aws

開發人員指南

Amazon Lex V1

Amazon Lex V1: 開發人員指南

Table of Contents

支援終止通知：2025 年 9 月 15 日， AWS 將停止對 Amazon Lex V1 的支援。2025 年 9 月 15 日之後，您將無法再存取 Amazon Lex V1 主控台或 Amazon Lex V1 資源。如果您使用的是 Amazon Lex V2，請改參閱 Amazon Lex V2 指南。

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 是一種 AWS 服務，用於使用語音和文字為應用程式建置對話介面。透過 Amazon Lex，支援 Amazon Alexa 的相同對話引擎現在可供任何開發人員使用，讓您能夠在新的和現有的應用程式中建置複雜的自然語言聊天機器人。Amazon Lex 提供自然語言理解 (NLU) 和自動語音辨識 (ASR) 的深度功能和彈性，因此您可以使用栩栩如生的對話互動來建立高度吸引人的使用者體驗，並建立新的產品類別。

Amazon Lex 可讓任何開發人員快速建置對話式聊天機器人。使用 Amazon Lex，不需要任何深度學習專業知識，若要建立機器人，您只需在 Amazon Lex 主控台中指定基本對話流程即可。Amazon Lex 會管理對話，並動態調整對話中的回應。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 提供與 的預先建置整合 AWS Lambda，而且您可以輕鬆地與 AWS 平台上的許多其他服務整合，包括 Amazon Cognito AWS Mobile Hub、Amazon CloudWatch 和 Amazon DynamoDB。與 Lambda 整合可讓機器人存取預先建置的無伺服器企業連接器，以連結至 SaaS 應用程式中的資料，例如 Salesforce、HubSpot 或 Marketo。

使用 Amazon Lex 的一些優點包括：

- 簡單 – Amazon Lex 會引導您使用主控台在幾分鐘內建立自己的聊天機器人。您只提供幾個範例片語，而 Amazon Lex 會建置完整的自然語言模型，讓機器人可以透過語音和文字進行互動，以提出問題、取得答案並完成複雜的任務。

- 民主化深度學習技術 – Amazon Lex 採用與 Alexa 相同的技術，提供 ASR 和 NLU 技術來建立語音語言理解 (SLU) 系統。透過 SLU，Amazon Lex 接受自然語言語音和文字輸入，了解輸入背後的意圖，並透過調用適當的業務函數來實現使用者意圖。

  語音辨識和自然語言理解是電腦科學中最具挑戰性的問題，需要複雜的深度學習演算法接受大量資料和基礎設施的訓練。Amazon Lex 讓所有開發人員都能使用與 Alexa 相同的技術。Amazon Lex 聊天機器人將傳入語音轉換為文字，並了解使用者意圖產生智慧回應，因此您可以專注於為客戶建置具有差異化附加價值的機器人，以定義透過對話界面實現的全新產品類別。

- 無縫部署和擴展 – 使用 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 可讓您輕鬆發佈語音或文字聊天機器人，以用於行動裝置、Web 應用程式和聊天服務（例如 Facebook Messenger)。Amazon Lex 會自動擴展規模，因此您不需要擔心佈建硬體和管理基礎設施來強化機器人體驗。

- 內建與 AWS 平台的整合 – Amazon Lex 與其他 AWS 服務具有原生互通性，例如 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。

- 成本效益 – 使用 Amazon Lex，無需預付成本或最低費用。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。使用 Amazon Lex 免費方案，您可以輕鬆地試用 Amazon Lex，無需任何初始投資。

您是 Amazon Lex 的第一次使用者嗎？

如果您是 Amazon Lex 的初次使用者，我們建議您依序閱讀下列各節：

1. Amazon Lex 入門 – 在此區段中，您會設定您的帳戶並測試 Amazon Lex。
2. API 參考 – 本節提供其他範例，您可以用來探索 Amazon Lex。

Amazon Lex：運作方式

Amazon Lex 可讓您使用採用與 Amazon Alexa 相同技術的語音或文字界面來建置應用程式。以下是您在使用 Amazon Lex 時執行的一般步驟：

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 主控台提供的測試時段用戶端。
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。

開始之前，請先熟悉下列 Amazon Lex 核心概念和術語：

- 機器人 – 機器人會執行自動化任務，例如訂購比薩、預訂飯店、訂購花等。Amazon Lex 機器人採用自動語音辨識 (ASR) 和自然語言理解 (NLU) 功能。每個機器人在您的帳戶中都必須具有唯一的名稱。

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

- 意圖 – 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的。
  - 表達用語範例 – 使用者如何傳達意圖。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 – 在使用者提供必要資訊後，您希望如何實現意圖 （例如，向當地比薩店下訂單）。我們建議您建立 Lambda 函數來滿足意圖。

您可以選擇性地設定意圖，讓 Amazon Lex 直接將資訊傳回用戶端應用程式，以執行必要的履行。

除了訂購比薩等自訂意圖之外，Amazon Lex 還提供內建意圖，以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 – 意圖可能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。在執行時間，Amazon Lex 會提示使用者輸入特定的槽值。使用者必須提供所有必要插槽的值，Amazon Lex 才能實現意圖。

  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。對於每個插槽，您提供槽類型和提示，讓 Amazon Lex 傳送給用戶端以從使用者引出資料。使用者可以使用包含其他單字的槽值來回覆，例如「請大比薩」或「用小寫貼圖」。Amazon Lex 仍然可以了解預期的槽值。

- 槽類型 – 每個槽都有一個類型。您可以建立自訂槽類型或使用內建槽類型。每個槽類型在您的帳戶中都必須具有唯一的名稱。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。

如需可使用 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。

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

主題

Amazon Lex 支援的語言

Amazon Lex V1 支援各種語言和地區設定。下表列出支援的語言和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

支援的語言和地區

Amazon Lex V1 支援下列語言和地區設定。

Code	語言和地區設定
de-DE	德文 （ 德文 ）
en-AU	英文 (澳洲)
en-GB	英文 (英國)
en-IN	英文 (印度)

Code	語言和地區設定
zh-TW	英文 (美國)
es-419	西班牙文 （拉丁美洲）
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 功能支援的語言和地區

所有語言和地區都支援所有 Amazon Lex 功能，除非此表格列出。

功能	支援的語言和地區設定
設定意圖內容	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 中的其他資源類型包括意圖、槽類型、別名和機器人管道關聯。

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。如果您願意，可以透過 AWS CLI 或自己的自訂程式使用模型建置 API 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 執行時間 API 將請求傳送至機器人。例如，當使用

者說「我想要訂購比薩」時，您的用戶端會使用其中一個執行時間 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，並將其用於 意圖。使用這些 Lambda 函數程式碼掛勾來執行執行時間活動，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

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、 和 GetBuiltinSlotTypes分別取得 Amazon Lex 內建意圖、Amazon Lex 內建意圖清單，或您可以在機器人中使用的內建槽類型清單。
- 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 來建立自訂工具，以管理您的 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。

為了確保一次只有一個操作更新資源，Amazon Lex 會使用檢查總和。當您使用 Put APIPutBot PutBotAlias 操作 —PutIntent、 或 PutSlotType— 來更新資源時，您必須在請求中傳遞資源的目前檢查

總和。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。到達 Amazon Lex 的第一個請求符合資源的目前檢查總和。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 GetBot-GetIntent、 和 GetSlotType- 最終一致。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

用戶端應用程式使用以下執行時間 API 操作與 Amazon Lex 通訊：

- PostContent – 接受語音或文字輸入，並傳回意圖資訊和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時間 API 呼叫特定的 Amazon Lex 機器人來處理表達用語：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 執行時間 API 操作，將此使用者輸入傳送至機器人。從使用者輸入，Amazon Lex 會辨識使用者請求是用於機器人中定義的`OrderPizza`意圖。Amazon Lex 會與使用者進行對話，以收集所需資訊或槽資料，例如比薩大小、配料和比薩數量。使用者提供所有必要的槽資料後，Amazon Lex 會叫用 Lambda 函數程式碼掛勾以滿足意圖，或根據意圖的設定，將意圖資料傳回給用戶端。

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，而不是客服人員來處理客戶查詢。您可以使用 8 kHz 音訊格式，直接從電話將音訊傳送至 Amazon Lex。

Amazon Lex 主控台中的測試視窗使用 PostContent API 將文字和語音請求傳送至 Amazon Lex。您可以在Amazon Lex 入門 練習中使用此測試視窗。

Lambda 函數做為程式碼掛鉤

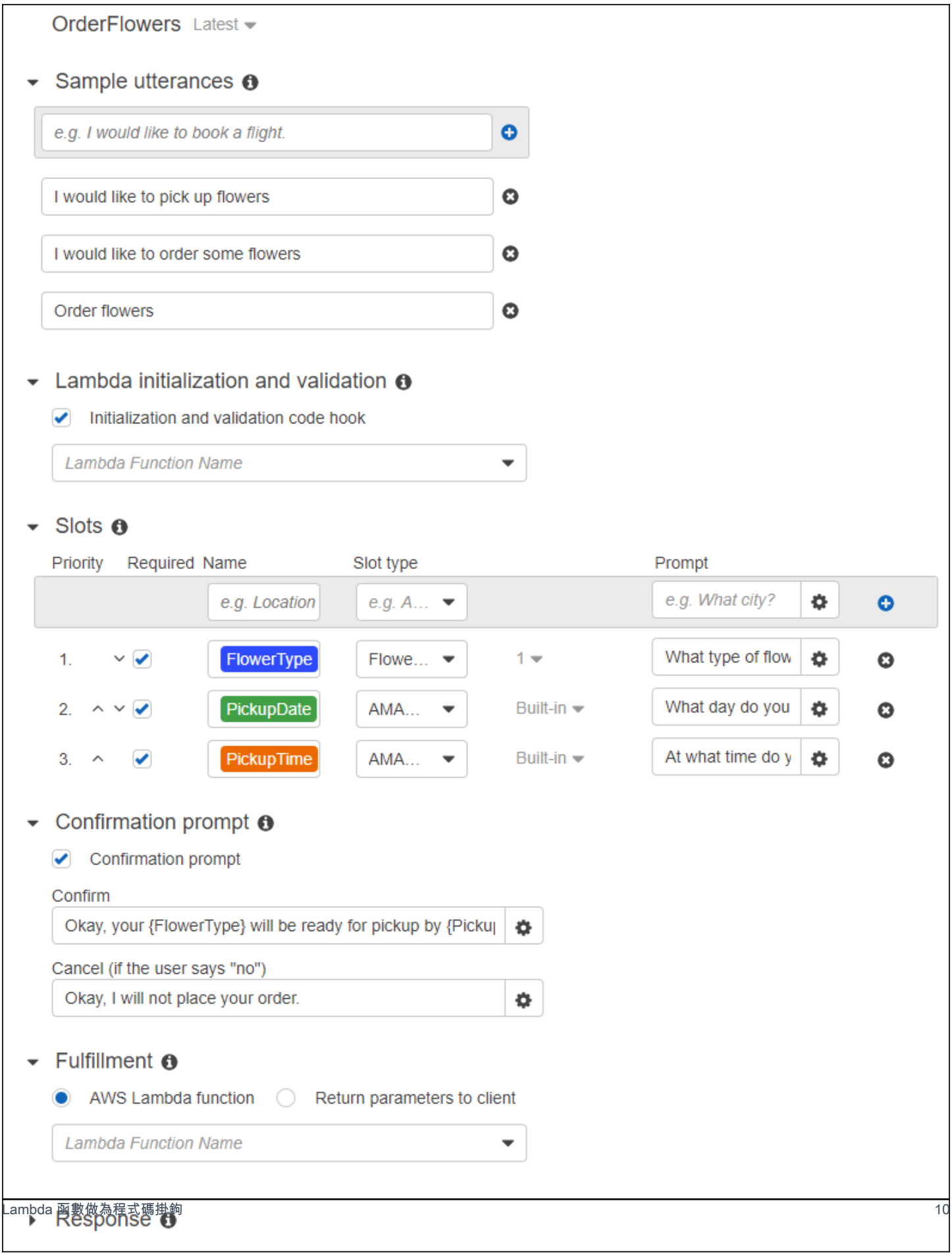
您可以設定 Amazon Lex 機器人以叫用 Lambda 函數做為程式碼掛勾。程式碼掛勾有多個用途：

- 自訂使用者互動 - 例如，當 Joe 要求可用的比薩配料時，您可以使用先前對 Joe 選擇的知識來顯示配料的子集。
- 驗證使用者的輸入 - 假設 Jen 想要在幾小時後挑選花。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 提供其比薩訂單的所有資訊之後，Amazon Lex 可以叫用 Lambda 函數來使用本機比薩來下訂單。

當您設定意圖時，您可以在下列位置將 Lambda 函數指定為程式碼掛勾：

- 用於初始化和驗證的對話方塊程式碼掛勾 - 每個使用者輸入都會叫用此 Lambda 函數，假設 Amazon Lex 了解使用者意圖。
- 履行程式碼掛勾 - 在使用者提供滿足意圖所需的所有槽資料之後，就會叫用此 Lambda 函數。

您可以選擇意圖，並在 Amazon Lex 主控台中設定程式碼掛勾，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 函數接收的事件資料具有 欄位，可將發起人識別為對話方塊或履行程式碼掛勾。您可以使用此資訊來執行程式碼的適當部分。

您可以使用 Lambda 函數來建置可導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，指示 Amazon Lex 採取特定動作。例如，您可以使用ElicitSlot對話方塊動作，指示 Amazon Lex 向使用者要求不需要的槽值。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?
  ```

  如果不了解使用者的意圖，Amazon Lex 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 判斷使用者訂購比薩的意圖後，會將這些訊息傳送給用戶端，以向使用者取得資訊。

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。在執行時間，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，假設您在機器人的 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)。在執行時間，Amazon Lex 會將這些預留位置取代為 值，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 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，您可以消除在 Lambda 函數中建構提示的需求。

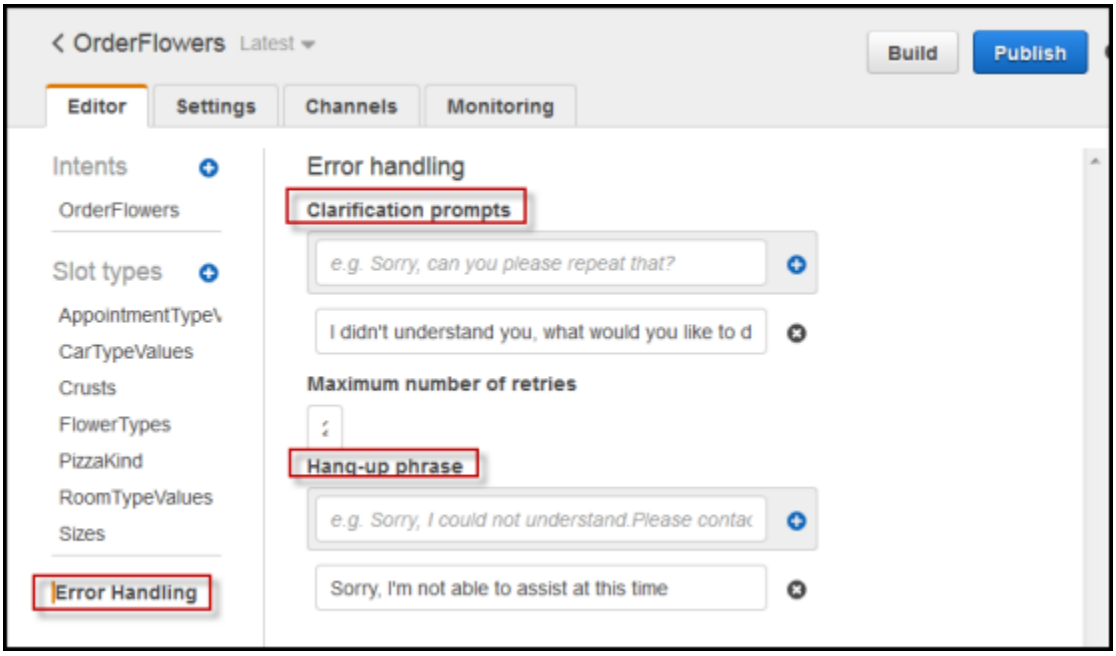
用於設定訊息的內容

建立機器人時，您可以在不同的內容中建立訊息，例如機器人中的釐清提示、槽值的提示，以及意圖的訊息。Amazon Lex 會在每個內容中選擇適當的訊息，以傳回給使用者。您可以針對每個內容提供一組訊息。如果您這樣做，Amazon Lex 會從群組隨機選擇一個訊息。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，您可以覆寫您在建置時設定的任何訊息。不過，使用其中任何訊息不需要 Lambda 函數。

機器人訊息

您可以使用釐清提示和工作階段結束訊息來設定機器人。在執行時間，如果 Amazon Lex 不了解使用者的意圖，則會使用釐清提示。您可以在傳送工作階段結束訊息之前，設定 Amazon Lex 請求釐清的次數。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。

如果您使用具有意圖的 Lambda 函數，Lambda 函數可能會傳回指示 Amazon Lex 詢問使用者意圖的回應。如果 Lambda 函數不提供這類訊息，Amazon Lex 會使用釐清提示。

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。在執行時間，Amazon Lex 會使用其中一個訊息來提示使用者提供槽的值。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。

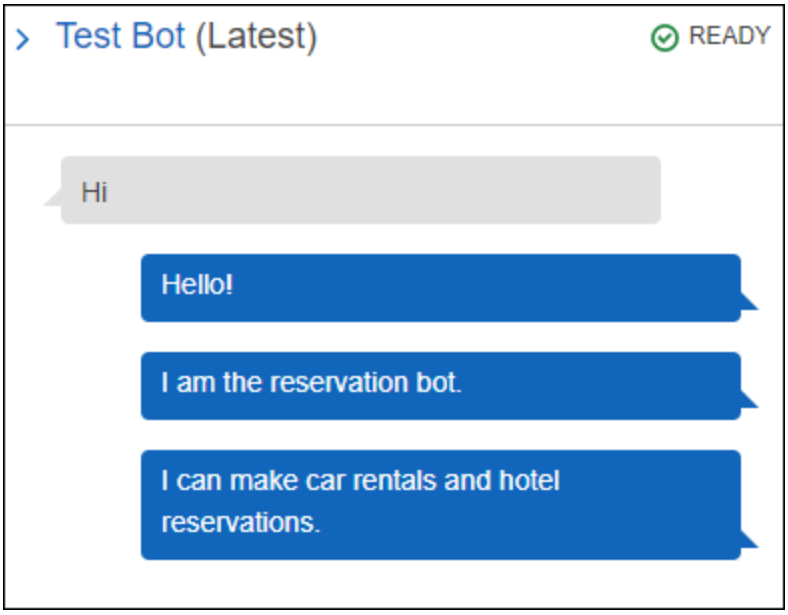
回應

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。在執行時間，Amazon Lex 會從每個訊息群組中選取一則訊息來建置回應。如需有關訊息群組的詳細資訊，請參閱訊息群組。

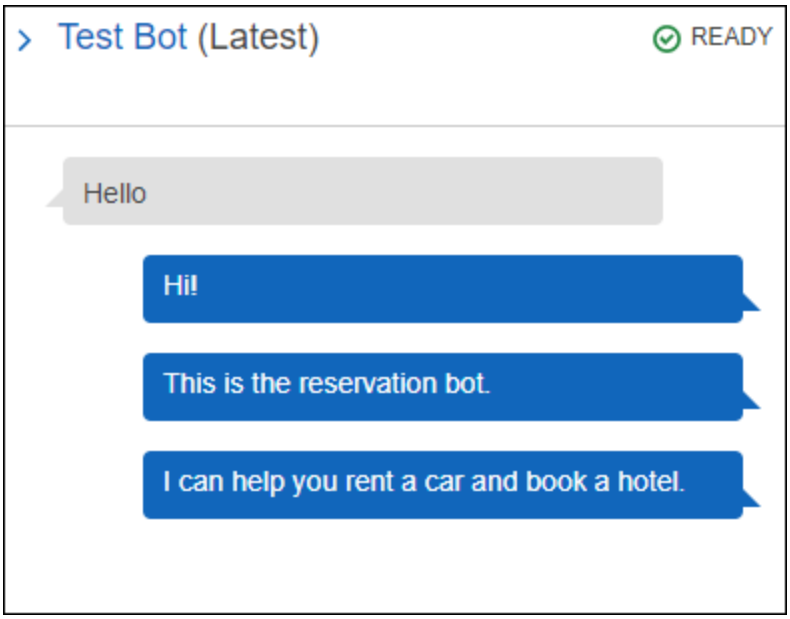
例如，您的第一個訊息群組可以包含不同的問候語：「哈囉」、「嗨」和「您好」。第二個訊息群組可以包含不同形式的簡介：「我是預約機器人」和「這是預約機器人。」 第三個訊息群組可以溝通機器

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，一個互動可以是下列項目：

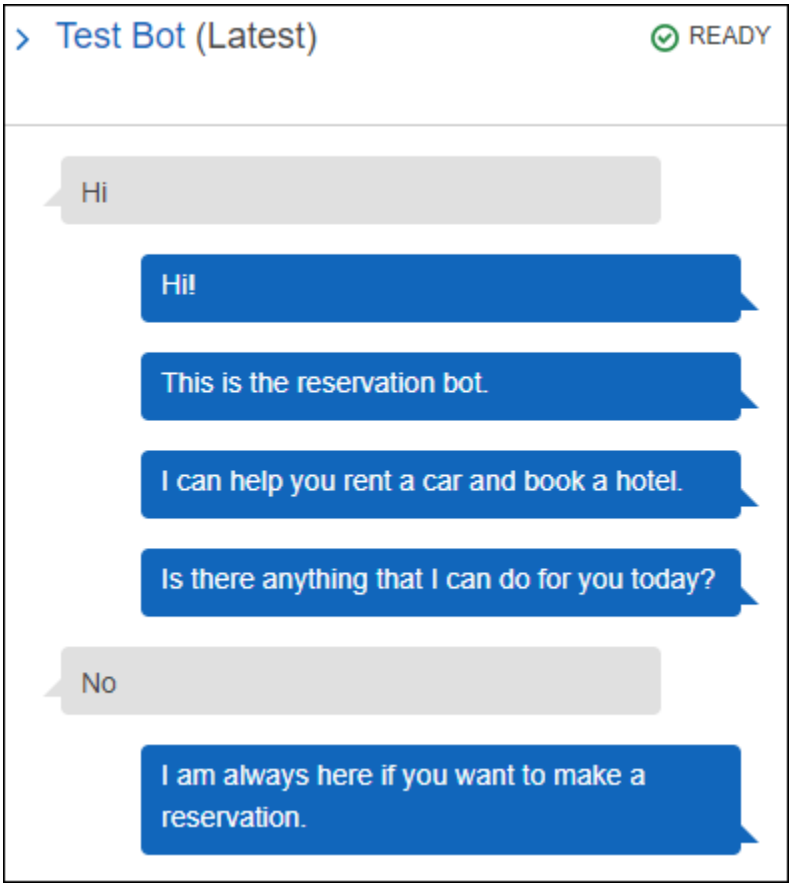


另一個可以是下列項目：



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。下圖提供範例：



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，請參閱下圖：

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，或當您將 PostContent操作的Accept標頭設定為 時text/plain;charset=utf8，Amazon Lex 支援下列格式的訊息：

- PlainText- 訊息包含純 UTF-8 文字。
- SSML- 訊息包含語音輸出的文字格式。
- 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，每個訊息群組各一個。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一訊息。例如，如果您定義五個訊息來引出槽值，Amazon Lex 會隨機選擇其中一個訊息並將其傳回給用戶端。

如果您希望 Amazon Lex 在執行時間請求中傳回特定類型的訊息給用戶端，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訊息，Amazon Lex 會隨機傳回一個訊息。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動態建置回應時，請使用訊息群組。當 Amazon Lex 傳回用戶端應用程式的回應時，會從每個群組隨機選擇一個訊息。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。如果您未建立訊息群組，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。

當 Amazon Lex 從群組傳送訊息時，回應的 `Message` 欄位包含包含訊息的逸出 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 — 訊息為 UTF-8 純文字。
- SSML — 訊息為語音合成標記語言 (SSML)。
- CustomPayload — 訊息為您指定的自訂格式。

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。

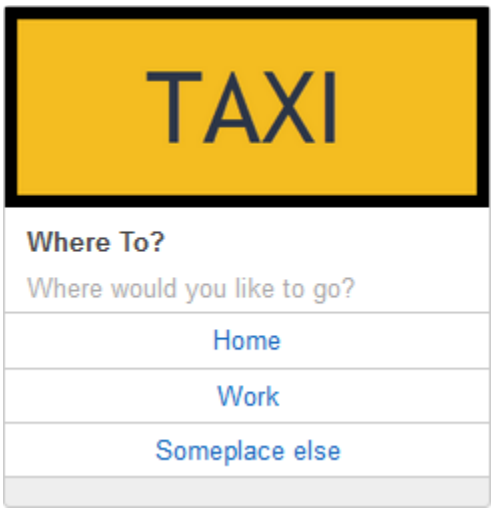
如需有關 x-amz-lex:accept-content-types 標頭的詳細資訊，請參閱設定回應類型。

回應卡

> **Note**
>
> 回應卡不適用於 Amazon Connect 聊天。不過，如需類似的功能，請參閱將互動式訊息新增至聊天。

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 傳送給用戶端應用程式的每個提示傳送回應卡。您可以搭配 Facebook Messenger、Slack、Twilio 和您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 會收到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。

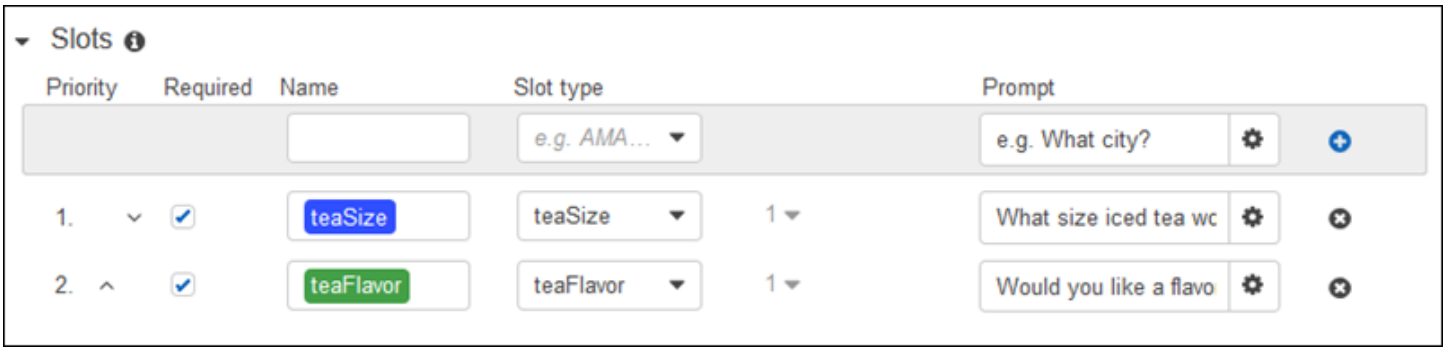
您是在設定意圖時建立回應卡。您可以在建置時間使用主控台或 PutIntent 操作定義靜態回應卡。或者，您可以在 Lambda 函數的執行時間定義動態回應卡。如果您同時定義靜態和動態回應卡，會以動態回應卡為優先。

Amazon Lex 會以用戶端了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，如果用戶端是 Facebook Messenger，Amazon Lex 會將回應卡轉換為一般範本。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。

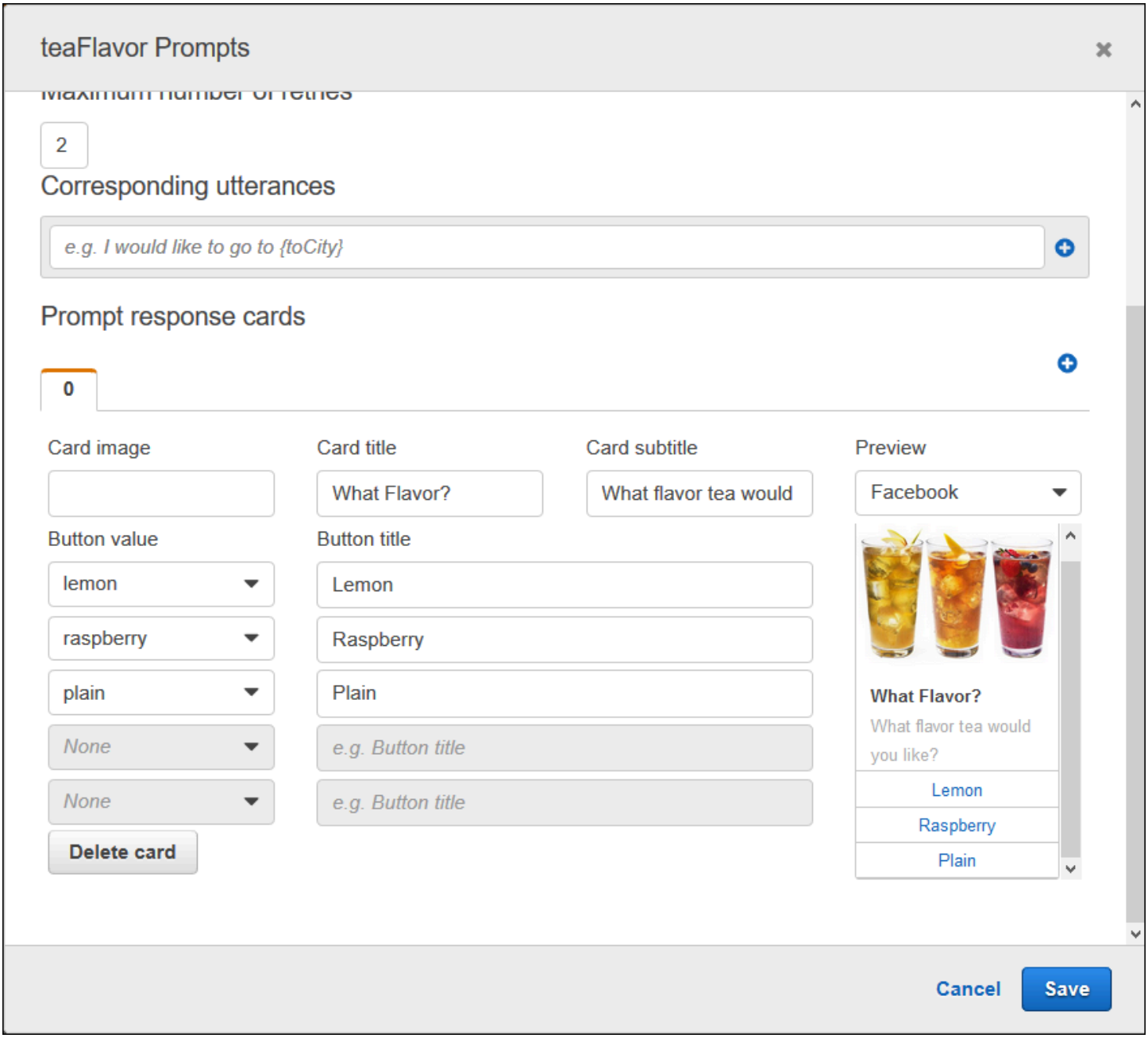
您只能搭配 PostText 操作使用回應卡。您無法搭配 PostContent 操作使用回應卡。

定義靜態回應卡

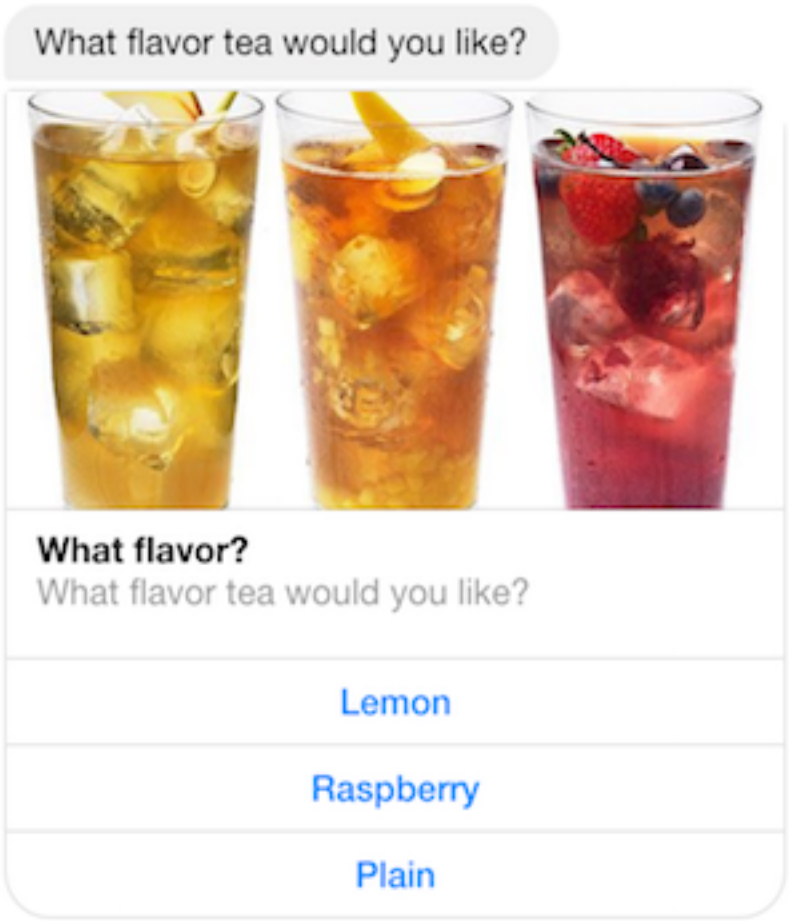
當您建立意圖時，使用 PutBot操作或 Amazon Lex 主控台定義靜態回應卡。靜態回應卡是與意圖同時定義。請在回應為固定時使用靜態回應卡。假設您要建立具有一個意圖的機器人，當中有個口味的槽。您在定義口味槽時指定提示，如以下主控台螢幕擷取畫面所示：



定義提示時，您可以選擇將回應卡與 PutBot操作建立關聯，並在 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。在執行時間，Amazon Lex 會將這些參考取代為工作階段屬性的適當值。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

若要在執行時間動態產生回應卡，請使用初始化和驗證 Lambda 函數做為意圖。在 Lambda 函數的執行時間決定回應時，請使用動態回應卡。為了回應使用者輸入，Lambda 函數會產生回應卡，並在回應的 `dialogAction`區段中傳回它。如需詳細資訊，請參閱回應格式。

以下是來自 Lambda 函數的部分回應，顯示 `responseCard`元素。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 排程預約。

管理對話內容

對話內容是使用者、您的應用程式或 Lambda 函數提供給 Amazon Lex 機器人以滿足意圖的資訊。對話內容包括使用者提供的槽資料、用戶端應用程式設定的請求屬性，以及用戶端應用程式和 Lambda 函數建立的工作階段屬性。

主題

- 設定意圖內容
- 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

設定意圖內容

您可以根據內容讓 Amazon Lex 觸發意圖。內容是狀態變數，可在您定義機器人時與意圖建立關聯。

當您使用主控台或使用 PutIntent操作建立意圖時，您可以設定意圖的內容。您只能在英文 (US) (en-US) 地區設定中使用內容，而且只有在您使用 PutBot操作建立機器人`true`時，才將 `enableModelImprovements` 參數設定為 。

內容、輸出內容和輸入內容有兩種關係類型。滿足相關聯的意圖時，輸出內容會變成作用中。輸出內容會在來自 PostText或 PostContent操作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。啟用內容後，它會在定義內容時設定的轉彎次數或時間限制內保持作用中狀態。

輸入內容會指定可辨識意圖的條件。只有當所有輸入內容都處於作用中狀態時，才能在對話期間辨識意圖。沒有輸入內容的意圖一律符合辨識資格。

Amazon Lex 會透過使用輸出內容滿足意圖，自動管理已啟用內容的生命週期。您也可以在呼叫 `PostContent`或 `PostText`操作時設定作用中內容。

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

例如，假設您有意預訂已設定為傳回名為 "book_car_fulfilled" 的輸出內容的租車。滿足意圖時，Amazon Lex 會設定輸出內容變數 "book_car_fulfilled"。由於 "book_car_fulfilled" 是作用中內容，只要使用者表達用語被識別為嘗試引出該意圖，「book_car_fulfilled」內容集為輸入內容的意圖現在就會視為辨識。您可以將此用於預訂租車後才有意義的意圖，例如透過電子郵件傳送收據或修改保留。

輸出內容

Amazon Lex 會在滿足意圖時啟用意圖的輸出內容。您可以使用輸出內容來控制符合追蹤目前意圖資格的意圖。

每個內容都有在工作階段中維護的參數清單。這些參數是已滿足意圖的槽值。您可以使用這些參數來預先填入其他意圖的槽值。如需詳細資訊，請參閱 使用預設槽值。

當您使用主控台或 PutIntent操作建立意圖時，您可以設定輸出內容。您可以使用多個輸出內容來設定意圖。滿足意圖時，所有輸出內容都會啟用，並在 PostText或 PostContent回應中傳回。

以下顯示使用主控台將輸出內容指派給意圖。

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周轉次數。轉彎是從您的應用程式到 Amazon Lex 的一個請求。一旦輪換次數或時間過期，內容就不再處於作用中狀態。

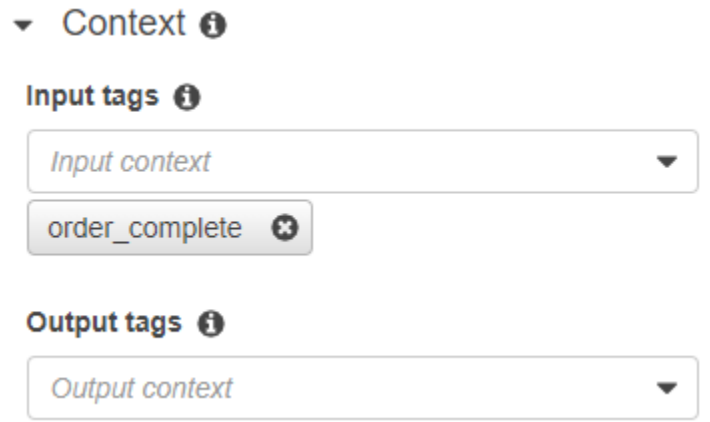
您的應用程式可以視需要使用輸出內容。例如，您的應用程式可以使用輸出內容來：

- 根據內容變更應用程式的行為。例如，旅遊應用程式的內容 "book_car_fulfilled" 可能具有與 "rental_function_fulfilled" 不同的動作。
- 將輸出內容傳回 Amazon Lex，做為下一個表達式的輸入內容。如果 Amazon Lex 將表達用語視為嘗試引出意圖，則會使用內容來限制可以傳回給具有指定內容的意圖。

輸入內容

您可以設定輸入內容來限制對話中辨識意圖的點。沒有輸入內容的意圖一律符合辨識資格。

您可以使用 主控台或 `PutIntent`操作來設定意圖回應的輸入內容。意圖可以有多個輸入內容。以下顯示使用主控台將輸入內容指派給意圖。



對於具有多個輸入內容的意圖，所有內容都必須處於作用中狀態，才能觸發意圖。您可以在呼叫 PostText、 PostContent或 PutSession操作時設定輸入內容。

您可以在 中設定槽，以從目前作用中內容取得預設值。當 Amazon Lex 辨識新意圖但未收到槽值時，會使用預設值。當您定義槽#context-name.parameter-name時，請以 形式指定內容名稱和槽名稱。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，您可以指定當使用者輸入未提供槽時，要為新意圖填入的槽值來源。此來源可以是先前的對話方塊、請求或工作階段屬性，或是您在建置時間設定的固定值。

您可以使用下列 做為預設值的來源。

- 上一個對話方塊 （ 內容 ） – #context-name.parameter-name
- 工作階段屬性 – 【attribute-name】
- 請求屬性 – <attribute-name>
- 固定值 – 任何不符合先前值的值

當您使用 PutIntent操作將插槽新增至意圖時，您可以新增預設值清單。預設值會依列出的順序使用。例如，假設您的意圖具有具有下列定義的槽：

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

辨識意圖時，名為 "reservation-start-date" 的槽會將其值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" 參數的值做為預設值。
2. 如果「book-car-fulfilled」內容未啟用，或未設定「startDate」參數，則會使用「reservationStartDate」工作階段屬性的值做為預設值。
3. 如果未使用前兩個預設值，則槽沒有預設值，Amazon Lex 會照常引出值。

如果槽使用預設值，即使需要，也不會引出槽。

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 會將新資訊傳遞回用戶端應用程式。例如：

- 在練習 1：使用藍圖建立 Amazon Lex 機器人（主控台）中，範例機器人使用 `price` 工作階段屬性，來維持花朵的價格。Lambda 函數會根據排序的花朵類型來設定此屬性。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 — 在比薩訂購機器人中，用戶端應用程式會在第一次呼叫 PostContent或 PostText操作時，以工作階段屬性的形式傳遞使用者的位置。例如 `"Location": "111 Maple Street"`。Lambda 函數會使用此資訊來尋找最接近的比薩來下訂單。
- 個人化提示 — 設定提示和回應卡以參考工作階段屬性。例如，「[FirstName] 你好，想要什麼配料？」 如果您以工作階段屬性 (`{"FirstName": "Jo"}`) 傳遞使用者的名字，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保留。Amazon Lex 會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 函數可以在回應中建立工作階段屬性。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在對 Amazon Lex 的請求中不包含`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`操作，Amazon Lex 會使用儲

存的工作階段屬性 () 呼叫 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至用戶端應用程式。

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，您可以建立屬性的string-to-string映射。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的資訊的方式。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 還提供訊息平台的預先定義屬性。如需該些屬性的清單，請參閱在訊息平台上部署 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- 訊息包含純 UTF-8 文字。
- SSML- 訊息包含語音輸出的文字格式。
- 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 只會在回應的 Message 欄位中傳回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
亞太地區 (新加坡)	Asia/Singapore
亞太地區 (雪梨)	Australia/Sydney
亞太地區 (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，如果使用者回應「您希望包裹在哪一天遞送？」tomorrow的提示。套件的實際交付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，而洛杉磯的人使用預設時區訂購了「明天」要交付的套件，則該套件將於 17 號而非 16 號交付。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，您可以建立屬性的string-to-string映射。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 - 槽資料和工作階段屬性 - 直到對話工作階段結束為止。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。如果使用者提供一些槽資料，但未完成購買鞋子，Amazon Lex 會記住整個工作階段的所有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。透過 AWS 命令列界面 (AWS CLI) 或 API，您在使用 PutBot 操作建立或更新機器人時，透過設定 idleSessionTTLInSeconds 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，其中包含訂單號碼。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

工作階段和請求屬性是屬性與值的字串至字串對應。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"
         }
      }
   ]
}
```

這個資料陣列不會妥當地轉譯為字串至字串對應。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，如果您使用 JavaScript，可以使用 `JSON.stringify` 操作將物件轉換成 JSON，以及使用 `JSON.parse` 操作將 JSON 文字轉換成 JavaScript 物件：

```
// 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);
```

若要透過 `PostContent` 操作傳送工作階段屬性，您必須先以 base64 編碼後再將其新增至請求標頭，如以下 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

當使用者表達表達表達的話，Amazon Lex 會使用自然語言理解 (NLU) 來了解使用者的請求並傳回適當的意圖。根據預設，Amazon Lex 會傳回機器人定義的最可能意圖。

在某些情況下，Amazon Lex 可能很難判斷最可能的意圖。例如，使用者可能會說出含糊不清的話語，或者可能有兩個相似的意圖。為了協助判斷適當的意圖，您可以將您的網域知識與替代意圖清單的可信度分數結合。可信度分數是 Amazon Lex 提供的評分，顯示意圖是正確意圖的可信度。

若要判斷兩種替代意圖之間的差異，您可以比較其可信度分數。例如，如果一個意圖的可信度分數為 0.95，而另一個意圖的可信度分數為 0.65，則第一個意圖可能是正確的。不過，如果一個意圖的分數為 0.75，而另一個意圖的分數為 0.72，則兩個意圖之間存在模棱兩可的情況，您可能可以在應用程式中使用網域知識來區分。

您也可以使用可信度分數來建立測試應用程式，以判斷意圖表達用語的變更是否會對機器人的行為產生影響。例如，您可以使用一組表達式取得機器人意圖的可信度分數，然後使用新的表達式更新意圖。然後，您可以檢查可信度分數，查看是否有改善。

Amazon Lex 傳回的可信度分數是比較值。您不應依賴它們做為絕對分數。這些值可能會根據 Amazon Lex 的改進而變更。

當您使用可信度分數時，Amazon Lex 會傳回最可能的意圖和最多 4 個替代意圖，以及每個回應中的相關分數。如果所有可信度分數都低於閾值，則 Amazon Lex 會包含 `AMAZON.FallbackIntent`、`AMAZON.KendraSearchIntent`或兩者，如果您已設定這些分數。您可以使用預設閾值，也可以設定自己的閾值。

下列 JSON 程式碼顯示 PostText操作回應中的 `alternativeIntents` 欄位。

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

建立或更新機器人時設定閾值。您可以使用 API 或 Amazon Lex 主控台。對於下列區域，您需要選擇加入，才能啟用準確性改善和可信度分數。在 主控台的進階選項區段中選擇可信度分數。使用 API，在呼叫 PutBot操作時設定 `enableModelImprovements` 參數。 ：

- 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)

在所有其他區域中，依預設提供準確性改善和可信度分數支援。

若要變更可信度閾值，請在主控台或使用 PutBot操作進行設定。閾值必須是介於 1.00 和 0.00 之間的數字。

若要使用 主控台，請在建立或更新機器人時設定可信度閾值。

在建立機器人時設定可信度閾值 （主控台）

- 在建立機器人上，在可信度分數閾值欄位中輸入值。

更新可信度閾值 （主控台）

1. 從機器人清單中，選擇要更新的機器人。
2. 選擇 Settings (設定) 標籤。
3. 在左側導覽中，選擇一般。
4. 更新可信度分數閾值欄位中的值。

設定或更新可信度閾值 (SDK)

- 設定 PutBot操作的 `nluIntentConfidenceThreshold` 參數。下列 JSON 程式碼顯示要設定的參數。

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

工作階段管理

若要變更 Amazon Lex 在與 使用者的對話中使用的意圖，您可以使用對話方塊程式碼勾點 Lambda 函數的回應，也可以在自訂應用程式中使用工作階段管理 APIs。

使用 Lambda 函數

當您使用 Lambda 函數時，Amazon Lex 會使用包含函數輸入內容的 JSON 結構來呼叫它。JSON 結構包含一個名為 的欄位`currentIntent`，其中包含 Amazon Lex 識別為使用者表達詞最可能意圖的意圖。JSON 結構也包含一個`alternativeIntents`欄位，其中包含最多四個額外的意圖，可滿足使用者的意圖。每個意圖都包含一個名為 的欄位`nluIntentConfidenceScore`，其中包含 Amazon Lex 指派給意圖的可信度分數。

若要使用替代意圖，請在 `ConfirmIntent`或 Lambda 函數的`ElicitSlot`對話方塊動作中指定它。

如需詳細資訊，請參閱使用 Lambda 函數。

使用工作階段管理 API

若要使用與目前意圖不同的意圖，請使用 PutSession操作。例如，如果您決定第一個替代方案優於 Amazon Lex 選擇的意圖，您可以使用 PutSession操作來變更意圖，以便使用者與之互動的下一個意圖是您選擇的意圖。

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，以及疑難排解對話的問題。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、音訊輸入的文字記錄和相關聯的中繼資料儲存在 CloudWatch Logs 中。音訊日誌將音訊輸入儲存在 Amazon S3 中。您可以使用 AWS KMS 客戶受管 CMKs 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 主控台中可用測試機器人的對話。啟用別名的對話日誌後，PostContent或該別名PostText的操作會將文字或音訊表達用語記錄在設定的 CloudWatch Logs 日誌群組或 S3 儲存貯體中。

主題

- 對話日誌的 IAM 政策
- 設定對話日誌
- 加密對話日誌
- 在 Amazon CloudWatch Logs 中檢視文字日誌
- 在 Amazon S3 中存取音訊日誌
- 使用 CloudWatch 指標監控對話日誌狀態

對話日誌的 IAM 政策

根據您選取的記錄類型，Amazon Lex 需要使用 Amazon CloudWatch Logs 和 Amazon Simple Storage Service (S3) 儲存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

若要啟用對話日誌，您必須授予 CloudWatch Logs 和 Amazon S3 的寫入許可。如果您為 S3 物件啟用物件加密，則需要將存取許可授予用於加密物件的 AWS KMS 金鑰。

您可以使用 IAM AWS 管理主控台、IAM API 或 AWS Command Line Interface 來建立角色和政策。這些指示使用 AWS CLI 來建立角色和政策。如需有關使用主控台建立政策的資訊，請參閱《AWS Identity and Access Management 使用者指南》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 (^)。

為對話日誌建立 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。此政策文件會將 Amazon Lex 新增為角色的信任實體。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。

   JSON

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

```
}
```

2. 在 中 AWS CLI，執行下列命令來建立對話日誌的 IAM 角色。

```
aws iam create-role \
    --role-name role-name \
    --assume-role-policy-document file://
LexConversationLogsAssumeRolePolicyDocument.json
```

接著，建立政策並將其連接至 角色，讓 Amazon Lex 能夠寫入 CloudWatch Logs。

建立將對話文字記錄到 CloudWatch Logs 的 IAM 政策

1. 在名為 的目前目錄中建立文件**LexConversationLogsCloudWatchLogsPolicy.json**，將下列 IAM 政策新增至其中，然後儲存它。
2. 在 中 AWS CLI，建立將寫入許可授予 CloudWatch Logs 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 政策以將音訊記錄到 S3 儲存貯體

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。

   JSON

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

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

2. 在 中 AWS CLI，建立將寫入許可授予 S3 儲存貯體的 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
```

授予許可以傳遞 IAM 角色

當您使用 主控台 AWS Command Line Interface、 或 AWS 開發套件來指定用於對話日誌的 IAM 角色時，指定對話日誌的使用者 IAM 角色必須具有將角色傳遞給 Amazon Lex 的許可。若要允許使用者將角色傳遞給 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。如需詳細資訊，請參閱《 AWS Identity and Access Management 使用者指南》中的授予使用者將角色傳遞至 AWS 服務和 IAM 和條件內容金鑰的許可。 AWS STS

JSON

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

```
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::111122223333:role/role-name",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "lex.amazonaws.com"
                },
                "StringLike": {
                    "iam:AssociatedResourceARN":
 "arn:aws:lex:region:123456789012: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 AWS 您的帳戶中使用 Amazon CloudWatch Logs 日誌群組。您可以使用任何有效的日誌群組。日誌群組必須與 Amazon Lex 機器人位於相同的區域。如需建立 CloudWatch Logs 日誌群組的詳細資訊，請參閱《Amazon CloudWatch Logs 使用者指南》中的使用日誌群組和日誌串流。

若要存放音訊日誌，請在 AWS 您的帳戶中使用 Amazon S3 儲存貯體。您可以使用任何有效的 S3 儲存貯體。儲存貯體必須與 Amazon Lex 機器人位於相同的區域。如需建立 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。

您必須為 IAM 角色提供政策，讓 Amazon Lex 能夠寫入已設定的日誌群組或儲存貯體。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 custom-suffix選項將自訂尾碼新增至角色，如下所示：

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

您用來啟用對話日誌的 IAM 角色必須具有 `iam:PassRole`許可。下列政策應連接至角色。

JSON

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

啟用對話日誌

使用主控台開啟日誌

1. 開啟 Amazon Lex 主控台 https：//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. 針對文字記錄，輸入 Amazon CloudWatch Logs 日誌群組名稱。
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 CloudWatch Logs 日誌群組的 Amazon Resource Name (ARN)，這是日誌的目的地

2. 將 conversationLogs 欄位iamRoleArn的成員設定為 IAM 角色的 Amazon Resource Name (ARN)，該角色具有在指定資源上啟用對話日誌所需的許可。

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，請設定用於加密之金鑰的 ARN kmsKeyArn成員。

2. 將 conversationLogs 欄位iamRoleArn的成員設定為 IAM 角色的 Amazon Resource Name (ARN)，該角色具有在指定資源上啟用對話日誌所需的許可。

停用對話日誌

使用主控台關閉日誌

1. 開啟 Amazon Lex 主控台 https：//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 CMKs 來加密 CloudWatch Logs 日誌群組和 S3 儲存貯體中的資料。

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

您可以在 Amazon Lex 用於文字日誌的 CloudWatch Logs 日誌群組上使用 AWS KMS 金鑰啟用加密。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 Logs 使用者指南》中的使用 加密 CloudWatch Logs AWS KMS中的日誌資料。 Amazon CloudWatch

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 金鑰來加密音訊物件。如需詳細資訊，請參閱《Amazon S3 Simple Storage Service 開發人員指南》中的適用於 S3 儲存貯體的 Amazon S3 預設加密。

如果您選擇加密音訊日誌，Amazon Lex 需要 AWS KMS 許可。您需要將其他政策連接至用於對話日誌的 IAM 角色。如果您在 S3 儲存貯體上使用預設加密，您的政策必須授予對該儲存貯體所設定 AWS KMS 金鑰的存取權。如果您在音訊日誌設定中指定 AWS KMS 金鑰，則必須授予該金鑰的存取權。

如果您未建立對話日誌的角色，請參閱對話日誌的 IAM 政策。

使用 AWS KMS 金鑰來加密音訊日誌，以建立 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。
2. 在 中 AWS CLI，建立授予許可的 IAM 政策，以使用 AWS KMS 金鑰來加密音訊日誌。

```
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
```

在 Amazon CloudWatch Logs 中檢視文字日誌

Amazon Lex 會將對話的文字日誌儲存在 Amazon CloudWatch Logs 中。若要檢視日誌，您可以使用 CloudWatch Logs 主控台或 API。如需詳細資訊，請參閱《Amazon CloudWatch Logs 使用者指南》中的使用篩選條件模式搜尋日誌資料和 CloudWatch Logs Insights 查詢語法。 CloudWatch Amazon CloudWatch

使用 Amazon Lex 主控台檢視日誌

1. 開啟 Amazon Lex 主控台 https：//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. 選擇文字日誌下方的連結，以在 CloudWatch 主控台中檢視別名的日誌。

您也可以使用 CloudWatch 主控台或 API 來檢視您的日誌項目。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字首。

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。
- 只有在 AMAZON.KendraSearchIntent提出搜尋 Amazon Kendra 索引的請求時，才會出現 kendraResponse 欄位。
- 在機器人的 Lambda 函數中指定替代意圖時， developerOverride 欄位為 true。
- 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 Logs。如需更多資訊，請參閱對話日誌的 IAM 政策。

在 Amazon S3 中存取音訊日誌

Amazon Lex 會將對話的音訊日誌存放在 S3 儲存貯體中。

使用主控台存取音訊日誌

1. 開啟 Amazon Lex 主控台 https：//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. 選擇音訊日誌下方的連結，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 來存取音訊日誌。您可以在 Amazon Lex 主控台或`GetBotAlias`操作回應的 `resourcePrefix` 欄位中看到音訊檔案的 S3 物件金鑰字首。

使用 CloudWatch 指標監控對話日誌狀態

使用 Amazon CloudWatch 監控對話日誌的交付指標。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` 命名空間中為對話日誌提供四個指標：

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

如需詳細資訊，請參閱對話日誌的 CloudWatch 指標。

成功指標顯示 Amazon Lex 已成功將您的音訊或文字日誌寫入其目的地。

失敗指標顯示 Amazon Lex 無法將音訊或文字日誌交付至指定的目的地。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：

- 確定 Amazon Lex 是 IAM 角色的受信任實體。
- 對於文字記錄，請確定 CloudWatch Logs 日誌群組存在。對於音訊記錄，確定 S3 儲存貯體存在。
- 確定 Amazon Lex 用來存取 CloudWatch Logs 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許可。
- 請確定 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 AWS KMS 金鑰進行 S3 加密，請確定沒有防止 Amazon Lex 使用您的金鑰的政策，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，請參閱對話日誌的 IAM 政策。

使用 Amazon Lex API 管理工作階段

當使用者開始與您的機器人對話時，Amazon Lex 會建立工作階段。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。首先，使用 Lambda 函數搭配 `PostContent`或 `PostText`操作，該操作會在每次對話後呼叫。如需詳細資訊，請參閱使用 Lambda 函數。另一個是使用應用程式中的 Amazon Lex 執行期 API 來變更工作階段狀態。

Amazon Lex 執行時間 API 提供可讓您管理工作階段資訊以與機器人對話的操作。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。將對話方塊動作`type`欄位設定為 `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，並使用此資訊來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。

工作階段操作會與您的履行 Lambda 函數搭配使用。例如，如果您的 Lambda 函數傳回 Failed做為履行狀態，您可以使用 PutSession操作，將對話方塊動作類型設定為 close，並將對話方塊動作類型設定為 ReadyForFulfillment fulfillmentState，以重試履行步驟。

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 將使用新意圖開始與使用者的對話。

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，讓 Amazon Lex 只傳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 將以您的歡迎意圖中的提示回應，以開始與您的使用者的對話。

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，如果您想要變更 Amazon Lex 用來引發槽值的訊息，您可以設定 message和 messageFormat 欄位。

機器人部署選項

目前，Amazon Lex 提供下列機器人部署選項：

- AWS Mobile SDK – 您可以使用 AWS Mobile SDK 建置與 Amazon Lex 通訊的行動應用程式。SDKs
- Facebook Messenger – 您可以將 Facebook Messenger 頁面與 Amazon Lex 機器人整合，讓 Facebook 上的最終使用者可以與機器人通訊。在目前的實作中，這項整合僅支援文字輸入訊息。
- Slack – 您可以將 Amazon Lex 機器人與 Slack 訊息應用程式整合。
- Twilio – 您可以將 Amazon Lex 機器人與 Twilio Simple Messaging Service (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 可讓您使用標準內建意圖和槽類型。

主題

內建意圖

對於常見動作，您可以使用標準內建意圖程式庫。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，例如範例表達用語。

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表達用語
- 設定內建意圖的槽

將內建意圖新增至機器人

1. 登入 AWS 管理主控台 ，並在 https://console.aws.amazon.com/lex/：// 開啟 Amazon Lex 主控台。
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. 對於複製內建意圖，請為意圖命名，然後選擇新增。
7. 視需要為您的機器人設定意圖。

主題

- AMAZON.StartOverIntent
- AMAZON.StopIntent

> **Note**
>
> 對於英文 (US) (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 不支援下列意圖：
>
> - `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單字和片語。您的應用程式可以使用此意圖來移除槽類型值和其他屬性，然後再結束與使用者的互動。

常見表達用語：

- 取消
- 永不介意
- 忘記它

AMAZON.FallbackIntent

當使用者對意圖的輸入不是機器人預期的輸入時，您可以設定 Amazon Lex 叫用備用意圖。例如，如果使用者輸入「我想要訂購 Candy」不符合`OrderFlowers`機器人中的意圖，Amazon Lex 會叫用備用意圖來處理回應。

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。

Amazon Lex 在這些情況下符合備用意圖：

- 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 函數
- 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備用意圖，Amazon Lex 會使用備用意圖。如果您需要機器人具有取消陳述式，您可以使用履行函數進行備用意圖，以提供與取消陳述式相同的行為。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釐清提示，且使用者未以有效意圖開始對話，Amazon Lex 會立即呼叫您的備用意圖 。

當您在沒有釐清提示的情況下使用備用意圖時，Amazon Lex 不會在這些情況下呼叫備用：

- 在使用者回應後續提示，但不提供意圖時。例如，為了回應表示「您今天是否想要其他項目？」的後續提示，使用者表示「是」。Amazon Lex 傳回 400 錯誤的請求例外狀況，因為它沒有要傳送給使用者的釐清提示以取得意圖。
- 使用 AWS Lambda 函數時，您會傳回`ElicitIntent`對話方塊類型。由於 Amazon Lex 沒有向使用者取得意圖的釐清提示，因此會傳回 400 錯誤的請求例外狀況。
- 在使用 `PutSession` 操作時，您會傳送 `ElicitIntent` 對話方塊類型。由於 Amazon Lex 沒有向使用者取得意圖的釐清提示，因此會傳回 400 錯誤的請求例外狀況。

使用 Lambda 函數搭配備用意圖

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。

如果您在備用意圖中使用履行 Lambda 函數，您可以使用此函數來呼叫另一個意圖，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。

您可以在備用意圖 Lambda 函數中執行任何動作，您可以在履行函數中針對任何其他意圖執行這些動作。如需使用 建立履行函數的詳細資訊 AWS Lambda，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 `ElicitIntent` 對話方塊動作來提示使用者使用不同的意圖。如果 Amazon Lex 無法在設定的嘗試次數之後推斷使用者的意圖，則會再次叫用備用意圖。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備用意圖的次數。如果呼叫 Lambda 函數的次數超過您在 Lambda 函數中設定的閾值，您的 Lambda 函數可以採取不同的動作。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

回應指出使用者在與機器人互動時需要幫助的單字或片語。調用此意圖時，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、詢問有關協助領域的後續問題，或將互動交給人力客服人員。

常見表達用語：

- 說明
- 協助我
- 您可以協助我

AMAZON.KendraSearchIntent

若要搜尋您已使用 Amazon Kendra 編製索引的文件，請使用 AMAZON.KendraSearchIntent意圖。當 Amazon Lex 無法判斷與使用者對話中的下一個動作時，它會觸發搜尋意圖。

AMAZON.KendraSearchIntent 僅適用於英文 (US) (en-US) 地區設定和美國東部 （維吉尼亞北部）、美國西部 （奧勒岡） 和歐洲 （愛爾蘭） 區域。

Amazon Kendra 是以machine-learning-based搜尋服務，可將 PDF 文件或 Microsoft Word 檔案等自然語言文件編製索引。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 索引建立常見問答集機器人。

如果您為機器人設定AMAZON.KendraSearchIntent意圖，Amazon Lex 會在無法判斷槽或意圖的使用者表達用語時呼叫意圖。例如，如果您的機器人針對名為「比薩填入」的槽類型發出回應，而使用者說「什麼是比薩？」，Amazon Lex 會呼叫 AMAZON.KendraSearchIntent 來處理問題。如果 Amazon Kendra 沒有回應，則對話會繼續如機器人中所設定。

當您在相同的機器人AMAZON.FallbackIntent中同時使用 AMAZON.KendraSearchIntent和時，Amazon Lex 會使用意圖，如下所示：

1. Amazon Lex 會呼叫 AMAZON.KendraSearchIntent。意圖會呼叫 Amazon Kendra Query操作。
2. 如果 Amazon Kendra 傳回回應，Amazon Lex 會將結果顯示給使用者。
3. 如果 Amazon Kendra 沒有回應，Amazon Lex 會重新提示使用者。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 辨識的表達用語，例如填入槽值或確認意圖，則與使用者的對話會依照機器人的設定繼續進行。
   - 如果使用者的回應不包含 Amazon Lex 辨識的表達用語，Amazon Lex 會再次呼叫 Query操作。
4. 如果在設定的重試次數之後沒有回應，Amazon Lex 會呼叫 AMAZON.FallbackIntent並結束與使用者的對話。

有三種方式可以使用 AMAZON.KendraSearchIntent 向 Amazon Kendra 提出請求：

- 讓搜尋意圖為您提出請求。Amazon Lex 以使用者的表達用語做為搜尋字串呼叫 Amazon Kendra。建立意圖時，您可以定義查詢篩選條件字串，以限制 Amazon Kendra 傳回的回應數目。Amazon Lex 會在查詢請求中使用篩選條件。
- 將其他查詢參數新增至請求，以使用對話方塊 Lambda 函數縮小搜尋結果範圍。您可以將包含 Amazon Kendra 查詢參數kendraQueryFilterString的欄位新增至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，它們優先於您在建立意圖時定義的查詢篩選條件。
- 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 Amazon Kendra 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，您可以指定字串，做為 Amazon Kendra 查詢的屬性篩選條件。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。如需屬性篩選條件的詳細資訊，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 傳送給 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 索引建立常見問答集機器人。

Amazon Kendra 搜尋的 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 擔任具有呼叫 Amazon Kendra Query意圖許可的執行期角色。您使用的 IAM 設定取決於您使用 Amazon Lex AMAZON.KendraSearchIntent 主控台或使用 AWS 開發套件或 AWS Command Line Interface () 建立AWS CLI。使用主控台時，您可以選擇將呼叫 Amazon Kendra 的許可新增至 Amazon Lex 服務連結角色，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，您必須特別使用角色來呼叫Query操作。

連接許可

您可以使用 主控台將存取 Amazon Kendra Query操作的許可連接到預設的 Amazon Lex 服務連結角色。當您將許可連接到服務連結角色時，您不需要特別建立和管理執行期角色，即可連線到 Amazon Kendra 索引。

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 政策連接至主控台存取角色。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。

JSON

```
{
    "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，指定呼叫 Amazon Kendra Query操作時要使用的執行時間角色。

您用來指定執行期角色的使用者、角色或群組必須具有 iam:PassRole許可。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《 使用者指南》中的 IAM 和 AWS STS 條件內容金鑰。 AWS Identity and Access Management

JSON

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

Amazon Lex 呼叫 Amazon Kendra 所需的執行時間角色必須具有 kendra:Query許可。當您將現有的 IAM 角色用於呼叫 Amazon Kendra Query操作的許可時，該角色必須連接下列政策。

您可以使用 IAM 主控台、IAM API 或 AWS CLI 來建立政策，並將其連接至角色。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
2. 在 中 AWS CLI，執行下列命令來建立執行 Amazon Kendra Query操作的 IAM 政策。

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

3. 將政策連接至您用來呼叫 Query操作的 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，或使用您在AMAZON.KendraSearchIntent為機器人建立 時建立的角色。下列程序說明如何選擇要使用的 IAM 角色。

指定 AMAZON.KendraSearchIntent 的執行時間角色。

1. 登入 AWS 管理主控台 ，並在 https://console.aws.amazon.com/lex/ ：// 開啟 Amazon Lex 主控台。
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，讓機器人查詢 Amazon Kendra 索引，請選擇新增 Amazon Kendra 許可。
   - 若要使用具有呼叫 Amazon Kendra Query操作許可的角色，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 對目前對話相關項目的回應，請在建立機器人時新增 queryFilterString 參數，以使用工作階段和請求屬性做為篩選條件。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 之前取代值。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 參數的範例，該queryFilterString參數使用名為 的工作階段屬性"SourceURI"來篩選 Amazon Kendra 查詢。

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

以下是 參數的範例，該queryFilterString參數使用名為 的請求屬性"DepartmentName"來篩選 Amazon Kendra 查詢。

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

AMAZON.KendraSearchInteng 篩選條件使用與 Amazon Kendra 搜尋篩選條件相同的格式。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 使用的查詢篩選條件字串AMAZON.KendraSearchIntent，必須針對每個篩選條件的第一個字母使用小寫字母。例如，以下是 的有效查詢篩選條件AMAZON.KendraSearchIntent。

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

```
                "value": {
                    "stringValue": "Washington"
                }
            }
        }
    ]
}
```

使用搜尋回應

Amazon Kendra 傳回對意圖conclusion陳述式中搜尋的回應。除非履行 Lambda 函數產生結論訊息，否則意圖必須具有conclusion陳述式。

Amazon Kendra 有四種類型的回應。

- x-amz-lex:kendra-search-response-question_answer-question-<N> – 來自符合搜尋之常見問答集的問題。
- x-amz-lex:kendra-search-response-question_answer-answer-<N> – 符合搜尋之常見問答集的答案。
- x-amz-lex:kendra-search-response-document-<N> – 索引中與表達用語文字相關的文件摘錄。
- x-amz-lex:kendra-search-response-document-link-<N> – 索引中與表達用語文字相關的文件 URL。
- x-amz-lex:kendra-search-response-answer-<N> – 來自索引中回答問題之文件的摘錄。

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，這些變數由 Amazon Kendra 回應中的請求屬性取代。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 Amazon Kendra 的任何回應：

- 「我為您找到了一個常見問題項目：((x-amz-lex:kendra-search-response-question_answer-question-1)), and the answer is ((x-amz-lex:kendra-search-response-question_answer-answer-1))」
- 「我找到了一份有用文件的摘錄：((x-amz-lex:kendra-search-response-document-1))」

- 「我認為您的問題的答案是 ((x-amz-lex:kendra-search-response-answer-1))」

使用 Lambda 函數管理請求和回應

AMAZON.KendraSearchIntent 意圖可以使用您的對話方塊程式碼掛勾和履行程式碼掛勾來管理對 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鉤 Lambda 函數，而當您想要修改回應時，請使用履行程式碼掛鉤 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勾來建立要傳送至 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掛勾，Amazon Lex 會從使用者表達用語建構查詢，並在您提供意圖時queryFilterString，使用您在設定意圖時提供的 。

您可以使用對話方塊程式碼掛接回應中的兩個欄位來修改對 Amazon Kendra 的請求：

- kendraQueryFilterString – 使用此字串來指定 Amazon Kendra 請求的屬性篩選條件。您可以使用索引中定義的任何索引欄位來篩選查詢。如需篩選條件字串的結構，請參閱《Amazon Kendra 開發人員指南》中的使用文件屬性來篩選查詢。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 – 使用此字串來指定 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。

建立篩選條件或查詢字串之後，您會將回應傳送至 Amazon Lex，並將回應的 dialogAction 欄位設定為 delegate。Amazon Lex 會將查詢傳送至 Amazon Kendra，然後將查詢回應傳回至履行程式碼掛勾。

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，查詢回應會傳回至AMAZON.KendraSearchIntent履行 Lambda 函數。程式碼掛鉤的輸入事件包含來自 Amazon Kendra 的完整回應。查詢資料與 Amazon Kendra Query操作傳回的查詢資料位於相同的結構中。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，或程式碼掛鉤未傳回回應中的訊息，Amazon Lex 會使用conclusion陳述式進行回應。

範例：為 Amazon Kendra 索引建立常見問答集機器人

此範例會建立使用 Amazon Kendra 索引的 Amazon Lex 機器人，以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，並將其設定為使用 Amazon Kendra 索引。
4. 透過詢問存放在 Amazon Kendra 索引中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 （ 主控台 ） 入門。

建立常見問題機器人

1. 登入 AWS 管理主控台 ，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 – 為機器人命名，指出其用途，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。
   - COPPA – 選擇否。
   - 使用者表達用語儲存 – 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 機器人時需要此意圖，但不會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. 選擇儲存意圖。

現在，建立意圖來搜尋 Amazon Kendra 索引及其應傳回的回應訊息。

建立 AMAZON.KendraSearchIntent 意圖和回應訊息

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. 在搜尋意圖方塊中，輸入 **AMAZON.KendraSearchIntent**，然後從清單中選擇它。
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

回應單字和片語，讓使用者暫停與機器人的互動，以便稍後可以返回。您的 Lambda 函數或應用程式需要在工作階段變數中儲存意圖資料，或者您需要在繼續目前意圖時使用 GetSession操作來擷取意圖資料。

常見表達用語：

- 暫停
- 暫停

AMAZON.RepeatIntent

回應可讓使用者重複上一個訊息的單字和片語。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 GetSession操作來取得先前的意圖資訊。

常見表達用語：

- 重複
- 再說一次
- 重複

AMAZON.ResumeIntent

回應單字和片語可讓使用者繼續先前暫停的意圖。您的 Lambda 函數或應用程式必須管理恢復先前意圖所需的資訊。

常見表達用語：

- 繼續
- 繼續
- 繼續

AMAZON.StartOverIntent

回應讓使用者停止處理目前意圖並從頭開始的單字和片語。您可以使用 Lambda 函數或 `PutSession`操作再次引出第一個槽值。

常見表達用語：

- 重新開始
- 重新啟動
- 再次啟動

AMAZON.StopIntent

回應指出使用者想要停止處理目前意圖並結束與機器人互動的單字和片語。您的 Lambda 函數或應用程式應該清除任何現有的屬性和槽類型值，然後結束互動。

常見表達用語：

- stop
- off
- 關閉

內建槽類型

Amazon Lex 支援內建槽類型，可定義如何辨識和處理槽中的資料。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	
AMAZON.Airport	辨識代表機場的單字。	所有地區設定	
AMAZON.AlphaNumeric	辨識由字母和數字組成的字詞。	韓文以外的所有地區設定 (ko-KR)	
AMAZON.City	辨識代表城市的單字。	所有地區設定	

槽類型	簡短描述	支援的地區設定
AMAZON.Country	識別代表國家/地區的單字。	所有地區設定
AMAZON.DATE	識別代表日期的單字，並將其轉換為標準格式。	所有地區設定
AMAZON.DURATION	識別代表持續時間的單字，並將其轉換為標準格式。	所有地區設定
AMAZON.EmailAddress	識別代表電子郵件地址的單字，並將其轉換為標準電子郵件地址。	所有地區設定
AMAZON.FirstName	辨識代表名字的單字。	所有地區設定
AMAZON.LastName	識別代表姓氏的單字。	所有地區設定
AMAZON.NUMBER	識別數字單字並將其轉換為數字。	所有地區設定
AMAZON.Percentage	識別代表百分比的單字，並將其轉換為數字和百分比符號 (%)。	所有地區設定
AMAZON.PhoneNumber	識別代表電話號碼的單字，並將其轉換為數字字串。	所有地區設定
AMAZON.SpeedUnit	識別代表速度單位的單字，並將其轉換為標準縮寫。	英文 (美國) (en-US)

槽類型	簡短描述	支援的地區設定	
AMAZON.State	識別代表狀態的單字。	所有地區設定	
AMAZON.StreetName	識別代表街道名稱的單字。	除了英文 (US) (en-US) 之外的所有地區設定	
AMAZON.TIME	辨識指出時間的單字，並將其轉換為時間格式。	所有地區設定	
AMAZON.WeightUnit	識別代表權重單位的單字，並將其轉換為標準縮寫	英文 (美國) (en-US)	

> **Note**
>
> 對於英文 (US) (en-US) 地區設定，Amazon Lex 支援 Alexa 技能套件中的槽類型。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。

AMAZON.Airport

提供機場清單。範例包括：

- 約翰甘乃迪納國際機場
- 墨爾本機場

AMAZON.AlphaNumeric

辨識由字母和數字組成的字串，例如 **APQ123**。

此槽類型不適用於韓文 (ko-KR) 地區設定。

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

使用標準規則表達式。Amazon Lex 在規則表達式中支援下列字元：

- A-Z、a-z
- 0-9

Amazon Lex 也支援規則表達式中的 Unicode 字元。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 (.)

規則表達式的長度上限為 300 個字元。使用規則表達式存放在 AMAZON.AlphaNumeric 槽類型中的字串最大長度為 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。如果表達式無效，Amazon Lex 會傳回錯誤訊息。

當您在槽類型中使用規則表達式時，Amazon Lex 會根據規則表達式檢查該類型的槽輸入。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。

AMAZON.City

提供當地和世界城市的清單。插槽類型可識別城市名稱的常見變化。Amazon Lex 不會從變異轉換為正式名稱。

範例：

- 紐約
- 雷克雅維克
- 東京
- 凡爾賽

AMAZON.Country

全球國家/地區的名稱。範例：

- 澳洲
- 德國
- 日本
- 美國
- 烏拉圭

AMAZON.DATE

將代表日期的單字轉換為日期格式。

日期會以 ISO-8601 日期格式提供給您的意圖。您的意圖在槽中收到的日期可能會因使用者說出的特定片語而有所不同。

- 對應至特定日期的方位，例如「今天」、「現在」或「11 月二十五日」，會轉換為完整日期：2020-11-25。這預設為目前日期或之後的日期。

- 對應至特定週的方位，例如「本週」或「下週」，會轉換為一週第一天的日期。在 ISO-8601 格式中，一週從星期一開始，到星期日結束。例如，如果今天是 2020-11-25，則「下週」會轉換為 2020-11-30。
- 對應到一個月，但不是特定日期的張量，例如「下個月」會轉換為該月的最後一天。例如，如果今天是 2020-11-25，則「下個月」會轉換為 2020-12-31。
- 對應至一年，但不是特定月份或日期的方位，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」會轉換為 2021-12-31。

AMAZON.DURATION

將指示持續時間的單字轉換為數值持續時間。

持續時間會解析為以 ISO-8601 持續時間格式 為基礎的格式PnYnMnWnDTnHnMnS。P 表示這是持續時間， n是數值，而 後面的大寫字母n是特定的日期或時間元素。例如， P3D表示 3 天。T 用於表示剩餘值代表時間元素，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
- 「八週」： P8W
- 「七年」： P7Y
- 「五小時十分鐘」： PT5H10M
- 「兩年三小時十分鐘」： P2YT3H10M

AMAZON.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 (+)。

AMAZON.FirstName

常用的名字。此槽類型可辨識正式名稱和非正式暱稱。傳送到您意圖的名稱是使用者傳送的值。Amazon Lex 不會從 nick 名稱轉換為正式名稱。

對於聽起來類似但拼寫不同的名字，Amazon Lex 會傳送單一通用表單給您的意圖。

在英文 (US) (en-US) 地區設定中，使用槽名稱 AMAZON.US_First_Name。

範例：

- Emily
- John
- 索菲

AMAZON.LastName

常用的姓氏。對於聽起來像是拼法不同的名稱，Amazon Lex 會傳送單一通用表單給您的意圖。

在英文 (US) (en-US) 地區設定中，使用槽名稱 AMAZON.US_Last_Name。

範例：

- 布魯斯基
- 大賽爾文
- Evers
- 剖析
- Welt

AMAZON.NUMBER

將表達數字的單字或數字轉換為數字，包括小數。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

Input	回應
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%	0.4%
23.5%	23.5%
百分之二十五	25%

AMAZON.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。

Type	說明	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

Type	說明	Input	結果
			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

AMAZON.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 (US) (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
每秒公尺數、mps、MPS、m/s	mps
每小時海哩數、節	節

AMAZON.State

國家/地區內的地理和政治區域名稱。

範例：

- 巴瓦利亞
- 福島縣
- 太平洋西北部

- 多倫多
- 威爾斯

AMAZON.StreetName

典型街道地址內的街道名稱。這只包含街道名稱，不包含房屋號碼。

此插槽類型不適用於英文 (US) (en-US) 地區設定。

範例：

- 坎培拉大街
- 前街
- 市場道路

AMAZON.TIME

將代表時間的字詞轉換成時間值。包含不明確時間的解析度。當使用者進入模棱兩可的時間時，Amazon Lex 會使用 Lambda 事件的 `slotDetails` 屬性，將模棱兩可時間的解析度傳遞給您的 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。在此情況下，`slots`映射中的值為 `null`，而`slotDetails`實體包含兩個可能的 時間解析度。Amazon Lex 會將下列項目輸入 Lambda 函數：

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

當使用者以不明確的時間回應時，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「10：00 PM」回應交付時間的提示，Amazon Lex 會將以下內容輸入 Lambda 函數：

```
"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。

AMAZON.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 (US) (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
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。槽類型是 Amazon Lex 用來訓練機器學習模型以辨識槽值的值清單。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 會維護插槽可能解析度的清單。清單中的每個項目都會提供解析度值，Amazon Lex 會將其視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 Lambda 函數，函數的輸入事件會包含稱為 的解析清單`slotDetails`。下列範例顯示 Lambda 函數輸入的槽和槽詳細資訊區段：

```
"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

Amazon Lex 可讓您混淆或隱藏插槽的內容，使內容不可見。若要保護擷取為槽值的敏感資料，您可以啟用槽混淆來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}
```

如果表達式包含括號字元 ({})，Amazon Lex 會以兩個反斜線 (\\) 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 PostContent和 PostText操作的回應，而槽值可用於您的驗證和履行 Lambda 函數。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，如果 Amazon Lex 在表達式中辨識出槽值和槽值，則會混淆槽值。如果沒有識別到槽值，Amazon Lex 不會混淆表達式。

在第二輪和之後輪換時，Amazon Lex 知道要引出的槽，以及槽值是否應該混淆。如果 Amazon Lex 辨識到槽值，則會混淆該值。如果 Amazon Lex 無法辨識值，則會混淆整個表達式。遺漏表達用語中的任何槽值都不會被混淆。

Amazon Lex 也不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 來選擇哪些插槽混淆。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 的回應指出文字的整體情緒是正面、中性、負面或混合。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， `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 代表您呼叫 Amazon Comprehend，以判斷機器人處理的每個表達用語中的情緒。透過啟用情緒分析，即表示您同意 Amazon Comprehend 的服務條款和協議。如需 Amazon Comprehend 定價的詳細資訊，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通道，您可以將中繼資料指派給每個資源做為標籤。標籤是您指派給 AWS 資源的標籤。每個標籤皆包含索引鍵與值。

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 資源都支援標記，因此您可以將相同的標籤指派給不同 服務中的資源，以指出資源相關。例如，您可以標記機器人及其搭配相同標籤使用的 Lambda 函數。
- 配置成本。您可以在 AWS 帳單與成本管理 儀表板上啟用標籤。 AWS 會使用標籤來分類您的成本，並傳送每月成本分配報告給您。對於 Amazon Lex，您可以使用別名特定的標籤來配置每個別名的成本，$LATEST但別名除外。您可以使用 Amazon Lex $LATEST 機器人的標籤來配置別名的成本。如需詳細資訊，請參閱AWS 帳單與成本管理 《 使用者指南》中的使用成本分配標籤。
- 控制對資源的存取。您可以使用 Amazon Lex 的標籤來建立政策，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 搭配 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，請參閱使用標籤存取資源。

您可以使用 AWS 管理主控台、 AWS Command Line Interface或 Amazon Lex API 來使用標籤。

為您的資源建立標籤

如果您使用的是 Amazon Lex 主控台，您可以在建立資源時標記資源，也可以稍後新增標籤。您也可以使用主控台來更新或移除現有的標籤。

如果您使用 AWS CLI 或 Amazon Lex API，您可以使用下列操作來管理 資源的標籤：

- ListTagsForResource – 檢視與資源相關聯的標籤。
- PutBot 和 PutBotAlias – 當您建立機器人或機器人別名時套用標籤。
- 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

下列基本限制適用於 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
- 索引鍵和值的有效字元 – a–z、A–Z、0–9、空格和下列字元：_ . : / = + - 和 @
- 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 管理主控台 ，並在 https://console.aws.amazon.com/lex/：// 開啟 Amazon Lex 主控台。
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 管理主控台 ，並在 https://console.aws.amazon.com/lex/：// 開啟 Amazon Lex 主控台。
2. 選擇您要新增機器人別名的機器人。
3. 選擇設定。
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 管理主控台 ，並在 https://console.aws.amazon.com/lex/ ：// 開啟 Amazon Lex 主控台。
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 管理主控台 ，並在 https://console.aws.amazon.com/lex/ ：// 開啟 Amazon Lex 主控台。
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 管理主控台 ，並在 https://console.aws.amazon.com/lex/ ：// 開啟 Amazon Lex 主控台。
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 管理主控台 ，並在 https://console.aws.amazon.com/lex/ ：// 開啟 Amazon Lex 主控台。
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 管理主控台 ，並在 https://console.aws.amazon.com/lex/：// 開啟 Amazon Lex 主控台。
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put-bot AWS CLI 命令顯示建立機器人時，您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，請參閱步驟四：開始使用 (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put-bot-alias AWS CLI 命令顯示建立機器人別名時，您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 命令來顯示與機器人、機器人別名、機器人管道相關聯的資源。

```
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"]'
```

Amazon Lex 入門

Amazon Lex 提供可與現有應用程式整合的 API 操作。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SDKs，您向 Amazon Lex 提出的請求會使用您提供的登入資料自動簽署和驗證。這是建置您的應用程式的建議選擇。
- AWS CLI — 您可以使用 AWS CLI 存取任何 Amazon Lex 功能，而無需撰寫任何程式碼。
- AWS 主控台 — 主控台是開始測試和使用 Amazon Lex 的最簡單方法

如果您是初次使用 Amazon Lex，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 AWS
2. 建立使用者

註冊 AWS

如果您已有 AWS 帳戶，請略過此任務。

當您註冊 Amazon Web Services (AWS) 時， AWS 您的帳戶會自動註冊所有 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，您只需為使用的資源付費。如果您是新 AWS 客戶，可免費開始使用 Amazon Lex。如需更多詳細資訊，請參閱 AWS 免費用量方案。

如果您已有 AWS 帳戶，請跳到下一個任務。若您尚未擁有 AWS 帳戶，請使用下列程序建立帳戶。

建立 AWS 帳戶

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或簡訊，並在電話鍵盤輸入驗證碼。

   當您註冊 時 AWS 帳戶，AWS 帳戶根使用者會建立 。根使用者有權存取該帳戶中的所有 AWS 服務 和資源。作為安全最佳實務，請將管理存取權指派給使用者，並且僅使用根使用者來執行需要根使用者存取權的任務。

寫下您的 AWS 帳戶 ID，因為下一個任務需要它。

建立使用者

中的服務 AWS，例如 Amazon Lex，會要求您在存取憑證時提供憑證，以便服務可以判斷您是否具有存取該服務所擁有資源的許可。主控台需要您的密碼。不過，我們不建議您 AWS 使用 AWS 帳戶的登入資料來存取 。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許可的 IAM 群組
- 將管理許可授予您建立的 使用者。

然後，您可以使用特殊 URL 和使用者的登入資料 AWS 來存取 。

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。如需說明，請參閱《IAM 使用者指南》中的建立您的第一個使用者和管理員群組。
2. 身為 使用者，您可以使用 AWS 管理主控台 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)

- 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，請下載並設定它。

> **Important**
>
> 您不需要 AWS CLI 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，請返回此處進行設定。

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：
   - 使用 進行設定 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 組態檔案的結尾。您在執行 AWS CLI 命令時使用此設定檔。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)

了解如何使用 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，這是預先定義的機器人，可提供所有必要的機器人組態。您只需要進行最基本的工作來測試端對端設定。

  此外，您可以使用 提供的 Lambda 函數藍圖 AWS Lambda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 函數做為程式碼掛勾。當中有提供範本程式碼。
- 練習 3 — 發佈機器人，然後建立新的版本。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：使用藍圖建立 Amazon Lex 機器人 （ 主控台 ）

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，然後在 Amazon Lex 主控台中測試它。

  本練習將使用 OrderFlowers 藍圖。如需藍圖的相關資訊，請參閱 Amazon Lex 和 AWS Lambda 藍圖。

- 建立 AWS Lambda 函數，並在 Lambda 主控台中測試該函數。處理請求時，您的機器人會呼叫此 Lambda 函數。在本練習中，您會使用 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勾，以滿足意圖。測試端對端的體驗。

以下各節說明藍圖的作用。

Amazon Lex Bot：藍圖概觀

您可以使用 OrderFlowers 藍圖來建立 Amazon Lex 機器人。如需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？」
  - `PickupDate` 槽的提示 –「您想要在哪一天拿取{FlowerType}？」
  - `PickupTime` 槽的提示 –「您想要在什麼時間拿取{FlowerType}？」
  - 確認陳述式 –「好的，您的 {FlowerType} 將在 {PickupDate} 的 {PickupTime} 準備好收件。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、驗證和履行任務。因此，在建立 Lambda 函數之後，您可以透過指定與程式碼掛勾相同的 Lambda 函數來更新意圖組態，以同時處理初始化和驗證和履行任務。

- 做為初始化和驗證程式碼掛勾，Lambda 函數會執行基本驗證。例如，如果使用者提供非正常營業時間的收件時間，Lambda 函數會指示 Amazon Lex 在這段時間內重新提示使用者。
- 做為履行程式碼掛勾的一部分，Lambda 函數會傳回摘要訊息，指出已放置花順序（也就是已履行意圖）。

後續步驟

步驟 1：建立 Amazon Lex 機器人（主控台）

步驟 1：建立 Amazon Lex 機器人（主控台）

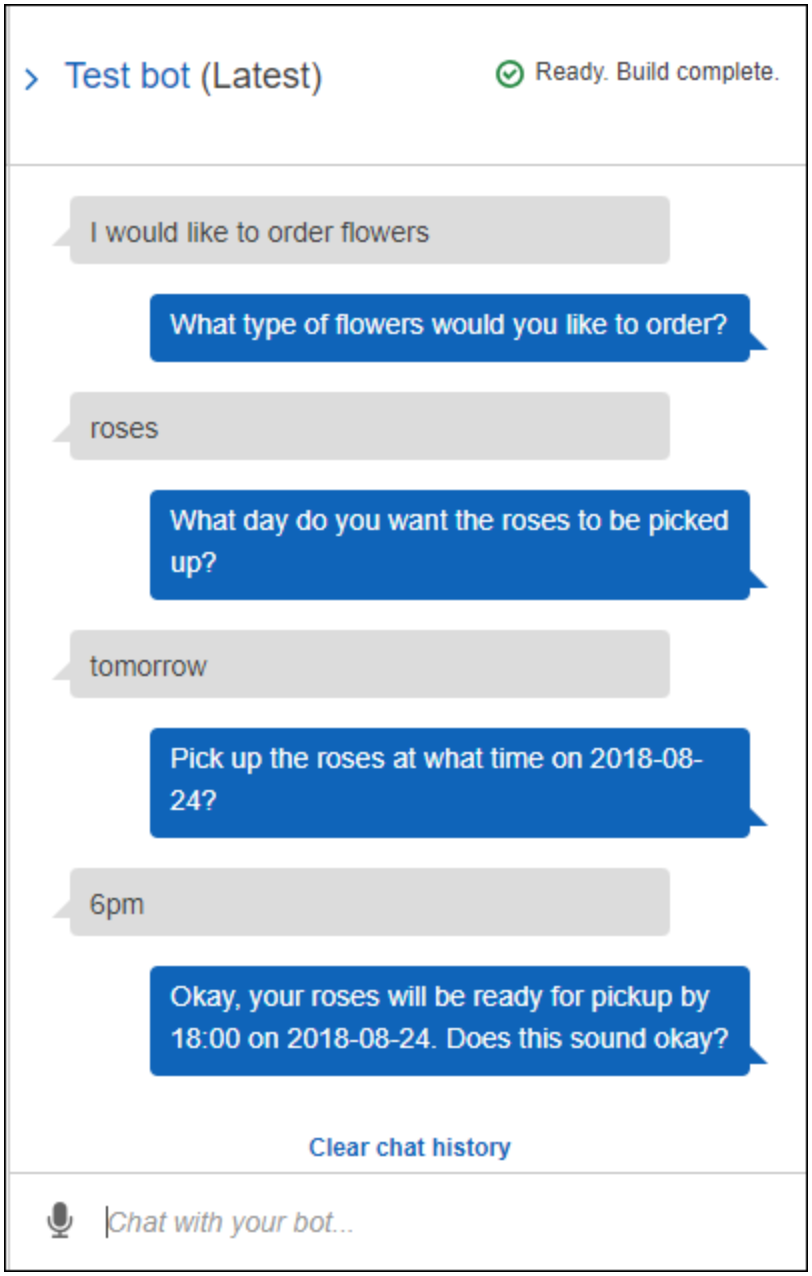
本練習將建立一個用於訂花的機器人，名為 OrderFlowersBot。

建立 Amazon Lex 機器人（主控台）

1. 登入 AWS 管理主控台，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**。
   - 對於使用者表達用語儲存，請選擇適當的回應。
4. 選擇建立。主控台向 Amazon Lex 提出儲存組態所需的請求。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，您使用 Lambda 函數來實現意圖，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)

本節說明在範例對話中，用戶端與 Amazon Lex 之間每個使用者輸入的資訊流程。

此範例使用主控台測試視窗與機器人進行對話。

開啟 Amazon Lex 測試視窗

1. 登入 AWS 管理主控台 ，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. 從主控台右側，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)
- 步驟 2b (選用)：檢閱輸入型資訊流程的詳細資訊 (主控台)

步驟 2a (選用)：檢閱口語化資訊流程的詳細資訊 (主控台)

本節說明當用戶端使用語音傳送請求時，用戶端與 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 和內文都會提供資訊給 Amazon Lex：

   - 請求 URI – 提供機器人名稱 (OrderFlowers)、機器人別名 ($LATEST) 和使用者名稱 （識別使用者的隨機字串）。 content表示這是 PostContent API 請求 （而非PostText請求）。
   - 請求標頭
     - x-amz-lex-session-attributes – base64 編碼值代表 "{}"。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 – 提供回應中傳回的音訊 Amazon Lex 文字記錄
- 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 會在目前意圖的內容中解譯輸入串流（它會記住已向此使用者詢問與FlowerType槽相關的資訊）。Amazon Lex 會先更新目前意圖的槽值。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 會在目前意圖的內容中解譯輸入串流（它會記住已向此使用者詢問與PickupDate槽相關的資訊）。Amazon Lex 會更新目前意圖的槽 (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 會在目前意圖的內容中解譯輸入串流（它會記住已向此使用者詢問與PickupTime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，才能繼續實現意圖。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，並了解使用者想要繼續訂單。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 會傳回包含下列項目的回應：

```
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。請注意下列有關 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 (其他範例：建立 Amazon 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 （主控台）

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": {}
}
```

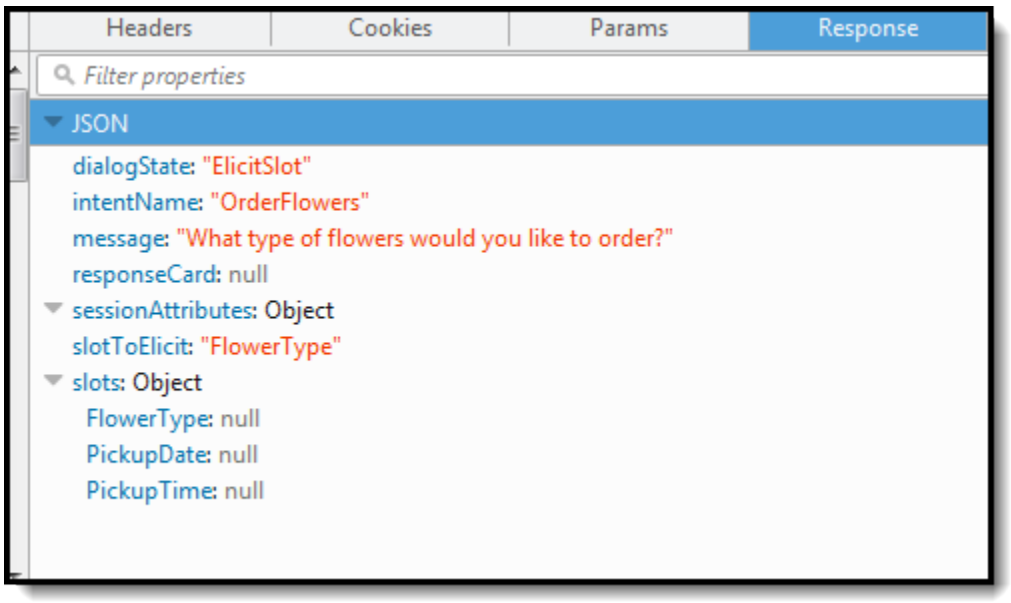
   請求 URI 和內文都會提供資訊給 Amazon Lex：

   - 請求 URI – 提供機器人名稱 (OrderFlowers)、機器人別名 ($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 inputText，Amazon Lex 會偵測意圖 (OrderFlowers)。此意圖沒有任何程式碼掛鉤（即 Lambda 函數），用於初始化和驗證使用者輸入或履行。

   Amazon Lex 選擇其中一個意圖的槽 (FlowerType) 來引出值。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：

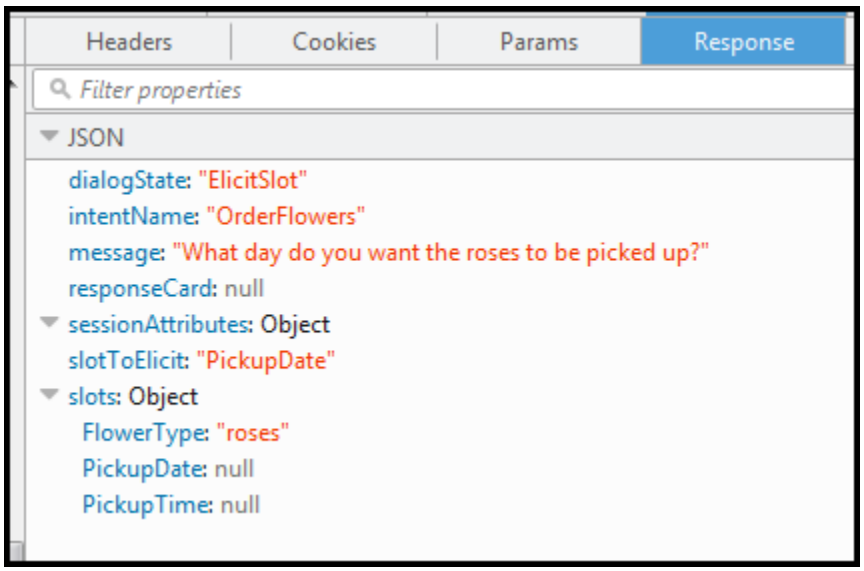
```
POST /bot/OrderFlowers/alias/$LATEST/user/4o9wwdhx6nlheferh6a73fujd3118f5w/text
"Content-Type":"application/json"
"Content-Encoding":"amz-1.0"

{
    "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 會先在目前意圖的內容`inputText`中解譯 ，該服務會記住已向特定使用者詢問`FlowerType`槽的相關資訊。Amazon Lex 會先更新目前意圖的槽值，並針對槽選擇另一個槽 (`PickupDate`) 及其提示訊息之一：您希望在哪一天收取玫瑰？—。

   然後，Amazon Lex 會傳回下列回應：



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：

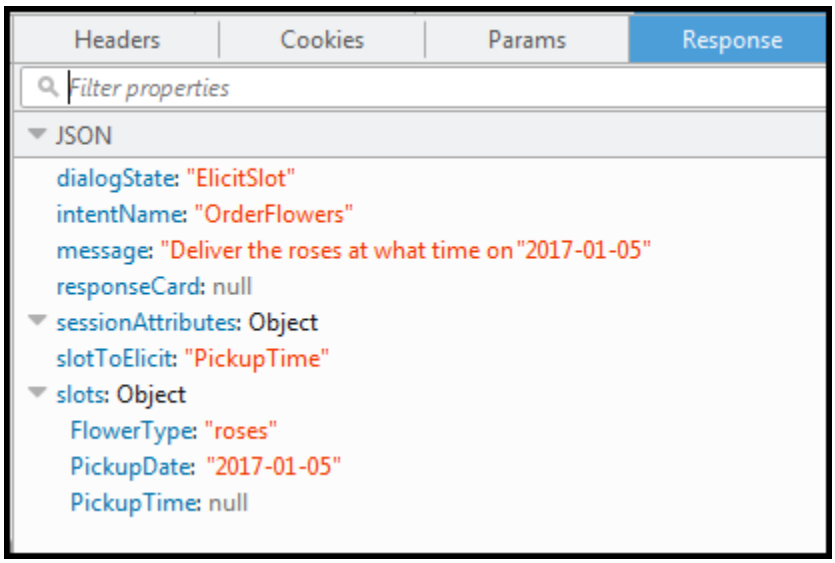
```
POST /bot/OrderFlowers/alias/$LATEST/user/4o9wwdhx6nlheferh6a73fujd3118f5w/text
"Content-Type":"application/json"
"Content-Encoding":"amz-1.0"

{
    "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 會先在目前意圖的內容`inputText`中解譯 - 服務會記住已向特定使用者詢問`PickupDate`槽的相關資訊。Amazon Lex 會更新目前意圖的槽 (`PickupDate`) 值。其將選擇另一個槽 (`PickupTime`) 來引出槽值。它會傳回其中一個值引出提示 - 在 2017-01-05 的什麼時間交付玫瑰？ - 給用戶端。

   然後，Amazon Lex 會傳回下列回應：



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：

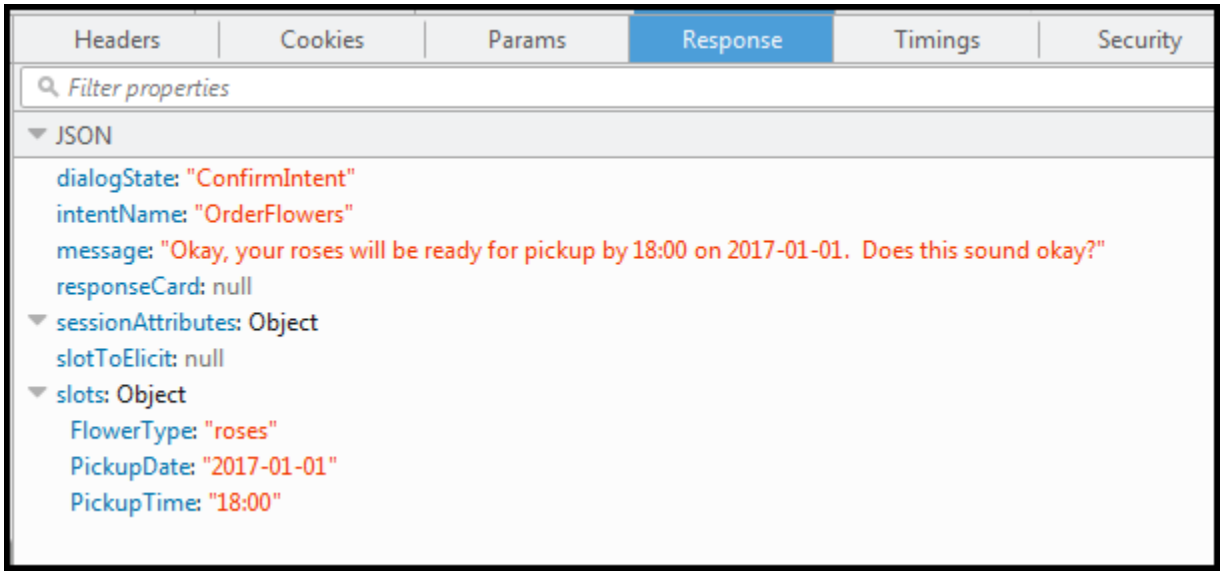
```
POST /bot/OrderFlowers/alias/$LATEST/user/4o9wwdhx6nlheferh6a73fujd3118f5w/text
"Content-Type":"application/json"
"Content-Encoding":"amz-1.0"

{
    "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 會先在目前意圖的內容`inputText`中解譯 - 服務會記住已向特定使用者詢問`PickupTime`槽的相關資訊。Amazon Lex 會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。

   `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，才能繼續實現意圖。Amazon Lex 會傳送下列訊息給用戶端，在訂購花之前請求確認：



用戶端顯示回應中的訊息。

5. 使用者輸入：好

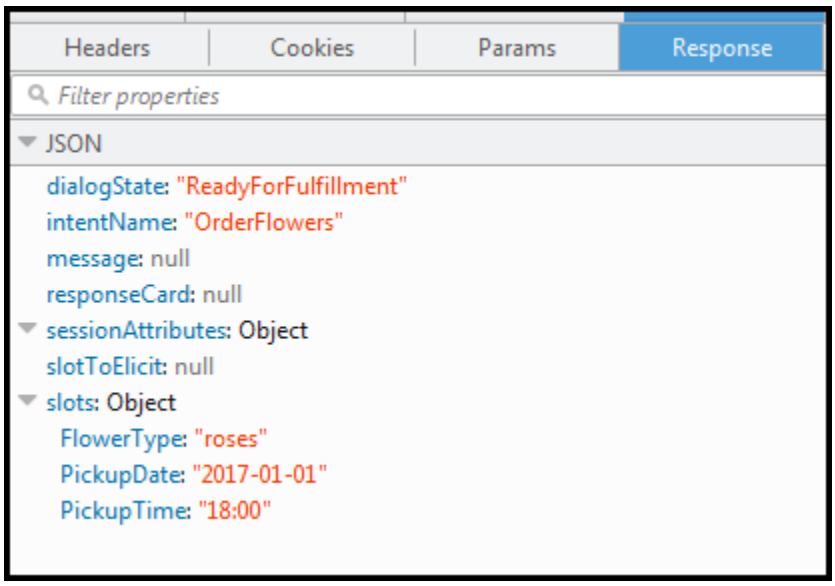
   a. 用戶端 (主控台) 傳送以下 PostText 請求給 Amazon Lex：

```
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 會在確認目前意圖`inputText`的情況下解譯 。其已理解使用者想要完成下單。使用 將`OrderFlowers`意圖設定為`ReturnIntent`履行活動 （ 沒有 Lambda 函數來履行意圖 ）。因此，Amazon Lex 會將下列槽資料傳回給用戶端。



Amazon Lex 將 `dialogState` 設定為 `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。請注意下列有關 Lambda 函數的事項：

   - Lambda 函數會在每次使用者輸入後驗證槽資料。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。
   - 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 (其他範例：建立 Amazon 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 （主控台）

步驟 3：建立 Lambda 函數（主控台）

建立 Lambda 函數（使用 lex-order-flowers-python 藍圖），AWS Lambda 並使用主控台中的範例事件資料執行測試調用。

您會返回 Amazon Lex 主控台，並將 Lambda 函數新增為程式碼掛勾，以滿足`OrderFlowersBot`您在上一節建立的 中的`OrderFlowers`意圖。

建立 Lambda 函數 (主控台)

1. 登入 AWS 管理主控台 並在 https：//https://console.aws.amazon.com/lambda/ 開啟 AWS Lambda 主控台。
2. 選擇建立函數。
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 Node.js 和 Python 中提供。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`)。
   - 針對執行角色，選擇使用基本 Lambda 許可建立新角色。
   - 保留其他預設值。
5. 選擇建立函數。
6. 如果您使用的地區設定不是英文 (US) (en-US)，請更新意圖名稱，如中所述更新特定地區設定的藍圖。
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。此範例事件符合 Amazon Lex 請求/回應模型（請參閱 使用 Lambda 函數)。為測試事件命名 (`LexOrderFlowersTest`)。
   c. 選擇建立。
   d. 選擇 Test (測試) 來測試程式碼掛勾。
   e. 確認 Lambda 函數已成功執行。在此情況下，回應符合 Amazon Lex 回應模型。

後續步驟

步驟 4：將 Lambda 函數新增為 Code Hook （主控台）

步驟 4：將 Lambda 函數新增為 Code Hook （主控台）

在本節中，您將更新 OrderFlowers 意圖使用 Lambda 函數的組態，如下所示：

- 首先，使用 Lambda 函數做為程式碼掛勾來執行`OrderFlowers`意圖的履行。您會測試機器人，並確認您收到來自 Lambda 函數的履行訊息。Amazon Lex 只有在您提供訂購花的所有必要槽資料之後，才會叫用 Lambda 函數。
- 將相同的 Lambda 函數設定為程式碼掛勾，以執行初始化和驗證。您測試並確認 Lambda 函數執行驗證 （當您提供槽資料時）。

新增 Lambda 函數做為程式碼掛勾 （主控台）

1. 在 Amazon Lex 主控台中，選取 OrderFlowers 機器人。主控台會顯示 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. 新增 Lambda 函數做為履行程式碼掛鉤並進行測試。

   a. 在編輯器中，選擇AWS Lambda 函數做為履行，然後選取您在上一個步驟 () 中建立的 Lambda 函數`OrderFlowersCodeHook`。選擇確定以授予 Amazon Lex 叫用 Lambda 函數的許可。

      您正在將此 Lambda 函數設定為程式碼掛勾，以滿足意圖。Amazon Lex 只有在擁有使用者的所有必要槽資料以滿足意圖之後，才會叫用此函數。

   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。

   最後一個陳述式 "Thanks， your order for roses....." 是來自您設定為程式碼掛勾之 Lambda 函數的回應。在上一節中，沒有 Lambda 函數。現在您使用 Lambda 函數來實際實現`OrderFlowers`意圖。

3. 新增 Lambda 函數做為初始化和驗證程式碼掛鉤，並測試。

   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 函數接收的輸入事件具有欄位 (`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。

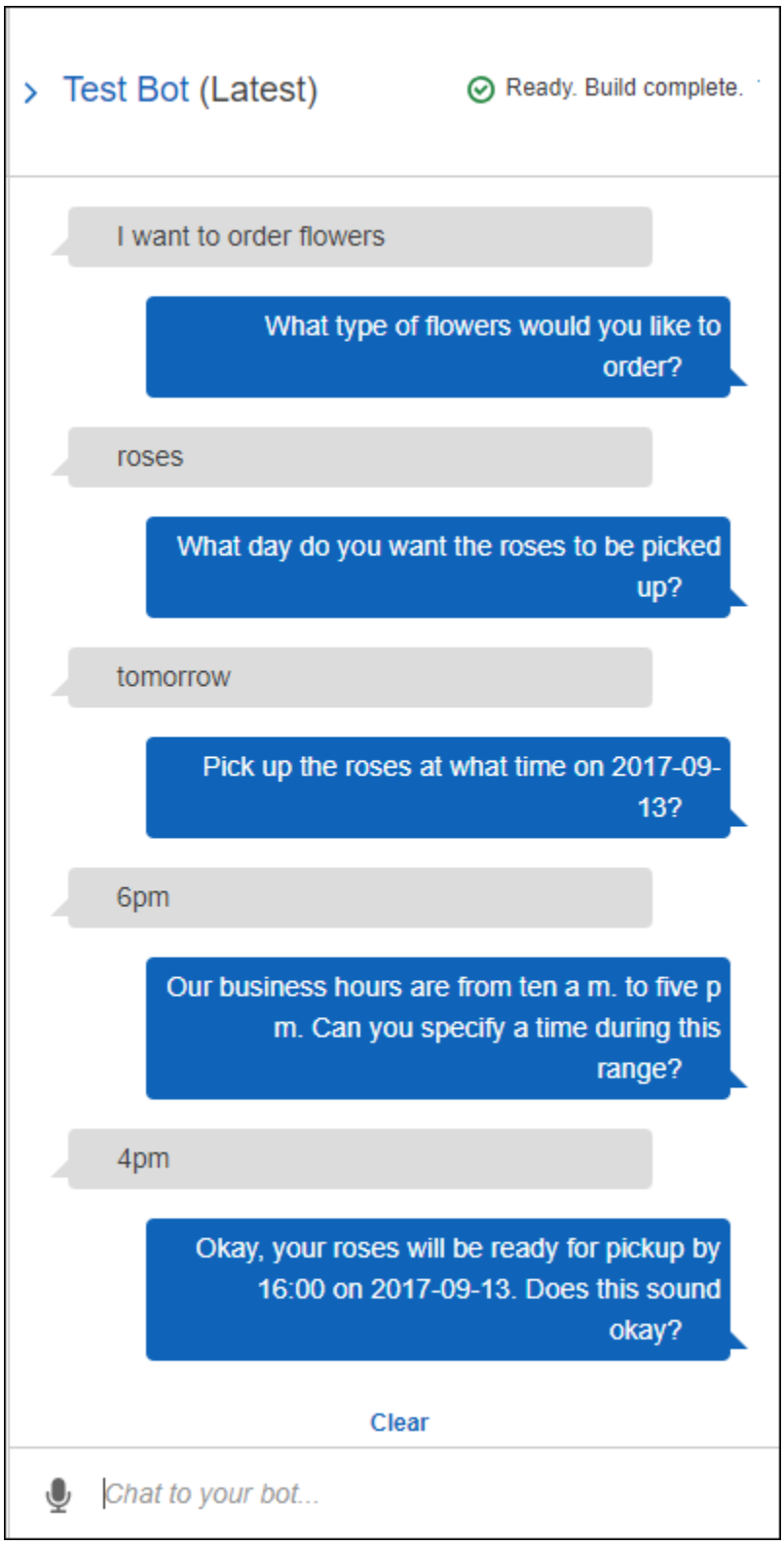
b. 從編輯器的 Options (選項)中選擇 Initialization and validation (初始化和驗證)。

c. 同樣地，選取相同的 Lambda 函數。

d. 選擇 Build (建置)。

e. 測試機器人。

   您現在可以與 Amazon Lex 交談，如下圖所示。若要測試驗證部分，請選擇時間下午 6 點，您的 Lambda 函數會傳回回應 (「我們的營業時間是上午 10 點到下午 5 點」)，並再次提示您。在您提供所有有效的槽資料後，Lambda 函數會履行順序。


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)

本節說明用戶端與 Amazon Lex 之間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執行時間 API 將請求傳送至 Amazon Lex，並相應地顯示請求和回應詳細資訊。如需用戶端與用戶端使用 PostContent API 的 Amazon Lex 之間資訊流程的範例，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 和內文都會提供資訊給 Amazon Lex：

   - 請求 URI – 提供機器人名稱 (OrderFlowers)、機器人別名 ($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 inputText，Amazon Lex 會偵測意圖 (OrderFlowers)。此意圖是以 Lambda 函數設定為程式碼掛勾，用於使用者資料初始化和驗證。因此，Amazon Lex 透過將下列資訊做為事件資料傳遞，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 僅支援 1.0 版本。
- `invocationSource` – 指出 Lambda 函數調用的目的。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槽資料以滿足意圖。
- `currentIntent` 資訊 – 所有槽值均設定為 null。

c. 此時，所有槽值都是 null。Lambda 函數不需要驗證。Lambda 函數會傳回下列回應給 Amazon Lex：

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

```
            "PickupDate": null
        }
    }
}
```

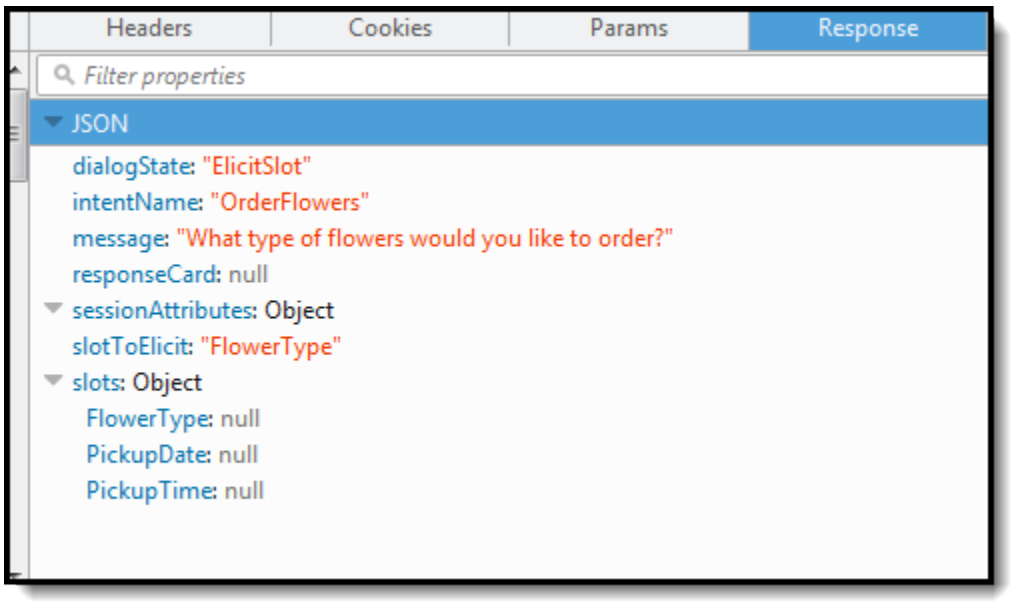
如需回應格式的相關資訊，請參閱回應格式。

注意下列事項：

- `dialogAction.type` – 透過將此值設定為 `Delegate`，Lambda 函數會將決定下一個動作過程的責任委派給 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 接下來要做什麼，如以下幾個步驟所示。

d. 根據 `dialogAction.type`，Amazon Lex 會決定下一個動作。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請求傳送至 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 會先在目前意圖的內容inputText中解譯 。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 已將FlowerType槽更新為玫瑰。

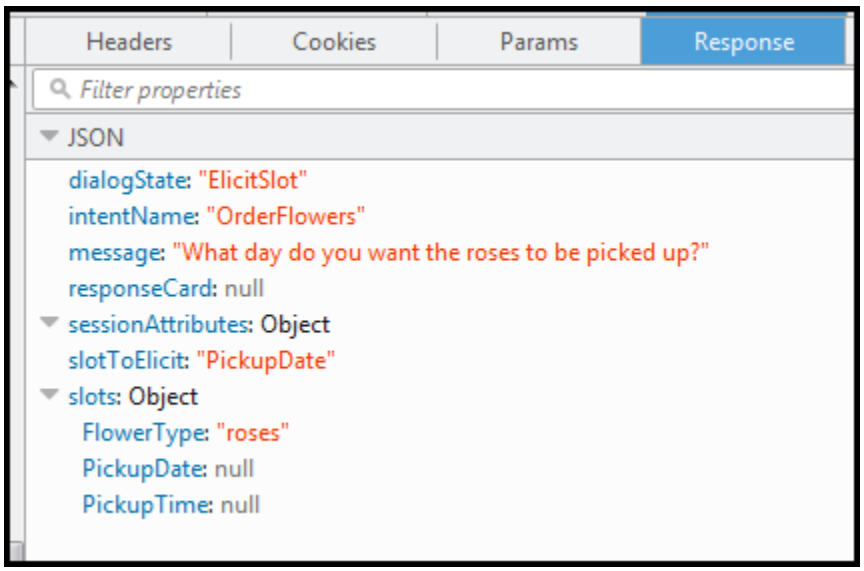
c. 根據 invocationSource的值DialogCodeHook，Lambda 函數會執行使用者資料驗證。它會將 roses辨識為有效的槽值 （並設定為工作階段屬性）Price，並將下列回應傳回給 Amazon Lex。

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

注意下列事項：

- sessionAttributes – Lambda 函數已新增 Price（玫瑰的） 做為工作階段屬性。
- dialogAction.type – 設定為 Delegate。使用者資料有效，因此 Lambda 函數會指示 Amazon Lex 選擇下一個動作。

d. 根據 dialogAction.type，Amazon Lex 選擇下一個動作。Amazon Lex 知道它需要更多槽資料，因此會根據意圖組態，挑選具有最高優先順序的下一個未填充槽 (PickupDate)。Amazon Lex 會根據意圖組態選取其中一個引出值提示訊息：「您希望在哪一天撿起玫瑰？」，針對此槽傳送下列回應回用戶端：



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請求傳送至 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 會記住內容，也就是它正在為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 會傳送下列事件來叫用 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 已透過設定 PickupDate值currentIntent.slots來更新 。另請注意，服務會將 sessionAttributes 傳遞至 Lambda 函數。

c. 根據 invocationSource的值DialogCodeHook，Lambda 函數會執行使用者資料驗證。它會辨識PickupDate槽值是否有效，並傳回下列回應給 Amazon Lex：

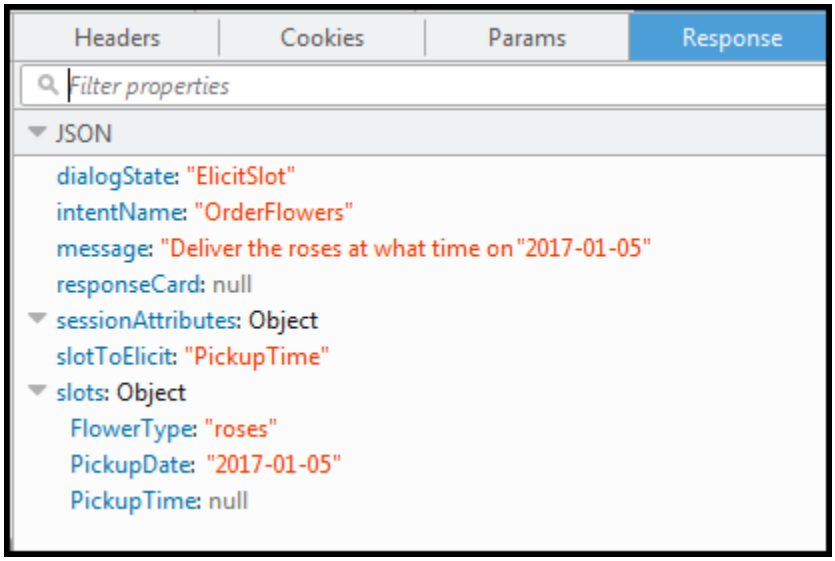
```
{
    "sessionAttributes": {
        "Price": 25
    },
    "dialogAction": {
        "type": "Delegate",
        "slots": {
            "PickupTime": null,
            "FlowerType": "roses",
            "PickupDate": "2017-01-05"
        }
    }
}
```

注意下列事項：

- sessionAttributes – 未改變。

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

d. 根據 `dialogAction.type`，Amazon Lex 選擇下一個動作。Amazon Lex 知道它需要更多槽資料，因此會根據意圖組態，挑選具有最高優先順序的下一個未填充槽 (`PickupTime`)。Amazon Lex 會根據意圖組態為此槽選取其中一個提示訊息 (「在 2017-01-05 的什麼時間交付玫瑰？」)，並將下列回應傳回給用戶端：



用戶端會在回應中顯示訊息 –「在 2017-01-05 的什麼時間交付玫瑰？」

4. 使用者：下午 4 點

a. 用戶端會將下列PostText請求傳送至 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 了解內容。其明白這是稍早引出的 PickupTime 槽的資料。在這種情況下，它會知道 inputText值適用於PickupTime插槽。然後，Amazon Lex 會傳送下列事件來叫用 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 已透過設定 PickupTime值currentIntent.slots來更新 。

c. 根據 invocationSource的值DialogCodeHook，Lambda 函數會執行使用者資料驗證。它會辨識PickupDate槽值是否有效，並傳回下列回應給 Amazon Lex。

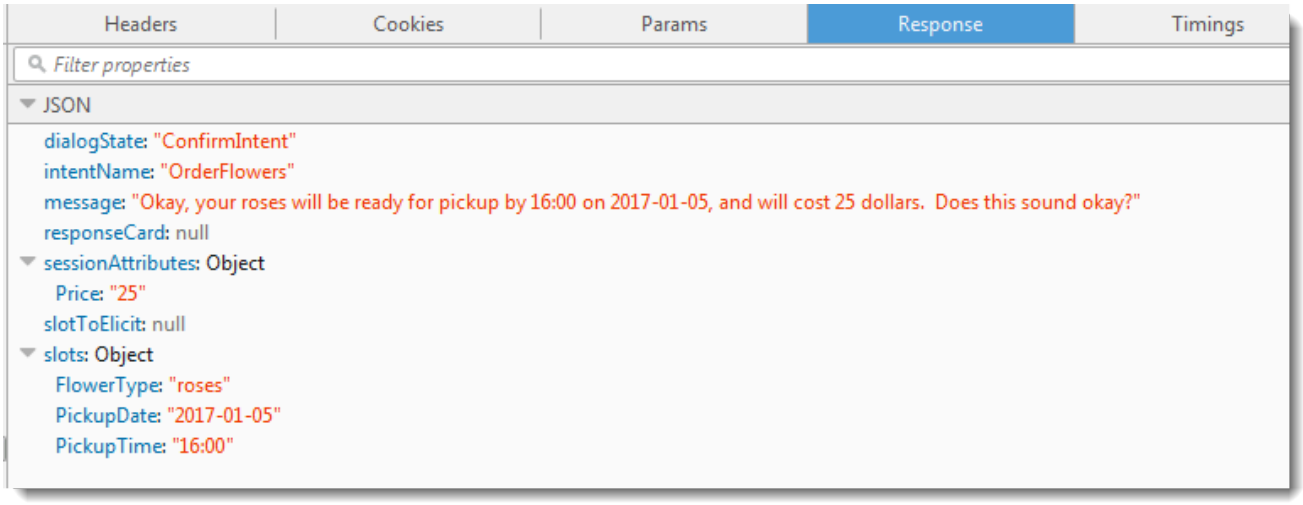
```
{
    "sessionAttributes": {
        "Price": 25
    },
    "dialogAction": {
        "type": "Delegate",
        "slots": {
            "PickupTime": "16:00",
            "FlowerType": "roses",
```

```
            "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`。使用者資料有效，因此 Lambda 函數會指示 Amazon Lex 選擇下一個動作。

d. 目前，Amazon Lex 知道它擁有所有槽資料。該意圖設定了一則確認提示。因此，Amazon Lex 會傳送下列回應給使用者，在滿足意圖之前要求確認：



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請求傳送至 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 會在確認目前意圖inputText的情況下解譯 。Amazon Lex 了解使用者想要繼續訂單。這次 Amazon Lex 會傳送下列事件來叫用 Lambda 函數以滿足意圖，這會在傳送至 Lambda 函數invocationSourceFulfillmentCodeHook時將 設定為 。Amazon Lex 也會將 confirmationStatus設定為 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 將此值設定為 FulfillmentCodeHook，指示 Lambda 函數實現意圖。
- confirmationStatus – 設定為 Confirmed。

c. 這次，Lambda 函數會滿足OrderFlowers意圖，並傳回下列回應：

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

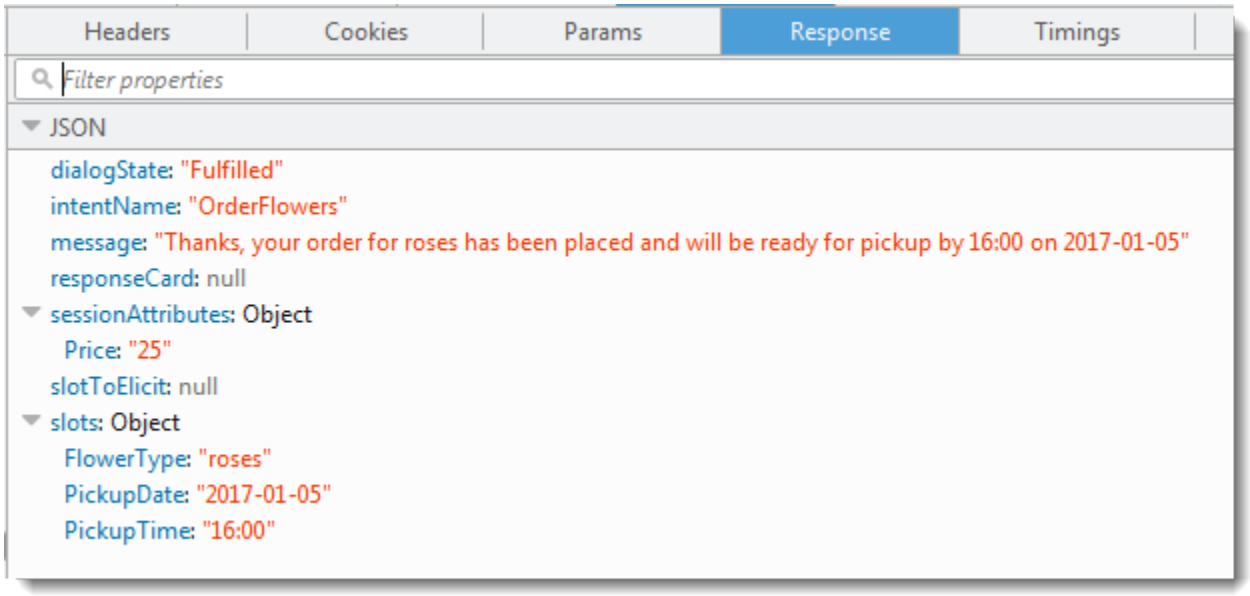
```
    },
    "dialogAction": {
        "type": "Close",
        "fulfillmentState": "Fulfilled",
        "message": {
            "contentType": "PlainText",
            "content": "Thanks, your order for roses has been placed and will
 be ready for pickup by 16:00 on 2017-01-05"
        }
    }
}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

注意下列事項：

- 設定 `dialogAction.type`- Lambda 函數將此值設定為 `Close`，指示 Amazon Lex 不預期使用者回應。
- `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端。

然後，Amazon Lex 會將下列項目傳回給用戶端：



請注意：

- `dialogState` – Amazon Lex 將此值設定為 `fulfilled`。
- `message` – 是 Lambda 函數提供的相同訊息。

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、將無效的槽資料值重設為 null，並要求 Amazon Lex 提示使用者提供有效的資料。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。為了回應，Lambda 函數會更新工作階段屬性`Price`中的 ，保持花訂單的執行總數。

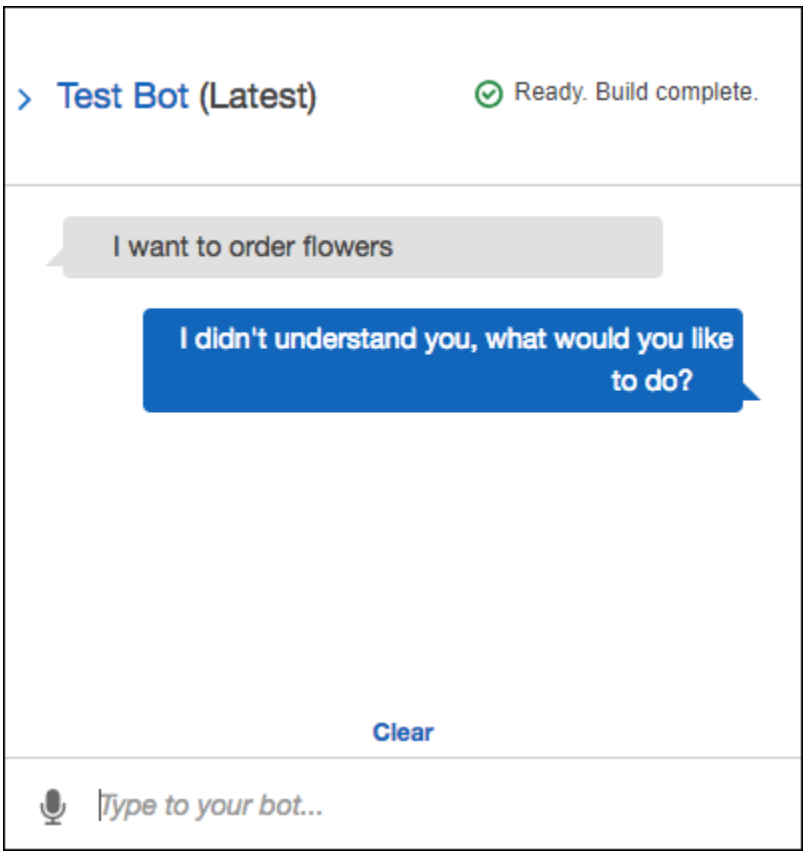
   Lambda 函數也會執行履行活動。

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，以建置機器學習模型來識別和回應使用者的意圖。嘗試輸入「我想要訂購花朵」，如下列測試時段所示。Amazon Lex 無法辨識文字，並以「我不了解您，您想要做什麼？」回應 您可以透過加入更多表達用語來改善機器學習模型。

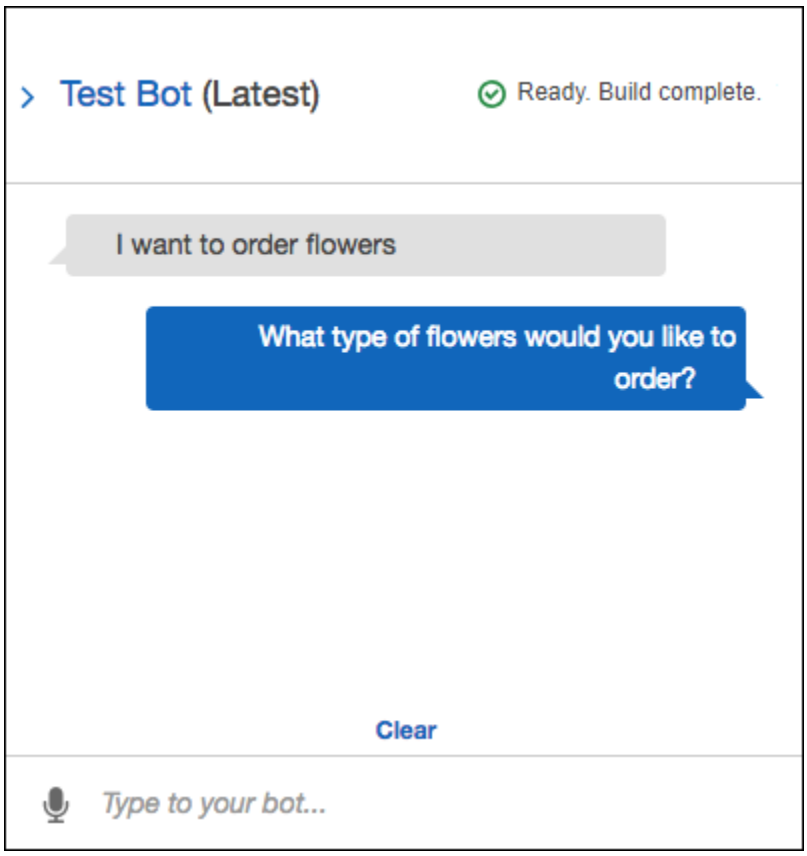


您新增的每個表達用語都會為 Amazon Lex 提供有關如何回應使用者的詳細資訊。您不需要新增確切的表達用語，Amazon Lex 會從您提供的範例進行一般化，以辨識完全相符和類似的輸入。

加入表達用語 (主控台)

1. 將表達用語「我想要花朵」新增至意圖編輯器的範例表達用語區段，如下圖所示，然後按一下新表達用語旁的加號圖示。

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「我想要訂購花朵」。Amazon Lex 會辨識片語，並以「您想要訂購哪種類型的花？」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 管理主控台 ，並在 https：//https://console.aws.amazon.com/lex/ 開啟 Amazon Lex 主控台。

2. 從機器人清單中，選擇 OrderFlowers 旁的核取方塊。
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. 從意圖清單中選擇 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除帳戶的所有 Amazon Lex 資源。如果需要，您可以使用 Lambda 主控台來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，您可以使用 Amazon Lex 主控台建立訂購比薩的自訂機器人 (`OrderPizzaBot`)。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 函式
- 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 函式

首先，建立滿足比薩訂單的 Lambda 函數。您可以在 Amazon Lex 機器人中指定此函數，並在下一節中建立。

建立 Lambda 函數

1. 登入 AWS 管理主控台 ，並在 https：//https://console.aws.amazon.com/lambda/ 開啟 AWS Lambda 主控台。
2. 選擇建立函數。

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。

   請執行下列操作：

   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。

若要測試 Lambda 函數：

1. 登入 AWS 管理主控台 ，並在 https：//https://console.aws.amazon.com/lambda/ 開啟 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 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。

```
{
  "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. 選擇建立。

AWS Lambda 會建立測試，然後您返回 函數頁面。選擇測試，Lambda 會執行您的 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

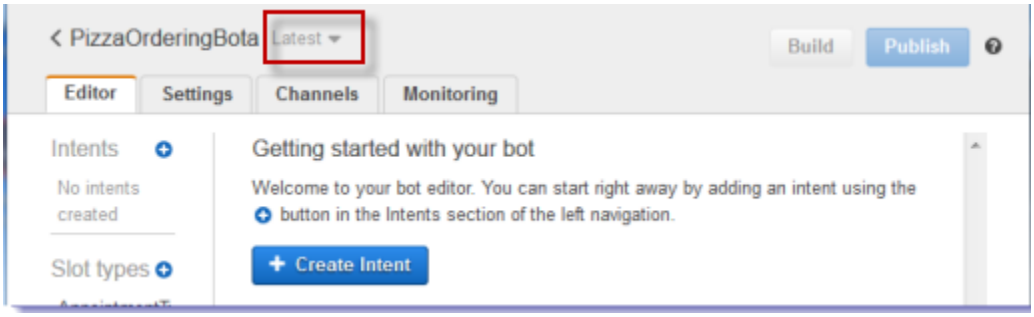
使用最少的必要資訊，建立 PizzaOrderingBot 機器人。稍後再為該機器人加入意圖，也就是使用者想要執行的動作。

建立機器人

1. 登入 AWS 管理主控台 ，並在 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：為您的機器人選擇語言和地區設定。
      - 輸出語音：Salli
      - 工作階段逾時：5 分鐘
      - COPPA：選擇適當的回應。
      - 使用者表達用語儲存：選擇適當的回應。

c. 選擇建立。

主控台會向 Amazon Lex 傳送建立新機器人的請求。Amazon Lex 會將機器人版本設定為 $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 資源具有 $LATEST 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)。

主控台會將請求傳送至 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

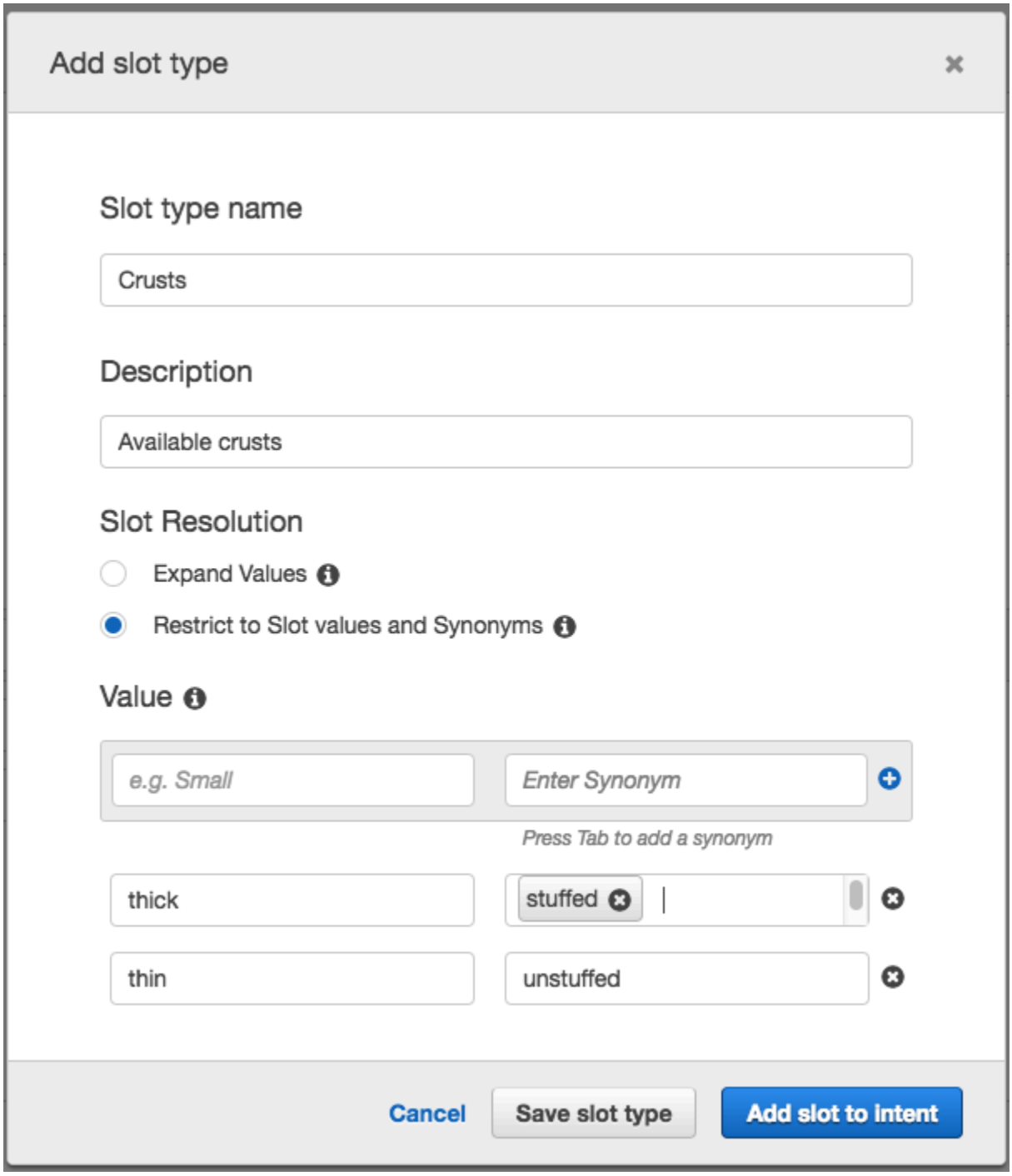
建立槽類型

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 – 輸入 **`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 (+)。

   對話方塊看起來應該如下圖所示：



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	提示詞
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

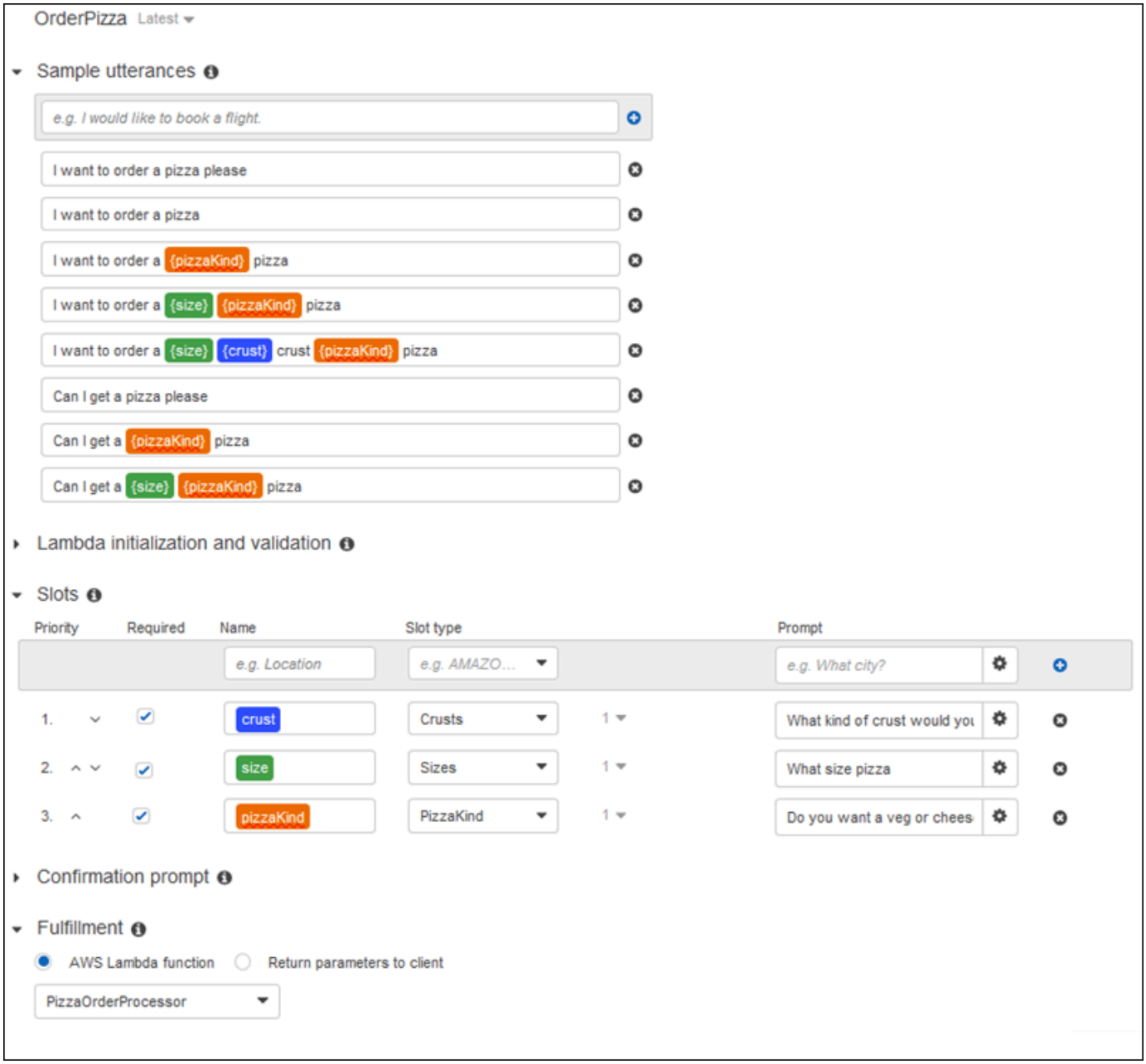
設定 `OrderPizza` 意圖以便完成使用者訂購比薩的接單工作。

設定意圖

- 在 OrderPizza 組態頁面上，依如下所述設定意圖：
  - 表達用語範例 – 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 Lambda 函數許可對話方塊，請選擇確定以授予呼叫 `PizzaOrderProcessor` Lambda 函數的`OrderPizza`意圖許可。
- 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。

     如果服務無法在設定的連續請求數之後判斷使用者的意圖，Amazon Lex 會傳回包含此訊息的回應。

   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。

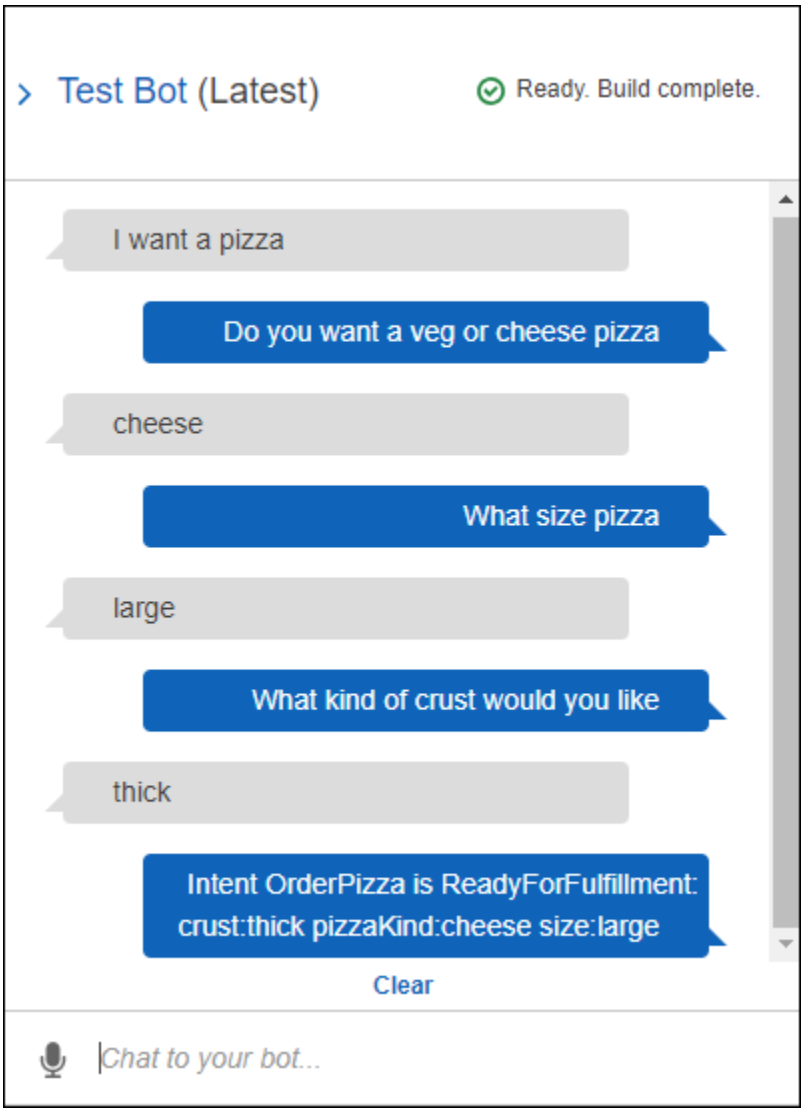
建置及測試機器人

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

   Amazon Lex 為機器人建置機器學習模型。當您測試機器人時，主控台會使用執行時間 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您的 Amazon Lex 機器人通訊。

   - 例如，您可以說出或輸入以下內容：



- 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「我想要訂購比薩」時，Amazon Lex 會偵測意圖 (OrderPizza)。然後，Amazon Lex 會詢問槽資訊。

  在您提供所有槽資訊後，Amazon Lex 會叫用您為意圖設定的 Lambda 函數。

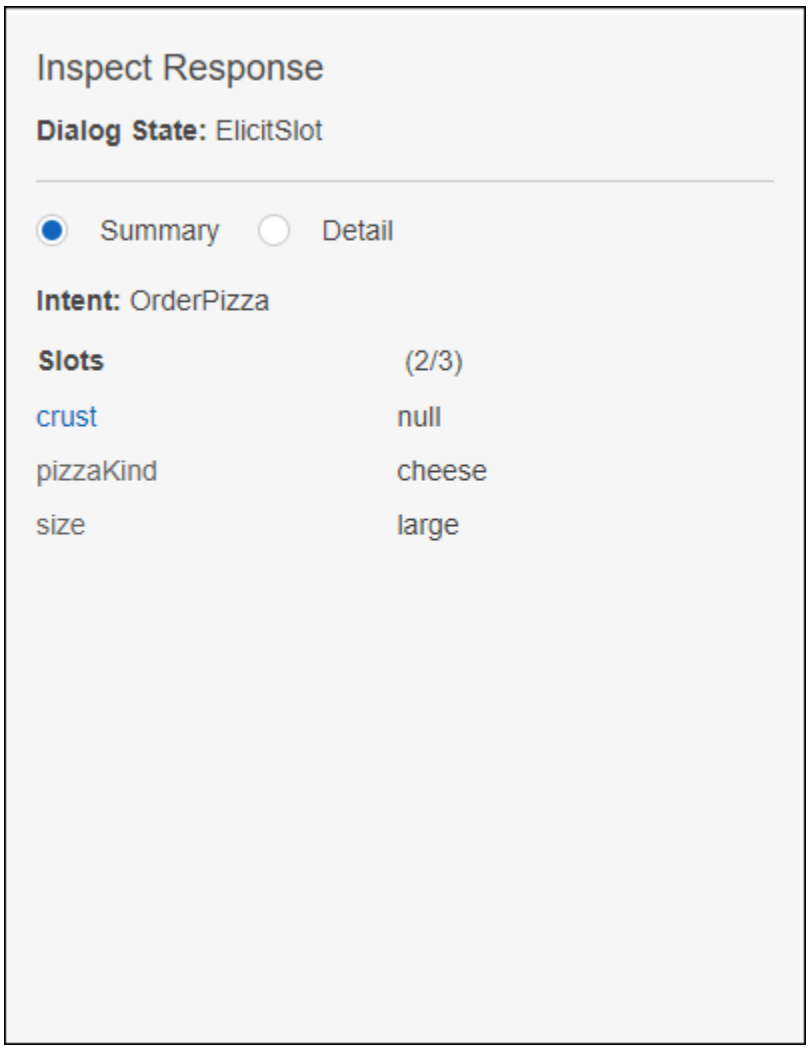
  Lambda 函數會傳回訊息 (「好的，我已訂購您的 ...」) 至 Amazon Lex，Amazon Lex 會傳回給您。

檢測回應

在聊天視窗下方是一個窗格，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。

- 對話方塊狀態 – 與使用者對話的目前狀態。其可能是 ElicitIntent、ElicitSlot、ConfirmIntent 或 Fulfilled。

- 摘要 – 顯示對話方塊的簡化檢視，其中顯示正在履行意圖的槽值，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，更深入地檢視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 管理主控台 ，並在 https://console.aws.amazon.com/lex/：// 開啟 Amazon Lex 主控台。
2. 從機器人清單中選擇 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. 從意圖清單中選擇 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
- 使用 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 會取得 `$LATEST`版本的快照副本來發佈新版本。
- 建立指向新版本的別名。

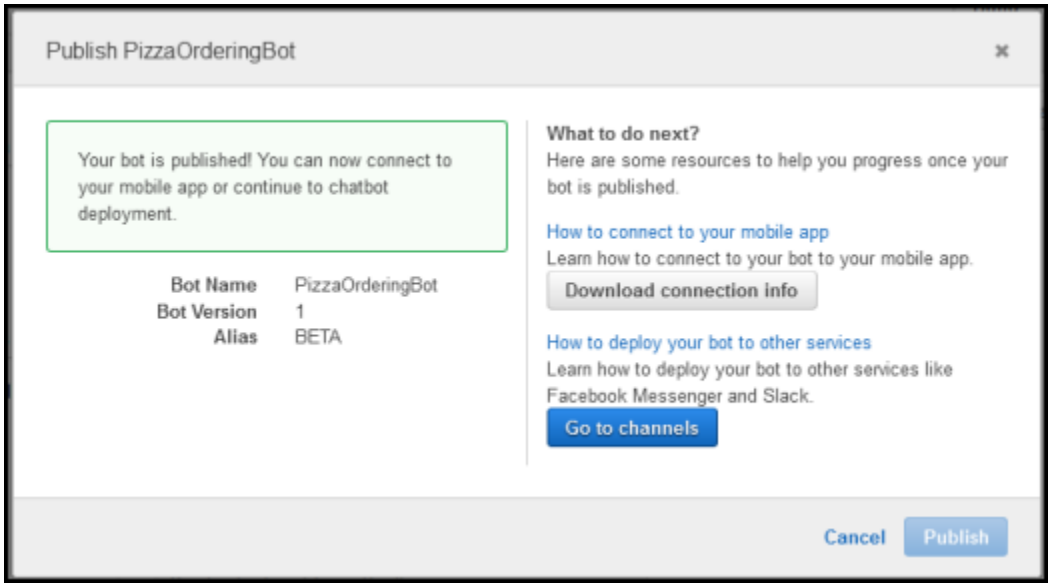
如需有關版本控制與別名的詳細資訊，請參閱版本控制與別名。

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 (發布)。
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[將 Amazon Lex 機器人與 Facebook Messenger 整合]。

步驟四：開始使用 (AWS CLI)

在此步驟中，您會使用 AWS CLI 來建立、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 中建立的機器人，以新增額外的範例表達用語。Amazon Lex 使用範例表達用語來建置機器人的機器學習模型。
- 練習 3 — 更新您在練習 1 中建立的機器人，以新增 Lambda 函數來驗證使用者輸入並滿足意圖。
- 練習 4 — 發佈您在練習 1 中建立的槽類型、意圖和機器人資源版本。版本是無法變更之資源的快照。
- 練習 5 — 為您在練習 1 中建立的機器人建立別名。

- 練習 6：刪除您在練習 1 中建立的槽類型、意圖和機器人，以及您在練習 5 中建立的別名，以清除您的帳戶。

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，您會使用 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 會擔任 AWS Identity and Access Management 服務連結角色，以代表您的機器人呼叫 AWS 服務。您帳戶中的角色會連結至 Amazon Lex 使用案例，並具有預先定義的許可。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 中 AWS CLI，輸入下列命令：

   ```
   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：

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 呼叫 PutSlotType操作 AWS CLI 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 中 AWS CLI，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：

後續步驟

步驟四：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"
}
```

步驟四：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**
>
> 下列 AWS CLI 範例已針對 Unix、Linux 和 macOS 格式化。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 中 AWS CLI，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。

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

下列程式碼提供建置 Amazon Lex 機器人所需的 JSON `OrderFlowers` 資料：

```
{
    "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**
>
> 下列 AWS CLI 範例已針對 Unix、Linux 和 macOS 格式化。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 中 AWS CLI，開始與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 會辨識使用者的意圖，並透過傳回下列回應開始對話：

   ```
   {
       "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"
```

在您確認訂單後，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。您可以使用 Amazon Polly text-to-speech 操作產生音訊檔案。

若要執行本練習中的命令，您需要知道要執行 Amazon Lex 和 Amazon Polly 命令的區域。如需 Amazon Lex 的區域清單，請參閱 執行時間服務配額。如需 Amazon Polly 的區域清單，請參閱《Amazon Web Services 一般參考》中的AWS 區域和端點。

> **Note**
>
> 下列 AWS CLI 範例已針對 Unix、Linux 和 macOS 格式化。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. 若要將音訊檔案傳送至 Amazon Lex，請執行下列命令。Amazon Lex 會將來自回應的音訊儲存在指定的輸出檔案中。

```
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 會回應第一個槽的請求。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. 若要訂購玫瑰，請建立下列音訊檔案，並將其傳送至 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. 若要設定交付日期，請建立下列音訊檔案並將其傳送至 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. 若要設定交付時間，請建立下列音訊檔案並將其傳送至 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. 若要確認交付，請建立下列音訊檔案並將其傳送至 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
```

在您確認交付後，Amazon Lex 會傳送確認滿足意圖的回應：

```
{
    "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)

若要改善 Amazon Lex 用來辨識使用者請求的機器學習模型，請將另一個範例表達用語新增至機器人。

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**
>
> 下列 AWS CLI 範例已針對 Unix、Linux 和 macOS 格式化。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 會將輸出傳送至名為 的檔案 **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後的意圖傳送至 Amazon Lex：

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

Amazon Lex 會傳送下列回應：

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 中 AWS CLI，取得OrderFlowersBot機器人的定義，並使用下列命令將其儲存至 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 中 AWS CLI，透過對 執行下列命令來建置機器人的新修訂版：

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

伺服器的回應為：

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，以驗證使用者輸入並滿足使用者對機器人的意圖。

新增 Lambda 表達式是一個五步驟的程序。

1. 使用 Lambda AddPermission 函數來啟用呼叫 Lambda 調用操作的`OrderFlowers`意圖。
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。

如果您在新增`InvokeFunction`許可之前將 Lambda 函數新增至意圖，您會收到下列錯誤訊息：

```
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.
```

本練習使用來自 的 Lambda 函數練習 1：使用藍圖建立 Amazon Lex 機器人（主控台）。如需建立 Lambda 函數的說明，請參閱 步驟 3：建立 Lambda 函數（主控台）。

> **Note**
>
> 下列 AWS CLI 範例已針對 Unix、Linux 和 macOS 格式化。用於 Windows 時，請將 "\$LATEST" 變更為 $LATEST。

將 Lambda 函數新增至意圖

1. 在 中 AWS CLI，新增`OrderFlowers`意圖的`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 會傳送下列回應：

   ```
   {
       "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 會將輸出傳送至名為 的檔案**`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，將更新的意圖傳送至 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 中 AWS CLI，取得OrderFlowersBot機器人的定義，並將其儲存至檔案：

   ```
   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 中 AWS CLI，建置機器人的新修訂版：

```
aws lex-models put-bot \
    --region region \
    --name OrderFlowersBot \
    --cli-input-json file://OrderFlowersBot-V3.json
```

伺服器的回應為：

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**
>
> 下列 AWS CLI 範例已針對 Unix、Linux 和 macOS 格式化。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 中 AWS CLI，取得最新版的槽類型：

```
aws lex-models get-slot-type \
    --region region \
    --name FlowerTypes \
    --slot-type-version "\$LATEST"
```

Amazon Lex 的回應如下。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 的回應如下。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**
>
> 下列 AWS CLI 範例已針對 Unix、Linux 和 macOS 格式化。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 中 AWS CLI，取得OrderFlowers意圖的$LATEST版本，並將其儲存至 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 中 AWS CLI，儲存意圖的修訂：

```
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**
>
> 下列 AWS CLI 範例已針對 Unix、Linux 和 macOS 格式化。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 中 AWS CLI，取得OrderFlowersBot機器人的$LATEST版本，並將其儲存至 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 中 AWS CLI，儲存機器人的新修訂版。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 中 AWS CLI，取得您在 中建立的`OrderFlowersBot`機器人版本練習 4：發佈版本 (AWS CLI)。

```
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 中 AWS CLI，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 支援發佈版本的機器人、意圖和槽類型，讓您可以控制用戶端應用程式使用的實作。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。當您準備好更新機器人時，您會發佈第 2 版，並將別名變更為指向新版本。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

當您為 Amazon Lex 資源建立版本時，您可以建立資源的快照，以便使用建立版本時現有的資源。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。

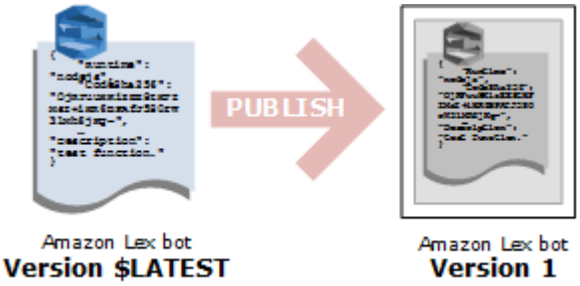
$LATEST 是您的資源的工作複本。您只能更新 $LATEST 版本，而且直到您發佈第一個版本之前，$LATEST 是您所擁有資源的唯一版本。

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。

您的機器人$LATEST版本應僅用於手動測試。Amazon Lex 會限制您可以對機器人$LATEST版本提出的執行時間請求數量。

發佈 Amazon Lex 資源版本

當您發佈資源時，Amazon Lex 會複製 $LATEST版本，並將其儲存為編號版本。發佈的版本無法變更。



您可以使用 Amazon Lex 主控台或 CreateBotVersion操作建立和發佈版本。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版本以建立新的版本，並將版本編號增加 1。版本編號絕不會重複使用。例如，如果您移除編號為版本 10 的資源，然後重新建立，Amazon Lex 指派的下一個版本編號是版本 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**
>
> 只有在上次發佈的版本與版本不同時，Amazon Lex 才會發佈新$LATEST版本。如果您嘗試發佈$LATEST版本而不修改，Amazon Lex 不會建立或發佈新版本。

更新 Amazon Lex 資源

您只能更新 Amazon Lex 機器人的$LATEST版本、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

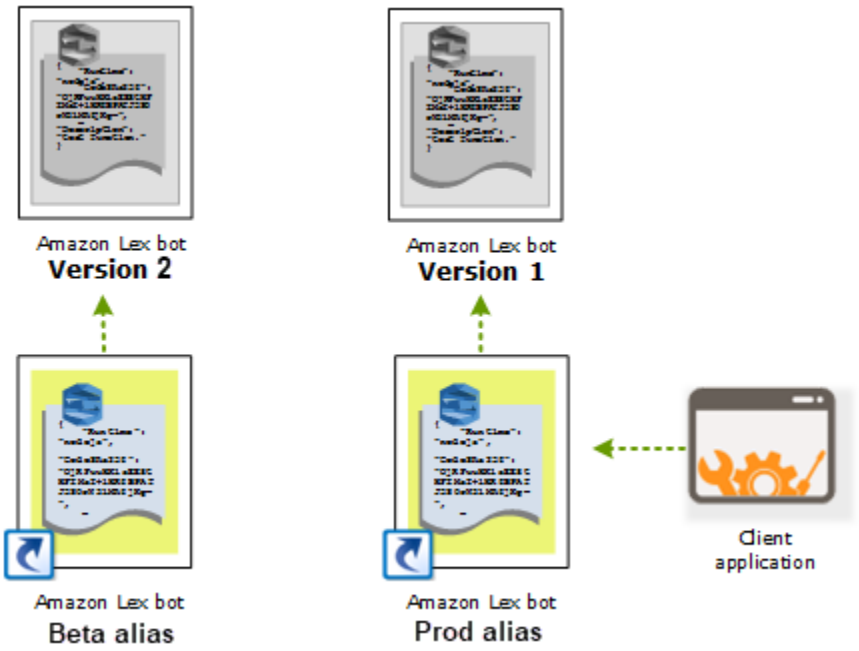
Amazon Lex 支援使用主控台或其中一個 API 操作刪除資源或版本：

- DeleteBot
- DeleteBotVersion
- DeleteBotAlias
- DeleteBotChannelAssociation
- DeleteIntent
- DeleteIntentVersion
- DeleteSlotType
- DeleteSlotTypeVersion

別名

別名是 Amazon Lex 機器人特定版本的指標。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。

下列範例顯示 Amazon Lex 機器人的兩個版本，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 會等到所有目前工作階段的工作階段逾時到期，再開始使用新版本。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 函數以用作 Amazon Lex 機器人的程式碼掛鉤。您可以識別 Lambda 函數，以在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，您的機器人會將意圖資訊傳回至用戶端應用程式以進行履行。

主題

Lambda 函數輸入事件和回應格式

本節說明 Amazon Lex 提供給 Lambda 函數的事件資料結構。使用此資訊來剖析 Lambda 程式碼中的輸入。它也會說明 Amazon Lex 預期 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。當您撰寫 Lambda 函數時，請使用此資訊。

> **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 認為目前意圖最符合使用者目前意圖的可信度。

  slots 是針對意圖設定的槽名稱對應至 Amazon Lex 在使用者對話中辨識的槽值。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，如果使用者回應提示「您想要什麼顏色的汽車？」 搭配「pizza」，Amazon Lex 將傳回「pizza」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「您想要訂購大型起司比薩嗎？」，取決於使用者回應，此欄位的值可以是 Confirmed或 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。」在此情況下，後續 Lambda 事件具有更新的槽值，PizzaSize設定為 medium。Amazon Lex 會將 confirmationStatus設定為 None，因為使用者修改了一些槽資料，要求 Lambda 函數執行使用者資料驗證。

- alternativeIntents – 如果您啟用可信度分數，Amazon Lex 最多會傳回四個替代意圖。每個意圖都包含一個分數，指出 Amazon Lex 根據使用者的表達用語，認為意圖是正確意圖的可信度。

  替代意圖的內容與 currentIntent 欄位的內容相同。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 – 此值由用戶端應用程式提供。Amazon Lex 會將它傳遞給 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 – 若要指出 Amazon Lex 調用 Lambda 函數的原因，它會將此設定為下列其中一個值：

  - `DialogCodeHook` – Amazon Lex 會設定此值來指示 Lambda 函數初始化函數，並驗證使用者的資料輸入。

    當意圖設定為調用 Lambda 函數做為初始化和驗證程式碼掛勾時，Amazon Lex 會在 Amazon Lex 了解意圖之後，在每個使用者輸入（表達式）上調用指定的 Lambda 函數。

    > **Note**
    >
    > 如果意圖不明確，Amazon Lex 無法叫用 Lambda 函數。

  - `FulfillmentCodeHook` – Amazon Lex 會設定此值來指示 Lambda 函數實現意圖。

    如果意圖設定為調用 Lambda 函數做為履行程式碼掛勾，Amazon Lex 只會在擁有所有槽資料以滿足意圖之後，才會將 `invocationSource` 設定為此值。

在您的意圖組態中，您可以有兩個單獨的 Lambda 函數來初始化和驗證使用者資料，以及實現意圖。您也可以使用一個 Lambda 函數來執行兩者。在這種情況下，您的 Lambda 函數可以使用 `invocationSource`值來遵循正確的程式碼路徑。

- outputDialogMode – 對於每個使用者輸入，用戶端會使用其中一個執行時間 API 操作，PostContent或 ，將請求傳送至 Amazon LexPostText。Amazon Lex 使用請求參數來判斷對用戶端的回應是文字還是語音，並相應地設定此欄位。

  Lambda 函數可以使用此資訊來產生適當的訊息。例如，如果用戶端預期語音回應，您的 Lambda 函數可能會傳回語音合成標記語言 (SSML)，而不是文字。

- messageVersion – 訊息的版本，可識別進入 Lambda 函數的事件資料格式，以及來自 Lambda 函數的預期回應格式。

  > **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 – 用戶端在請求中傳送的應用程式特定工作階段屬性。如果您希望 Amazon Lex 將它們包含在用戶端的回應中，您的 Lambda 函數應在回應中將這些函數傳回 Amazon Lex。如需詳細資訊，請參閱設定工作階段屬性

- requestAttributes – 用戶端在請求中傳送的請求特定屬性。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 – 意圖狀態的相關資訊。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 – 最後表達用語的 Amazon Comprehend 情緒分析結果。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。

- activeContexts – 與此使用者對話期間處於作用中狀態的一或多個內容。
  - timeToLive – 與使用者對話中內容保持作用中的時間長度或轉數。
  - name – 內容的名稱。
  - 參數 的鍵/值對清單包含啟動內容之意圖的槽名稱和值。

  如需詳細資訊，請參閱設定意圖內容。

回應格式

Amazon Lex 預期 Lambda 函數會以下列格式回應：

```
{
  "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。sessionAttributes、 recentIntentSummaryView和 activeContexts 欄位為選用， dialogAction 欄位為必要欄位。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。如果您的 Lambda 函數未傳回工作階段屬性，則透過 API 或 Lambda 函數sessionAttributes傳遞的最後一個已知值會保留。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"
    }
 ]
```

activeContexts

選用。如果包含，請設定一或多個內容的值。例如，您可以包含內容，讓一或多個意圖將該內容做為輸入，在對話的下一個回合中符合辨識資格。

回應中未包含的任何作用中內容都會減少其time-to-live並且可能仍在下次請求時處於作用中狀態。

如果您為輸入事件中包含的內容指定time-to-live，則其將在下次請求時處於非作用中狀態。

如需詳細資訊，請參閱設定意圖內容。

dialogAction

必要. dialogAction 欄位會引導 Amazon Lex 進行下一個動作，並說明在 Amazon Lex 將回應傳回給用戶端後，使用者預期會發生的情況。

type 欄位會指出後續動作，它也會決定 Lambda 函數需要作為dialogAction值的一部分提供的其他欄位。

- Close — 通知 Amazon Lex 不要預期來自使用者的回應。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 使用此值來設定 中的 dialogState 欄位PostContent或對用戶端應用程式的PostText回應。message 和 responseCard 欄位是選用的。如果您未指定訊息，Amazon Lex 會使用為意圖設定的再見訊息或後續訊息。

```
"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 使用者應該提供是或否的答案，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。Lambda 函數傳回message的欄位內容優先於意圖中confirmationPrompt指定的 。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 — 指示 Amazon Lex 根據機器人組態選擇下一個動作。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 使用者應該回應包含意圖的表達用語。例如，「我想要大型比薩。」，亦即指出 OrderPizzaIntent。另一方面，表達用語「大」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釐清提示。如果未定義釐清提示，Amazon Lex 會傳回 400 錯誤的請求例外狀況。

```
{
  "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 預期使用者會在回應中提供槽值。

  `intentName`、`slotToElicit` 和 `slots` 欄位是必要的。`message` 和 `responseCard` 欄位是選用的。如果您未指定訊息，Amazon Lex 會使用針對插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 Lambda 藍圖

Amazon Lex 主控台提供預先設定的範例機器人（稱為機器人藍圖），讓您可以在主控台中快速建立和測試機器人。對於這些機器人藍圖，也會提供 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以 Lambda 函數做為程式碼掛勾設定的機器人，並測試end-to-end設定，而不必撰寫程式碼。

您可以使用下列 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，並將其設定為使用 Lambda 函數做為程式碼掛勾，請參閱 練習 1：使用藍圖建立 Amazon Lex 機器人（主控台）。如需使用其他藍圖的範例，請參閱其他範例：建立 Amazon Lex 機器人。

更新特定地區設定的藍圖

如果您在英文 (US) (en-US) 以外的地區使用藍圖，則需要更新任何意圖的名稱，以包含地區設定。例如，如果您使用`OrderFlowers`藍圖，則需要執行下列動作。

- 尋找靠近 Lambda dispatch函數程式碼結尾的函數。
- 在 dispatch函數中，更新意圖的名稱，以包含您正在使用的地區設定。例如，如果您使用的是英文（澳洲）(en-AU) 地區設定，請變更行：

  ```
  if intent_name == 'OrderFlowers':
  ```

  至

  ```
  if intent_name == 'OrderFlowers_enAU':
  ```

其他藍圖使用其他意圖名稱，應該在使用前如上述進行更新。

部署 Amazon Lex 機器人

本節提供在各種傳訊平台和行動應用程式中部署 Amazon Lex 機器人的範例。

主題

在訊息平台上部署 Amazon Lex 機器人

本節說明如何在 Facebook、Slack 和 Twilio 訊息平台上部署 Amazon Lex 機器人。

> **Note**
>
> 儲存 Facebook、Slack 或 Twilio 組態時，Amazon Lex 會使用 AWS Key Management Service 客戶受管金鑰來加密資訊。您第一次建立其中一個傳訊平台的頻道時，Amazon Lex 會建立預設的客戶受管金鑰 (aws/lex)。或者，您可以使用 建立自己的客戶受管金鑰 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》。

當訊息平台傳送請求至 Amazon Lex 時，它會將平台特定資訊做為請求屬性包含到您的 Lambda 函數。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。下表列出傳訊平台傳送至機器人 Lambda 函數的請求屬性。

常見的請求屬性

屬性	描述
`channel-id`	來自 Amazon Lex 的頻道端點識別符。
`channel-name`	來自 Amazon Lex 的頻道名稱。
`channel-type`	下列其中一值：

屬性	描述
	• Facebook • Kik • Slack • Twilio-SMS
webhook-endpoint-url	頻道的 Amazon Lex 端點。

Facebook 請求屬性

屬性	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	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	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	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。

將 Amazon Lex 機器人與 Facebook Messenger 整合

本練習說明如何將 Facebook Messenger 與您的 Amazon Lex 機器人整合。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. 將 Facebook Messenger 與您的 Amazon Lex 機器人整合
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，請建立並部署機器人。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：使用藍圖建立 Amazon Lex 機器人（主控台）。

1. 建立 Amazon Lex 機器人。如需說明，請參閱練習 1：使用藍圖建立 Amazon Lex 機器人（主控台）。
2. 部署機器人並建立別名。如需說明，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

步驟 3：整合 Facebook Messenger 與 Amazon Lex Bot

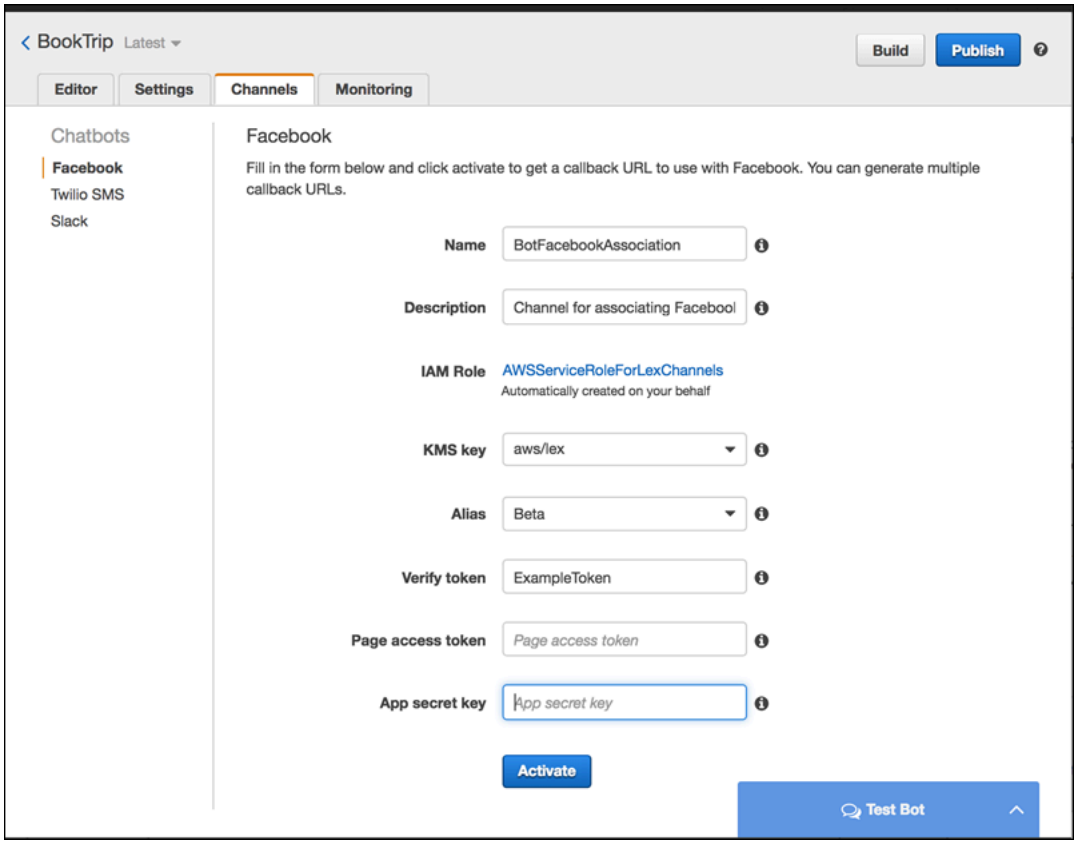
在本節中，您將 Facebook Messenger 與您的 Amazon Lex 機器人整合。

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 管理主控台 ，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：

- 針對回呼 URL，輸入程序中稍早在 Amazon Lex 主控台提供的回呼 URL。
- 針對驗證權杖，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。這會在 Facebook 和 Amazon Lex 之間啟動交握。

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

您現在可以從 Facebook Messenger 開始與 Amazon Lex 機器人的對話。

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（主控台）所提供測試用的同一組表達用語。

將 Amazon Lex 機器人與 Kik 整合

本練習提供將 Amazon Lex 機器人與 Kik 訊息應用程式整合的說明。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 Amazon Lex 機器人與 Kik 機器人整合。
4. 使用 Kik 與您的 Amazon Lex 機器人進行對話，以測試 Amazon Lex 機器人與 Kik 之間的關聯。

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：使用藍圖建立 Amazon Lex 機器人（主控台）

1. 建立 Amazon Lex 機器人。如需說明，請參閱練習 1：使用藍圖建立 Amazon Lex 機器人（主控台）。
2. 部署機器人並建立別名。如需說明，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，將其連線至 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 機器人整合

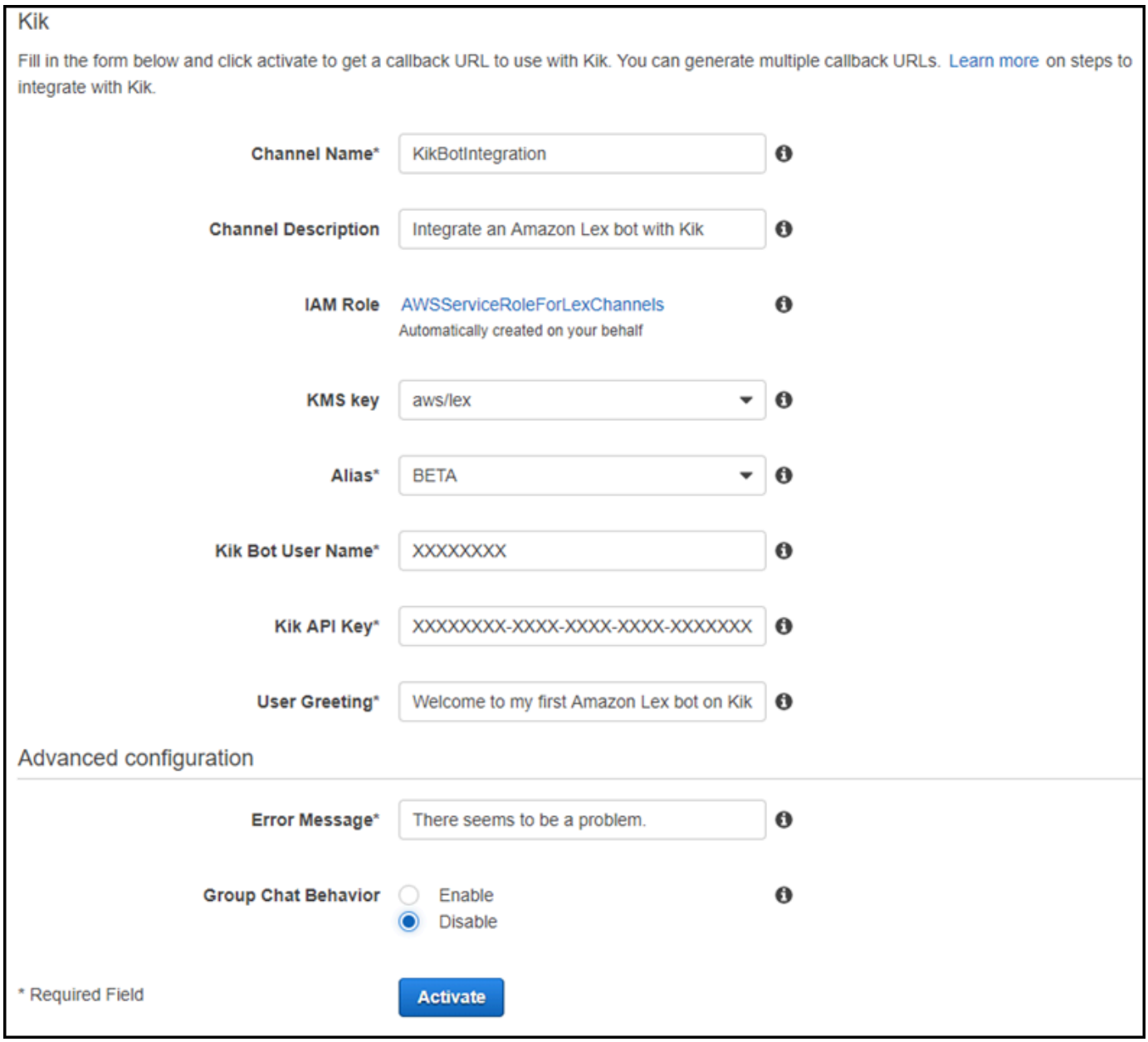
步驟 3：將 Kik 機器人與 Amazon Lex 機器人整合

現在您已建立 Amazon Lex 機器人和 Kik 機器人，即可在 Amazon Lex 中建立頻道關聯。啟用關聯時，Amazon Lex 會自動使用 Kik 設定回呼 URL。

1. 登入 AWS 管理主控台，並在 https：//https://console.aws.amazon.com/lex/ 開啟 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. 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

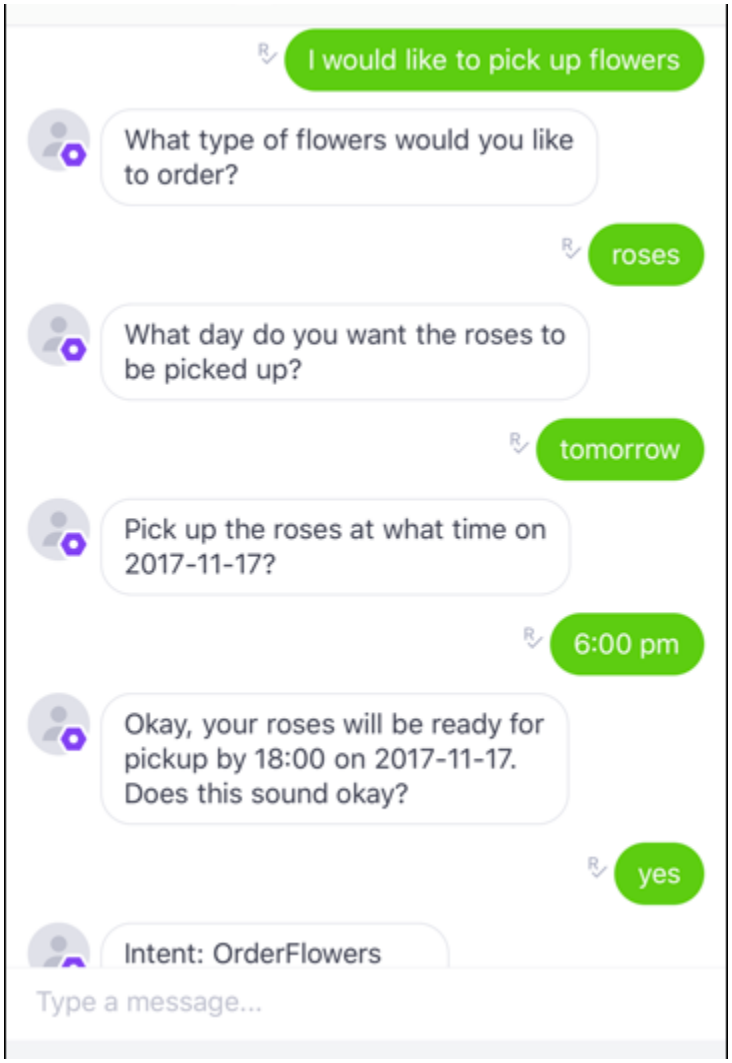
[步驟 4：測試整合](#)

步驟 4：測試整合

現在您已在 Amazon Lex 機器人和 Kik 之間建立關聯，您可以使用 Kik 應用程式來測試關聯。

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：



當您輸入每個片語時，Amazon Lex 機器人會透過 Kik 回應您為每個槽建立的提示。

將 Amazon Lex 機器人與 Slack 整合

本練習提供將 Amazon Lex 機器人與 Slack 訊息應用程式整合的說明。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：使用藍圖建立 Amazon Lex 機器人（主控台）

1. 建立 Amazon Lex 機器人。如需說明，請參閱練習 1：使用藍圖建立 Amazon Lex 機器人（主控台）。
2. 部署機器人並建立別名。如需說明，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，您可以使用此資訊在 Amazon Lex 主控台中設定機器人頻道關聯。

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，選擇互動與捷徑。
     - 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。您將在 Amazon Lex 主控台中新增機器人頻道關聯後更新此 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

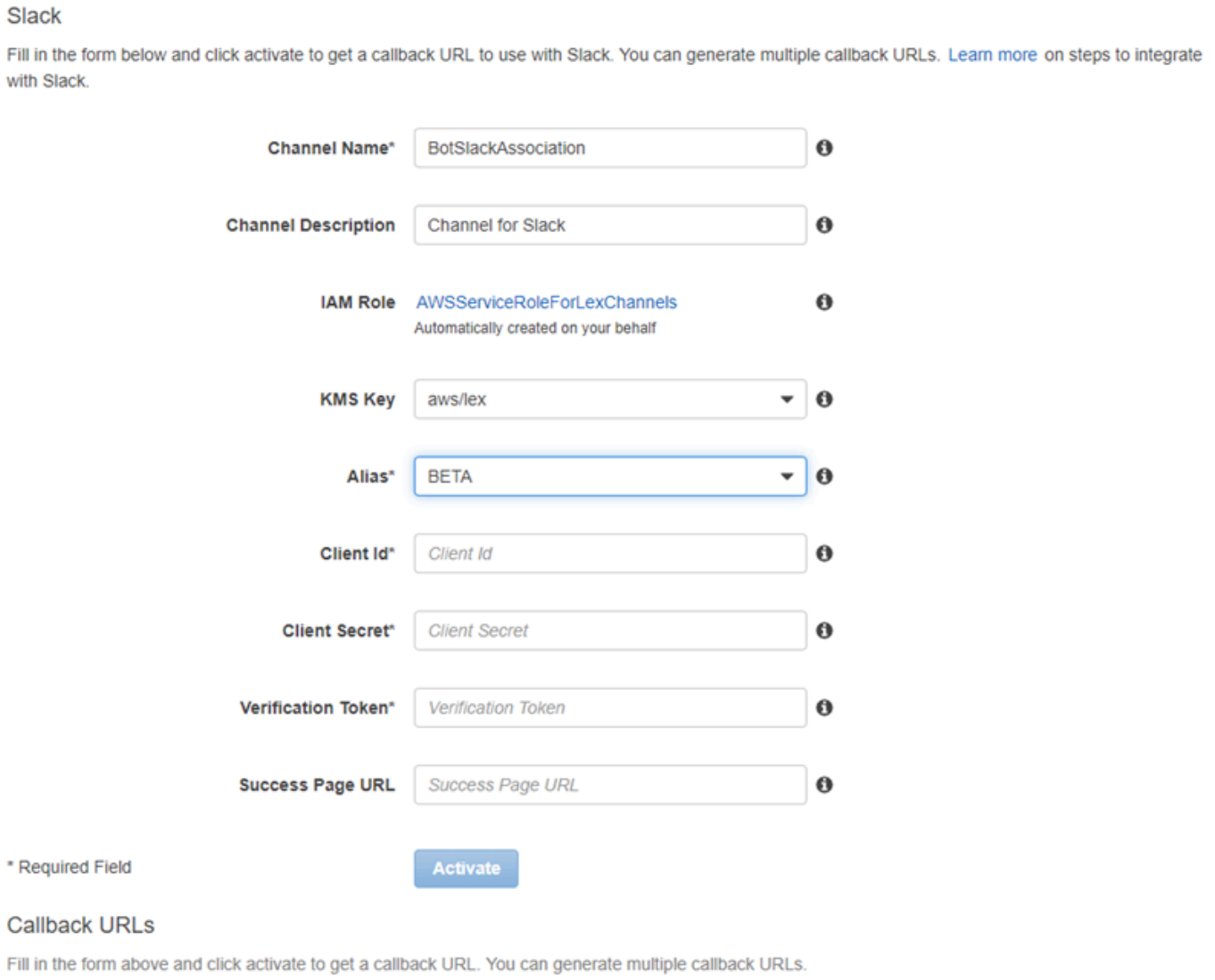
步驟 4：整合 Slack 應用程式與 Amazon Lex 機器人

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 中新增機器人頻道關聯。

在 Amazon Lex 主控台中，啟用機器人頻道關聯，將機器人與您的 Slack 應用程式建立關聯。啟用機器人頻道關聯時，Amazon Lex 會傳回兩個 URLs(後退 URL 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，並在 https：//https://console.aws.amazon.com/lex/ 開啟 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：

   - Postback URL 是接聽 Slack 事件的 Amazon Lex 機器人端點。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 URL 是 Amazon Lex 機器人與 Slack 進行 OAuth 交握的端點。

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 URLs區段中，新增 Amazon Lex 在上一個步驟中提供的 OAuth URL。選擇 Add a new Redirect URL (新增重新導向 URL)，然後選擇 Save URLs (儲存 URL)。
   c. 在機器人字符範圍區段中，使用新增 OAuth 範圍按鈕新增兩個許可。以下列文字篩選清單：
      - **`chat:write`**
      - **`team:read`**
3. 透過將請求 URL 值更新為 Amazon Lex 在上一個步驟中提供的後退 URL 來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後退 URL。
   - 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. 從訊息索引標籤啟用傳送訊息，如下所示：
   - 從左側選單中，選擇應用程式首頁。
   - 在顯示標籤區段中，選擇允許使用者從訊息索引標籤傳送斜線命令和訊息。

後續步驟

步驟 6：測試整合

步驟 6：測試整合

現在請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. 與連結到 Amazon Lex 機器人的 Slack 應用程式進行聊天。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 Code Hook （主控台）。

將 Amazon Lex 機器人與 Twilio 可程式設計簡訊整合

本練習提供將 Amazon Lex 機器人與 Twilio 簡單簡訊服務 (SMS) 整合的說明。您會執行以下步驟：

1. 建立 Amazon Lex 機器人
2. 將 Twilio 可程式 SMS 與您的機器人 Amazon Lex 整合
3. 在行動電話上使用 SMS 服務測試設定，以與 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 Messaging Service Endpoint 與 Amazon Lex Bot
- 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 機器人，請建立並部署機器人。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：使用藍圖建立 Amazon Lex 機器人 （主控台）。

1. 建立 Amazon Lex 機器人。如需說明，請參閱練習 1：使用藍圖建立 Amazon Lex 機器人（主控台）。
2. 部署機器人並建立別名。如需說明，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 Messaging Service Endpoint 與 Amazon Lex Bot

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 主控台中啟用機器人頻道關聯。啟用機器人頻道關聯後，Amazon Lex 會傳回回呼 URL。請記錄此回呼 URL，因為稍後將會用到。

   a. 登入 AWS 管理主控台 ，並在 https://console.aws.amazon.com/lex/：// 開啟 Amazon Lex 主控台。

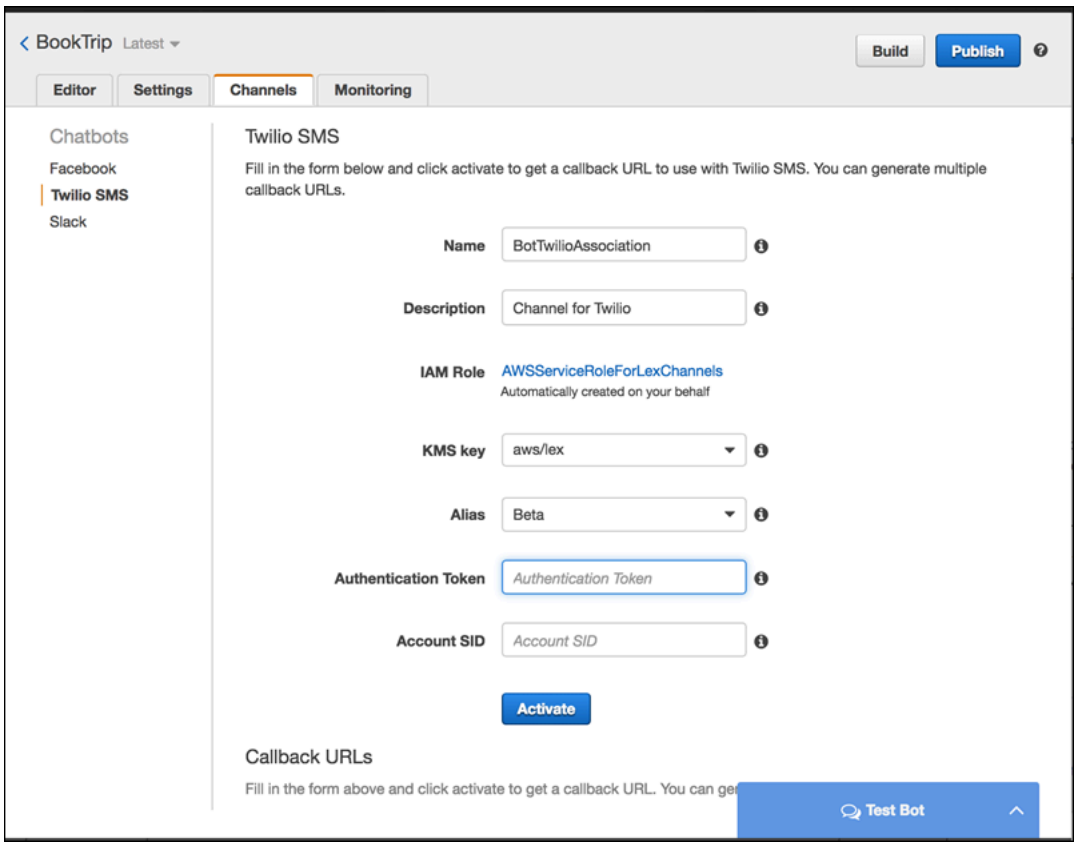
   b. 選擇您在步驟 1 中建立的 Amazon Lex 機器人。

   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 SMS 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 REQUEST URL 值設定為 Amazon Lex 在上述步驟中提供的回呼 URL，以更新簡訊服務的傳入設定組態。

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。

      您傳送訊息至此號碼，並從行動電話與 Amazon Lex 機器人進行簡訊互動。

   b. 確認您的行動電話列為已驗證來電者 ID。

      如果不是，請遵循 Twilio 主控台上的指示，以啟用您計劃用於測試的行動電話。

      現在，您可以使用行動電話傳送訊息至映射至 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 Code Hook （主控台）。

在行動應用程式中部署 Amazon Lex 機器人

使用 AWS Amplify，您可以將 Amazon Lex 機器人與行動或 Web 應用程式整合。如需詳細資訊，請參閱 AWS Amplify 文件中的互動 – 入門。

匯入和匯出 Amazon Lex 機器人、意圖和槽類型

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。

您可以匯出 Amazon Lex（共用或修改它們）或 Alexa 技能格式的機器人、意圖和槽類型。您只能以 Amazon Lex 格式匯入。

匯出資源時，您必須以與您匯出至 Amazon Lex 或 Alexa Skills Kit 的服務相容的格式匯出資源。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、意圖或槽類型，您可以使用 Amazon Lex 主控台或 GetExport操作。請使用 StartImport 匯入機器人、意圖或參數槽類型。

主題

- 以 Amazon Lex 格式匯出和匯入
- 匯出至 Alexa 技能

以 Amazon Lex 格式匯出和匯入

若要從 Amazon Lex 匯出機器人、意圖和槽類型，以便重新匯入 Amazon Lex，您可以使用 Amazon Lex 格式的建立 JSON 檔案。您可以編輯此檔案中的資源，並將其匯入 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 格式匯出
- 以 Amazon Lex 格式匯入

- 匯入及匯出的 JSON 格式

以 Amazon Lex 格式匯出

將您的 Amazon Lex 機器人、意圖和槽類型匯出為您可以匯入 AWS 帳戶的格式。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。匯出完成時，您會取得 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. 登入 AWS 管理主控台，並在 https：//https://console.aws.amazon.com/lex/ 開啟 Amazon Lex 主控台。
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 會將機器人匯出至 .zip 封存檔中包含的 JSON 檔案。若要更新機器人，請修改 JSON 文字，然後將其匯入回 Amazon Lex。

下一步驟

以 Amazon Lex 格式匯入

以 Amazon Lex 格式匯入

將資源匯出至 Amazon Lex 格式的 JSON 檔案後，您可以將包含資源的 JSON 檔案匯入一或多個 AWS 帳戶。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「Amazon Lex 如何與 IAM 搭配使用」。

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，如果匯入意圖的帳戶沒有呼叫意圖使用之 Lambda 函數的許可，則在對槽類型或意圖進行變更之前，匯入會失敗。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 機器人的 `enableModelImprovements` 參數設定為 `false`，則必須開啟包含機器人定義的 .zip 檔案，並將 `enableModelImprovements` 參數變更為`true`下列區域中的 ：
>
> - 亞太地區 (新加坡) (ap-southeast-1)
> - 亞太地區 (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. 登入 AWS 管理主控台，並在 https：//https://console.aws.amazon.com/lex/ 開啟 Amazon Lex 主控台。
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

下列範例顯示以 Amazon Lex 格式匯出和匯入槽類型、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。

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。

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 管理主控台 ，並在 https://console.aws.amazon.com/lex/ ：// 開啟 Amazon Lex 主控台。
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 和 Alexa Skills Kit 有下列不同：
>
> - 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 檔案拖曳至 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)。

其他範例：建立 Amazon Lex 機器人

以下各節提供其他 Amazon Lex 練習和step-by-step說明。

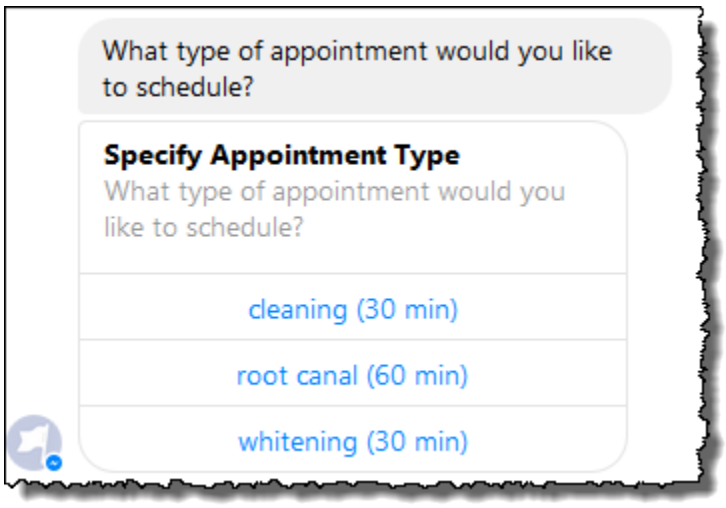
主題

排程預約

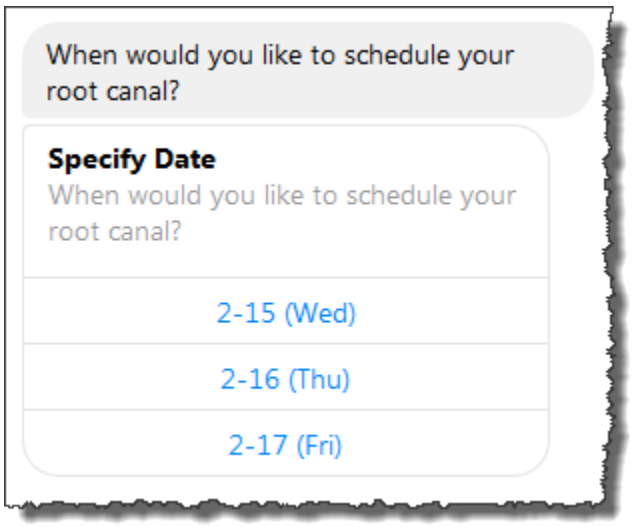
在本練習中的範例機器人會為牙醫診所排定預約。範例同時說明如何使用回應卡，利用按鈕來取得使用者輸入。範例特別說明以動態方式在執行時間產生回應卡。

您可以在建置時間設定回應卡（也稱為靜態回應卡），或在 AWS Lambda 函數中動態產生回應卡。在此範例中，機器人使用以下回應卡：

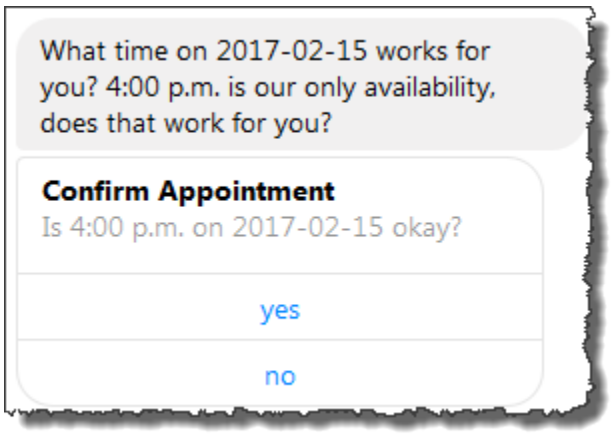
- 列出預約類型按鈕的回應卡。如需範例，請參閱下圖：



- 列出預約日期按鈕的回應卡。如需範例，請參閱下圖：



- 列出按鈕以確認建議之預約時間的回應卡。如需範例，請參閱下圖：



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。您可以使用 AWS Lambda 函數來動態產生這些回應卡。Lambda 函數會在回應 Amazon Lex 時傳回回應卡。Amazon Lex 會在回應用戶端時包含回應卡。

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：

- 建立和測試機器人 (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 Lambda 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 函數做為程式碼掛勾，以執行初始化、驗證和履行任務。

  > **Note**
  >
  > 提供的範例 Lambda 函數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 函數做為程式碼掛勾。然後，測試端對端體驗。
- 將排程預約機器人發佈到 Facebook Messenger，以便您可以查看作用中的回應卡 (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)
- Lambda 函數藍圖概觀 (lex-make-appointment-python)
- 步驟 1：建立 Amazon Lex 機器人
- 步驟 2：建立 Lambda 函數
- 步驟 3：更新意圖：設定程式碼掛勾
- 步驟 4：將機器人部署在 Facebook Messenger 平台上
- 資訊流程的詳細資訊

機器人藍圖概觀 (ScheduleAppointment)

在本練習中，您用來建立機器人的 ScheduleAppointment 藍圖已預先設定以下設定：

- 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」
    - 「預約 {AppointmentType}」

    如果使用者說出其中任何內容，Amazon Lex 會判斷 MakeAppointment是意圖，然後使用提示來引出槽資料。
  - 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？」
    - Date 槽的提示 -「我應該何時排定您的 {AppointmentType}？」
    - Time 槽提示 –「您想要在何時排程 {AppointmentType}？」 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 ScheduleAppointment 機器人藍圖建立之機器人的程式碼掛勾。

此 Lambda 函數藍圖程式碼可以同時執行初始化/驗證和履行任務。

- Lambda 函數程式碼顯示以牙醫預約的範例可用性為基礎的動態對話 （在實際應用程式中，您可以使用行事曆）。對於使用者指定的一天或日期，程式碼設定如下：
  - 如果沒有可用的預約，Lambda 函數會傳回回應，指示 Amazon Lex 提示使用者改日或日期（將dialogAction類型設定為 ElicitSlot)。如需詳細資訊，請參閱回應格式。

- 如果指定的日期只有一個預約可用，Lambda 函數會建議回應中的可用時間，並指示 Amazon Lex 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。
- 如果有多個可用的預約，Lambda 函數會在回應 Amazon Lex 時傳回可用時間的清單。Amazon Lex 會傳回回應給用戶端，其中包含來自 Lambda 函數的訊息。

- 做為履行程式碼掛勾，Lambda 函數會傳回摘要訊息，指出已排定預約 （ 也就是已履行意圖 ）。

> **Note**
>
> 在這個範例中，我們示範如何使用回應卡。Lambda 函數會建構回應卡並將其傳回給 Amazon Lex。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，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。如需說明，請參閱將 Amazon Lex 機器人與 Facebook Messenger 整合。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，它會傳遞事件資料做為輸入。其中一個事件欄位是`invocationSource`，Lambda 函數會使用此欄位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：建立 Amazon Lex 機器人

步驟 1：建立 Amazon Lex 機器人

在本節中，您可以使用 ScheduleAppointment 藍圖建立 Amazon Lex 機器人，該藍圖在 Amazon Lex 主控台中提供。

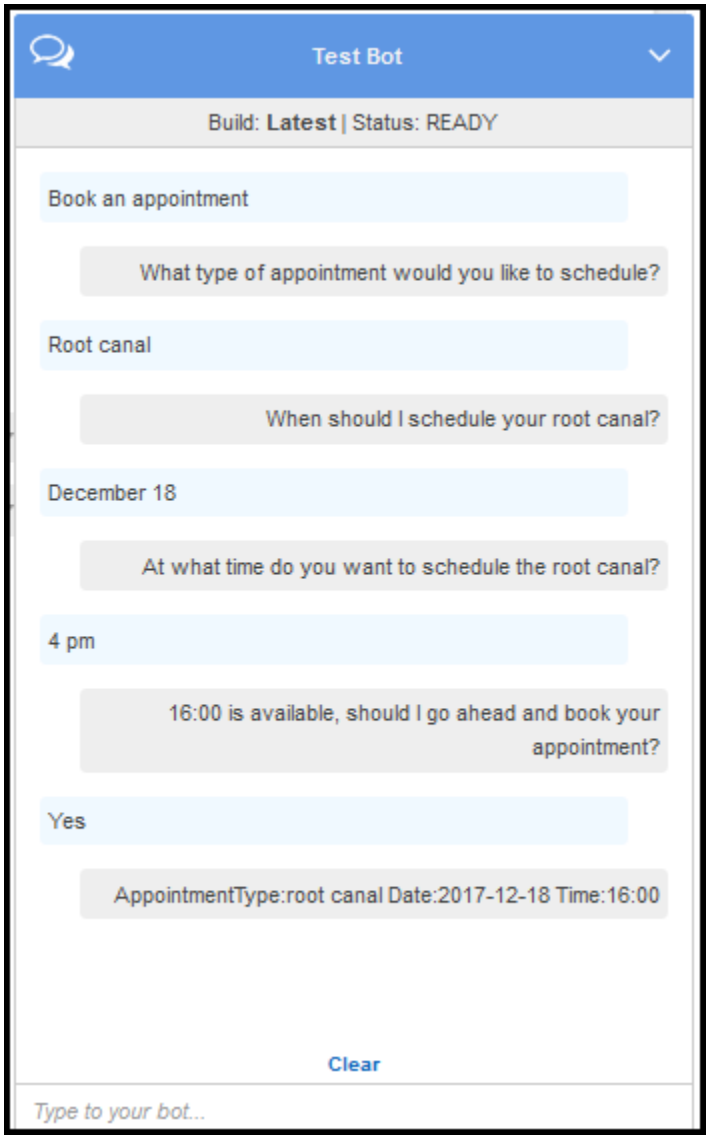
1. 登入 AWS 管理主控台 並開啟位於 https：//https://console.aws.amazon.com/lex/ 的 Amazon Lex 主控台。
2. 在 Bots (機器人) 頁面，選擇 Create (建立)。
3. 在 [Create your Lex bot] 頁面，執行下列動作：
   - 選擇 ScheduleAppointment 藍圖。
   - 保留預設機器人名稱 (ScheduleAppointment)。
4. 選擇建立。

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 為機器人建置機器學習模型。當您在主控台中測試機器人時，主控台會使用執行時間 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. 主控台會顯示 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 事件資料來叫用 Lambda 函數，以測試 Lambda 函數。

1. 登入 AWS 管理主控台 ，並在 https：//https://console.aws.amazon.com/lambda/ 開啟 AWS Lambda 主控台。
2. 選擇 Create a Lambda function (建立 Lambda 函數)。
3. 對於 Select blueprint (選取藍圖)，輸入 **`lex`** 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. 如果您使用英文 (US) (en-US) 以外的地區設定，請更新意圖名稱，如中所述更新特定地區設定的藍圖。
7. 測試 Lambda 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，並將值設定為符合來自 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 函數已成功執行。在此情況下，回應符合 Amazon Lex 回應模型。

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勾。

1. 在 Amazon Lex 主控台中，選取 ScheduleAppointment 機器人。主控台會顯示 MakeAppointment 意圖。如下修改意圖組態。

   > **Note**
   >
   > 您只能更新任何 Amazon Lex 資源的 $LATEST 版本，包括意圖。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。

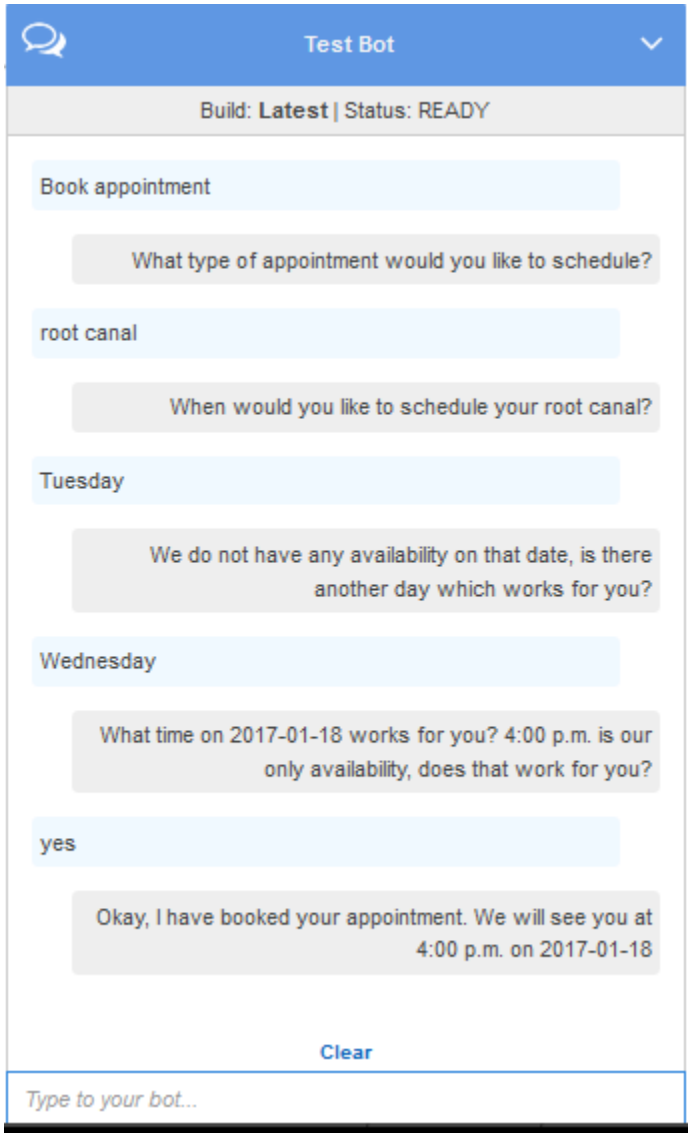
   a. 在選項區段中，選擇初始化和驗證程式碼掛鉤，然後從清單中選擇 Lambda 函數。

   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 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。

如需說明，請參閱將 Amazon Lex 機器人與 Facebook Messenger 整合。

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 Amazon Lex 的回應中包含回應卡。Amazon Lex 會在回覆用戶端時包含回應卡。本部分說明下列兩項：

- 用戶端與 Amazon Lex 之間的資料流程。

  本節假設用戶端使用PostText執行時間 API 將請求傳送至 Amazon Lex，並相應地顯示請求/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**
  >
  > 如需用戶端與用戶端使用 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 函數之間的資料流程。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 Facebook Messenger 用戶端，該用戶端不會在請求中將工作階段屬性傳遞給 Amazon 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來測試機器人，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 和內文都會提供資訊給 Amazon Lex：

- 請求 URI – 提供機器人名稱 (ScheduleAppointment)、機器人別名 ($LATEST) 和使用者名稱 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 `inputText`，Amazon Lex 會偵測意圖 (`MakeAppointment`)。服務會叫用設定為程式碼掛勾的 Lambda 函數，透過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 函數調用的目的。在這種情況下，目的是執行使用者資料初始化和驗證。(Amazon Lex 知道使用者尚未提供所有槽資料以滿足意圖。)
- messageVersion – 目前 Amazon Lex 僅支援 1.0 版本。

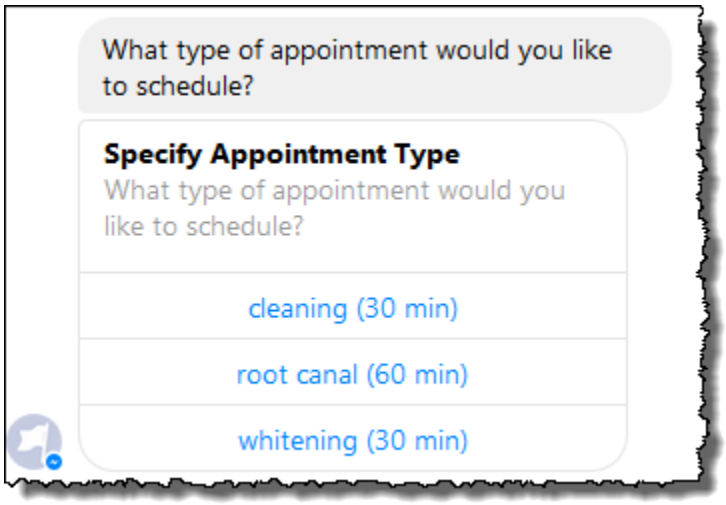
c. 目前，所有槽值都是 null (沒有可驗證的內容)。Lambda 函數會傳回下列回應給 Amazon Lex，引導服務引出AppointmentType插槽的資訊。如需回應格式的相關資訊，請參閱回應格式。

```
{
    "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,
```

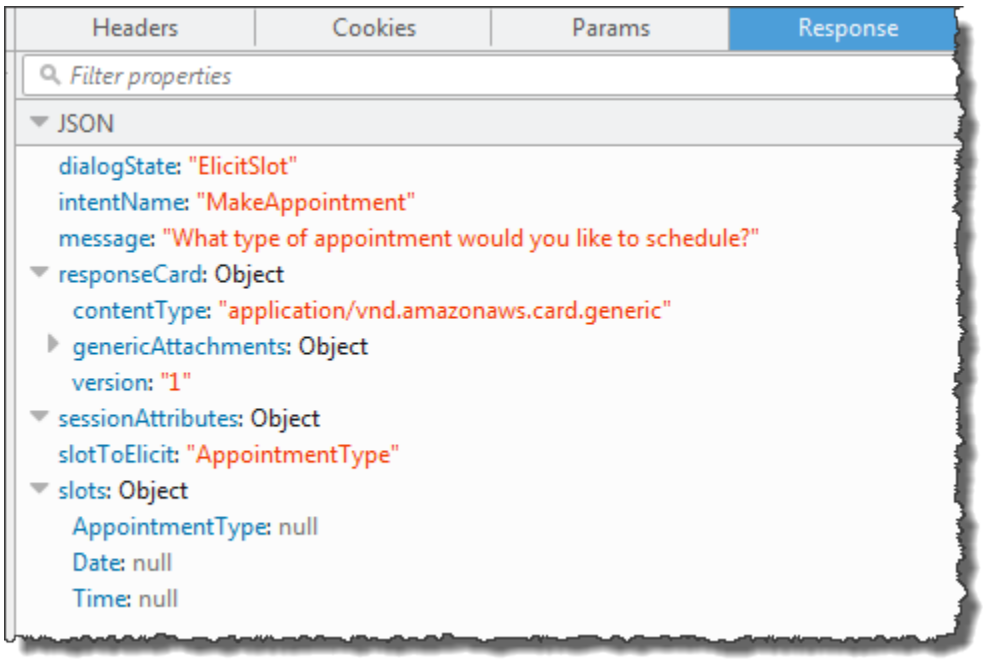
```
            "Date": null,
            "Time": null
        },
        "type": "ElicitSlot",
        "message": {
            "content": "What type of appointment would you like to schedule?",
            "contentType": "PlainText"
        }
    },
    "sessionAttributes": {}
}
```

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 引出slotToElicit欄位中指定插槽的值。Lambda 函數也提供 message以傳達給使用者。
- responseCard – 識別AppointmentType槽的可能值清單。支援回應卡的用戶端 （例如 Facebook Messenger) 會顯示回應卡，以允許使用者選擇預約類型，如下圖所示：



d. 如 Lambda 函數回應dialogAction.type中的 所示，Amazon Lex 會將下列回應傳回給用戶端：



用戶端會讀取回應，然後顯示訊息：「您想要安排哪種類型的預約？」 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`請求傳送至 Amazon Lex （已新增換行以方便閱讀）：

```
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。
- Amazon Lex 會將currentIntent.slots插槽中的 AppointmentType 欄位設定為 root canal。
- Amazon Lex 只會在用戶端和 Lambda 函數之間傳遞 sessionAttributes 欄位。

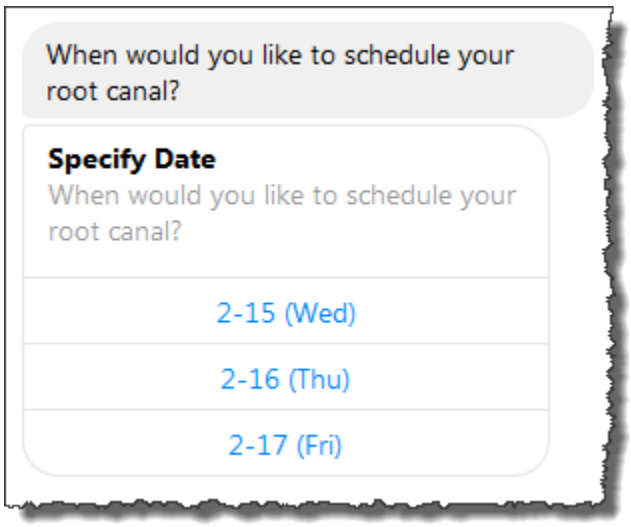
c. Lambda 函數會驗證使用者輸入，並將下列回應傳回 Amazon Lex，指示服務引出預約日期的值。

```
{
    "dialogAction": {
        "slotToElicit": "Date",
        "intentName": "MakeAppointment",
        "responseCard": {
            "genericAttachments": [
                {
                    "buttons": [
                        {
                            "text": "2-15 (Wed)",
                            "value": "Wednesday, February 15, 2017"
                        },
```

```
                        {
                            "text": "2-16 (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：

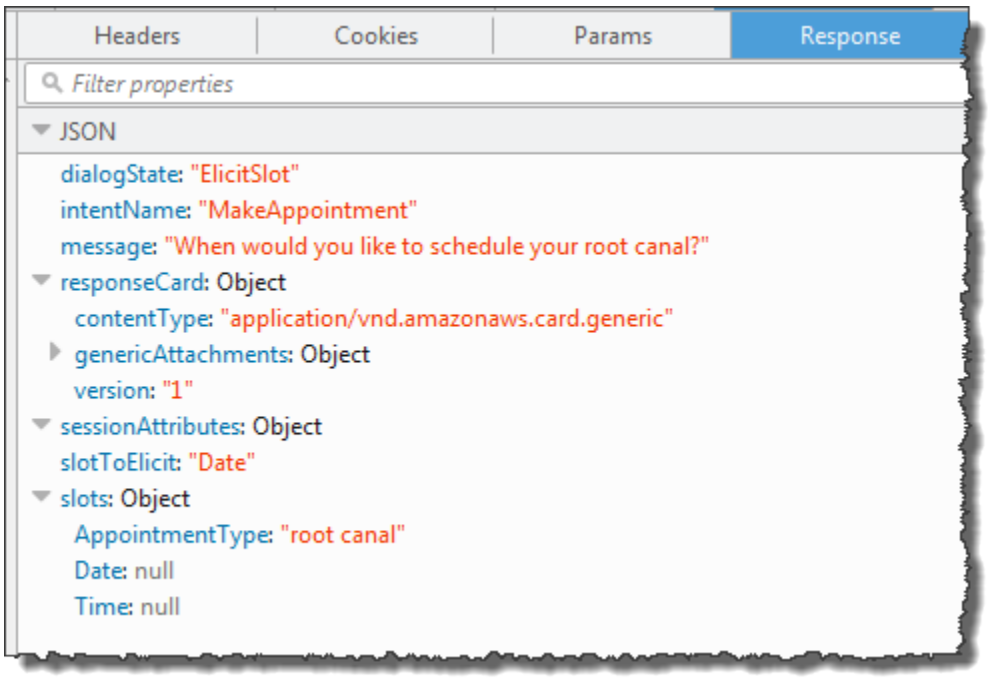
- `type` – 透過將此欄位設定為 `ElicitSlot`，Lambda 函數會指示 Amazon Lex 引出`slotToElicit`欄位中指定插槽的值。Lambda 函數也提供 `message`以傳達給使用者。
- `responseCard` – 識別`Date`槽的可能值清單。支援回應卡 （ 例如 Facebook Messenger) 的用戶端會顯示回應卡，允許使用者選擇預約日期，如下圖所示：



雖然 Lambda 函數傳回五個日期，但用戶端 (Facebook Messenger) 對於回應卡有三個按鈕的限制。因此，您在螢幕擷取畫面中只會看到前三個值。

這些日期會在 Lambda 函數中硬式編碼。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，您必須在 Lambda 函數中動態產生回應卡。

d. Amazon Lex 注意到 `dialogAction.type`，並將下列回應傳回給用戶端，其中包含 Lambda 函數回應中的資訊。



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請求傳送至 Amazon Lex （已新增換行以方便閱讀）：

```
POST /bot/BookTrip/alias/$LATEST/user/bijt6rovckwecnzesbthrr1d7lv3ja3n/text
"Content-Type":"application/json"
"Content-Encoding":"amz-1.0"

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

   b. Amazon Lex 會在下列事件中以參數傳送 ，以叫用 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。
- Amazon Lex 會將currentIntent.slots插槽中的 Date 欄位設定為 2017-02-16。
- Amazon Lex 只會在用戶端和 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預約。它會傳回下列回應給 Amazon Lex，指示服務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"
                        },
                        {
```

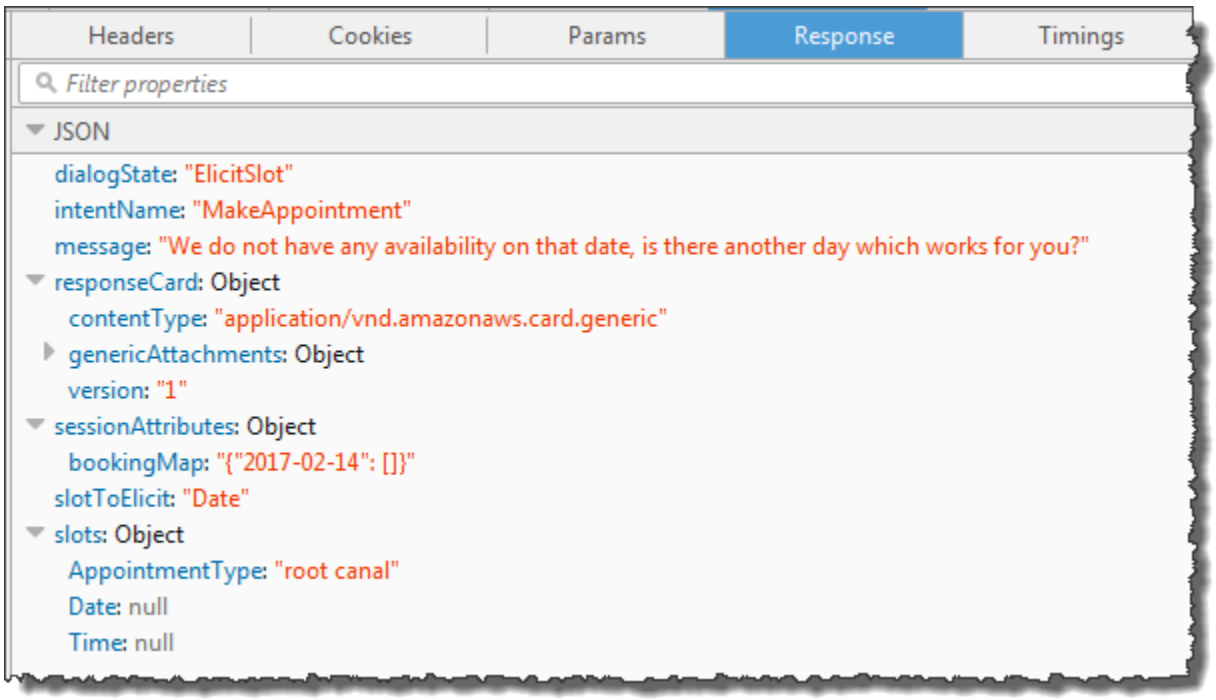
```
                            "text": "2-20 (Mon)",
                            "value": "Monday, February 20, 2017"
                        },
                        {
                            "text": "2-21 (Tue)",
                            "value": "Tuesday, February 21, 2017"
                        }
                    ],
                    "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 函數也提供適當的 message 以傳達給使用者。
  - 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 注意到 `dialogAction.type`，並將下列回應傳回給用戶端，其中包含 Lambda 函數回應中的資訊。



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`請求傳送至 Amazon Lex：

```
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持工作階段狀態，您必須在 Lambda 函數中執行此操作。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": {
    }
}
```

Amazon Lex 已更新`currentIntent.slots`，方法是將`Date`插槽設定為 `2017-02-15`。

c. Lambda 函數會驗證使用者輸入，並將下列回應傳回 Amazon Lex，並指示它引出預約時間的值。

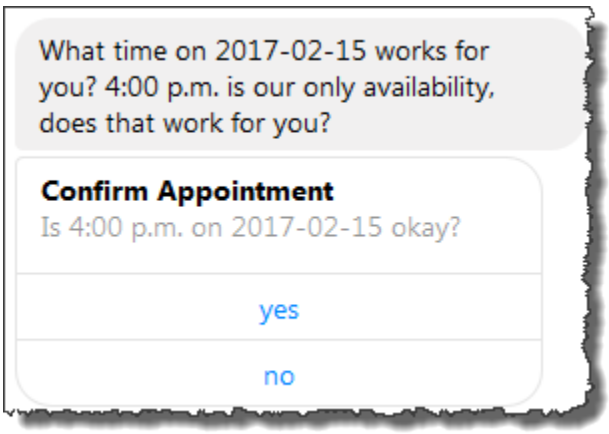
```
{
    "dialogAction": {
        "slots": {
```

```
            "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：

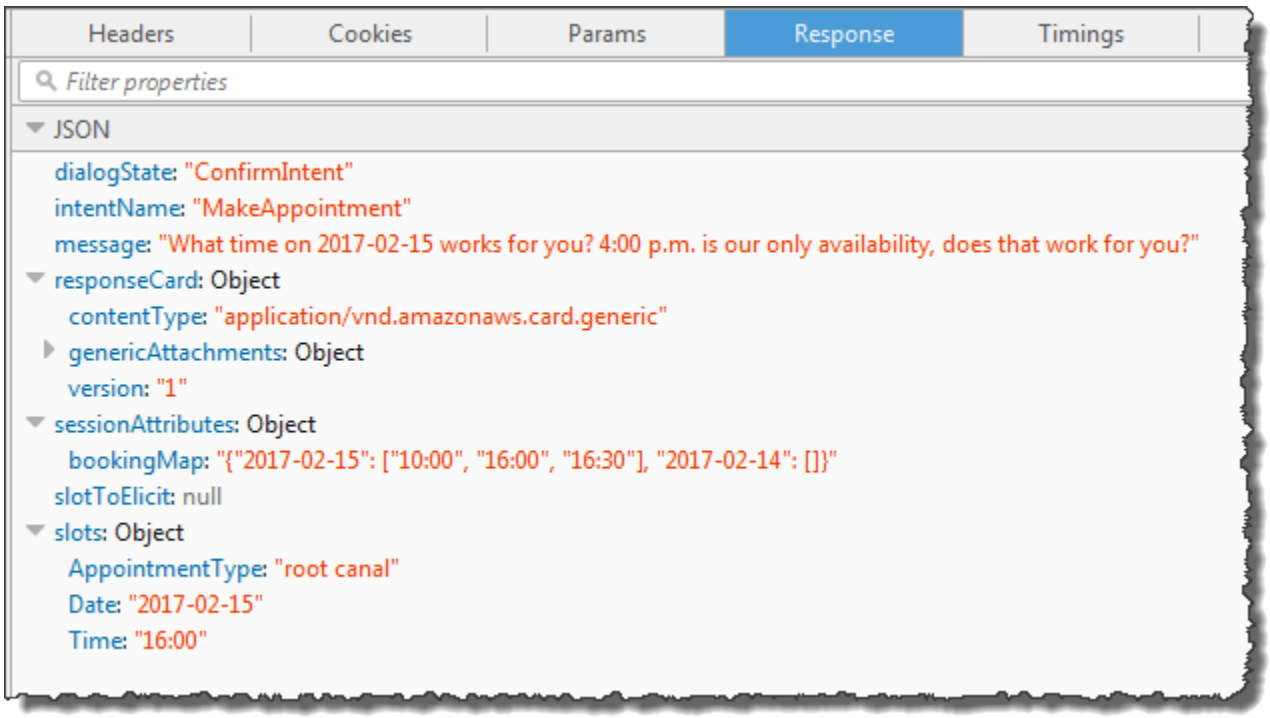
- dialogAction 欄位：

- `type` – Lambda函數會將此值設定為 `ConfirmIntent`，指示 Amazon Lex 取得使用者在 中建議的預約時間確認`message`。
- `responseCard` – 傳回使用者可選擇的是/否值清單。如果用戶端支援回應卡，它會顯示回應卡，如下例所示：



- `sessionAttributes` - Lambda 函數會將`bookingMap`工作階段屬性的值設定為該日期的預約日期和可用的預約。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`中的 所示，Amazon Lex 會將下列回應傳回給用戶端：



用戶端會顯示訊息：2017-02-15 的什麼時間適用於您？下午 4：00 是我們唯一的可用性，這是否適用於您？

5. 使用者：選擇 **yes**。

Amazon Lex 會使用下列事件資料叫用 Lambda 函數。由於使用者已回應 **yes**，Amazon Lex confirmationStatus會將 設定為 Confirmed，並將 中的 Time 欄位currentIntent.slots 設定為 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 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 不預期使用者回應。
- `dialogAction.fulfillmentState` 設為 `Fulfilled`，指出意圖已順利達到。

用戶端會顯示訊息：好的，我已預約您的預約。我們將在 2017-02-15 下午 4：00 與您相見。

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，您可以在 Amazon Lex 主控台中使用測試用戶端來測試機器人 (BookTrip)。用戶端會使用PostText執行時間 API 操作，針對每個使用者輸入將請求傳送至 Amazon Lex。

此範例中的 BookTrip 機器人已設定兩個意圖 (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。利用使用者在先前 BookHotel 意圖中所提供資訊 (也就是目的地城市，以及入住和退房日期)、您設定用來初始化和驗證 BookCar 意圖的程式碼掛勾 (Lambda 函數)、初始化 BookCar 意圖的槽資料 (也就是目的地、取車城市、取車日期和還車日期)。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 函數之間傳遞這些值。Amazon Lex 不會維護或修改任何工作階段屬性。

- currentReservation – 包含進行中保留和其他相關資訊的槽資料。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，如果使用者預訂飯店，然後正在預訂租車、此工作階段屬性便會儲存之前 BookHotel 意圖的槽資料。

- confirmationContext – 當 Lambda 函數根據先前保留的槽資料（如果有的話）預先填入一些槽資料AutoPopulate時，Lambda 函數會將此值設定為 。如此可跨意圖資訊共享。例如，如果使用者先前預訂了飯店，但現在想要預訂租車，Amazon Lex 可以提示使用者確認（或拒絕）正在預訂與其飯店預訂相同的城市和日期的租車

在本練習中，您會使用藍圖來建立 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 Lambda 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) – 這已預先設定如下：
  - 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」

    如果使用者說出其中任何內容，Amazon Lex 會判斷 BookHotel是意圖，然後提示使用者輸入槽資料。
  - 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？」
    - 確認陳述式 –「好的，我允許您從 {CheckInDate} 開始在 {Location} 停留 {Nights} 晚。要我預訂嗎？」
    - 拒絕 –「好的，我已取消您目前的預訂」。

- 意圖 2 (BookCar) – 這已預先設定如下：
  - 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」

    如果使用者說出其中任何內容，Amazon Lex 會判斷 BookCar 是意圖，然後提示使用者輸入槽資料。
  - 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？」
    - CarType 槽提示 –「您想要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」
    - 確認陳述式 –「好的，我允許您在 {PickUpCity} 中從 {PickUpDate} 到 {ReturnDate} 租用 {CarType}。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， AWS Lambda 還提供了藍圖 (lex-book-trip-python)，您可以用它做為機器人藍圖的程式碼掛勾。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 Lambda 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勾，以初始化/驗證使用者資料輸入和實現意圖，藉此更新兩個意圖的組態 (BookCar 和 BookHotel)。

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。

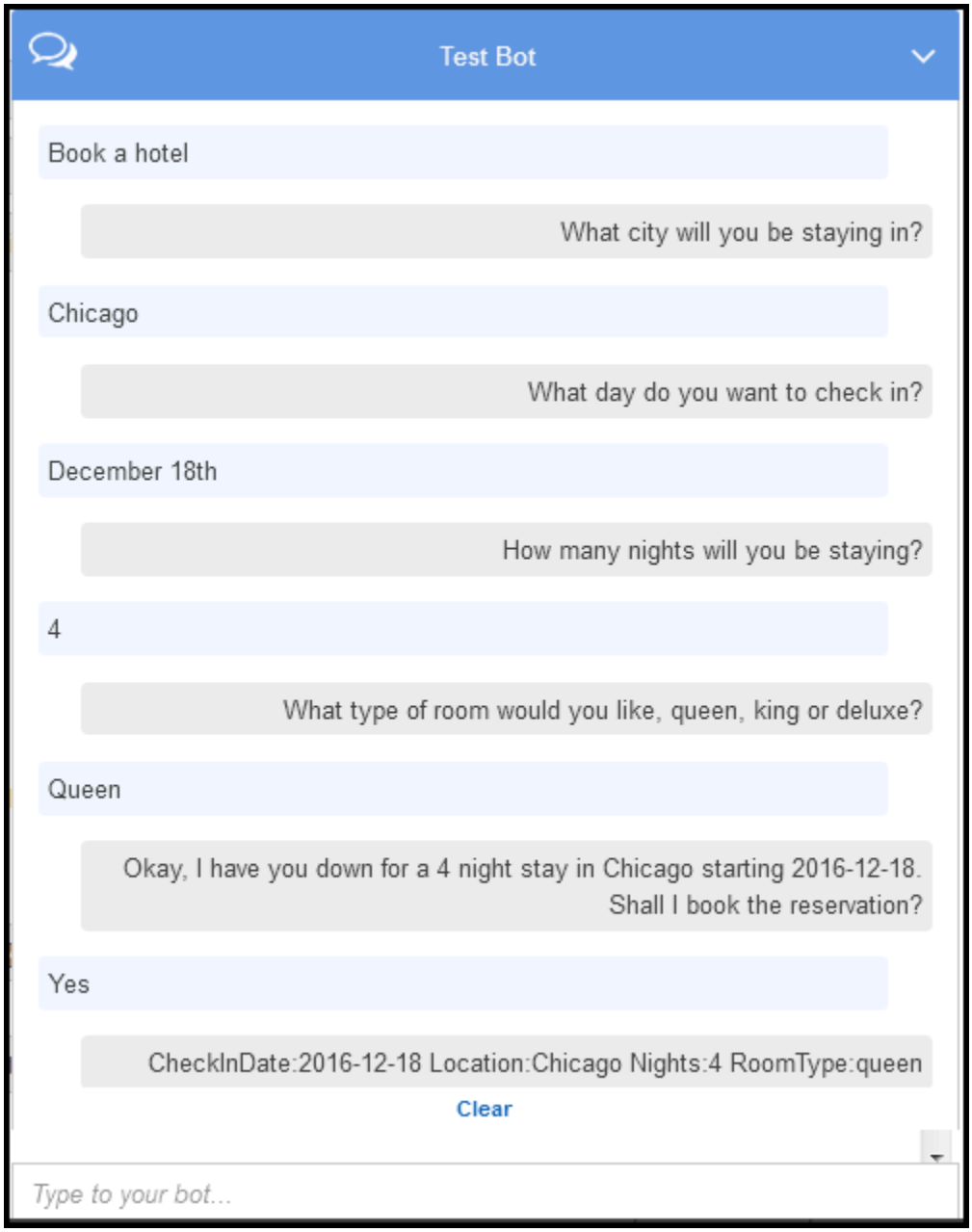
後續步驟

步驟 2：建立 Amazon Lex 機器人

步驟 2：建立 Amazon Lex 機器人

在本節中，您會建立 Amazon Lex 機器人 (BookTrip)。

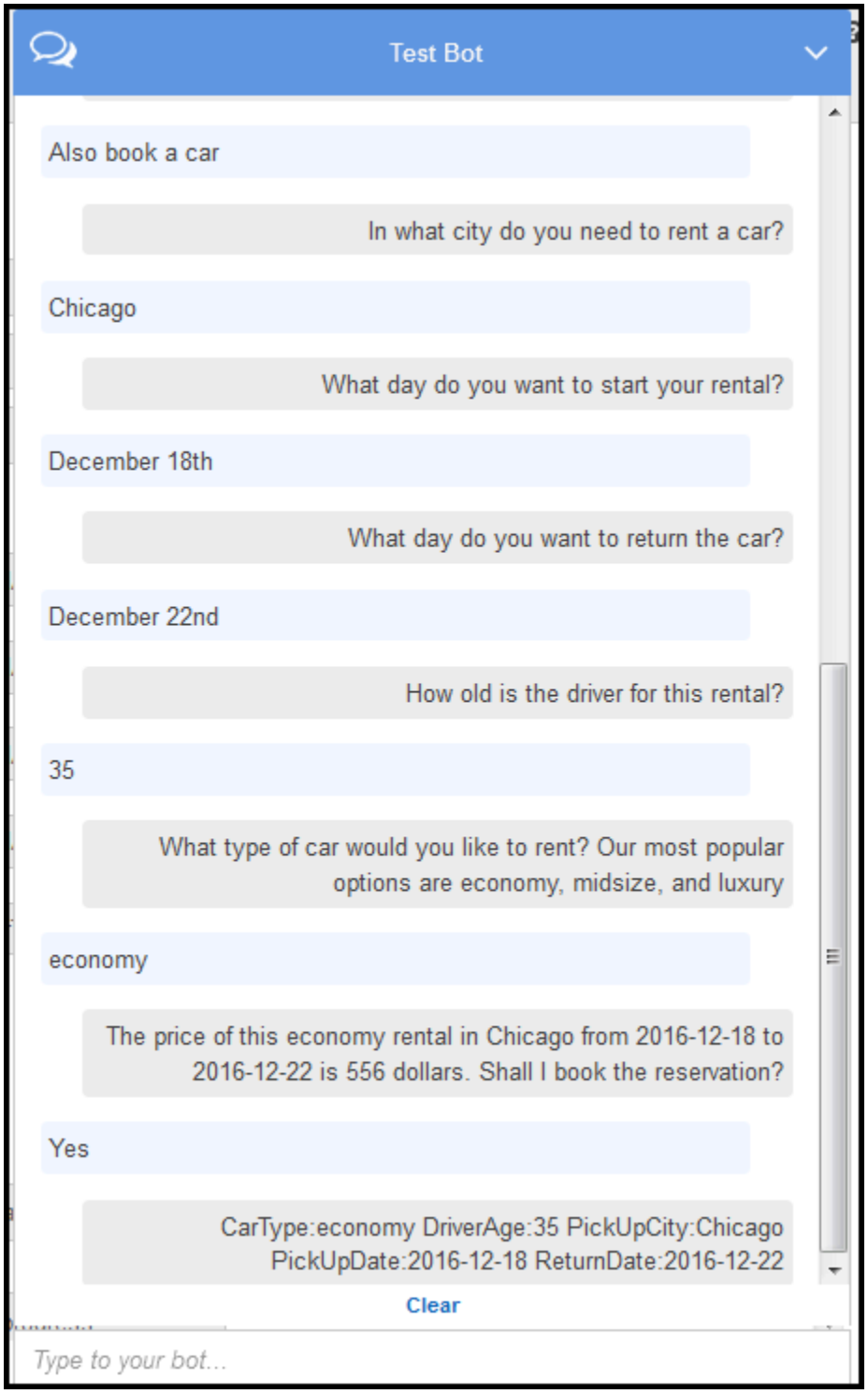
1. 登入 AWS 管理主控台 ，並在 https：//https://console.aws.amazon.com/lex/ 開啟 Amazon Lex 主控台。
2. 在 Bots (機器人) 頁面，選擇 Create (建立)。
3. 在 Create your Lex bot (建立您的機器人) 頁面，
   - 選擇 BookTrip 藍圖。
   - 保留預設機器人名稱 (BookTrip)。
4. 選擇建立。主控台會將一系列請求傳送至 Amazon Lex 以建立機器人。注意下列事項：
5. 主控台會顯示 BookTrip 機器人。在 Editor (編輯器) 索引標籤上，檢閱預先設定之意圖的詳細資訊 (BookHotel 和 BookCar)。
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：



從初始使用者輸入 (「預訂飯店」)，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。使用者提供所有槽資料後，Amazon Lex 會傳回回應給用戶端，其中包含所有使用者輸入做為訊息的訊息。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
```

現在您繼續對話，並嘗試在以下對話中預訂租車。



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 函數

步驟 3：建立 Lambda 函數

在本節中，您會使用 AWS Lambda 主控台中提供的藍圖 (lex-book-trip-python) 來建立 Lambda 函數。您也可以使用主控台提供的範例事件資料來叫用 Lambda 函數，藉此測試 Lambda 函數。

此 Lambda 函數是以 Python 撰寫。

1. 登入 AWS 管理主控台 ，並在 https：//https://console.aws.amazon.com/lambda/ 開啟 AWS Lambda 主控台。
2. 選擇 建立函數 。
3. 選擇 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. 選擇建立函數。
6. 如果您使用的地區設定不是英文 (US) (en-US)，請更新意圖名稱，如中所述更新特定地區設定的藍圖。
7. 測試 Lambda 函數。您調用 Lambda 函數兩次，同時使用範例資料來預訂租車和預訂飯店。
   a. 從選取測試事件下拉式清單中選擇設定測試事件。
   b. 從範例事件範本清單中選擇 Amazon Lex Book Hotel。

此範例事件符合 Amazon Lex 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. 驗證 Lambda 函數是否已成功執行。在此情況下，回應符合 Amazon Lex 回應模型。

e. 重複步驟。這次從範例事件範本清單中選擇 Amazon Lex Book Car。Lambda 函數會處理租車保留。

後續步驟

步驟 4：將 Lambda 函數新增為程式碼掛鉤

步驟 4：將 Lambda 函數新增為程式碼掛鉤

在本節中，您將 Lambda 函數新增為初始化/驗證和履行活動的程式碼掛勾，以更新 BookCar 和 BookHotel 意圖的組態。請務必選擇意圖的 $LATEST 版本，因為您只能更新 Amazon Lex 資源的 $LATEST 版本。

1. 在 Amazon Lex 主控台中，選擇 BookTrip 機器人。
2. 在 Editor (編輯器) 索引標籤中，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. 新增 Lambda 函數做為初始化和驗證程式碼掛勾，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)。
      - 從清單中選擇您的 Lambda 函數。

   c. 新增 Lambda 函數做為履行程式碼掛勾，如下所示：

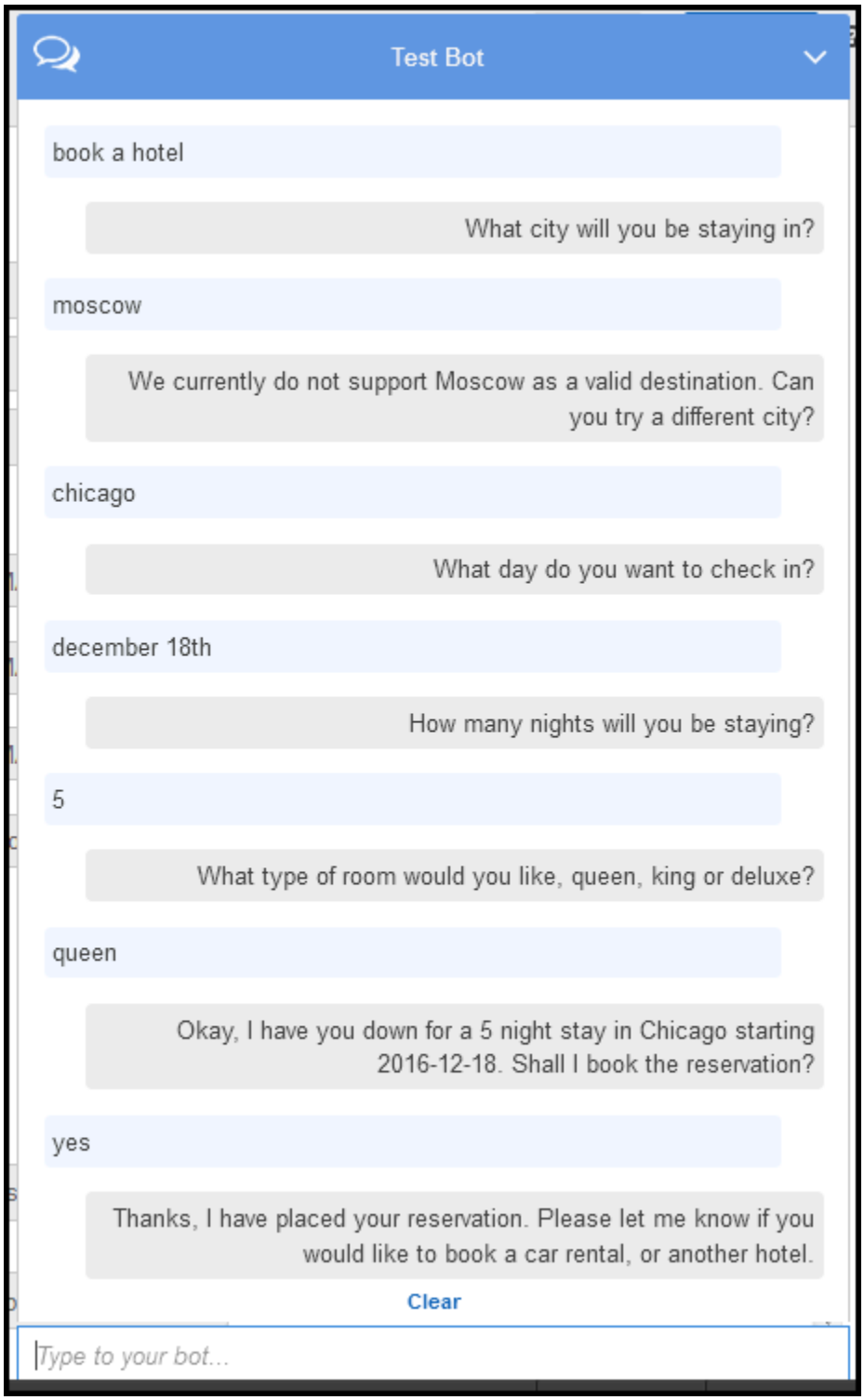
      - 在 Fulfillment (履行)，選擇 AWS Lambda function (AWS Lambda 函數)。
      - 從清單中選擇您的 Lambda 函數。
      - 選擇 Goodbye message (再見訊息) 並輸入訊息。

   d. 選擇儲存。

3. 在 Editor (編輯器) 索引標籤中，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。

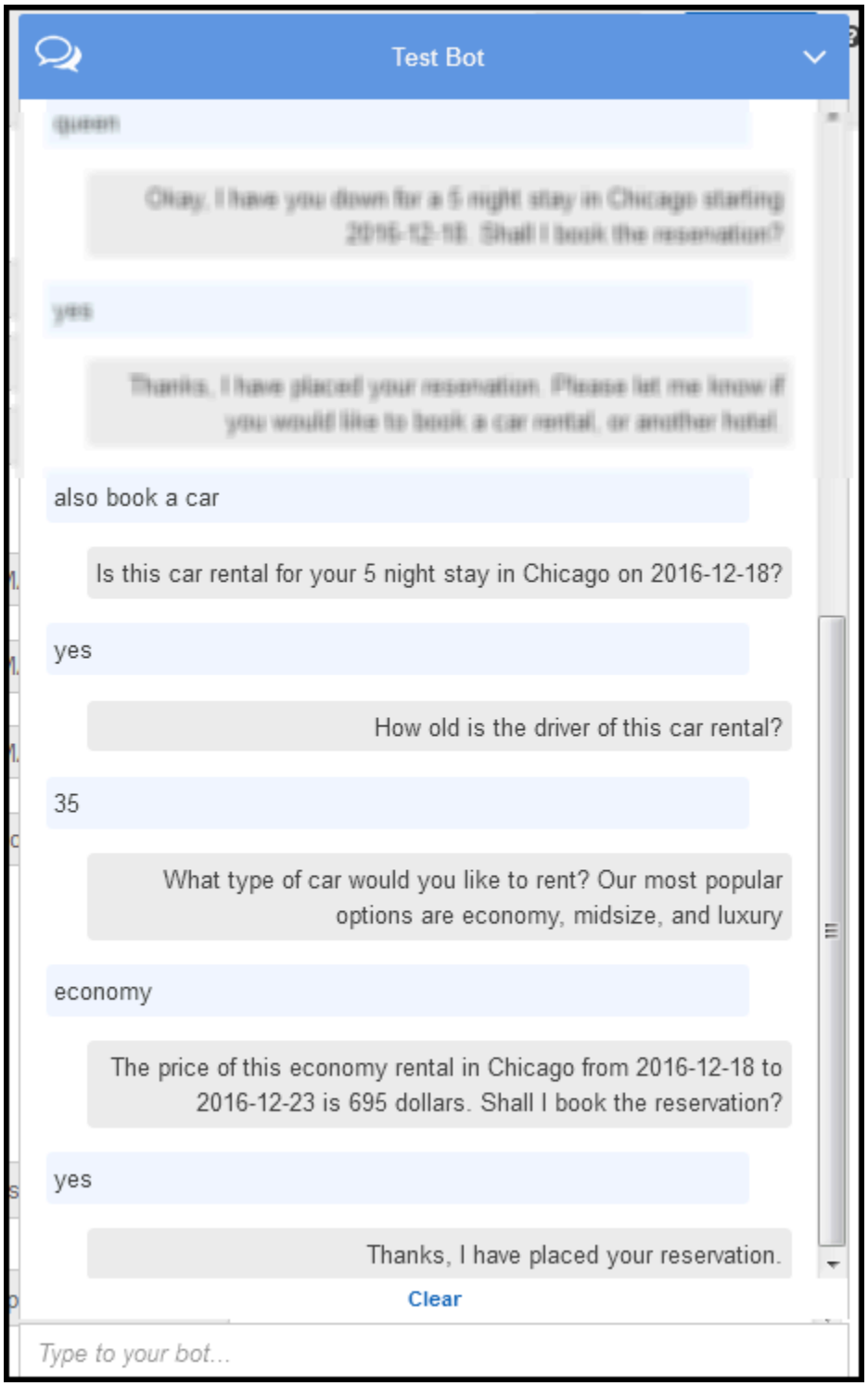
4. 選擇 Build (建置)。主控台會將一系列請求傳送至 Amazon Lex 以儲存組態。

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，您可以在以下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，以及從 Amazon Lex 到 Lambda 函數之資料流程的詳細資訊，請參閱 資料流程：預訂飯店意圖。

6. 繼續對話並預訂租車，如下圖所示：



當您選擇預訂租車時，用戶端（主控台）會將包含工作階段屬性（來自先前的對話 BookHotel) 的請求傳送至 Amazon Lex。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，以及從 Amazon Lex 到 Lambda 函數之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 Lex 主控台中提供的測試時段用戶端與 Amazon Lex BookTrip 機器人進行對話。本節說明下列各項：

- 用戶端與 Amazon Lex 之間的資料流程。

  本節假設用戶端使用`PostText`執行時間 API 將請求傳送至 Amazon Lex，並相應地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**
  >
  > 如需用戶端與用戶端使用 `PostContent` API 的 Amazon Lex 之間資訊流程的範例，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 函數之間的資料流程。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 和內文都會提供資訊給 Amazon Lex：

   - 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`。一開始，這是一個空的物件，Lambda 函數會先設定工作階段屬性。

   b. 從 `inputText`，Amazon Lex 會偵測意圖 (BookHotel)。此意圖是以 Lambda 函數設定為使用者資料初始化/驗證的程式碼掛勾。因此，Amazon Lex 會透過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：

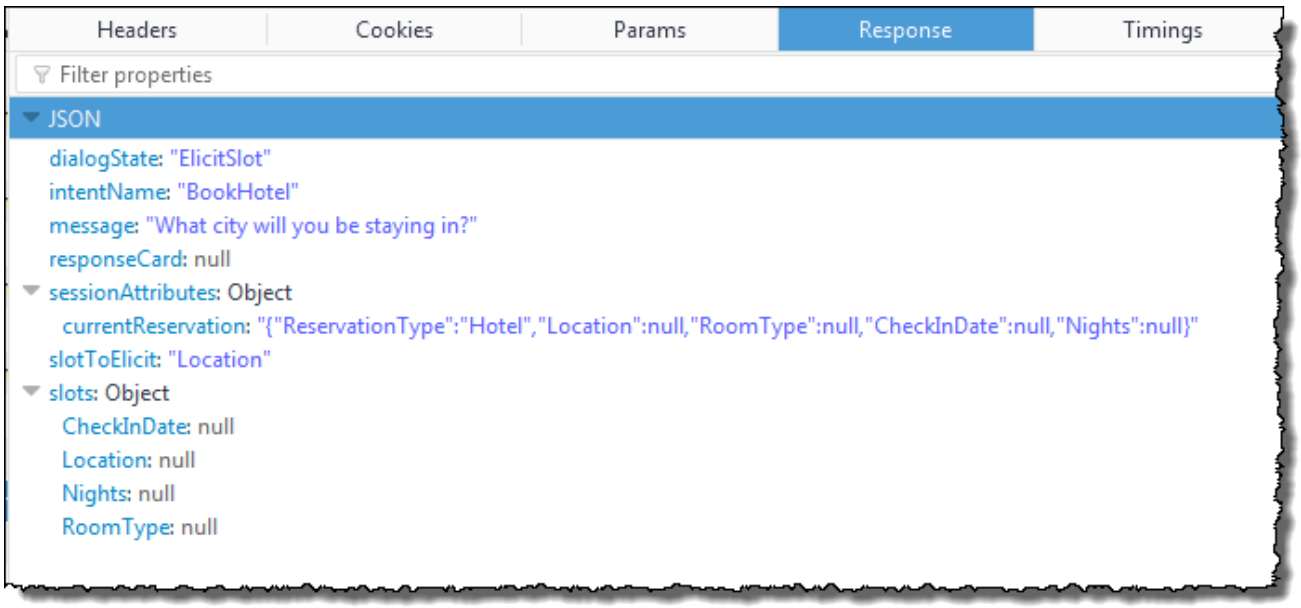
- `messageVersion` – 目前 Amazon Lex 僅支援 1.0 版本。
- `invocationSource` – 指出 Lambda 函數調用的目的。在這種情況下，它是執行使用者資料初始化和驗證（此時 Amazon Lex 知道使用者尚未提供所有槽資料以滿足意圖）。
- `currentIntent` – 所有槽值設定為 null。

c. 此時，所有槽值都是 null。Lambda 函數不需要驗證。Lambda 函數會傳回下列回應給 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 只會傳遞這些值。
> - `dialogAction.type` – 透過將此值設定為 `Delegate`，Lambda 函數會將後續動作的責任委派給 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 接下來要做什麼。

d. 根據 `dialogAction.type`，Amazon Lex 會決定下一個動作過程，從使用者取得`Location`槽的資料。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`請求傳送至 Amazon Lex（為便於閱讀而新增換行）：

```
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 會先在目前意圖的內容inputText中解譯（服務會記住已向特定使用者詢問Location槽的相關資訊）。它會更新目前意圖的槽值，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 只是將工作階段屬性傳遞至 Lambda 函數。
> - 對於 `currentIntent.slots`，Amazon Lex 已將`Location`插槽更新為 `Moscow`。

c. Lambda 函數會執行使用者資料驗證，並判斷 `Moscow` 是無效的位置。

> **Note**
>
> 本練習中的 Lambda 函數具有有效城市的簡單清單，`Moscow`且不在清單中。在生產應用程式中，您可以使用後端資料庫來取得此資訊。

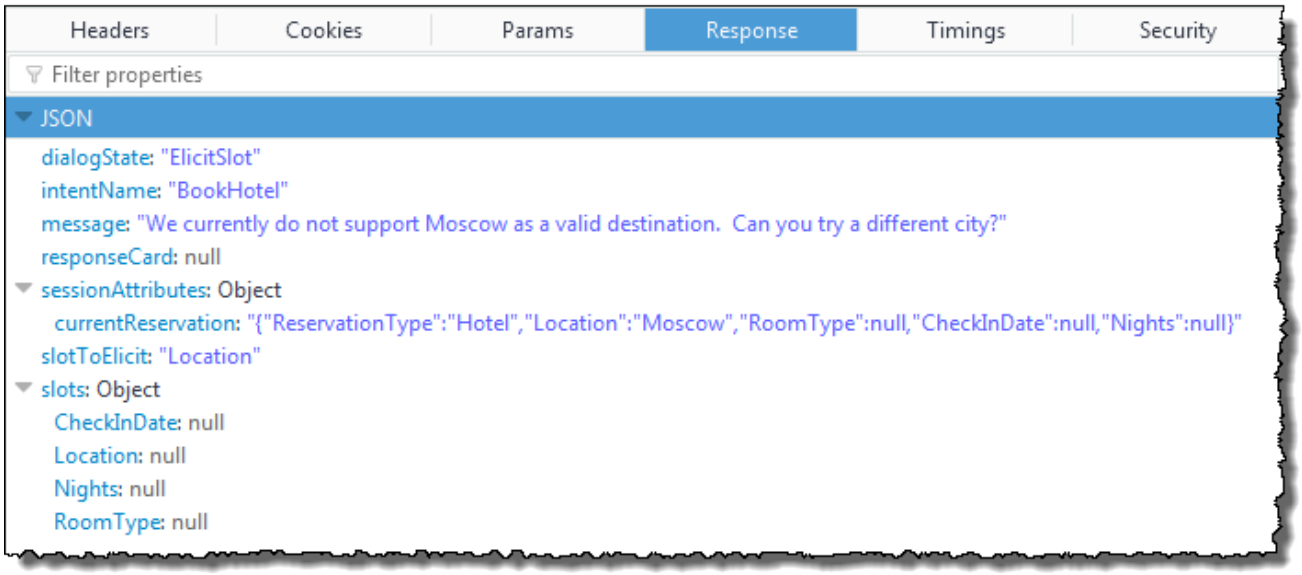
它會將槽值重設回 null，並指示 Amazon Lex 透過傳送下列回應，再次提示使用者輸入另一個值：

```
{
    "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，這會指示 Amazon Lex 透過提供下列項目再次提示使用者：
>   - 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 注意到 ，dialogAction.type並在下列回應中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請求傳送至 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 知道內容，它正在引出Location槽的資料。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"
```

```
    }
}
```

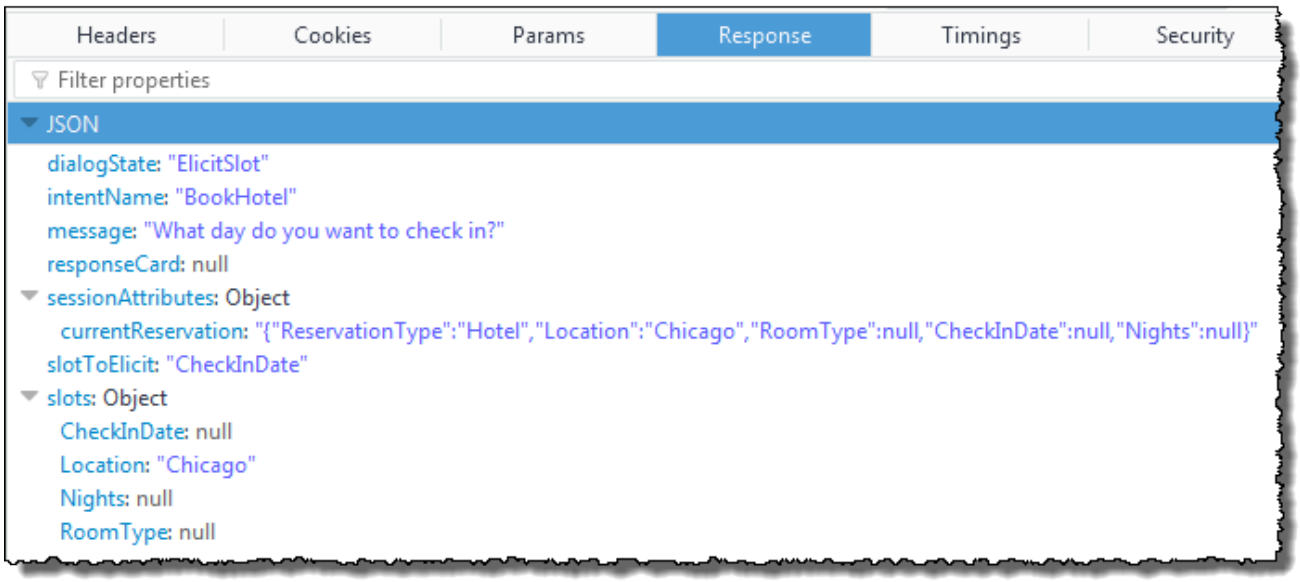
Amazon Lex 透過將Location插槽設定為 currentIntent.slots來更新 Chicago。

c. 根據 invocationSource的值DialogCodeHook，Lambda 函數會執行使用者資料驗證。它會辨識Chicago為有效的槽值，相應地更新工作階段屬性，然後將以下回應傳回給 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 函數會將 設定為 Location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，Lambda 函數會指示 Amazon Lex 選擇下一個動作。

d. 根據 dialogAction.type，Amazon Lex 選擇下一個動作。Amazon Lex 知道它需要更多槽資料，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：

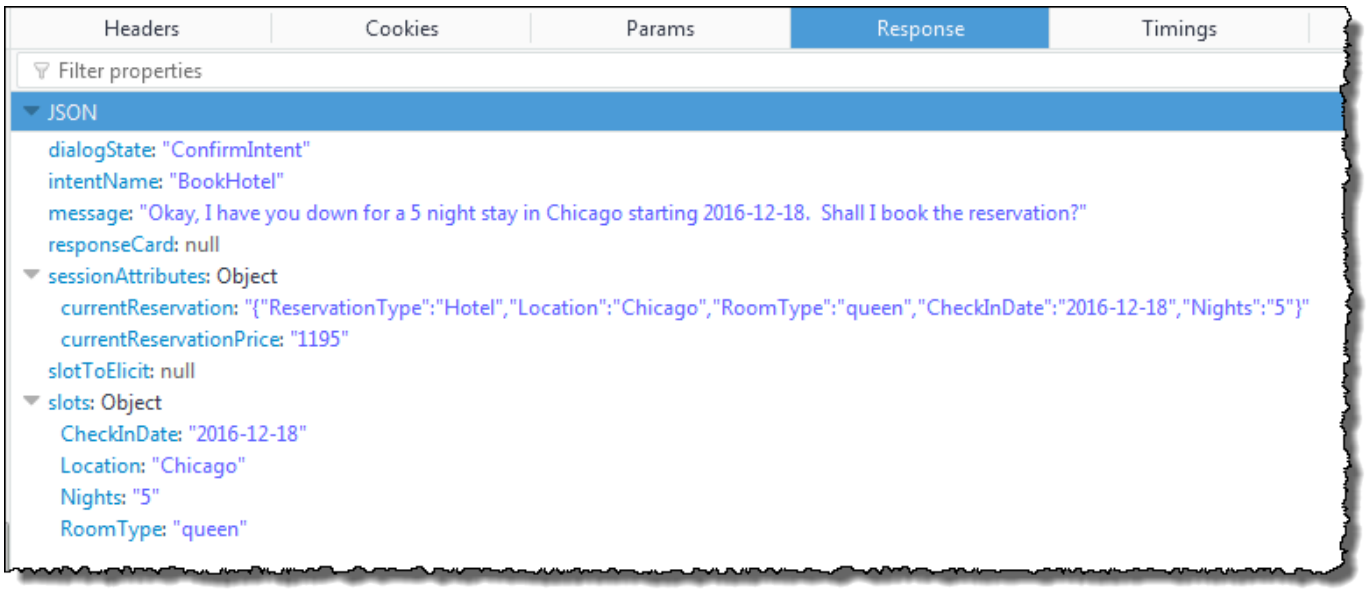


用戶端顯示訊息：「您要在哪一天入住？」

4. 使用者互動會繼續 – 使用者提供資料，Lambda 函數會驗證資料，然後將下一個動作過程委派給 Amazon Lex。最後，使用者提供所有槽資料，Lambda 函數會驗證所有使用者輸入，然後 Amazon Lex 會辨識其具有所有槽資料。

> **Note**
>
> 在本練習中，使用者提供所有槽資料後，Lambda 函數會計算飯店保留的價格，並將其傳回為另一個工作階段屬性 (`currentReservationPrice`)。

此時，意圖已準備好履行，但 BookHotel 意圖已設定為需要使用者確認才能 Amazon Lex 履行意圖的確認提示。因此，Amazon Lex 會在預訂飯店之前，將下列訊息傳送給請求確認的用戶端：



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請求傳送至 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 inputText會在確認目前意圖的情況下解譯 。Amazon Lex 了解使用者想要繼續保留。這次，Amazon Lex 會呼叫 Lambda 函數，透過傳送下列事件來實現意圖。透過在事件FulfillmentCodeHook中invocationSource將 設定為 ，它會傳送至 Lambda 函數。Amazon Lex 也會將 confirmationStatus設定為 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 將此值設定為 `FulfillmentCodeHook`，指示 Lambda 函數實現意圖。
> - `confirmationStatus` – 已設定為 `Confirmed`。

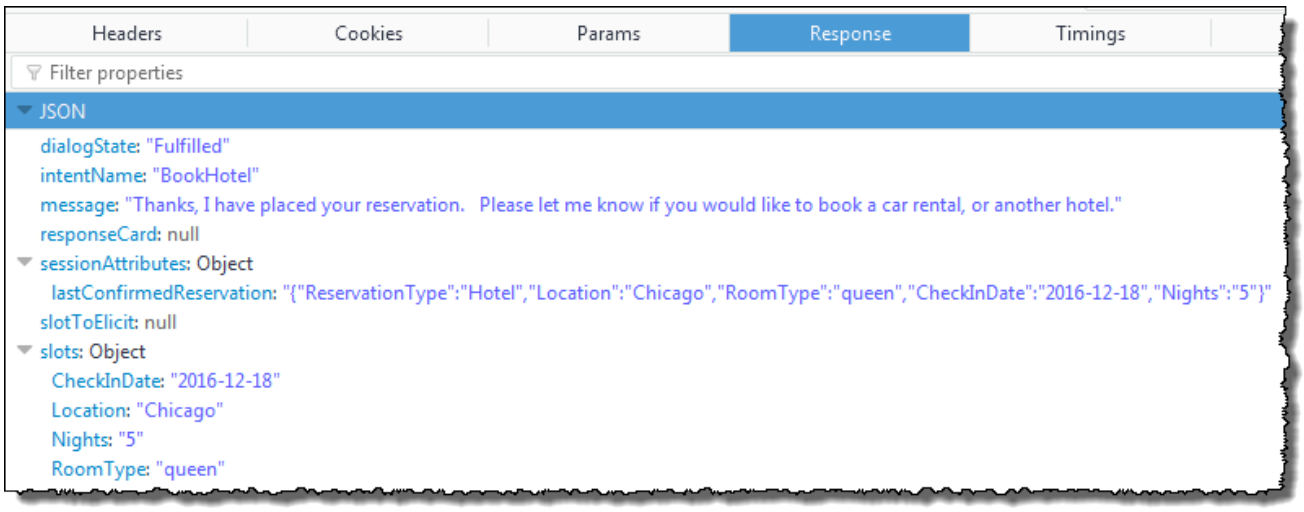
c. 這次，Lambda 函數滿足 BookHotel 意圖，Amazon Lex 完成保留，然後傳回下列回應：

```
{
    "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（而非 `currentReservation`、`currentReservationPrice`）。
> - `dialogAction.type` – Lambda 函數將此值設定為 `Close`，表示 Amazon Lex 不會預期使用者回應。
> - `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 將此值設定為 `Fulfilled`。
> - `message` – 與 Lambda 函數提供的訊息相同。

用戶端將顯示該訊息。

資料流程：預訂租車意圖

此練習中的 BookTrip 機器人支援兩個意圖 (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。

具體而言，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 意圖的槽 (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. 用戶端會將下列`PostText`請求傳送至 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 偵測意圖 (BookCar)`inputText`。此意圖也會設定為叫用 Lambda 函數，以執行使用者資料的初始化和驗證。Amazon Lex 使用下列事件叫用 Lambda 函數：

```
{
    "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 僅支援 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` – 它包含意圖名稱和槽。此時，所有插槽值都是 null。

c. Lambda 函數會注意到所有 null 槽值，無需驗證。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?"
        }
    }
}
```

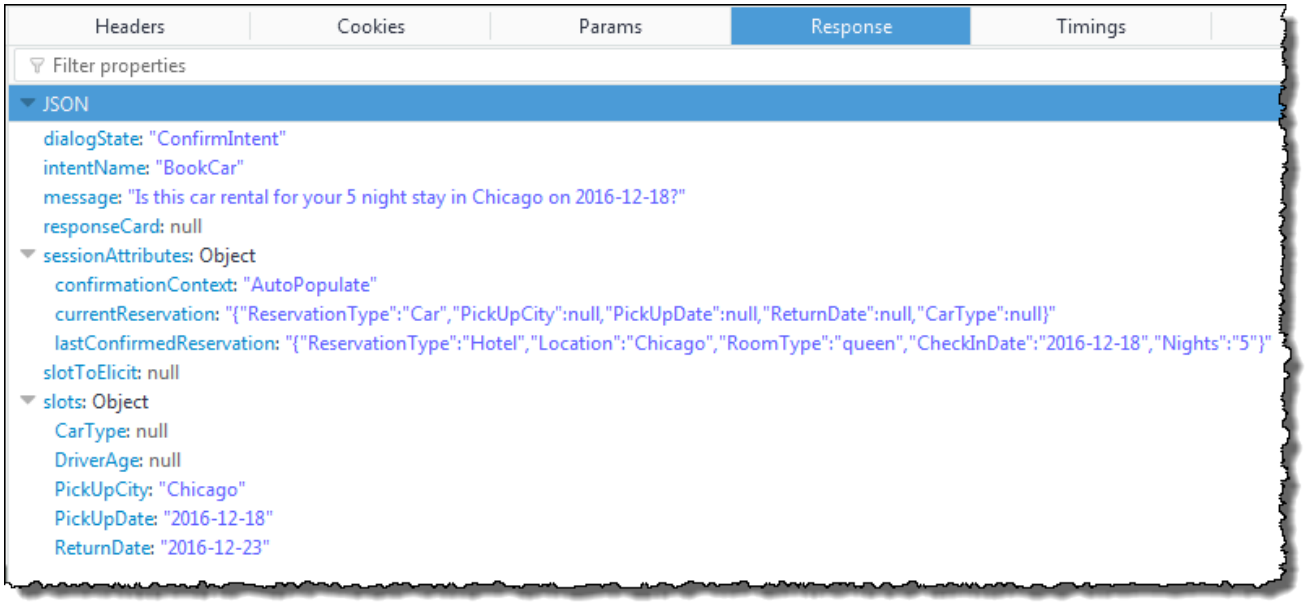
> **Note**
>
> - 除了 之外`lastConfirmedReservation`，Lambda 函數還包含更多工作階段屬性 (`currentReservation` 和 `confirmationContext`)。

- `dialogAction.type` 設定為 `ConfirmIntent`，這會通知 Amazon Lex 使用者不需要回覆 (confirmationContext 設定為 AutoPopulate，Lambda 函數知道是/否使用者回覆是取得執行的 Lambda 函數初始化 （ 自動填入槽資料 ） 的使用者確認。

  Lambda 函數也在回應中包含 中的資訊性訊息，`dialogAction.message`以便 Amazon Lex 傳回用戶端。

  > **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 函數會使用此術語指示 Amazon Lex 取得使用者的是/否回覆。

d. 根據 `dialogAction.type`，Amazon Lex 會傳回下列回應給用戶端：



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. 用戶端會將下列`PostText`請求傳送至 Amazon Lex。

```
POST /bot/BookTrip/alias/$LATEST/user/wch89kjqcpkds8seny7dly5x3otq68j3/text
"Content-Type":"application/json"
```

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

{
   "inputText":"yes",
   "sessionAttributes":{
      "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 會讀取 inputText並知道內容（要求使用者確認自動人口）。Amazon Lex 會傳送下列事件來叫用 Lambda 函數：

```
{
    "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"
    }
}
```

由於使用者回覆是，Amazon Lex 會將 confirmationStatus設定為 Confirmed。

c. 從 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：

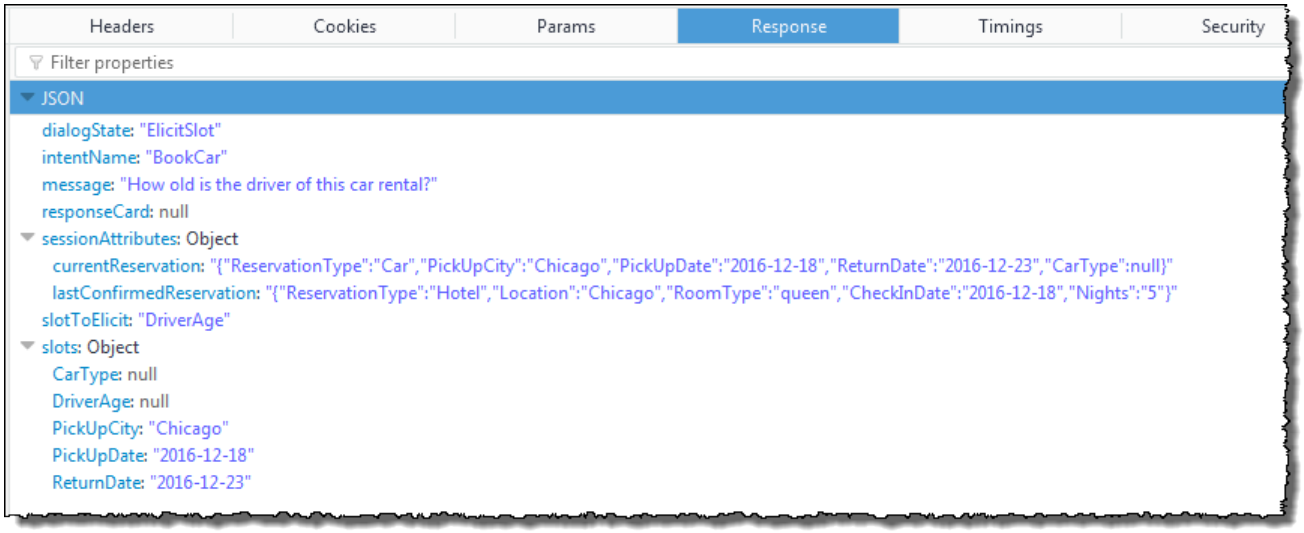
- 以它預先填入的槽值更新 currentReservation 工作階段屬性。
- 將 dialogAction.type 設定為 ElicitSlot
- 將 slotToElicit 值設定為 DriverAge。

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 傳回下列回應：



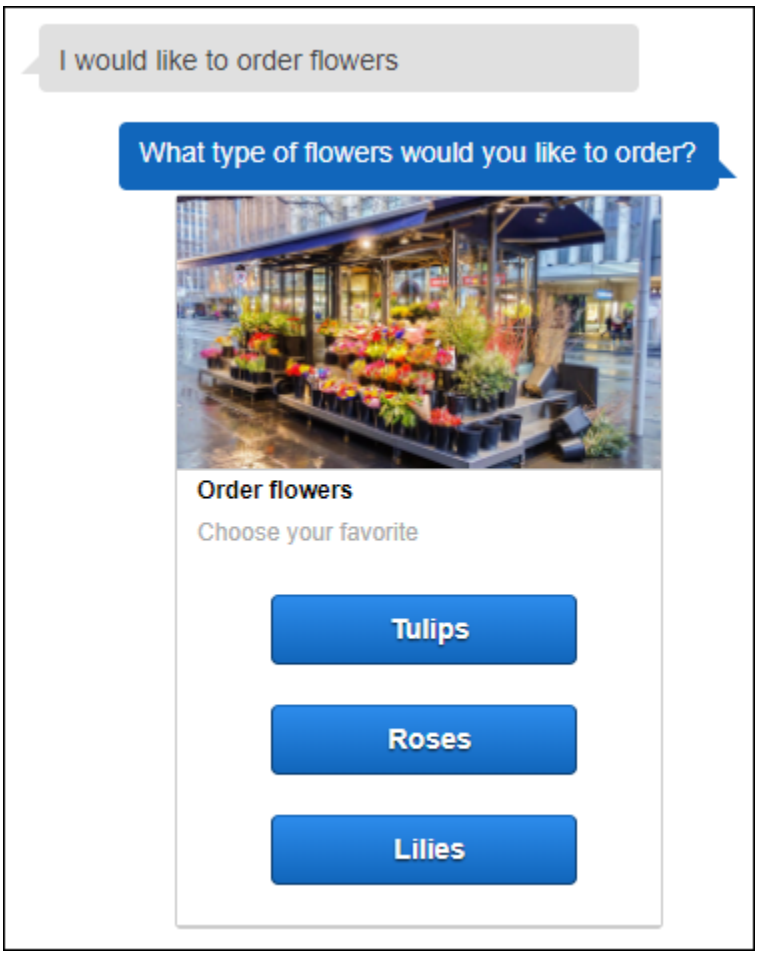
用戶端會顯示訊息「此租車的駕駛人多大？」 而對談繼續。

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。您要建立一個支援 OrderFlowers 意圖的機器人，然後為 FlowerType 槽加入回應卡以更新該意圖。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?
```

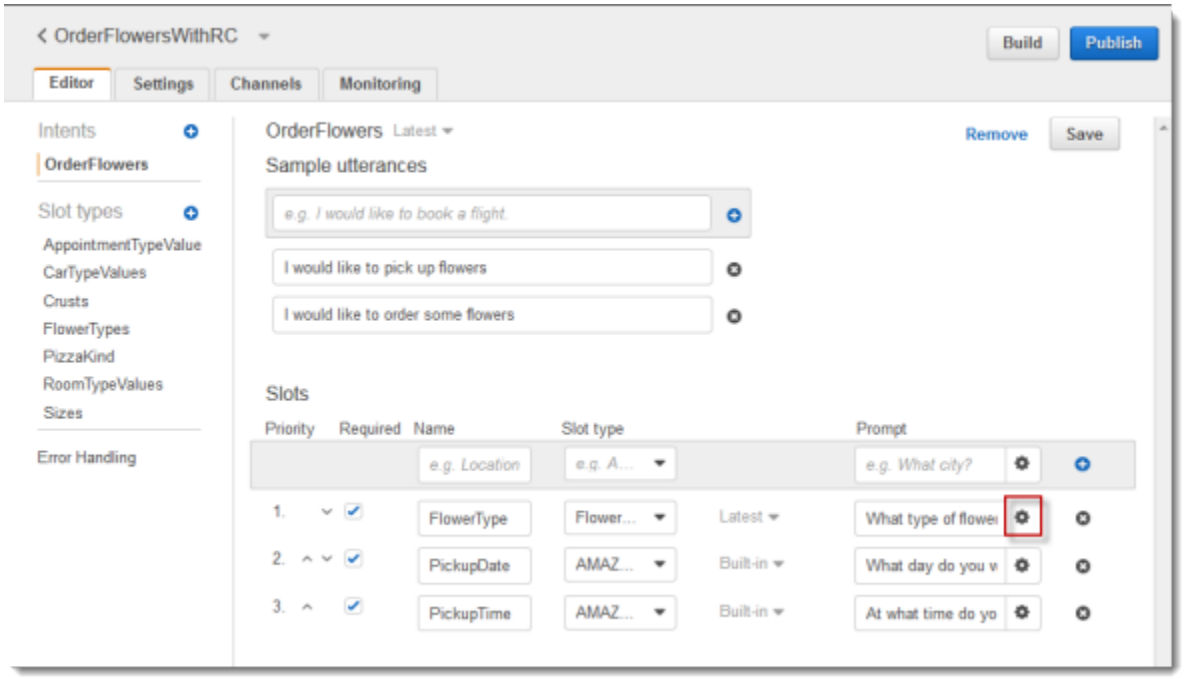
回應卡如下所示：



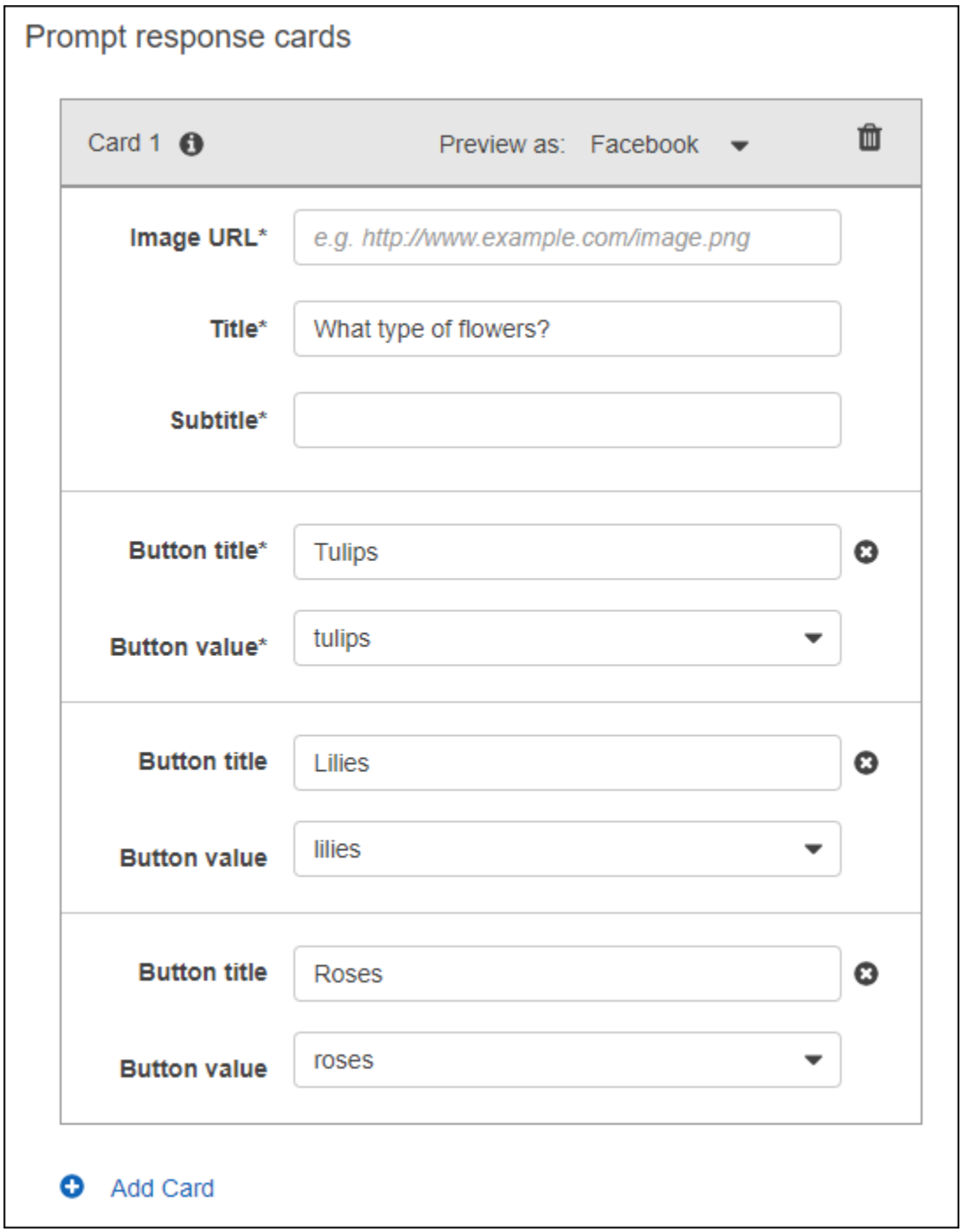
機器人使用者可以輸入文字或從花種清單中選擇。此回應卡有設定圖像，其將如下所示出現在用戶端中。如需回應卡的詳細資訊，請參閱回應卡。

使用回應卡建立及測試機器人：

1. 依照入門練習 1 的指示，建立及測試 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。如需說明，請參閱 練習 1：使用藍圖建立 Amazon Lex 機器人（主控台）。
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「什麼類型的花朵」旁的設定齒輪圖示 提示來設定 的回應卡`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。如需說明，請參閱 將 Amazon Lex 機器人與 Facebook Messenger 整合。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。如需說明，請參閱 將 Amazon Lex 機器人與 Slack 整合。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。如需說明，請參閱 將 Amazon Lex 機器人與 Twilio 可程式設計簡訊整合。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。

更新張量

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用語。為了改善使用者體驗，您應將這類表達用語加入至機器人。

在下列情況中，不會產生差異統計資料：

- 欄位`childDirected`在建立機器人時設定為 true。
- 您使用槽混淆搭配一或多個槽。
- 您選擇不參與改善 Amazon Lex。

> **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。

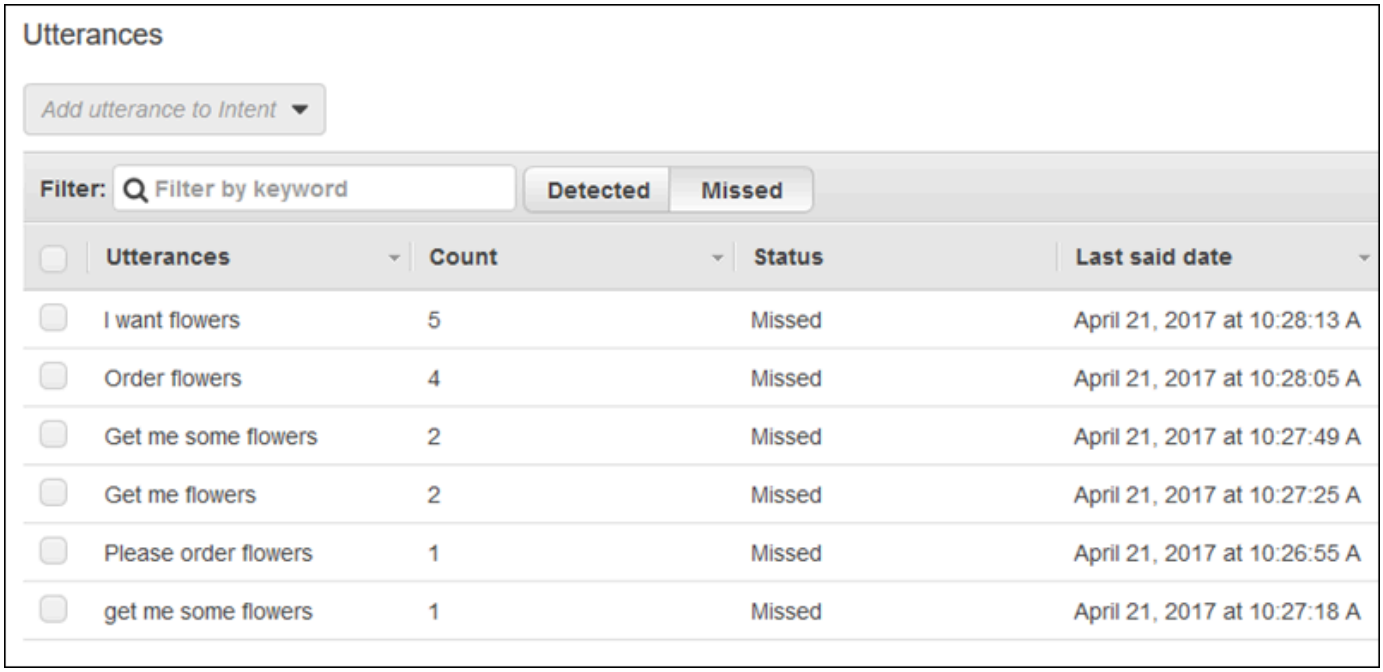
您可以查看不同版本機器人適用的表達用語。若要變更您正在查看表達用語之機器人的版本，請從機器人名稱旁的下拉式清單選擇不同的版本。

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。如需說明，請參閱 練習 1：使用藍圖建立 Amazon Lex 機器人（主控台）。
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 收集有關遺漏用語的使用資料。表達用語資料為每天產生一次，通常在深夜產生。
4. 登入 AWS 管理主控台 ，並在 https://console.aws.amazon.com/lex/ ：// 開啟 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 個遺漏的表達用語。



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，您可以使用 JavaScript AWS 和服務，為網站的訪客建立互動式體驗。您可以自 AWS AI 部落格 選擇已記錄的範例：

- 部署適用於聊天機器人的 Web UI — 示範功能完整的 Web UI，為 Amazon Lex 聊天機器人提供 Web 用戶端。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。

- "Greetings， visitor！"—與 Amazon Lex 建立您的 Web 使用者—示範如何使用 Amazon Lex、瀏覽器中的適用於 JavaScript 的 AWS 開發套件，以及 Amazon Cognito 在您的網站上建立對話體驗。

- 在瀏覽器中擷取語音輸入並將其傳送至 Amazon Lex：示範使用瀏覽器中的適用於 JavaScript 的 SDK，在網站中內嵌以語音為基礎的聊天機器人。應用程式會記錄音訊，將音訊傳送至 Amazon Lex，然後播放回應。

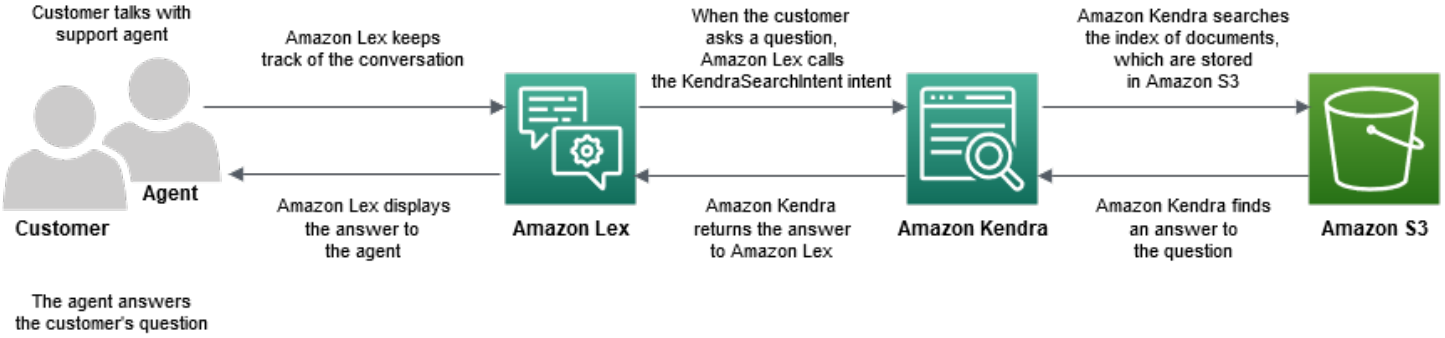
客服中心客服人員助理

在本教學課程中，您會使用 Amazon Lex 搭配 Amazon Kendra 來建置客服人員協助機器人，協助客戶支援客服人員並將其發佈為 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 Amazon Kendra 的詳細資訊，請參閱《Amazon Kendra 開發人員指南》。

Amazon Lex 機器人廣泛用於客服中心，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，會將對話轉接給客戶支援員工。

在本教學課程中，我們會建立客服人員用來即時回應客戶查詢的 Amazon Lex 機器人。透過讀取機器人提供的答案，代理程式無需手動查詢答案。

您在本教學課程中建立的機器人和 Web 應用程式可透過快速提供正確的資源，協助客服人員有效率地準確地回應客戶。下圖顯示 Web 應用程式的運作方式。



如圖所示，文件的 Amazon Kendra 索引存放在 Amazon Simple Storage Service (Amazon S3) 儲存貯體中。如果您還沒有 S3 儲存貯體，您可以在建立 Amazon Kendra 索引時設定一個儲存貯體。除了 Amazon S3 之外，您將在本教學課程中使用 Amazon Cognito。Amazon Cognito 會管理將機器人部署為 Web 應用程式的許可。

在本教學課程中，您會建立 Amazon Kendra 索引，提供客戶問題的答案、建立機器人並新增意圖，讓它根據與客戶的對話來建議答案、設定 Amazon Cognito 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

預估成本：Amazon Kendra 索引為每小時 2.50 美元，1000 個 Amazon Lex 請求為 0.75 美元。完成本練習後，Amazon Kendra 索引會繼續執行。請務必將其刪除，以避免不必要的成本。

注意：請務必為本教學課程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
- 步驟 3：新增自訂和內建意圖
- 步驟 4：設定 Amazon Cognito
- 步驟 5：將您的機器人部署為 Web 應用程式
- 步驟 6：使用 機器人

步驟 1：建立 Amazon Kendra 索引

首先建立回答客戶問題的 Amazon Kendra 文件索引。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 會將索引文件中找到的答案傳回給機器人，並將它們顯示給客服人員。

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、Microsoft Office (.doc、.ppt)、PDF 和文字格式為非結構化和半結構化資料編製索引。

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 （ 主控台 ） 入門。

若要新增有助於回答客戶查詢的問題和答案 (FAQs)，請參閱《Amazon Kendra 開發人員指南》中的新增問題和答案。在本教學課程中，請使用 GitHub 上的 ML_FAQ.csv 檔案。

下一步驟

步驟 2：建立 Amazon Lex 機器人

步驟 2：建立 Amazon Lex 機器人

Amazon Lex 提供呼叫中心代理程式與 Amazon Kendra 索引之間的界面。它會追蹤客服人員與客戶之間的對話，並根據客戶提出的問題呼叫`AMAZON.KendraSearchIntent`意圖。意圖是使用者想要執行的動作。

Amazon Kendra 會搜尋索引文件，並傳回在機器人中顯示的答案給 Amazon Lex。只有代理程式可以看到此答案。

建立客服人員助理機器人

1. 登入 AWS 管理主控台 並開啟位於 https：//https://console.aws.amazon.com/lex/ 的 Amazon Lex 主控台。
2. 在導覽窗格中，選擇 Bots (機器人)。
3. 選擇建立。
4. 選擇自訂機器人並設定機器人。
   a. 機器人名稱 – 輸入指出機器人用途的名稱，例如 **`AgentAssistBot`**。
   b. 輸出語音 – 選擇無。
   c. 工作階段逾時 – 輸入 **5**。

d. COPPA – 選擇否。

5. 選擇建立。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

步驟 3：新增自訂和內建意圖

意圖代表呼叫中心客服人員希望機器人執行的動作。在這種情況下，客服人員希望機器人根據客服人員與客戶的對話來建議回應和資源。

Amazon Lex 有兩種類型的意圖：自訂意圖和內建意圖。 `AMAZON.KendraSearchIntent` 是內建意圖。機器人使用 `AMAZON.KendraSearchIntent` 意圖來查詢索引，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，您必須建立至少一個具有至少一個範例表達用語的自訂意圖。只有建置您的客服人員助理機器人才需要此意圖。它不會執行任何其他 函數。意圖的表達用語不得回答客戶可能提出的任何問題。這可確保`AMAZON.KendraSearchIntent`呼叫 來回答客戶查詢。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. 針對範例表達用語，輸入描述性表達用語，例如 **`Required utterance`**。
5. 選擇儲存意圖。

新增 AMAZON.KendraSearchIntent 意圖和回應訊息

1. 在導覽窗格中，選擇意圖旁的加號 (+)。
2. 選擇搜尋現有的意圖。
3. 在搜尋意圖方塊中，輸入 **`AMAZON.KendraSearchIntent`**，然後從清單中選擇它。
4. 為意圖提供描述性名稱，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. 選擇您希望意圖搜尋的索引，
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 應用程式安全且具有存取控制。Amazon Cognito 使用身分集區來提供 AWS 憑證，以授予使用者存取其他 AWS 服務的權限。在本教學課程中，它提供 Amazon Lex 的存取權。

建立身分集區時，Amazon Cognito 會為已驗證和未驗證的使用者提供 AWS Identity and Access Management (IAM) 角色。您可以透過新增授予 Amazon Lex 存取權的政策來修改 IAM 角色。

設定 Amazon Cognito

1. 登入 AWS 管理主控台 並開啟位於 https：//https://console.aws.amazon.com/cognito/ 的 Amazon Cognito 主控台。
2. 選擇 Manage Identity Pools (管理身分集區)。
3. 選擇 Create new identity pool (建立新的身分池)。
4. 設定身分集區。
   a. 身分集區名稱 – 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. 在識別要與新身分集區搭配使用的 IAM 角色頁面上，選擇檢視詳細資訊。
7. 記錄 IAM 角色名稱。您稍後會修改它們。
8. 選擇 Allow (允許)。
9. 在 Amazon Cognito 入門頁面上，針對平台選擇 JavaScript。
10. 在取得 AWS 登入資料區段中，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 管理主控台 並開啟位於 https：//https://console.aws.amazon.com/iam/ 的 IAM 主控台。
    b. 在導覽窗格的存取管理下，選擇角色。
    c. 在搜尋方塊中，輸入已驗證 IAM 角色的名稱，然後選擇其旁邊的核取方塊。

        i. 選擇連接政策。
        ii. 在搜尋方塊中，輸入 **AmazonLexRunBotsOnly** 並選擇其旁邊的核取方塊。
        iii. 選擇連接政策。
    d. 在搜尋方塊中輸入未驗證的 IAM 角色名稱，然後選擇其旁邊的核取方塊。

        i. 選擇連接政策。
        ii. 在搜尋方塊中，輸入 **AmazonLexRunBotsOnly** 並選擇其旁邊的核取方塊。
        iii. 選擇連接政策。

下一步驟

步驟 5：將您的機器人部署為 Web 應用程式

步驟 5：將您的機器人部署為 Web 應用程式

將機器人部署為 Web 應用程式

1. 將位於 https：//https://github.com/awsdocs/amazon-lex-developer-guide/blob/master/example_apps/agent_assistance_bot/ 的儲存庫下載至您的電腦。
2. 導覽至下載的儲存庫，並在編輯器中開啟 index.html 檔案。
3. 進行下列變更：

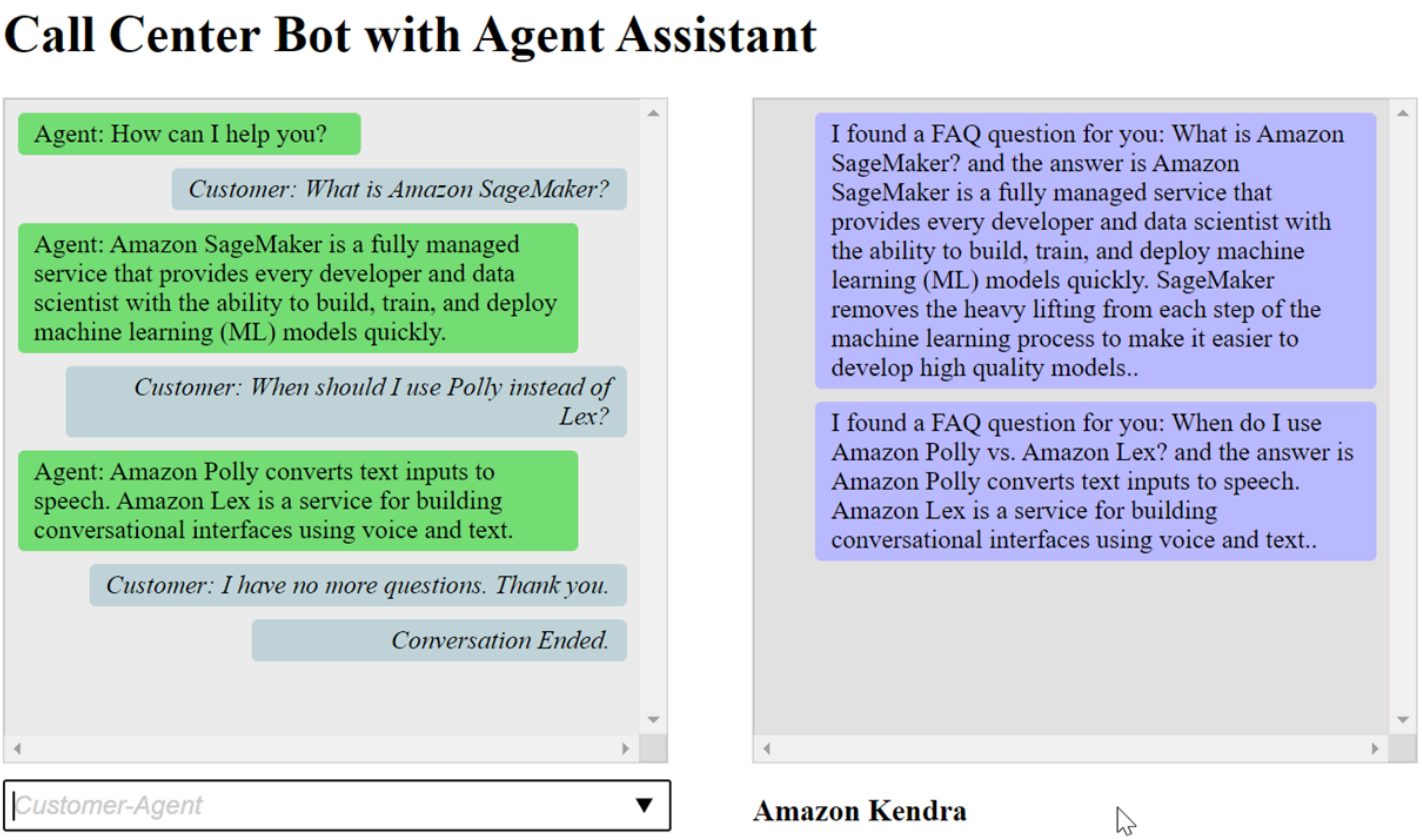
    a. 在 `AWS.config.credentials`區段中，輸入您的區域名稱和身分集區 ID。
    b. 在 `Amazon Lex runtime parameters`區段中，輸入機器人名稱。

c. 儲存檔案。

步驟 6：使用 機器人

基於示範目的，您會以客戶和客服人員的身分向機器人提供輸入。為了區分兩者，客戶提出的問題以「客戶：」開頭，客服人員提供的答案以「客服人員：」開頭。您可以從建議的輸入功能表中進行選擇。

開啟 來執行您的 Web 應用程式`index.html`，與您的機器人進行類似下列影像的對話：



index.html 檔案中的 `pushChat()`函數說明如下。

```
            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 執行時間 API 會將其傳送至 Amazon Lex。

`showMessage(daText, senderRequest, displayWindow)` 函數會在聊天視窗中顯示客服人員與客戶之間的對話。Amazon Kendra 建議的回應會顯示在相鄰視窗中。當客戶說 時，對話就會結束 **“I have no more questions. Thank you.”**

注意：不使用時，請刪除您的 Amazon Kendra 索引。

遷移機器人

Amazon Lex V2 API 使用更新的資訊架構，可簡化機器人中多種語言的資源版本控制和支援。如需詳細資訊，請參閱《Amazon Lex V2 開發人員指南》中的遷移指南。 Amazon Lex V2

若要使用這些新功能，您需要遷移機器人。當您遷移機器人時，Amazon Lex 會提供下列項目：

- 遷移會將您的自訂意圖和槽類型複製到 Amazon Lex V2 機器人。
- 您可以將多種語言新增至相同的 Amazon Lex V2 機器人。在 Amazon Lex V1 中，您會為每個語言建立個別的機器人。您可以將每個使用不同語言的多個 Amazon Lex V1 機器人遷移至一個 Amazon Lex V2 機器人。
- Amazon Lex 會將 Amazon Lex V1 內建插槽類型和意圖映射至 Amazon Lex V2 內建插槽類型和意圖。如果無法遷移內建項目，Amazon Lex 會傳回一則訊息，告訴您接下來該怎麼做。

遷移程序不會遷移下列項目：

- 別名
- Amazon Kendra 索引
- AWS Lambda 函數
- 對話日誌設定
- 簡訊管道，例如 Slack
- Tags (標籤)

若要遷移機器人，您的使用者或角色必須具有 Amazon Lex 和 Amazon Lex V2 API 操作的 IAM 許可。如要了解必要的許可，請參閱 允許使用者將機器人遷移至 Amazon Lex V2 APIs。

遷移機器人（主控台）

使用 Amazon Lex V1 主控台將機器人的結構遷移至 Amazon Lex V2 機器人。

使用主控台將機器人遷移至 Amazon Lex V2 API

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

2. 從左側功能表中，選擇遷移工具。
3. 從機器人清單中，選擇您要遷移的機器人，然後選擇遷移。
4. 選擇您要遷移的機器人版本，然後輸入要遷移的機器人名稱。如果您輸入現有 Amazon Lex V2 機器人的名稱，Amazon Lex V1 機器人會遷移至詳細資訊中顯示的語言，並覆寫語言的草稿版本。
5. 選擇下一步。
6. 選擇 Amazon Lex 用來執行機器人 Amazon Lex V2 API 版本的 IAM 角色。您可以選擇建立具有執行機器人所需最低許可的新角色，也可以選擇現有的 IAM 角色。
7. 選擇下一步。
8. 檢閱遷移的設定。如果看起來沒問題，請選擇開始遷移。

開始遷移程序後，您會返回遷移工具開始頁面。您可以在歷史記錄表格中監控遷移的進度。當遷移狀態欄顯示完成遷移完成時。

Amazon Lex 使用 Amazon Lex V2 API 中的 `StartImport`操作來匯入遷移的機器人。每個遷移都會在 Amazon Lex V2 主控台匯入歷史記錄表中看到一個項目。

在遷移期間，Amazon Lex 可能會在機器人中找到無法遷移的資源。您會收到無法遷移之每個資源的錯誤或警告訊息。每個訊息都包含 文件的連結，說明如何解決問題。

遷移 Lambda 函數

Amazon Lex V2 會變更為機器人定義 Lambda 函數的方式。它只允許機器人中每種語言在別名中有一個 Lambda 函數。如需遷移 Lambda 函數的詳細資訊，請參閱 將 Lambda 函數從 Amazon Lex V1 遷移至 Amazon Lex V2。

遷移訊息

在遷移期間，Amazon Lex 可能會找到無法遷移至同等 Amazon Lex V2 資源的資源，例如內建槽類型。發生這種情況時，Amazon Lex 會傳回遷移訊息，說明發生的情況，並提供 文件的連結，告訴您如何修正遷移問題。下列各節說明遷移機器人時可能發生的問題，以及如何修正問題。

主題

- 內建意圖
- 內建插槽類型

- 對話日誌
- 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 中不支援的內建意圖時，意圖會映射到 Amazon Lex V2 機器人中的自訂意圖。自訂意圖不包含表達用語。若要繼續使用意圖，請新增範例表達用語。

內建插槽類型

任何使用 Amazon Lex V2 中不支援之插槽類型的已遷移插槽都不會獲得插槽類型值。若要使用此槽：

- 建立自訂插槽類型
- 新增槽類型預期的槽類型值
- 更新插槽以使用新的自訂插槽類型

對話日誌

遷移不會更新 Amazon Lex V2 機器人的對話日誌設定。

設定對話日誌

1. 在 https：//https://console.aws.amazon.com/lexv2 開啟 Amazon Lex V2 主控台。
2. 從機器人清單中，選擇您要設定其對話日誌的機器人。
3. 從左側功能表中，選擇別名，然後從清單中選擇別名。
4. 在對話日誌區段中，選擇管理對話日誌以設定機器人別名的對話日誌。

訊息群組

Amazon Lex V2 每個訊息群組僅支援一個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超過三個訊息，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 使用不同的機制進行追蹤、釐清和掛斷提示。

對於後續提示，使用內容轉移在履行後切換到不同的意圖。

例如，假設您有意預訂設定為傳回名為 的輸出內容的租車book_car_fulfilled。
滿足意圖時，Amazon Lex 會將輸出內容變數設定為 book_car_fulfilled。由於 book_car_fulfilled 是主動內容，只要使用者表達用語被識別為試圖引出該意圖，則將考慮使用book_car_fulfilled作為輸入內容的意圖進行辨識。您可以將此用於預訂租車後才有意義的意圖，例如透過電子郵件傳送收據或修改保留。

Amazon Lex V2 不支援釐清提示和掛斷片語 （中止陳述式）。Amazon Lex V2 機器人包含預設備用意圖，如果沒有相符意圖，則會叫用該意圖。若要傳送具有重試的釐清提示，請設定 Lambda 函數，並在備用意圖中啟用對話方塊程式碼掛勾。Lambda 函數可以輸出澄清提示做為回應，以及工作階段屬性中的重試值。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對話。

其他 Amazon Lex V1 功能

遷移工具僅支援 Amazon Lex V1 機器人及其基礎意圖、槽類型和槽的遷移。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
- 機器人管道：在訊息平台上部署 Amazon Lex V2 機器人
- 對話日誌設定：使用對話日誌進行監控
- Amazon Kendra 索引：AMAZON.KendraSearchIntent
- Lambda 函數：使用 AWS Lambda 函數
- 標籤：標記資源

將 Lambda 函數從 Amazon Lex V1 遷移至 Amazon Lex V2

Amazon Lex V2 僅允許一個 Lambda 函數用於機器人中的每個語言。Lambda 函數及其設定是針對您在執行時間使用的機器人別名進行設定。

如果針對該意圖啟用了對話方塊和履行程式碼掛勾，則會針對該語言中的所有意圖叫用 Lambda 函數。

Amazon Lex V2 Lambda 函數的輸入和輸出訊息格式與 Amazon Lex V1 不同。這些是 Lambda 函數輸入格式的差異。

- Amazon Lex V2 會將 currentIntent和 alternativeIntents 結構取代為 interpretations結構。每個解釋都包含意圖、意圖的 NLU 可信度分數，以及選用的情緒分析。
- Amazon Lex V2 會將 Amazon Lex V1 activeContextssessionAttributes中的 移至統一sessionState結構。此結構提供對話目前狀態的相關資訊，包括原始請求 ID。
- Amazon Lex V2 不會傳回 recentIntentSummaryView。請改用 sessionState結構中的資訊。
- Amazon Lex V2 輸入在 bot 屬性localeId中提供 botId和 。
- 輸入結構包含 inputMode 屬性，可提供輸入類型的相關資訊：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 sessionAttributes結構會由 Amazon Lex V2 中的 sessionState結構取代。
- recentIntentSummaryView 輸出中不包含 。
- Amazon Lex V1 dialogAction結構分為兩個結構，dialogAction這是sessionState結構的一部分，messages在 dialogAction.type為 時為必要ElicitIntent。Amazon Lex 會從此結構選擇要向使用者顯示的訊息。

當您使用 Amazon Lex V2 APIs 建置機器人時，每個語言的每個機器人別名只有一個 Lambda 函數，而不是每個意圖的 Lambda 函數。如果您想要繼續使用個別的函數，您可以建立路由器函數，為每個意圖啟用個別的函數。以下是您可以針對應用程式使用或修改的路由器函數。

```
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
```

已更新欄位的清單

下表提供有關 Amazon Lex V2 Lambda 請求和回應中更新欄位的詳細資訊。您可以使用這些資料表來映射版本之間的欄位。

請求

下列欄位已更新為 Lambda 函數請求格式。

作用中內容

`activeContexts` 結構現在是`sessionState`結構的一部分。

V1 結構	V2 結構
activeContexts	sessionState.activeContexts
activeContexts【*】.timeToLive	sessionState.activeContexts[*].timeToLive
activeContexts【*】.timeToLive.timeToLiveInSeconds	sessionState.activeContexts[*].timeToLive.timeToLiveInSeconds
activeContexts【*】.timeToLive.turnsToLive	sessionState.activeContexts[*].timeToLive.turnsToLive
activeContexts【*】.name	sessionState.activeContexts[*].name
activeContexts【*】.parameters	sessionState.activeContexts[*].contextAttributes

替代意圖

從索引 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` 會從 Amazon Lex V2 中的請求結構中移除 。若要查看來自 的詳細資訊`recentIntentSummaryView`，請使用 GetSession操作。

V1 結構	V2 結構
alternativeIntents	interpretations【1：*】
recentIntentSummaryView	N/A

機器人

在 Amazon Lex V2 中，機器人和別名具有識別符。機器人 ID 是 Codehook 輸入的一部分。包含別名 ID，但不包含別名名稱。Amazon Lex V2 支援相同機器人的多個地區設定，因此包含地區設定 ID。

V1 結構	V2 結構
機器人	機器人
bot.name	bot.name
N/A	bot.id
bot.alias	N/A
N/A	bot.aliasId
bot.version	bot.version
N/A	bot.localeId

目前意圖

`sessionState.intent` 結構包含作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。解譯清單中的第一個元素一律與 相同`sessionState.intent`。

V1 結構	V2 結構
currentIntent	sessionState.intent OR 解釋【0】.intent
currentIntent.name	sessionState.intent.name OR 解釋 【0】.intent.name
currentIntent.nluConfidenceScore	interpretations【0】.nluConfidence.score

對話方塊動作

`confirmationStatus` 欄位現在是 `sessionState` 結構的一部分。

V1 結構	V2 結構
currentIntent.confirmationStatus	sessionState.intent.confirmationState OR interpretations【0】.intent.confirmationState
N/A	sessionState.intent.state OR 解釋 【*】.intent.state

Amazon Kendra

`kendraResponse` 欄位現在是 `sessionState`和 `interpretations`結構的一部分。

V1 結構	V2 結構
kendraResponse	sessionState.intent.kendraResponse OR 解釋【0】.intent.kendraResponse

情緒

`sentimentResponse` 結構會移至新的`interpretations`結構。

V1 結構	V2 結構
sentimentResponse	interpretations【0】.sentimentResponse

V1 結構	V2 結構
sentimentResponse.sentimentLabel	interpretations【0】.sentimentResponse .sentiment
sentimentResponse.sentimentScore	interpretations【0】.sentimentResponse .sentimentScore

槽

Amazon Lex V2 在`sessionState.intent`結構內提供單一`slots`物件，其中包含解析的值、解譯值，以及使用者所說內容的原始值。Amazon Lex V2 也支援多值插槽，方法是將 設定為`slotShapeList`並設定`values`清單。`value` 欄位支援單一值插槽，其形狀假設為 `Scalar`。

V1 結構	V2 結構
currentIntent.slots	sessionState.intent.slots OR interpret ations【0】.intent.slots
currentIntent.slots[*].value	sessionState.intent.slots[*].value.interprete dValue OR interpretations【0】.intent.s lots【*】.value.interpretedValue
N/A	sessionState.intent.slots[*].value.shape OR interpretations【0】.intent.slots【*】.shape
N/A	sessionState.intent.slots[*].values OR interpret ations【0】.intent.slots【*】.values
currentIntent.slotDetails	sessionState.intent.slots OR interpret ations【0】.intent.slots
currentIntent.slotDetails[*].resolutions	sessionState.intent.slots[*].resolvedValues OR interpretations【0】.intent.slots【*】.r esolvedValues

V1 結構	V2 結構
currentIntent.slotDetails[*].originalValue	sessionState.intent.slots[*].originalValue OR interpretations【0】.intent.slots【*】.originalValue

其他

Amazon Lex V2 `sessionId` 欄位與 Amazon Lex V1 中的 `userId` 欄位相同。Amazon Lex V2 也會傳送發起`inputMode`人的 ：文字、DTMF 或語音。

V1 結構	V2 結構
userId	sessionId
inputTranscript	inputTranscript
invocationSource	invocationSource
outputDialogMode	responseContentType
messageVersion	messageVersion
sessionAttributes	sessionState.sessionAttributes
requestAttributes	requestAttributes
N/A	inputMode
N/A	originatingRequestId

回應

下列欄位已在 Lambda 函數回應訊息格式中變更。

作用中內容

`activeContexts` 結構已移至 `sessionState`結構。

V1 結構	V2 結構
activeContexts	sessionState.activeContexts
activeContexts【*】.timeToLive	sessionState.activeContexts[*].timeToLive
activeContexts【*】.timeToLive.timeToLiveInSeconds	sessionState.activeContexts[*].timeToLive.timeToLiveInSeconds
activeContexts【*】.timeToLive.turnsToLive	sessionState.activeContexts[*].timeToLive.turnsToLive
activeContexts【*】.name	sessionState.activeContexts[*].name
activeContexts【*】.parameters	sessionState.activeContexts[*].contextAttributes

對話方塊動作

`dialogAction` 結構已移至 `sessionState`結構。您現在可以在對話方塊動作中指定多個訊息，而`genericAttachments`結構現在是 `imageResponseCard` 結構。

V1 結構	V2 結構
dialogAction	sessionState.dialogAction
dialogAction.type	sessionState.dialogAction.type
dialogAction.slotToElicit	sessionState.intent.dialogAction.slotToElicit
dialogAction.type.fulfillmentState	sessionState.intent.state
dialogAction.message	messages
dialogAction.message.contentType	messages【*】.contentType
dialogAction.message.content	messages【*】.content
dialogAction.responseCard	messages【*】.imageResponseCard

V1 結構	V2 結構
dialogAction.responseCard.version	N/A
dialogAction.responseCard.contentType	messages【*】.contentType
dialogAction.responseCard.genericAttachments	N/A
dialogAction.responseCard.genericAttachments[*].title	messages【*】.imageResponseCard.title
dialogAction.responseCard.genericAttachments[*].subTitle	messages【*】.imageResponseCard.subtitle
dialogAction.responseCard.genericAttachments[*].imageUrl	messages【*】.imageResponseCard.imageUrl
dialogAction.responseCard.genericAttachments[*].buttons	messages【*】.imageResponseCard.buttons
dialogAction.responseCard.genericAttachments[*].buttons[*].value	messages【*】.imageResponseCard.buttons[*].value
dialogAction.responseCard.genericAttachments[*].buttons[*].text	messages【*】.imageResponseCard.buttons[*].text
dialogAction.kendraQueryRequestPayload	dialogAction.kendraQueryRequestPayload
dialogAction.kendraQueryFilterString	dialogAction.kendraQueryFilterString

意圖和槽

屬於`dialogAction`結構的意圖和槽欄位現在是`sessionState`結構的一部分。

V1 結構	V2 結構
dialogAction.intentName	sessionState.intent.name
dialogAction.slots	sessionState.intent.slots

V1 結構	V2 結構
dialogAction.slots[*].key	sessionState.intent.slots[*].key
dialogAction.slots[*].value	sessionState.intent.slots[*].value.interpreted Value
N/A	sessionState.intent.slots[*].value.shape
N/A	sessionState.intent.slots[*].values

其他

`sessionAttributes` 結構現在是`sessionState`結構的一部分。結構`recentIntentSummaryReview`已移除。

V1 結構	V2 結構
sessionAttributes	sessionState.sessionAttributes
recentIntentSummaryView	N/A

Amazon Lex 中的安全

的雲端安全性 AWS 是最高優先順序。身為 AWS 客戶，您可以受益於資料中心和網路架構，這些架構專為滿足最安全敏感組織的需求而建置。

安全性是 AWS 與您之間的共同責任。共同責任模型 將此描述為雲端的安全和雲端內的安全：

- 雲端的安全性 – AWS 負責保護在 AWS Cloud 中執行 AWS 服務的基礎設施。 AWS 也為您提供可安全使用的服務。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AWS 合規計劃範圍內的服務。
- 雲端的安全性 – 您的責任取決於您使用 AWS 的服務。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 時套用共同責任模型。下列主題說明如何設定 Amazon Lex 以符合您的安全與合規目標。您也將了解如何使用其他 AWS 服務，協助您監控和保護 Amazon Lex 資源。

主題

Amazon Lex 中的資料保護

Amazon Lex 會收集客戶內容以進行故障診斷，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 存放四種類型的內容：

- 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

任何專為兒童設計的 Amazon Lex 機器人皆受兒童線上隱私權保護法 (COPPA) 規範。您可以使用主控台或 Amazon Lex API 將 childDirected 欄位設定為 ，告知 Amazon Lex 機器人受 COPPA 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
- 傳輸中加密
- 金鑰管理

靜態加密

Amazon Lex 會加密儲存的使用者表達用語。

主題

- 範例表達用語
- 客戶表達用語
- 工作階段屬性
- 請求屬性

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。此資訊會靜態加密，並用於建置機器人和建立使用者體驗。

客戶表達用語

除非 childDirected 欄位設定為 ，否則 Amazon Lex 會加密使用者傳送給機器人的表達用語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，輸入會無限期儲存。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為機器人設定的所有 AWS Lambda 函數。如果 Lambda 函數新增或更新工作階段屬性，Amazon Lex 會將新資訊傳遞回用戶端應用程式。

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通訊。它使用 HTTPS 和 AWS 簽章來與其他 服務通訊，例如 Amazon Polly 和 AWS Lambda 代表您的應用程式。

金鑰管理

Amazon Lex 會保護您的內容不受未經授權的內部金鑰使用。

Amazon Lex 的 Identity and Access Management

AWS Identity and Access Management (IAM) 是一種 AWS 服務 ，可協助管理員安全地控制對 AWS 資源的存取。IAM 管理員可控制誰可以進行身分驗證 （登入） 和授權 （具有許可） 來使用 Amazon Lex 資源。IAM 是您可以免費使用 AWS 服務 的 。

主題

目標對象

使用方式 AWS Identity and Access Management (IAM) 會根據您的角色而有所不同：

- 服務使用者 — 若無法存取某些功能，請向管理員申請所需許可 (請參閱 對 Amazon Lex 身分和存取進行故障診斷)
- 服務管理員 — 負責設定使用者存取權並提交相關許可請求 (請參閱 Amazon Lex 如何與 IAM 搭配使用)
- IAM 管理員 — 撰寫政策以管理存取控制 (請參閱 Amazon Lex 的身分型政策範例)

使用身分驗證

身分驗證是您 AWS 使用身分憑證登入 的方式。您必須以 AWS 帳戶根使用者、IAM 使用者或擔任 IAM 角色進行身分驗證。

您可以使用身分來源的登入資料，例如 AWS IAM Identity Center (IAM Identity Center)、單一登入身分驗證或 Google/Facebook 登入資料，以聯合身分的形式登入。如需有關登入的詳細資訊，請參閱《AWS 登入 使用者指南》中的如何登入您的 AWS 帳戶。

對於程式設計存取， AWS 提供 SDK 和 CLI 以密碼編譯方式簽署請求。如需詳細資訊，請參閱《IAM 使用者指南》中的 API 請求的AWS 第 4 版簽署程序。

AWS 帳戶 根使用者

當您建立 時 AWS 帳戶，您會從一個名為 AWS 帳戶 theroot 使用者的登入身分開始，該身分具有對所有 AWS 服務 和 資源的完整存取權。強烈建議不要使用根使用者來執行日常任務。有關需要根使用者憑證的任務，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

最佳實務是要求人類使用者使用聯合身分提供者，以 AWS 服務 使用臨時憑證存取 。

聯合身分是您企業目錄、Web 身分提供者的使用者，或使用來自身分來源的 AWS 服務 憑證存取 Directory Service 。聯合身分會擔任角色，而該角色會提供臨時憑證。

若需集中化管理存取權限，建議使用 AWS IAM Identity Center。如需詳細資訊，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https://docs.aws.amazon.com/IAM/latest/UserGuide/id_users.html是一種身分具備單人或應用程式的特定許可權。建議以臨時憑證取代具備長期憑證的 IAM 使用者。如需詳細資訊，請參閱《IAM 使用者指南》中的要求人類使用者使用聯合身分提供者來 AWS 使用臨時憑證存取 。

IAM 群組會指定 IAM 使用者集合，使管理大量使用者的許可權更加輕鬆。如需詳細資訊，請參閱《IAM 使用者指南》中的 IAM 使用者的使用案例。

IAM 角色

IAM 角色https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles.html的身分具有特定許可權，其可以提供臨時憑證。您可以透過從使用者切換到 IAM 角色 （主控台） 或呼叫 AWS CLI 或 AWS API 操作來擔任角色。如需詳細資訊，請參閱《IAM 使用者指南》中的擔任角色的方法。

IAM 角色適用於聯合身分使用者存取、臨時 IAM 使用者許可、跨帳戶存取權與跨服務存取，以及在 Amazon EC2 執行的應用程式。如需詳細資訊，請參閱《IAM 使用者指南》中的 IAM 中的快帳戶資源存取。

使用政策管理存取權

您可以透過建立政策並將其連接到身分或資源 AWS 來控制 AWS 中的存取。政策定義與身分或資源相關聯的許可。當委託人提出請求時 AWS ， 會評估這些政策。大多數政策會以 JSON 文件 AWS 形式存放在 中。如需進一步了解 JSON 政策文件，請參閱《IAM 使用者指南》中的 JSON 政策概觀。

管理員會使用政策，透過定義哪些主體可在哪些條件下對哪些資源執行動作，以指定可存取的範圍。

預設情況下，使用者和角色沒有許可。IAM 管理員會建立 IAM 政策並將其新增至角色，供使用者後續擔任。IAM 政策定義動作的許可，無論採用何種方式執行。

身分型政策

身分型政策是附加至身分 (使用者、使用者群組或角色) 的 JSON 許可政策文件。這類政策控制身分可對哪些資源執行哪些動作，以及適用的條件。如需了解如何建立身分型政策，請參閱《IAM 使用者指南》中的透過客戶管理政策定義自訂 IAM 許可。

身分型政策可分為內嵌政策 (直接內嵌於單一身分) 與受管政策 (可附加至多個身分的獨立政策)。如需了解如何在受管政策及內嵌政策之間做選擇，請參閱《IAM 使用者指南》中的在受管政策與內嵌政策之間選擇。

資源型政策

資源型政策是附加到資源的 JSON 政策文件。範例包括 IAM 角色信任政策與 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。您必須在資源型政策中指定主體。

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來自 IAM 的 AWS 受管政策。

其他政策類型

AWS 支援其他政策類型，可設定更多常見政策類型授予的最大許可：

- 許可界限 — 設定身分型政策可授與 IAM 實體的最大許可。如需詳細資訊，請參閱《 IAM 使用者指南》中的 IAM 實體許可界限。
- 服務控制政策 (SCP) — 為 AWS Organizations中的組織或組織單位指定最大許可。如需詳細資訊，請參閱《AWS Organizations 使用者指南》中的服務控制政策。
- 資源控制政策 (RCP) — 設定您帳戶中資源可用許可的上限。如需詳細資訊，請參閱《AWS Organizations 使用者指南》中的資源控制政策 (RCP)。
- 工作階段政策 — 在以程式設計方式為角色或聯合身分使用者建立臨時工作階段時，以參數形式傳遞的進階政策。如需詳細資訊，請參《IAM 使用者指南》中的工作階段政策。

多種政策類型

當多種類型的政策適用於請求時，產生的許可會更複雜而無法理解。若要了解如何 AWS 在涉及多種政策類型時決定是否允許請求，請參閱《IAM 使用者指南》中的政策評估邏輯。

Amazon Lex 如何與 IAM 搭配使用

在您使用 IAM 管理 Amazon Lex 的存取權之前，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 支援
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

若要全面了解 Amazon Lex 和其他 AWS 服務如何與大多數 IAM 功能搭配使用，請參閱《IAM 使用者指南》中的AWS 與 IAM 搭配使用的 服務。

Amazon Lex 的身分型政策

支援身分型政策：是

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。如需了解如何建立身分型政策，請參閱《IAM 使用者指南》中的透過客戶管理政策定義自訂 IAM 許可。

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。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 的身分型政策範例

若要檢視 Amazon Lex 身分型政策的範例，請參閱 Amazon Lex 的身分型政策範例。

Amazon Lex 內的資源型政策

支援資源型政策：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。委託人可以包含帳戶、使用者、角色、聯合身分使用者或 AWS 服務。

如需啟用跨帳戶存取權，您可以在其他帳戶內指定所有帳戶或 IAM 實體作為資源型政策的主體。如需詳細資訊，請參閱《IAM 使用者指南》中的 IAM 中的快帳戶資源存取。

Amazon Lex 的政策動作

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。

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。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 動作的清單，請參閱《服務授權參考》中的 Amazon Lex 定義的動作。

Amazon Lex 中的政策動作在動作之前使用下列字首：

```
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 的政策資源

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。最佳實務是使用其 Amazon Resource Name (ARN) 來指定資源。若動作不支援資源層級許可，使用萬用字元 (*) 表示該陳述式適用於所有資源。

```
"Resource": "*"
```

Amazon Lex 機器人資源 ARN 的格式如下。

```
arn:aws:lex:${Region}:${Account}:bot:${Bot-Name}
```

如需 ARNs 格式的詳細資訊，請參閱 Amazon Resource Name (ARNs AWS 和服務命名空間。

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:*"
```

有些 Amazon Lex 動作無法在特定資源上執行，例如用於建立資源的動作。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 ARNs，請參閱《服務授權參考》中的 Amazon Lex 定義的資源。若要了解您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 的政策條件索引鍵

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` 元素會根據定義的條件，指定陳述式的執行時機。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。若要查看所有 AWS 全域條件索引鍵，請參閱《IAM 使用者指南》中的AWS 全域條件內容索引鍵。

若要查看 Amazon Lex 條件索引鍵的清單，請參閱《服務授權參考》中的 Amazon Lex 的條件索引鍵。若要了解您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 許可政策的 `Condition` 元素中包含這些金鑰。

Amazon Lex 中的 ACLs

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 搭配 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，根據稱為標籤的屬性定義許可權。您可以將標籤連接至 IAM 實體 AWS 和資源，然後設計 ABAC 政策，以便在主體的標籤符合資源上的標籤時允許操作。

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》中的使用 ABAC 授權定義許可。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 使用標籤存取資源。

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。

Action	Resource Type (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

Action	Resource Type (資源類型)	條件索引鍵	備註
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	
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。

Action	Resource Type (資源類型)	條件索引鍵	備註
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。
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。

Action	Resource Type (資源類型)	條件索引鍵	備註
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	

搭配 Amazon Lex 使用臨時登入資料

支援臨時憑證：是

臨時登入資料提供 AWS 資源的短期存取權，當您使用聯合或切換角色時，會自動建立。 AWS 建議您動態產生臨時登入資料，而不是使用長期存取金鑰。如需詳細資訊，請參閱《IAM 使用者指南》中的IAM 中的臨時安全憑證與可與 IAM 搭配運作的AWS 服務。

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 GetFederationToken 等 AWS STS API 操作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

轉送存取工作階段 (FAS) 使用呼叫 的委託人許可 AWS 服務，並結合 AWS 服務 請求向下游服務提出請求。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。

> **Warning**
>
> 變更服務角色的許可可能會中斷 Amazon Lex 功能。只有在 Amazon Lex 提供指引時，才能編輯服務角色。

在 Amazon Lex 中選擇 IAM 角色

Amazon Lex 使用服務連結角色來呼叫 Amazon Comprehend 和 Amazon Polly。它在您的 AWS Lambda 函數上使用資源層級許可來叫用它們。

您必須提供 IAM 角色才能啟用對話標記。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 的一種服務角色 AWS 服務。服務可以擔任代表您執行動作的角色。服務連結角色會出現在您的 中 AWS 帳戶 ，並由服務擁有。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 的身分型政策範例

根據預設，使用者和角色沒有建立或修改 Amazon Lex 資源的許可。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。

如需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 (主控台)。

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 ARNs 格式，請參閱《服務授權參考》中的 Amazon Lex 的動作、資源和條件索引鍵。

主題

- 政策最佳實務
- 使用 Amazon Lex 主控台
- 允許使用者檢視他們自己的許可

- 刪除所有 Amazon Lex 機器人
- 允許使用者將機器人遷移至 Amazon Lex V2 APIs
- 使用標籤存取資源

政策最佳實務

身分型政策會判斷您帳戶中的某個人員是否可以建立、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 受管政策並邁向最低權限許可 – 若要開始將許可授予您的使用者和工作負載，請使用將許可授予許多常見使用案例的 AWS 受管政策。它們可在您的 中使用 AWS 帳戶。我們建議您定義特定於使用案例 AWS 的客戶受管政策，以進一步減少許可。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。如果透過特定 等使用服務動作 AWS 服務，您也可以使用條件來授予其存取權 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，可協助您撰寫安全且實用的政策。如需詳細資訊，請參閱《IAM 使用者指南》中的使用 IAM Access Analyzer 驗證政策。
- 需要多重要素驗證 (MFA) – 如果您的案例需要 IAM 使用者或 中的根使用者 AWS 帳戶，請開啟 MFA 以提高安全性。如需在呼叫 API 操作時請求 MFA，請將 MFA 條件新增至您的政策。如需詳細資訊，請參閱《IAM 使用者指南》中的透過 MFA 的安全 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 主控台

若要存取 Amazon Lex 主控台，您必須擁有一組最低許可。這些許可必須允許您列出和檢視 中 Amazon Lex 資源的詳細資訊 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。

對於僅呼叫 AWS CLI 或 AWS API 的使用者，您不需要允許最低主控台許可。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 提供由 建立和管理的獨立 IAM 政策，以解決許多常見的使用案例 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)。

下列 AWS 受管政策是 Amazon Lex 特有的，您可以連接到帳戶中的群組和角色：

- AmazonLexReadOnly — 授予 Amazon Lex 資源的唯讀存取權。
- AmazonLexRunBotsOnly — 授予執行 Amazon Lex 對話式機器人的存取權。
- 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。同時授予將名稱開頭為 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。

AmazonLexFullAccess 政策不會授予使用者使用`KendraSearchIntent`意圖查詢 Amazon Kendra 索引的許可。若要查詢索引，您必須將其他許可新增至政策。如要了解必要的許可，請參閱 Amazon Kendra 搜尋的 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連接到需要這些許可的 IAM 角色或群組。

如需 Amazon Lex 的 AWS 受管政策詳細資訊，請參閱 AWS Amazon Lex 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。此政策包含在主控台或使用 或 AWS CLI AWS API 以程式設計方式完成此動作的許可。

```
{
    "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 機器人

此範例政策授予您 AWS 帳戶中的使用者刪除您帳戶中任何機器人的許可。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "lex:DeleteBot"
            ],
```

```
            "Resource": [
                "*"
            ]
        }
    ]
}
```

允許使用者將機器人遷移至 Amazon Lex V2 APIs

下列 IAM 許可政策允許使用者開始將機器人從 Amazon Lex 遷移至 Amazon Lex V2 APIs，並查看遷移清單及其進度。

使用標籤存取資源

此範例政策授予您 AWS 帳戶中的使用者或角色許可，以使用具有 金鑰**Department**和值 標記的任何資源來使用 `PostText`操作**Support**。

AWS Amazon Lex 的 受管政策

AWS 受管政策是由 AWS AWS 受管政策建立和管理的獨立政策旨在為許多常用案例提供許可，以便您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予特定使用案例的最低權限許可，因為這些許可可供所有 AWS 客戶使用。我們建議您定義特定於使用案例的客戶管理政策，以便進一步減少許可。

您無法變更 AWS 受管政策中定義的許可。如果 AWS 更新受 AWS 管政策中定義的許可，則更新會影響政策連接的所有主體身分（使用者、群組和角色）。當新的 AWS 服務 啟動或新的 API 操作可用於現有服務時， AWS 最有可能更新 AWS 受管政策。

如需詳細資訊，請參閱 IAM 使用者指南中的 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 – 在模型建置服務中唯讀存取 Amazon Lex 和 Amazon Lex V2 資源。

JSON

```
{
    "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，允許執行 Amazon Lex 和 Amazon Lex V2 對話式機器人的存取權。

許可詳細資訊

此政策包含以下許可：

- lex – Amazon Lex 和 Amazon Lex V2 執行時間中所有動作的唯讀存取權。

JSON

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

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "lex:PostContent",
                "lex:PostText",
                "lex:PutSession",
                "lex:GetSession",
                "lex:DeleteSession",
                "lex:RecognizeText",
                "lex:RecognizeUtterance",
                "lex:StartConversation"
            ],
            "Resource": "*"
        }
    ]
}
```

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 對話式機器人。

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存取 Amazon Lex 和 Amazon Lex V2 模型建置和執行時間服務中的所有動作。
- cloudwatch – 允許主體檢視 Amazon CloudWatch 指標和警示。
- iam – 允許主體建立和刪除服務連結角色、傳遞角色，以及將政策連接到角色並分離。Amazon Lex 操作的許可僅限於「lex.amazonaws.com」，Amazon Lex V2 操作的許可僅限於「lexv2.amazonaws.com」。
- kendra – 允許主體列出 Amazon Kendra 索引。
- kms – 允許主體描述 AWS KMS 金鑰和別名。
- lambda – 允許主體列出 AWS Lambda 函數和管理連接到任何 Lambda 函數的許可。
- polly – 允許主體描述 Amazon Polly 語音和合成語音。

JSON

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

AWS 受管政策的 Amazon Lex 更新

檢視自此服務開始追蹤 Amazon Lex AWS 受管政策更新以來的詳細資訊。如需此頁面變更的自動提醒，請訂閱 Amazon Lex Amazon Lex 的文件歷史記錄頁面上的 RSS 摘要。

變更	描述	Date
AmazonLexFullAccess – 更新現有政策	Amazon Lex 新增了新的許可，以允許唯讀存取 Amazon Lex V2 模型建置服務操作。	2021 年 8 月 18 日
AmazonLexReadOnly – 更新至現有政策	Amazon Lex 新增了新的許可，以允許唯讀存取 Amazon Lex V2 模型建置服務操作。	2021 年 8 月 18 日
AmazonLexRunBotsOnly – 更新至現有政策	Amazon Lex 新增了新的許可，以允許唯讀存取 Amazon Lex V2 執行期服務操作。	2021 年 8 月 18 日
Amazon Lex 開始追蹤變更	Amazon Lex 開始追蹤其 AWS 受管政策的變更。	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 的唯一 IAM 角色類型。服務連結角色是由 Amazon Lex 預先定義，並包含該服務代表您呼叫其他 AWS 服務所需的所有許可。

服務連結角色可讓您更輕鬆地設定 Amazon Lex，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，否則只有 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，且該許可政策無法附加至其他 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。這可保護您的 Amazon Lex 資源，因為您不會不小心移除存取資源的許可。

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來叫用 Amazon Polly 來合成機器人的語音回應、呼叫 Amazon Comprehend 進行情緒分析，以及選用 Amazon Kendra 來搜尋索引。
- AWSServiceRoleForLexChannels – Amazon Lex 使用此服務連結角色，在管理頻道時將文字發佈到您的機器人。

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。

為 Amazon Lex 建立服務連結角色

您不需要手動建立服務連結角色，當您在 中建立機器人、機器人管道或 Amazon Kendra 搜尋意圖時 AWS 管理主控台，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、頻道關聯或 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 建立具有 AWSServiceRoleForLexBots 使用案例的服務連結角色。在 中 AWS CLI 建立具有 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。如需詳細資訊，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**
>
> 如果您嘗試刪除資源時，Amazon Lex 服務正在使用該角色，則刪除可能會失敗。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 來刪除 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 Service 連結角色支援的區域

Amazon Lex 支援在提供服務的所有區域中使用服務連結角色。如需詳細資訊，請參閱 Amazon Lex 端點和配額。

對 Amazon Lex 身分和存取進行故障診斷

使用以下資訊來協助您診斷和修正使用 Amazon Lex 和 IAM 時可能遇到的常見問題。

主題

- 我無權在 Amazon Lex 中執行動作
- 我未獲得執行 iam:PassRole 的授權

- 我想要允許 以外的人員 AWS 帳戶 存取我的 Amazon Lex 資源

我無權在 Amazon Lex 中執行動作

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。

我未獲得執行 iam:PassRole 的授權

如果您收到錯誤，告知您無權執行 iam:PassRole動作，您的政策必須更新，以允許您將角色傳遞給 Amazon Lex。

有些 AWS 服務 可讓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 marymajor 使用者嘗試使用主控台在 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 沒有將角色傳遞給服務的許可。

```
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。

我想要允許 以外的人員 AWS 帳戶 存取我的 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 Amazon Lex 如何與 IAM 搭配使用。
- 若要了解如何 AWS 帳戶 在您擁有的 資源之間提供存取權，請參閱《IAM 使用者指南》中的在您擁有 AWS 帳戶 的另一個 IAM 使用者中提供存取權。
- 若要了解如何將資源的存取權提供給第三方 AWS 帳戶，請參閱《IAM 使用者指南》中的將存取權提供給第三方 AWS 帳戶 擁有。
- 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)。
- 如需了解使用角色和資源型政策進行跨帳戶存取之間的差異，請參閱《IAM 使用者指南》中的 IAM 中的跨帳戶資源存取。

Amazon Lex 中的監控

監控對於維護 Amazon Lex 聊天機器人的可靠性、可用性和效能至關重要。本主題說明如何使用 Amazon CloudWatch Logs 和 AWS CloudTrail 來監控 Amazon Lex，並描述 Amazon Lex 執行時間和頻道關聯指標。

主題

- 使用 Amazon CloudWatch 監控 Amazon Lex
- 使用 AWS CloudTrail 日誌監控 Amazon Lex API 呼叫

使用 Amazon CloudWatch 監控 Amazon Lex

若要追蹤 Amazon Lex 機器人的運作狀態，請使用 Amazon CloudWatch。使用 CloudWatch，您可以取得個別 Amazon Lex 操作或 帳戶全域 Amazon Lex 操作的指標。您也可以設定 CloudWatch 警示，以便在一或多個指標超過您定義的閾值時收到通知。例如，您可以監控在特定期間內對機器人提出的請求數量、檢視成功請求的延遲，或在錯誤超過閾值時發出警示。

Amazon Lex 的 CloudWatch 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。維度是一組用來識別指標的名稱與數值對。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 管理主控台、 AWS CLI或 CloudWatch API 取得 Amazon Lex 的指標。您可以透過其中一個 Amazon AWS 軟體開發套件 (SDKs) 或 CloudWatch API 工具來使用 CloudWatch API。Amazon Lex 主控台會根據來自 CloudWatch API 的原始資料顯示圖形。

您必須擁有適當的 CloudWatch 許可，才能使用 CloudWatch 監控 Amazon Lex。如需詳細資訊，請參閱《Amazon CloudWatch 使用者指南》中的 Amazon CloudWatch 的身分驗證和存取控制。 Amazon CloudWatch

檢視 Amazon Lex 指標

使用 Amazon Lex 主控台或 CloudWatch 主控台檢視 Amazon Lex 指標。

檢視指標 (Amazon Lex 主控台)

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

若要檢視指標 (CloudWatch 主控台)

1. 登入 AWS 管理主控台 ，並在 https：//https://console.aws.amazon.com/cloudwatch/ 開啟 CloudWatch 主控台。
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 警示會監看指定期間內的單一指標，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。動作是根據指標在您指定的數個期間內相對於指定閾值的值。當警示變更狀態時，CloudWatch 也可以傳送 Amazon SNS 訊息給您。

CloudWatch 警示只有在狀態變更且在您指定的期間持續存在時，才會叫用動作。

設定警示

1. 登入 AWS 管理主控台 ，並在 https：//https://console.aws.amazon.com/cloudwatch/ 開啟 CloudWatch 主控台。
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. 如果您希望 CloudWatch 在達到警示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 ALARM。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)。

Amazon Lex 執行期的 CloudWatch 指標

下表說明 Amazon Lex 執行時間指標。

指標	Description
KendraIndexAccessError	Amazon Lex 無法存取 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
KendraLatency	Amazon Kendra 回應來自 的請求所需的時間AMAZON.KendraSearchIntent 。

指標	Description
	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
KendraSuccess	從 AMAZON.KendraSearchIntent 到 Amazon Kendra 索引的成功請求數量。 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	Description
KendraSystemErrors	Amazon Lex 無法查詢 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
KendraThrottledEvents	Amazon Kendra 調節來自 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

指標	Description
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
RuntimeConcurrency	指定期間內並行連線的數量。 RuntimeConcurrency 會以 的形式報告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 其他操作的有效維度： • 操作、BotName、BotVersion • 操作、BotName、BotAlias 單位：計數

指標	Description
RuntimeInvalidLambdaResponses	指定期間內的 invalid AWS Lambda (Lambda) 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
RuntimeLambdaErrors	指定期間內的 Lambda 執行時間錯誤數目。 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

指標	Description
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
RuntimeSucessfulRequestLatency **Important** 此指標是 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

指標	Description
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
RuntimeThrottledEvents	已調節的請求數目。Amazon Lex 會在收到超過您帳戶每秒交易限制的請求時調節請求。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 服務限制。 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	Description
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 執行期指標使用 AWS/Lex 命名空間，並在下列維度中提供指標。您可以在 CloudWatch 主控台中依維度分組指標：

維度	Description
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。

Amazon Lex 頻道關聯的 CloudWatch 指標

頻道關聯是 Amazon Lex 與 Facebook 等簡訊頻道之間的關聯。下表說明 Amazon Lex 頻道關聯指標。

指標	Description
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 在指定期間內由簡訊管道傳送的訊息受到限流的次數。
BotChannelOutboundThrottledEvents	從 Amazon Lex 到訊息管道的傳出事件在指定期間內受到限流的次數。
BotChannelRequestCount	指定期間內在管道上提出的請求數量。
BotChannelResponseCardErrors	Amazon Lex 無法在指定期間內張貼回應卡的次數。
BotChannelSystemErrors	在指定期間內，頻道在 Amazon Lex 中發生的內部錯誤數目。

Amazon Lex 頻道關聯指標使用 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指標：

維度	Description
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。

對話日誌的 CloudWatch 指標

Amazon Lex 使用下列指標進行對話記錄：

指標	Description
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：Count
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：Count
ConversationLogsTextDeliverySuccess	在指定期間內成功交付至 CloudWatch Logs 的文字日誌數量。 單位：Count
ConversationLogsTextDeliveryFailure	在指定期間內無法交付至 CloudWatch Logs 的文字日誌數量。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 Logs 日誌群組。 單位：Count

Amazon Lex 對話日誌指標使用 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指標。

維度	Description
BotAlias	依機器人別名將指標分組。

維度	Description
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 已與 服務整合 AWS CloudTrail，此服務提供由 Amazon Lex AWS 中的使用者、角色或服務所採取之動作的記錄。CloudTrail 會將 Amazon Lex 的 API 呼叫子集擷取為事件，包括來自 Amazon Lex 主控台的呼叫，以及來自對 Amazon Lex APIs程式碼呼叫。如果您建立線索，則可以將 CloudTrail 事件持續交付至 Amazon S3 儲存貯體，包括 Amazon Lex 的事件。即使您未設定追蹤，依然可以透過 CloudTrail 主控台中的事件歷史記錄檢視最新事件。使用 CloudTrail 收集的資訊，您可以判斷向 Amazon Lex 提出的請求、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，包括如何設定及啟用，請參閱《AWS CloudTrail 使用者指南》。

CloudTrail 中的 Amazon Lex 資訊

當您建立 AWS 帳戶時，會在您的帳戶上啟用 CloudTrail。當 Amazon Lex 中發生支援的事件活動時，該活動會與事件歷史記錄中的其他 AWS 服務事件一起記錄在 CloudTrail 事件中。您可以在 AWS 帳戶中檢視、搜尋和下載最近的事件。如需詳細資訊，請參閱《使用 CloudTrail 事件歷史記錄檢視事件》https://docs.aws.amazon.com/awscloudtrail/latest/userguide/view-cloudtrail-events.html。

若要持續記錄您 AWS 帳戶中的事件，包括 Amazon Lex 的事件，請建立追蹤。線索能讓 CloudTrail 將日誌檔案交付至 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。線索會記錄 AWS 分割區中所有區域的事件，並將日誌檔案交付至您指定的 S3 儲存貯體。此外，您可以設定其他 AWS 服務，以進一步分析和處理 CloudTrail 日誌中所收集的事件資料。如需詳細資訊，請參閱：

- 建立追蹤的概觀
- CloudTrail 支援的服務和整合
- 設定 CloudTrail 的 Amazon SNS 通知
- 從多個區域接收 CloudTrail 日誌檔案，以及從多個帳戶接收 CloudTrail 日誌檔案

Amazon Lex 支援將下列操作記錄為 CloudTrail 日誌檔案中的事件：

- 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

每一筆事件或日誌專案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 userIdentity 元素。

如需有關在 CloudTrail 日誌中記錄的 Amazon Lex 動作的資訊，請參閱 Amazon Lex Model Building Service。例如，對 PutBot、 GetBot和 DeleteBot操作的呼叫會在 CloudTrail 日誌中產生項目。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 日誌檔案項目

權杖是一種組態，能讓事件以日誌檔案的形式交付至您指定的 S3 儲存貯體。CloudTrail 日誌檔案包含一個或多個日誌項目。一個事件為任何來源提出的單一請求，並包含請求動作、請求的日期和時間、請求參數等資訊。CloudTrail 日誌檔案並非依公有 API 呼叫追蹤記錄的堆疊排序，因此不會以任何特定順序出現。

下列範例 CloudTrail 日誌項目會顯示呼叫 `PutBot`操作的結果。

Amazon Lex 的合規驗證

在多個合規計畫中，第三方稽核人員會評估 Amazon Lex 的安全與 AWS 合規。Amazon Lex 是符合 HIPAA 資格的服務。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。

您使用 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 符合 PCI 等標準， AWS 會提供下列資源協助您：

- 安全與合規快速入門指南 – 部署指南，討論架構考量，並提供在 上部署以安全與合規為重心的基準環境的步驟 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 – 工作手冊和指南的集合，可能適用於您的產業和位置
- AWS Config – 評估資源組態是否符合內部實務、產業準則和法規的服務
- AWS Security Hub CSPM – 內安全狀態的完整檢視 AWS ，可協助您檢查是否符合安全產業標準和最佳實務

如需特定合規計劃範圍內 AWS 的服務清單，請參閱合規計劃範圍內的 AWS 服務。如需一般資訊，請參閱 AWS 合規計劃。

Amazon Lex 中的彈性

AWS 全球基礎設施是以 AWS 區域和可用區域為基礎建置的。 AWS 區域提供多個實體隔離和隔離的可用區域，這些區域以低延遲、高輸送量和高度備援的網路連接。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 AWS 區域和可用區域的詳細資訊，請參閱 AWS 全球基礎設施。

除了 AWS 全球基礎設施之外，Amazon Lex 還提供數種功能，以協助支援您的資料彈性和備份需求。

Amazon Lex 中的基礎設施安全

作為受管服務，Amazon Lex 受到 Amazon Web Services：安全程序概觀白皮書中所述的 全球網路安全程序的保護 AWS 。

您可以使用發佈的 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，其中可能包含根據來源 IP 地址的限制。您也可以使用 Amazon Lex 政策來控制來自特定 Amazon Virtual Private Cloud (Amazon VPC) 端點或特定 VPCs存取。實際上，這只會隔離網路中特定 VPC 對指定 Amazon Lex 資源 AWS 的網路存取。

Amazon Lex 中的準則和配額

下列各節提供使用 Amazon Lex 時的指導方針和配額。

主題

支援的區域

如需可使用 Amazon Lex AWS 的區域清單，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 – 中的所有 Amazon Lex 模型建置和執行時間 API 操作都會API 參考使用簽章 V4 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 使用 Amazon Simple Storage Service (S3) API 參考中授權標頭：在單一區塊 (AWS Signature 第 4 版） 中傳輸承載的簽章計算中所述的未簽章承載選項。

  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，Amazon Lex 如何從使用者表達用語擷取槽值：

Amazon Lex 會使用您在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。從使用者表達用語 "Tell me the prediction for ..." Amazon Lex 了解以下什麼是 zodiac 符號。

當 valueSelectionStrategy 欄位設定為ORIGINAL_VALUE使用 PutSlotType操作，或在主控台中選取展開值時，如果使用者說「告知我地球預測」，Amazon Lex 會推斷「地球」是 ，並將其ZodiacSign傳遞給用戶端應用程式或 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之語音互動的結果時，不保證槽值的大小寫。例如，如果您要引出 AMAZON.Movie 內建槽類型的值，而使用者說或輸入「強風，Amazon Lex 可能會傳回「強風、「強風」或「強風」。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 (D.V.D.)
  - 以空格分隔的大寫字母 (D V D)

- Amazon Lex 不支援 Alexa Skills Kit 支援的 AMAZON.LITERAL 內建槽類型。不過，Amazon Lex 支援建立自訂插槽類型，您可以使用這些類型來實作此功能。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。相對日期和時間會在 Amazon Lex 正在處理請求的區域中解析。

  對於AMAZON.TIME內建插槽類型，如果使用者未指定時間在中午之前或之後，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。

- 在機器人中提供可轉換的訓練資料，會降低 Amazon Lex 了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。此表達用語不會對應到 Amazon Lex 在建置時為機器人建置語言模型時可以從中學習的特定意圖。因此，當使用者在執行時間輸入此表達用語時，Amazon Lex 無法挑選具有高度可信度的意圖。

  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 也有了解使用者確認的語言模型。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 進行身分驗證，您可以使用 Amazon Cognito 使用者 ID 做為 Amazon Lex 使用者 ID。請注意，為機器人設定的任何 Lambda 函數都必須有自己的身分驗證機制，才能識別代表 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，您可以定義具有範例表達用語的意圖 (`NothingIntent`) (「我不需要任何內容」、「結束」、「再見」)、沒有插槽，也沒有設定為程式碼掛勾的 Lambda 函數。這可讓使用者從容地關閉對話。

配額

本節說明 Amazon Lex 中目前的配額。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。增加服務配額可能需要幾天的時間。如果您要在較大的專案中增加配額，請務必將此時間新增至您的計劃。

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。

- 在文字模式中使用 PostContent或 PostText操作時，與機器人的並行對話數目上限為 $LATEST 2 個，所有其他別名則為 50 個。配額會分別套用至每個 API。

- 在語音模式下使用 PostContent操作時，與機器人的並行文字模式對話數目上限為 $LATEST 2，所有其他別名則為 125。配額會分別套用至每個 API。

- 機器人$LATEST別名的並行工作階段管理呼叫數目上限 (PutSession、 和 DeleteSession) 為 2GetSession，所有其他別名則為 50。

- Lambda 函數的輸入大小上限為 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$LATEST版本應僅用於手動測試。Amazon Lex 會限制您可以對機器人$LATEST版本提出的執行時間請求數量。

- 當您更新機器人$LATEST版本時，Amazon Lex 會使用機器人$LATEST版本終止任何用戶端應用程式的任何進行中對話。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對話後，直到工作階段過期，Amazon Lex 會使用相同的機器人版本，即使您更新機器人別名以指向另一個版本。

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

- 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。

- 您最多可以將 250 個意圖與機器人建立關聯。

- 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。

- 每個 AWS 帳戶最多可以建立 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 個表達用語，但如果您使用的表達用語較少，Amazon Lex 可能更能夠辨識所提供集合之外的輸入。

- 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 不支援內建意圖的槽引出。您無法建立 Lambda 函數，以從內建意圖衍生的意圖傳回回應中的`ElicitSlot`指令。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詞。

API 參考

本節提供 Amazon Lex API 操作的文件。如需可使用 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 Model Building Service 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 Runtime Service 支援下列動作：

- DeleteSession
- GetSession
- PostContent
- PostText

- PutSession

Amazon Lex 模型建置服務

Amazon Lex Model Building Service 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

根據版本建立新的機器人$LATEST版本。如果自您建立最後一個$LATEST版本以來，此資源的版本尚未變更，Amazon Lex 不會建立新的版本。它會傳回上次建立的版本。

> **Note**
>
> 您只能更新機器人的$LATEST版本。您無法更新使用 `CreateBotVersion`操作建立的編號版本。

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

您要建立新版本之機器人的名稱。名稱區分大小寫。

長度限制：長度下限為 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$LATEST版本的特定修訂。如果您指定檢查總和，且機器人的$LATEST版本具有不同的檢查總和，則會傳回PreconditionFailedException例外狀況，且 Amazon Lex 不會發佈新版本。如果您未指定檢查總和，Amazon Lex 會發佈 $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

識別已建立之機器人版本的檢查總和。

類型：字串

childDirected

對於使用 Amazon Lex Model Building Service 建立的每個 Amazon Lex 機器人，您必須指定您對 Amazon Lex 的使用是否與網站、程式或其他應用程式相關，而該網站、程式或其他應用程式全部或部分針對 13 以下兒童，並在 `false` `childDirected` 欄位中指定 `true` 或 以遵守兒童線上隱私權保護法 (COPPA)。透過`true`在 `childDirected` 欄位中指定 ，您確認使用 Amazon Lex

與網站、程式或其他應用程式相關，而該網站、程式或其他應用程式全部或部分以 13 以下兒童為目標，且受到 COPPA 的約束。透過false在 childDirected 欄位中指定 ，您確認 Amazon Lex 的使用與網站、程式或其他應用程式無關，而該網站、程式或其他應用程式全部或部分針對 13 以下且受 COPPA 約束的 兒童。您無法為 childDirected 欄位指定預設值，該值無法準確反映您對 Amazon Lex 的使用是否與網站、程式或其他應用程式相關，而該網站、程式或其他應用程式是針對或部分針對未滿 13 歲且受 COPPA 規範的 13 歲兒童。

如果您使用 Amazon Lex 與網站、程式或其他應用程式有關，而該網站、程式或其他應用程式全部或部分導向至未滿 13 歲的孩童，則您必須根據 COPPA 取得任何必要的可驗證父系同意。如需將 Amazon Lex 與網站、程式或其他應用程式搭配使用的相關資訊，而這些網站、程式或其他應用程式全部或部分針對未滿 13 歲的孩童，請參閱 Amazon Lex 常見問答集。

類型：布林值

clarificationPrompt

Amazon Lex 在不了解使用者請求時使用的訊息。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

指出使用者輸入的表達用語是否應傳送至 Amazon Comprehend 進行情緒分析。

類型：布林值

enableModelImprovements

指出機器人是否使用準確性改進。 true 指出機器人正在使用改進，否則是 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 保留對話中所收集資料的秒數上限。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

此機器人$LATEST版本更新的日期。

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

長度限制：長度下限為 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。Amazon Lex 建置機器人後，會將 status設定為 READY。如果 Amazon Lex 無法建

置機器人，則會將 status設定為 FAILED。Amazon Lex 會在failureReason回應元素中傳回失敗的原因。

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

Amazon Lex 用來與使用者進行語音互動的 Amazon Polly 語音 ID。

類型：字串

錯誤

BadRequestException

請求格式不正確。例如，值無效或缺少必要欄位。請檢查欄位值，然後再試一次。

HTTP 狀態碼：400

ConflictException

處理請求時發生衝突。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

找不到請求中指定的資源。請檢查資源，然後再試一次。

HTTP 狀態碼：404

PreconditionFailedException

您嘗試變更的資源檢查總和與請求中的檢查總和不相符。請檢查資源的檢查總和，然後再試一次。

HTTP 狀態碼：412

另請參閱

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列界面 V2
- AWS 適用於 .NET V4 的 SDK
- AWS 適用於 C++ 的 SDK
- AWS 適用於 Go 的 SDK v2
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 JavaScript V3 的 SDK
- AWS 適用於 Kotlin 的 SDK
- AWS 適用於 PHP V3 的 SDK
- AWS 適用於 Python 的 SDK
- AWS 適用於 Ruby V3 的 SDK

CreateIntentVersion

服務：Amazon Lex Model Building Service

根據意圖的版本建立新的意圖$LATEST版本。如果自上次更新後，此意圖的$LATEST版本尚未變更，Amazon Lex 不會建立新的版本。它會傳回您建立的最後一個版本。

> **Note**
>
> 您只能更新意圖的$LATEST版本。您無法更新使用 CreateIntentVersion操作建立的編號版本。

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

您要建立新版本之意圖的名稱。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

請求接受採用 JSON 格式的下列資料。

checksum

應用於建立新$LATEST版本的意圖版本總和檢查碼。如果您指定檢查總和，且意圖的$LATEST版本具有不同的檢查總和，Amazon Lex 會傳回PreconditionFailedException例外狀況，而不會發佈新版本。如果您未指定檢查總和，Amazon Lex 會發佈 $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

已建立意圖版本的檢查總和。

類型：字串

conclusionStatement

在 `fulfillmentActivity` 欄位指定的 Lambda 函數滿足意圖後，Amazon Lex 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 會先使用 確認使用者意圖的提示，再予以滿足。

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 會為每個使用者輸入調用此 Lambda 函數。

類型：CodeHook 物件

followUpPrompt

如果已定義，Amazon Lex 會在滿足意圖後使用此提示來請求額外的使用者活動。

類型：FollowUpPrompt 物件

fulfillmentActivity

描述如何實現意圖。

類型：FulfillmentActivity 物件

inputContexts

物件陣列`InputContext`，列出 Amazon Lex 在與使用者的對話中必須處於作用中狀態的內容，以選擇意圖。

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

若有組態資訊，用於將 Amazon Kendra 索引與`AMAZON.KendraSearchIntent`意圖連線。

類型：KendraConfiguration 物件

lastUpdatedDate

意圖更新的日期。

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

物件陣列OutputContext，列出意圖在滿足意圖時啟用的內容。

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符。

類型：字串

rejectionStatement

如果使用者對 中定義的問題回答「否」confirmationPrompt，Amazon Lex 會回應此陳述式，以確認意圖已取消。

類型：Statement 物件

sampleUtterances

為意圖設定的範例表達用語陣列。

類型：字串陣列

陣列成員：項目數下限為 0。最多 1500 個項目。

長度限制：長度下限為 1。長度上限為 200。

slots

槽類型的陣列，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

指派給新版本意圖的版本編號。

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

請求格式不正確。例如，值無效或缺少必要欄位。請檢查欄位值，然後再試一次。

HTTP 狀態碼：400

ConflictException

處理請求時發生衝突。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

找不到請求中指定的資源。請檢查資源，然後再試一次。

HTTP 狀態碼：404

PreconditionFailedException

您嘗試變更的資源檢查總和與請求中的檢查總和不相符。請檢查資源的檢查總和，然後再試一次。

HTTP 狀態碼：412

另請參閱

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列界面 V2
- AWS 適用於 .NET V4 的 SDK
- AWS 適用於 C++ 的 SDK
- AWS 適用於 Go 的 SDK v2
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 JavaScript V3 的 SDK
- AWS 適用於 Kotlin 的 SDK
- AWS 適用於 PHP V3 的 SDK
- AWS 適用於 Python 的 SDK
- AWS 適用於 Ruby V3 的 SDK

CreateSlotTypeVersion

服務：Amazon Lex Model Building Service

根據指定插槽類型的版本，建立新的插槽類型$LATEST版本。如果此資源的$LATEST版本在您建立的最後一個版本之後沒有變更，Amazon Lex 不會建立新的版本。它會傳回您建立的最後一個版本。

> **Note**
>
> 您只能更新插槽類型的$LATEST版本。您無法更新使用 `CreateSlotTypeVersion`操作建立的編號版本。

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的槽類型名稱。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

您要發佈之槽類型的$LATEST版本總和檢查碼。如果您指定檢查總和，且槽類型的$LATEST版本具有不同的檢查總和，Amazon Lex 會傳回PreconditionFailedException例外狀況，而不會發佈新版本。如果您未指定檢查總和，Amazon Lex 會發佈 $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

槽類型的$LATEST版本總和檢查碼。

類型：字串

createdDate

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

定義槽類型可採用之值的EnumerationValue物件清單。

類型：EnumerationValue 物件陣列

陣列成員：項目數下限為 0。10000 個項目的數量上限。

lastUpdatedDate

槽類型更新的日期。當您建立資源時，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

內建槽類型使用槽類型的父項。

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

延伸父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 用來判斷槽值的策略。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

請求格式不正確。例如，值無效或缺少必要欄位。請檢查欄位值，然後再試一次。

HTTP 狀態碼：400

ConflictException

處理請求時發生衝突。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

找不到請求中指定的資源。請檢查資源，然後再試一次。

HTTP 狀態碼：404

PreconditionFailedException

您嘗試變更的資源檢查總和與請求中的檢查總和不相符。請檢查資源的檢查總和，然後再試一次。

HTTP 狀態碼：412

另請參閱

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列界面 V2
- AWS 適用於 .NET V4 的 SDK
- AWS 適用於 C++ 的 SDK
- AWS 適用於 Go 的 SDK v2
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 JavaScript V3 的 SDK
- AWS 適用於 Kotlin 的 SDK
- AWS 適用於 PHP V3 的 SDK

- AWS 適用於 Python 的 SDK
- AWS 適用於 Ruby 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。若要刪除特定版本的機器人，請使用 DeleteBotVersion操作。DeleteBot 操作不會立即移除機器人結構描述。相反地，它會標記為刪除，並在稍後移除。

Amazon Lex 無限期地存放表達用語，以改善機器人回應使用者輸入的能力。刪除機器人時，不會移除這些表達用語。若要移除表達用語，請使用 DeleteUtterances操作。

如果機器人有別名，則無法刪除它。相反地， DeleteBot操作會傳回ResourceInUseException例外狀況，其中包含參照機器人的別名參考。若要移除機器人的參考，請刪除別名。如果您再次收到相同的例外狀況，請刪除參考別名，直到DeleteBot操作成功為止。

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。

長度限制：長度下限為 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

請求格式不正確。例如，值無效或缺少必要欄位。請檢查欄位值，然後再試一次。

HTTP 狀態碼：400

ConflictException

處理請求時發生衝突。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

找不到請求中指定的資源。請檢查資源，然後再試一次。

HTTP 狀態碼：404

ResourceInUseException

您嘗試刪除的資源是由另一個資源所參考。使用此資訊移除您嘗試刪除之資源的參考。

例外狀況的內文包含描述資源的 JSON 物件。

```
{ "resourceType": BOT | BOTALIAS | BOTCHANNEL | INTENT,

"resourceReference": {

"name": string, "version": string } }
```

exampleReference

描述參考您嘗試刪除之資源的資源。此物件會傳回為ResourceInUseException例外狀況的一部分。

HTTP 狀態碼：400

另請參閱

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列界面 V2
- AWS 適用於 .NET V4 的 SDK
- AWS 適用於 C++ 的 SDK
- AWS 適用於 Go 的 SDK v2
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 JavaScript V3 的 SDK
- AWS 適用於 Kotlin 的 SDK
- AWS 適用於 PHP V3 的 SDK
- AWS 適用於 Python 的 SDK
- AWS 適用於 Ruby V3 的 SDK

DeleteBotAlias

服務：Amazon Lex Model Building Service

刪除指定機器人的別名。

您無法刪除用於機器人與簡訊管道之間關聯的別名。如果在頻道關聯中使用別名，DeleteBot操作會傳回ResourceInUseException例外狀況，其中包含參考機器人的頻道關聯。您可以透過刪除頻道關聯來移除別名的參考。如果您再次收到相同的例外狀況，請刪除參考關聯，直到DeleteBotAlias操作成功為止。

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。

長度限制：長度下限為 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

請求格式不正確。例如，值無效或缺少必要欄位。請檢查欄位值，然後再試一次。

HTTP 狀態碼：400

ConflictException

處理請求時發生衝突。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

找不到請求中指定的資源。請檢查資源，然後再試一次。

HTTP 狀態碼：404

ResourceInUseException

您嘗試刪除的資源是由另一個資源所參考。使用此資訊移除您嘗試刪除之資源的參考。

例外狀況的內文包含描述資源的 JSON 物件。

```
{ "resourceType": BOT | BOTALIAS | BOTCHANNEL | INTENT,
```

```
"resourceReference": {

"name": string, "version": string } }
```

exampleReference

描述參考您嘗試刪除之資源的資源。此物件會傳回為ResourceInUseException例外狀況的一部分。

HTTP 狀態碼：400

另請參閱

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列界面 V2
- AWS 適用於 .NET V4 的 SDK
- AWS 適用於 C++ 的 SDK
- AWS 適用於 Go 的 SDK v2
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 JavaScript V3 的 SDK
- AWS 適用於 Kotlin 的 SDK
- AWS 適用於 PHP V3 的 SDK
- AWS 適用於 Python 的 SDK
- AWS 適用於 Ruby V3 的 SDK

DeleteBotChannelAssociation

服務：Amazon Lex Model Building Service

刪除 Amazon Lex 機器人與訊息平台之間的關聯。

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

指向要建立此關聯的 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。

長度限制：長度下限為 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

請求格式不正確。例如，值無效或缺少必要欄位。請檢查欄位值，然後再試一次。

HTTP 狀態碼：400

ConflictException

處理請求時發生衝突。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

找不到請求中指定的資源。請檢查資源，然後再試一次。

HTTP 狀態碼：404

另請參閱

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列界面 V2
- AWS 適用於 .NET V4 的 SDK
- AWS 適用於 C++ 的 SDK
- AWS 適用於 Go 的 SDK v2
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 JavaScript V3 的 SDK
- AWS 適用於 Kotlin 的 SDK
- AWS 適用於 PHP V3 的 SDK
- AWS 適用於 Python 的 SDK
- AWS 適用於 Ruby V3 的 SDK

DeleteBotVersion

服務：Amazon Lex Model Building Service

刪除機器人的特定版本。若要刪除機器人的所有版本，請使用 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。

長度限制：長度下限為 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 DeleteBot操作。

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

請求格式不正確。例如，值無效或缺少必要欄位。請檢查欄位值，然後再試一次。

HTTP 狀態碼：400

ConflictException

處理請求時發生衝突。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

找不到請求中指定的資源。請檢查資源，然後再試一次。

HTTP 狀態碼：404

ResourceInUseException

您嘗試刪除的資源是由另一個資源所參考。使用此資訊移除您嘗試刪除之資源的參考。

例外狀況的內文包含描述資源的 JSON 物件。

```
{ "resourceType": BOT | BOTALIAS | BOTCHANNEL | INTENT,

"resourceReference": {

"name": string, "version": string } }
```

exampleReference

描述參考您嘗試刪除之資源的資源。此物件會傳回為ResourceInUseException例外狀況的一部分。

HTTP 狀態碼：400

另請參閱

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列界面 V2
- AWS 適用於 .NET V4 的 SDK
- AWS 適用於 C++ 的 SDK
- AWS 適用於 Go 的 SDK v2
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 JavaScript V3 的 SDK
- AWS 適用於 Kotlin 的 SDK
- AWS 適用於 PHP V3 的 SDK
- AWS 適用於 Python 的 SDK
- AWS 適用於 Ruby V3 的 SDK

DeleteIntent

服務：Amazon Lex Model Building Service

刪除所有版本的意圖，包括 $LATEST版本。若要刪除特定版本的意圖，請使用 DeleteIntentVersion操作。

只有在未參考時，您才能刪除意圖的版本。若要刪除一或多個機器人中參考的意圖 （ 請參閱 Amazon Lex：運作方式)，您必須先移除這些參考。

> **Note**
>
> 如果您收到ResourceInUseException例外狀況，它會提供範例參考，顯示意圖的參考位置。若要移除意圖的參考，請更新或刪除機器人。如果您在嘗試再次刪除意圖時收到相同的例外狀況，請重複 ，直到意圖沒有參考且對 的呼叫DeleteIntent成功為止。

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

請求格式不正確。例如，值無效或缺少必要欄位。請檢查欄位值，然後再試一次。

HTTP 狀態碼：400

ConflictException

處理請求時發生衝突。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

找不到請求中指定的資源。請檢查資源，然後再試一次。

HTTP 狀態碼：404

ResourceInUseException

您嘗試刪除的資源是由另一個資源所參考。使用此資訊移除您嘗試刪除之資源的參考。

例外狀況的內文包含描述資源的 JSON 物件。

```
{ "resourceType": BOT | BOTALIAS | BOTCHANNEL | INTENT,
```

```
"resourceReference": {
"name": string, "version": string } }
```

exampleReference

描述參考您嘗試刪除之資源的資源。此物件會傳回為ResourceInUseException例外狀況的一部分。

HTTP 狀態碼：400

另請參閱

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列界面 V2
- AWS 適用於 .NET V4 的 SDK
- AWS 適用於 C++ 的 SDK
- AWS 適用於 Go 的 SDK v2
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 JavaScript V3 的 SDK
- AWS 適用於 Kotlin 的 SDK
- AWS 適用於 PHP V3 的 SDK
- AWS 適用於 Python 的 SDK
- AWS 適用於 Ruby 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 DeleteIntent操作。

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

要刪除的意圖版本。您無法刪除意圖的$LATEST版本。若要刪除$LATEST版本，請使用 DeleteIntent操作。

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

請求格式不正確。例如，值無效或缺少必要欄位。請檢查欄位值，然後再試一次。

HTTP 狀態碼：400

ConflictException

處理請求時發生衝突。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

找不到請求中指定的資源。請檢查資源，然後再試一次。

HTTP 狀態碼：404

ResourceInUseException

您嘗試刪除的資源是由另一個資源所參考。使用此資訊移除您嘗試刪除之資源的參考。

例外狀況的內文包含描述資源的 JSON 物件。

```
{ "resourceType": BOT | BOTALIAS | BOTCHANNEL | INTENT,

"resourceReference": {

"name": string, "version": string } }
```

exampleReference

描述參考您嘗試刪除之資源的資源。此物件會傳回為ResourceInUseException例外狀況的一部分。

HTTP 狀態碼：400

另請參閱

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列界面 V2
- AWS 適用於 .NET V4 的 SDK
- AWS 適用於 C++ 的 SDK
- AWS 適用於 Go 的 SDK v2
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 JavaScript V3 的 SDK
- AWS 適用於 Kotlin 的 SDK
- AWS 適用於 PHP V3 的 SDK
- AWS 適用於 Python 的 SDK
- AWS 適用於 Ruby V3 的 SDK

DeleteSlotType

服務：Amazon Lex Model Building Service

刪除槽類型的所有版本，包括 $LATEST版本。若要刪除特定版本的插槽類型，請使用 DeleteSlotTypeVersion操作。

只有在未參考時，您才能刪除插槽類型的版本。若要刪除一或多個意圖中參考的槽類型，您必須先移除這些參考。

> **Note**
>
> 如果您收到ResourceInUseException例外狀況，例外狀況會提供範例參考，顯示引用槽類型的意圖。若要移除槽類型的參考，請更新意圖或刪除它。如果您在嘗試再次刪除槽類型時收到相同的例外狀況，請重複此動作，直到槽類型沒有參考且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

請求格式不正確。例如，值無效或缺少必要欄位。請檢查欄位值，然後再試一次。

HTTP 狀態碼：400

ConflictException

處理請求時發生衝突。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

找不到請求中指定的資源。請檢查資源，然後再試一次。

HTTP 狀態碼：404

ResourceInUseException

您嘗試刪除的資源是由另一個資源所參考。使用此資訊移除您嘗試刪除之資源的參考。

例外狀況的內文包含描述資源的 JSON 物件。

```
{ "resourceType": BOT | BOTALIAS | BOTCHANNEL | INTENT,
```

```
"resourceReference": {
"name": string, "version": string } }
```

exampleReference

描述參考您嘗試刪除之資源的資源。此物件會傳回為ResourceInUseException例外狀況的一部分。

HTTP 狀態碼：400

另請參閱

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列界面 V2
- AWS 適用於 .NET V4 的 SDK
- AWS 適用於 C++ 的 SDK
- AWS 適用於 Go 的 SDK v2
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 JavaScript V3 的 SDK
- AWS 適用於 Kotlin 的 SDK
- AWS 適用於 PHP V3 的 SDK
- AWS 適用於 Python 的 SDK
- AWS 適用於 Ruby 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 DeleteSlotType操作。

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

要刪除的槽類型版本。您無法刪除插槽類型的$LATEST版本。若要刪除$LATEST版本，請使用 DeleteSlotType操作。

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

請求格式不正確。例如，值無效或缺少必要欄位。請檢查欄位值，然後再試一次。

HTTP 狀態碼：400

ConflictException

處理請求時發生衝突。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

找不到請求中指定的資源。請檢查資源，然後再試一次。

HTTP 狀態碼：404

ResourceInUseException

您嘗試刪除的資源是由另一個資源所參考。使用此資訊移除您嘗試刪除之資源的參考。

例外狀況的內文包含描述資源的 JSON 物件。

```
{ "resourceType": BOT | BOTALIAS | BOTCHANNEL | INTENT,

"resourceReference": {

"name": string, "version": string } }
```

exampleReference

描述參考您嘗試刪除之資源的資源。此物件會傳回為ResourceInUseException例外狀況的一部分。

HTTP 狀態碼：400

另請參閱

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列界面 V2
- AWS 適用於 .NET V4 的 SDK
- AWS 適用於 C++ 的 SDK
- AWS 適用於 Go 的 SDK v2
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 JavaScript V3 的 SDK
- AWS 適用於 Kotlin 的 SDK
- AWS 適用於 PHP V3 的 SDK
- AWS 適用於 Python 的 SDK
- AWS 適用於 Ruby V3 的 SDK

DeleteUtterances

服務：Amazon Lex Model Building Service

刪除儲存的表達用語。

Amazon Lex 會儲存使用者傳送給機器人的表達用語。差異會儲存 15 天以供 GetUtterancesView操作使用，然後無限期儲存，以用於改善機器人回應使用者輸入的能力。

使用 DeleteUtterances操作來手動刪除特定使用者的預存表達用語。當您使用 DeleteUtterances操作時，為了改善機器人回應使用者輸入的能力而儲存的表達用語會立即刪除。儲存供 GetUtterancesView操作使用的差異會在 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

存放表達用語的機器人名稱。

長度限制：長度下限為 2。長度上限為 50。

模式：^([A-Za-z]_?)+$

必要：是

userId

表達用語之使用者的唯一識別符。這是在包含表達用語的 PostContent 或 PostText 操作請求中傳送的使用者 ID。

長度限制：長度下限為 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

請求格式不正確。例如，值無效或缺少必要欄位。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

找不到請求中指定的資源。請檢查資源，然後再試一次。

HTTP 狀態碼：404

另請參閱

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列界面 V2
- AWS 適用於 .NET V4 的 SDK
- AWS 適用於 C++ 的 SDK
- AWS 適用於 Go 的 SDK v2
- AWS 適用於 Java V2 的 SDK

- AWS 適用於 JavaScript V3 的 SDK
- AWS 適用於 Kotlin 的 SDK
- AWS 適用於 PHP V3 的 SDK
- AWS 適用於 Python 的 SDK
- AWS 適用於 Ruby 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和機器人版本或別名。

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。

長度限制：長度下限為 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$LATEST版本特定修訂的機器人總和檢查碼。

類型：字串

childDirected

對於使用 Amazon Lex Model Building Service 建立的每個 Amazon Lex 機器人，您必須指定您對 Amazon Lex 的使用是否與網站、程式或其他應用程式相關，而該網站、程式或其他應用程式全部或部分針對 13 以下兒童，並在 `false` `childDirected` 欄位中指定 `true` 或 以遵守兒童線上隱私權保護法 (COPPA)。透過`true`在 `childDirected` 欄位中指定 ，您確認使用 Amazon Lex 與網站、程式或其他應用程式相關，而該網站、程式或其他應用程式全部或部分以 13 以下兒童為目標，且受到 COPPA 的約束。透過`false`在 `childDirected` 欄位中指定 ，您確認 Amazon Lex 的使用與網站、程式或其他應用程式無關，而該網站、程式或其他應用程式全部或部分針對 13 以下且受 COPPA 約束的 兒童。您無法為 `childDirected` 欄位指定預設值，該值無法準確反映您對 Amazon Lex 的使用是否與網站、程式或其他應用程式相關，而該網站、程式或其他應用程式是針對或部分針對未滿 13 歲且受 COPPA 規範的 13 歲兒童。

如果您使用 Amazon Lex 與網站、程式或其他應用程式有關，而該網站、程式或其他應用程式全部或部分導向至未滿 13 歲的孩童，則您必須根據 COPPA 取得任何必要的可驗證父系同意。如需將 Amazon Lex 與網站、程式或其他應用程式搭配使用的相關資訊，而這些網站、程式或其他應用程式全部或部分針對未滿 13 歲的孩童，請參閱 Amazon Lex 常見問答集。

類型：布林值

clarificationPrompt

當 Amazon Lex 不了解使用者的請求時，會使用的訊息。如需詳細資訊，請參閱PutBot。

類型：Prompt 物件

createdDate

機器人建立的日期。

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表達用語傳送至 Amazon Comprehend 進行情緒分析。

類型：布林值

enableModelImprovements

指出機器人是否使用準確性改進。 `true` 指出機器人正在使用改進，否則是 `false`。

類型：布林值

failureReason

如果 `status`是 `FAILED`，Amazon Lex 會說明建置機器人失敗的原因。

類型：字串

idleSessionTTLInSeconds

Amazon Lex 保留對話中所收集資料的秒數上限。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。當您建立資源時，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

長度限制：長度下限為 2。長度上限為 50。

模式：^([A-Za-z]_?)+$

nluIntentConfidenceThreshold

在 PostContent 或 PostText 回應中傳回替代意圖時AMAZON.KendraSearchIntent，決定 Amazon Lex 插入 AMAZON.FallbackIntent、 或兩者位置的分數。如果所有意圖的可信度分數都低於此值，AMAZON.FallbackIntent則會插入 。只有在為機器人設定時，AMAZON.KendraSearchIntent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 時，正在建置機器人以供測試和使用。

如果機器人的狀態為 READY_BASIC_TESTING，您可以使用機器人意圖中指定的確切表達用語來測試機器人。當機器人準備好進行完整測試或執行時，狀態為 READY。

如果建置機器人發生問題，狀態為 ， FAILED 欄位會failureReason說明機器人未建置的原因。

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，版本一律為 $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

Amazon Lex 用來與使用者進行語音互動的 Amazon Polly 語音 ID。如需詳細資訊，請參閱PutBot。

類型：字串

錯誤

BadRequestException

請求格式不正確。例如，值無效或缺少必要欄位。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

找不到請求中指定的資源。請檢查資源，然後再試一次。

HTTP 狀態碼：404

另請參閱

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列界面 V2
- AWS 適用於 .NET V4 的 SDK
- AWS 適用於 C++ 的 SDK
- AWS 適用於 Go 的 SDK v2
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 JavaScript V3 的 SDK
- AWS 適用於 Kotlin 的 SDK
- AWS 適用於 PHP V3 的 SDK
- AWS 適用於 Python 的 SDK
- AWS 適用於 Ruby 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。

長度限制：長度下限為 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

長度限制：長度下限為 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

機器人別名的檢查總和。

類型：字串

conversationLogs

決定 Amazon Lex 如何使用別名對話日誌的設定。

類型：ConversationLogsResponse 物件

createdDate

建立機器人別名的日期。

類型：Timestamp

description

機器人別名的描述。

類型：字串

長度限制：長度下限為 0。長度上限為 200。

lastUpdatedDate

機器人別名的更新日期。當您建立資源時，建立日期和上次更新的日期相同。

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

請求格式不正確。例如，值無效或缺少必要欄位。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

找不到請求中指定的資源。請檢查資源，然後再試一次。

HTTP 狀態碼：404

另請參閱

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列界面 V2
- AWS 適用於 .NET V4 的 SDK
- AWS 適用於 C++ 的 SDK
- AWS 適用於 Go 的 SDK v2
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 JavaScript V3 的 SDK
- AWS 適用於 Kotlin 的 SDK
- AWS 適用於 PHP V3 的 SDK
- AWS 適用於 Python 的 SDK
- AWS 適用於 Ruby 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。

長度限制：長度下限為 2。長度上限為 50。

模式：`^([A-Za-z]_?)+$`

必要：是

[maxResults](#)

回應中傳回的別名數目上限。預設值為 50。

有效範圍：最小值為 1。最大值為 50。

[nameContains](#)

在機器人別名名稱中符合的子字串。如果別名名稱的任何部分符合子字串，則會傳回別名。例如，"xyz" 同時符合 "xyzabc" 和 "abcxyz"。

長度限制：長度下限為 1。長度上限為 100。

模式：`^([A-Za-z]_?)+$`

[nextToken](#)

用於擷取下一個別名頁面的分頁字符。如果此呼叫的回應被截斷，Amazon Lex 會在回應中傳回分頁字符。若要擷取下一個頁面的別名，請在下一個請求中指定分頁字符。

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物件都會描述機器人別名。

類型：BotAliasMetadata 物件陣列

nextToken

用於擷取別名下一頁的分頁字符。如果此呼叫的回應被截斷，Amazon Lex 會在回應中傳回分頁字符。若要擷取下一個頁面的別名，請在下一個請求中指定分頁字符。

類型：字串

錯誤

BadRequestException

請求格式不正確。例如，值無效或缺少必要欄位。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

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列界面 V2
- AWS 適用於 .NET V4 的 SDK
- AWS 適用於 C++ 的 SDK
- AWS 適用於 Go 的 SDK v2
- AWS 適用於 Java V2 的 SDK

- AWS 適用於 JavaScript V3 的 SDK
- AWS 適用於 Kotlin 的 SDK
- AWS 適用於 PHP V3 的 SDK
- AWS 適用於 Python 的 SDK
- AWS 適用於 Ruby V3 的 SDK

GetBotChannelAssociation

服務：Amazon Lex Model Building Service

傳回 Amazon Lex 機器人與訊息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

指向要建立此關聯之 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。

長度限制：長度下限為 2。長度上限為 50。

模式：`^([A-Za-z]_?)+$`

必要：是

name

機器人與頻道之間的關聯名稱。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

指向要建立此關聯之 Amazon Lex 機器人特定版本的別名。

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

映射項目：最多 10 個項目。

botName

Amazon Lex 機器人的名稱。

類型：字串

長度限制：長度下限為 2。長度上限為 50。

模式：^([A-Za-z]_?)+$

createdDate

機器人與頻道之間的關聯建立日期。

類型：Timestamp

description

機器人與頻道之間的關聯描述。

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

機器人與頻道之間的關聯名稱。

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

機器人頻道的狀態。

- CREATED - 頻道已建立並可供使用。
- IN_PROGRESS - 頻道建立進行中。

- 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

請求格式不正確。例如，值無效或缺少必要欄位。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

找不到請求中指定的資源。請檢查資源，然後再試一次。

HTTP 狀態碼：404

另請參閱

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列界面 V2

- AWS 適用於 .NET V4 的 SDK
- AWS 適用於 C++ 的 SDK
- AWS 適用於 Go 的 SDK v2
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 JavaScript V3 的 SDK
- AWS 適用於 Kotlin 的 SDK
- AWS 適用於 PHP V3 的 SDK
- AWS 適用於 Python 的 SDK
- AWS 適用於 Ruby 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頻道清單。

`GetBotChannelAssociations` 操作需要 `lex:GetBotChannelAssociations`動作的許可。

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

指向要建立此關聯之 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

關聯中 Amazon Lex 機器人的名稱。

長度限制：長度下限為 2。長度上限為 50。

模式：`^([A-Za-z]_?)+$`

必要：是

maxResults

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

在頻道關聯名稱中符合的子字串。如果名稱的任何部分符合子字串，則會傳回關聯。例如，"xyz" 同時符合 "xyzabc" 和 "abcxyz"。若要傳回所有機器人頻道關聯，請使用連字號 ("-") 做為 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

用於擷取下一頁關聯的分頁字符。如果此呼叫的回應被截斷，Amazon Lex 會在回應中傳回分頁字符。若要擷取下一頁的關聯，請在下一個請求中指定分頁字符。

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，可提供 Amazon Lex 機器人及其與頻道之關聯的相關資訊。

類型：BotChannelAssociation 物件陣列

nextToken

擷取下一頁關聯的分頁字符。如果此呼叫的回應被截斷，Amazon Lex 會在回應中傳回分頁字符。若要擷取下一頁的關聯，請在下一個請求中指定分頁字符。

類型：字串

錯誤

BadRequestException

請求格式不正確。例如，值無效或缺少必要欄位。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

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列界面 V2

- AWS 適用於 .NET V4 的 SDK
- AWS 適用於 C++ 的 SDK
- AWS 適用於 Go 的 SDK v2
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 JavaScript V3 的 SDK
- AWS 適用於 Kotlin 的 SDK
- AWS 適用於 PHP V3 的 SDK
- AWS 適用於 Python 的 SDK
- AWS 適用於 Ruby 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$LATEST版本的資訊。
- 如果您未指定 nameContains 欄位，操作會傳回所有機器人$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

在請求將傳回的回應中傳回的機器人數量上限。預設為 10。

有效範圍：最小值為 1。最大值為 50。

nameContains

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符合子字串，則會傳回機器人。例如，"xyz" 同時符合 "xyzabc" 和 "abcxyz"。

長度限制：長度下限為 2。長度上限為 50。

模式：^([A-Za-z]_?)+$

nextToken

擷取機器人下一頁的分頁字符。如果此呼叫的回應被截斷，Amazon Lex 會在回應中傳回分頁字符。若要擷取機器人的下一頁，請在下一個請求中指定分頁字符。

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

物件陣列botMetadata，每個機器人各有一個項目。

類型：BotMetadata 物件陣列

nextToken

如果回應被截斷，則會包含分頁字符，您可以在下一個請求中指定以擷取機器人的下一頁。

類型：字串

錯誤

BadRequestException

請求格式不正確。例如，值無效或缺少必要欄位。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

找不到請求中指定的資源。請檢查資源，然後再試一次。

HTTP 狀態碼：404

另請參閱

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列界面 V2
- AWS 適用於 .NET V4 的 SDK
- AWS 適用於 C++ 的 SDK
- AWS 適用於 Go 的 SDK v2
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 JavaScript V3 的 SDK
- AWS 適用於 Kotlin 的 SDK
- AWS 適用於 PHP V3 的 SDK
- AWS 適用於 Python 的 SDK
- AWS 適用於 Ruby V3 的 SDK

GetBotVersions

服務：Amazon Lex Model Building Service

取得機器人所有版本的相關資訊。

GetBotVersions 操作會傳回每個版本的機器人的BotMetadata物件。例如，如果機器人有三個編號版本，GetBotVersions操作會在回應中傳回四個BotMetadata物件，每個編號版本各一個，版本各一個$LATEST。

GetBotVersions 操作一律會傳回至少一個版本，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

在回應中傳回的機器人版本數目上限。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。

長度限制：長度下限為 2。長度上限為 50。

模式：^([A-Za-z]_?)+$

必要：是

nextToken

用於擷取下一頁機器人版本的分頁字符。如果此呼叫的回應被截斷，Amazon Lex 會在回應中傳回分頁字符。若要擷取下一頁的版本，請在下一個請求中指定分頁字符。

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

物件陣列BotMetadata，每個編號版本的機器人各一個，以及$LATEST版本一個。

類型：BotMetadata 物件陣列

nextToken

用於擷取下一頁機器人版本的分頁字符。如果此呼叫的回應被截斷，Amazon Lex 會在回應中傳回分頁字符。若要擷取下一頁的版本，請在下一個請求中指定分頁字符。

類型：字串

錯誤

BadRequestException

請求格式不正確。例如，值無效或缺少必要欄位。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

找不到請求中指定的資源。請檢查資源，然後再試一次。

HTTP 狀態碼：404

另請參閱

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列界面 V2
- AWS 適用於 .NET V4 的 SDK
- AWS 適用於 C++ 的 SDK
- AWS 適用於 Go 的 SDK v2
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 JavaScript V3 的 SDK
- AWS 適用於 Kotlin 的 SDK
- AWS 適用於 PHP V3 的 SDK
- AWS 適用於 Python 的 SDK
- AWS 適用於 Ruby 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

內建意圖的唯一識別符。若要尋找意圖的簽章，請參閱 Alexa Skills Kit 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

內建意圖的唯一識別符。

類型：字串

slots

物件陣列BuiltinIntentSlot，意圖中每個槽類型的一個項目。

類型：BuiltinIntentSlot 物件陣列

supportedLocales

意圖支援的地區設定清單。

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

請求格式不正確。例如，值無效或缺少必要欄位。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

找不到請求中指定的資源。請檢查資源，然後再試一次。

HTTP 狀態碼：404

另請參閱

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列界面 V2
- AWS 適用於 .NET V4 的 SDK
- AWS 適用於 C++ 的 SDK
- AWS 適用於 Go 的 SDK v2
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 JavaScript V3 的 SDK
- AWS 適用於 Kotlin 的 SDK
- AWS 適用於 PHP V3 的 SDK
- AWS 適用於 Python 的 SDK
- AWS 適用於 Ruby 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

意圖支援的地區設定清單。

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

擷取下一頁意圖的分頁字符。如果此 API 呼叫被截斷，Amazon Lex 會在回應中傳回分頁字符。若要擷取下一頁的意圖，請在下一個請求中使用分頁字符。

signatureContains

符合內建意圖簽章的子字串。如果其簽章的任何部分符合子字串，則會傳回意圖。例如，"xyz" 同時符合 "xyzabc" 和 "abcxyz"。若要尋找意圖的簽章，請參閱 Alexa Skills Kit 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

物件陣列builtinIntentMetadata，回應中的每個意圖各一個。

類型：BuiltinIntentMetadata 物件陣列

nextToken

擷取下一頁意圖的分頁字符。如果此 API 呼叫的回應遭到截斷，Amazon Lex 會在回應中傳回分頁字符。若要擷取下一頁的意圖，請在下一個請求中指定分頁字符。

類型：字串

錯誤

BadRequestException

請求格式不正確。例如，值無效或缺少必要欄位。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

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列界面 V2
- AWS 適用於 .NET V4 的 SDK
- AWS 適用於 C++ 的 SDK
- AWS 適用於 Go 的 SDK v2
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 JavaScript V3 的 SDK
- AWS 適用於 Kotlin 的 SDK
- AWS 適用於 PHP V3 的 SDK
- AWS 適用於 Python 的 SDK
- AWS 適用於 Ruby 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 Skills Kit 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

插槽類型支援的地區設定清單。

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

回應中傳回的槽類型數目上限。預設為 10。

有效範圍：最小值為 1。最大值為 50。

nextToken

擷取下頁槽類型的分頁字符。如果此 API 呼叫的回應遭到截斷，Amazon Lex 會在回應中傳回分頁字符。若要擷取下頁的槽類型，請在下一個請求中指定分頁字符。

signatureContains

要符合內建槽類型簽章的子字串。如果其簽章的任何部分符合子字串，則會傳回槽類型。例如，"xyz" 同時符合 "xyzabc" 和 "abcxyz"。

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

如果回應被截斷，則回應包含一個分頁字符，您可以在下一個請求中使用它來擷取槽類型的下一頁。

類型：字串

slotTypes

物件陣列BuiltInSlotTypeMetadata，傳回每個槽類型一個項目。

類型：BuiltinSlotTypeMetadata 物件陣列

錯誤

BadRequestException

請求格式不正確。例如，值無效或缺少必要欄位。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

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列界面 V2
- AWS 適用於 .NET V4 的 SDK
- AWS 適用於 C++ 的 SDK
- AWS 適用於 Go 的 SDK v2
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 JavaScript V3 的 SDK
- AWS 適用於 Kotlin 的 SDK
- AWS 適用於 PHP V3 的 SDK
- AWS 適用於 Python 的 SDK
- AWS 適用於 Ruby 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 - 匯出已完成。
- FAILED - 無法完成匯出。

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

提供匯出資源位置的 S3 預先簽章 URL。匯出的資源是 ZIP 封存檔，其中包含 JSON 格式的匯出資源。封存的結構可能會變更。您的程式碼不應依賴封存結構。

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

請求格式不正確。例如，值無效或缺少必要欄位。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

找不到請求中指定的資源。請檢查資源，然後再試一次。

HTTP 狀態碼：404

另請參閱

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列界面 V2
- AWS 適用於 .NET V4 的 SDK
- AWS 適用於 C++ 的 SDK
- AWS 適用於 Go 的 SDK v2
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 JavaScript V3 的 SDK
- AWS 適用於 Kotlin 的 SDK
- AWS 適用於 PHP V3 的 SDK
- AWS 適用於 Python 的 SDK

- AWS 適用於 Ruby V3 的 SDK

GetImport

服務：Amazon Lex Model Building Service

取得從 `StartImport`操作開始之匯入任務的相關資訊。

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

要傳回之匯入任務資訊的識別符。

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

建立匯入任務的日期和時間的時間戳記。

類型：Timestamp

failureReason

描述匯入任務無法完成原因的字串。

類型：字串陣列

importId

特定匯入任務的識別符。

類型：字串

importStatus

匯入任務的狀態。如果狀態為 FAILED，您可以從 failureReason 欄位取得失敗的原因。

類型：字串

有效值:IN_PROGRESS | COMPLETE | FAILED

mergeStrategy

當現有資源與匯入檔案中的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

提供給匯入任務的名稱。

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

請求格式不正確。例如，值無效或缺少必要欄位。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

找不到請求中指定的資源。請檢查資源，然後再試一次。

HTTP 狀態碼：404

另請參閱

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列界面 V2
- AWS 適用於 .NET V4 的 SDK
- AWS 適用於 C++ 的 SDK
- AWS 適用於 Go 的 SDK v2
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 JavaScript V3 的 SDK
- AWS 適用於 Kotlin 的 SDK

- AWS 適用於 PHP V3 的 SDK
- AWS 適用於 Python 的 SDK
- AWS 適用於 Ruby V3 的 SDK

GetIntent

服務：Amazon Lex Model Building Service

傳回意圖的相關資訊。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

意圖的檢查總和。

類型：字串

conclusionStatement

在 fulfillmentActivity元素中指定的 Lambda 函數滿足意圖後，Amazon Lex 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 會使用提示來確認意圖，然後再滿足使用者的請求。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 會為每個使用者輸入調用此 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 會使用此提示在意圖實現之後請求額外的使用者活動。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

物件陣列InputContext，列出 Amazon Lex 在與使用者的對話中必須處於作用中狀態的內容，以選擇意圖。

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

若有組態資訊，請使用 AMAZON.KendraSearchIntent意圖連線至 Amazon Kendra 索引。

類型：KendraConfiguration 物件

lastUpdatedDate

意圖更新的日期。當您建立資源時，建立日期和上次更新的日期相同。

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

物件陣列OutputContext，列出意圖在滿足意圖時啟用的內容。

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符。

類型：字串

rejectionStatement

如果使用者對 中定義的問題回答「否」confirmationPrompt，Amazon Lex 會回應此陳述式，以確認意圖已取消。

類型：Statement 物件

sampleUtterances

為意圖設定的範例表達用語陣列。

類型：字串陣列

陣列成員：項目數下限為 0。最多 1500 個項目。

長度限制：長度下限為 1。長度上限為 200。

slots

為意圖設定的意圖插槽陣列。

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

請求格式不正確。例如，值無效或缺少必要欄位。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

找不到請求中指定的資源。請檢查資源，然後再試一次。

HTTP 狀態碼：404

另請參閱

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列界面 V2
- AWS 適用於 .NET V4 的 SDK
- AWS 適用於 C++ 的 SDK
- AWS 適用於 Go 的 SDK v2
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 JavaScript V3 的 SDK
- AWS 適用於 Kotlin 的 SDK
- AWS 適用於 PHP V3 的 SDK
- AWS 適用於 Python 的 SDK
- AWS 適用於 Ruby V3 的 SDK

GetIntents

服務：Amazon Lex Model Building Service

傳回意圖資訊，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。

操作需要 lex:GetIntents動作的許可。

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

要符合意圖名稱的子字串。如果其名稱的任何部分符合子字串，則會傳回意圖。例如，"xyz" 同時符合 "xyzabc" 和 "abcxyz"。

長度限制：長度下限為 1。長度上限為 100。

模式：^([A-Za-z]_?)+$

nextToken

擷取下一頁意圖的分頁字符。如果此 API 呼叫的回應遭到截斷，Amazon Lex 會在回應中傳回分頁字符。若要擷取下一頁的意圖，請在下一個請求中指定分頁字符。

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

如果回應被截斷，則回應會包含分頁字符，您可以在下一個請求中指定以擷取下一頁的意圖。

類型：字串

錯誤

BadRequestException

請求格式不正確。例如，值無效或缺少必要欄位。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

找不到請求中指定的資源。請檢查資源，然後再試一次。

HTTP 狀態碼：404

另請參閱

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列界面 V2
- AWS 適用於 .NET V4 的 SDK
- AWS 適用於 C++ 的 SDK
- AWS 適用於 Go 的 SDK v2
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 JavaScript V3 的 SDK
- AWS 適用於 Kotlin 的 SDK
- AWS 適用於 PHP V3 的 SDK
- AWS 適用於 Python 的 SDK
- AWS 適用於 Ruby V3 的 SDK

GetIntentVersions

服務：Amazon Lex Model Building Service

取得所有 意圖版本的相關資訊。

GetIntentVersions 操作會傳回每個版本意圖的 IntentMetadata 物件。例如，如果意圖有三個編號版本，GetIntentVersions操作會在回應中傳回四個IntentMetadata物件，每個編號版本各一個，版本各一個$LATEST。

GetIntentVersions 操作一律會傳回至少一個版本，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

應傳回版本之意圖的名稱。

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

用於擷取意圖版本的下一頁的分頁字符。如果此呼叫的回應被截斷，Amazon Lex 會在回應中傳回分頁字符。若要擷取下一頁的版本，請在下一個請求中指定分頁字符。

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

物件陣列IntentMetadata，每個編號版本的意圖各一個，加上$LATEST版本一個。

類型：IntentMetadata 物件陣列

nextToken

用於擷取意圖版本的下一頁的分頁字符。如果此呼叫的回應被截斷，Amazon Lex 會在回應中傳回分頁字符。若要擷取下一頁的版本，請在下一個請求中指定分頁字符。

類型：字串

錯誤

BadRequestException

請求格式不正確。例如，值無效或缺少必要欄位。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

找不到請求中指定的資源。請檢查資源，然後再試一次。

HTTP 狀態碼：404

另請參閱

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列界面 V2
- AWS 適用於 .NET V4 的 SDK
- AWS 適用於 C++ 的 SDK
- AWS 適用於 Go 的 SDK v2
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 JavaScript V3 的 SDK
- AWS 適用於 Kotlin 的 SDK
- AWS 適用於 PHP V3 的 SDK
- AWS 適用於 Python 的 SDK
- AWS 適用於 Ruby V3 的 SDK

GetMigration

服務：Amazon Lex Model Building Service

提供持續或完整從 Amazon Lex V1 機器人遷移至 Amazon Lex V2 機器人的詳細資訊。使用此操作來檢視與遷移相關的遷移提醒和警告。

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

要檢視之遷移的唯一識別符。StartMigration 操作`migrationID`會傳回 。

長度限制：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

提醒和警告清單，指出 Amazon Lex V1 機器人遷移至 Amazon Lex V2 時發生問題。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

遷移的唯一識別符。這與呼叫 `GetMigration`操作時所使用的識別符相同。

類型：字串

長度限制：固定長度為 10。

模式：`^[0-9a-zA-Z]+$`

migrationStatus

指出遷移的狀態。當狀態為遷移完成`COMPLETE`且機器人可在 Amazon Lex V2 中使用時。可能需要解決提醒和警告，才能完成遷移。

類型：字串

有效值:`IN_PROGRESS | COMPLETED | FAILED`

migrationStrategy

用來執行遷移的策略。

- CREATE_NEW - 建立新的 Amazon Lex V2 機器人，並將 Amazon Lex V1 機器人遷移至新的機器人。
- 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地區設定。如果地區設定不存在，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

遷移開始的日期和時間。

類型：Timestamp

v1BotLocale

Amazon Lex V1 機器人的地區設定已遷移至 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

遷移至 Amazon Lex V2 的 Amazon Lex V1 機器人名稱。 Amazon Lex V2

類型：字串

長度限制：長度下限為 2。長度上限為 50。

模式：^([A-Za-z]_?)+$

v1BotVersion

Amazon Lex V1 機器人的版本已遷移至 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

要遷移至 Amazon Lex V1 之 Amazon Lex V2 機器人的唯一識別符。 Amazon Lex V1

類型：字串

長度限制：固定長度為 10。

模式：^[0-9a-zA-Z]+$

v2BotRole

Amazon Lex 用來執行 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

請求格式不正確。例如，值無效或缺少必要欄位。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

找不到請求中指定的資源。請檢查資源，然後再試一次。

HTTP 狀態碼：404

另請參閱

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列界面 V2
- AWS 適用於 .NET V4 的 SDK
- AWS 適用於 C++ 的 SDK
- AWS 適用於 Go 的 SDK v2
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 JavaScript V3 的 SDK
- AWS 適用於 Kotlin 的 SDK
- AWS 適用於 PHP V3 的 SDK
- AWS 適用於 Python 的 SDK
- AWS 適用於 Ruby V3 的 SDK

GetMigrations

服務：Amazon Lex Model Building Service

取得 Amazon Lex V1 和 Amazon Lex V2 之間的遷移清單。

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

回應中傳回的遷移數目上限。預設為 10。

有效範圍：最小值為 1。最大值為 50。

migrationStatusEquals

篩選清單，使其僅包含指定狀態的遷移。

有效值:IN_PROGRESS | COMPLETED | FAILED

nextToken

擷取下一頁遷移的分頁字符。如果對此操作的回應被截斷，Amazon Lex 會在回應中傳回分頁字符。若要擷取下一頁的遷移，請在請求中指定分頁字符。

sortByAttribute

要排序遷移清單的欄位。您可以依 Amazon Lex V1 機器人名稱或開始遷移的日期和時間進行排序。

有效值:V1_BOT_NAME | MIGRATION_DATE_TIME

sortByOrder

因此排序清單的順序。

有效值:ASCENDING | DESCENDING

v1BotNameContains

篩選清單，使其僅包含名稱包含指定字串的機器人。字串會與機器人名稱中的任何位置相符。

長度限制：長度下限為 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 Amazon Lex V1 遷移至 Amazon Lex V2 的摘要陣列。若要查看遷移的詳細資訊，請在呼叫 GetMigration操作時使用migrationId摘要中的 。

類型：MigrationSummary 物件陣列

nextToken

如果回應被截斷，則包含分頁字符，您可以在下一個請求中指定以擷取下一頁遷移。

類型：字串

錯誤

BadRequestException

請求格式不正確。例如，值無效或缺少必要欄位。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

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列界面 V2
- AWS 適用於 .NET V4 的 SDK
- AWS 適用於 C++ 的 SDK
- AWS 適用於 Go 的 SDK v2
- AWS 適用於 Java V2 的 SDK

- AWS 適用於 JavaScript V3 的 SDK
- AWS 適用於 Kotlin 的 SDK
- AWS 適用於 PHP V3 的 SDK
- AWS 適用於 Python 的 SDK
- AWS 適用於 Ruby 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。除了指定槽類型名稱之外，您還必須指定槽類型版本。

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

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

槽類型的$LATEST版本總和檢查碼。

類型：字串

createdDate

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

定義槽類型可採用之值的EnumerationValue物件清單。

類型：EnumerationValue 物件陣列

陣列成員：項目數下限為 0。10000 個項目的數量上限。

lastUpdatedDate

槽類型更新的日期。當您建立資源時，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

做為插槽類型父項的內建插槽類型。

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

延伸父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 用來判斷槽值的策略。如需詳細資訊，請參閱PutSlotType。

類型：字串

有效值:ORIGINAL_VALUE | TOP_RESOLUTION

version

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

請求格式不正確。例如，值無效或缺少必要欄位。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

找不到請求中指定的資源。請檢查資源，然後再試一次。

HTTP 狀態碼：404

另請參閱

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列界面 V2

- AWS 適用於 .NET V4 的 SDK
- AWS 適用於 C++ 的 SDK
- AWS 適用於 Go 的 SDK v2
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 JavaScript V3 的 SDK
- AWS 適用於 Kotlin 的 SDK
- AWS 適用於 PHP V3 的 SDK
- AWS 適用於 Python 的 SDK
- AWS 適用於 Ruby V3 的 SDK

GetSlotTypes

服務：Amazon Lex Model Building Service

傳回槽類型資訊，如下所示：

- 如果您指定 nameContains 欄位， 會傳回包含指定字串之所有槽類型的$LATEST版本。
- 如果您未指定 nameContains 欄位， 會傳回所有槽類型$LATEST版本的相關資訊。

操作需要 lex:GetSlotTypes動作的許可。

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

回應中傳回的槽類型數目上限。預設為 10。

有效範圍：最小值為 1。最大值為 50。

nameContains

在槽類型名稱中符合的子字串。如果名稱的任何部分符合子字串，則會傳回槽類型。例如，"xyz" 同時符合 "xyzabc" 和 "abcxyz"。

長度限制：長度下限為 1。長度上限為 100。

模式：^([A-Za-z]_?)+$

nextToken

擷取下頁槽類型的分頁字符。如果此 API 呼叫的回應遭到截斷，Amazon Lex 會在回應中傳回分頁字符。若要擷取下頁的槽類型，請在下一個請求中指定分頁字符。

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

如果回應被截斷，它包含一個分頁字符，您可以在下一個請求中指定，以擷取槽類型的下一頁。

類型：字串

slotTypes

物件陣列，每個插槽類型各一個，提供插槽類型名稱、版本和描述等資訊。

類型：SlotTypeMetadata 物件陣列

錯誤

BadRequestException

請求格式不正確。例如，值無效或缺少必要欄位。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

找不到請求中指定的資源。請檢查資源，然後再試一次。

HTTP 狀態碼：404

另請參閱

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列界面 V2
- AWS 適用於 .NET V4 的 SDK
- AWS 適用於 C++ 的 SDK
- AWS 適用於 Go 的 SDK v2
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 JavaScript V3 的 SDK
- AWS 適用於 Kotlin 的 SDK
- AWS 適用於 PHP V3 的 SDK
- AWS 適用於 Python 的 SDK
- AWS 適用於 Ruby V3 的 SDK

GetSlotTypeVersions

服務：Amazon Lex Model Building Service

取得槽類型所有版本的相關資訊。

GetSlotTypeVersions 操作會針對每個版本的槽類型傳回SlotTypeMetadata物件。例如，如果槽類型有三個編號版本，GetSlotTypeVersions操作會在回應中傳回四個SlotTypeMetadata物件，每個編號版本各一個，版本各一個$LATEST。

GetSlotTypeVersions 操作一律會傳回至少一個版本，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

回應中傳回的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

應傳回版本的槽類型名稱。

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

用於擷取下頁槽類型版本的分頁字符。如果此呼叫的回應被截斷，Amazon Lex 會在回應中傳回分頁字符。若要擷取下一頁的版本，請在下一個請求中指定分頁字符。

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

用於擷取下頁槽類型版本的分頁字符。如果此呼叫的回應被截斷，Amazon Lex 會在回應中傳回分頁字符。若要擷取下一頁的版本，請在下一個請求中指定分頁字符。

類型：字串

slotTypes

物件陣列SlotTypeMetadata，每個編號版本的插槽類型各一個，加上$LATEST版本一個。

類型：SlotTypeMetadata 物件陣列

錯誤

BadRequestException

請求格式不正確。例如，值無效或缺少必要欄位。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

找不到請求中指定的資源。請檢查資源，然後再試一次。

HTTP 狀態碼：404

另請參閱

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列界面 V2
- AWS 適用於 .NET V4 的 SDK
- AWS 適用於 C++ 的 SDK
- AWS 適用於 Go 的 SDK v2
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 JavaScript V3 的 SDK
- AWS 適用於 Kotlin 的 SDK
- AWS 適用於 PHP V3 的 SDK
- AWS 適用於 Python 的 SDK
- AWS 適用於 Ruby V3 的 SDK

GetUtterancesView

服務：Amazon Lex Model Building Service

使用 GetUtterancesView操作來取得有關使用者對機器人表達之表達用語的資訊。您可以使用此清單來調校機器人回應的表達用語。

例如，假設您已建立機器人來訂購花。您的使用者使用機器人一段時間後，請使用 GetUtterancesView操作來查看他們提出的請求，以及他們是否成功。您可能會發現無法辨識表達用語「我想要花朵」。您可以將此表達用語新增至OrderFlowers意圖，讓您的機器人辨識該表達用語。

發佈新版本的機器人之後，您可以取得舊版本和新版本的相關資訊，以便比較兩個版本的效能。

表達用語統計資料為每天產生一次。資料在過去 15 天內可供使用。您可以在每個請求中請求最多 5 個版本的機器人資訊。Amazon Lex 會傳回機器人在過去 15 天內最常收到的表達用語。回應包含每個版本最多 100 個表達用語的相關資訊。

在下列情況下，不會產生張量統計資料：

- 建立機器人時， childDirected 欄位設定為 true。
- 您正在將插槽混淆與一或多個插槽搭配使用。
- 您選擇不參與改善 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

應傳回表達用語資訊的機器人名稱。

長度限制：長度下限為 2。長度上限為 50。

模式：^([A-Za-z]_?)+$

必要：是

botVersions

應傳回表達用語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

若要傳回辨識和處理的表達用語，請使用 Detected。若要傳回無法辨識的表達用語，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

傳回表達用語資訊的機器人名稱。

類型：字串

長度限制：長度下限為 2。長度上限為 50。

模式：`^([A-Za-z]_?)+$`

utterances

物件陣列，每個UtteranceList物件都包含描述機器人處理之表達用語的UtteranceData物件清單。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 天內最常收到的表達用語。

類型：UtteranceList 物件陣列

錯誤

BadRequestException

請求格式不正確。例如，值無效或缺少必要欄位。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

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列界面 V2
- AWS 適用於 .NET V4 的 SDK
- AWS 適用於 C++ 的 SDK
- AWS 適用於 Go 的 SDK v2
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 JavaScript V3 的 SDK
- AWS 適用於 Kotlin 的 SDK
- AWS 適用於 PHP V3 的 SDK
- AWS 適用於 Python 的 SDK
- AWS 適用於 Ruby V3 的 SDK

ListTagsForResource

服務：Amazon Lex Model Building Service

取得與指定資源相關聯的標籤清單。只有機器人、機器人別名和機器人管道可以有與其相關聯的標籤。

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

要取得標籤清單之資源的 Amazon Resource Name (ARN)。

長度限制：長度下限為 1。長度上限為 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

請求格式不正確。例如，值無效或缺少必要欄位。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

找不到請求中指定的資源。請檢查資源，然後再試一次。

HTTP 狀態碼：404

另請參閱

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列界面 V2
- AWS 適用於 .NET V4 的 SDK
- AWS 適用於 C++ 的 SDK

- AWS 適用於 Go 的 SDK v2
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 JavaScript V3 的 SDK
- AWS 適用於 Kotlin 的 SDK
- AWS 適用於 PHP V3 的 SDK
- AWS 適用於 Python 的 SDK
- AWS 適用於 Ruby V3 的 SDK

PutBot

服務：Amazon Lex Model Building Service

建立 Amazon Lex 對話式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，以及機器人是否導向 13 以下兒童。您可以使用它來稍後新增意圖，或從現有機器人中移除意圖。當您建立具有最少資訊的機器人時，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置機器人。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版本的現有值。Amazon Lex 會移除您在請求中未提供 值的任何欄位，除了設定為其預設值的 idleTTLInSeconds和 privacySettings 欄位。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。

長度限制：長度下限為 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 無法了解使用者在內容中的輸入時，會嘗試多次引出資訊。之後，Amazon Lex 會將 中定義的訊息abortStatement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，在比薩訂購機器人中，Amazon Lex 可能會詢問使用者「您想要哪種類型的地殼？」 如果使用者的回應不是預期的回應之一 （例如，「薄餅皮、「深盤」等），Amazon Lex 會嘗試再次引出正確的回應。

例如，在比薩訂購應用程式中， OrderPizza 可能是其中一個意圖。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。

如果您已定義備用意圖，則不會將取消陳述式傳送給使用者，而是改用備用意圖。如需詳細資訊，請參閱 AMAZON.FallbackIntent。

類型：Statement 物件

必要：否

checksum

識別 $LATEST版本的特定修訂。

當您建立新的機器人時，請將 checksum 欄位保留空白。如果您指定檢查總和，您會收到BadRequestException例外狀況。

當您想要更新機器人時，請將 checksum 欄位設定為最新版本的檢查總和$LATEST。如果您未指定 checksum 欄位，或檢查總和與$LATEST版本不相符，您會收到PreconditionFailedException例外狀況。

類型：字串

必要：否

childDirected

對於使用 Amazon Lex Model Building Service 建立的每個 Amazon Lex 機器人，您必須指定您對 Amazon Lex 的使用是否與網站、程式或其他應用程式相關，而該網站、程式或其他應用程式全部或部分針對 13 以下兒童，並在 false childDirected 欄位中指定 true 或 以遵守兒童線上隱私權保護法 (COPPA)。透過true在 childDirected 欄位中指定 ，您確認使用 Amazon Lex 與網站、程式或其他應用程式相關，而該網站、程式或其他應用程式全部或部分以 13 以下兒童

為目標，且受到 COPPA 的約束。透過`false`在 `childDirected` 欄位中指定 ，您確認 Amazon Lex 的使用與網站、程式或其他應用程式無關，而該網站、程式或其他應用程式全部或部分針對 13 以下且受 COPPA 約束的 兒童。您無法為 `childDirected` 欄位指定預設值，該值無法準確反映您對 Amazon Lex 的使用是否與網站、程式或其他應用程式相關，而該網站、程式或其他應用程式是針對或部分針對未滿 13 歲且受 COPPA 規範的 13 歲兒童。

如果您使用 Amazon Lex 與網站、程式或其他應用程式有關，而該網站、程式或其他應用程式全部或部分導向至未滿 13 歲的孩童，則您必須根據 COPPA 取得任何必要的可驗證父系同意。如需將 Amazon Lex 與網站、程式或其他應用程式搭配使用的相關資訊，而這些網站、程式或其他應用程式全部或部分針對未滿 13 歲的孩童，請參閱 Amazon Lex 常見問答集。

類型：布林值

必要：是

clarificationPrompt

當 Amazon Lex 不了解使用者的意圖時，它會使用此訊息來釐清。若要指定 Amazon Lex 應該重複澄清提示的次數，請使用 `maxAttempts` 欄位。如果 Amazon Lex 仍然不了解，則會在 `abortStatement` 欄位中傳送訊息。

當您建立釐清提示時，請確定它建議使用者的正確回應。例如，對於訂購比薩和飲料的機器人，您可以建立此釐清提示：「您想要做什麼？ 您可以說「訂購比薩」或「訂購飲料」。

如果您已定義備用意圖，如果重複說明提示在`maxAttempts`欄位中定義的次數，則會叫用它。如需詳細資訊，請參閱 AMAZON.FallbackIntent。

如果您未定義釐清提示，Amazon Lex 會在執行時間傳回 400 錯誤的請求例外狀況，在三種情況下：

- 後續提示 - 當使用者回應後續提示，但未提供意圖時。例如，為了回應顯示「您今天是否要其他項目？」的後續提示 使用者說「是」。Amazon Lex 會傳回 400 錯誤的請求例外狀況，因為它沒有要傳送給使用者的釐清提示以取得意圖。
- Lambda 函數 - 使用 Lambda 函數時，您會傳回`ElicitIntent`對話方塊類型。由於 Amazon Lex 沒有向使用者取得意圖的釐清提示，因此會傳回 400 錯誤的請求例外狀況。
- PutSession 操作 - 使用 `PutSession`操作時，您會傳送`ElicitIntent`對話方塊類型。由於 Amazon Lex 沒有向使用者取得意圖的釐清提示，因此會傳回 400 錯誤的請求例外狀況。

類型：Prompt 物件

必要：否

createVersion

當設定為true新編號版本的機器人時，會建立。這與呼叫 CreateBotVersion操作相同。如果您未指定 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 時，使用者表達用語會傳送至 Amazon Comprehend 進行情緒分析。如果您未指定 detectSentiment，則預設值為 false。

類型：布林值

必要：否

enableModelImprovements

設定為 true 以允許存取自然語言理解改進。

當您將 enableModelImprovements 參數設定為 true時，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準確性改進。

對於 en-US 地區設定，您可以將 enableModelImprovements 參數設定為 false 的區域為：

- 美國東部 (維吉尼亞北部) (us-east-1)
- 美國西部 (奧勒岡) (us-west-2)
- 亞太地區 (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`true`預設為 。在這些區域和地區設定 參數以`false`擲回`ValidationException`例外狀況。

類型：布林值

必要：否

idleSessionTTLInSeconds

Amazon Lex 保留對話中所收集資料的秒數上限。

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，假設使用者選擇 OrderPizza 意圖，但透過下訂單中途取得附屬項目。如果使用者未在指定的時間內完成訂單，Amazon Lex 會捨棄其收集的槽資訊，而且使用者必須重新開始。

如果您未在`PutBot`操作請求中包含 `idleSessionTTLInSeconds`元素，Amazon Lex 會使用預設值。如果請求取代現有的機器人，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。每個意圖代表使用者可以表達的命令。例如，比薩訂購機器人可能支援 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地區設定相容。

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

決定在 PostContent 或 PostText 回應中傳回替代意圖時AMAZON.FallbackIntent，Amazon Lex 將插入 AMAZON.KendraSearchIntent、 或兩者的閾值。 AMAZON.FallbackIntent和 AMAZON.KendraSearchIntent 只有在為機器人設定時才插入。

您必須將 enableModelImprovements 參數設定為 true，才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)
- 亞太地區 (雪梨) (ap-southeast-2)
- 歐洲 (愛爾蘭) (eu-west-1)

在其他區域中， enableModelImprovements 參數true預設為 。

例如，假設機器人的可信度閾值設定為 0.80 和 AMAZON.FallbackIntent。Amazon Lex 傳回具有下列可信度分數的三種替代意圖：IntentA (0.70)、IntentB (0.60)、IntentC (0.50)。PostText 操作的回應將是：

- AMAZON.FallbackIntent
- IntentA
- IntentB
- IntentC

類型：Double

有效範圍：最小值為 0。最大值為 1。

必要：否

processBehavior

如果您將 processBehavior元素設定為 BUILD，Amazon Lex 會建置機器人以執行機器人。如果您將 元素設定為 SAVE Amazon Lex， 會儲存機器人，但不會建置它。

如果您未指定此值，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。您只能在建立機器人時新增標籤，無法使用 PutBot操作來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 ID。 Amazon Lex 為語音設定的地區設定必須符合機器人的地區設定。如需詳細資訊，請參閱《Amazon Polly 開發人員指南》中的 Amazon Polly 中的語音。 Amazon Polly

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

您建立之機器人的檢查總和。

類型：字串

childDirected

對於使用 Amazon Lex Model Building Service 建立的每個 Amazon Lex 機器人，您必須指定您對 Amazon Lex 的使用是否與網站、程式或其他應用程式相關，而該網站、程式或其他應用程式全部或部分針對 13 以下兒童，並在 `false` `childDirected` 欄位中指定 `true` 或 以遵守兒童線上隱私權保護法 (COPPA)。透過`true`在 `childDirected` 欄位中指定 ，您確認 Amazon Lex 的使用與網站、程式或其他應用程式相關，而該網站、程式或其他應用程式是全部或部分針對或鎖定 13 以下兒童，且受 COPPA 約束。透過`false`在 `childDirected` 欄位中指定 ，您確認 Amazon Lex 的使用與網站、程式或其他應用程式無關，該網站、程式或其他應用程式全部或部分針對 13 以下兒童，並受 COPPA 約束。您無法為 `childDirected` 欄位指定預設值，該值無法準確反映您對 Amazon Lex 的使用是否與網站、程式或其他應用程式相關，而該網站、程式或其他應用程式是針對或部分針對未滿 13 歲且受 COPPA 規範的 13 歲兒童。

如果您使用 Amazon Lex 與網站、程式或其他應用程式有關，而該網站、程式或其他應用程式全部或部分導向至未滿 13 歲的孩童，則您必須根據 COPPA 取得任何必要的可驗證父系同意。如需有關使用 Amazon Lex 與網站、程式或其他應用程式相關的資訊，而這些應用程式是針對或部分針對未滿 13 歲的孩童，請參閱 Amazon Lex 常見問答集。

類型：布林值

clarificationPrompt

當 Amazon Lex 不了解使用者的意圖時，會使用的提示。如需詳細資訊，請參閱PutBot。

類型：Prompt 物件

createdDate

機器人建立的日期。

類型：Timestamp

createVersion

True 如果建立新版本的機器人。如果請求中未指定 createVersion 欄位，則回應中的 createVersion 欄位會設為 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表達用語傳送至 Amazon Comprehend 以進行情緒分析。如果未在請求中指定 detectSentiment 欄位，則 detectSentiment 欄位會在回應false中。

類型：布林值

enableModelImprovements

指出機器人是否使用準確性改進。 true 指出機器人正在使用改進，否則是 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 保留對話中所收集資料的最大時間長度。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。當您建立資源時，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

長度限制：長度下限為 2。長度上限為 50。

模式：^([A-Za-z]_?)+$

nluIntentConfidenceThreshold

在 PostContent 或 PostText 回應中傳回替代意圖時AMAZON.KendraSearchIntent，決定 Amazon Lex 插入 AMAZON.FallbackIntent、 或兩者位置的分數。如果所有意圖的可信度分數都低於此值，AMAZON.FallbackIntent則會插入 。只有在為機器人設定時，AMAZON.KendraSearchIntent才會插入 。

類型：Double

有效範圍：最小值為 0。最大值為 1。

status

當您傳送建立將 processBehavior設定為 之機器人的請求時BUILD，Amazon Lex 會將status回應元素設定為 BUILDING。

在 READY_BASIC_TESTING 狀態中，您可以使用使用者輸入來測試機器人，這些輸入完全符合針對機器人意圖和槽類型中的值所設定的表達用語。

如果 Amazon Lex 無法建置機器人，Amazon Lex 會將 status設定為 FAILED。Amazon Lex 會在failureReason回應元素中傳回失敗的原因。

當您將 processBehavior 設定為 時SAVE，Amazon Lex 會將狀態碼設定為 NOT BUILT。

當機器人處於 READY 狀態時，您可以測試和發佈機器人。

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清單。

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，版本一律為 $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

Amazon Lex 用來與使用者進行語音互動的 Amazon Polly 語音 ID。如需詳細資訊，請參閱PutBot。

類型：字串

錯誤

BadRequestException

請求格式不正確。例如，值無效或缺少必要欄位。請檢查欄位值，然後再試一次。

HTTP 狀態碼：400

ConflictException

處理請求時發生衝突。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

您嘗試變更的資源檢查總和與請求中的檢查總和不相符。請檢查資源的檢查總和，然後再試一次。

HTTP 狀態碼：412

另請參閱

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列界面 V2
- AWS 適用於 .NET V4 的 SDK
- AWS 適用於 C++ 的 SDK
- AWS 適用於 Go 的 SDK v2
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 JavaScript V3 的 SDK
- AWS 適用於 Kotlin 的 SDK
- AWS 適用於 PHP V3 的 SDK
- AWS 適用於 Python 的 SDK
- AWS 適用於 Ruby V3 的 SDK

PutBotAlias

服務：Amazon Lex Model Building Service

為指定版本的機器人建立別名，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。

長度限制：長度下限為 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 $LATEST版本的特定修訂。

當您建立新的機器人別名時，請將 checksum 欄位保留空白。如果您指定檢查總和，您會收到BadRequestException例外狀況。

當您想要更新機器人別名時，請將 checksum 欄位設定為$LATEST最新版本的檢查總和。如果您未指定 checksum 欄位，或檢查總和與$LATEST版本不相符，您會收到PreconditionFailedException例外狀況。

類型：字串

必要：否

conversationLogs

別名對話日誌的設定。

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 PutBotAlias操作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

長度限制：長度下限為 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

目前版本的別名檢查總和。

類型：字串

conversationLogs

決定 Amazon Lex 如何使用別名對話日誌的設定。

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

機器人別名的更新日期。當您建立資源時，建立日期和上次更新的日期相同。

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清單。

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

請求格式不正確。例如，值無效或缺少必要欄位。請檢查欄位值，然後再試一次。

HTTP 狀態碼：400

ConflictException

處理請求時發生衝突。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

您嘗試變更的資源檢查總和與請求中的檢查總和不相符。請檢查資源的檢查總和，然後再試一次。

HTTP 狀態碼：412

另請參閱

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列界面 V2
- AWS 適用於 .NET V4 的 SDK
- AWS 適用於 C++ 的 SDK
- AWS 適用於 Go 的 SDK v2
- AWS 適用於 Java V2 的 SDK

- AWS 適用於 JavaScript V3 的 SDK
- AWS 適用於 Kotlin 的 SDK
- AWS 適用於 PHP V3 的 SDK
- AWS 適用於 Python 的 SDK
- AWS 適用於 Ruby V3 的 SDK

PutIntent

服務：Amazon Lex Model Building Service

建立意圖或取代現有的意圖。

若要定義使用者和機器人之間的互動，您可以使用一或多個意圖。例如，對於比薩訂購機器人，您可以建立`OrderPizza`意圖。

若要建立意圖或取代現有的意圖，您必須提供下列項目：

- 意圖名稱。例如 `OrderPizza`。
- 表達用語範例。例如，「我可以訂購比薩嗎？」 和「我想要訂購比薩。」
- 要收集的資訊。您可以為機器人向使用者請求的資訊指定槽類型。您可以指定標準槽類型，例如日期或時間，或自訂槽類型，例如比薩的大小和地殼。
- 如何實現意圖。您可以提供 Lambda 函數或設定意圖，將意圖資訊傳回至用戶端應用程式。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。如果您將意圖設定為將意圖資訊傳回至用戶端應用程式。

您可以在請求中指定其他選用資訊，例如：

- 要求使用者確認意圖的確認提示。例如，「Shall I order your pizza？」
- 滿足意圖後傳送給使用者的結論陳述式。例如，「我下了您的比薩訂單。」
- 後續提示，要求使用者進行其他活動。例如，詢問「您想要搭配比薩訂購飲料嗎？」

如果您指定現有的意圖名稱來更新意圖，Amazon Lex 會將意圖`$LATEST`版本中的值取代為請求中的值。Amazon Lex 會移除您在請求中未提供的欄位。如果您未指定必要欄位，Amazon Lex 會擲回例外狀況。當您更新意圖`$LATEST`版本時，任何使用意圖`$LATEST`版本之機器人`status`的欄位都會設定為`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。

名稱不符合內建意圖名稱，或內建意圖名稱與 "AMAZON"。已移除。例如，由於有名為 的內建意圖AMAZON.HelpIntent，因此您無法建立名為 的自訂意圖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 $LATEST版本的特定修訂。

當您建立新的意圖時，請將 checksum 欄位保留空白。如果您指定檢查總和，您會收到BadRequestException例外狀況。

當您想要更新意圖時，請將 checksum 欄位設定為$LATEST最新版本的檢查總和。如果您未指定 checksum 欄位，或檢查總和與$LATEST版本不相符，您會收到PreconditionFailedException例外狀況。

類型：字串

必要：否

conclusionStatement

您希望 Amazon Lex 在 Lambda 函數成功履行意圖之後，將 陳述式傳達給使用者。

只有在您在 中提供 Lambda 函數時，此元素才相關fulfillmentActivity。如果您將意圖傳回用戶端應用程式，則無法指定此元素。

> **Note**
>
> followUpPrompt 和 conclusionStatement是互斥的。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，使用 OrderPizza意圖，您可能想要先確認順序是否正確，再進行放置。對於其他意圖，例如僅回應使用者問題的意圖，您可能不需要在提供資訊之前要求使用者確認。

> **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true新編號版本的意圖時，即會建立。這與呼叫 CreateIntentVersion操作相同。如果您未指定 createVersion，則預設值為 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叫用的 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，並預先填入一些值。例如，如果您發現 John 不容許黏附，您可以將對應的意圖槽 GlutenIntolerant設為 true。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 會在滿足意圖後使用此提示來請求其他活動。例如，在滿足OrderPizza意圖之後，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」，則會使用為機器人設定的釐清提示來回應。
- 如果使用者說「是」並繼續表達可觸發意圖的表達用語，則會針對意圖開始對話。
- 如果使用者說「否」，則會回應為後續提示設定的拒絕陳述式。
- 如果無法辨識表達用語，則會再次重複追蹤提示。

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，在使用者提供比薩訂單的所有資訊後，會fulfillmentActivity定義機器人如何向本機比薩商店下訂單。

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，使用披薩下訂單）。

類型：FulfillmentActivity 物件

必要：否

inputContexts

物件陣列InputContext，列出 Amazon Lex 在與使用者的對話中必須處於作用中狀態的內容，以選擇意圖。

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

使用連線至 Amazon Kendra 索引的AMAZON.KendraSearchIntent意圖所需的組態資訊。如需詳細資訊，請參閱 AMAZON.KendraSearchIntent。

類型：KendraConfiguration 物件

必要：否

outputContexts

物件陣列OutputContext，列出意圖在滿足意圖時啟用的內容。

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。若要尋找意圖的簽章，請參閱 Alexa Skills Kit 中的標準內建意圖。

類型：字串

必要：否

rejectionStatement

當使用者對 中定義的問題回答「否」時confirmationPrompt，Amazon Lex 會回應此陳述式，以確認意圖已取消。

> **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

使用者可能說出來表示意圖的表達用語（字串）陣列。例如，「我想要 {PizzaSize} 比薩」、「Order {Quantity} {PizzaSize} 比薩」。

在每個表達用語中，槽名稱是以大括號括住。

類型：字串陣列

陣列成員：項目數下限為 0。最多 1500 個項目。

長度限制：長度下限為 1。長度上限為 200。

必要：否

slots

意圖槽陣列。在執行時間，Amazon Lex 會使用槽中定義的提示，從使用者引出所需的槽值。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

已建立或更新的意圖$LATEST版本總和檢查碼。

類型：字串

conclusionStatement

在fulfillmentActivity意圖中指定的 Lambda 函數滿足意圖之後，Amazon Lex 會將此陳述式傳達給使用者。

類型：Statement 物件

confirmationPrompt

如果在意圖中定義，Amazon Lex 會提示使用者先確認意圖，再履行意圖。

類型：Prompt 物件

createdDate

建立意圖的日期。

類型：Timestamp

createVersion

True 如果建立了新版本的意圖。如果請求中未指定 createVersion 欄位，則回應中的 createVersion 欄位會設為 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

如果在意圖中定義，Amazon Lex 會為每個使用者輸入調用此 Lambda 函數。

類型：CodeHook 物件

followUpPrompt

如果在意圖中定義，Amazon Lex 會在意圖實現後使用此提示來請求額外的使用者活動。

類型：FollowUpPrompt 物件

fulfillmentActivity

如果在意圖中定義，Amazon Lex 會在使用者提供意圖所需的所有資訊後調用此 Lambda 函數以滿足意圖。

類型：FulfillmentActivity 物件

inputContexts

物件陣列InputContext，列出 Amazon Lex 在與使用者的對話中必須處於作用中狀態的內容，以選擇意圖。

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

若有組態資訊，則需要連線至 Amazon Kendra 索引並使用AMAZON.KendraSearchIntent意圖。

類型：KendraConfiguration 物件

lastUpdatedDate

意圖更新的日期。當您建立資源時，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

物件陣列OutputContext，列出意圖在滿足意圖時啟用的內容。

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

此意圖所根據之內建意圖的唯一識別符。

類型：字串

rejectionStatement

如果使用者對 Amazon Lex confirmationPrompt 中定義的問題回答「否」，則使用此陳述式回應，以確認意圖已取消。

類型：Statement 物件

sampleUtterances

為意圖設定的範例表達用語陣列。

類型：字串陣列

陣列成員：項目數下限為 0。最多 1500 個項目。

長度限制：長度下限為 1。長度上限為 200。

slots

為意圖設定的意圖插槽陣列。

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。對於新意圖，版本一律為 $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

請求格式不正確。例如，值無效或缺少必要欄位。請檢查欄位值，然後再試一次。

HTTP 狀態碼：400

ConflictException

處理請求時發生衝突。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

您嘗試變更的資源檢查總和與請求中的檢查總和不相符。請檢查資源的檢查總和，然後再試一次。

HTTP 狀態碼：412

另請參閱

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列界面 V2
- AWS 適用於 .NET V4 的 SDK
- AWS 適用於 C++ 的 SDK
- AWS 適用於 Go 的 SDK v2
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 JavaScript V3 的 SDK
- AWS 適用於 Kotlin 的 SDK
- AWS 適用於 PHP V3 的 SDK
- AWS 適用於 Python 的 SDK
- AWS 適用於 Ruby 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。

若要建立自訂插槽類型，請指定插槽類型的名稱和一組列舉值，這是此類型插槽可擔任的值。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$LATEST版本中的現有值。Amazon Lex 會移除您在請求中未提供的欄位。如果您未指定必要欄位，Amazon Lex 會擲回例外狀況。當您更新槽類型的$LATEST版本時，如果機器人使用包含槽類型的意圖$LATEST版本，機器人status的欄位會設定為 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。

名稱不符合內建槽類型名稱，或內建槽類型名稱與 "AMAZON"。已移除。例如，由於有名為 的內建插槽類型AMAZON.DATE，因此您無法建立名為 的自訂插槽類型DATE。

如需內建插槽類型的清單，請參閱 Alexa Skills Kit 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 $LATEST版本的特定修訂。

當您建立新的插槽類型時，請將checksum欄位保留空白。如果您指定檢查總和，您會收到BadRequestException例外狀況。

當您想要更新插槽類型時，請將 checksum 欄位設定為最新版本的檢查總和$LATEST。如果您未指定 checksum 欄位，或檢查總和與$LATEST版本不相符，您會收到PreconditionFailedException例外狀況。

類型：字串

必要：否

createVersion

當設定為true新編號版本的槽類型時，即會建立。這與呼叫 CreateSlotTypeVersion操作相同。如果您未指定 createVersion，則預設值為 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

定義槽類型可採用之值的EnumerationValue物件清單。每個值都可以有一個 清單synonyms，這些是額外的值，可協助訓練機器學習模型有關其針對插槽解析的值。

規則表達式槽類型不需要列舉值。所有其他槽類型都需要列舉值的清單。

當 Amazon Lex 解析槽值時，會產生解析度清單，其中包含最多五個可能的槽值。如果您使用的是 Lambda 函數，此解析清單會傳遞給函數。如果您不是使用 Lambda 函數，您可以選擇傳回使用者輸入的值或解析清單中的第一個值作為槽值。valueSelectionStrategy 欄位指出要使用的選項。

類型：EnumerationValue 物件陣列

陣列成員：項目數下限為 0。10000 個項目的數量上限。

必要：否

parentSlotTypeSignature

做為插槽類型父項的內建插槽類型。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

延伸父內建插槽類型的組態資訊。組態會新增至父槽類型的設定。

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 用來傳回槽類型值的槽解析策略。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 - 如果插槽有解析度清單，請傳回解析度清單中的第一個值做為插槽類型值。如果沒有解析清單，則傳回 null。

如果您未指定 valueSelectionStrategy，則預設為 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

槽類型的$LATEST版本總和檢查碼。

類型：字串

createdDate

槽類型的建立日期。

類型：Timestamp

createVersion

True 如果已建立新版本的槽類型。如果請求中未指定 createVersion 欄位，則回應中的 createVersion 欄位會設為 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

定義槽類型可採用之值的EnumerationValue物件清單。

類型：EnumerationValue 物件陣列

陣列成員：項目數下限為 0。10000 個項目的數量上限。

lastUpdatedDate

槽類型更新的日期。當您建立槽類型時，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

做為插槽類型父項的內建插槽類型。

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

延伸父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 用來判斷槽值的槽解析策略。如需詳細資訊，請參閱PutSlotType。

類型：字串

有效值:ORIGINAL_VALUE | TOP_RESOLUTION

version

槽類型的版本。對於新的插槽類型，版本一律為 $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

請求格式不正確。例如，值無效或缺少必要欄位。請檢查欄位值，然後再試一次。

HTTP 狀態碼：400

ConflictException

處理請求時發生衝突。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

您嘗試變更的資源檢查總和與請求中的檢查總和不相符。請檢查資源的檢查總和，然後再試一次。

HTTP 狀態碼：412

另請參閱

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列界面 V2
- AWS 適用於 .NET V4 的 SDK
- AWS 適用於 C++ 的 SDK
- AWS 適用於 Go 的 SDK v2
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 JavaScript V3 的 SDK
- AWS 適用於 Kotlin 的 SDK
- AWS 適用於 PHP V3 的 SDK
- AWS 適用於 Python 的 SDK
- AWS 適用於 Ruby 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

指定當現有資源具有相同名稱時，StartImport操作應採取的動作。

- FAIL_ON_CONFLICT - 在匯入檔案中的資源與現有資源之間的第一次衝突時，會停止匯入操作。造成衝突的資源名稱位於 GetImport 操作回應的 failureReason欄位中。

  OVERWRITE_LATEST - 即使與現有資源發生衝突，匯入操作仍會繼續。現有資源的 $LASTEST 版本會以匯入檔案的資料覆寫。

類型：字串

有效值:OVERWRITE_LATEST | FAIL_ON_CONFLICT

必要：是

payload

二進位格式的 zip 封存。封存應包含一個檔案，其中包含要匯入之資源的 JSON 檔案。資源應符合 `resourceType` 欄位中指定的類型。

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。每個資源也會匯出其相依的任何資源。

- 機器人匯出相依意圖。
- 意圖匯出相依槽類型。

類型：字串

有效值:`BOT | INTENT | SLOT_TYPE`

必要：是

tags

要新增至匯入機器人的標籤清單。您只能在匯入機器人時新增標籤，您無法將標籤新增至意圖或槽類型。

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

請求匯入任務的日期和時間的時間戳記。

類型：Timestamp

importId

特定匯入任務的識別符。

類型：字串

importStatus

匯入任務的狀態。如果狀態為 FAILED，您可以使用 GetImport操作取得失敗的原因。

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

提供給匯入任務的名稱。

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

新增至匯入機器人的標籤清單。

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

請求格式不正確。例如，值無效或缺少必要欄位。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

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列界面 V2
- AWS 適用於 .NET V4 的 SDK
- AWS 適用於 C++ 的 SDK
- AWS 適用於 Go 的 SDK v2
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 JavaScript V3 的 SDK
- AWS 適用於 Kotlin 的 SDK
- AWS 適用於 PHP V3 的 SDK
- AWS 適用於 Python 的 SDK
- AWS 適用於 Ruby V3 的 SDK

StartMigration

服務：Amazon Lex Model Building Service

開始將機器人從 Amazon Lex V1 遷移至 Amazon Lex V2。當您想要利用 Amazon Lex V2 的新功能時，請遷移您的機器人。

如需詳細資訊，請參閱《Amazon Lex 開發人員指南》中的遷移機器人。

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

用來執行遷移的策略。

- CREATE_NEW - 建立新的 Amazon Lex V2 機器人，並將 Amazon Lex V1 機器人遷移至新的機器人。
- 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地區設定。如果地區設定不存在，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

您要遷移至 Amazon Lex V2 的 Amazon Lex V1 機器人名稱。 Amazon Lex V2

類型：字串

長度限制：長度下限為 2。長度上限為 50。

模式：^([A-Za-z]_?)+$

必要：是

v1BotVersion

要遷移至 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

您要遷移 Amazon Lex V2 機器人的 Amazon Lex V1 機器人名稱。

- 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

Amazon Lex 用來執行 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符。

類型：字串

長度限制：固定長度為 10。

模式：^[0-9a-zA-Z]+$

migrationStrategy

用來執行遷移的策略。

類型：字串

有效值:CREATE_NEW | UPDATE_EXISTING

migrationTimestamp

遷移開始的日期和時間。

類型：Timestamp

v1BotLocale

用於 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

v1BotName

您要遷移至 Amazon Lex V2 的 Amazon Lex V1 機器人名稱。 Amazon Lex V2

類型：字串

長度限制：長度下限為 2。長度上限為 50。

模式：^([A-Za-z]_?)+$

v1BotVersion

要遷移至 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 機器人的唯一識別符。

類型：字串

長度限制：固定長度為 10。

模式：^[0-9a-zA-Z]+$

v2BotRole

Amazon Lex 用來執行 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 使用者或角色無權呼叫遷移機器人所需的 Amazon Lex V2 APIs。

HTTP 狀態碼：403

BadRequestException

請求格式不正確。例如，值無效或缺少必要欄位。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

找不到請求中指定的資源。請檢查資源，然後再試一次。

HTTP 狀態碼：404

另請參閱

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列界面 V2
- AWS 適用於 .NET V4 的 SDK
- AWS 適用於 C++ 的 SDK
- AWS 適用於 Go 的 SDK v2
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 JavaScript V3 的 SDK
- AWS 適用於 Kotlin 的 SDK
- AWS 適用於 PHP V3 的 SDK
- AWS 適用於 Python 的 SDK
- AWS 適用於 Ruby V3 的 SDK

TagResource

服務：Amazon Lex Model Building Service

將指定的標籤新增到指定的資源。如果標籤索引鍵已存在，則會以新值取代現有值。

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

要標記之機器人、機器人別名或機器人管道的 Amazon Resource Name (ARN)。

長度限制：長度下限為 1。長度上限為 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索引鍵清單。如果標籤索引鍵已存在，則會以新值取代現有值。

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

請求格式不正確。例如，值無效或缺少必要欄位。請檢查欄位值，然後再試一次。

HTTP 狀態碼：400

ConflictException

處理請求時發生衝突。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

找不到請求中指定的資源。請檢查資源，然後再試一次。

HTTP 狀態碼：404

另請參閱

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列界面 V2
- AWS 適用於 .NET V4 的 SDK
- AWS 適用於 C++ 的 SDK
- AWS 適用於 Go 的 SDK v2
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 JavaScript V3 的 SDK
- AWS 適用於 Kotlin 的 SDK
- AWS 適用於 PHP V3 的 SDK
- AWS 適用於 Python 的 SDK
- AWS 適用於 Ruby V3 的 SDK

UntagResource

服務：Amazon Lex Model Building Service

從機器人、機器人別名或機器人管道移除標籤。

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

要從中移除標籤之資源的 Amazon Resource Name (ARN)。

長度限制：長度下限為 1。長度上限為 1011。

必要：是

tagKeys

要從資源中移除的標籤索引鍵清單。如果資源上不存在標籤索引鍵，則會予以忽略。

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

請求格式不正確。例如，值無效或缺少必要欄位。請檢查欄位值，然後再試一次。

HTTP 狀態碼：400

ConflictException

處理請求時發生衝突。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

找不到請求中指定的資源。請檢查資源，然後再試一次。

HTTP 狀態碼：404

另請參閱

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列界面 V2
- AWS 適用於 .NET V4 的 SDK
- AWS 適用於 C++ 的 SDK
- AWS 適用於 Go 的 SDK v2
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 JavaScript V3 的 SDK
- AWS 適用於 Kotlin 的 SDK

- AWS 適用於 PHP V3 的 SDK
- AWS 適用於 Python 的 SDK
- AWS 適用於 Ruby V3 的 SDK

Amazon Lex 執行期服務

Amazon Lex Runtime Service 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

用於包含工作階段資料的機器人的別名。

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之使用者的識別符。

長度限制：長度下限為 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

用於與工作階段資料相關聯之機器人的別名。

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符。

類型：字串

userId

用戶端應用程式使用者的 ID。

類型：字串

長度限制：長度下限為 2。長度上限為 100。

模式：`[0-9a-zA-Z._:-]+`

錯誤

BadRequestException

請求驗證失敗、內容中沒有可用的訊息，或機器人建置失敗、仍在進行中，或包含未建置的變更。

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 ID。

HTTP 狀態碼：409

InternalFailureException

內部服務錯誤。重試 呼叫。

HTTP 狀態碼：500

LimitExceededException

超過限制。

HTTP 狀態碼：429

NotFoundException

找不到參考的資源 （例如 Amazon Lex 機器人或別名）。

HTTP 狀態碼：404

另請參閱

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列界面 V2
- AWS 適用於 .NET V4 的 SDK
- AWS 適用於 C++ 的 SDK
- AWS 適用於 Go 的 SDK v2
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 JavaScript V3 的 SDK
- AWS 適用於 Kotlin 的 SDK
- AWS 適用於 PHP V3 的 SDK
- AWS 適用於 Python 的 SDK
- AWS 適用於 Ruby 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

用於包含工作階段資料的機器人的別名。

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結構中傳回之意圖的字串。

當您指定篩選條件時，只會傳回 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 使用此項目來識別使用者與您的機器人的對話。

長度限制：長度下限為 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

工作階段的作用中內容清單。滿足意圖或呼叫 PostContent、 或 PutSession操作時PostText，可以設定內容。

您可以使用內容來控制可追蹤意圖的意圖，或修改應用程式的操作。

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

描述機器人的目前狀態。

類型：DialogAction 物件

recentIntentSummaryView

有關工作階段中所用意圖的資訊陣列。陣列最多可包含三個摘要。如果在工作階段中使用超過三個意圖，recentIntentSummaryView操作會包含最後三個使用意圖的相關資訊。

如果您在請求中設定 checkpointLabelFilter 參數，則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。最多 3 個項目。

sessionAttributes

代表工作階段特定內容資訊的鍵/值對映射。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、內容中沒有可用的訊息，或機器人建置失敗、仍在進行中，或包含未建置的變更。

HTTP 狀態碼：400

InternalFailureException

內部服務錯誤。重試 呼叫。

HTTP 狀態碼：500

LimitExceededException

超過限制。

HTTP 狀態碼：429

NotFoundException

找不到參考的資源（例如 Amazon Lex 機器人或別名）。

HTTP 狀態碼：404

另請參閱

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列界面 V2
- AWS 適用於 .NET V4 的 SDK
- AWS 適用於 C++ 的 SDK
- AWS 適用於 Go 的 SDK v2
- AWS 適用於 Java V2 的 SDK

- AWS 適用於 JavaScript V3 的 SDK
- AWS 適用於 Kotlin 的 SDK
- AWS 適用於 PHP V3 的 SDK
- AWS 適用於 Python 的 SDK
- AWS 適用於 Ruby 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。用戶端會使用此 API 在執行時間將文字和音訊請求傳送至 Amazon Lex。Amazon Lex 會使用為機器人建置的機器學習模型來解譯使用者輸入。

`PostContent` 操作支援 8kHz 和 16kHz 的音訊輸入。您可以使用 8kHz 音訊，在電話音訊應用程式中實現更高的語音辨識準確度。

為了回應，Amazon Lex 會傳回下一個訊息以傳達給使用者。請考慮下列範例訊息：

- 對於「我想要比薩」的使用者輸入，Amazon Lex 可能會傳回包含訊息引出槽資料的回應 （ 例如，`PizzaSize`)：「您想要多大的比薩？」。
- 使用者提供所有比薩訂單資訊後，Amazon Lex 可能會傳回包含訊息的回應，以取得使用者確認：「訂購比薩？」。
- 使用者回覆「是」至確認提示後，Amazon Lex 可能會傳回結論陳述式：「感謝您，您的起司比薩已訂購。」

並非所有 Amazon Lex 訊息都需要使用者的回應。例如，結論陳述式不需要回應。有些訊息只需要是或否回應。除了 之外`message`，Amazon Lex 還提供回應中訊息的其他內容，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：

- 如果訊息是要引出槽資料，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` 標頭設定為目前內容中的意圖名稱
  - `x-amz-lex-slot-to-elicit` 標頭設定為 `message` 為其引出資訊的槽名稱
  - `x-amz-lex-slots` 標頭設定為以其目前值為意圖設定的槽映射
- 如果訊息是確認提示，則 `x-amz-lex-dialog-state`標頭會設為 ，`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釐清提示，表示不了解使用者意圖，則 `x-amz-dialog-state`標頭會設為 `ElicitIntent`，而 `x-amz-slot-to-elicit`標頭會省略。

此外，Amazon Lex 也會傳回應用程式特定的 `sessionAttributes`。如需詳細資訊，請參閱管理對話內容。

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

您可以將此值傳遞為 Accept HTTP 標頭。

Amazon Lex 在回應中傳回的訊息可以根據請求中的 Accept HTTP 標頭值為文字或語音。

- 如果值為 text/plain; charset=utf-8，Amazon Lex 會在回應中傳回文字。
- 如果值以 開頭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做為值，Amazon Lex 會以 MPEG 格式傳回語音。
- 如果值為 audio/pcm，則傳回的語音audio/pcm為 16 位元、小端數格式。
- 以下是可接受的值：
  - 音訊/mpeg
  - 音訊/霧
  - 音訊/pcm
  - text/plain； charset=utf-8
  - audio/*（預設為 mpeg)

activeContexts

用於請求的作用中內容清單。在滿足先前的意圖時，或藉由在請求中包含內容，即可啟用內容。

如果您未指定內容清單，Amazon Lex 將使用工作階段的目前內容清單。如果您指定空白清單，工作階段的所有內容都會清除。

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

您傳遞此值做為 Content-Type HTTP 標頭。

指出音訊格式或文字。標頭值必須以下列其中一個字首開頭：

- PCM 格式的音訊資料必須按小端位元組順序排列。
  - 音訊/l16；速率 = 16000；頻道 = 1
  - audio/x-l16； sample-rate=16000； channel-count=1
  - audio/lpcm； sample-rate=8000； sample-size-bits=16； channel-count=1； is-big-endian=false
- Opus 格式
  - audio/x-cbr-opus-with-preamble；preamble-size=0；位元速率=256000；frame-size-milliseconds=4
- 文字格式
  - text/plain； charset=utf-8

必要：是

requestAttributes

您可以將此值傳遞為 x-amz-lex-request-attributes HTTP 標頭。

Amazon Lex 與用戶端應用程式之間傳遞的請求特定資訊。此值必須是具有字串索引鍵和值的 JSON 序列化和 base64 編碼映射。requestAttributes 和 sessionAttributes標頭的總大小限制為 12 KB。

命名空間x-amz-lex:會保留給特殊屬性。請勿建立任何字首為 的請求屬性x-amz-lex:。

如需詳細資訊，請參閱設定請求屬性。

sessionAttributes

您傳遞此值做為 `x-amz-lex-session-attributes` HTTP 標頭。

Amazon Lex 與用戶端應用程式之間傳遞的應用程式特定資訊。此值必須是具有字串索引鍵和值的 JSON 序列化和 base64 編碼映射。`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 使用此項目來識別使用者與您的機器人的對話。在執行時間，每個請求必須包含 `userID` 欄位。

若要決定應用程式要使用的使用者 ID，請考慮下列因素。

- `userID` 欄位不得包含使用者的任何個人身分識別資訊，例如姓名、個人身分證號碼或其他最終使用者個人資訊。
- 如果您希望使用者在一個裝置上開始對話，並在另一個裝置上繼續對話，請使用使用者特定的識別符。
- 如果您希望相同的使用者能夠在兩個不同的裝置上進行兩個獨立的對話，請選擇裝置特定的識別符。
- 使用者無法使用相同機器人的兩個不同版本進行兩個獨立對話。例如，使用者無法與相同機器人的 PROD 和 BETA 版本進行對話。如果您預期使用者需要與兩個不同的版本進行對話，例如在測試時，請在使用者 ID 中包含機器人別名，以分隔兩個對話。

長度限制：長度下限為 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

PCM 或 Opus 音訊格式或文字格式的使用者輸入，如 `Content-Type` HTTP 標頭中所述。

您可以將音訊資料串流到 Amazon Lex，也可以建立本機緩衝區，在傳送之前擷取所有音訊資料。一般而言，如果您串流音訊資料，而不是在本機緩衝資料，就能獲得更好的效能。

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

工作階段的作用中內容清單。滿足意圖或呼叫 PostContent、 或 PutSession操作時PostText，可以設定內容。

您可以使用內容來控制可追蹤意圖的意圖，或修改應用程式的操作。

alternativeIntents

一到四個可能適用於使用者意圖的替代意圖。

每個替代方案都包含一個分數，指出 Amazon Lex 對意圖符合使用者意圖的可信度。意圖會依可信度分數排序。

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版本的機器人效能是否優於另一個版本。

長度限制：長度下限為 1。長度上限為 64。

模式：`[0-9]+|\$LATEST`

contentType

請求中 `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 會將下列其中一個值傳回為 `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 想要引出使用者的意圖。請考量下列範例：

  例如，使用者可能會說出意圖 (「我想要訂購比薩」)。如果 Amazon Lex 無法從此表達式推斷使用者意圖，則會傳回此對話方塊狀態。

- `ConfirmIntent` - Amazon Lex 預期收到「是」或「否」回應。

  例如，Amazon Lex 想要在滿足意圖之前確認使用者。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，但讓它成為厚皮比薩」或「否，我想要點一杯。」Amazon Lex 可以處理這類額外資訊 （在這些範例中，更新餅皮類型槽，或將意圖從 OrderPizza 變更為 OrderDrink)。

- `ElicitSlot` - Amazon Lex 預期目前意圖的槽值。

  例如，假設 Amazon Lex 在回應中傳送此訊息：「您想要什麼大小的比薩？」。使用者可能會以槽值 （例如 "medium") 回覆。使用者也可能在回應中提供其他資訊 （例如，「中厚餅皮比薩」)。Amazon Lex 可以適當地處理這類額外資訊。

- `Fulfilled` - 傳達 Lambda 函數已成功滿足意圖。
- `ReadyForFulfillment` - 傳達用戶端必須滿足請求的內容。
- `Failed` - 傳達與使用者的對話失敗。

  這可能由於各種原因而發生，包括使用者無法對來自 服務的提示提供適當的回應 （您可以設定 Amazon Lex 可提示使用者提供特定資訊的次數），或者 Lambda 函數無法滿足意圖。

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。您可以使用此資訊來判斷 Amazon Lex 是否正確處理您傳送的音訊。

`encodedInputTranscript` 欄位以 base-64 編碼。您必須先解碼 欄位，才能使用 值。

encodedMessage

要傳達給使用者的訊息。訊息可能來自機器人的組態或 Lambda 函數。

如果意圖未使用 Lambda 函數設定，或者 Lambda 函數在其回應`dialogAction.type`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 無法了解使用者輸入，則會使用釐清提示訊息。

建立意圖時，您可以將訊息指派給群組。當訊息指派給群組時，Amazon Lex 會從回應中的每個群組傳回一則訊息。訊息欄位是包含訊息的逸出 JSON 字串。如需傳回之 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 會在回應中將其傳遞給用戶端。

`encodedMessage` 欄位以 base-64 編碼。您必須先解碼 欄位，才能使用 值。

長度限制：長度下限為 1。長度上限為 1366。

inputTranscript

此標頭已棄用。

用來處理請求的文字。

您只能在 de-DE、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。在所有其他地區中， `inputTranscript` 欄位為 null。您應該改用 `encodedInputTranscript` 欄位。

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。您可以使用此資訊來判斷 Amazon Lex 是否正確處理您傳送的音訊。

intentName

Amazon Lex 目前知道的使用者意圖。

message

此標頭已棄用。

您只能在 de-DE、en-AU、en-GB、en-US、es-419、es-ES、es-US、fr-CA、fr-FR 和 it-IT 區域中使用此欄位。在所有其他地區中，`message` 欄位為 null。您應該改用 `encodedMessage` 欄位。

要傳達給使用者的訊息。訊息可能來自機器人的組態或 Lambda 函數。

如果意圖未使用 Lambda 函數設定，或者 Lambda 函數在其回應`dialogAction.type`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 無法了解使用者輸入，則會使用釐清提示訊息。

建立意圖時，您可以將訊息指派給群組。當訊息指派給群組時，Amazon Lex 會從回應中的每個群組傳回一則訊息。訊息欄位是包含訊息的逸出 JSON 字串。如需傳回之 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 會在回應中將其傳遞給用戶端。

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` - 訊息包含純 UTF-8 文字。
- `CustomPayload` - 訊息是用戶端的自訂格式。
- `SSML` - 訊息包含語音輸出的文字格式。
- `Composite` - 訊息包含逸出的 JSON 物件，其中包含訊息在建立意圖時指派給群組的一或多個訊息。

有效值:`PlainText | CustomPayload | SSML | Composite`

nluIntentConfidence

提供分數，指出 Amazon Lex 對傳回意圖是符合使用者意圖的意圖的可信度。分數介於 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 的改進而變更。

sentimentResponse

在表達用語中表達的情緒。

當機器人設定為將表達用語傳送至 Amazon Comprehend 進行情緒分析時，此欄位會包含分析結果。

sessionAttributes

代表工作階段特定內容資訊的鍵/值對映射。

sessionId

工作階段的唯一識別符。

slots

在對話期間，從使用者輸入偵測到的零或多個意圖槽（名稱/值對） Amazon Lex映射。欄位以 base-64 編碼。

Amazon Lex 會建立解析度清單，其中包含槽的可能值。傳回的值取決於建立或更新槽類型時valueSelectionStrategy選取的 。如果 valueSelectionStrategy 設定為 ORIGINAL_VALUE，則會傳回使用者提供的值，如果使用者值與槽值類似。如果 valueSelectionStrategy 設定為 TOP_RESOLUTION Amazon Lex，則會傳回解析清單中的第一個值，或者如果沒有解析清單，則為 null。如果您未指定 valueSelectionStrategy，則預設值為 ORIGINAL_VALUE。

slotToElicit

如果dialogState值為 ElicitSlot， 會傳回 Amazon Lex 為其引出值的槽名稱。

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）。這是以機器人組態和內容為基礎。例如，如果 Amazon Lex 不了解使用者意圖，則會傳送為機器人clarificationPrompt設定的 。如果意圖在採取履行動作之前需要確認，則會傳送 confirmationPrompt。另一個範例：假設 Lambda 函數成功實現意圖，並傳送訊息給使用者。然後，Amazon Lex 會在回應中傳送該訊息。

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) 失敗並發生內部服務錯誤。

HTTP 狀態碼：502

BadRequestException

請求驗證失敗、內容中沒有可用的訊息，或機器人建置失敗、仍在進行中，或包含未建置的變更。

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 ID。

HTTP 狀態碼：409

DependencyFailedException

AWS Lambda 或 Amazon Polly 等其中一個相依性擲出例外狀況。例如

- 如果 Amazon Lex 沒有足夠的許可來呼叫 Lambda 函數。
- 如果 Lambda 函數需要超過 30 秒才能執行。
- 如果履行 Lambda 函數傳回`Delegate`對話方塊動作，而不移除任何槽值。

HTTP 狀態碼：424

InternalFailureException

內部服務錯誤。重試 呼叫。

HTTP 狀態碼：500

LimitExceededException

超過限制。

HTTP 狀態碼：429

LoopDetectedException

不會使用此例外狀況。

HTTP 狀態碼：508

NotAcceptableException

請求中的接受標頭沒有有效的值。

HTTP 狀態碼：406

NotFoundException

找不到參考的資源（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

Content-Type 標頭 (PostContent API) 的值無效。

HTTP 狀態碼：415

範例

範例 1

在此請求中，URI 會識別機器人（流量）、機器人版本 ($LATEST) 和最終使用者名稱（使用者）。Content-Type 標頭可識別內文中音訊的格式。Amazon Lex 也支援其他格式。若要將音訊從一種格式轉換為另一種格式，如有必要，您可以使用 SoX 開放原始碼軟體。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， x-amz-lex-message標頭會顯示 Amazon Lex 傳回的回應。然後，用戶端可以將此回應傳送給使用者。相同的訊息會透過區塊編碼（依要求）以音訊/MPEG 格式傳送。

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

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列界面 V2
- AWS 適用於 .NET V4 的 SDK
- AWS 適用於 C++ 的 SDK
- AWS 適用於 Go 的 SDK v2
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 JavaScript V3 的 SDK
- AWS 適用於 Kotlin 的 SDK
- AWS 適用於 PHP V3 的 SDK
- AWS 適用於 Python 的 SDK
- AWS 適用於 Ruby V3 的 SDK

PostText

服務：Amazon Lex Runtime Service

將使用者輸入傳送至 Amazon Lex。用戶端應用程式可以使用此 API 在執行時間將請求傳送至 Amazon Lex。然後，Amazon Lex 會使用為機器人建置的機器學習模型來解譯使用者輸入。

為了回應，Amazon Lex 會傳回下一個 `message`，以向使用者傳遞`responseCard`要顯示的選用 。請考慮下列範例訊息：

- 對於使用者輸入「我想要比薩」，Amazon Lex 可能會傳回包含訊息引出槽資料的回應 （ 例如 PizzaSize)：「您想要什麼大小的比薩？」
- 使用者提供所有比薩訂單資訊後，Amazon Lex 可能會傳回包含訊息的回應，以取得使用者確認「使用比薩訂單繼續？」。
- 使用者回覆「是」的確認提示後，Amazon Lex 可能會傳回結論陳述式：「感謝您，您的起司比薩已訂購。」

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」使用者回應。除了 之外`message`，Amazon Lex 還提供回應中訊息的其他內容，例如，用於增強用戶端行為，以顯示適當的用戶端使用者介面。這些是回應中的 `slotToElicit`、`intentName`、 `dialogState`和 `slots` 欄位。請考量下列範例：

- 如果訊息是要引出槽資料，Amazon Lex 會傳回下列內容資訊：
  - `dialogState` 設定為 ElicitSlot
  - `intentName` 設定為目前內容中的意圖名稱
  - `slotToElicit` 設定為 `message` 為其引出資訊的槽名稱
  - `slots` 設定為針對意圖設定的插槽映射，具有目前已知值
- 如果訊息是確認提示，`dialogState`則 設定為 ConfirmIntent，而 `SlotToElicit` 設定為 null。
- 如果訊息是釐清提示 （ 針對意圖設定），指出不了解使用者意圖，`dialogState`則 會設定為 ElicitIntent，並將 `slotToElicit` 設定為 null。

此外，Amazon Lex 也會傳回應用程式特定的 `sessionAttributes`。如需詳細資訊，請參閱管理對話內容。

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 使用此項目來識別使用者與您的機器人的對話。在執行時間，每個請求必須包含 `userID` 欄位。

若要決定應用程式要使用的使用者 ID，請考慮下列因素。

- `userID` 欄位不得包含使用者的任何個人身分識別資訊，例如姓名、個人身分證號碼或其他最終使用者個人資訊。
- 如果您希望使用者在一個裝置上開始對話，並在另一個裝置上繼續對話，請使用使用者特定的識別符。
- 如果您希望相同的使用者能夠在兩個不同的裝置上進行兩個獨立的對話，請選擇裝置特定的識別符。
- 使用者無法使用相同機器人的兩個不同版本進行兩個獨立對話。例如，使用者無法與相同機器人的 PROD 和 BETA 版本進行對話。如果您預期使用者需要與兩個不同的版本進行對話，例如在測試時，請在使用者 ID 中包含機器人別名，以分隔兩個對話。

長度限制：長度下限為 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

用於請求的作用中內容清單。在滿足先前的意圖時，或藉由在請求中包含內容，即可啟用內容。

如果您未指定內容清單，Amazon Lex 將使用工作階段的目前內容清單。如果您指定空白清單，工作階段的所有內容都會清除。

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。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 與用戶端應用程式之間傳遞的請求特定資訊。

命名空間x-amz-lex:會保留給特殊屬性。請勿建立任何字首為 的請求屬性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 與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

工作階段的作用中內容清單。滿足意圖或呼叫 PostContent、 或 PutSession操作時PostText，可以設定內容。

您可以使用內容來控制可追蹤意圖的意圖，或修改應用程式的操作。

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

一到四個可能適用於使用者意圖的替代意圖。

每個替代方案都包含一個分數，指出 Amazon Lex 對意圖符合使用者意圖的可信度。意圖會依可信度分數排序。

類型：PredictedIntent 物件陣列

陣列成員：最多 4 個項目。

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版本的機器人效能是否優於另一個版本。

類型：字串

長度限制：長度下限為 1。長度上限為 64。

模式：`[0-9]+|\$LATEST`

dialogState

識別使用者互動的目前狀態。Amazon Lex 會將下列其中一個值傳回為 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 想要引發使用者意圖。

  例如，使用者可能會說出意圖 (「我想要訂購比薩」)。如果 Amazon Lex 無法從此表達式推斷使用者意圖，則會傳回此 dialogState。

- `ConfirmIntent` - Amazon Lex 預期收到「是」或「否」回應。

  例如，Amazon Lex 想要在滿足意圖之前確認使用者。

  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，但將其製作成厚皮比薩」或「否，我想要訂購飲料」。Amazon Lex 可以處理這類額外資訊 （在這些範例中，更新結構類型槽值，或將意圖從 OrderPizza 變更為 OrderDrink)。

- `ElicitSlot` - Amazon Lex 預期目前意圖的槽值。

  例如，假設 Amazon Lex 在回應中傳送此訊息：「您想要什麼大小的比薩？」。使用者可能會以槽值 （例如 "medium") 回覆。使用者也可能在回應中提供其他資訊 （例如，「中厚餅皮比薩」)。Amazon Lex 可以適當地處理這類額外資訊。

- `Fulfilled` - 傳達為意圖設定的 Lambda 函數已成功滿足意圖。
- `ReadyForFulfillment` - 傳達用戶端必須滿足意圖。
- `Failed` - 傳達與使用者的對話失敗。

  這可能有多種原因，包括使用者未針對來自 服務的提示提供適當的回應 （您可以設定 Amazon Lex 可提示使用者提供特定資訊的次數），或 Lambda 函數無法滿足意圖。

類型：字串

有效值:`ElicitIntent | ConfirmIntent | ElicitSlot | Fulfilled | ReadyForFulfillment | Failed`

intentName

Amazon Lex 目前知道的使用者意圖。

類型：字串

message

要傳達給使用者的訊息。訊息可能來自機器人的組態或 Lambda 函數。

如果意圖未使用 Lambda 函數設定，或 Lambda 函數傳回`Delegate`做為`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 無法了解使用者輸入，則會使用釐清提示訊息。

建立意圖時，您可以將訊息指派給群組。當訊息指派給群組時，Amazon Lex 會從回應中的每個群組傳回一則訊息。訊息欄位是包含訊息的逸出 JSON 字串。如需傳回之 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 會在回應中將其傳遞給用戶端。

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` - 訊息包含純 UTF-8 文字。
- `CustomPayload` - 訊息是由 Lambda 函數定義的自訂格式。
- `SSML` - 訊息包含語音輸出的文字格式。
- `Composite` - 訊息包含逸出的 JSON 物件，其中包含訊息在建立意圖時指派給群組的一或多個訊息。

類型：字串

有效值:`PlainText | CustomPayload | SSML | Composite`

nluIntentConfidence

提供分數，指出 Amazon Lex 對傳回意圖是符合使用者意圖的意圖的可信度。分數介於 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 的改進而變更。

類型：IntentConfidence 物件

responseCard

代表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，選擇槽旁的設定按鈕）或程式碼掛勾 (Lambda 函數）。

類型：ResponseCard 物件

sentimentResponse

以 和表達用語表示的情緒。

當機器人設定為將表達用語傳送至 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

代表工作階段特定內容資訊的鍵值對映射。

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 從對話中的使用者輸入偵測到的意圖槽。

Amazon Lex 會建立解析度清單，其中包含槽的可能值。傳回的值取決於建立或更新槽類型時valueSelectionStrategy選取的 。如果 valueSelectionStrategy 設定為 ORIGINAL_VALUE，則會傳回使用者提供的值，如果使用者值與槽值類似。如果 valueSelectionStrategy 設定為 TOP_RESOLUTION Amazon Lex，則會傳回解析清單中的第一個值，或者如果沒有解析清單，則為 null。如果您未指定 valueSelectionStrategy，則預設值為 ORIGINAL_VALUE。

類型：字串到字串映射

slotToElicit

如果dialogState值為 ElicitSlot， 會傳回 Amazon Lex 為其引出值的槽名稱。

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) 失敗並發生內部服務錯誤。

HTTP 狀態碼：502

BadRequestException

請求驗證失敗、內容中沒有可用的訊息，或機器人建置失敗、仍在進行中，或包含未建置的變更。

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 ID。

HTTP 狀態碼：409

DependencyFailedException

AWS Lambda 或 Amazon Polly 等其中一個相依性擲出例外狀況。例如

- 如果 Amazon Lex 沒有足夠的許可來呼叫 Lambda 函數。
- 如果 Lambda 函數需要超過 30 秒才能執行。
- 如果履行 Lambda 函數傳回`Delegate`對話方塊動作，而不移除任何槽值。

HTTP 狀態碼：424

InternalFailureException

內部服務錯誤。重試 呼叫。

HTTP 狀態碼：500

LimitExceededException

超過限制。

HTTP 狀態碼：429

LoopDetectedException

不會使用此例外狀況。

HTTP 狀態碼：508

NotFoundException

找不到參考的資源 （例如 Amazon Lex 機器人或別名）。

HTTP 狀態碼：404

另請參閱

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列界面 V2
- AWS 適用於 .NET V4 的 SDK
- AWS 適用於 C++ 的 SDK
- AWS 適用於 Go 的 SDK v2
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 JavaScript V3 的 SDK
- AWS 適用於 Kotlin 的 SDK
- AWS 適用於 PHP V3 的 SDK
- AWS 適用於 Python 的 SDK
- AWS 適用於 Ruby 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。使用此操作讓您的應用程式能夠設定機器人的狀態。

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

根據此欄位的值，Amazon Lex 在回應中傳回的訊息可以是文字或語音。

- 如果值為 `text/plain; charset=utf-8`，Amazon Lex 會在回應中傳回文字。
- 如果值以 開頭`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`做為值，Amazon Lex 會以 MPEG 格式傳回語音。
- 如果值為 `audio/pcm`，則會`audio/pcm`以 16 位元、小端數格式傳回語音。
- 以下是可接受的值：
  - `audio/mpeg`
  - `audio/ogg`
  - `audio/pcm`
  - `audio/*`（預設為 mpeg)
  - `text/plain; charset=utf-8`

botAlias

用於包含工作階段資料的機器人的別名。

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 ID。Amazon Lex 使用此項目來識別使用者與您的機器人的對話。

長度限制：長度下限為 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

用於請求的作用中內容清單。在滿足先前的意圖時，或藉由在請求中包含內容，即可啟用內容。

如果您未指定內容清單，Amazon Lex 將使用工作階段的目前內容清單。如果您指定空白清單，工作階段的所有內容都會清除。

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

設定機器人為了完成對話而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，在意圖上設定檢查點標籤，並修改意圖的屬性。您也可以使用它來移除或新增意圖摘要物件至清單。

您修改或新增至清單的意圖必須對機器人有意義。例如，意圖名稱必須對機器人有效。您必須為下列項目提供有效的值：

- `intentName`

- 槽名稱
- slotToElict

如果您在PutSession請求中傳送 recentIntentSummaryView 參數，新摘要檢視的內容會取代舊的摘要檢視。例如，如果GetSession請求在摘要檢視中傳回三個意圖，而您在摘要檢視中PutSession呼叫一個意圖，則對 的下一個呼叫GetSession只會傳回一個意圖。

類型：IntentSummary 物件陣列

陣列成員：項目數下限為 0。最多 3 個項目。

必要：否

sessionAttributes

代表工作階段特定內容資訊的鍵/值對映射。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

工作階段的作用中內容清單。

contentType

請求中 Accept HTTP 標頭中指定的內容類型。

dialogState

- ConfirmIntent - Amazon Lex 預期在滿足意圖之前有「是」或「否」回應以確認意圖。
- ElicitIntent - Amazon Lex 想要引出使用者的意圖。
- ElicitSlot - Amazon Lex 預期目前意圖的槽值。
- Failed - 傳達與使用者的對話失敗。這可能由於各種原因而發生，包括使用者未對來自 服務的提示提供適當的回應，或者 Lambda 函數無法滿足意圖。
- Fulfilled - 傳達 Lambda 函數已成功滿足意圖。
- ReadyForFulfillment - 傳達用戶端必須滿足意圖。

有效值:ElicitIntent | ConfirmIntent | ElicitSlot | Fulfilled | ReadyForFulfillment | Failed

encodedMessage

應該向使用者顯示的下一個訊息。

encodedMessage 欄位以 base-64 編碼。您必須先解碼 欄位，才能使用 值。

長度限制：長度下限為 1。長度上限為 1366。

intentName

目前意圖的名稱。

message

此標頭已棄用。

應該向使用者顯示的下一個訊息。

您只能在 de-DE、en-AU、en-GB、en-US、es-419、es-ES、es-US、fr-CA、fr-FR 和 it-IT 區域中使用此欄位。在所有其他地區中， message 欄位為 null。您應該改用 encodedMessage 欄位。

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 - 訊息包含純 UTF-8 文字。
- CustomPayload - 訊息是用戶端的自訂格式。
- SSML - 訊息包含語音輸出的文字格式。
- Composite - 訊息包含逸出的 JSON 物件，其中包含訊息在建立意圖時指派給群組的一或多個訊息。

有效值:PlainText | CustomPayload | SSML | Composite

sessionAttributes

代表工作階段特定內容資訊的鍵/值對映射。

sessionId

工作階段的唯一識別碼。

slots

Amazon Lex 在對話期間從使用者輸入偵測到的零或多個意圖槽映射。

Amazon Lex 會建立解析度清單，其中包含槽的可能值。傳回的值取決於建立或更新槽類型時valueSelectionStrategy選取的 。如果 valueSelectionStrategy 設定為 ORIGINAL_VALUE，則會傳回使用者提供的值，如果使用者值與槽值類似。如果 valueSelectionStrategy 設定為 TOP_RESOLUTION Amazon Lex，則會傳回解析清單中的第一個值，或者如果沒有解析清單，則為 null。如果您未指定 valueSelectionStrategy，則預設值為 ORIGINAL_VALUE。

slotToElicit

如果 dialogState是 ElicitSlot，則 會傳回 Amazon Lex 為其引出值的插槽名稱。

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) 失敗並發生內部服務錯誤。

HTTP 狀態碼：502

BadRequestException

請求驗證失敗、內容中沒有可用的訊息，或機器人建置失敗、仍在進行中，或包含未建置的變更。

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 ID。

HTTP 狀態碼：409

DependencyFailedException

AWS Lambda 或 Amazon Polly 等其中一個相依性擲出例外狀況。例如

- 如果 Amazon Lex 沒有足夠的許可來呼叫 Lambda 函數。
- 如果 Lambda 函數需要超過 30 秒才能執行。
- 如果履行 Lambda 函數傳回`Delegate`對話方塊動作，而不移除任何槽值。

HTTP 狀態碼：424

InternalFailureException

內部服務錯誤。重試 呼叫。

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。

HTTP 狀態碼：406

NotFoundException

找不到參考的資源（例如 Amazon Lex 機器人或別名）。

HTTP 狀態碼：404

另請參閱

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 命令列界面 V2
- AWS 適用於 .NET V4 的 SDK
- AWS 適用於 C++ 的 SDK
- AWS 適用於 Go 的 SDK v2
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 JavaScript V3 的 SDK
- AWS 適用於 Kotlin 的 SDK
- AWS 適用於 PHP V3 的 SDK
- AWS 適用於 Python 的 SDK
- AWS 適用於 Ruby V3 的 SDK

資料類型

Amazon Lex Model Building Service 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 Runtime Service 支援下列資料類型：

- ActiveContext
- ActiveContextTimeToLive

- Button
- DialogAction
- GenericAttachment
- IntentConfidence
- IntentSummary
- PredictedIntent
- ResponseCard
- SentimentResponse

Amazon Lex 模型建置服務

Amazon Lex Model Building Service 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

長度限制：長度下限為 2。長度上限為 50。

模式：`^([A-Za-z]_?)+$`

必要：否

botVersion

別名指向的 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

機器人別名的檢查總和。

類型：字串

必要：否

conversationLogs

決定 Amazon Lex 如何使用別名對話日誌的設定。

類型：ConversationLogsResponse 物件

必要：否

createdDate

機器人別名的建立日期。

類型：Timestamp

必要：否

description

機器人別名的描述。

類型：字串

長度限制：長度下限為 0。長度上限為 200。

必要：否

lastUpdatedDate

機器人別名的更新日期。當您建立資源時，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

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 適用於 C++ 的 SDK
- AWS 適用於 Java V2 的 SDK

- AWS 適用於 Ruby V3 的 SDK

BotChannelAssociation

服務：Amazon Lex Model Building Service

代表 Amazon Lex 機器人與外部訊息平台之間的關聯。

目錄

botAlias

指向要建立此關聯的 Amazon Lex 機器人特定版本的別名。

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

映射項目：最多 10 個項目。

必要：否

botName

正在建立此關聯的 Amazon Lex 機器人名稱。

> **Note**
>
> 目前，Amazon Lex 支援與 Facebook、Slack 和 Twilio 的關聯。

類型：字串

長度限制：長度下限為 2。長度上限為 50。

模式：`^([A-Za-z]_?)+$`

必要：否

createdDate

Amazon Lex 機器人與頻道之間的關聯建立日期。

類型：Timestamp

必要：否

description

您要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

機器人與頻道之間的關聯名稱。

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

機器人頻道的狀態。

- `CREATED` - 頻道已建立並可供使用。
- `IN_PROGRESS` - 頻道建立進行中。
- `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

透過指出 Amazon Lex 機器人與外部訊息平台之間建立的頻道類型，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

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱以下內容：

- AWS 適用於 C++ 的 SDK
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 Ruby V3 的 SDK

BotMetadata

服務：Amazon Lex Model Building Service

提供機器人的相關資訊。。

目錄

createdDate

機器人建立的日期。

類型：Timestamp

必要：否

description

機器人的說明。

類型：字串

長度限制：長度下限為 0。長度上限為 200。

必要：否

lastUpdatedDate

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

長度限制：長度下限為 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，版本一律為 $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

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱以下內容：

- AWS 適用於 C++ 的 SDK
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 Ruby 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

內建意圖的唯一識別符。若要尋找意圖的簽章，請參閱 Alexa Skills Kit 中的標準內建意圖。

類型：字串

必要：否

supportedLocales

意圖支援之區域性的識別符清單。

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

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱以下內容：

- AWS 適用於 C++ 的 SDK
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 Ruby V3 的 SDK

BuiltinIntentSlot

服務：Amazon Lex Model Building Service

提供用於內建意圖的槽的相關資訊。

目錄

name

為意圖定義的槽清單。

類型：字串

必要：否

另請參閱

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 適用於 C++ 的 SDK
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 Ruby V3 的 SDK

BuiltinSlotTypeMetadata

服務：Amazon Lex Model Building Service

提供內建插槽類型的相關資訊。

目錄

signature

內建插槽類型的唯一識別符。若要尋找槽類型的簽章，請參閱 Alexa Skills Kit 中的槽類型參考。

類型：字串

必要：否

supportedLocales

槽的目標地區設定清單。

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

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱以下內容：

- AWS 適用於 C++ 的 SDK
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 Ruby V3 的 SDK

CodeHook

服務：Amazon Lex Model Building Service

指定 Lambda 函數，可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。長度上限為 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

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱以下內容：

- AWS 適用於 C++ 的 SDK
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 Ruby V3 的 SDK

ConversationLogsRequest

服務：Amazon Lex Model Building Service

提供對話日誌所需的設定。

目錄

iamRoleArn

IAM 角色的 Amazon Resource Name (ARN)，具有寫入文字日誌的 CloudWatch Logs 和音訊日誌的 S3 儲存貯體的許可。如果啟用音訊加密，此角色也會為用於加密音訊日誌的 AWS KMS 金鑰提供存取許可。如需詳細資訊，請參閱建立對話日誌的 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

對話日誌的設定。您可以記錄對話文字、對話音訊或兩者。

類型：LogSettingsRequest 物件陣列

必要：是

另請參閱

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱以下內容：

- AWS 適用於 C++ 的 SDK
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 Ruby V3 的 SDK

ConversationLogsResponse

服務：Amazon Lex Model Building Service

包含對話日誌設定的相關資訊。

目錄

iamRoleArn

用於將日誌寫入 CloudWatch Logs 或 S3 儲存貯體的 IAM 角色的 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:[\w\-]+:iam::[\d]{12}:role/.+$`

必要：否

logSettings

對話日誌的設定。您可以記錄文字、音訊或兩者。

類型：LogSettingsResponse 物件陣列

必要：否

另請參閱

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 適用於 C++ 的 SDK
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 Ruby V3 的 SDK

EnumerationValue

服務：Amazon Lex Model Building Service

每個位置類型都可以有一組值。每個列舉值代表槽類型可以採用的值。

例如，比薩排序機器人可以有插槽類型，指定比薩應擁有的地殼類型。槽類型可以包含值

- 厚
- thin
- 填充

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

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱以下內容：

- AWS 適用於 C++ 的 SDK
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 Ruby V3 的 SDK

FollowUpPrompt

服務：Amazon Lex Model Building Service

滿足意圖之後的其他活動的提示。例如，在滿足`OrderPizza`意圖之後，您可能會提示使用者了解使用者是否想要訂購飲料。

目錄

prompt

提示來自使用者的資訊。

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

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱以下內容：

- AWS 適用於 C++ 的 SDK
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 Ruby V3 的 SDK

FulfillmentActivity

服務：Amazon Lex Model Building Service

描述在使用者提供意圖所需的所有資訊後，如何實現意圖。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，讓相關的邏輯存在於 雲端中，並將用戶端程式碼主要限制為 呈現。如果您需要更新邏輯，您只需更新 Lambda 函數；您不需要升級用戶端應用程式。

請考量下列範例：

- 在比薩訂購應用程式中，在使用者提供下訂單的所有資訊之後，您可以使用 Lambda 函數來下單搭配比薩。
- 在遊戲應用程式中，當使用者說「撿起石頭」時，此資訊必須返回用戶端應用程式，以便執行操作並更新圖形。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

應如何透過執行 Lambda 函數或將槽資料傳回用戶端應用程式來滿足意圖。

類型：字串

有效值:`ReturnIntent | CodeHook`

必要：是

codeHook

為滿足意圖而執行的 Lambda 函數的描述。

類型：CodeHook 物件

必要：否

另請參閱

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 適用於 C++ 的 SDK
- AWS 適用於 Java V2 的 SDK

- AWS 適用於 Ruby V3 的 SDK

InputContext

服務：Amazon Lex Model Building Service

必須為作用中的內容名稱，Amazon Lex 才能選取意圖。

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

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱以下內容：

- AWS 適用於 C++ 的 SDK
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 Ruby 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

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱以下內容：

- AWS 適用於 C++ 的 SDK
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 Ruby V3 的 SDK

IntentMetadata

服務：Amazon Lex Model Building Service

提供意圖的相關資訊。

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

意圖更新的日期。當您建立意圖時，建立日期和上次更新日期相同。

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

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 適用於 C++ 的 SDK
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 Ruby V3 的 SDK

KendraConfiguration

服務：Amazon Lex Model Building Service

提供 AMAZON.KendraSearchIntent 意圖的組態資訊。當您使用此意圖時，Amazon Lex 會搜尋指定的 Amazon Kendra 索引，並從索引傳回符合使用者表達式的文件。如需詳細資訊，請參閱 AMAZON.KendraSearchIntent。

目錄

kendraIndex

您希望 AMAZON.KendraSearchIntent 意圖搜尋的 Amazon Kendra 索引的 Amazon Resource Name (ARN)。索引必須與 Amazon Lex 機器人位於相同的帳戶和區域中。如果 Amazon Kendra 索引不存在，當您呼叫 PutIntent操作時，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 Kendra 索引許可的 IAM 角色的 Amazon Resource Name (ARN)。角色必須與 Amazon Lex 機器人位於相同的帳戶和區域中。如果角色不存在，當您呼叫 PutIntent操作時會收到例外狀況。

類型：字串

長度限制：長度下限為 20。長度上限為 2048。

模式：`arn:aws:iam::[0-9]{12}:role/.*`

必要：是

queryFilterString

Amazon Lex 傳送給 Amazon Kendra 的查詢篩選條件，以篩選查詢中的回應。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。

您可以在執行時間使用新的篩選條件字串覆寫此篩選條件字串。

類型：字串

長度限制：長度下限為 0。

必要：否

另請參閱

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱以下內容：

- AWS 適用於 C++ 的 SDK
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 Ruby V3 的 SDK

LogSettingsRequest

服務：Amazon Lex Model Building Service

用於設定對話日誌交付模式和目的地的設定。

目錄

destination

日誌的交付位置。文字日誌會交付至 CloudWatch Logs 日誌群組。音訊日誌會傳送到 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。文字日誌會交付至 CloudWatch Logs 日誌群組。音訊日誌會傳送到 S3 儲存貯體。

類型：字串

有效值:`AUDIO | TEXT`

必要：是

resourceArn

應交付日誌的 CloudWatch Logs 日誌群組或 S3 儲存貯體的 Amazon Resource Name (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 Resource Name (ARN)，用於加密交付至 S3 儲存貯體的音訊日誌。金鑰不適用於 CloudWatch Logs，S3 儲存貯體為選用。

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

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 適用於 C++ 的 SDK
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 Ruby V3 的 SDK

LogSettingsResponse

服務：Amazon Lex Model Building Service

對話日誌的設定。

目錄

destination

傳送日誌的目的地。

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之金鑰的 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:[\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

交付日誌的 CloudWatch Logs 日誌群組或 S3 儲存貯體的 Amazon Resource Name (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

資源字首是您指定包含音訊日誌的 S3 儲存貯體中 S3 物件金鑰的第一個部分。對於 CloudWatch Logs，它是您指定之日誌群組中日誌串流名稱的字首。

類型：字串

長度限制：長度上限為 1024。

必要：否

另請參閱

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱以下內容：

- AWS 適用於 C++ 的 SDK
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 Ruby 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

識別訊息所屬的訊息群組。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

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 適用於 C++ 的 SDK
- AWS 適用於 Java V2 的 SDK

- AWS 適用於 Ruby V3 的 SDK

MigrationAlert

服務：Amazon Lex Model Building Service

提供 Amazon Lex 在遷移期間傳送的提醒和警告的相關資訊。提醒包含如何解決問題的相關資訊。

目錄

details

有關提醒的其他詳細資訊。

類型：字串陣列

必要：否

message

說明發出提醒原因的訊息。

類型：字串

必要：否

referenceURLs

Amazon Lex 文件的連結，說明如何解決提醒。

類型：字串陣列

必要：否

type

提醒的類型。有兩種類型的提醒：

- ERROR - 遷移發生問題，無法解決。遷移會停止。
- WARN - 遷移發生問題，需要手動變更新的 Amazon Lex V2 機器人。遷移會繼續。

類型：字串

有效值:ERROR | WARN

必要：否

另請參閱

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱以下內容：

- AWS 適用於 C++ 的 SDK
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 Ruby V3 的 SDK

MigrationSummary

服務：Amazon Lex Model Building Service

提供將機器人從 Amazon Lex V1 遷移至 Amazon Lex V2 的相關資訊。

目錄

migrationId

Amazon Lex 指派給遷移的唯一識別符。

類型：字串

長度限制條件：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 時COMPLETE，機器人可在 Amazon Lex V2 中使用。可能需要解決提醒和警告，才能完成遷移。

類型：字串

有效值:IN_PROGRESS | COMPLETED | FAILED

必要：否

migrationStrategy

用來執行遷移的策略。

類型：字串

有效值:CREATE_NEW | UPDATE_EXISTING

必要：否

migrationTimestamp

遷移開始的日期和時間。

類型：Timestamp

必要：否

v1BotLocale

作為遷移來源的 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

作為遷移來源的 Amazon Lex V1 機器人名稱。

類型：字串

長度限制：長度下限為 2。長度上限為 50。

模式：^([A-Za-z]_?)+$

必要：否

v1BotVersion

作為遷移來源的 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

Amazon Lex V2 的唯一識別符，是遷移的目的地。

類型：字串

長度限制條件：固定長度為 10。

模式：^[0-9a-zA-Z]+$

必要：否

v2BotRole

Amazon Lex 用來執行 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

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱以下內容：

- AWS 適用於 C++ 的 SDK
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 Ruby V3 的 SDK

OutputContext

服務：Amazon Lex Model Building Service

滿足意圖時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

在 PostContent或 PostText回應中第一次傳送內容後，內容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

對話次數會變成內容應該處於作用中狀態。對話輪換是 PostContent或 PostText請求，以及來自 Amazon Lex 的對應回應。

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

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱以下內容：

- AWS 適用於 C++ 的 SDK
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 Ruby 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 會選擇要用來提示使用者的其中一個訊息。如需詳細資訊，請參閱Amazon Lex：運作方式。

目錄

maxAttempts

提示使用者提供資訊的次數。

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，每個都提供訊息字串及其類型。您可以指定純文字或語音合成標記語言 (SSML) 中的訊息字串。

類型：Message 物件陣列

陣列成員：項目數下限為 1。最多 15 個項目。

必要：是

responseCard

回應卡。Amazon Lex 在 `PostText` API 回應中，於執行時間使用此提示。它會取代回應卡中預留位置的工作階段屬性和槽值。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。長度上限為 50000。

必要：否

另請參閱

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱以下內容：

- AWS 適用於 C++ 的 SDK
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 Ruby V3 的 SDK

ResourceReference

服務：Amazon Lex Model Building Service

描述參考您嘗試刪除之資源的資源。此物件會傳回為ResourceInUseException例外狀況的一部分。

目錄

name

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

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

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱以下內容：

- AWS 適用於 C++ 的 SDK
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 Ruby V3 的 SDK

Slot

服務：Amazon Lex Model Building Service

識別特定槽的版本。

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

槽的說明。

類型：字串

長度限制：長度下限為 0。長度上限為 200。

必要：否

obfuscationSetting

決定對話日誌和儲存的表達式中是否混淆了槽。當您混淆插槽時，值會以大括號 ({}) 中的插槽名稱取代。例如，如果槽名稱為 "full_name"，則混淆值會取代為 "{full_name}"。如需詳細資訊，請參閱插槽混淆 。

類型：字串

有效值:NONE | DEFAULT_OBFUSCATION

必要：否

priority

指示 Amazon Lex 從使用者引出此槽值的順序。例如，如果意圖有兩個優先順序為 1 和 2 的插槽，AWS Amazon Lex 會先為優先順序為 1 的插槽引出值。

如果多個插槽具有相同的優先順序，Amazon Lex 引發值的順序是任意的。

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的槽類型的一組可能回應。使用者從回應卡中選擇選項，而不是使用文字來回覆。

類型：字串

長度限制：長度下限為 1。長度上限為 50000。

必要：否

sampleUtterances

如果您知道使用者可能回應 Amazon Lex 槽值請求的特定模式，您可以提供這些措辭以提高準確性。這是選用的。在大多數情況下，Amazon Lex 能夠了解使用者表達用語。

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

槽的類型，您定義的自訂槽類型，或其中一個內建槽類型。

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

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 用來從使用者引出槽值的提示。

類型：Prompt 物件

必要：否

另請參閱

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 適用於 C++ 的 SDK
- AWS 適用於 Java V2 的開發套件
- AWS 適用於 Ruby V3 的 SDK

SlotDefaultValue

服務：Amazon Lex Model Building Service

槽的預設值。

目錄

defaultValue

槽的預設值。您可以指定下列其中一個選項：

- `#context-name.slot-name` - 內容 "context-name" 中的槽值 "slot-name"。
- `{attribute}` - 工作階段屬性「屬性」的槽值。
- `'value'` - 離散值 "value"。

類型：字串

長度限制：長度下限為 1。長度上限為 202。

必要：是

另請參閱

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 適用於 C++ 的 SDK
- AWS 適用於 Java V2 的開發套件
- AWS 適用於 Ruby V3 的 SDK

SlotDefaultValueSpec

服務：Amazon Lex Model Building Service

包含槽的預設值。當 Amazon Lex 尚未決定槽的值時，會使用預設值。

目錄

defaultValueList

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指定要使用的預設值。

選擇的預設值會根據您在清單中指定的順序進行選取。例如，如果您以該順序指定內容變數和固定值，Amazon Lex 會使用可用的內容變數，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

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 適用於 C++ 的 SDK
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 Ruby V3 的 SDK

SlotTypeConfiguration

服務：Amazon Lex Model Building Service

提供槽類型的組態資訊。

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

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱以下內容：

- AWS 適用於 C++ 的 SDK
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 Ruby V3 的 SDK

SlotTypeMetadata

服務：Amazon Lex Model Building Service

提供槽類型的相關資訊。

目錄

createdDate

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

槽類型更新的日期。當您建立資源時，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

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

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 適用於 C++ 的 SDK
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 Ruby 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。

使用標準規則表達式。Amazon Lex 在規則表達式中支援下列字元：

- A-Z、a-z
- 0-9
- Unicode 字元 ("\ u<Unicode>")

以四位數表示 Unicode 字元，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

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱以下內容：

- AWS 適用於 C++ 的 SDK
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 Ruby V3 的 SDK

Statement

服務：Amazon Lex Model Building Service

將資訊傳遞給使用者的訊息集合。在執行時間，Amazon Lex 會選取要傳遞的訊息。

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。

必要：是

responseCard

在執行時間，如果用戶端使用 PostText API，Amazon Lex 會在回應中包含回應卡。它會取代回應卡中預留位置的所有工作階段屬性和槽值。

類型：字串

長度限制：長度下限為 1。長度上限為 50000。

必要：否

另請參閱

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱以下內容：

- AWS 適用於 C++ 的 SDK
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 Ruby 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金鑰/值對清單。標籤索引鍵和值可以包含 Unicode 字母、數字、空格和下列任何符號：_ . : / = + - @。

目錄

key

標籤的金鑰。金鑰不區分大小寫，而且必須是唯一的。

類型：字串

長度限制：長度下限為 1。長度上限為 128。

必要：是

value

與金鑰相關聯的值。該值可以是空字串，但不能是 null。

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

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 適用於 C++ 的 SDK
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 Ruby V3 的 SDK

UtteranceData

服務：Amazon Lex Model Building Service

提供有關對機器人發出的單一表達式的資訊。

目錄

count

表達式的處理次數。

類型：整數

必要：否

distinctUsers

使用表達式的個體總數。

類型：整數

必要：否

firstUtteredDate

第一次記錄表達式的日期。

類型：Timestamp

必要：否

lastUtteredDate

上次記錄表達式的日期。

類型：Timestamp

必要：否

utteranceString

使用者輸入的文字或音訊剪輯的文字表示。

類型：字串

長度限制：長度下限為 1。長度上限為 2000。

必要：否

另請參閱

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 適用於 C++ 的 SDK
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 Ruby V3 的 SDK

UtteranceList

服務：Amazon Lex Model Building Service

提供已對機器人特定版本發出的表達式清單。清單最多包含 100 個表達用語。

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，其中包含已對機器人發出的表達式相關資訊。物件數量上限為 100。

類型：UtteranceData 物件陣列

必要：否

另請參閱

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱以下內容：

- AWS 適用於 C++ 的 SDK
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 Ruby V3 的 SDK

Amazon Lex 執行期服務

Amazon Lex Runtime Service 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

內容是變數，其中包含使用者與 Amazon Lex 之間對話目前狀態的相關資訊。滿足意圖時，Amazon Lex 可以自動設定內容，也可以使用 `PutText`、`PutContent` 或 `PutSession`操作在執行時間設定內容。

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

目前內容的狀態變數。您可以使用這些值作為後續事件中槽的預設值。

類型：字串到字串映射

映射項目：0 個項目的最小數量。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

內容保持作用中的時間長度或轉數。

類型：ActiveContextTimeToLive 物件

必要：是

另請參閱

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 適用於 C++ 的 SDK
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 Ruby V3 的 SDK

ActiveContextTimeToLive

服務：Amazon Lex Runtime Service

內容保持作用中的時間長度或轉數。

目錄

timeToLiveInSeconds

在 PostContent或 PostText回應中第一次傳送內容後，內容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

對話次數會變成內容應該處於作用中狀態。對話輪換是 PostContent或 PostText請求，以及來自 Amazon Lex 的對應回應。

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

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 適用於 C++ 的 SDK
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 Ruby V3 的 SDK

Button

服務：Amazon Lex Runtime Service

代表要在用戶端平台 (Facebook、Slack 等） 上顯示的選項

目錄

text

使用者在按鈕上可見的文字。

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"。當使用者選擇按鈕時，傳送的值可以是「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

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱以下內容：

- AWS 適用於 C++ 的 SDK
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 Ruby 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，並提供動作發生之內容的相關資訊。使用 DialogAction 資料類型將互動設定為特定狀態，或將互動傳回至先前的狀態。

目錄

type

機器人在與使用者互動時應該採取的下一個動作。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好完成。這是是/否的問題，例如「下訂單？」
- Close - 表示 不會有使用者的回應。例如，陳述式「已下訂單」不需要回應。
- Delegate - 下一個動作是由 Amazon Lex 決定。
- ElicitIntent - 下一個動作是判斷使用者想要履行的意圖。
- ElicitSlot - 下一個動作是從使用者引出槽值。

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 函數無法滿足意圖。
- Fulfilled - 意圖已由與意圖相關聯的 Lambda 函數實現。
- ReadyForFulfillment - 存在意圖所需的所有資訊，且用戶端應用程式已準備好履行意圖。

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

應該向使用者顯示的訊息。如果您未指定訊息，Amazon Lex 將使用為意圖設定的訊息。

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 - 訊息包含純 UTF-8 文字。
- CustomPayload - 訊息是用戶端的自訂格式。
- SSML - 訊息包含語音輸出的文字格式。
- Composite - 訊息包含逸出的 JSON 物件，其中包含一或多個訊息。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

已收集的槽及其值的映射。

類型：字串到字串映射

必要：否

slotToElicit

應從使用者引出的槽名稱。

類型：字串

必要：否

另請參閱

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱以下內容：

- AWS 適用於 C++ 的 SDK
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 Ruby V3 的 SDK

GenericAttachment

服務：Amazon Lex Runtime Service

代表顯示提示時向使用者呈現的選項。它可以是影像、按鈕、連結或文字。

目錄

attachmentLinkUrl

回應卡附件的 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

要向使用者顯示的選項清單。

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

標題下方顯示的字幕。

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

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱以下內容：

- AWS 適用於 C++ 的 SDK
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 Ruby V3 的 SDK

IntentConfidence

服務：Amazon Lex Runtime Service

提供分數，指出 Amazon Lex 對意圖是滿足使用者意圖的可信度。

目錄

score

表示 Amazon Lex 對意圖滿足使用者意圖的信心的分數。範圍介於 0.00 和 1.00 之間。分數越高表示可信度越高。

類型：Double

必要：否

另請參閱

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱以下內容：

- AWS 適用於 C++ 的 SDK
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 Ruby V3 的 SDK

IntentSummary

服務：Amazon Lex Runtime Service

提供有關意圖狀態的資訊。您可以使用此資訊來取得意圖的目前狀態，以便處理意圖，或讓您將意圖傳回其先前的狀態。

目錄

dialogActionType

機器人在與使用者互動時應該採取的下一個動作。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好完成。這是是/否的問題，例如「下訂單？」
- Close - 表示 不會有使用者的回應。例如，「已下訂單」陳述式不需要回應。
- ElicitIntent - 下一個動作是判斷使用者想要履行的意圖。
- ElicitSlot - 下一個動作是從使用者引出槽值。

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。您可以使用此標籤來返回先前的意圖。

使用 GetSessionRequest操作的 checkpointLabelFilter 參數，將操作傳回的意圖篩選給只有指定標籤的意圖。

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之後的意圖狀態。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 會將此值設為 Denied。可能值如下：

- Confirmed - 使用者已回應「是」確認提示，確認意圖已完成且已準備好完成。
- Denied - 使用者已回應「否」確認提示。
- None - 從未提示使用者進行確認；或者，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 函數無法滿足意圖。
- Fulfilled - 意圖已由與意圖相關聯的 Lambda 函數實現。
- ReadyForFulfillment - 存在意圖所需的所有資訊，且用戶端應用程式已準備好履行意圖。

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

已收集的槽及其值的映射。

類型：字串到字串映射

必要：否

slotToElicit

要從使用者引出的下一個槽。如果沒有要引出的槽，則欄位為空白。

類型：字串

必要：否

另請參閱

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱以下內容：

- AWS 適用於 C++ 的 SDK
- AWS 適用於 Java V2 的開發套件
- AWS 適用於 Ruby V3 的 SDK

PredictedIntent

服務：Amazon Lex Runtime Service

Amazon Lex 建議滿足使用者意圖的意圖。包括意圖的名稱、Amazon Lex 對滿足使用者的意圖的信心，以及為意圖定義的槽。

目錄

intentName

Amazon Lex 建議滿足使用者意圖的意圖名稱。

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。

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

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱以下內容：

- AWS 適用於 C++ 的 SDK
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 Ruby 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 會取代可用的工作階段屬性和槽值，然後傳回。回應卡也可以來自 Lambda 函數（`fulfillmentActivity` `dialogCodeHook`和 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

代表選項的連接物件陣列。

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

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱以下內容：

- AWS 適用於 C++ 的 SDK
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 Ruby V3 的 SDK

SentimentResponse

服務：Amazon Lex Runtime Service

表達的情緒。

將機器人設定為傳送表達用語至 Amazon Comprehend 以進行情緒分析時，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

如需在其中一種語言特定 AWS SDKs中使用此 API 的詳細資訊，請參閱下列內容：

- AWS 適用於 C++ 的 SDK
- AWS 適用於 Java V2 的 SDK
- AWS 適用於 Ruby 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

下表說明每個 Amazon Lex 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 訂閱源。

變更	描述	日期
新功能	Amazon Lex 現在支援韓文 (ko-KR) 地區設定。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）地區設定。如需詳細資訊，請參閱 Amazon Lex 中支援的語言。	2021 年 7 月 15 日
新功能	Amazon Lex 現在提供將機器人遷移至 Amazon Lex V2 API 的工具。如需詳細資訊，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) 地區。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。如需詳細資訊，請參閱設定意圖內容。	2020 年 11 月 19 日

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) 區域。如需支援地區設定的完整清單，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 (US) (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（英國）(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。	2023 年 9 月 8 日
新功能	Amazon Lex 現在有 7 種新的內建意圖和 9 種新的內建插槽類型。如需詳細資訊，請參閱內建意圖和插槽類型。	2023 年 9 月 8 日
新範例	了解如何建立 Amazon Lex 機器人，客戶支援代理程式可以使用該機器人搜尋 Amazon Kendra 的答案來回答客戶的問題。如需詳細資訊，請參閱範例：客服中心客服人員助理。	2020 年 8 月 10 日
新功能	Amazon Lex 現在可根據可信度分數傳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	Amazon Lex 現已在亞太區域（東京）(ap-northeast-1) 推出。	2020 年 6 月 30 日
新功能	Amazon Lex 現在支援搜尋 Amazon Kendra 索引，以取得常見問題的答案。如需詳細資訊，請參閱 AMAZON.KendraSearchIntent。	2020 年 6 月 11 日
新功能	Amazon Lex 現在會在對話日誌中傳回更多資訊。如需詳細資訊，請參閱在 Amazon CloudWatch Logs 中檢視文字日誌。	2020 年 6 月 9 日
區域擴展	Amazon Lex 現已在亞太區域（新加坡）(ap-southeast-1)、歐洲（法蘭克福）(eu-central-1) 和歐洲（倫敦）(eu-west-2) 推出。	2020 年 4 月 23 日
新功能	Amazon Lex 現在支援標記。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 現在支援 AMAZON.AlphaNumeric 內建插槽類型的規則表達式。如需詳細資訊，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，並在這些日誌中混淆槽值。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	Amazon Lex 現已在亞太區域（雪梨）(ap-southeast-2) 推出。	2019 年 12 月 17 日
新功能	Amazon Lex 現在符合 HIPAA 規範。如需詳細資訊，請參閱 Amazon Lex 的合規驗證。	2019 年 12 月 10 日
新功能	Amazon Lex 現在可將使用者表達用語傳送至 Amazon Comprehend，以分析表達用語的情緒。如需詳細資訊，請參閱 情緒分析。	2019 年 11 月 21 日
新功能	Amazon Lex 現在符合 SOC 規範。如需詳細資訊，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 規範。如需詳細資訊，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	Amazon Lex 現已在美國西部（奧勒岡）(us-west-2) 推出。	2018 年 5 月 8 日
新功能	新增支援以 Amazon Lex 格式匯出和匯入。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	Amazon Lex 現已在歐洲（愛爾蘭）(eu-west-1) 推出。	2017 年 11 月 21 日
新功能	新增在 Kik 上部署 Amazon Lex 機器人的支援。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。如需詳細資訊，請參閱使用 AWS CloudTrail 日誌監控 Amazon Lex API 呼叫。	2017 年 8 月 15 日

展開的文件	新增 的入門範例 AWS CLI。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	這是 Amazon Lex 使用者指南的第一個版本。	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。