄位。請檢查欄位值，然後再試一次。

HTTP 狀態碼：400

ConflictException

處理要求時發生衝突。請再次嘗試您的請求。

HTTP 狀態碼：409

InternalFailureException

發生內部 Amazon Lex 錯誤。請再次嘗試您的請求。

HTTP 狀態碼：500

LimitExceededException

請求超過限制。請再次嘗試您的請求。

HTTP 狀態碼：429

PreconditionFailedException

您嘗試變更之資源的總和檢查碼與要求中的總和檢查碼不符。檢查資源的總和檢查碼，然後再試一次。

HTTP 狀態碼：412

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面
- AWS SDK for .NET
- AWS SDK for C++
- AWS 適用於轉到 V2 的 SDK
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 JavaScript V3 的 SDK
- AWS SDK for PHP
- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

PutSlotType

服務：Amazon Lex Model Building Service

建立自訂插槽類型或取代現有的自訂插槽類型。

若要建立自訂槽類型，請指定插槽類型的名稱和一組列舉值，這些值是此類型槽可承擔的值。如需詳細資訊，請參閱 Amazon Lex 運作方式。

如果您指定現有插槽類型的名稱，請求中的欄位會取代插槽類型$LATEST版本中的現有值。Amazon Lex 會移除您在請求中未提供的欄位。如果您未指定必填欄位，Amazon Lex 就會擲回例外狀況。當您更新插槽類型的$LATEST版本時，如果機器人使用包含插槽類型的意圖$LATEST版本，則機器人的status欄位會設定為NOT_BUILT。

這項操作需要 lex:PutSlotType 動作的許可。

請求語法

```
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"
}
```

URI 請求參數

請求會使用下列 URI 參數。

name

位置類型的名稱。名稱不區分大小寫。

該名稱不能匹配內置插槽類型名稱，或帶有「AMAZON」的內置插槽類型名稱。刪除。例如，由於有一個名為的內置插槽類型AMAZON.DATE，因此您無法創建名為的自定義插槽類型DATE。

如需內建插槽類型的清單，請參閱 Alexa 技能套件中的插槽類型參考。

長度限制：長度下限為 1。長度上限為 100。

模式：^([A-Za-z]_?)+$

必要：是

請求主體

請求接受採用 JSON 格式的下列資料。

checksum

識別版本的特定修訂$LATEST版本。

當您建立新插槽類型時，請將checksum欄位保留空白。如果你指定一個校驗和，你會得到一個BadRequestException異常。

當您要更新插槽類型時，請將checksum欄位設定為該版本最新修訂$LATEST版的總和檢查碼。如果您未指定 checksum欄位，或者總和檢查碼與$LATEST版本不相符，則會出現PreconditionFailedException例外狀況。

類型：字串

必要：否

createVersion

當設置true為插槽類型的新編號版本被創建。這與呼叫作CreateSlotTypeVersion業相同。如果未指定createVersion，預設值為false。

類型：布林值

必要：否

description

位置類型的描述。

類型：字串

長度限制：長度下限為 0。長度上限為 200。

必要：否

enumerationValues

定義插槽類型可採用的值的EnumerationValue物件清單。每個值都可以有一份清單synonyms，這些清單是額外的值，可協助訓練機器學習模型，瞭解它為槽所解析的值。

正則表達式插槽類型不需要枚舉值。所有其他插槽類型都需要列舉值清單。

Amazon Lex 解析插槽值時，會產生一份解析清單，其中包含插槽最多五個可能值。如果您使用的是 Lambda 函數，則會將此解析清單傳遞給函數。如果您不使用 Lambda 函數，您可以選擇將使用者輸入的值或解析度清單中的第一個值傳回為插槽值。此valueSelectionStrategy欄位指示要使用的選項。

類型：EnumerationValue 物件陣列

陣列成員：項目數下限為 0。最多可包含 1 萬個項目。

必要：否

parentSlotTypeSignature

用作插槽類型的父插槽類型的內建插槽類型。當您定義父插槽類型時，新的插槽類型與父插槽類型具有所有相同的模型組態。

僅支援 AMAZON.AlphaNumeric。

類型：字串

長度限制：長度下限為 1。長度上限為 100。

模式：^((AMAZON\.)_?|[A-Za-z]_?)+

必要：否

slotTypeConfigurations

擴充上層內建插槽類型的組態資訊。模型組態會新增至父插槽類型的設定中。

類型：SlotTypeConfiguration 物件陣列

陣列成員：項目數下限為 0。項目數上限為 10。

必要：否

valueSelectionStrategy

決定 Amazon Lex 用來傳回插槽類型值的插槽解析策略。該欄位可以設定為下列其中一個值：

- ORIGINAL_VALUE-如果使用者值與槽值類似，則傳回使用者輸入的值。
- TOP_RESOLUTION-如果有插槽的解析度清單，請傳回解析度清單中的第一個值作為插槽類型值。如果沒有解析清單，則傳回 null。

如果未指定valueSelectionStrategy，預設值為ORIGINAL_VALUE。

類型：字串

有效值:ORIGINAL_VALUE | TOP_RESOLUTION

必要：否

回應語法

```
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"
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

checksum

插槽類型$LATEST版本的總和檢查碼。

類型：字串

createdDate

插槽類型的建立日期。

類型：Timestamp

createVersion

True如果創建了插槽類型的新版本。如果未在請求中指定該字createVersion段，則該createVersion字段在響應中設置為 false。

類型：布林值

description

位置類型的描述。

類型：字串

長度限制：長度下限為 0。長度上限為 200。

enumerationValues

定義插槽類型可採用的值的EnumerationValue物件清單。

類型：EnumerationValue 物件陣列

陣列成員：項目數下限為 0。最多可包含 1 萬個項目。

lastUpdatedDate

更新插槽類型的日期。建立插槽類型時，建立日期與上次更新日期相同。

類型：Timestamp

name

位置類型的名稱。

類型：字串

長度限制：長度下限為 1。長度上限為 100。

模式：^([A-Za-z]_?)+$

parentSlotTypeSignature

用作插槽類型的父插槽類型的內建插槽類型。

類型：字串

長度限制：長度下限為 1。長度上限為 100。

模式：^((AMAZON\.)_?|[A-Za-z]_?)+

slotTypeConfigurations

擴充上層內建插槽類型的組態資訊。

類型：SlotTypeConfiguration 物件陣列

陣列成員：項目數下限為 0。項目數上限為 10。

valueSelectionStrategy

Amazon Lex 用來判斷插槽值的插槽解析策略。如需詳細資訊，請參閱 PutSlotType。

類型：字串

有效值:ORIGINAL_VALUE | TOP_RESOLUTION

version

插槽類型的版本。對於新插槽類型，版本始終是$LATEST。

類型：字串

長度限制：長度下限為 1。長度上限為 64。

模式：\$LATEST|[0-9]+

錯誤

BadRequestException

請求的格式不正確。例如，值無效或缺少必填欄位。請檢查欄位值，然後再試一次。

HTTP 狀態碼：400

ConflictException

處理要求時發生衝突。請再次嘗試您的請求。

HTTP 狀態碼：409

InternalFailureException

發生內部 Amazon Lex 錯誤。請再次嘗試您的請求。

HTTP 狀態碼：500

LimitExceededException

請求超過限制。請再次嘗試您的請求。

HTTP 狀態碼：429

PreconditionFailedException

您嘗試變更之資源的總和檢查碼與要求中的總和檢查碼不符。檢查資源的總和檢查碼，然後再試一次。

HTTP 狀態碼：412

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面
- AWS SDK for .NET
- AWS SDK for C++
- AWS 適用於轉到 V2 的 SDK
- AWS SDK for Java V2 的开发
- AWS 適用於 JavaScript V3 的 SDK
- AWS SDK for PHP
- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

StartImport

服務：Amazon Lex Model Building Service

啟動將資源匯入至 Amazon Lex 的任務。

請求語法

```
POST /imports/ HTTP/1.1
Content-type: application/json

{
   "mergeStrategy": "string",
   "payload": blob,
   "resourceType": "string",
   "tags": [
      {
         "key": "string",
         "value": "string"
      }
   ]
}
```

URI 請求參數

請求不會使用任何 URI 參數。

請求主體

請求接受採用 JSON 格式的下列資料。

mergeStrategy

指定當存在具有相同名稱的現有資源時，StartImport作業應採取的動作。

- FAIL_ON_CONFLACT-匯入檔案中的資源與現有資源之間的第一次衝突時，匯入作業會停止。造成衝突的資源名稱位於對GetImport作業的回應failureReason欄位中。

  OVERWRITE_LATEST-即使與現有資源發生衝突，匯入作業也會繼續進行。現有資源的 $LASTEST 版本會以匯入檔案中的資料覆寫。

類型：字串

有效值:OVERWRITE_LATEST | FAIL_ON_CONFLICT

必要：是

payload

二進制格式的 zip 歸檔。歸檔應該包含一個文件，一個包含要導入的資源的 JSON 文件。資源應與resourceType欄位中指定的類型相符。

類型：Base64 編碼的二進位資料物件

必要：是

resourceType

指定要匯出的資源類型。每個資源也會匯出它所依賴的任何資源。

- 機器人會匯出相依意圖。
- 意圖匯出從屬槽類型。

類型：字串

有效值:BOT | INTENT | SLOT_TYPE

必要：是

tags

要添加到導入機器人的標籤列表。您只能在匯入機器人時新增標籤，無法將標籤新增至意圖或位置類型。

類型：Tag 物件陣列

陣列成員：項目數下限為 0。項目數上限為 200。

必要：否

回應語法

```
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"
      }
   ]
}
```

回應元素

如果動作成功，則服務傳回 HTTP 201 回應。

服務會傳回下列 JSON 格式的資料。

createdDate

要求匯入工作之日期和時間的時間戳記。

類型：Timestamp

importId

特定匯入工作的識別碼。

類型：字串

importStatus

匯入工作的狀態。如果狀態為FAILED，您可以使用GetImport作業取得失敗的原因。

類型：字串

有效值:IN_PROGRESS | COMPLETE | FAILED

mergeStrategy

發生合併衝突時要採取的動作。

類型：字串

有效值:OVERWRITE_LATEST | FAIL_ON_CONFLICT

name

指定給匯入工作的名稱。

類型：字串

長度限制：長度下限為 1。長度上限為 100。

模式：[a-zA-Z_]+

resourceType

要匯入的資源類型。

類型：字串

有效值:BOT | INTENT | SLOT_TYPE

tags

添加到導入機器人的標籤列表。

類型：Tag 物件陣列

陣列成員：項目數下限為 0。項目數上限為 200。

錯誤

BadRequestException

請求的格式不正確。例如，值無效或缺少必填欄位。請檢查欄位值，然後再試一次。

HTTP 狀態碼：400

InternalFailureException

發生內部 Amazon Lex 錯誤。請再次嘗試您的請求。

HTTP 狀態碼：500

LimitExceededException

請求超過限制。請再次嘗試您的請求。

HTTP 狀態碼：429

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面
- AWS SDK for .NET
- AWS SDK for C++
- AWS 適用於轉到 V2 的 SDK
- AWS SDK for Java V2 的開發
- AWS 適用於 JavaScript V3 的 SDK
- AWS SDK for PHP
- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

StartMigration

服務：Amazon Lex Model Building Service

開始將機器人從 Amazon Lex V1 遷移到 Amazon Lex V2。當您想要利用 Amazon Lex V2 的新功能時，請移轉您的機器人。

如需詳細資訊，請參閱 Amazon Lex 開發人員指南中的移轉機器人。

請求語法

```
POST /migrations HTTP/1.1
Content-type: application/json

{
   "migrationStrategy": "string",
   "v1BotName": "string",
   "v1BotVersion": "string",
   "v2BotName": "string",
   "v2BotRole": "string"
}
```

URI 請求參數

請求不會使用任何 URI 參數。

請求主體

請求接受採用 JSON 格式的下列資料。

migrationStrategy

用來執行移轉的策略。

- CREATE_NEW-建立新的 Amazon Lex V2 機器人，並將 Amazon Lex V1 機器人移轉至新的機器人。
- UPDATE_EXISTING-覆寫現有的 Amazon Lex V2 機器人中繼資料和要遷移的地區設定。它不會變更 Amazon Lex V2 機器人中的任何其他語言環境。如果語言環境不存在，則會在 Amazon Lex V2 機器人中建立新的地區設定。

類型：字串

有效值:CREATE_NEW | UPDATE_EXISTING

必要：是

v1BotName

您要移轉至 Amazon Lex V2 之亞馬 Amazon Lex V1 機器人的名稱。

類型：字串

長度約束：最小長度為 2。長度上限為 50。

模式：^([A-Za-z]_?)+$

必要：是

v1BotVersion

要移轉至 Amazon Lex V2 的機器人版本。您可以遷移$LATEST版本以及任何編號的版本。

類型：字串

長度限制：長度下限為 1。長度上限為 64。

模式：\$LATEST|[0-9]+

必要：是

v2BotName

您要將 Amazon Lex V1 機器人移轉到的目標之 Amazon Lex V2 機器人的名稱。

- 如果 Amazon Lex V2 機器人不存在，您必須使用CREATE_NEW遷移策略。
- 如果 Amazon Lex V2 機器人存在，您必須使用UPDATE_EXISTING遷移策略來變更 Amazon Lex V2 機器人的內容。

類型：字串

長度限制：長度下限為 1。長度上限為 100。

模式：^([0-9a-zA-Z][_-]?)+$

必要：是

v2BotRole

亞馬遜萊克斯用來執行亞馬遜萊克 Amazon Lex V2 機器人的 IAM 角色。

類型：字串

長度限制：長度下限為 20。長度上限為 2048。

模式：^arn:[\w\-]+:iam::[\d]{12}:role/.+$

必要：是

回應語法

```
HTTP/1.1 202
Content-type: application/json

{
   "migrationId": "string",
   "migrationStrategy": "string",
   "migrationTimestamp": number,
   "v1BotLocale": "string",
   "v1BotName": "string",
   "v1BotVersion": "string",
   "v2BotId": "string",
   "v2BotRole": "string"
}
```

回應元素

如果動作成功，則服務傳回 HTTP 202 回應。

服務會傳回下列 JSON 格式的資料。

migrationId

Amazon Lex 指派給遷移的唯一識別碼。

類型：字串

長度約束：固定長度為 10。

模式：^[0-9a-zA-Z]+$

migrationStrategy

用來執行移轉的策略。

類型：字串

有效值:CREATE_NEW | UPDATE_EXISTING

migrationTimestamp

移轉開始的日期和時間。

類型：Timestamp

v1BotLocale

Amazon Lex V1 機器人所使用的地區設定。

類型：字串

有效值: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

您要移轉至 Amazon Lex V2 之亞馬 Amazon Lex V1 機器人的名稱。

類型：字串

長度約束：最小長度為 2。長度上限為 50。

模式：^([A-Za-z]_?)+$

v1BotVersion

要移轉至 Amazon Lex V2 的機器人版本。

類型：字串

長度限制：長度下限為 1。長度上限為 64。

模式：\$LATEST|[0-9]+

v2BotId

Amazon Lex V2 機器人的唯一識別碼。

類型：字串

長度約束：固定長度為 10。

模式：^[0-9a-zA-Z]+$

v2BotRole

亞馬遜萊克斯用來執行亞馬遜萊克 Amazon Lex V2 機器人的 IAM 角色。

類型：字串

長度限制：長度下限為 20。長度上限為 2048。

模式：`^arn:[\w\-]+:iam::[\d]{12}:role/.+$`

錯誤

AccessDeniedException

您的 IAM 使用者或角色沒有呼叫遷移機器人所需的 Amazon Lex V2 API 的權限。

HTTP 狀態碼：403

BadRequestException

請求的格式不正確。例如，值無效或缺少必填欄位。請檢查欄位值，然後再試一次。

HTTP 狀態碼：400

InternalFailureException

發生內部 Amazon Lex 錯誤。請再次嘗試您的請求。

HTTP 狀態碼：500

LimitExceededException

請求超過限制。請再次嘗試您的請求。

HTTP 狀態碼：429

NotFoundException

找不到要求中指定的資源。請檢查資源，然後再試一次。

HTTP 狀態碼：404

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面
- AWS SDK for .NET
- AWS SDK for C++
- AWS 適用於轉到 V2 的 SDK
- AWS SDK for Java V2 的开发
- AWS 適用於 JavaScript V3 的 SDK
- AWS SDK for PHP
- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

TagResource

服務：Amazon Lex Model Building Service

將指定的標籤新增至指定的資源。如果標籤鍵已存在，則現有值將替換為新值。

請求語法

```
POST /tags/resourceArn HTTP/1.1
Content-type: application/json

{
   "tags": [
      {
         "key": "string",
         "value": "string"
      }
   ]
}
```

URI 請求參數

請求會使用下列 URI 參數。

resourceArn

要標記的機器人、機器人別名或機器人通道的 Amazon 資源名稱 (ARN)。

長度限制：長度下限為 1。最大長度為 1011。

必要：是

請求主體

請求接受採用 JSON 格式的下列資料。

tags

要新增至資源的標籤鍵清單。如果標籤鍵已存在，則現有值將替換為新值。

類型：Tag 物件陣列

陣列成員：項目數下限為 0。項目數上限為 200。

必要：是

回應語法

```
HTTP/1.1 204
```

回應元素

如果動作成功，則服務會送回具有空 HTTP 主體的 HTTP 204 回應。

錯誤

BadRequestException

請求的格式不正確。例如，值無效或缺少必填欄位。請檢查欄位值，然後再試一次。

HTTP 狀態碼：400

ConflictException

處理要求時發生衝突。請再次嘗試您的請求。

HTTP 狀態碼：409

InternalFailureException

發生內部 Amazon Lex 錯誤。請再次嘗試您的請求。

HTTP 狀態碼：500

LimitExceededException

請求超過限制。請再次嘗試您的請求。

HTTP 狀態碼：429

NotFoundException

找不到要求中指定的資源。請檢查資源，然後再試一次。

HTTP 狀態碼：404

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面
- AWS SDK for .NET
- AWS SDK for C++
- AWS 適用於轉到 V2 的 SDK
- AWS SDK for Java V2 的軟件
- AWS 適用於 JavaScript V3 的 SDK
- AWS SDK for PHP
- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

UntagResource

服務：Amazon Lex Model Building Service

從機器人、機器人別名或機器人通道移除標籤。

請求語法

```
DELETE /tags/resourceArn?tagKeys=tagKeys HTTP/1.1
```

URI 請求參數

請求會使用下列 URI 參數。

resourceArn

要從中刪除標籤的資源的 Amazon 資源名稱（ARN）。

長度限制：長度下限為 1。最大長度為 1011。

必要：是

tagKeys

要從資源中移除的標籤鍵清單。如果資源上沒有標籤鍵，則會忽略該鍵。

陣列成員：項目數下限為 0。項目數上限為 200。

長度限制：長度下限為 1。長度上限為 128。

必要：是

請求主體

請求沒有請求主體。

回應語法

```
HTTP/1.1 204
```

回應元素

如果動作成功，則服務會送回具有空 HTTP 主體的 HTTP 204 回應。

錯誤

BadRequestException

請求的格式不正確。例如，值無效或缺少必填欄位。請檢查欄位值，然後再試一次。

HTTP 狀態碼：400

ConflictException

處理要求時發生衝突。請再次嘗試您的請求。

HTTP 狀態碼：409

InternalFailureException

發生內部 Amazon Lex 錯誤。請再次嘗試您的請求。

HTTP 狀態碼：500

LimitExceededException

請求超過限制。請再次嘗試您的請求。

HTTP 狀態碼：429

NotFoundException

找不到要求中指定的資源。請檢查資源，然後再試一次。

HTTP 狀態碼：404

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面
- AWS SDK for .NET
- AWS SDK for C++
- AWS 適用於轉到 V2 的 SDK
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 JavaScript V3 的 SDK
- AWS SDK for PHP

- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

Amazon Lex 運行時間服務

Amazon Lex 運行時間服務支援下列動作：

- DeleteSession
- GetSession
- PostContent
- PostText
- PutSession

DeleteSession

服務：Amazon Lex Runtime Service

移除指定機器人、別名和使用者 ID 的工作階段資訊。

請求語法

```
DELETE /bot/botName/alias/botAlias/user/userId/session HTTP/1.1
```

URI 請求參數

請求會使用下列 URI 參數。

botAlias

包含工作階段資料之機器人使用的別名。

必要：是

botName

包含工作階段資料的機器人名稱。

必要：是

userId

與工作階段資料相關聯的使用者識別碼。

長度約束：最小長度為 2。長度上限為 100。

模式：`[0-9a-zA-Z._:-]+`

必要：是

請求主體

請求沒有請求主體。

回應語法

```
HTTP/1.1 200
Content-type: application/json

{
```

```
    "botAlias": "string",
    "botName": "string",
    "sessionId": "string",
    "userId": "string"
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

botAlias

與工作階段資料相關聯的機器人使用的別名。

類型：字串

botName

與工作階段資料相關聯的機器人名稱。

類型：字串

sessionId

工作階段的唯一識別碼。

類型：字串

userId

用戶端應用程式使用者的 ID。

類型：字串

長度約束：最小長度為 2。長度上限為 100。

模式：`[0-9a-zA-Z._:-]+`

錯誤

BadRequestException

請求驗證失敗，上下文中沒有可用的消息，或者機器人構建失敗，仍在進行中或包含未構建的更改。

HTTP 狀態碼：400

ConflictException

兩個用戶端使用相同的 AWS 帳戶、Amazon Lex 機器人和使用者識別碼。

HTTP 狀態碼：409

InternalFailureException

內部服務錯誤。重試通話。

HTTP 狀態碼：500

LimitExceededException

超過限制。

HTTP 狀態碼：429

NotFoundException

找不到所參考的資源 (例如 Amazon Lex 機器人或別名)。

HTTP 狀態碼：404

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面
- AWS SDK for .NET
- AWS SDK for C++
- AWS 適用於轉到 V2 的 SDK
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 JavaScript V3 的 SDK
- AWS SDK for PHP
- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

GetSession

服務：Amazon Lex Runtime Service

傳回指定機器人、別名和使用者 ID 的工作階段資訊。

請求語法

```
GET /bot/botName/alias/botAlias/user/userId/session/?
checkpointLabelFilter=checkpointLabelFilter HTTP/1.1
```

URI 請求參數

請求會使用下列 URI 參數。

botAlias

包含工作階段資料之機器人使用的別名。

必要：是

botName

包含工作階段資料的機器人名稱。

必要：是

checkpointLabelFilter

用來篩選`recentIntentSummaryView`結構中傳回之比對方式的字串。

當您指定篩選器時，只會傳回其`checkpointLabel`欄位設定為該字串的意圖。

長度限制：長度下限為 1。長度上限為 255。

模式：`[a-zA-Z0-9-]+`

userId

用戶端應用程式使用者的 ID。Amazon Lex 使用此功能來識別使用者與您的機器人的對話。

長度約束：最小長度為 2。長度上限為 100。

模式：`[0-9a-zA-Z._:-]+`

必要：是

請求主體

請求沒有請求主體。

回應語法

```
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"
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

activeContexts

工作階段的使用中前後關聯清單。上下文可以在實現意圖時或通過調用PostContentPostText、或PutSession操作來設置。

您可以使用前後關聯來控制可以跟進意圖的意圖，或修改應用程式的作業。

類型：ActiveContext 物件陣列

陣列成員：項目數下限為 0。項目數上限為 20。

dialogAction

描述機器人目前的狀態。

類型：DialogAction 物件

recentIntentSummaryView

工作階段中使用之意圖的相關資訊陣列。陣列最多可包含三個摘要。如果在工作階段中使用了三個以上的意圖，則recentIntentSummaryView作業會包含有關最後三個使用的意圖的資訊。

如果您在要求中設定checkpointLabelFilter參數，陣列只會包含具有指定標籤的意圖。

類型：IntentSummary 物件陣列

陣列成員：項目數下限為 0。3 個項目的最大數量。

sessionAttributes

表示會話特定上下文信息的鍵/值對的映射。它包含 Amazon Lex 和用戶端應用程式之間傳遞的應用程式資訊。

類型：字串到字串映射

sessionId

工作階段的唯一識別碼。

類型：字串

錯誤

BadRequestException

請求驗證失敗，上下文中沒有可用的消息，或者機器人構建失敗，仍在進行中或包含未構建的更改。

HTTP 狀態碼：400

InternalFailureException

內部服務錯誤。重試通話。

HTTP 狀態碼：500

LimitExceededException

超過限制。

HTTP 狀態碼：429

NotFoundException

找不到所參考的資源 (例如 Amazon Lex 機器人或別名)。

HTTP 狀態碼：404

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面
- AWS SDK for .NET
- AWS SDK for C++
- AWS 適用於轉到 V2 的 SDK

- AWS SDK for Java V2 的開發
- AWS 適用於 JavaScript V3 的 SDK
- AWS SDK for PHP
- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

PostContent

服務：Amazon Lex Runtime Service

將使用者輸入 (文字或語音) 傳送至 Amazon Lex。用戶端使用此 API 在執行階段將文字和音訊請求傳送至 Amazon Lex。Amazon Lex 使用專為機器人建立的機器學習模型來解譯使用者輸入。

該PostContent操作支持 8 千赫和 16 千赫的音頻輸入。您可以使用 8kHz 音訊，在電話音訊應用中達到更高的語音辨識準確度。

作為回應，Amazon Lex 會傳回下一則要傳送給使用者的訊息。請考慮下列範例訊息：

- 對於使用者輸入「我想要披薩」，Amazon Lex 可能會傳回回應，其中包含引出插槽資料的訊息 (例如PizzaSize)：「您想要什麼大小的披薩？」。
- 使用者提供所有披薩訂單資訊之後，Amazon Lex 可能會傳回回覆訊息，以取得使用者確認：「訂購披薩？」。
- 使用者在確認提示回覆「是」之後，Amazon Lex 可能會傳回結論聲明：「謝謝您，您的起司披薩已訂購完畢。」

並非所有 Amazon Lex 訊息都需要使用者回應。例如，結論陳述式不需要回應。有些訊息只需要「是」或「不」回應。除了 Amazon Lex 之外message，Amazon Lex 還在回應中提供有關訊息的其他內容，您可以用來增強用戶端行為，例如顯示適當的用戶端使用者介面。請考量下列範例：

- 如果訊息是要引發位置資料，Amazon Lex 會傳回下列內容資訊：
  - x-amz-lex-dialog-state標頭設定為 ElicitSlot
  - x-amz-lex-intent-name在當前上下文中設置為意圖名稱的標題
  - x-amz-lex-slot-to-elicit標頭設置為引出信息的message插槽名稱
  - x-amz-lex-slots標頭設置為具有當前值的意圖配置的插槽的映射
- 如果訊息是確認提示，則標x-amz-lex-dialog-state頭會設定為，Confirmation且會省略x-amz-lex-slot-to-elicit標頭。
- 如果訊息是針對意圖設定的澄清提示，表示使用者意圖未瞭解，則標x-amz-dialog-state頭會設定為，ElicitIntent且會省略x-amz-slot-to-elicit標頭。

此外，Amazon Lex 也會傳回您的應用程式特定sessionAttributes。如需詳細資訊，請參閱管理交談前後關聯。

請求語法

```
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
```

URI 請求參數

請求會使用下列 URI 參數。

accept

您可以將此值當做 Accept HTTP 標頭傳遞。

Amazon Lex 在回應中傳回的訊息可以是文字或語音，以請求中的 Accept HTTP 標頭值為基礎。

- 如果該值為text/plain; charset=utf-8，Amazon Lex 會在回應中傳回文字。
- 如果值的開頭為audio/，Amazon Lex 會在回應中傳回語音。Amazon Lex 使用 Amazon Polly 產生語音 (使用您在Accept標頭中指定的組態)。例如，如果您指定audio/mpeg為該值，Amazon Lex 會傳回 MPEG 格式的語音。
- 如果值是audio/pcm，則返回的語音是 16 audio/pcm 位，小端格式。
- 以下是可接受的值：
  - 音頻/MPEG
  - 音頻/OGG
  - 音頻/PCM
  - 文本/普通; 字符集 = UTF-8
  - 音頻/*（默認為 MPEG）

activeContexts

請求的使用中前後關聯清單。當滿足以前的意圖時，可以激活上下文，或者通過在請求中包含上下文，

如果您未指定環境清單，Amazon Lex 將使用工作階段的目前環境清單。如果您指定空白清單，則會清除工作階段的所有前後關聯。

botAlias

Amazon Lex 機器人的別名。

必要：是

botName

Amazon Lex 機器人的名稱。

必要：是

contentType

您可以將此值當做 Content-Type HTTP 標頭傳遞。

指示音訊格式或文字。標頭值必須以下列其中一個前置字元開頭：

- PCM 格式，音頻數據必須是小端字節順序。
  - 音頻/L16; 速率 = 16; 頻道 = 1
  - 音頻/X-L16; 採樣速率 = 16; 通道計數 = 1
  - 音頻/流量厘米; 採樣速率 = 8000; = 16; 通道計數 = 1; = 假 sample-size-bits is-big-endian
- 作品格式
  - 音訊/x-cbr-opus-with-起頭訊號；序言大小 =0；位元速率 =256000；=4 frame-size-milliseconds
- 文字格式
  - 文本/普通; 字符集 = UTF-8

必要：是

requestAttributes

您可以將此值當做 x-amz-lex-request-attributes HTTP 標頭傳遞。

Amazon Lex 和用戶端應用程式之間傳遞的請求特定資訊。該值必須是 JSON 序列化和 base64 編碼的對應，其中包含字符串鍵和值。requestAttributes和sessionAttributes標頭的總大小限制為 12 KB。

命名空x-amz-lex:間保留給特殊屬性。不要使用前綴創建任何請求屬性x-amz-lex:。

如需詳細資訊，請參閱設定請求屬性。

sessionAttributes

您可以將此值當做 x-amz-lex-session-attributes HTTP 標頭傳遞。

Amazon Lex 和用戶端應用程式之間傳遞的應用程式特定資訊。該值必須是 JSON 序列化和 base64 編碼的對應，其中包含字符串鍵和值。sessionAttributes和requestAttributes標頭的總大小限制為 12 KB。

如需詳細資訊，請參閱設定工作階段屬性。

userId

用戶端應用程式使用者的 ID。Amazon Lex 使用此功能來識別使用者與您的機器人的對話。在執行階段，每個要求都必須包含userID欄位。

若要決定應用程式使用的使用者 ID，請考量下列因素。

- 該字userID段不得包含用戶的任何個人身份信息，例如姓名，個人識別號碼或其他最終用戶的個人信息。
- 如果您希望使用者在一部裝置上開始對話，並在另一部裝置上繼續，請使用使用者特定的識別碼。
- 如果您希望相同的用戶能夠在兩個不同的設備上進行兩個獨立的對話，請選擇特定於設備的標識符。
- 用戶不能與同一機器人的兩個不同版本進行兩個獨立的對話。例如，使用者無法與同一機器人的 PROD 和 BETA 版本進行對話。如果您預期使用者需要與兩個不同版本進行對話，例如在測試時，請在使用者 ID 中包含機器人別名以分隔這兩個對話。

長度約束：最小長度為 2。長度上限為 100。

模式：[0-9a-zA-Z._:-]+

必要：是

請求主體

請求接受下列二進位資料。

inputStream

使用者輸入 PCM 或 Opus 音訊格式或文字格式，如 Content-Type HTTP 標頭所述。

您可以將音訊資料串流到 Amazon Lex，也可以建立本機緩衝區，以便在傳送之前擷取所有音訊資料。一般來說，如果您串流音訊資料而不是在本機緩衝資料，則會獲得更好的效能。

必要：是

回應語法

```
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
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

回應會傳回下列 HTTP 標頭。

activeContexts

工作階段的使用中前後關聯清單。上下文可以在實現意圖時或通過調用PostContentPostText、或PutSession操作來設置。

您可以使用前後關聯來控制可以跟進意圖的意圖，或修改應用程式的作業。

alternativeIntents

可能適用於用戶意圖的一到四個替代意圖。

每個替代方案都包含一個分數，指出 Amazon Lex 對於意圖符合使用者意圖的信心程度。色彩比對方式會依信賴度分數排序。

botVersion

回應對話的機器人版本。您可以使用此資訊來協助判斷某個機器人的某個版本是否比另一個版本更好。

長度限制：長度下限為 1。長度上限為 64。

模式：[0-9]+|\$LATEST

contentType

要求中 Accept HTTP 標頭中指定的內容類型。

dialogState

識別使用者互動的目前狀態。Amazon Lex 返回以下值之一dialogState。用戶端可以選擇性地使用此資訊來自訂使用者介面。

- ElicitIntent-Amazon Lex 想要引出用戶的意圖。請考量下列範例：

  例如，用戶可能會說出意圖（「我想點一個比薩餅」）。如果 Amazon Lex 無法從這個話語推斷使用者意圖，就會傳回此對話方塊狀態。

- ConfirmIntent-Amazon Lex 期待「是」或「否」的回應。

  例如，Amazon Lex 希望使用者在履行意圖之前確認。使用者可能會回應其他資訊，而不是簡單的「是」或「否」回應。例如，「是的，但要做一個厚皮披薩」或「不，我想點一杯飲料。」 Amazon Lex 可以處理此類額外資訊 (在這些範例中，更新外殼類型插槽或將意圖從變更 OrderPizza 為 OrderDrink)。

- ElicitSlot-Amazon Lex 預期目前意圖的插槽值。

  例如，假設在響應 Amazon Lex 發送此消息：「你想要什麼尺寸的披薩？」。用戶可能會回復插槽值（例如，「中等」）。用戶還可能在響應中提供其他信息（例如，「中等厚度的地殼披薩」）。Amazon Lex 可以適當地處理這類額外資訊。

- Fulfilled-傳達 Lambda 函數已成功實現意圖。
- ReadyForFulfillment-傳達客戶端必須滿足請求。
- Failed-傳達與使用者的對話失敗。

  發生這種情況的原因有多少，包括使用者未對服務的提示提示提供適當的回應 (您可以設定 Amazon Lex 可以提示使用者提供特定資訊的次數)，或 Lambda 函數無法達成意圖。

有效值:ElicitIntent | ConfirmIntent | ElicitSlot | Fulfilled | ReadyForFulfillment | Failed

encodedInputTranscript

用來處理請求的文字。

如果輸入是音訊串流，encodedInputTranscript 欄位會包含從音訊串流擷取的文字。這是實際處理以識別意圖和槽值的文字。您可以使用此資訊判斷 Amazon Lex 是否正確處理您傳送的音訊。

該encodedInputTranscript字段以 64 為基編碼。您必須先解碼欄位，才能使用該值。

encodedMessage

要傳達給使用者的訊息。訊息可以來自機器人的組態或來自 Lambda 函數。

如果未使用 Lambda 函數設定意圖，或 Lambda 函數dialogAction.type在其回應中傳回Delegate為，Amazon Lex 會決定下一個動作過程，並根據目前的互動內容從機器人的組態中選取適當的訊息。例如，如果 Amazon Lex 無法理解使用者輸入，則會使用澄清提示訊息。

當您建立意圖時，您可以將訊息指派給群組。將訊息指派給群組時，Amazon Lex 會從回應中的每個群組傳回一則訊息。消息字段是包含消息的轉義 JSON 字符串。如需有關傳回之 JSON 字串結構的詳細資訊，請參閱支援的訊息格式。

如果 Lambda 函數傳回訊息，Amazon Lex 會在回應中將訊息傳遞給用戶端。

該encodedMessage字段以 64 為基編碼。您必須先解碼欄位，才能使用該值。

長度限制：長度下限為 1。最大長度為 1366。

inputTranscript

此標頭已被棄用。

用來處理請求的文字。

您只能在德、恩-歐、EN-GB、EN-US、ES-419、ES-ES、ES-US、FR-CA、FR-FR 和 IT IT 語言環境中使用此欄位。在所有其他地區設定中，此inputTranscript欄位為空值。您應該改用該encodedInputTranscript字段。

如果輸入是音訊串流，inputTranscript 欄位會包含從音訊串流擷取的文字。這是實際處理以識別意圖和槽值的文字。您可以使用此資訊判斷 Amazon Lex 是否正確處理您傳送的音訊。

intentName

Amazon Lex 所知道的目前使用者意圖。

message

此標頭已被棄用。

您只能在德、EN-AU、EN-GB、EN-US、ES-419、ES-ES、ES-US、FR-CA、FR-FR 和 IT IT 地區設定中使用此欄位。在所有其他地區設定中，此message欄位為空值。您應該改用該encodedMessage字段。

要傳達給使用者的訊息。訊息可以來自機器人的組態或來自 Lambda 函數。

如果未使用 Lambda 函數設定意圖，或 Lambda 函數dialogAction.type在其回應中傳回Delegate為，Amazon Lex 會決定下一個動作過程，並根據目前的互動內容從機器人的組態中選取適當的訊息。例如，如果 Amazon Lex 無法理解使用者輸入，則會使用澄清提示訊息。

當您建立意圖時，您可以將訊息指派給群組。將訊息指派給群組時，Amazon Lex 會從回應中的每個群組傳回一則訊息。消息字段是包含消息的轉義 JSON 字符串。如需有關傳回之 JSON 字串結構的詳細資訊，請參閱支援的訊息格式。

如果 Lambda 函數傳回訊息，Amazon Lex 會在回應中將訊息傳遞給用戶端。

長度限制：長度下限為 1。長度上限為 1024。

messageFormat

回應訊息的格式。下列其中一值：

- PlainText-郵件包含純 UTF-8 文字。
- CustomPayload-訊息是用戶端的自訂格式。
- SSML-訊息包含格式化為語音輸出的文字。
- Composite-訊息包含逸出的 JSON 物件，其中包含一或多個來自建立意圖時所指派訊息之群組的訊息。

有效值:PlainText | CustomPayload | SSML | Composite

nluIntentConfidence

提供一個分數，指出 Amazon Lex 對於傳回的意圖是符合使用者意圖的信心程度。該分數在 0.0 和 1.0 之間。

分數是相對分數，而不是絕對分數。分數可能會根據 Amazon Lex 的改進而改變。

sentimentResponse

在話語中表達的情緒。

當機器人設定為將語音傳送至 Amazon Comprehend 進行情緒分析時，此欄位會包含分析結果。

sessionAttributes

表示會話特定上下文信息的鍵/值對的映射。

sessionId

工作階段的唯一識別碼。

slots

在交談期間從使用者輸入中偵測到零個或多 Amazon Lex 意圖插槽 (名稱/值對) 的地圖。該字段以 64 為基編碼。

Amazon Lex 會建立一個解析度清單，其中包含插槽的可能值。它傳回的值由建立或更新插槽類型時`valueSelectionStrategy`所選取的值決定。如果設定`valueSelectionStrategy`為`ORIGINAL_VALUE`，則傳回使用者提供的值 (如果使用者值與槽值類似)。如果設定`valueSelectionStrategy`為 `TOP_RESOLUTION` Amazon Lex，則會傳回解析清單中的第一個值，如果沒有解析度清單，則傳回 null。如果未指定`valueSelectionStrategy`，則預設值為`ORIGINAL_VALUE`。

slotToElicit

如果`dialogState`值為`ElicitSlot`，則傳回 Amazon Lex 引出值的插槽名稱。

回應傳回以下內容作為 HTTP 主體。

audioStream

要傳達給使用者的提示 (或陳述式)。這是基於機器人配置和上下文。例如，如果 Amazon Lex 不瞭解使用者意圖，就會傳送針對機器人`clarificationPrompt`設定的設定。如果意圖在執行履行作業之前需要確認，則會傳送`confirmationPrompt`。另一個範例：假設 Lambda 函數成功完成了意圖，並傳送訊息傳送給使用者。然後 Amazon Lex 在響應中發送該消息。

錯誤

BadGatewayException

Amazon Lex 機器人仍在建置中，或其中一個相依服務 (Amazon Polly、AWS Lambda) 失敗，並出現內部服務錯誤。

HTTP 狀態碼：502

BadRequestException

請求驗證失敗，上下文中沒有可用的消息，或者機器人構建失敗，仍在進行中或包含未構建的更改。

HTTP 狀態碼：400

ConflictException

兩個用戶端使用相同的 AWS 帳戶、Amazon Lex 機器人和使用者識別碼。

HTTP 狀態碼：409

DependencyFailedException

其中一個依賴關係，如 AWS Lambda 或 Amazon Polly，拋出了一個異常。例如

- 如果 Amazon Lex 沒有足夠的許可來呼叫 Lambda 函數。
- 如果一個 Lambda 函數需要超過 30 秒的時間來執行。
- 如果履行 Lambda 函數在未移除任何插槽值的情況下傳回`Delegate`對話方塊動作。

狀態碼：

InternalFailureException

內部服務錯誤。重試通話。

HTTP 狀態碼：500

LimitExceededException

超過限制。

HTTP 狀態碼：429

LoopDetectedException

未使用此例外狀況。

狀態碼：

NotAcceptableException

請求中的接受標頭沒有有效的值。

狀態碼：

NotFoundException

找不到所參考的資源 (例如 Amazon Lex 機器人或別名)。

HTTP 狀態碼：404

RequestTimeoutException

輸入語音太長。

HTTP 狀態碼：408

UnsupportedMediaTypeException

內容類型標頭 (PostContentAPI) 具有無效的值。

HTTP 狀態碼：415

範例

範例 1

在此要求中，URI 會識別機器人 (流量)、機器人版本 ($LATEST) 和一般使用者名稱 (某個使用者)。標Content-Type頭標識正文中音頻的格式。Amazon Lex 也支持其他格式。要將音頻從一種格式轉換為另一種格式，如有必要，可以使用 SoX 開源軟件。您可以透過新增 Accept HTTP 標頭來指定要取得回應的格式。

在回應中，標x-amz-lex-message頭會顯示 Amazon Lex 傳回的回應。然後，客戶端可以將此響應發送給用戶。通過分塊編碼（根據要求）以音頻/MPEG 格式發送相同的消息。

請求範例

```
"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]"
…
```

回應範例

```
"HTTP/1.1 200 OK[\r][\n]"
"x-amzn-RequestId: cc8b34af-cebb-11e6-a35c-55f3a992f28d[\r][\n]"
"x-amz-lex-message: Sorry, can you repeat that?[\r][\n]"
"x-amz-lex-dialog-state: ElicitIntent[\r][\n]"
"x-amz-lex-session-attributes: eyJ1c2VyTmFtZSI6IkJvYiJ9[\r][\n]"
"Content-Type: audio/mpeg[\r][\n]"
"Transfer-Encoding: chunked[\r][\n]"
"Date: Fri, 30 Dec 2016 18:14:28 GMT[\r][\n]"
"[\r][\n]"
"2000[\r][\n]"
"ID3[0x4][0x0][0x0][0x0][0x0][0x0]#TSSE[0x0][0x0][0x0][0xf][0x0][0x0]
[0x3]Lavf57.41.100[0x0][0x0][0x0][0x0][0x0][0x0][0x0][0x0][0x0][0x0][0x0][0xff]
[0xf3]`[0xc4][0x0][0x1b]{[0x8d][0xe8][0x1]C[0x18][0x1][0x0]J[0xe0]`b[0xdd][0xd1]
[0xb][0xfd][0x11][0xdf][0xfe]";[0xbb][0xbb][0x9f][0xee][0xee][0xee][0xee]|DDD/[0xff]
[0xff][0xff][0xff]www?D[0xf7]w^?[0xff][0xfa]h[0x88][0x85][0xfe][0x88][0x88][0x88]
[[0xa2]'[0xff][0xfa]"{[0x9f][0xe8][0x88]]D[0xeb][0xbb][0xbb][0xa2]!u[0xfd][0xdd][0xdf]
[0x88][0x94][0x0]F[0xef][0xa1]8[0x0][0x82]w[0x88]N[0x0][0x0][0x9b][0xbb][0xe8][0xe
…
```

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面
- AWS SDK for .NET
- AWS SDK for C++

- AWS 適用於轉到 V2 的 SDK
- AWS SDK for Java V2 的开发
- AWS 適用於 JavaScript V3 的 SDK
- AWS SDK for PHP
- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

PostText

服務：Amazon Lex Runtime Service

傳送使用者輸入到 Amazon Lex。用戶端應用程式可以使用此 API 在執行時間傳送請求至 Amazon Lex。然後，Amazon Lex 會使用為機器人建立的機器學習模型來解譯使用者輸入。

作為回應，Amazon Lex 返回下一個傳遞`message`給用戶一個可選`responseCard`的顯示。請考慮下列範例訊息：

- 對於用戶輸入「我想吃披薩」，Amazon Lex 可能會返回帶有引出插槽數據的消息的響應（例如，PizzaSize）：「您想要什麼尺寸的披薩？」
- 使用者提供所有披薩訂單資訊後，Amazon Lex 可能會傳回回應，並附上訊息，以取得使用者確認「繼續進行披薩訂單？」。
- 使用者以「是」回覆確認提示後，Amazon Lex 可能會傳回結論陳述：「謝謝您，您的起司披薩已訂購完畢。」

並非所有 Amazon Lex 訊息都需要使用者回應。例如，結論陳述式不需要回應。有些訊息只需要「是」或「否」的使用者回應。除了 Amazon Lex 之外`message`，Amazon Lex 還在回應中提供有關訊息的其他內容，您可能會用來增強用戶端行為，例如顯示適當的用戶端使用者介面。這些是響應中的`slotToElicit`、`dialogState`、`intentName`、和`slots`字段。請考量下列範例：

- 如果訊息是要引發位置資料，Amazon Lex 會傳回下列內容資訊：
  - `dialogState`設定為 ElicitSlot
  - `intentName`設定為目前前後關聯中的意圖名稱
  - `slotToElicit`設定為引出資訊的`message`插槽名稱
  - `slots`設置為插槽的映射，為意圖配置，具有當前已知值
- 如果訊息是確認提示，`dialogState`則會將設定`SlotToElicit`為 ConfirmIntent 且設定為 null。
- 如果訊息是澄清提示 (針對意圖設定)，指出使用者意圖未瞭解，`dialogState`則會將設定`slotToElicit`為 ElicitIntent 且設定為 null。

此外，Amazon Lex 也會傳回您的應用程式特定`sessionAttributes`。如需詳細資訊，請參閱管理交談前後關聯。

請求語法

```
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"
   }
}
```

URI 請求參數

請求會使用下列 URI 參數。

botAlias

Amazon Lex 機器人的別名。

必要：是

botName

Amazon Lex 機器人的名稱。

必要：是

userId

用戶端應用程式使用者的 ID。Amazon Lex 使用此功能來識別使用者與您的機器人的對話。在執行階段，每個要求都必須包含`userID`欄位。

若要決定應用程式使用的使用者 ID，請考量下列因素。

- 該字userID段不得包含用戶的任何個人身份信息，例如姓名，個人識別號碼或其他最終用戶的個人信息。
- 如果您希望使用者在一部裝置上開始對話，並在另一部裝置上繼續，請使用使用者特定的識別碼。
- 如果您希望相同的用戶能夠在兩個不同的設備上進行兩個獨立的對話，請選擇特定於設備的標識符。
- 用戶不能與同一機器人的兩個不同版本進行兩個獨立的對話。例如，使用者無法與同一機器人的 PROD 和 BETA 版本進行對話。如果您預期使用者需要與兩個不同版本進行對話，例如在測試時，請在使用者 ID 中包含機器人別名以分隔這兩個對話。

長度約束：最小長度為 2。長度上限為 100。

模式：[0-9a-zA-Z._:-]+

必要：是

請求主體

請求接受採用 JSON 格式的下列資料。

activeContexts

請求的使用中前後關聯清單。當滿足以前的意圖時，可以激活上下文，或者通過在請求中包含上下文，

如果您未指定環境清單，Amazon Lex 將使用工作階段的目前環境清單。如果您指定空白清單，則會清除工作階段的所有前後關聯。

類型：ActiveContext 物件陣列

陣列成員：項目數下限為 0。項目數上限為 20。

必要：否

inputText

使用者輸入的文字 (Amazon Lex 解譯此文字)。

當您使用 AWS CLI 時，您無法在--input-text參數中傳遞 URL。請改用--cli-input-json參數傳遞 URL。

類型：字串

長度限制：長度下限為 1。長度上限為 1024。

必要：是

requestAttributes

Amazon Lex 和用戶端應用程式之間傳遞的請求特定資訊。

命名空x-amz-lex:間保留給特殊屬性。請勿使用前綴創建任何請求屬性x-amz-lex:。

如需詳細資訊，請參閱設定請求屬性。

類型：字串到字串映射

必要：否

sessionAttributes

Amazon Lex 和用戶端應用程式之間傳遞的應用程式特定資訊。

如需詳細資訊，請參閱設定工作階段屬性。

類型：字串到字串映射

必要：否

回應語法

```
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"
}
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

服務會傳回下列 JSON 格式的資料。

activeContexts

工作階段的使用中前後關聯清單。上下文可以在實現意圖時或通過調用PostContentPostText、或PutSession操作來設置。

您可以使用前後關聯來控制可以跟進意圖的意圖，或修改應用程式的作業。

類型：ActiveContext 物件陣列

陣列成員：項目數下限為 0。項目數上限為 20。

alternativeIntents

可能適用於用戶意圖的一到四個替代意圖。

每個替代方案都包含一個分數，指出 Amazon Lex 對於意圖符合使用者意圖的信心程度。比對方式會依可信度分數排序。

類型：PredictedIntent 物件陣列

陣列成員：4 個項目的上限。

botVersion

回應對話的機器人版本。您可以使用此資訊來協助判斷某個機器人的某個版本是否比另一個版本更好。

類型：字串

長度限制：長度下限為 1。長度上限為 64。

模式：[0-9]+|\$LATEST

dialogState

識別使用者互動的目前狀態。Amazon Lex 返回以下值之一dialogState。用戶端可以選擇性地使用此資訊來自訂使用者介面。

- ElicitIntent-Amazon Lex 想要引起用戶的意圖。

  例如，用戶可能會說出意圖（「我想點披薩」）。如果 Amazon Lex 無法從這個話語推斷出使用者意圖，就會傳回此對話方塊狀態。

- ConfirmIntent-Amazon Lex 期待「是」或「否」的回應。

  例如，Amazon Lex 希望使用者在履行意圖之前確認。

  使用者可能會回應其他資訊，而不是簡單的「是」或「否」。例如，「是的，但要做厚皮披薩」或「不，我想點一杯飲料」。Amazon Lex 可以處理此類額外資訊 (在這些範例中，將外殼類型插槽值更新，或將意圖從變更 OrderPizza 為 OrderDrink)。

- ElicitSlot-Amazon Lex 預期目前的意圖會有插槽值。

  例如，假設在響應 Amazon Lex 發送此消息：「你想要什麼尺寸的披薩？」。用戶可能會回复插槽值（例如，「中等」）。用戶還可能在響應中提供其他信息（例如，「中厚地殼披薩」）。Amazon Lex 可以適當地處理這類額外資訊。

- Fulfilled-傳達針對意圖設定的 Lambda 函數已成功完成意圖。
- ReadyForFulfillment-傳達客戶端必須履行的意圖。
- Failed-傳達與使用者的對話失敗。

  發生這種情況的原因有多少，包括使用者未對服務提示提供適當的回應 (您可以設定 Amazon Lex 可以提示使用者提供特定資訊的次數)，或 Lambda 函數無法達成意圖。

類型：字串

有效值:ElicitIntent | ConfirmIntent | ElicitSlot | Fulfilled | ReadyForFulfillment | Failed

intentName

目前使用者意圖 Amazon Lex 知道。

類型：字串

message

要傳達給使用者的訊息。訊息可以來自機器人的組態或來自 Lambda 函數。

如果未使用 Lambda 函數設定意圖，或是 Lambda 函數傳回`Delegate`為`dialogAction.type`其回應，Amazon Lex 會決定下一個動作過程，並根據目前的互動內容從機器人的組態中選取適當的訊息。例如，如果 Amazon Lex 無法理解使用者輸入，則會使用澄清提示訊息。

當您建立意圖時，您可以將訊息指派給群組。將訊息指派給群組時，Amazon Lex 會從回應中的每個群組傳回一則訊息。消息字段是包含消息的轉義 JSON 字符串。如需有關傳回之 JSON 字串結構的詳細資訊，請參閱支援的訊息格式。

如果 Lambda 函數傳回訊息，Amazon Lex 會在回應中將訊息傳遞給用戶端。

類型：字串

長度限制：長度下限為 1。長度上限為 1024。

messageFormat

回應訊息的格式。下列其中一值：

- `PlainText`-郵件包含純 UTF-8 文字。
- `CustomPayload`-訊息是 Lambda 函數所定義的自訂格式。
- `SSML`-訊息包含格式化為語音輸出的文字。
- `Composite`-訊息包含逸出的 JSON 物件，其中包含一或多個來自建立意圖時所指派訊息之群組的訊息。

類型：字串

有效值:`PlainText | CustomPayload | SSML | Composite`

nluIntentConfidence

提供一個分數，指出 Amazon Lex 對於傳回的意圖是符合使用者意圖的信心程度。該分數在 0.0 和 1.0 之間。如需詳細資訊，請參閱可信度分數。

分數是相對分數，而不是絕對分數。分數可能會根據 Amazon Lex 的改進而改變。

類型：IntentConfidence 物件

responseCard

表示使用者必須回應目前提示的選項。回應卡可以來自機器人組態 (在 Amazon Lex 主控台中，選擇插槽旁的設定按鈕) 或程式碼掛接 (Lambda 函數)。

類型：ResponseCard 物件

sentimentResponse

表達的情緒和說話。

當機器人設定為將語音傳送至 Amazon Comprehend 進行情緒分析時，此欄位會包含分析結果。

類型：SentimentResponse 物件

sessionAttributes

代表會話特定上下文信息的鍵-值對的映射。

類型：字串到字串映射

sessionId

工作階段的唯一識別碼。

類型：字串

slots

Amazon Lex 從交談中的使用者輸入中偵測到的意圖插槽。

Amazon Lex 會建立一個解析度清單，其中包含插槽的可能值。它傳回的值由建立或更新插槽類型時valueSelectionStrategy所選取的值決定。如果設定valueSelectionStrategy為ORIGINAL_VALUE，則傳回使用者提供的值 (如果使用者值與槽值類似)。如果設定valueSelectionStrategy為 TOP_RESOLUTION Amazon Lex，則會傳回解析清單中的第一個值，如果沒有解析度清單，則傳回 null。如果未指定valueSelectionStrategy，則預設值為ORIGINAL_VALUE。

類型：字串到字串映射

slotToElicit

如果dialogState值為ElicitSlot，則傳回 Amazon Lex 引出值的插槽名稱。

類型：字串

錯誤

BadGatewayException

Amazon Lex 機器人仍在建置中，或其中一個相依服務 (Amazon Polly、AWS Lambda) 失敗，並出現內部服務錯誤。

HTTP 狀態碼：502

BadRequestException

請求驗證失敗，上下文中沒有可用的消息，或者機器人構建失敗，仍在進行中或包含未構建的更改。

HTTP 狀態碼：400

ConflictException

兩個用戶端使用相同的 AWS 帳戶、Amazon Lex 機器人和使用者識別碼。

HTTP 狀態碼：409

DependencyFailedException

其中一個依賴關係，如 AWS Lambda 或 Amazon Polly，拋出了一個異常。例如

- 如果 Amazon Lex 沒有足夠的許可來呼叫 Lambda 函數。
- 如果一個 Lambda 函數需要超過 30 秒的時間來執行。
- 如果履行 Lambda 函數傳回`Delegate`對話方塊動作，但不移除任何插槽值。

狀態碼：

InternalFailureException

內部服務錯誤。重試通話。

HTTP 狀態碼：500

LimitExceededException

超過限制。

HTTP 狀態碼：429

LoopDetectedException

未使用此例外狀況。

狀態碼：

NotFoundException

找不到所參考的資源 (例如 Amazon Lex 機器人或別名)。

HTTP 狀態碼：404

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面
- AWS SDK for .NET
- AWS SDK for C++
- AWS 適用於轉到 V2 的 SDK
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 JavaScript V3 的 SDK
- AWS SDK for PHP
- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

PutSession

服務：Amazon Lex Runtime Service

使用 Amazon Lex 機器人建立一個新的工作階段或修改現有的工作階段。使用此作業可讓您的應用程式設定機器人的狀態。

如需詳細資訊，請參閱管理工作階段。

請求語法

```
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"
  }
}
```

URI 請求參數

請求會使用下列 URI 參數。

accept

Amazon Lex 在回應中傳回的訊息可以是文字或語音，視此欄位的值而定。

- 如果該值為text/plain; charset=utf-8，Amazon Lex 會在回應中傳回文字。
- 如果值的開頭為audio/，Amazon Lex 會在回應中傳回語音。Amazon Lex 使用 Amazon Polly 在您指定的組態中產生語音。例如，如果您指定audio/mpeg為該值，Amazon Lex 會傳回 MPEG 格式的語音。
- 如果該值是audio/pcm，則語音返回為 16 audio/pcm 位，小端格式。
- 以下是可接受的值：
  - audio/mpeg
  - audio/ogg
  - audio/pcm
  - audio/*（默認為 MPEG）
  - text/plain; charset=utf-8

botAlias

包含工作階段資料之機器人使用的別名。

必要：是

botName

包含工作階段資料的機器人名稱。

必要：是

userId

用戶端應用程式使用者的識別碼。Amazon Lex 使用此功能來識別使用者與您的機器人的對話。

長度約束：最小長度為 2。長度上限為 100。

模式：[0-9a-zA-Z._:-]+

必要：是

請求主體

請求接受採用 JSON 格式的下列資料。

activeContexts

請求的使用中前後關聯清單。當滿足以前的意圖時，可以激活上下文，或者通過在請求中包含上下文，

如果您未指定環境清單，Amazon Lex 將使用工作階段的目前環境清單。如果您指定空白清單，則會清除工作階段的所有前後關聯。

類型：ActiveContext 物件陣列

陣列成員：項目數下限為 0。項目數上限為 20。

必要：否

dialogAction

設定機器人完成對話時應採取的下一個動作。

類型：DialogAction 物件

必要：否

recentIntentSummaryView

機器人最近意圖的摘要。您可以使用意圖摘要檢視來設定意圖上的檢查點標籤，並修改意圖的屬性。您也可以使用它來移除意圖摘要物件，或將意圖摘要物件新增至清單。

您修改或新增至清單的意圖對於機器人來說必須有意義。例如，意圖名稱必須對機器人有效。您必須提供下列項目的有效值：

- intentName
- 插槽名稱
- slotToElict

如果您在要PutSession求中傳送recentIntentSummaryView參數，則新摘要檢視的內容會取代舊的摘要檢視。例如，如果GetSession請求在摘要視圖中返回三個意圖，並且您在摘要視圖中以一個意圖調PutSession用，則下一次調用GetSession將只返回一個意圖。

類型：IntentSummary 物件陣列

陣列成員：項目數下限為 0。3 個項目的最大數量。

必要：否

sessionAttributes

表示會話特定上下文信息的鍵/值對的映射。它包含 Amazon Lex 和用戶端應用程式之間傳遞的應用程式資訊。

類型：字串到字串映射

必要：否

回應語法

```
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
```

回應元素

如果動作成功，則服務傳回 HTTP 200 回應。

回應會傳回下列 HTTP 標頭。

activeContexts

工作階段的使用中前後關聯清單。

contentType

要求中 Accept HTTP 標頭中指定的內容類型。

dialogState

- ConfirmIntent-Amazon Lex 預期「是」或「否」回應，以便在完成意圖之前確認意圖。
- ElicitIntent-Amazon Lex 想要引出用戶的意圖。
- ElicitSlot-Amazon Lex 預期目前意圖的插槽值。
- Failed-傳達與使用者的對話失敗。發生這種情況的原因有多種，包括使用者未針對服務的提示提示提供適當的回應，或是 Lambda 函數無法達成意圖。
- Fulfilled-傳達 Lambda 函數已成功實現意圖。
- ReadyForFulfillment-傳達客戶端必須履行的意圖。

有效值:ElicitIntent | ConfirmIntent | ElicitSlot | Fulfilled | ReadyForFulfillment | Failed

encodedMessage

應該向用戶顯示的下一條消息。

該encodedMessage字段以 64 為基編碼。您必須先解碼欄位，才能使用該值。

長度限制：長度下限為 1。最大長度為 1366。

intentName

目前意圖的名稱。

message

此標頭已被棄用。

應該向用戶顯示的下一條消息。

您只能在德、恩-歐、EN-GB、EN-US、ES-419、ES-ES、ES-US、FR-CA、FR-FR 和 IT IT 語言環境中使用此欄位。在所有其他地區設定中，此message欄位為空值。您應該改用該encodedMessage字段。

長度限制：長度下限為 1。長度上限為 1024。

messageFormat

回應訊息的格式。下列其中一值：

- PlainText-郵件包含純 UTF-8 文字。
- CustomPayload-訊息是用戶端的自訂格式。
- SSML-訊息包含格式化為語音輸出的文字。
- Composite-訊息包含逸出的 JSON 物件，其中包含一或多個來自建立意圖時所指派訊息之群組的訊息。

有效值:PlainText | CustomPayload | SSML | Composite

sessionAttributes

表示會話特定上下文信息的鍵/值對的映射。

sessionId

工作階段的唯一識別碼。

slots

在交談期間從使用者輸入中偵 Amazon Lex 到零個或多個意圖插槽的地圖。

Amazon Lex 會建立一個解析度清單，其中包含插槽的可能值。它傳回的值由建立或更新插槽類型時valueSelectionStrategy所選取的值決定。如果設定valueSelectionStrategy為ORIGINAL_VALUE，則傳回使用者提供的值 (如果使用者值與槽值類似)。如果設定valueSelectionStrategy為 TOP_RESOLUTION Amazon Lex，則會傳回解析清單中的第一個值，如果沒有解析度清單，則傳回 null。如果您未指定預valueSelectionStrategy設值為ORIGINAL_VALUE。

slotToElicit

如果dialogState是ElicitSlot，則返回 Amazon Lex 引出值的插槽名稱。

回應傳回以下內容作為 HTTP 主體。

audioStream

要傳達給使用者的訊息的音訊版本。

錯誤

BadGatewayException

Amazon Lex 機器人仍在建置中，或其中一個相依服務 (Amazon Polly、AWS Lambda) 失敗，並顯示內部服務錯誤。

HTTP 狀態碼：502

BadRequestException

請求驗證失敗，上下文中沒有可用的消息，或者機器人構建失敗，仍在進行中或包含未構建的更改。

HTTP 狀態碼：400

ConflictException

兩個用戶端使用相同的 AWS 帳戶、Amazon Lex 機器人和使用者識別碼。

HTTP 狀態碼：409

DependencyFailedException

其中一個依賴關係，如 AWS Lambda 或 Amazon Polly，拋出了一個異常。例如

- 如果 Amazon Lex 沒有足夠的許可來呼叫 Lambda 函數。
- 如果一個 Lambda 函數需要超過 30 秒的時間來執行。
- 如果履行 Lambda 函數在未移除任何插槽值的情況下傳回`Delegate`對話方塊動作。

狀態碼：

InternalFailureException

內部服務錯誤。重試通話。

HTTP 狀態碼：500

LimitExceededException

超過限制。

HTTP 狀態碼：429

NotAcceptableException

請求中的接受標頭沒有有效的值。

狀態碼：

NotFoundException

找不到所參考的資源 (例如 Amazon Lex 機器人或別名)。

HTTP 狀態碼：404

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列介面
- AWS SDK for .NET
- AWS SDK for C++
- AWS 適用於轉到 V2 的 SDK
- AWS SDK for Java V2 的开发
- AWS 適用於 JavaScript V3 的 SDK
- AWS SDK for PHP
- AWS 適用於 Python 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

資料類型

Amazon Lex 模型構建服務支援下列資料類型：

- BotAliasMetadata
- BotChannelAssociation
- BotMetadata
- BuiltinIntentMetadata
- BuiltinIntentSlot

- BuiltinSlotTypeMetadata
- CodeHook
- ConversationLogsRequest
- ConversationLogsResponse
- EnumerationValue
- FollowUpPrompt
- FulfillmentActivity
- InputContext
- Intent
- IntentMetadata
- KendraConfiguration
- LogSettingsRequest
- LogSettingsResponse
- Message
- MigrationAlert
- MigrationSummary
- OutputContext
- Prompt
- ResourceReference
- Slot
- SlotDefaultValue
- SlotDefaultValueSpec
- SlotTypeConfiguration
- SlotTypeMetadata
- SlotTypeRegexConfiguration
- Statement
- Tag
- UtteranceData
- UtteranceList

Amazon Lex 運行時服務支援下列資料類型：

- ActiveContext
- ActiveContextTimeToLive
- Button
- DialogAction
- GenericAttachment
- IntentConfidence
- IntentSummary
- PredictedIntent
- ResponseCard
- SentimentResponse

Amazon Lex 模型構建服務

Amazon Lex 模型構建服務支援下列資料類型：

- 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

服務：Amazon Lex Model Building Service

提供機器人別名的相關資訊。

目錄

botName

別名所指向的機器人的名稱。

類型：字串

長度約束：最小長度為 2。長度上限為 50。

模式：`^([A-Za-z]_?)+$`

必要：否

botVersion

別名所指向的 Amazon Lex 機器人版本。

類型：字串

長度限制：長度下限為 1。長度上限為 64。

模式：`\$LATEST|[0-9]+`

必要：否

checksum

機器人別名的總和檢查碼。

類型：字串

必要：否

conversationLogs

決定 Amazon Lex 如何將交談日誌用於別名的設定。

類型：ConversationLogsResponse 物件

必要：否

createdDate

建立機器人別名的日期。

類型：Timestamp

必要：否

description

機器人別名的說明。

類型：字串

長度限制：長度下限為 0。長度上限為 200。

必要：否

lastUpdatedDate

更新機器人別名的日期。建立資源時，建立日期和上次更新日期相同。

類型：Timestamp

必要：否

name

機器人別名的名稱。

類型：字串

長度限制：長度下限為 1。長度上限為 100。

模式：`^([A-Za-z]_?)+$`

必要：否

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS 適用於 Java V2 的 SDK

- AWS 適用於紅寶石 V3 的 SDK

BotChannelAssociation

服務：Amazon Lex Model Building Service

代表 Amazon Lex 機器人與外部簡訊平台之間的關聯。

目錄

botAlias

指向建立此關聯之特定 Amazon Lex 機器人版本的別名。

類型：字串

長度限制：長度下限為 1。長度上限為 100。

模式：`^([A-Za-z]_?)+$`

必要：否

botConfiguration

提供與訊息平台通訊所需的資訊。

類型：字串到字串映射

地圖項目：最多 10 個項目。

必要：否

botName

正在建立此關聯之 Amazon Lex 機器人的名稱。

> **Note**
>
> 目前，Amazon Lex 支持與臉書和鬆弛和 Twilio 的關聯。

類型：字串

長度約束：最小長度為 2。長度上限為 50。

模式：`^([A-Za-z]_?)+$`

必要：否

createdDate

建立 Amazon Lex 機器人與通道之間關聯的日期。

類型：Timestamp

必要：否

description

您正在建立之關聯的文字描述。

類型：字串

長度限制：長度下限為 0。長度上限為 200。

必要：否

failureReason

如果`status`是`FAILED`，Amazon Lex 會提供無法建立關聯的原因。

類型：字串

必要：否

name

機器人與通道之間的關聯名稱。

類型：字串

長度限制：長度下限為 1。長度上限為 100。

模式：`^([A-Za-z]_?)+$`

必要：否

status

機器人通道的狀態。

- `CREATED`-該頻道已創建並準備使用。
- `IN_PROGRESS`-頻道創建正在進行中。
- `FAILED`-建立頻道時發生錯誤。如需失敗原因的相關資訊，請參閱`failureReason`欄位。

類型：字串

有效值:`IN_PROGRESS | CREATED | FAILED`

必要：否

type

透過指示 Amazon Lex 機器人與外部簡訊平台之間建立的通道類型來指定關聯類型。

類型：字串

有效值:`Facebook | Slack | Twilio-Sms | Kik`

必要：否

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS 適用於 Java V2 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

BotMetadata

服務：Amazon Lex Model Building Service

提供有關機器人的資訊。

目錄

createdDate

建立機器人的日期。

類型：Timestamp

必要：否

description

機器人的描述。

類型：字串

長度限制：長度下限為 0。長度上限為 200。

必要：否

lastUpdatedDate

機器人更新的日期。當您建立機器人時，建立日期和上次更新日期相同。

類型：Timestamp

必要：否

name

機器人的名稱。

類型：字串

長度約束：最小長度為 2。長度上限為 50。

模式：`^([A-Za-z]_?)+$`

必要：否

status

機器人的狀態。

類型：字串

有效值:BUILDING | READY | READY_BASIC_TESTING | FAILED | NOT_BUILT

必要：否

version

機器人的版本。對於新的機器人，版本始終是$LATEST。

類型：字串

長度限制：長度下限為 1。長度上限為 64。

模式：\$LATEST|[0-9]+

必要：否

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS SDK for Java V2 的軟件
- AWS 適用於紅寶石 V3 的 SDK

BuiltinIntentMetadata

服務：Amazon Lex Model Building Service

提供內建意圖的中繼資料。

目錄

signature

內置意圖的唯一標識符。若要尋找意圖的簽名，請參閱 Alexa 技能套件中的標準內建意圖。

類型：字串

必要：否

supportedLocales

意圖支援之地區設定的識別碼清單。

類型：字串陣列

有效值:`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`

必要：否

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS SDK for Java V2 的軟件
- AWS 適用於紅寶石 V3 的 SDK

BuiltinIntentSlot

服務：Amazon Lex Model Building Service

提供內建意圖中使用之插槽的相關資訊。

目錄

name

為意圖定義的狹槽清單。

類型：字串

必要：否

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS 適用於 Java V2 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

BuiltinSlotTypeMetadata

服務：Amazon Lex Model Building Service

提供有關內建插槽類型的資訊。

目錄

signature

內建插槽類型的唯一識別碼。若要尋找插槽類型的簽名，請參閱 Alexa 技能套件中的插槽類型參考。

類型：字串

必要：否

supportedLocales

插槽的目標語言環境清單。

類型：字串陣列

有效值:`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`

必要：否

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS SDK for Java V2 的開發
- AWS 適用於紅寶石 V3 的 SDK

CodeHook

服務：Amazon Lex Model Building Service

指定 Lambda 函數，以驗證對機器人的請求或滿足使用者對機器人的請求。

目錄

messageVersion

您希望 Amazon Lex 用來叫用 Lambda 函數的請求-回應版本。如需詳細資訊，請參閱 使用 Lambda 函數。

類型：字串

長度限制：長度下限為 1。最大長度為 5。

必要：是

uri

Lambda 函數的 Amazon Resource Name (ARN)。

類型：字串

長度限制：長度下限為 20。長度上限為 2048。

模式：`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-_]+)?`

必要：是

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS SDK for Java V2 的軟件
- AWS 適用於紅寶石 V3 的 SDK

ConversationLogsRequest

服務：Amazon Lex Model Building Service

提供交談記錄所需的設定。

目錄

iamRoleArn

IAM 角色的 Amazon 資源名稱 (ARN) 具有寫入文字 CloudWatch 日誌的日誌和 S3 儲存貯體以取得音訊日誌的權限。如果啟用音訊加密，此角色也會為用於加密音訊記錄的 AWS KMS 金鑰提供存取權限。如需詳細資訊，請參閱為交談記錄建立 IAM 角色和政策。

類型：字串

長度限制：長度下限為 20。長度上限為 2048。

模式：`^arn:[\w\-]+:iam::[\d]{12}:role/.+$`

必要：是

logSettings

交談記錄的設定。您可以記錄對話文本，對話音頻或兩者。

類型：LogSettingsRequest 物件陣列

必要：是

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS SDK for Java V2 的开发
- AWS 適用於紅寶石 V3 的 SDK

ConversationLogsResponse

服務：Amazon Lex Model Building Service

包含交談記錄設定的相關資訊。

目錄

iamRoleArn

用來將日誌寫入日誌或 S3 儲存貯體的 IAM 角色的 Amazon 資源名稱 (ARN)。 CloudWatch

類型：字串

長度限制：長度下限為 20。長度上限為 2048。

模式：`^arn:[\w\-]+:iam::[\d]{12}:role/.+$`

必要：否

logSettings

交談記錄的設定。您可以記錄文本，音頻或兩者。

類型：LogSettingsResponse 物件陣列

必要：否

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS SDK for Java V2 的开发
- AWS 適用於紅寶石 V3 的 SDK

EnumerationValue

服務：Amazon Lex Model Building Service

每個位置類型都可以有一組值。每個列舉值都代表插槽類型可以採用的值。

例如，披薩訂購機器人可能有一個插槽類型，該插槽類型指定了比薩餅應具有的外殼類型。插槽類型可以包含值

- 厚
- thin
- 填充的

目錄

value

槽類型的值。

類型：字串

長度限制：長度下限為 1。長度上限為 140。

必要：是

synonyms

與槽類型值相關的其他值。

類型：字串陣列

長度限制：長度下限為 1。長度上限為 140。

必要：否

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS SDK for Java V2 的開發
- AWS 適用於紅寶石 V3 的 SDK

FollowUpPrompt

服務：Amazon Lex Model Building Service

完成意圖後提示其他活動。例如，在實現`OrderPizza`意圖後，您可能會提示用戶確定用戶是否要訂購飲料。

目錄

prompt

提示使用者輸入資訊。

類型：Prompt 物件

必要：是

rejectionStatement

如果使用者對`prompt`欄位中定義的問題回答「否」，Amazon Lex 會回應此陳述式，以確認意圖已取消。

類型：Statement 物件

必要：是

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS SDK for Java V2 的軟件
- AWS 適用於紅寶石 V3 的 SDK

FulfillmentActivity

服務：Amazon Lex Model Building Service

描述使用者提供意圖所需的所有資訊之後，如何達成意圖。您可以提供 Lambda 函數來處理意圖，也可以將意圖資訊傳回給用戶端應用程式。我們建議您使用 Lambda 函數，以便相關邏輯存在於雲端，並將用戶端程式碼主要限制為簡報。如果您需要更新邏輯，則只需更新 Lambda 函數；不需要升級用戶端應用程式。

請考量下列範例：

- 在比薩餅訂購應用程序中，在用戶提供下訂單的所有信息後，您可以使用 Lambda 函數向比薩店下訂單。
- 在遊戲應用程式中，當使用者說「拿起岩石」時，這項資訊必須回到用戶端應用程式，以便它可以執行作業並更新圖形。在這種情況下，您希望 Amazon Lex 將意圖資料傳回給用戶端。

目錄

type

透過執行 Lambda 函數或將插槽資料傳回至用戶端應用程式，應該如何實現意圖。

類型：字串

有效值:`ReturnIntent | CodeHook`

必要：是

codeHook

為了達成意圖而執行的 Lambda 函數的說明。

類型：CodeHook 物件

必要：否

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS SDK for Java V2 的开发

- AWS 適用於紅寶石 V3 的 SDK

InputContext

服務：Amazon Lex Model Building Service

必須處於作用中狀態，才能由 Amazon Lex 選取意圖的內容名稱。

目錄

name

內容的名稱。

類型：字串

長度限制：長度下限為 1。長度上限為 100。

模式：`^([A-Za-z]_?)+$`

必要：是

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS SDK for Java V2 的開發
- AWS 適用於紅寶石 V3 的 SDK

Intent

服務：Amazon Lex Model Building Service

識別意圖的特定版本。

目錄

intentName

意圖的名稱。

類型：字串

長度限制：長度下限為 1。長度上限為 100。

模式：`^([A-Za-z]_?)+$`

必要：是

intentVersion

意圖的版本。

類型：字串

長度限制：長度下限為 1。長度上限為 64。

模式：`\$LATEST|[0-9]+`

必要：是

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS SDK for Java V2 的軟件
- AWS 適用於紅寶石 V3 的 SDK

IntentMetadata

服務：Amazon Lex Model Building Service

提供有關意圖的資訊。

目錄

createdDate

建立意圖的日期。

類型：Timestamp

必要：否

description

意圖的描述。

類型：字串

長度限制：長度下限為 0。長度上限為 200。

必要：否

lastUpdatedDate

更新意圖的日期。建立意圖時，建立日期與上次更新日期相同。

類型：Timestamp

必要：否

name

意圖的名稱。

類型：字串

長度限制：長度下限為 1。長度上限為 100。

模式：`^([A-Za-z]_?)+$`

必要：否

version

意圖的版本。

類型：字串

長度限制：長度下限為 1。長度上限為 64。

模式：`\$LATEST|[0-9]+`

必要：否

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS SDK for Java V2 的軟件
- AWS 適用於紅寶石 V3 的 SDK

KendraConfiguration

服務：Amazon Lex Model Building Service

提供亞馬遜的組態資訊。 KendraSearchIntent意圖。當您使用此意圖時，Amazon Lex 會搜尋指定的 Amazon Kendra 索引，並從索引中傳回符合使用者話語的文件。如需詳細資訊，請參閱 AMAZON。KendraSearchIntent。

目錄

kendraIndex

Amazon 資源名稱（ARN）Amazon Kendra 索引，你想要亞馬遜。 KendraSearchIntent 意圖搜索。索引必須與 Amazon Lex 機器人位於相同的帳戶和區域。如果 Amazon Kendra 索引不存在，則在呼叫PutIntent作業時會收到例外狀況。

類型：字串

長度限制：長度下限為 20。長度上限為 2048。

模式：arn:aws:kendra:[a-z]+-[a-z]+-[0-9]:[0-9]{12}:index\/[a-zA-Z0-9][a-zA-Z0-9_-]*

必要：是

role

具有搜尋 Amazon 肯德拉索引之權限的 IAM 角色的亞馬遜資源名稱 (ARN)。角色必須與 Amazon Lex 機器人位於相同的帳戶和區域。如果角色不存在，則在調用PutIntent操作時會出現異常。

類型：字串

長度限制：長度下限為 20。長度上限為 2048。

模式：arn:aws:iam::[0-9]{12}:role/.*

必要：是

queryFilterString

Amazon Lex 傳送給 Amazon Kendra 的查詢篩選器，以篩選查詢的回應。該篩選器的格式由 Amazon Kendra 定義。如需詳細資訊，請參閱篩選查詢。

您可以在執行階段使用新的篩選字串覆寫此篩選字串。

類型：字串

長度限制：長度下限為 0。

必要：否

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS 適用於 Java V2 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

LogSettingsRequest

服務：Amazon Lex Model Building Service

用來設定交談記錄的傳遞模式和目的地的設定。

目錄

destination

將傳送記錄的位置。文字記錄會傳送至記 CloudWatch 錄檔記錄群組。音訊日誌會傳送至 S3 儲存貯體。

類型：字串

有效值:`CLOUDWATCH_LOGS | S3`

必要：是

logType

要啟用的記錄類型。文字記錄會傳送至記 CloudWatch錄檔記錄群組。音訊日誌會傳送至 S3 儲存貯體。

類型：字串

有效值:`AUDIO | TEXT`

必要：是

resourceArn

應在其中交付 CloudWatch 日誌日誌群組或 S3 儲存貯體的 Amazon 資源名稱 (ARN)。

類型：字串

長度限制：長度下限為 1。長度上限為 2048。

模式：`^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])$`

必要：是

kmsKeyArn

AWS KMS 客戶受管金鑰的 Amazon 資源名稱 (ARN)，用於加密傳遞到 S3 儲存貯體的音訊日誌。金鑰不適用於 CloudWatch 日誌，對於 S3 儲存貯體而言是選用的。

類型：字串

長度限制：長度下限為 20。長度上限為 2048。

模式：`^arn:[\w\-]+:kms:[\w\-]+:[\d]{12}:(?:key\/[\w\-]+|alias\/[a-zA-Z0-9:\/_\-]{1,256})$`

必要：否

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS SDK for Java V2 的开发
- AWS 適用於紅寶石 V3 的 SDK

LogSettingsResponse

服務：Amazon Lex Model Building Service

交談記錄的設定。

目錄

destination

傳送記錄檔的目的地。

類型：字串

有效值:CLOUDWATCH_LOGS | S3

必要：否

kmsKeyArn

用於加密 S3 儲存貯體中音訊日誌的金鑰的 Amazon 資源名稱 (ARN)。

類型：字串

長度限制：長度下限為 20。長度上限為 2048。

模式：^arn:[\w\-]+:kms:[\w\-]+:[\d]{12}:(?:key\/[\w\-]+|alias\/[a-zA-Z0-9:\/_\-]{1,256})$

必要：否

logType

已啟用的記錄類型。

類型：字串

有效值:AUDIO | TEXT

必要：否

resourceArn

日誌日誌群組或交付 CloudWatch 日誌的 S3 儲存貯體的 Amazon 資源名稱 (ARN)。

類型：字串

長度限制：長度下限為 1。長度上限為 2048。

模式：`^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])$`

必要：否

resourcePrefix

資源前置詞是您指定要包含音訊日誌的 S3 儲存貯體中 S3 物件金鑰的第一部分。對於 CloudWatch 記錄檔，它是您指定的記錄群組內記錄資料流名稱的前置詞。

類型：字串

長度限制：長度上限為 1024。

必要：否

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS SDK for Java V2 的軟件
- AWS 適用於紅寶石 V3 的 SDK

Message

服務：Amazon Lex Model Building Service

提供訊息文字及其類型的訊息物件。

目錄

content

訊息的文字。

類型：字串

長度限制：長度下限為 1。長度上限為 1000。

必要：是

contentType

訊息字串的內容類型。

類型：字串

有效值:`PlainText | SSML | CustomPayload`

必要：是

groupNumber

識別郵件所屬的訊息群組。將群組指派給訊息時，Amazon Lex 會從回應中的每個群組傳回一則訊息。

類型：整數

有效範圍：最小值為 1。最大值為 5。

必要：否

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS SDK for Java V2 的軟件

- AWS 適用於紅寶石 V3 的 SDK

MigrationAlert

服務：Amazon Lex Model Building Service

提供 Amazon Lex 在遷移期間傳送的警示和警告的相關資訊。警示包含如何解決問題的相關資訊。

目錄

details

有關警示的其他詳細資訊。

類型：字串陣列

必要：否

message

說明發出警示的原因的訊息。

類型：字串

必要：否

referenceURLs

Amazon Lex 文件的連結，其中說明如何解決警示。

類型：字串陣列

必要：否

type

警示的類型。警示有兩種類型：

- ERROR-移轉發生無法解決的問題。移轉會停止。
- WARN-遷移發生問題，需要手動變更新的 Amazon Lex V2 機器人。移轉會繼續進行。

類型：字串

有效值:ERROR | WARN

必要：否

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS SDK for Java V2 的軟件
- AWS 適用於紅寶石 V3 的 SDK

MigrationSummary

服務：Amazon Lex Model Building Service

提供將機器人從亞馬遜 Lex V1 遷移到亞馬遜萊克 Amazon Lex V2 的相關資訊。

目錄

migrationId

Amazon Lex 指派給遷移的唯一識別碼。

類型：字串

長度約束：固定長度為 10。

模式：`^[0-9a-zA-Z]+$`

必要：否

migrationStatus

操作的狀態。當狀態為COMPLETE機器人可在 Amazon Lex V2 中使用時。可能有警示和警告需要解決才能完成移轉。

類型：字串

有效值:`IN_PROGRESS | COMPLETED | FAILED`

必要：否

migrationStrategy

用來執行移轉的策略。

類型：字串

有效值:`CREATE_NEW | UPDATE_EXISTING`

必要：否

migrationTimestamp

移轉開始的日期和時間。

類型：Timestamp

必要：否

v1BotLocale

作為移轉來源的 Amazon Lex V1 機器人的地區設定。

類型：字串

有效值:`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`

必要：否

v1BotName

作為移轉來源的 Amazon Lex V1 機器人的名稱。

類型：字串

長度約束：最小長度為 2。長度上限為 50。

模式：`^([A-Za-z]_?)+$`

必要：否

v1BotVersion

作為移轉來源的 Amazon Lex V1 機器人版本。

類型：字串

長度限制：長度下限為 1。長度上限為 64。

模式：`\$LATEST|[0-9]+`

必要：否

v2BotId

作為遷移目的地之 Amazon Lex V2 的唯一識別碼。

類型：字串

長度約束：固定長度為 10。

模式：`^[0-9a-zA-Z]+$`

必要：否

v2BotRole

亞馬遜萊克斯用來執行亞馬遜萊克 Amazon Lex V2 機器人的 IAM 角色。

類型：字串

長度限制：長度下限為 20。長度上限為 2048。

模式：`^arn:[\w\-]+:iam::[\d]{12}:role/.+$`

必要：否

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS SDK for Java V2 的开发
- AWS 適用於紅寶石 V3 的 SDK

OutputContext

服務：Amazon Lex Model Building Service

符合意圖時所設定的輸出內容規格。

目錄

name

內容的名稱。

類型：字串

長度限制：長度下限為 1。長度上限為 100。

模式：^([A-Za-z]_?)+$

必要：是

timeToLiveInSeconds

前後關聯在PostContent或PostText回應中第一次傳送之後，應該處於作用中狀態的秒數。您可以設定介於 5 到 86,400 秒 (24 小時) 之間的值。

類型：整數

有效範圍：最小值為 5。最大值為 86400。

必要：是

turnsToLive

上下文應該處於活動狀態的交談次數。對話回合是一個PostContent或PostText請求以及來自 Amazon Lex 的相應回應。

類型：整數

有效範圍：最小值為 1。最大值為 20。

必要：是

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS SDK for Java V2 的开发
- AWS 適用於紅寶石 V3 的 SDK

Prompt

服務：Amazon Lex Model Building Service

從使用者取得資訊。若要定義提示，請提供一或多個訊息，並指定從使用者取得資訊的嘗試次數。如果您提供多個訊息，Amazon Lex 會選擇其中一個訊息來提示使用者。如需詳細資訊，請參閱 Amazon Lex 運作方式。

目錄

maxAttempts

提示使用者輸入資訊的次數。

類型：整數

有效範圍：最小值為 1。最大值為 5。

必要：是

messages

物件陣列，每個物件都會提供訊息字串及其類型。您可以在純文字或語音合成標記語言 (SSML) 中指定郵件字串。

類型：Message 物件陣列

陣列成員：項目數下限為 1。最多 15 個項目數。

必要：是

responseCard

回應卡。Amazon Lex 會在執行時期的 `PostText` API 回應中使用此提示。它替代了在響應卡佔位符會話屬性和槽值。如需詳細資訊，請參閱 使用回應卡。

類型：字串

長度限制：長度下限為 1。最大長度為 5 萬。

必要：否

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS SDK for Java V2 的開發
- AWS 適用於紅寶石 V3 的 SDK

ResourceReference

服務：Amazon Lex Model Building Service

描述參照您嘗試刪除之資源的資源。該對象作為ResourceInUseException異常的一部分返回。

目錄

name

正在使用您嘗試刪除之資源的資源名稱。

類型：字串

長度限制：長度下限為 1。長度上限為 100。

模式：[a-zA-Z_]+

必要：否

version

正在使用您嘗試刪除之資源的資源版本。

類型：字串

長度限制：長度下限為 1。長度上限為 64。

模式：\$LATEST|[0-9]+

必要：否

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS 適用於 Java V2 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

Slot

服務：Amazon Lex Model Building Service

識別特定插槽的版本。

目錄

name

位置的名稱。

類型：字串

長度限制：長度下限為 1。長度上限為 100。

模式：`^([A-Za-z](-|_|.)?)+$`

必要：是

slotConstraint

指定位置是必要項目還是選用項目。

類型：字串

有效值:`Required | Optional`

必要：是

defaultValueSpec

槽的預設值清單。當 Amazon Lex 尚未決定插槽的值時，會使用預設值。您可以從內容變數、工作階段屬性和定義的值指定預設值。

類型：SlotDefaultValueSpec 物件

必要：否

description

插槽的描述。

類型：字串

長度限制：長度下限為 0。長度上限為 200。

必要：否

obfuscationSetting

決定是否在交談記錄和儲存語音中混淆插槽。當您混淆插槽時，值會由大括號 ({}) 中的插槽名稱取代。例如，如果插槽名稱是「完整名稱」，則模糊化的值會取代為「{full_name}」。如需詳細資訊，請參閱插槽模糊化。

類型：字串

有效值:NONE | DEFAULT_OBFUSCATION

必要：否

priority

指示 Amazon Lex 向使用者引出此插槽值的順序。例如，如果意圖有兩個優先順序為 1 和 2 的插槽，AWS Amazon Lex 會先針對優先順序為 1 的插槽提出一個值。

如果多個插槽共用相同的優先順序，Amazon Lex 產生值的順序是任意的。

類型：整數

有效範圍：最小值為 0。最大值為 100。

必要：否

responseCard

文字型用戶端所使用之插槽類型的一組可能回應。用戶從響應卡中選擇一個選項，而不是使用文本來回复。

類型：字串

長度限制：長度下限為 1。最大長度為 5 萬。

必要：否

sampleUtterances

如果您知道使用者可能會針對插槽值回應 Amazon Lex 請求的特定模式，您可以提供這些語音以提高準確性。這是選用的。在大多數情況下，Amazon Lex 能夠瞭解使用者的話語。

類型：字串陣列

陣列成員：項目數下限為 0。項目數上限為 10。

長度限制：長度下限為 1。長度上限為 200。

必要：否

slotType

插槽的類型，可能是您定義的自訂插槽類型，或是其中一種內建插槽類型。

類型：字串

長度限制：長度下限為 1。長度上限為 100。

模式：`^((AMAZON\.)_?|[A-Za-z]_?)+`

必要：否

slotTypeVersion

插槽類型的版本。

類型：字串

長度限制：長度下限為 1。長度上限為 64。

模式：`\$LATEST|[0-9]+`

必要：否

valueElicitationPrompt

Amazon Lex 用來向使用者引出插槽值的提示。

類型：Prompt 物件

必要：否

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS SDK for Java V2 的軟件
- AWS 適用於紅寶石 V3 的 SDK

SlotDefaultValue

服務：Amazon Lex Model Building Service

插槽的預設值。

目錄

defaultValue

插槽的預設值。您可以指定下列其中一個選項：

- `#context-name.slot-name`-上下文「上下文名稱」中的插槽值「插槽名稱」。
- `{attribute}`-階段作業屬性「屬性」的插槽值。
- `'value'`-離散值「值」。

類型：字串

長度限制：長度下限為 1。最大長度為 202。

必要：是

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS SDK for Java V2 的开发
- AWS 適用於紅寶石 V3 的 SDK

SlotDefaultValueSpec

服務：Amazon Lex Model Building Service

包含槽的預設值。當 Amazon Lex 尚未決定插槽的值時，會使用預設值。

目錄

defaultValueList

插槽的預設值。您可以指定多個預設值。例如，您可以從相符的內容變數、工作階段屬性或固定值中指定要使用的預設值。

系統會根據您在清單中指定的順序來選取預設值。例如，如果您以該順序指定上下文變數和固定值，Amazon Lex 會使用上下文變數 (如果可用)，否則會使用固定值。

類型：SlotDefaultValue 物件陣列

陣列成員：項目數下限為 0。項目數上限為 10。

必要：是

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS SDK for Java V2 的开发
- AWS 適用於紅寶石 V3 的 SDK

SlotTypeConfiguration

服務：Amazon Lex Model Building Service

提供插槽類型的組態資訊。

目錄

regexConfiguration

用來驗證位置值的規則運算式。

類型：SlotTypeRegexConfiguration 物件

必要：否

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS 適用於 Java V2 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

SlotTypeMetadata

服務：Amazon Lex Model Building Service

提供插槽類型的相關資訊。

目錄

createdDate

插槽類型的建立日期。

類型：Timestamp

必要：否

description

位置類型的描述。

類型：字串

長度限制：長度下限為 0。長度上限為 200。

必要：否

lastUpdatedDate

更新插槽類型的日期。建立資源時，建立日期和上次更新日期相同。

類型：Timestamp

必要：否

name

位置類型的名稱。

類型：字串

長度限制：長度下限為 1。長度上限為 100。

模式：`^([A-Za-z]_?)+$`

必要：否

version

插槽類型的版本。

類型：字串

長度限制：長度下限為 1。長度上限為 64。

模式：`\$LATEST|[0-9]+`

必要：否

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS 適用於 Java V2 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

SlotTypeRegexConfiguration

服務：Amazon Lex Model Building Service

提供用來驗證位置值的規則運算式。

目錄

pattern

用來驗證位置值的規則運算式。

使用標準規則運算式。Amazon Lex 在規則運算式中支援下列字元：

- A-Z、a-z
- 0-9
- 統一碼字符 (「\ u<Unicode>」)

用四個數字表示萬國碼字符，例如「\ u0041」或「\ u005a」。

不支援下列規則運算式：

- 無限的重複項：*、+ 或 {x,}，沒有上限。
- 萬用字元 (.)

類型：字串

長度限制：長度下限為 1。長度上限為 100。

必要：是

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS 適用於 Java V2 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

Statement

服務：Amazon Lex Model Building Service

將資訊傳達給使用者的訊息集合。在執行階段，Amazon Lex 會選取要傳送的訊息。

目錄

messages

訊息物件的集合。

類型：Message 物件陣列

陣列成員：項目數下限為 1。最多 15 個項目數。

必要：是

responseCard

在執行階段，如果用戶端使用 PostTextAPI，Amazon Lex 會在回應中包含回應卡。它替換所有的會話屬性和插槽值在響應卡中的佔位符。

類型：字串

長度限制：長度下限為 1。最大長度為 5 萬。

必要：否

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS SDK for Java V2 的軟件
- AWS 適用於紅寶石 V3 的 SDK

Tag

服務：Amazon Lex Model Building Service

識別機器人、機器人別名或機器人通道的索引鍵/值配對清單。標籤鍵和值可以由 Unicode 字母、數字、空格和下列任何符號組成：_.:/= +-@。

目錄

key

標籤的索引鍵。金鑰不區分大小寫，且必須是唯一的。

類型：字串

長度限制：長度下限為 1。長度上限為 128。

必要：是

value

與鍵相關聯的值。該值可能是空字符串，但不能為空。

類型：字串

長度限制：長度下限為 0。長度上限為 256。

必要：是

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS SDK for Java V2 的軟件
- AWS 適用於紅寶石 V3 的 SDK

UtteranceData

服務：Amazon Lex Model Building Service

提供有關對機器人進行的單個語音的信息。

目錄

count

處理語音的次數。

類型：整數

必要：否

distinctUsers

使用該語音的個人總數。

類型：整數

必要：否

firstUtteredDate

首次記錄該話語的日期。

類型：Timestamp

必要：否

lastUtteredDate

上次記錄該話語的日期。

類型：Timestamp

必要：否

utteranceString

使用者輸入的文字或音訊片段的文字表示。

類型：字串

長度限制：長度下限為 1。最大長度為 2000 年。

必要：否

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS SDK for Java V2 的開發
- AWS 適用於紅寶石 V3 的 SDK

UtteranceList

服務：Amazon Lex Model Building Service

提供已對您機器人特定版本進行的語音清單。此清單最多包含 100 個語音。

目錄

botVersion

處理清單的機器人版本。

類型：字串

長度限制：長度下限為 1。長度上限為 64。

模式：`\$LATEST|[0-9]+`

必要：否

utterances

一或多個UtteranceData物件，其中包含對機器人所做語音的相關資訊。物件的最大數目為 100。

類型：UtteranceData 物件陣列

必要：否

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS SDK for Java V2 的軟件
- AWS 適用於紅寶石 V3 的 SDK

Amazon Lex 運行服務

Amazon Lex 運行服務支援下列資料類型：

- ActiveContext
- ActiveContextTimeToLive

- Button
- DialogAction
- GenericAttachment
- IntentConfidence
- IntentSummary
- PredictedIntent
- ResponseCard
- SentimentResponse

ActiveContext

服務：Amazon Lex Runtime Service

上下文是一個變數，其中包含使用者和 Amazon Lex 之間交談目前狀態的相關資訊。Amazon Lex 可在滿足意圖時自動設定上下文，也可以使用、或PutSession作業在執行時間設定上下文。PutContent PutText

目錄

name

內容的名稱。

類型：字串

長度限制：長度下限為 1。長度上限為 100。

模式：^([A-Za-z]_?)+$

必要：是

parameters

目前上下文的狀態變數。您可以使用這些值作為後續事件中插槽的預設值。

類型：字串到字串映射

地圖項目：0 個項目的最小數目。項目數上限為 10。

索引鍵長度限制：長度下限為 1。長度上限為 100。

值長度限制：長度下限為 1。長度上限為 1024。

必要：是

timeToLive

前後關聯保持活動狀態的時間長度或迴轉數。

類型：ActiveContextTimeToLive 物件

必要：是

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS SDK for Java V2 的軟件
- AWS 適用於紅寶石 V3 的 SDK

ActiveContextTimeToLive

服務：Amazon Lex Runtime Service

前後關聯保持活動狀態的時間長度或迴轉數。

目錄

timeToLiveInSeconds

前後關聯在PostContent或PostText回應中第一次傳送之後，應該處於作用中狀態的秒數。您可以設定介於 5 到 86,400 秒 (24 小時) 之間的值。

類型：整數

有效範圍：最小值為 5。最大值為 86400。

必要：否

turnsToLive

上下文應該處於活動狀態的交談次數。對話回合是一個PostContent或PostText請求以及來自 Amazon Lex 的相應回應。

類型：整數

有效範圍：最小值為 1。最大值為 20。

必要：否

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS SDK for Java V2 的开发
- AWS 適用於紅寶石 V3 的 SDK

Button

服務：Amazon Lex Runtime Service

代表要在客戶端平台上顯示的選項（臉譜，鬆弛等）

目錄

text

按鈕上的使用者可見的文字。

類型：字串

長度限制：長度下限為 1。長度上限為 15。

必要：是

value

當使用者選擇按鈕時，傳送至 Amazon Lex 的值。例如，考慮按鈕文字「NYC」。當用戶選擇按鈕時，發送的值可以是「紐約市」。

類型：字串

長度限制：長度下限為 1。長度上限為 1000。

必要：是

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS 適用於 Java V2 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

DialogAction

服務：Amazon Lex Runtime Service

描述機器人在與使用者互動時應採取的下一個動作，並提供動作發生之前後關聯的相關資訊。使用資DialogAction料類型將互動設定為特定狀態，或將互動回復到先前的狀態。

目錄

type

機器人與使用者互動時應採取的下一個動作。可能值如下：

- ConfirmIntent-下一個動作是詢問使用者意圖是否已完成並準備完成。這是一個是/否的問題，例如「下訂單？」
- Close-表示不會有來自用戶的響應。例如，「您的訂單已下訂單」聲明不需要回復。
- Delegate-下一個動作由 Amazon Lex 決定。
- ElicitIntent-下一個動作是決定使用者想要履行的意圖。
- ElicitSlot-下一個動作是從使用者引出插槽值。

類型：字串

有效值:ElicitIntent | ConfirmIntent | ElicitSlot | Close | Delegate

必要：是

fulfillmentState

意圖的履行狀態。可能值如下：

- Failed-與意圖相關聯的 Lambda 函數無法完成意圖。
- Fulfilled-與意圖相關聯的 Lambda 函數已完成意圖。
- ReadyForFulfillment-意圖所需的所有信息都存在，並準備由客戶端應用程序實現的意圖。

類型：字串

有效值:Fulfilled | Failed | ReadyForFulfillment

必要：否

intentName

意圖的名稱。

類型：字串

必要：否

message

應該顯示給用戶的消息。如果您未指定訊息，Amazon Lex 將使用針對意圖設定的訊息。

類型：字串

長度限制：長度下限為 1。長度上限為 1024。

必要：否

messageFormat

- PlainText-郵件包含純 UTF-8 文字。
- CustomPayload-訊息是用戶端的自訂格式。
- SSML-訊息包含格式化為語音輸出的文字。
- Composite-訊息包含含有一或多則訊息的逸出 JSON 物件。如需詳細資訊，請參閱訊息群組。

類型：字串

有效值:PlainText | CustomPayload | SSML | Composite

必要：否

slots

已收集的插槽及其價值觀的地圖。

類型：字串到字串映射

必要：否

slotToElicit

應該從用戶引起的插槽的名稱。

類型：字串

必要：否

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS SDK for Java V2 的軟件
- AWS 適用於紅寶石 V3 的 SDK

GenericAttachment

服務：Amazon Lex Runtime Service

表示在顯示提示時呈現給用戶的選項。它可以是圖像，按鈕，鏈接或文本。

目錄

attachmentLinkUrl

附件到響應卡的 URL。

類型：字串

長度限制：長度下限為 1。長度上限為 2048。

必要：否

buttons

顯示給用戶的選項列表。

類型：Button 物件陣列

陣列成員：項目數下限為 0。項目數上限為 5。

必要：否

imageUrl

顯示給使用者的影像 URL。

類型：字串

長度限制：長度下限為 1。長度上限為 2048。

必要：否

subTitle

標題下方顯示的副標題。

類型：字串

長度限制：長度下限為 1。長度上限為 80。

必要：否

title

選項的標題。

類型：字串

長度限制：長度下限為 1。長度上限為 80。

必要：否

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS 適用於 Java V2 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

IntentConfidence

服務：Amazon Lex Runtime Service

提供一個分數，指出 Amazon Lex 的意圖是滿足使用者意圖的信心。

目錄

score

一個分數，指出 Amazon Lex 對意圖滿足使用者意圖的信心程度。介於 0.00 和 1.00 之間的範圍。分數越高表示信心越高。

類型：Double

必要：否

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS 適用於 Java V2 的 SDK
- AWS 適用於紅寶石 V3 的 SDK

IntentSummary

服務：Amazon Lex Runtime Service

提供意圖狀態的相關資訊。您可以使用此資訊來取得意圖的目前狀態，以便您可以處理意圖，或讓您可以將意圖返回到其先前的狀態。

目錄

dialogActionType

機器人與使用者互動時應採取的下一個動作。可能值如下：

- ConfirmIntent-下一個動作是詢問使用者意圖是否已完成並準備完成。這是一個是/否的問題，例如「下訂單？」
- Close-表示不會有來自用戶的響應。例如，「您的訂單已下訂單」聲明不需要回復。
- ElicitIntent-下一個動作是決定使用者想要履行的意圖。
- ElicitSlot-下一個動作是從使用者引出插槽值。

類型：字串

有效值:ElicitIntent | ConfirmIntent | ElicitSlot | Close | Delegate

必要：是

checkpointLabel

識別特定意圖的使用者定義標籤。您可以使用此標籤返回到先前的意圖。

使用GetSessionRequest作業的checkpointLabelFilter參數，將作業傳回的意圖篩選為僅具有指定標籤的意圖。

類型：字串

長度限制：長度下限為 1。長度上限為 255。

模式：[a-zA-Z0-9-]+

必要：否

confirmationStatus

使用者回應確認提示後的意圖狀態。如果使用者確認意圖，Amazon Lex 會將此欄位設定為Confirmed。如果使用者拒絕意圖，Amazon Lex 會將此值設定為Denied。可能值如下：

- Confirmed-使用者已對確認提示回應「是」，確認意圖已完成且已準備好完成。
- Denied-使用者已對確認提示回應「否」。
- None-從未提示使用者進行確認；或者，系統提示使用者，但未確認或拒絕提示。

類型：字串

有效值:None | Confirmed | Denied

必要：否

fulfillmentState

意圖的履行狀態。可能值如下：

- Failed-與意圖相關聯的 Lambda 函數無法完成意圖。
- Fulfilled-與意圖相關聯的 Lambda 函數已完成意圖。
- ReadyForFulfillment-意圖所需的所有信息都存在，並準備由客戶端應用程序實現的意圖。

類型：字串

有效值:Fulfilled | Failed | ReadyForFulfillment

必要：否

intentName

意圖的名稱。

類型：字串

必要：否

slots

已收集的插槽及其價值觀的地圖。

類型：字串到字串映射

必要：否

slotToElicit

從使用者引出的下一個插槽。如果沒有插槽可引出，則該欄位為空白。

類型：字串

必要：否

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS SDK for Java V2 的軟件
- AWS 適用於紅寶石 V3 的 SDK

PredictedIntent

服務：Amazon Lex Runtime Service

Amazon Lex 建議符合使用者意圖的意圖。包括意圖的名稱、Amazon Lex 確定使用者意圖得到滿足的信心，以及針對意圖定義的插槽。

目錄

intentName

Amazon Lex 建議的意圖名稱可滿足使用者的意圖。

類型：字串

必要：否

nluIntentConfidence

指出 Amazon Lex 對意圖滿足使用者意圖的信心程度。

類型：IntentConfidence 物件

必要：否

slots

與預測意圖相關聯的槽和槽值。

類型：字串到字串映射

必要：否

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS SDK for Java V2 的軟件
- AWS 適用於紅寶石 V3 的 SDK

ResponseCard

服務：Amazon Lex Runtime Service

如果您在建立機器人時設定回應卡，Amazon Lex 會替換可用的工作階段屬性和插槽值，然後將其傳回。響應卡也可以來自 Lambda 函數（`dialogCodeHook`並且`fulfillmentActivity`意圖）。

目錄

contentType

回應的內容類型。

類型：字串

有效值:`application/vnd.amazonaws.card.generic`

必要：否

genericAttachments

表示選項的附件物件陣列。

類型：GenericAttachment 物件陣列

陣列成員：項目數下限為 0。項目數上限為 10。

必要：否

version

回應卡格式的版本。

類型：字串

必要：否

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS SDK for Java V2 的开发
- AWS 適用於紅寶石 V3 的 SDK

SentimentResponse

服務：Amazon Lex Runtime Service

在話語中表達的情緒。

當機器人設定為將語音傳送至 Amazon Comprehend 進行情緒分析時，此欄位結構會包含分析結果。

目錄

sentimentLabel

Amazon Comprehend 最有信心的推斷情緒。

類型：字串

必要：否

sentimentScore

正確推斷情緒的可能性。

類型：字串

必要：否

另請參閱

如需在其中一個特定語言 AWS SDK 中使用此 API 的詳細資訊，請參閱下列內容：

- AWS SDK for C++
- AWS SDK for Java V2 的軟件
- AWS 適用於紅寶石 V3 的 SDK

Amazon Lex 的文件歷史記錄

- 文件最近更新時間：2021 年 9 月 9 日

下表說明 Amazon Lex 每個版本的重要變更。如需有關此文件更新的通知，您可以訂閱 RSS 摘要。

變更	描述	日期
新功能	Amazon Lex 現在支持韓語（KO-KR）語言環境。如需詳細資訊，請參閱 Amazon Lex 支援的語言。	2021 年 9 月 9 日
新功能	Amazon Lex 現在支援英文 (印度) 語言環境。如需詳細資訊，請參閱 Amazon Lex 支援的語言。	2021 年 7 月 15 日
新功能	Amazon Lex 現在提供了一個工具，將機器人遷移到 Amazon Lex V2 API。如需更多詳細資訊，請參閱遷移機器人。	2021 年 7 月 13 日
新功能	Amazon Lex 現在支援日文 (日本) 地區設定。如需詳細資訊，請參閱 Amazon Lex 支援的語言。	2021 年 4 月 1 日
新功能	Amazon Lex 現在支援德文 (德文) (DE-DE) 和西班牙文 (拉丁美洲) (es-419) 語言環境。如需詳細資訊，請參閱 Amazon Lex 支援的語言。	2020 年 11 月 23 日
新功能	Amazon Lex 現在支援使用內容來管理啟用意圖。如需更多	2020 年 11 月 19 日

	詳細資訊，請參閱設定意圖內容。	
新功能	Amazon Lex 現在支援法文 (FR-FR)、加拿大法文 (FR-CA)、義大利文 (IT-IT) 和西班牙文 (ES-ES) 地區設定。如需支援的區域的完整清單，請參閱 Amazon Lex 支援的語言。	2020 年 11 月 11 日
新功能	Amazon Lex 現在支援西班牙文 (美國) (ES-US) 地區設定。如需詳細資訊，請參閱 Amazon Lex 支援的語言。	2020 年 9 月 22 日
新功能	Amazon Lex 現在支援英文 (英式) (en-GB) 地區設定。如需詳細資訊，請參閱 Amazon Lex 支援的語言。	2020 年 9 月 15 日
新功能	Amazon Lex 現在支援英文 (澳洲) (en-au) 地區設定。如需詳細資訊，請參閱 Amazon Lex 支援的語言。	2020 年 9 月 8 日
新功能	Amazon Lex 現在有 7 種新的內建對應方式和 9 種新的內建插槽類型。如需詳細資訊，請參閱內建色彩比對方式和插槽類型。	2020 年 9 月 8 日
新範例	了解如何透過 Amazon Kendra 搜尋答案，建立客戶支援專員可用來回答客戶問題的 Amazon Lex 機器人。如需詳細資訊，請參閱範例：呼叫中心代理助理員。	2020 年 8 月 10 日

新功能	Amazon Lex 現在可以根據可信度分數返回最多四個替代意圖。如需詳細資訊，請參閱使用可信度分數。	2020 年 8 月 6 日
區域擴展	亞 Amazon Lex 域 (東京) (ap-northeast-1) 及亞太區域 (東京) (ap-northeast-1)。	2020 年 6 月 30 日
新功能	Amazon Lex 現在支援搜尋 Amazon Kendra 索引，以取得常見問題的解答。如需更多詳細資訊，請參閱 Amazon。KendraSearchIntent。	2020 年 6 月 11 日
新功能	Amazon Lex 現在會在交談日誌中傳回更多資訊。如需詳細資訊，請參閱在 Amazon 日誌中檢視文字 CloudWatch 日誌。	2020 年 6 月 9 日
區域擴展	亞太區域 (倫敦) (ap-west-2)、歐洲 (倫敦) (west-2)、歐洲 (倫敦) (west-2)、歐洲 (倫敦) (ap-west-2) 及歐洲 (倫敦) (ap-west-2) 及歐洲 (倫敦) (ap-west-2) 及歐洲 (倫敦) (west-2)	2020 年 4 月 23 日
新功能	Amazon Lex 現在支持標記。您可以使用標記來識別資源、配置成本及控制存取。如需更多詳細資訊，請參閱標記您的 Amazon Lex 資源。	2020 年 3 月 12 日

新功能	Amazon Lex 現在支持亞馬遜的正則表達式。 AlphaNumeric 內建插槽類型。如需更多詳細資訊，請參閱 Amazon。AlphaNumeric。	2020 年 2 月 6 日
新功能	Amazon Lex 現在可以記錄對話資訊，並混淆這些日誌中的插槽值。如需詳細資訊，請參閱建立對話日誌和槽混淆。	2019 年 12 月 19 日
區域擴展	亞 Amazon Lex 域 (雪梨) (ap-southeast-2) 及亞太區域 (雪梨) (ap-southeast-2)。	2019 年 12 月 17 日
新功能	Amazon Lex 現在符合 HIPAA 規定。如需更多詳細資訊，請參閱 Amazon Lex 的合規驗證。	2019 年 12 月 10 日
新功能	Amazon Lex 現在可以將使用者話語傳送到 Amazon Comprehend，以分析話語的情緒。如需詳細資訊，請參閱情緒分析。	2019 年 11 月 21 日
新功能	Amazon Lex 為 SOC 合規。如需更多詳細資訊，請參閱 Amazon Lex 的合規驗證。	2019 年 11 月 19 日
新功能	Amazon Lex 現在符合 PCI 標準。如需更多詳細資訊，請參閱 Amazon Lex 的合規驗證。	2019 年 10 月 17 日
新功能	新增對於將檢查點新增至意圖，以便在對話期間輕鬆返回意圖的支援。如需詳細資訊，請參閱管理工作階段。	2019 年 10 月 10 日

新功能	新增對 AMAZON.FallbackIntent 的支援讓機器人可以在使用者輸入未如預期時處理情況。如需更多詳細資訊，請參閱 Amazon。FallbackIntent。	2019 年 10 月 3 日
新功能	Amazon Lex 可讓您管理機器人的工作階段資訊。如需詳細資訊，請參閱使用 Amazon Lex API 管理工作階段。	2019 年 8 月 8 日
區域擴展	美國西部 (奧勒岡) (us-west-2)。	2018 年 5 月 8 日
新功能	增加了對以 Amazon Lex 格式導出和導入的支持。如需詳細資訊，請參閱匯入和匯出 Amazon Lex 機器人、意圖和插槽類型。	2018 年 2 月 13 日
新功能	Amazon Lex 現在支援適用於機器人的其他回應訊息。如需詳細資訊，請參閱回應。	2018 年 2 月 8 日
區域擴展	歐洲 (愛爾蘭) (ap-west-2)、歐洲 (愛爾蘭) (ap-west-2) 及使用 Amazon Lex。	2017 年 11 月 21 日
新功能	增加了對在 Kik 上部署亞馬遜萊克斯機器人的支持。如需詳細資訊，請參閱將 Amazon Lex 機器人與 Kik 整合。	2017 年 11 月 20 日
新功能	新增對新的內建槽類型和請求屬性的支援。如需詳細資訊，請參閱「內建槽類型」和「設定請求屬性」。	2017 年 11 月 3 日

新功能	新增匯出到 Alexa Skills Kit 功能。如需詳細資訊，請參閱「匯出至 Alexa 技能」。	2017 年 9 月 7 日
新功能	新增槽類型值的同義詞支援。如需詳細資訊，請參閱「自訂槽類型」。	2017 年 8 月 31 日
新功能	新增 AWS CloudTrail 整合。如需更多詳細資訊，請參閱使用AWS CloudTrail日誌監視 Amazon Lex 的 API 呼叫。	2017 年 8 月 15 日
擴充文件	新增 AWS CLI 的入門範例。如需詳細資訊，請參閱 https://docs.aws.amazon.com/lex/latest/dg/gs-cli.html	2017 年 5 月 22 日
新的指南	此為 Amazon Lex 使用者指南的第一版。	2017 年 4 月 19 日

AWS 詞彙表

如需最新的 AWS 術語，請參閱《AWS 詞彙表 參考》中的 AWS 詞彙表。