aws

V2 開發者指南

Amazon Lex

Amazon Lex: V2 開發者指南

Table of Contents

什麼是亞馬遜萊克斯 V2？

Amazon Lex V2 是一種 AWS 服務，適用於使用語音和文字為應用程式建立交談界面。Amazon Lex V2 提供自然語言理解 (NLU) 和自動語音辨識 (ASR) 的深入功能和彈性，因此您可以透過逼真的對話互動建立高度互動的使用者體驗，並建立新類別的產品。

Amazon Lex V2 可讓任何開發人員快速建置交談機器人。有了 Amazon Lex V2，就不需要深度學習專業知識 — 若要建立機器人，您可以在 Amazon Lex V2 主控台中指定基本的對話流程。Amazon Lex V2 可管理對話方塊，並動態調整交談中的回應。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 V2 提供整合功能AWS Lambda，您可以與 AWS 平台上的許多其他服務整合，包括亞馬遜連接、亞馬遜理解和亞馬遜肯德拉。與 Lambda 整合可讓機器人存取預先建立的無伺服器企業連接器，以連結至 SaaS 應用程式 (例如 Salesforce) 中的資料。

對於 2022 年 8 月 17 日之後建立的機器人，您可以使用條件式分支來控制與機器人的對話流程。您可以使用條件式分支建立複雜的交談，而不需要撰寫 Lambda 程式碼。

亞馬遜萊克斯 V2 提供以下優點：

- 簡單性 — Amazon Lex V2 可引導您使用主控台，在幾分鐘內建立自己的機器人。您提供了幾個範例片語，Amazon Lex V2 會建立完整的自然語言模型，機器人可以透過該模型使用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。

- 民主化的深度學習技術 — Amazon Lex V2 提供 ASR 和 NLU 技術來建立語音語言理解 (SLU) 系統。透過 SLU，Amazon Lex V2 採用自然語言語音和文字輸入、瞭解輸入背後的意圖，並透過叫用適當的商務功能來滿足使用者意圖。

  語音辨識和自然語言理解是電腦科學中最難解決的問題，需要針對大量資料和基礎架構進行訓練，複雜的深度學習演算法才能進行訓練。Amazon Lex V2 讓所有開發人員都能觸及深度學習技術。Amazon Lex V2 機器人可將傳入的語音轉換為文字，並瞭解使用者產生智慧型回應的意圖，以便您可以專注於為客戶建置具有附加價值的機器人，並定義透過交談界面實現的全新產品類別。

- 無縫部署和擴展 — 使用 Amazon Lex V2，您可以直接從 Amazon Lex V2 主控台建置、測試和部署機器人。Amazon Lex V2 可讓您發佈語音或文字機器人，以便在行動裝置、網路應用程式和聊天服務 (例如 Facebook 即時通) 上使用。亞馬遜萊克斯 V2 自動擴展。您不必擔心佈建硬體和管理基礎架構，為您的機器人體驗提供支援。

- 與 AWS 平台的內建整合 — Amazon Lex V2 可與其他 AWS 服務 (例如AWS Lambda和亞馬遜 CloudWatch) 原生運作。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 — 有了 Amazon Lex V2，就沒有前期成本或最低費用。您只需就發出的文字或語音請求付費。定pay-as-you-go價和每個請求的低成本使服務成為構建對話界面的具有成本效益的方式。使用 Amazon Lex V2 免費方案，您可以輕鬆試用 Amazon Lex V2，無需任何初始投資。

支付亞馬遜萊克斯

Amazon Lex V2 只會針對您發出的文字或語音請求向您收取費用。此模型提供可變成本的服務，可隨著業務成長，同時為您提供 AWS 基礎設施的成本優勢。如需詳細資訊，請參閱 Amazon 萊克斯定價。

當您註冊時AWS，您的AWS帳戶會自動註冊中的所有服務AWS，包括 Amazon Lex。不過，您只需針對所使用的服務付費。如果您是亞馬遜萊克斯的新客戶，您可以免費開始使用亞馬遜萊克斯。如需詳細資訊，請參閱 AWS 免費方案。

若要查看您的帳單，請前往 AWS Billing and Cost Management 主控台中的帳單與成本管理儀表板。若要進一步了解 AWS 帳戶 帳單，請參閱《AWS Billing 使用者指南》。如果您有關於 AWS 帳單和 AWS 帳戶 的任何問題，請聯絡 AWS Support。

您是亞馬遜 Lex V2 的首次使用者嗎？

如果您是 Amazon Lex V2 的首次使用者，建議您依序閱讀下列各節：

1. 運作方式— 本節介紹 Amazon Lex V2，以及您用來建立聊天機器人的功能。
2. 開始使用 Amazon Lex V2— 在本節中，您可以設定帳戶並測試 Amazon Lex V2。
3. API 參考 — 本節包含有關 API 操作的詳細資訊。

最新功能

探索以下適用於 Amazon Lex V2 的最新功能：

主題

區域支援 AWS GovCloud (美國西部)

Amazon Lex V2 現已在 AWS GovCloud (美國西部) 推出。

- Amazon Lex 端點和配額

Amazon Lex V2 的生成人工智能

Amazon Lex V2 現在可讓您為機器人充分利用 Amazon 基岩的生成人工智慧功能。

- 描述性的機器人
  - 什麼是新的帖子
  - 文件
- 輔助插槽分辨率
  - 什麼是新的帖子
  - 文件
- 話語生成
  - 什麼是新的帖子
  - 文件
- AMAZON.QnAIntent（會話常見問題解答）
  - 什麼是新的帖子
  - 文件
- AWS Machine Learning 部落格文章

亞馬遜。確認是/否/可能/不知道消歧義的內置插槽。

Amazon Lex V2 現在提供AMAZON.Confimation內建插槽，可改善插槽確認的準確性，以及是/否/可能/不知道回應。

- 文件

使用分析衡量業務績效

Amazon Lex V2 現在讓使用者能夠在分析儀表板上檢視意圖和插槽的效能。

- 什麼是新的帖子
- 文件

使用測試工作台評估機器人效

Amazon Lex V2 現在讓使用者能夠建立和執行測試集，以測量機器人效能並改善機器人指標。

- 什麼是新的帖子
- 文件
- AWS Machine Learning 部落格文章

垂直特定機器人範本

Amazon Lex V2 現在為使用者提供預先建立的機器人範本，其中包含對 ready-to-use 話流程，以及訓練資料和對話提示，適用於語音和聊天模式。

- 什麼是新的帖子
- 文件

機器人網路

Amazon Lex V2 現在讓使用者能夠將多個機器人合併到單一網路中，並能夠根據使用者輸入將請求路由到適當的機器人。

- 什麼是新的帖子
- 文件

視覺對話生成器

Amazon Lex V2 現在提供拖放式交談產生器，可在豐富的視覺環境中使用意圖，輕鬆設計和視覺化交談路徑。

- 什麼是新的帖子
- 文件
- AWS Machine Learning 部落格文章

複合槽類型

Amazon Lex V2 現在讓使用者能夠使用邏輯運算式將多個插槽合併到複合插槽中。

- 什麼是新的帖子

- 文件

條件式分支

Amazon Lex V2 現在讓使用者能夠撰寫條件，以更妥善地控制客戶與機器人交談時所採取的路徑。

- 什麼是新的帖子
- 文件

自動聊天機器人設計

Amazon Lex V2 現在為使用者提供從交談記錄自動設計聊天機器人的選項。閱讀以了解使用示例。

- 什麼是新的帖子
- 文件
- AWS Machine Learning 部落格文章
- Amazon Lex 自動 Chatbot 設計器頁面

運行時提示

Amazon Lex V2 現在為使用者提供設定執行時期提示的選項，以改善片語的辨識度，以改善插槽值的引出。

- 什麼是新的帖子
- 文件

自定義詞彙

Amazon Lex V2 現在為使用者提供建立自訂字彙 (可包含適當名詞或網域特定字詞的片語清單) 的選項，讓 Amazon Lex V2 能夠在音訊輸入中辨識。

- 什麼是新的帖子
- 文件

- AWS Machine Learning 部落格文章

語法插槽類型

Amazon Lex V2 現在讓使用者能夠依照語音辨識文法規格 (SRGS) 編寫 XML 格式的文法，以收集對話中的資訊。

- 什麼是新的帖子
- 文件
- AWS Machine Learning 部落格文章

運作方式

Amazon Lex V2 可讓您使用文字或語音界面建立應用程式，以便與使用者交談。以下是與亞馬遜 Lex V2 工作的典型步驟：

1. 建立機器人並新增一或多種語言。配置機器人，使其了解用戶的目標，與用戶進行對話以引出信息，並滿足用戶的意圖。
2. 測試機器人。您可以使用 Amazon Lex V2 主控台提供的測試視窗用戶端。
3. 發佈版本並建立別名。
4. 部署機器人。您可以在自己的應用程式或訊息平台 (例如 Facebook 即時通或 Slack) 上部署機器人

在開始之前，請先熟悉下列 Amazon Lex V2 核心概念和術語：

- 機器人 — 機器人執行自動化任務，例如訂購比薩餅，預訂酒店，訂購鮮花等。Amazon Lex V2 機器人由自動語音辨識 (ASR) 和自然語言理解 (NLU) 功能提供支援。

  Amazon Lex V2 機器人可以理解以文字或語音提供的使用者輸入，以及交談自然語言。
- 語言 — Amazon Lex V2 機器人可以使用一或多種語言進行交談。每種語言都獨立於其他語言，您可以將 Amazon Lex V2 設定為使用者使用原生字詞和片語與使用者交談。如需詳細資訊，請參閱亞馬遜萊克斯 V2 支援的語言和語言環境。
- 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，您可能會建立訂購比薩餅和飲料的意圖。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`**。
  - 語音範例 — 使用者可能如何傳達意圖。例如，用戶可能會說「我可以點披薩嗎」或「我想點披薩」。
  - 如何達成意圖 — 您希望在使用者提供必要資訊之後達成意圖的方式。我們建議您建立 Lambda 函數來實現意圖。

    您可以選擇性地設定意圖，讓 Amazon Lex V2 將資訊傳回給用戶端應用程式，以進行必要的履行。

  除了自訂意圖之外，Amazon Lex V2 還提供內建意圖以快速設定機器人。如需詳細資訊，請參閱內建槽。

  Amazon Lex 一律包含每個機器人的後援意圖。只要 Amazon Lex 無法推斷用戶的意圖，就會使用後備意圖。如需詳細資訊，請參閱AMAZON.FallbackIntent。

- 插槽-意圖可以需要零個或多個槽或參數。您將槽新增為意圖組態的一部分。在執行階段，Amazon Lex V2 會提示使用者輸入特定的插槽值。使用者必須先提供所有必要插槽的值，Amazon Lex V2 才能達成意圖。

  例如，`OrderPizza`意圖需要插槽，例如大小，外殼類型和比薩餅數量。對於每個插槽，您可以提供插槽類型以及 Amazon Lex V2 傳送給用戶端的一或多個提示，以向使用者取得值。用戶可以使用包含其他單詞的插槽值進行回復，例如「請大披薩」或「讓我們堅持小」。亞馬遜 Lex V2 仍然了解插槽值。

- 插槽類型 — 每個插槽都有一個類型。您可以創建自己的插槽類型，也可以使用內置插槽類型。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`。

  亞馬遜 Lex V2 還提供內置插槽類型。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，請參閱內建槽。

- 版本 — 版本是作品的編號快照，您可以將其發佈在工作流程的不同部分中使用，例如開發、測試版部署和生產環境。創建版本後，您可以使用製作版本時存在的機器人。建立版本之後，在您繼續處理應用程式時，該版本會保持不變。

- 別名 — 別名是指向特定機器人版本的指標。使用別名，您可以更新用戶端應用程式所使用的版本。例如，您可以將別名指向版本 1 的機器人。當您準備好要更新機器人時，即可發佈版本 2 並將別名變更為指向新的版本。由於您的應用程式是使用別名而非特定版本，所有您的用戶端皆無需進行更新便能獲得新功能。

如需提供 Amazon Lex V2 的AWS區域清單，請參閱 Amazon Lex V2 端點和亞馬遜網路服務一般參考中的配額。

亞馬遜萊克斯 V2 支援的語言和語言環境

Amazon Lex V2 支援各種語言和語言環境。本主題提供支援的語言、支援這些語言的功能，以及改善機器人效能的特定語言指引。

支援的語言和地區設定

Amazon Lex V2 支援下列語言和地區設定。

代碼	語言和地區設定
AR_AE	灣阿拉伯語 (阿拉伯聯合大公國)
加州	加泰羅尼亞語（西班
DE_AT	德文 (奧地利)
德 _ 德	德文 (德國)
EN_AU	英文 (澳洲)
EN_GB	英文 (英國)
恩 _ 在	英文 (印度)
en_US	英文 (美國)
恩 _ZA	英文 (南非)
es_419	西班牙文 (拉丁美洲)
E_ES	西班牙語（西班牙）
美國	西班牙文 (美國)
菲	芬蘭文 (芬蘭)
FR_CA	法文 (加拿大)
FR_FR	法文 (法國)
嗨在	印地語（印度）
它	義大利文 (義大利)
太平紳士	日文 (日本)
科	韓文 (韓國)
NL_NL	荷蘭文 (荷蘭)

代碼	語言和地區設定
否 (_NO)	挪威語（挪威）
PL_PL	波蘭文 (波蘭)
聚四氟乙烯	葡萄牙文 (巴西)
PT_PT	葡萄牙語（葡萄牙）
SV_SE	瑞典語（瑞典）
ZH_CN	普通話 (中國)
ZH_ 香港	粵語 (香港)

Amazon Lex V2 功能支援的語言和語言環境

下表列出僅限於特定語言和地區設定的 Amazon Lex V2 功能。所有其他 Amazon Lex V2 功能均支援所有語言和地區設定。

功能	支援的語言和地區設定
亞馬遜。 AlphaNumeric	除韓文 (ko_kr) 以外的所有語言和地區設定
AMAZON.KendraSearchIntent	英語 (美國) (en_US)
使用自訂詞彙改善語音辨識	英語 (英國) (英國) 英語 (美國) (en_US)
自動聊天機器人設計	英語 (美國) (en_US)
區域可用性	亞太區域 (新加坡) (ap-東南-1) 和非洲 (開普敦) (ap-南-1) 地區不提供下列語言和地區設定： • 海灣阿拉伯語 (阿拉伯聯合大公國) (AR_AE) • 加泰羅尼亞語（西班牙）(CA_ES) • 芬蘭語（芬蘭）(FI_Fi)

功能	支援的語言和地區設定
	• 印地語 (印度) (H_IN) • 荷蘭語（荷蘭）(NL_NL) • 挪威語（挪威）（否） • 波蘭文 (PL_PL) • 葡萄牙語（巴西）(PT_BR) • 葡萄牙語（葡萄牙）(PT_PT) • 瑞典文 (SV_SE) • 普通話 (中華人民共和國) • 粵語 (香港) (香港)
設定意圖上下文	英語 (美國) (en_US)
語法插槽類型	英語（澳大利亞）(en_au) 英語 (英國) (英國) 英語 (美國) (en_US)
在插槽中使用多個值	英語 (美國) (en_US)
使用執行時期提示改善位置值的辨識	英語 (英國) (英國) 英語 (美國) (en_US)
使用拼字樣式擷取位置值	英語（澳大利亞）(en_au) 英語 (英國) (英國) 英語 (美國) (en_US)
使用可信度分數	英語 (英國) (英國) 英語 (美國) (en_US)

亞馬遜萊克斯 V2 的語言指導

若要改善機器人的效能，您應該遵守下列語言的這些準則。

阿拉伯文

亞馬遜萊克斯 V2 訓練的阿拉伯語的品種是海灣阿拉伯語。為您的機器人提供範例語音時，請記住這一點。請注意，阿拉伯語腳本是從右到左編寫的。

印地語

Amazon Lex V2 能夠為印地文和英文之間自由切換的印地文終端使用者提供服務。如果您打算建置支援此語言切換的機器人，我們建議您採用下列最佳作法：

- 在 bot 定義中，用拉丁語腳本寫英語單詞。
- 至少 50% 的範例語音應代表同一句子中的語言切換。在這些話語中，使用梵文腳本作為印地語單詞，使用拉丁文字本作為英語單詞（例如，「 「)。
- 如果您希望用戶在拉丁文腳本中使用印地語單詞或戴納加里語腳本中的英語單詞與機器人進行交流，則應在拉丁文腳本中包含印地語單詞的示例（例如，「mujhe ek 票務簿 karni hai」）和德瓦納加里腳本中的英語單詞（例如「」在您的示例語音中。
- 如果您希望用戶使用完全印地語或完全使用英語的句子與機器人進行交流，那麼您應該包含完全使用一種語言的樣本語言（例如，「我想預訂票證」）。

區域

如需提供 Amazon Lex V2 的AWS區域清單，請參閱中的 AWS 區域和端點AWS 一般參考。

開始使用 Amazon Lex V2

Amazon Lex V2 提供可與現有應用程式整合的 API 操作。如需支援的作業清單，請參閱 API 參考。您可以使用下列任一選項：

- AWS 開發套件 — 使用開發套件時，系統會使用您提供的登入資料自動簽署和驗證 Amazon Lex V2 的請求。我們建議您使用 SDK 來建置應用程式。
- AWS CLI — 您可以使用存取任何 Amazon Lex V2 功能，而不必撰寫任何程式碼。 AWS CLI
- AWS 主控台 — 主控台是開始測試和使用 Amazon Lex V2 的最簡單方法

如果您是 Amazon Lex V2 的新手，我們建議您運作方式先閱讀。

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 V2 之前，請先完成下列工作：

1. 註冊成為 AWS
2. 建立 IAM 使用者

註冊成為 AWS

如果您已經有 AWS 帳戶，請跳過此任務。

當您註冊 Amazon Web Services (AWS) 時，您的 AWS 帳戶會自動註冊中的所有服務 AWS，包括 Amazon Lex V2。您只需支付實際使用服務的費用。

使用 Amazon Lex V2 時，您只需為使用的資源付費。如果您是新 AWS 客戶，可以免費開始使用 Amazon Lex V2。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，請跳至下一個工作。如果您沒有 AWS 帳戶，請使用下列步驟建立帳戶。

建立 AWS 帳號

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

   部分註冊程序需接收來電，並在電話鍵盤輸入驗證碼。

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

記下您的 AWS 帳戶 ID，因為下一個任務將需要它。

建立 IAM 使用者

中 AWS的服務 (例如 Amazon Lex V2) 需要您在存取登入資料時提供登入資料，以便服務判斷您是否有權存取該服務所擁有的資源。

建立 IAM 使用者帳戶以存取 Amazon Lex V2 的帳戶：

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

然後，您可以 AWS 使用特殊的 URL 和 IAM 使用者的登入資料進行存取。

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

如需 IAM 的詳細資訊，請參閱下列各項：

- AWS Identity and Access Management (IAM)

- 入門
- IAM 使用者指南

授與程式設計存取權

如果使用者想要與 AWS 之外的 AWS Management Console. 授與程式設計存 AWS取權的方式取決於正在存取的使用者類型。

若要授與使用者程式設計存取權，請選擇下列其中一個選項。

哪個使用者需要程式設計存取權？	到	By
人力身分 (IAM Identity Center 中管理的使用者)	使用臨時登入資料來簽署對 AWS CLI、 AWS SDK 或 AWS API 的程式設計要求。	請依照您要使用的介面所提供的指示操作。 • 有關 AWS CLI，請參閱《使用AWS Command Line Interface 者指南》 AWS IAM Identity Center中的〈配置使用〉。 AWS CLI • 如需 AWS SDK、工具和 AWS API，請參閱 AWS SDK 和工具參考指南中的 IAM 身分中心身分驗證。
IAM	使用臨時登入資料來簽署對 AWS CLI、 AWS SDK 或 AWS API 的程式設計要求。	遵循《IAM 使用者指南》中的〈將臨時登入資料搭配 AWS 資源使用〉中的指示
IAM	(不建議使用) 使用長期認證簽署對 AWS CLI、 AWS SDK 或 AWS API 的程式設計要求。	請依照您要使用的介面所提供的指示操作。 • 如需相關資訊 AWS CLI，請參閱使用指南中的使用 IAM 使用者登入資料進行驗證。AWS Command Line Interface

哪個使用者需要程式設計存取權？	到	By
		• 對於 AWS SDK 和工具，請參閱 AWS SDK 和工具參考指南中的使用長期憑據進行身份驗證。 • 如需 AWS API，請參閱 IAM 使用者指南中的管理 IAM 使用者存取金鑰。

下一步驟

步驟 2：開始使用（主控台）

步驟 2：開始使用（主控台）

學習如何使用 Amazon Lex V2 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 V2 機器人，該藍圖是一種預先定義的機器人，可提供所有必要的機器人組態。您只需執行最少的工作來測試 end-to-end 設定。
- 練習 2 — 檢閱用戶端應用程式和 Amazon Lex V2 機器人之間傳送的 JSON 結構。

主題

- 練習 1：從範例建立機器人
- 練習 2：檢閱交談流程

練習 1：從範例建立機器人

在本練習中，您會建立第一個 Amazon Lex V2 機器人，並在 Amazon Lex V2 主控台中進行測試。在本練習中，您將使用OrderFlowers範例。

範例概觀

您可以使用此OrderFlowers範例來建立 Amazon Lex V2 機器人。如需機器人結構的詳細資訊，請參閱運作方式。

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

若要建立 Amazon Lex V2 機器人 (主控台)

1. 登錄到AWS Management Console並打開 Amazon Lex 控制台 https://console.aws.amazon.com/lex/.
2. 選擇建立機器人。
3. 對於「建立」方法，請選擇「以範例開始」。
4. 在「範例機器人」區段中，OrderFlowers從清單中選擇。
5. 在「機器人設定」區段中，為機器人提供名稱和選用說明。該名稱在您的帳戶中必須是唯一的。
6. 在「權限」區段中，選擇「建立具有基本 Amazon Lex 權限的新角色」。這將建立具有 Amazon Lex V2 執行機器人所需許可的 AWS Identity and Access Management (IAM) 角色。
7. 在兒童在線隱私保護法（COPPA）部分中，做出適當的選擇。
8. 在 [工作階段逾時] 和 [進階設定] 區段中，保留預設值。

9. 選擇下一步。Amazon Lex V2 創建您的機器人。

建立機器人之後，您必須新增機器人支援的一或多種語言。一種語言包含機器人用於與用戶交談的意圖，插槽類型和插槽。

若要將語言新增至機器人

1. 在「語言」區段中，選擇支援的語言，然後新增說明。
2. 保留 [語音互動] 和 [意圖] 分類信賴度分數閾值欄位的預設值。
3. 選擇完成將語言添加到機器人。

選擇「完成」後，主控台會開啟意圖編輯器。您可以使用意圖編輯器來檢查機器人使用的意圖。完成檢查機器人後，您可以對其進行測試。

若要測試 OrderFlowers 機器人

1. 選擇頁面頂端的 [建置]。等待機器人建置。
2. 當構建完成時，選擇「測試」以打開測試窗口。
3. 測試機器人。從其中一個樣本的話語開始對話，例如「我想摘花」。

後續步驟

現在您已經使用範本建立了第一個機器人，您可以使用主控台建立自己的機器人。有關創建自定義機器人的說明以及有關創建機器人的更多信息，請參閱建築機器人。

練習 2：檢閱交談流程

在本練習中，您將檢閱用戶端應用程式和您在其中建立的 Amazon Lex V2 機器人之間傳送的 JSON 結構練習 1：從範例建立機器人。交談會使用RecognizeText作業來產生 JSON 結構。會RecognizeUtterance傳回與回應中的 HTTP 標頭相同的資訊。

JSON 結構由對話的每個回合劃分。轉彎是來自客戶端應用程序的請求和來自機器人的響應。

第 1

在對話的第一個回合中，用戶端應用程式會啟動與您的機器人的對話。URI 和請求的本文都提供請求的信息。

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

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

- URI 會識別用戶端應用程式正在與之通訊的機器人。它還包括用戶端應用程式產生的工作階段識別碼，用來識別使用者與機器人之間的特定交談。
- 要求的主體包含使用者輸入至用戶端應用程式的文字。在這種情況下，只會傳送文字，不過您的應用程式可以傳送其他資訊，例如要求屬性或工作階段狀態。有關更多資訊，請參閱 RecognizeText 操作。

從text Amazon Lex V2 偵測使用者的意圖，以訂購鮮花。Amazon Lex V2 選擇其中一個意圖的插槽 (FlowerType) 和插槽的其中一個提示，然後將下列回應傳送給用戶端應用程式。用戶端會顯示對使用者的回應。

```
{
    "interpretations": [
        {
            "intent": {
                "confirmationState": "None",
                "name": "OrderFlowers",
                "slots": {
                    "FlowerType": null,
                    "PickupDate": null,
                    "PickupTime": null
                },
                "state": "InProgress"
            },
            "nluConfidence": {
                "score": 0.95
            }
        },
        {
            "intent": {
                "name": "FallbackIntent",
                "slots": {}
```

```
            }
        }
    ],
    "messages": [
        {
            "content": "What type of flowers would you like to order?",
            "contentType": "PlainText"
        }
    ],
    "sessionId": "bf445a49-7165-4fcd-9a9c-a782493fba5c",
    "sessionState": {
        "dialogAction": {
            "slotToElicit": "FlowerType",
            "type": "ElicitSlot"
        },
        "intent": {
            "confirmationState": "None",
            "name": "OrderFlowers",
            "slots": {
                "FlowerType": null,
                "PickupDate": null,
                "PickupTime": null
            },
            "state": "InProgress"
        },
        "originatingRequestId": "9e8add70-4106-4a10-93f5-2ce2cb959e5f"
    }
}
```

第 2

接著 2，使用者會依次回應 Amazon Lex V2 機器人的提示，並使用填滿FlowerType插槽的值回應。

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

迴轉 2 的回應展示填滿的FlowerType槽，並提供引出下一個槽值的提示。

```
{
    "interpretations": [
```

```
        {
            "intent": {
                "confirmationState": "None",
                "name": "OrderFlowers",
                "slots": {
                    "FlowerType": {
                        "value": {
                            "interpretedValue": "dozen roses",
                            "originalValue": "dozen roses",
                            "resolvedValues": []
                        }
                    },
                    "PickupDate": null,
                    "PickupTime": null
                },
                "state": "InProgress"
            },
            "nluConfidence": {
                "score": 0.98
            }
        },
        {
            "intent": {
                "name": "FallbackIntent",
                "slots": {}
            }
        }
    ],
    "messages": [
        {
            "content": "What day do you want the dozen roses to be picked up?",
            "contentType": "PlainText"
        }
    ],
    "sessionId": "bf445a49-7165-4fcd-9a9c-a782493fba5c",
    "sessionState": {
        "dialogAction": {
            "slotToElicit": "PickupDate",
            "type": "ElicitSlot"
        },
        "intent": {
            "confirmationState": "None",
            "name": "OrderFlowers",
            "slots": {
```

```
                "FlowerType": {
                    "value": {
                        "interpretedValue": "dozen roses",
                        "originalValue": "dozen roses",
                        "resolvedValues": []
                    }
                },
                "PickupDate": null,
                "PickupTime": null
            },
            "state": "InProgress"
        },
        "originatingRequestId": "9e8add70-4106-4a10-93f5-2ce2cb959e5f"
    }
}
```

第 3

接著 3，使用者會依次回應 Amazon Lex V2 機器人的提示，並使用填滿PickupDate插槽的值回應。

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

對於迴轉 3 的回應FlowerType和PickupDate槽都已填滿，並提供引出最後一個槽值的提示。

```
{
    "interpretations": [
        {
            "intent": {
                "confirmationState": "None",
                "name": "OrderFlowers",
                "slots": {
                    "FlowerType": {
                        "value": {
                            "interpretedValue": "dozen roses",
                            "originalValue": "dozen roses",
                            "resolvedValues": []
                        }
                    },
                    "PickupDate": {
```

```
                    "value": {
                        "interpretedValue": "2022-12-28",
                        "originalValue": "next monday",
                        "resolvedValues": [
                            "2021-01-04"
                        ]
                    }
                },
                "PickupTime": null
            },
            "state": "InProgress"
        },
        "nluConfidence": {
            "score": 1.0
        }
    },
    {
        "intent": {
            "name": "FallbackIntent",
            "slots": {}
        }
    }
],
"messages": [
    {
        "content": "At what time do you want the 1 dozen roses to be picked up?",
        "contentType": "PlainText"
    }
],
"sessionId": "bf445a49-7165-4fcd-9a9c-a782493fba5c",
"sessionState": {
    "dialogAction": {
        "slotToElicit": "PickupTime",
        "type": "ElicitSlot"
    },
    "intent": {
        "confirmationState": "None",
        "name": "OrderFlowers",
        "slots": {
            "FlowerType": {
                "value": {
                    "interpretedValue": "dozen roses",
                    "originalValue": "dozen roses",
                    "resolvedValues": []
```

```
                    }
                },
                "PickupDate": {
                    "value": {
                        "interpretedValue": "2021-01-04",
                        "originalValue": "next monday",
                        "resolvedValues": [
                            "2021-01-04"
                        ]
                    }
                },
                "PickupTime": null
            },
            "state": "InProgress"
        },
        "originatingRequestId": "9e8add70-4106-4a10-93f5-2ce2cb959e5f",
        "sessionAttributes": {}
    }
}
```

第 4

第 4 回合，用戶提供意圖的最終插槽值，即花朵被拾取的時間。

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

在回應中，Amazon Lex V2 會傳送確認提示給使用者，以確認訂單是否正確。會dialogAction設定為，ConfirmIntent且confirmationState是None。

```
{
    "interpretations": [
        {
            "intent": {
                "confirmationState": "None",
                "name": "OrderFlowers",
                "slots": {
                    "FlowerType": {
                        "value": {
                            "interpretedValue": "dozen roses",
```

```
                        "originalValue": "dozen roses",
                        "resolvedValues": []
                    }
                },
                "PickupDate": {
                    "value": {
                        "interpretedValue": "2021-01-04",
                        "originalValue": "next monday",
                        "resolvedValues": [
                            "2021-01-04"
                        ]
                    }
                },
                "PickupTime": {
                    "value": {
                        "interpretedValue": "17:00",
                        "originalValue": "5 evening",
                        "resolvedValues": [
                            "17:00"
                        ]
                    }
                }
            },
            "state": "InProgress"
        },
        "nluConfidence": {
            "score": 1.0
        }
    },
    {
        "intent": {
            "name": "FallbackIntent",
            "slots": {}
        }
    }
],
"messages": [
    {
        "content": "Okay, your dozen roses will be ready for pickup by 17:00 on
2021-01-04.  Does this sound okay?",
        "contentType": "PlainText"
    }
],
"sessionId": "bf445a49-7165-4fcd-9a9c-a782493fba5c",
```

```
    "sessionState": {
        "dialogAction": {
            "type": "ConfirmIntent"
        },
        "intent": {
            "confirmationState": "None",
            "name": "OrderFlowers",
            "slots": {
                "FlowerType": {
                    "value": {
                        "interpretedValue": "dozen roses",
                        "originalValue": "dozen roses",
                        "resolvedValues": []
                    }
                },
                "PickupDate": {
                    "value": {
                        "interpretedValue": "2021-01-04",
                        "originalValue": "next monday",
                        "resolvedValues": [
                            "2021-01-04"
                        ]
                    }
                },
                "PickupTime": {
                    "value": {
                        "interpretedValue": "17:00",
                        "originalValue": "5 evening",
                        "resolvedValues": [
                            "17:00"
                        ]
                    }
                }
            },
            "state": "InProgress"
        },
        "originatingRequestId": "9e8add70-4106-4a10-93f5-2ce2cb959e5f"
    }
}
```

第 5

在最後一回合中，使用者會回應確認提示。

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

在回應中，Amazon Lex V2 傳送表示已透過將設定confirmationState為Confirmed和關閉dialogAction來達成意圖。所有插槽值都可供用戶端應用程式使用。

```
{
    "interpretations": [
        {
            "intent": {
                "confirmationState": "Confirmed",
                "name": "OrderFlowers",
                "slots": {
                    "FlowerType": {
                        "value": {
                            "interpretedValue": "dozen roses",
                            "originalValue": "dozen roses",
                            "resolvedValues": []
                        }
                    },
                    "PickupDate": {
                        "value": {
                            "interpretedValue": "2021-01-04",
                            "originalValue": "next monday",
                            "resolvedValues": [
                                "2021-01-04"
                            ]
                        }
                    },
                    "PickupTime": {
                        "value": {
                            "interpretedValue": "17:00",
                            "originalValue": "5 evening",
                            "resolvedValues": [
                                "17:00"
                            ]
                        }
```

```
                    }
                },
                "state": "Fulfilled"
            },
            "nluConfidence": {
                "score": 1.0
            }
        },
        {
            "intent": {
                "name": "FallbackIntent",
                "slots": {}
            }
        }
    ],
    "messages": [
        {
            "content": "Thanks. ",
            "contentType": "PlainText"
        }
    ],
    "sessionId": "bf445a49-7165-4fcd-9a9c-a782493fba5c",
    "sessionState": {
        "dialogAction": {
            "type": "Close"
        },
        "intent": {
            "confirmationState": "Confirmed",
            "name": "OrderFlowers",
            "slots": {
                "FlowerType": {
                    "value": {
                        "interpretedValue": "dozen roses",
                        "originalValue": "dozen roses",
                        "resolvedValues": []
                    }
                },
                "PickupDate": {
                    "value": {
                        "interpretedValue": "2021-01-04",
                        "originalValue": "next monday",
                        "resolvedValues": [
                            "2021-01-04"
                        ]
```

```
                    }
                },
                "PickupTime": {
                    "value": {
                        "interpretedValue": "17:00",
                        "originalValue": "5 evening",
                        "resolvedValues": [
                            "17:00"
                        ]
                    }
                }
            },
            "state": "Fulfilled"
        },
        "originatingRequestId": "9e8add70-4106-4a10-93f5-2ce2cb959e5f"
    }
}
```

建築機器人

您可以建立 Amazon Lex V2 機器人來與使用者互動，以取得資訊以完成任務。例如，您可以創建一個機器人來收集訂購一束鮮花或預訂酒店房間所需的信息。

若要建立機器人，您需要下列資訊：

1. 機器人用來與客戶互動的語言。您可以選擇一或多種語言，每種語言都包含獨立的意圖、插槽和插槽類型。
2. 機器人協助使用者完成的意圖或目標。機器人可以包含一個或多個意圖，例如訂購鮮花，或預訂酒店和租車。您需要決定使用者所做的陳述式或說話來啟動意圖。
3. 您需要從用戶那裡收集以實現意圖的信息或插槽。例如，您可能需要從用戶那裡獲取鮮花類型或酒店預訂的開始日期。您需要定義 Amazon Lex V2 用來向使用者引出插槽值的一或多個提示。
4. 您需要從使用者處取得的插槽類型。您可能需要建立自訂的插槽類型，例如使用者可以訂購的鮮花清單，或者您可以使用內建插槽類型，例如將`AMAZON.Date`插槽類型用於預約的開始日期。
5. 使用者互動會在意圖內部和意圖之間流動。您可以設定交談流程，以便在叫用意圖後定義使用者與機器人之間的互動。您可以建立 Lambda 函數來驗證並完成意圖。

主題

> **Note**
>
> 2022 年 8 月 17 日，Amazon Lex V2 發佈了與使用者對話管理方式的變更。此變更可讓您更好地控制使用者通過交談的路徑。如需詳細資訊，請參閱了解交談流程管理。在 2022 年 8 月 17 日之前建立的機器人不支援對話方塊程式碼掛接訊息、設定值、設定後續步驟和新增條件。

了解交談流程管理

2022 年 8 月 17 日，Amazon Lex V2 發佈了與使用者管理對話方式的變更。此變更可讓您更好地控制使用者通過交談的路徑。

在變更之前，Amazon Lex V2 會根據插槽的意圖優先順序引出對話來管理對話。您可以DialogAction在 Lambda 函數中使用動態修改此行為，並根據使用者輸入變更交談路徑。這可以通過跟踪對話的當前狀態，並以編程方式根據會話狀態決定下一步做什麼來完成。

透過這項變更，您可以使用 Amazon Lex V2 主控台或 API 建立交談路徑和條件式分支，而無需使用 Lambda 函數。Amazon Lex V2 會追蹤交談狀態，並根據建立機器人時定義的條件控制下一步要執行的動作。這使您可以在設計機器人時輕鬆創建複雜的對話。

這些變更可讓您完全控制與客戶的對話。但是，您不需要定義路徑。如果您未指定交談路徑，Amazon Lex V2 會根據意圖中插槽的優先順序建立預設路徑。您可以繼續使用 Lambda 函數來動態定義交談路徑。在這種情況下，交談會根據 Lambda 函數中設定的工作階段狀態繼續進行。

此更新提供下列項目：

- 用於建立具有複雜對話流程的機器人的全新主控台體驗。
- 用於建立機器人的現有 API 的更新，以支援新的對話流程。
- 在意圖調用時發送消息的初始響應。
- 插槽引出的新響應，Lambda 調用作為對話框代碼掛鉤和確認。
- 能夠在對話的每個回合指定後續步驟。
- 評估條件以設計多個對話路徑。
- 在交談期間的任何時間點設定插槽值和工作階段屬性。

對於較舊的機器人請注意以下事項

- 2022 年 8 月 17 日之前建立的機器人會繼續使用舊機制來管理對話流程。在該日期之後建立的機器人會使用新的交談流程管理方式。
- 在 2022 年 8 月 17 日之後透過匯入建立的新機器人會使用新的對話流程管理。現有漫遊器上的匯入會繼續使用舊的對話管理方式。
- 若要為 2022 年 8 月 17 日之前建立的機器人啟用新的對話流程管理，請匯出機器人，然後使用新的機器人名稱匯入機器人。從匯入新建立的機器人會使用新的對話流程管理。

對於 2022 年 8 月 17 日之後建立的新機器人，請注意下列事項：

- Amazon Lex V2 會完全遵循定義的對話流程，以提供所需體驗的設計。您應該設定所有流程分支，以避免在執行階段期間預設交談路徑。
- 應完全設定程式碼掛接後的對話步驟，因為不完整的步驟可能會導致機器人失敗。我們建議您驗證在 2022 年 8 月 17 日之前建立的機器人，因為對於這些機器人，不會在程式碼掛接後自動驗證交談步驟。

建立機器人

您可以透過下列方式使用 Amazon Lex V2 建立機器人：

1. 使用 Amazon Lex V2 主控台，透過網站界面建立機器人。如需詳細資訊，請參閱使用亞馬遜 Lex V2 主控台建立機器人。
2. 使用描述性機器人產生器，使用 Amazon 基岩的生成 AI 功能建立機器人。如需詳細資訊，請參閱使用描述性機器人構建器。
3. 使用機器人範本建立符合常見商業使用案例的預先設定機器人。如需詳細資訊，請參閱從機器人範本產生預先定義的。
4. 使用 AWSSDK 建立使用 API 作業的機器人。
5. 使用自動 Chatbot 設計器，使用客服人員和客戶之間現有的聊天記錄來創建機器人。如需詳細資訊，請參閱使用自動 Chatbot 設計器。
6. 匯入現有的機器人定義。如需詳細資訊，請參閱匯入中。
7. 用AWS CloudFormation於建立機器人。如需詳細資訊，請參閱 使用建立 Amazon Lex V2AWS CloudFormation。

主題

- 使用亞馬遜 Lex V2 主控台建立機器人
- 從機器人範本產生預先定義的
- 使用自動 Chatbot 設計器

使用亞馬遜 Lex V2 主控台建立機器人

通過定義名稱，描述和一些基本信息開始創建您的機器人。

若要建立機器人

1. 登錄到AWS Management Console並打開亞馬遜萊克斯控制台 https://console.aws.amazon.com/lex/.
2. 選擇建立機器人。
3. 在「建立方法」區段中，選擇「建立」。
4. 在「機器人設定」區段中，為機器人提供名稱和選用說明。
5. 在 IAM 許可區段中，選擇提供 Amazon Lex V2 存取其他AWS服務 AWS Identity and Access Management (例如 Amazon) 權限的 (IAM) 角色CloudWatch。您可以讓 Amazon Lex V2 建立角色，也可以選擇具有CloudWatch許可的現有角色。
6. 在《兒童線上隱私保護法》(COPPA) 區段中，選擇適當的回應。
7. 在閒置工作階段逾時區段中，選擇 Amazon Lex V2 在使用者開啟的情況下保持工作階段的持續時間。Amazon Lex V2 會在工作階段期間維護工作階段變數，以便您的機器人可以使用相同的變數繼續交談。
8. 在「進階設定」區段中，新增可協助識別機器人的標籤，並可用來控制存取和監控資源。
9. 選擇「下一步」以創建機器人並轉到添加語言。

從機器人範本產生預先定義的

Amazon Lex V2 提供預先建置的解決方案，可大規模建立體驗並推動數位參與。預先建置的機器人範本可自動化並標準化用戶端體驗。機器人範本提供語音和聊天模式的對ready-to-use話流程以及訓練資料和對話提示。您可以加快機器人解決方案的交付速度，同時最佳化資源，以便專注於客戶關係。

您可以根據您的業務使用案例建立預先建置的機器人。您可以使用主AWS CloudFormation控台為相關服務 (例如 Amazon S3、Amazon Connect 和 DynamoDB) 選取預先建置的選項。

目前，Amazon Lex V2 支援下列業務垂直產業：

- 金融服務
- 零售訂單
- 汽車保險
- 电信
- 航空服務
- 更多即將推出...

您可以使用提供的商務解決方案範本來建置機器人，並根據您的業務需求自訂機器人。

> **Note**
>
> 這些範本會透過AWS CloudFormation堆疊在 Amazon Lex V2 之外建立資源。堆疊可能需要在其他主控台 (例如 Lambda 和 DynamoDB) 中進行修改。

建置和部署機器人範本所需的先決條件：

- 一個 AWS 帳戶
- 訪問以下AWS服務：
  - 亞馬遜萊克斯 V2 創建機器人
  - 適用於商業登入功能的 Lambda
  - 用於建立資料表的動態資料表
  - 建立政策和角色的 IAM 存取權
  - AWSCloudFormation運行堆棧
- IAM 存取和秘密金鑰登入資料
- 亞馬遜連接實例（可選）

> **Note**
>
> 使用不同的AWS服務會產生各自的使用費用。

若要從亞馬遜 Lex V2 範本建立機器人：

1. 登錄到AWS Management Console並打開亞馬遜萊克斯控制台 https://console.aws.amazon.com/lex/.
2. 選擇顯示「從模板創建機器人」的橙色按鈕。
3. 選取您要用於機器人範本的業務垂直。注意：目前有 5 個機器人模板可用。更多即將推出。
4. 為您要使用的範本選取 [建立]。將開啟一個標籤AWS CloudFormation，您可以在其中編輯AWS CloudFormation堆疊的參數。您選擇的範本的所有選項都已完成。您也可以選取 [深入瞭解]，進一步瞭解機器人範本的運作方式。

5. 在AWS CloudFormation主控台中，AWS CloudFormation為所選範本的每個值建立預設組態。您也可以選取自己的堆疊名稱、AWS CloudFormation參數、Amazon DynamoDB 表和 (選用) 亞馬遜連接參數。
6. 在視窗底部，選取「建立堆疊」。
7. AWS CloudFormation在背景處理要求數分鐘，以設定新的機器人。附註：此程序會自動為 DynamoDB 表、Amazon Connect 聯絡流程和 Amazon Connect 執行個體建立資源。您可以在AWS CloudFormation主控台追蹤進度，然後在CloudFormation堆疊建立完成後導覽回 Amazon Lex V2 主控台。
8. 如果成功構建，將顯示一條消息，您可以選擇轉到機器人列表轉到機器人頁面，您可以在其中找到已準備好進行測試和使用的新機器人。

設定您的機器人範本

Lambda 函數 — 機器人範本會自動為您的部署建立所需的 Lambda 函數。如果範本解決方案中有多個機器人，則參數中會列出多個 Lambda 函AWS CloudFormation數。如果您有要與機器人一起部署的現有 Lambda 函數，則可以輸入自訂 Lambda 函數的名稱。

Amazon DynamoDB — 機器人範本會自動建立載入範例政策資料所需的 DynamoDB 表格。您也可以輸入自訂 DynamoDB 表格的名稱。您的自訂 DynamoDB 表格的格式化方式應與機器人範本部署建立的預設表格相同。

亞馬遜連接 — 您可以通過輸入 ConnectInstance ARN 和一個唯ContactFlowName一的，將亞馬遜連接實例配置為與新的機器人模板一起使用。透過使用 Amazon Connect，您可以使用端對端的 IVR 系統來測試您的機器人。

疑難排解機器人範本

- 檢查您是否具有建立所選範本的適當權限。使用者需要CloudFormation：CreateStack權限以及範本中所列AWS資源的權限。需要使用者權限的資源清單位於 [建立範本] 頁面的底部。
- 如果無法建立機器人範本，Amazon Lex V2 主控台內的紅色橫幅會提供負責建立範本之AWS CloudFormation堆疊的連結。在AWS CloudFormation控制台中，您可以查看事件選項卡以查看導致模板失敗的特定錯誤。檢閱AWS CloudFormation錯誤後，請參閱疑難排解以取得CloudFormation詳細資訊。
- 機器人範本僅適用於範例資料。您必須將資料填入 DynamoDB 表格，才能讓範本與您的自訂資料搭配使用。

使用自動 Chatbot 設計器

> **Note**
>
> 您只能使用英文 (美國) 語言的成績單。

自動 Chatbot 設計器可幫助您根據現有對話記錄設計機器人。它分析成績單，並建議具有意圖和插槽類型的初始設計。您可以對機器人設計進行迭代，添加提示，構建，測試和部署機器人。

使用 Amazon Lex V2 主控台或 API 建立新的機器人或將語言新增至機器人後，您可以上傳雙方之間的對話記錄。自動聊天機器人設計器會分析成績單，並確定機器人的意圖和插槽類型。它還標記了影響特定意圖或插槽類型的創建以供您審核的對話。

您可以使用 Amazon Lex V2 主控台或 API 來分析交談記錄，並建議機器人的意圖和插槽類型。

聊天機器人設計師完成分析後，您可以查看建議的意圖和插槽類型。新增建議的意圖或插槽類型後，您可以使用主控台或 API 修改或將其從機器人設計中刪除。

自動聊天機器人設計工具使用 Amazon Connect 結構描述的隱形眼鏡來支援交談記錄檔案。如果您使用不同的聯絡中心應用程式，則必須將對話記錄轉換為聊天機器人設計人員使用的格式。如需相關資訊，請參閱 輸入成績單格式。

若要使用自動聊天機器人設計工具，您必須允許執行設計人員存取權的 IAM 角色。如需特定 IAM 政策的相關資訊，請參閱允許使用者使用自動 Chatbot 設計工具。若要讓 Amazon Lex V2 使用選用AWS KMS金鑰加密輸出資料，您需要使用中所示的政策更新金鑰允許使用者使用 AWS KMS 金鑰來加密和解密檔案。

> **Note**
>
> 如果您使用KMS key，則無論使用的IAM角色為何，都必須提供KMS key策略。

主題

- 匯入對話文字單
- 建立意圖和插槽類型
- 輸入成績單格式

- 輸出成績單格式

匯入對話文字單

匯入對話記錄是一個三個步驟的程序：

1. 將成績單轉換為正確的格式，以準備要匯入的文字單。如果您使用的是 Amazon Connect 的隱形眼鏡，成績單已經是正確的格式。
2. 將文字稿上傳到 Amazon S3 儲存貯體。如果您使用隱形眼鏡，您的成績單已經在 S3 存儲桶中。
3. 使用 Amazon Lex V2 主控台或 API 操作來分析文字記錄。完成培訓所需的時間取決於成績單的數量和對話的複雜性。一般而言，每分鐘會分析 500 行成績單。

以下各節將說明上述每個步驟。

從 Amazon Connect 的隱形眼鏡導入成績單

Amazon Lex V2 自動聊天機器人設計工具與隱形眼鏡文字記錄檔案相容。若要使用隱形眼鏡記錄檔案，您必須開啟隱形眼鏡並記下其輸出檔案的位置。

從隱形眼鏡匯出成績單

1. 在 Amazon Connect 實例中打開隱形眼鏡。如需指示，請參閱 Amazon Connect 管理員指南中的啟用 Amazon Connect 隱形眼鏡。
2. 請注意 Amazon Connect 用於您的執行個體的 S3 儲存貯體的位置。若要查看位置，請在 Amazon Connect 主控台中開啟資料儲存頁面。如需指示，請參閱 Amazon Connect 管理員指南中的更新執行個體設定。

在您開啟隱形眼鏡並記下成績單檔案的位置之後，請前往以使用 Amazon Lex V2 主控台分析您的成績單取得匯入和分析成績單的說明。

準備成績單

通過創建成績單文件來準備成績單。

- 為每個對話創建一個成績單文件，列出雙方之間的互動。交談中的每個互動可以跨越多行。您可以同時提供編輯版本和未編輯的對話版本。
- 檔案必須是在中指定的 JSON 格式輸入成績單格式。

- 您必須提供至少 1,000 個對話回合。為了改善意圖和插槽類型的發現，您應該提供大約 10,000 個或更多的對話回合。自動聊天機器人設計師將僅處理前 70 萬回合。
- 您可以上傳的成績單文件數量沒有限制，也沒有文件大小限制。

如果您計劃依日期篩選匯入的成績單，檔案必須位於下列目錄結構中：

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

成績單檔案必須在檔案名稱的某處包含格式為 yyyy-mm-dd "" 的日期。

從其他客服中心應用程式匯出成績單

1. 使用聯絡中心應用程式的工具匯出對話。交談必須至少包含在中指定的資訊輸入成績單格式。
2. 將您的聯絡中心應用程式產生的成績單轉換為中所述的格式。輸入成績單格式您負責執行轉換。

我們提供三個腳本來準備成績單。這些類別為：

- 將隱形眼鏡成績單與 Amazon Lex V2 交談日誌相結合的指令碼。隱形眼鏡記錄不包括與 Amazon Lex V2 機器人互動的 Amazon Connect 交談的部分內容。此指令碼需要為 Amazon Lex V2 開啟交談日誌，以及查詢交談 CloudWatch 日誌記錄和隱形眼鏡 S3 儲存貯體的適當權限。
- 將 Amazon Transcribe 呼叫分析轉換為 Amazon Lex V2 輸入格式的指令碼。
- 將 Amazon Connect 聊天記錄轉換為 Amazon Lex V2 輸入格式的指令碼。

您可以從此 GitHub 存儲庫下載腳本：https://github.com/aws-samples/ amazon-lex-bot-recommendation-集成。

將您的成績單上傳到 S3 儲存貯體

如果您使用隱形眼鏡，則成績單文件已包含在 S3 存儲桶中。如需成績單檔案的位置和檔案名稱，請參閱 Amazon Connect 管理員指南中的隱形眼鏡輸出檔案範例。

如果您正在使用其他聯絡中心應用程式，但尚未為成績單檔案設定 S3 儲存貯體，請遵循此程序。否則，如果您有現有的 S3 儲存貯體，請在登入 Amazon S3 主控台後，從步驟 5 開始執行此程序。

將檔案上傳至 S3 儲存貯體

1. 登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/s3/ 的 Amazon S3 主控台。
2. 選擇 Create bucket (建立儲存貯體)。
3. 為值區命名並選擇 [地區]。該區域必須與您用於 Amazon Lex V2 的區域相同。根據您的使用案例的需要設定其他選項。
4. 選擇 建立儲存貯體 。
5. 從值區清單中，選擇現有值區或您剛建立的值區
6. 選擇 Upload (上傳)。
7. 新增您要上傳的成績單檔案。
8. 選擇 Upload (上傳)。

使用 Amazon Lex V2 主控台分析您的成績單

您只能使用空白語言的自動化機器人設計。您可以將新語言新增至現有的機器人，或建立新的機器人。

在新機器人中建立新語言

1. 登錄到AWS Management Console並打開 Amazon Lex 控制台 https://console.aws.amazon.com/lex/.
2. 選擇建立機器人
3. 選擇從自動 Chatbot 設計器開始。填寫資訊以建立新的機器人。
4. 選擇 Next (下一步)
5. 在 [新增語言至機器人] 中，填寫語言的資訊。
6. 在 S3 上的謄本檔案位置區段中，選擇包含成績單檔案的 S3 儲存貯體，以及檔案的本機路徑 (如有必要)。
7. 您可以選擇性地選擇下列項目：
   - 在處理過程中加密成績單數據的密AWS KMS鑰。如果您未選取金鑰，則會使用服務AWS KMS金鑰。
   - 若要篩選特定日期範圍的成績單。如果您選擇篩選成績單，它們必須位於正確的資料夾結構中。如需詳細資訊，請參閱準備成績單。
8. 選擇 Done (完成)。

等待 Amazon Lex V2 處理成績單。分析完成時，您會看到完成訊息。

如何停止分析您的成績單

如果您需要停止對已上傳的成績單的分析，您可以停止執行中的BotRecommendation工作，其BotRecommendationStatus狀態為處理中。您可以在從主控台提交作業之後，或使用 CLI SDK 進行 StopBotRecommendation API，按一下橫幅上的 [停止處理] 按鈕。如需詳細資訊，請參閱 StopBotRecommendation

呼叫之後StopBotRecommendation，內部會設定BotRecommendationStatus為，Stopping且不會向您收費。若要確定工作已停止，您可以呼叫 DescribeBotRecommendation API 並確認BotRecommendationStatus是Stopped. 這通常需要 3-4 分鐘。

呼叫 StopBotRecommendation API 後，系統不會向您收取處理費用。

建立意圖和插槽類型

聊天機器人設計師創建意圖和插槽類型後，您可以選擇要添加到機器人的意圖和插槽類型。您可以檢閱每個意圖和插槽類型的詳細資料，以協助您決定哪些建議與您的使用案例最相關。

您可以按一下建議的意圖名稱，以檢視聊天機器人設計人員建議的範例話語和插槽。如果您選取 [顯示相關聯的成績單]，您也可以捲動您提供的交談。這些成績單會影響聊天機器人設計者對此意圖的建議。如果您按一下範例語音，您可以檢閱主要對話和相關對話方式，這會影響該特定語音。

您可以按一下特定槽類型的名稱，以檢視已建議的槽值。如果您選取 [顯示相關聯的成績單]，您可以檢閱影響此插槽類型的交談，並且會反白顯示插槽類型的代理程式提示。如果您按一下特定的插槽類型值，您可以檢閱主要對話以及影響此值的相關對話方塊。

若要檢閱和新增對應方式和插槽類型

1. 登錄到AWS Management Console並打開 Amazon Lex 控制台 https://console.aws.amazon.com/lex/.
2. 從機器人列表中，選擇要使用的機器人。
3. 選擇 [檢視語言]。
4. 從語言清單中，選擇要使用的語言。
5. 在 [交談結構] 中選擇 [檢閱]。
6. 在意圖和插槽類型列表中，選擇要添加到機器人的類型。您可以選擇意圖或插槽類型，以查看詳細資訊和相關聯的成績單。

意圖是依 Amazon Lex V2 對意圖與已處理的成績單相關聯的信心來排序。

輸入成績單格式

以下是為您的機器人生成意圖和插槽類型的輸入文件格式。輸入檔案必須包含這些欄位。其他欄位會被忽略。

輸入格式與 Amazon Connect 的隱形眼鏡的輸出格式兼容。如果您使用隱形眼鏡，則無需修改成績單文件。如需詳細資訊，請參閱隱形眼鏡輸出檔案範例。如果您正在使用其他聯絡中心應用程式，則必須將成績單檔案轉換為此格式。

```
{
    "Participants": [
        {
            "ParticipantId": "string",
            "ParticipantRole": "AGENT | CUSTOMER"
        }
    ],
    "Version": "1.1.0",
    "ContentMetadata": {
        "RedactionTypes": [
            "PII"
        ],
        "Output": "Raw | Redacted"
    },
    "CustomerMetadata": {
        "ContactId": "string"
    },
    "Transcript": [
        {
            "ParticipantId": "string",
            "Id": "string",
            "Content": "string"
        }
    ]
}
```

輸入檔案中必須存在下列欄位：

- 參與者識別交談中的參與者，以及他們扮演的角色。
- 版本輸入檔案格式的版本。總是「1.1.0」。

- ContentMetadata指出您是否從成績單中移除敏感資訊。如果成績單包含敏感資訊，請將Output欄位設定為「原始」。
- CustomerMetadata交談的唯一識別碼。
- 成績單對話中各方之間的對話文本。交談的每一回合都會以唯一識別碼來識別。

輸出成績單格式

輸出成績單格式與輸入成績單格式幾乎相同。但是，它還包括一些客戶元數據和一個列出了影響意圖和插槽類型建議的細分的字段。您可以從主控台的「檢閱」頁面或使用 Amazon Lex V2 API 下載輸出文字稿。如需詳細資訊，請參閱輸入成績單格式。

```
{
    "Participants": [
        {
            "ParticipantId": "string",
            "ParticipantRole": "AGENT | CUSTOMER"
        }
    ],
    "Version": "1.1.0",
    "ContentMetadata": {

        "RedactionTypes": [
            "PII"
        ],
        "Output": "Raw | Redacted"
    },
    "CustomerMetadata": {
        "ContactId": "string",
        "FileName": "string",
        "InputFormat": "Lex"
    },
    "InfluencingSegments": [
        {
            "Id": "string",
            "StartTurnIndex": number,
            "EndTurnIndex": number,
            "Intents": [
                {
                    "Id": "string",
                    "Name": "string",
                    "SampleUtteranceIndex": [
```

```
                        {
                            "Index": number,
                            "Content": "String"
                        }
                    ]
                }
            ],
            "SlotTypes": [
                {
                    "Id": "string",
                    "Name": "string",
                    "SlotValueIndex": [
                        {
                            "Index": number,
                            "Content": "String"
                        }
                    ]
                }
            ]
        }
    ],
    "Transcript": [
        {

            "ParticipantId": "string",
            "Id": "string",
            "Content": "string"
        }
    ]
}
```

- CustomerMetadata— 該字段中添加了兩個字CustomerMetadata段，包含對話的輸入文件的名稱和輸入格式，始終是「Lex」。
- InfluencingSegments— 識別會影響意圖或插槽類型建議的交談區段。意圖或插槽類型的 ID 可識別受交談影響的特定識別碼。

新增語言

您可以將一個或多個語言和地區設定新增至您的機器人，以使其能夠以他們的語言與使用者通訊。您可以針對每種語言分別定義意圖、插槽和插槽類型，以便語音、提示和插槽值是特定於該語言的值。

您的機器人必須至少包含一種語言。

若要將語言新增至您的機器人

1. 在 [新增語言] 區段中，選擇您要使用的語言。您可以新增說明，以協助識別清單中的語言。
2. 如果您的機器人支援語音互動，請在「語音互動」區段中選擇 Amazon Lex V2 用來與使用者通訊的 Amazon Polly 語音。如果您的機器人不支援語音，請選擇「無」。
3. 對於意圖分類可信度分數閾值，請設定 Amazon Lex V2 用來判斷意圖是否為正確意圖的值。您可以在測試機器人之後調整此值。
4. 選擇 Add (新增)。

新增意圖

意圖是您的使用者想要完成的目標，例如訂購鮮花或預訂酒店。您的機器人必須至少有一個意圖。

根據預設，所有機器人都包含單一內建意圖，即後援意圖。當 Amazon Lex V2 無法辨識任何其他意圖時，就會使用此目的。例如，如果用戶對酒店預訂意圖說「我想訂購鮮花」，則會觸發後備意圖。

若要新增意圖

1. 登入AWS Management Console並打開亞馬遜萊克斯控制台https://console.aws.amazon.com/lex/。
2. 從機器人列表中，選擇要添加意圖的機器人，然後從新增語言選擇檢視語言。
3. 選擇要加入意圖的語言，然後選擇意圖。
4. 選擇加入意圖，為您的意圖命名，然後選擇新增。
5. 在意圖編輯器中，添加意圖的詳細信息。

   - 對話流程— 使用對話流程圖來查看與機器人的對話框的外觀。您可以選擇對話的不同部分，以跳至意圖編輯器的該部分。
   - 意圖細節— 為意圖提供名稱和描述，以協助識別意圖的目的。您也可以查看 Amazon Lex V2 指派給意圖的唯一識別碼。
   - 上下文— 設定意圖的輸入與輸出內容。上下文是與意圖相關聯的狀態變量。當一個意圖被實現的輸出上下文設置。只有在前後關聯處於活動狀態時，才能識別具有輸入上下文的意圖。沒有輸入上下文的意圖始終可以被識別。
   - 語音樣本— 您應該提供 10 個或更多的短語，您希望您的用戶用來發起意圖。Amazon Lex V2 概括了這些詞組，以識別使用者想要啟動意圖。

- 初步回應— 叫用意圖之後傳送給使用者的初始訊息。您可以提供回應、初始化值，並定義 Amazon Lex V2 在意圖開始時回應使用者所採取的下一個步驟。
- 插槽— 定義滿足意圖所需的槽或參數。每個插槽都有一個類型，該類型定義了可以在槽中輸入的值。您可以從自定義插槽類型中進行選擇，也可以選擇內置插槽類型。
- 確認-這些提示和響應用於確認或拒絕實現意圖。確認提示會要求使用者檢閱槽值。例如，「我已經預訂了星期五的酒店房間。這是正確的嗎？」 當使用者拒絕確認時，會將偏角回應傳送給使用者。您可以提供回應、設定值，並定義 Amazon Lex V2 對應使用者的確認或偏移回應所採取的下一個步驟。
- 履行— 在履行過程中發送給用戶的響應。您可以在出貨開始時設定出貨進度更新，並在出貨進行期間定期設定。例如，「我正在更改您的密碼，這可能需要幾分鐘的時間」和「我仍在處理您的請求」。出貨作業更新僅用於串流交談。您也可以設定履行後成功訊息、失敗訊息和逾時訊息。您可以傳送串流和定期對話的履行後訊息。例如，如果出貨成功，您可以傳送「我已變更密碼」。如果訂單出貨未成功，您可以傳送包含更多資訊的回應，例如：「我無法變更密碼，請聯絡服務台尋求協助」。如果出貨所花費的時間超過設定的逾時時間，您可以傳送訊息通知使用者，例如「我們的伺服器目前非常忙碌。請稍後再試一次。」 您可以提供回應、設定值，並定義 Amazon Lex V2 回應使用者所採取的下一個步驟。
- 關閉回應— 完成意圖並播放所有其他訊息後，傳送給使用者的回應。例如，感謝您預訂酒店房間。或者它可以提示用戶啟動不同的意圖，例如：「感謝您預訂房間，您想預訂租車嗎？」 在完成意圖並以結束回應回應後，您可以提供回應並設定後續後續動作。
- 程式碼掛鉤— 指示您是否正在使用AWS Lambda函數初始化意圖和驗證用戶輸入。您可以在用來執行機器人的別名中指定 Lambda 函數。

6. 選擇儲存意圖保存意圖。

> **Note**
>
> 2022 年 8 月 17 日，Amazon Lex V2 發佈了與使用者對話管理方式的變更。此變更可讓您更好地控制使用者通過交談的路徑。如需詳細資訊，請參閱了解交談流程管理。在 2022 年 8 月 17 日之前建立的機器人不支援對話方塊程式碼掛接訊息、設定值、設定後續步驟和新增條件。

以特定順序配置提示

您可以勾選下列方塊，將機器人設定為以預先定義的順序播放訊息依序播放訊息。否則，機器人會以隨機順序播放消息和變化。

已排序的提示允許訊息和訊息群組的變體在重試期間按順序播放。當使用者提供無效的提示回應或用於意圖確認時，您可以使用訊息的替代改寫方式。每個插槽最多可以設置兩個原始消息的變體。您可以選擇是按順序還是隨機播放消息。

有序提示支援所有四種類型的訊息：文字、自訂承載回應、SSML 和卡片群組。回應會在相同的訊息群組中排序。不同的消息組是獨立的。

主題

- 語音樣本
- 意圖結構
- 建立交談路徑
- 使用視覺對話生成器
- 內建槽

語音樣本

您可以建立範例語音，這些詞組是您希望使用者用來啟動意圖的詞組變體。例如，針對**BookFlight**意圖，您可能會包含下列語音：

1. 我想預訂航班
2. 幫我搭飛機
3. 請給我飛機票
4. 從 {*DepartureCity*} 飛往 {} 的航班 *DestinationCity*

您應該提供 10 個或更多樣本語音。提供代表用戶可能會說出的各種句子結構和單詞的樣本。也考慮不完整的句子，例如上面的例子 3 和 4。您也可以在範例語音中使用您為意圖定義的插槽，方法是將大括號包覆在插槽名稱周圍，如範例 4 中的 {*DepartureCity*} 所示。如果您在範例語音中包含插槽名稱，Amazon Lex V2 會以使用者在話語中提供的值填入意圖的插槽。

各種範例語彙可協助 Amazon Lex V2 一般化，以有效辨識使用者想要啟動意圖。

您可以在意圖編輯器、視覺化交談產生器或使用CreateIntent或 UpdateIntentAPI 作業中新增範例語音。您也可以利用 Amazon 基岩的生成人工智慧功能，自動產生範例語彙。如需詳細資訊，請參閱話語生成。

使用意圖編輯器或視覺對話生成器

1. 在「意圖」編輯器中，導覽至「範例語音」區段。在視覺化對話產生器中，找到 [開始] 區塊中的 [範例語音] 區段。
2. 在包含透明文字的方塊中**I want to book a flight**，輸入範例語調。選取「新增語音」以加入語音。
3. 檢視您在「預覽」或「純文字」模式中新增的範例語彙。在純文本中，每一行都是一個單獨的話語。在「預覽」模式中，將游標暫留在語音上以顯示下列選項：
   - 選取文字方塊以編輯語音。
   - 選取文字方塊右側的 x 按鈕以刪除語音。
   - 拖曳文字方塊左側的按鈕可變更範例語音的順序。
4. 使用頂端的搜尋列來搜尋範例語彙，並使用旁邊的下拉式選單來依您新增話語的順序或按字母順序排序。

使用 API 作業

1. 使用作業建立新意圖，或使用CreateIntent作業更新現有意圖。UpdateIntent
2. API 請求包括一個sampleUtterances字段，該字段映射到SampleUtterance對象數組。
3. 針對您要新增的每個範例語調，將SampleUtterance物件附加至陣列。加入範例語調作為欄位的utterance值。
4. 若要編輯和刪除範例語音，請傳送請求。UpdateIntent您在欄位中提供的語音清單會取代sampleUtterances現有的話語。

> **Important**
>
> 您在UpdateIntent要求中保留空白的任何欄位都會導致意圖中的現有組態遭到刪除。使用此DescribeIntent作業可傳回機器人組態，並將您不想刪除的任何組態複製到要UpdateIntent求中。

意圖結構

下列主題說明機器人在實現意圖時所採取的不同步驟，以及如何設定每個步驟：

主題

- 初步回應
- 槽
- 確認
- 實現
- 閉幕回應

初步回應

在 Amazon Lex V2 確定意圖之後，並在開始引出插槽值之前，系統會將初始回應傳送給使用者。您可以使用此回應來通知使用者已辨識的意圖，並為您收集的資訊做好準備，以達成意圖。

例如，如果目的是為汽車安排維修預約，則最初的回應可能是：

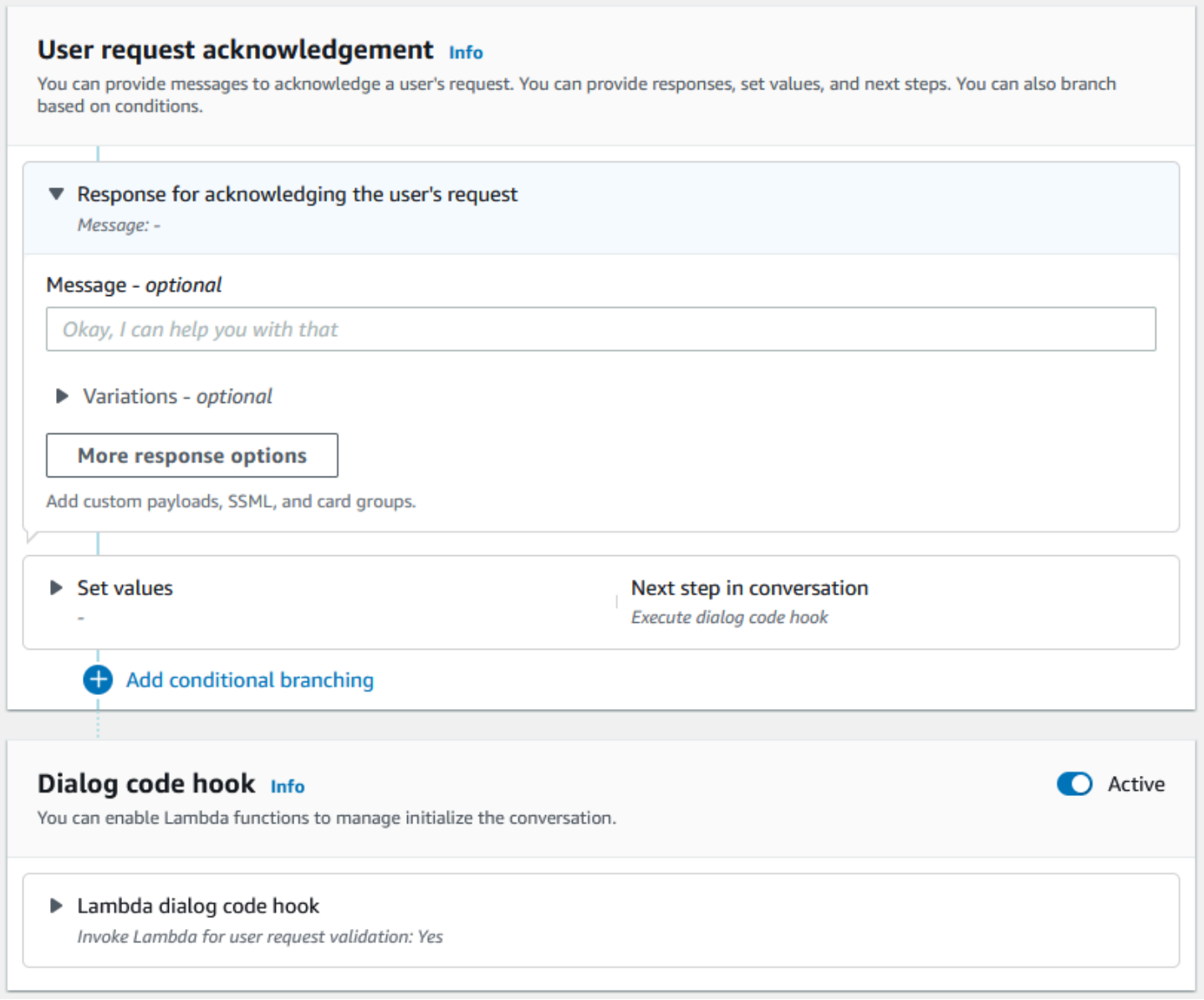
```
我可以幫你安排約會 您需要提供汽車的品牌，型號和年份。
```

不需要初始回應訊息。如果您沒有提供，Amazon Lex V2 會繼續遵循初始回應的下一個步驟。

您可以在初始回應中設定下列選項：

- 設定下一步— 您可以提供交談中的下一個步驟，例如跳至特定的對話方塊動作、引出特定位置，或跳至不同的意圖。如需詳細資訊，請參閱設定交談中的後續步驟。
- 設定值— 您可以設定插槽和工作階段屬性的值。如需詳細資訊，請參閱 在交談期間設定值
- 新增條件式分支-您可以在播放初始響應後應用條件。當條件評估為 true 時，會採取您定義的動作。如需詳細資訊，請參閱向分支對話添加條件。
- 執行對話框代碼掛鉤— 您可以定義 Lambda 程式碼掛接，以初始化資料並執行商業邏輯。如需詳細資訊，請參閱調用對話框代碼掛鉤。如果針對意圖啟用了執行 Lambda 函數的選項，則依預設會執行對話方塊程式碼掛接。您可以通過切換來禁用對話框代碼掛鉤作用中按鈕。

如果沒有條件或明確的下一個步驟，Amazon Lex V2 會依優先順序移至下一個插槽。



> **Note**
>
> 2022 年 8 月 17 日，Amazon Lex V2 發佈了與使用者對話管理方式的變更。此變更可讓您更好地控制使用者通過交談的路徑。如需詳細資訊，請參閱了解交談流程管理。在 2022 年 8 月 17 日之前建立的機器人不支援對話方塊程式碼掛接訊息、設定值、設定後續步驟和新增條件。

槽

槽是由用戶提供的值來實現意圖。有兩種類型的插槽：

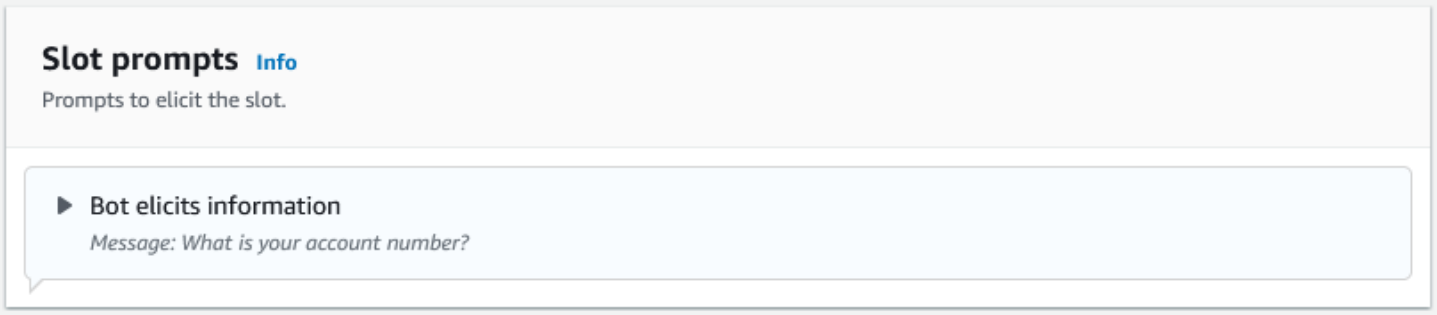
- 內置插槽類型— 您可以使用內置插槽類型捕獲標準值，例如數字，名稱和城市。如需支援的內建插槽類型清單，請參閱內建插槽類型。
- 自訂插槽類型— 您可以使用自訂槽類型來擷取特定於意圖的自訂值。例如，您可以使用自定義插槽類型將帳戶類型捕獲為「支票」或「儲蓄」。如需詳細資訊，請參閱自訂插槽類型。

若要定義意圖中的槽，您必須配置下列項目：

- 插槽資訊— 此欄位包含插槽的名稱和選擇性描述。例如，您可以將插槽名稱提供為」AccountNumber」以擷取帳號。如果需要插槽作為實現意圖的交談流程的一部分，則必須將其標記為必要。
- 插槽類型— 槽類型定義槽可接受的值清單。您可以建立自訂槽類型或使用預先定義的槽類型。
- 插槽提示— 插槽提示是向用戶提出收集信息的問題。您可以設定用來收集資訊的重試次數，以及用於每次重試的提示變化。您也可以在每次重試後啟用 Lambda 函數叫用，以處理擷取的輸入並嘗試解析為有效輸入。
- 等待並繼續（可選）— 通過啟用此行為，用戶可以說出諸如「保持一秒鐘」之類的短語，以使機器人等待他們找到信息並提供信息。此功能僅適用於串流交談。如需詳細資訊，請參閱讓機器人等待使用者提供更多資訊。
- 插槽捕獲響應— 您可以根據從使用者輸入擷取插槽值的結果，設定成功回應和失敗回應。
- 條件式分支-您可以在播放初始響應後應用條件。當條件評估為 true 時，會採取您定義的動作。如需詳細資訊，請參閱向分支對話添加條件。
- 對話框代碼掛鉤— 您也可以使用 Lambda 程式碼掛接來驗證插槽值並執行商業邏輯。如需詳細資訊，請參閱調用對話框代碼掛鉤。
- 使用者輸入類型— 您可以配置輸入類型，以便機器人可以接受特定的模式。默認情況下，接受音頻和 DTMF 模式。您可以選擇性地將其設定為「僅音訊」或「僅 DTMF」。
- 音訊輸入逾時和長度— 您可以配置音頻超時，包括語音超時和靜音超時。另外，您可以設置最大音頻長度。
- DTMF 輸入逾時、字元和長度— 您可以設定 DTMF 逾時以及刪除字元和結束字元。此外，您可以設定最大 DTMF 長度。
- 文字長度— 您可以設置文本模式的最大長度。

播放插槽提示後，使用者會提供槽值作為輸入。如果 Amazon Lex V2 不瞭解使用者提供的插槽值，它會重試引出插槽，直到它瞭解值或超過您為該插槽設定的最大重試次數為止。使用進階重試設定，您可以設定逾時、限制輸入類型，以及啟用或停用初始提示和重試的岔斷。每次嘗試擷取輸入後，Amazon Lex V2 都可以呼叫為機器人設定的 Lambda 函數，並提供用於重試的叫用標籤。例如，您可以使用

Lambda 函數來套用商務邏輯，嘗試將其解析為有效值。此 Lambda 函數可在以下位置啟用進階選項用於插槽提示。



您可以定義一旦輸入位置值或超過重試次數上限，機器人應該傳送給使用者的回應。例如，對於用於為汽車安排維修的機器人，您可以在輸入車輛識別號碼（VIN）時向用戶發送消息：

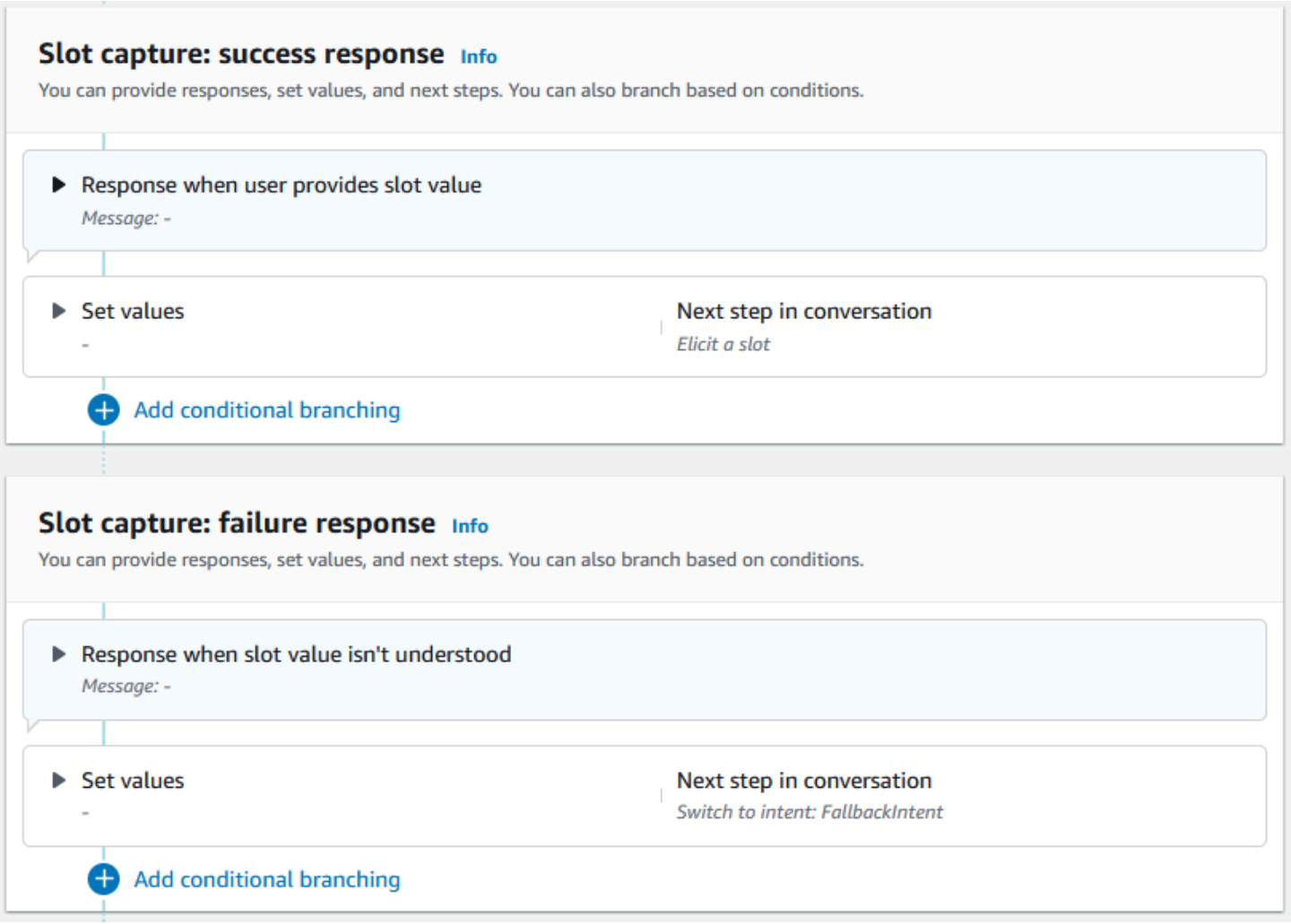
```
感謝您提供汽車的 VIN 號碼。我現在將繼續安排預約。
```

您可以建立兩個回應：

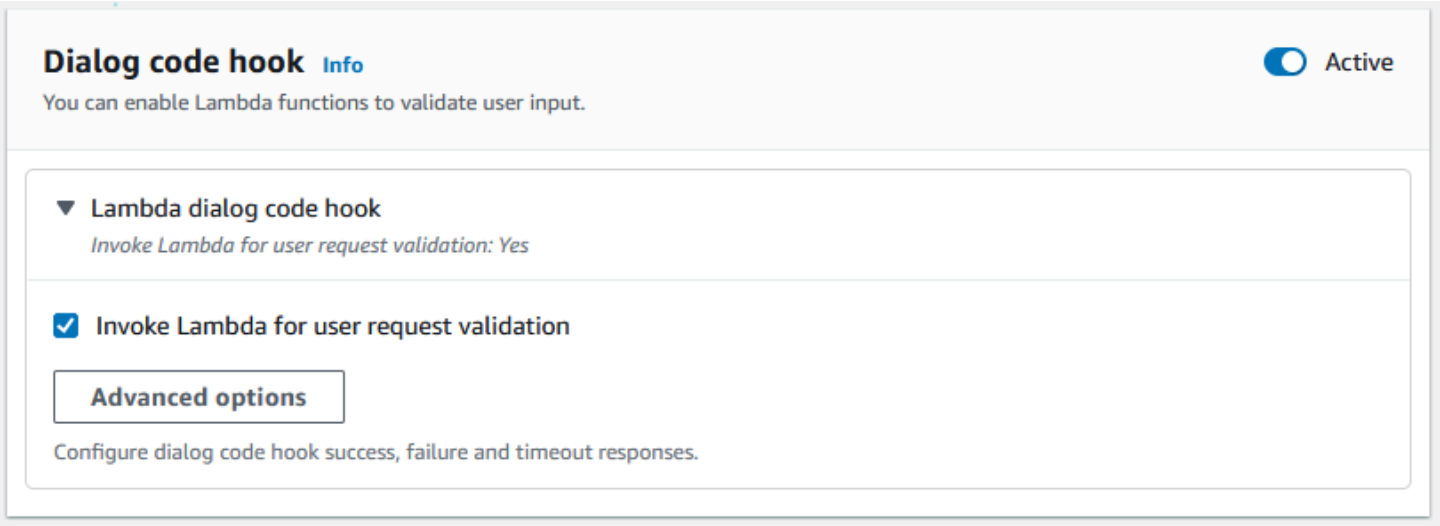
- 成功回應— 當亞馬遜 Lex V2 了解插槽值時發送。
- 故障回應— 當 Amazon Lex V2 在重試次數上限後無法瞭解使用者的插槽值時傳送。

您可以設定值、設定後續步驟，以及套用對應於每個回應的條件，以設計交談流程。

如果沒有條件或明確的下一個步驟，Amazon Lex V2 會依優先順序移至下一個插槽。



您可以使用 Lambda 函數來驗證使用者輸入的插槽值，並決定下一個動作應該是什麼。例如，您可以使用驗證函數來確保輸入的值落在正確的範圍內，或格式正確。若要啟用 Lambda 函數，請選擇調用拉姆達函數核取方塊和作用中「」中的按鈕對話框代碼掛鉤部分。您可以指定對話方塊程式碼掛接的叫用標籤。此叫用標籤可在 Lambda 函數中使用，以撰寫與插槽引出對應的商務邏輯。



意圖不需要的插槽不是主要交談流程的一部分。但是，如果用戶話語包含一個您的機器人識別為與可選插槽相對應的值，則它可以使用該值插入插槽。例如，如果您將商業智慧機器人設定為可選`City`插槽和用戶的話語**`What is the sales for April in San Diego?`**，機器人填充可選插槽**`San Diego`**。您可以將商務邏輯設定為使用選用的插槽值 (如果有的話)。

不能使用後續步驟引發意圖不需要的插槽。這些步驟只能在意圖引出期間填入 (如前面的範例所示)，也可以透過在 Lambda 函數中設定對話方塊狀態來引導這些步驟。如果使用 Lambda 函數引出插槽，您必須使用 Lambda 函數在插槽引出完成後決定交談中的下一個步驟。要在構建機器人時啟用下一步的支持，您必須將插槽標記為意圖所需的位置。

> **Note**
>
> 2022 年 8 月 17 日，Amazon Lex V2 發佈了與使用者對話管理方式的變更。此變更可讓您更好地控制使用者通過交談的路徑。如需詳細資訊，請參閱了解交談流程管理。在 2022 年 8 月 17 日之前建立的機器人不支援對話方塊程式碼掛接訊息、設定值、設定後續步驟和新增條件。

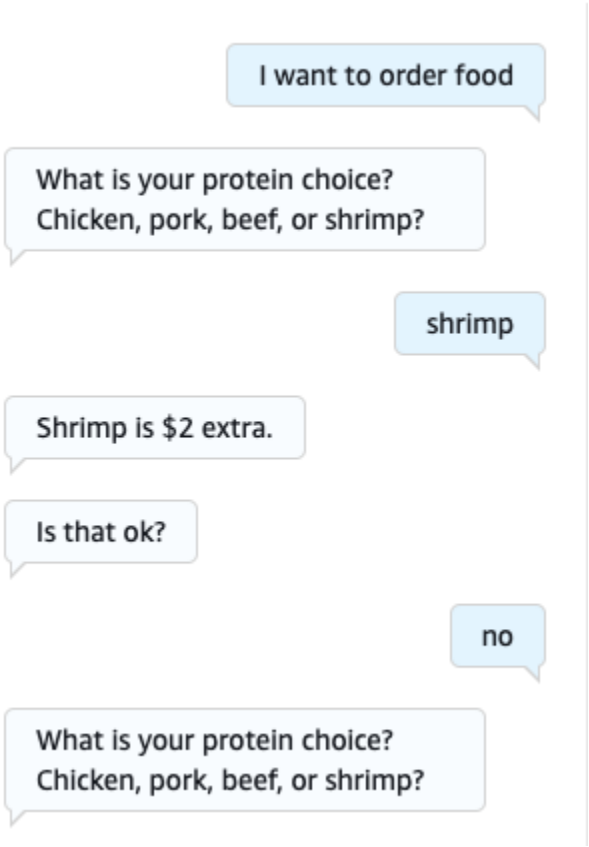
下列主題說明如何設定機器人以重新產生已填滿的插槽值，以及如何建立由多個值組成的位置：

主題

- 重新引出插槽
- 在插槽中使用多個值

重新引出插槽

您可以將機器人配置為重新引出已填充的插槽，方法是將該插槽值設置為，**null**並將對話中的下一步設置為循環以引出該插槽。例如，在客戶根據額外資訊拒絕確認插槽引出之後，您可能會想要重新引出插槽，如下列對話所示：



您可以從確認回應配置迴圈，以使用意圖編輯器或意圖編輯器重新引出插槽。使用視覺對話生成器

> **Note**
>
> 如果您事先將該插槽值設置為，則可以循環返回以在對話中的任何時間點重新引出插槽。**null**

用意圖編輯器複製上面的例子

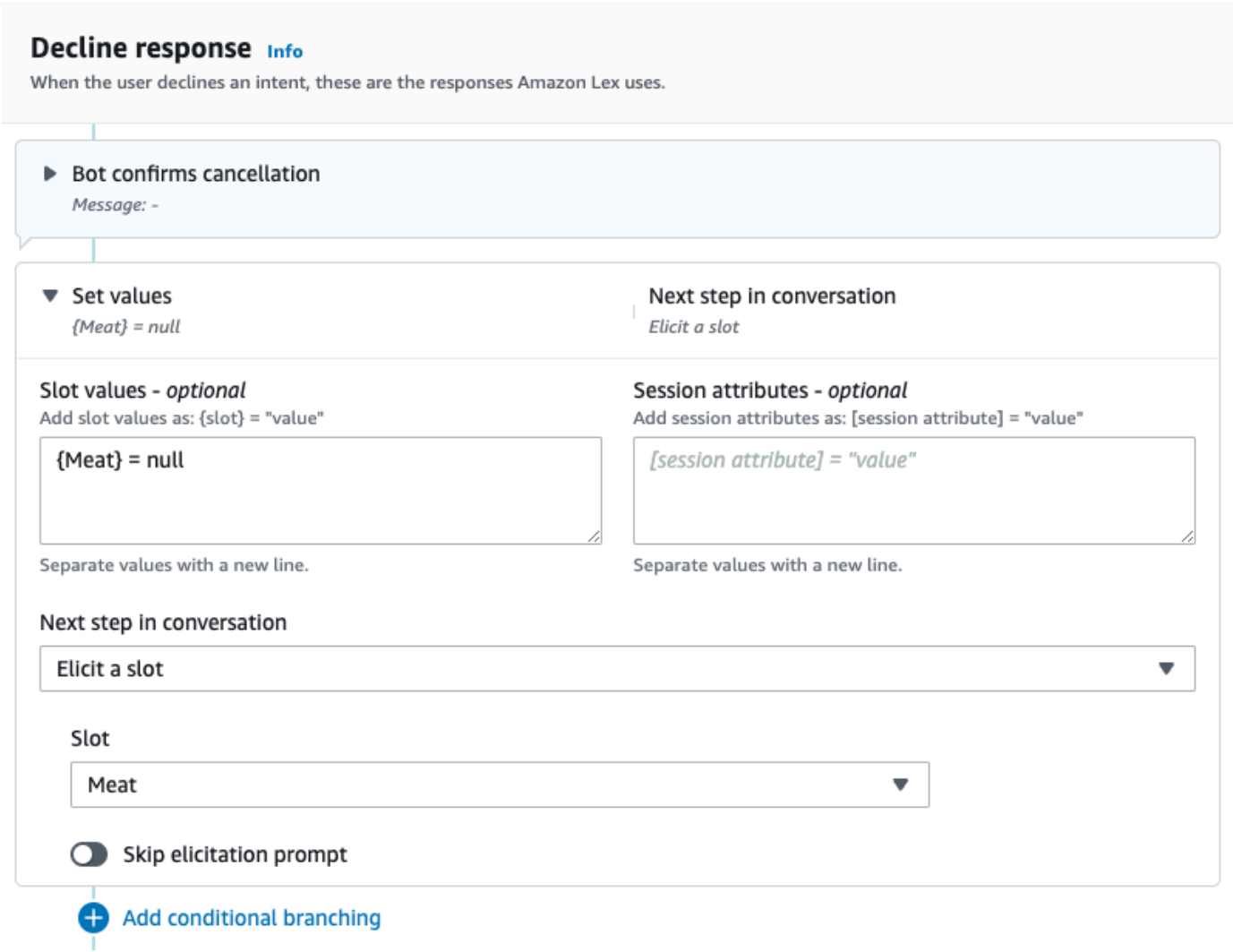
1. 在意圖編輯器的「確認」區段中，選取「提示」旁的向右箭頭，以確認展開區段的意圖。
2. 選取底部的「進階」選項。
3. 在 [拒絕回應] 區段中，選取 [設定值] 旁的向右箭頭以展開區段。按照以下步驟填寫此部分，如下圖所示：

a. 設置要重新引出的插槽值。**null**在此範例中，我們想要重新引出Meat插槽，因此我們在 Slot 值區段**{Meat} = null**中輸入。

b. 在下面的下一個對話步驟下的下拉菜單中，選擇引出一個插槽。

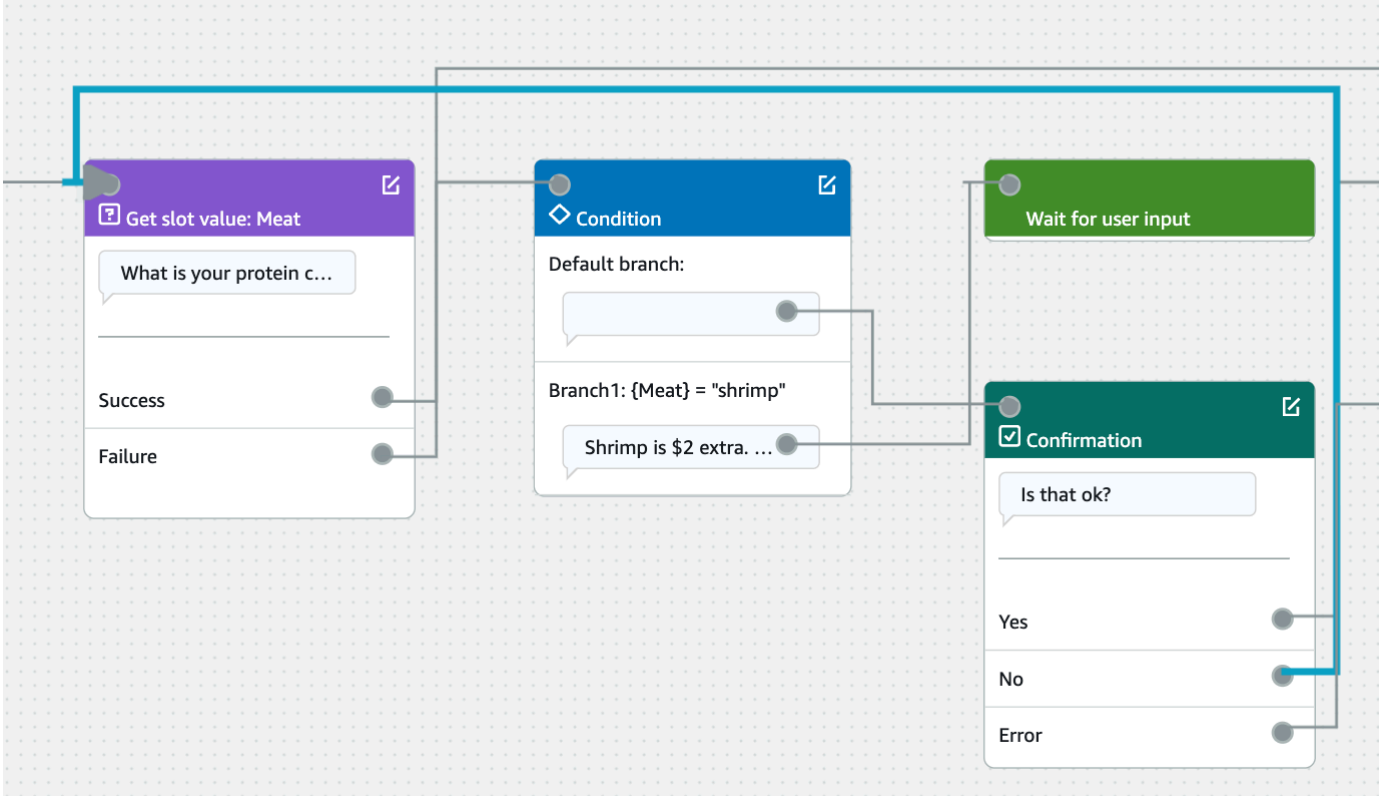
c. 將出現一個插槽部分。在它下面的下拉菜單中，選擇要重新引出的插槽。

d. 選取 [更新選項] 以確認您的變更。

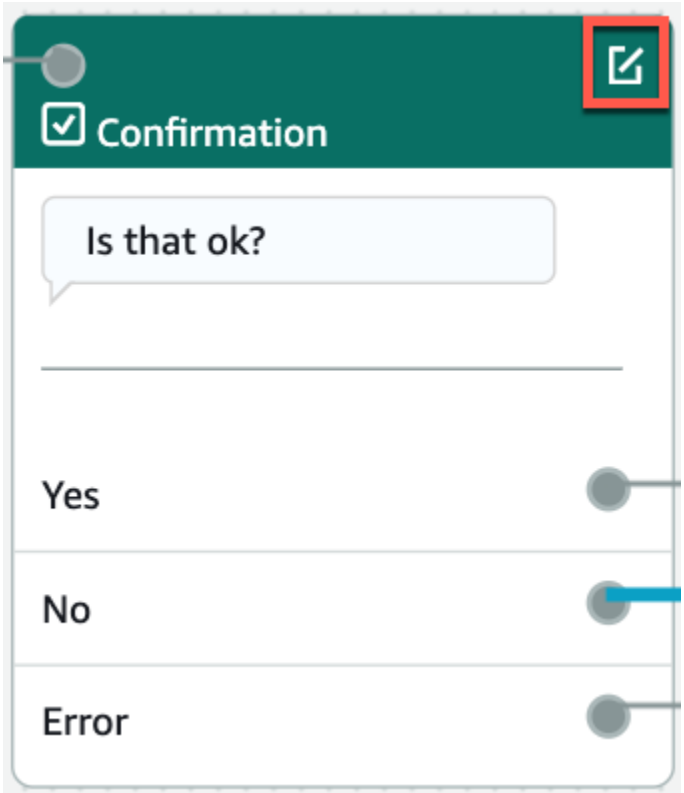


使用視覺對話構建器重現上述示例

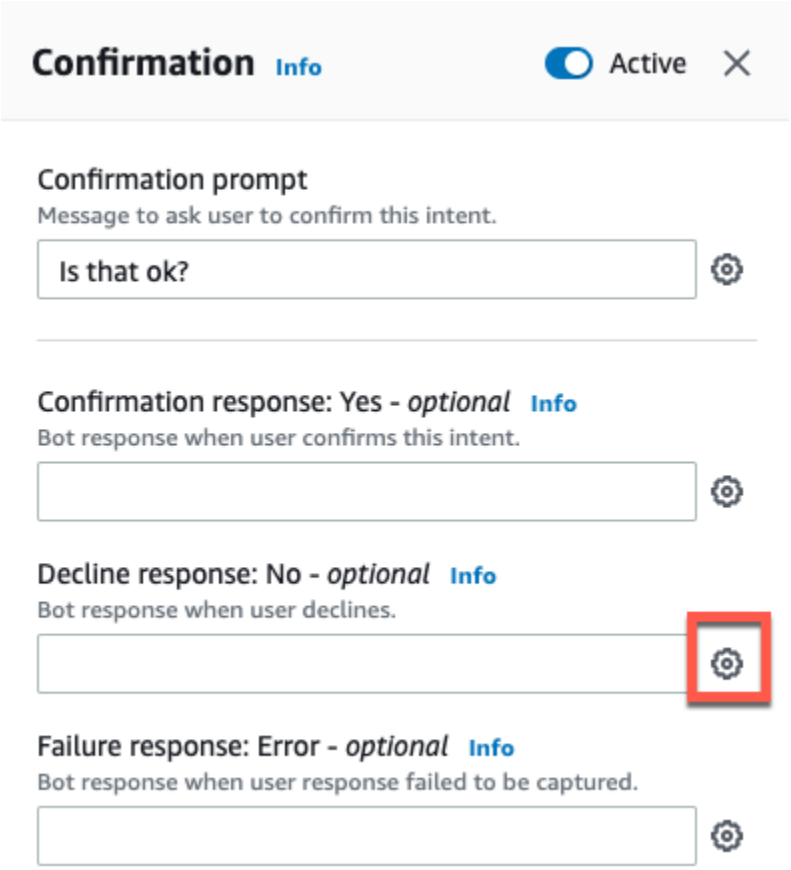
1. 建立從確認區塊的 No 埠連接至 G et 槽值的入埠：肉塊。



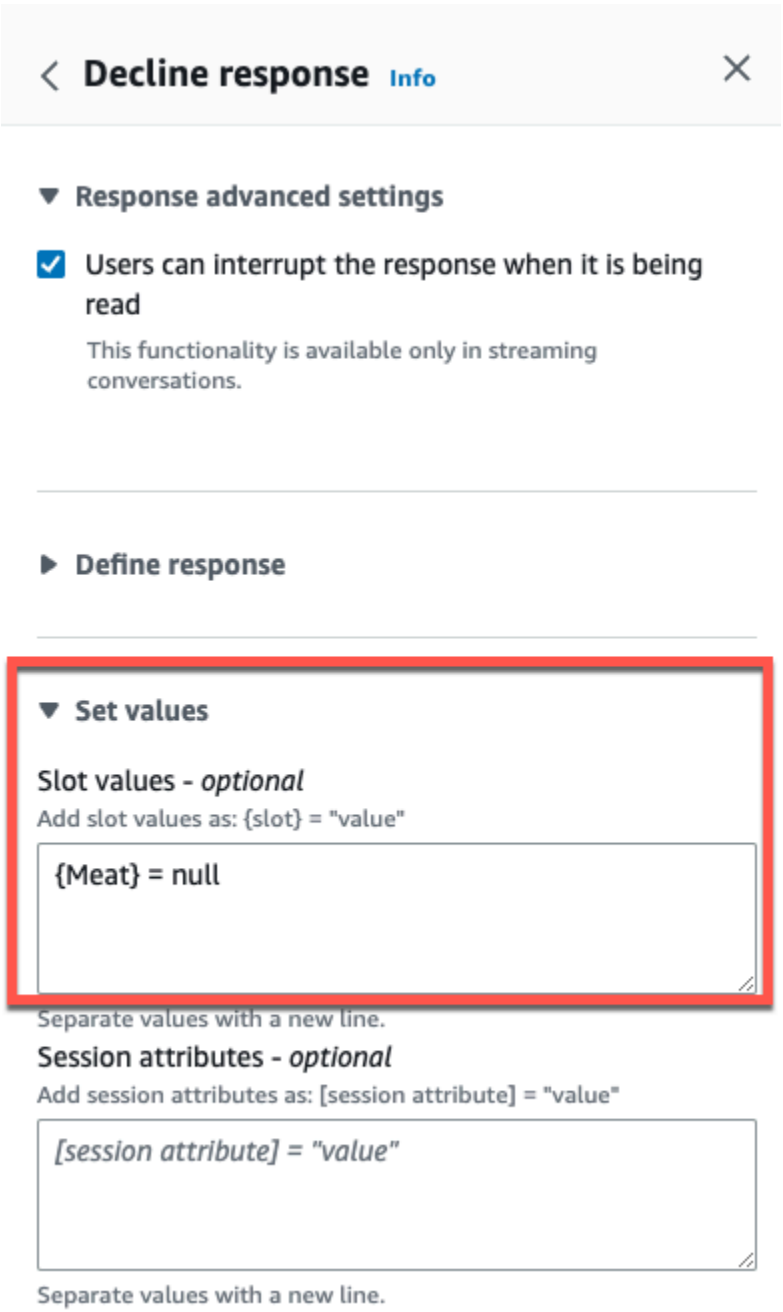
2. 點選「確認」區塊右上角的「編輯」圖示。



3. 在 Decilne 回應區段中，選取機器人回應旁邊的齒輪圖示。



4. 在「設定值」區段中，在「槽值」方塊中加入「{肉} = 空值」。



5. 選取「儲存意圖」。

在插槽中使用多個值

> **Note**
>
> 多個值插槽僅支援英文 (美國) 語言。

對於某些意圖，您可能希望捕獲單個插槽的多個值。例如，披薩訂購機器人可能有以下語調的意圖：

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

意圖預計{toppings}插槽包含客戶想要在比薩餅上的澆頭列表，例如「意大利辣香腸和菠蘿」。

要配置插槽以捕獲多個值，請將插槽上的allowMultipleValues字段設置為 true。您可以使用控制台或使用CreateSlot或UpdateSlot操作來設定欄位。

您只能將具有自訂槽類型的狹槽標記為多值槽。

對於多值插槽，Amazon Lex V2 會在回應RecognizeText或RecognizeUtterance作業時傳回一份插槽值清單。以下是機器人為「我想要一份意大利辣香腸和菠蘿比薩餅」而返回的插槽信息。OrderPizza

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

```
                    }
                }
            ]
        }
    }
```

多值插槽一律會傳回值清單。當語音只包含一個值時，返回的值列表僅包含一個響應。

Amazon Lex V2 可辨識以空格、逗號 (,) 和結合「和」分隔的多個值。多值插槽適用於文本和語音輸入。

您可以在提示中使用多值插槽。例如，您可以將意圖的確認提示設定為

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

當 Amazon Lex V2 向用戶發送提示時，它會發送「你想我訂購你的意大利辣香腸和菠蘿比薩餅嗎？」

多值插槽支援單一預設值。如果提供多個預設值，Amazon Lex V2 只會在插槽中填入第一個可用值。如需詳細資訊，請參閱使用預設插槽值。

您可以使用插槽模糊化來遮罩交談記錄中多值插槽的值。當您混淆槽值時，每個槽值的值都會被槽的名稱取代。如需詳細資訊，請參閱 隱藏交談記錄中的插槽值。

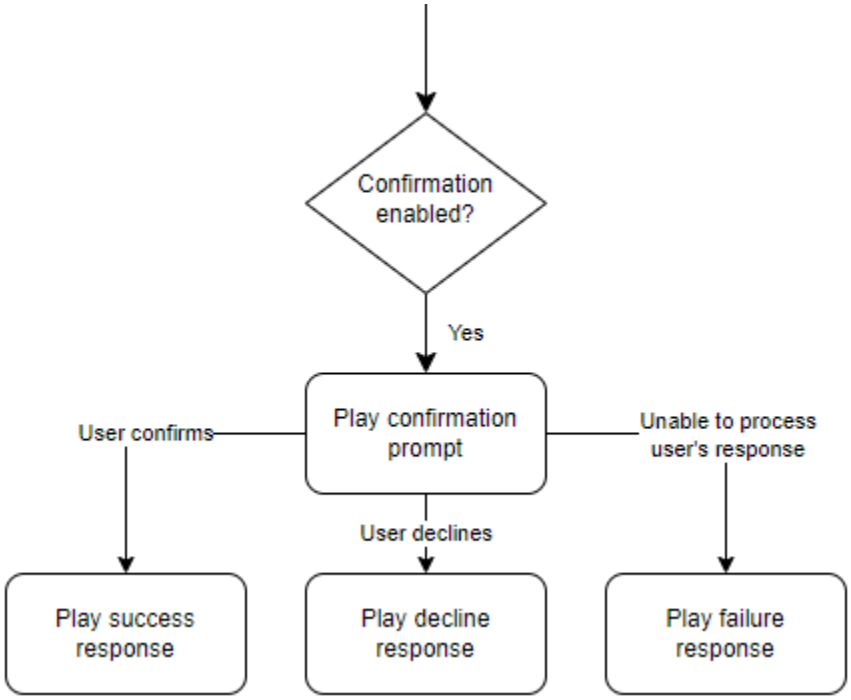
確認

與使用者的交談完成並填滿意圖的槽值之後，您可以設定確認提示，詢問使用者槽值是否正確。例如，為汽車安排維修約會的機器人可能會提示用戶以下內容：

```
我已經為你的服務 2017 本田思域定於 3 月 25 日下午 3:00. 這是所有的權利嗎？
```

您可以在確認提示中定義 3 種類型的回應：

- 確認回應— 當使用者確認意圖時，系統會將此回應傳送給使用者。例如，當使用者回覆「是」提示「您要下訂單嗎？」
- 拒絕回應— 當使用者拒絕意圖時，系統會將此回應傳送給使用者。例如，當使用者回覆「否」提示「您要下訂單嗎？」
- 故障回應— 無法處理確認提示時，系統會將此回應傳送給使用者。例如，如果用戶的響應無法理解或無法解析為是或否。

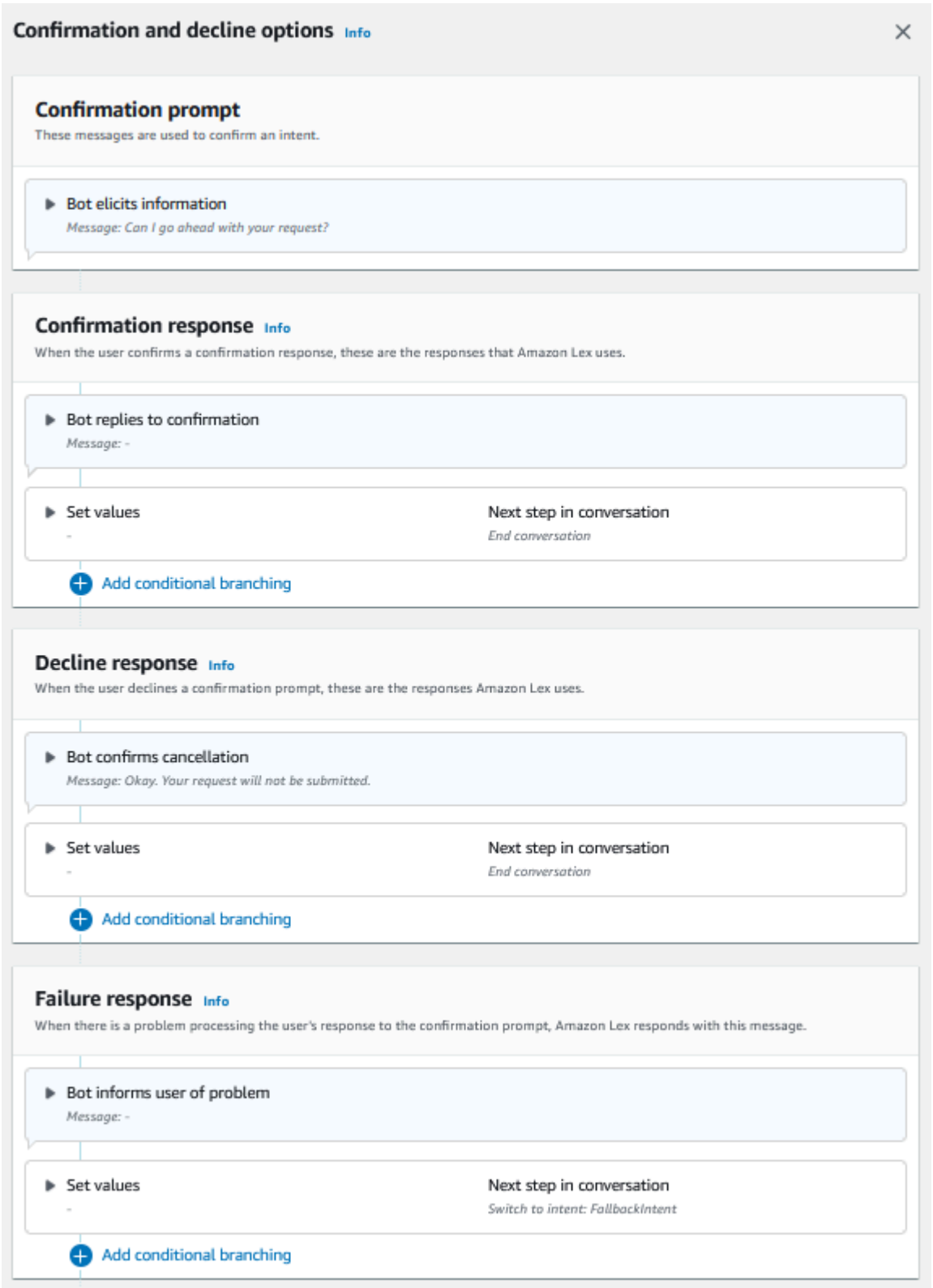


如果您未指定確認提示，Amazon Lex V2 會移至履行步驟或結束回應。

您可以設定值、設定後續步驟，以及套用與每個回應相對應的條件，以設計交談流程。在沒有條件或明確的下一個步驟的情況下，Amazon Lex V2 會移至履行步驟。

您也可以啟用對話方塊程式碼掛接，以驗證意圖中擷取的資訊，然後再傳送以進行履行。若要使用程式碼掛接，請在確認提示進階選項中啟用對話方塊程式碼掛接。此外，設定上一個狀態的下一個步驟，以執行對話方塊程式碼掛接。如需詳細資訊，請參閱調用對話框代碼掛鉤。

> **Note**
>
> 如果您使用程式碼掛接在執行階段觸發確認步驟，您必須將確認步驟標示為作用中在構建時。


Confirmation and decline options Info
Confirmation prompt
These messages are used to confirm an intent.
Bot elicits information
Message: Can I go ahead with your request?
Confirmation response Info
When the user confirms a confirmation response, these are the responses that Amazon Lex uses.
Bot replies to confirmation
Message: -
Set values
-
Next step in conversation
End conversation
Add conditional branching
Decline response Info
When the user declines a confirmation prompt, these are the responses Amazon Lex uses.
Bot confirms cancellation
Message: Okay. Your request will not be submitted.
Set values
-
Next step in conversation
End conversation
Add conditional branching
Failure response Info
When there is a problem processing the user's response to the confirmation prompt, Amazon Lex responds with this message.
Bot informs user of problem
Message: -
Set values
-
Next step in conversation
Switch to intent: FallbackIntent
Add conditional branching

> **Note**
>
> 2022 年 8 月 17 日，Amazon Lex V2 發佈了與使用者對話管理方式的變更。此變更可讓您更好地控制使用者通過交談的路徑。如需詳細資訊，請參閱了解交談流程管理。在 2022 年 8 月 17 日之前建立的機器人不支援對話方塊程式碼掛接訊息、設定值、設定後續步驟和新增條件。

使用 Lambda 函數來驗證意圖。

您可以定義 Lambda 程式碼掛接，以便在傳送意圖進行履行之前驗證意圖。若要使用程式碼掛接，請在確認提示進階選項中啟用對話方塊程式碼掛接。

當您使用程式碼掛接時，您可以定義 Amazon Lex V2 在程式碼掛接執行後採取的動作。您可以建立三種類型的回應：

- 成功回應— 當代碼掛鉤成功完成時發送給用戶。
- 故障回應— 當代碼掛鉤未成功運行或代碼掛鉤返回時發送給用戶`Failure`在響應中。
- 逾時回應— 當程式碼掛接未在其設定的逾時期間內完成時，傳送給使用者。

實現

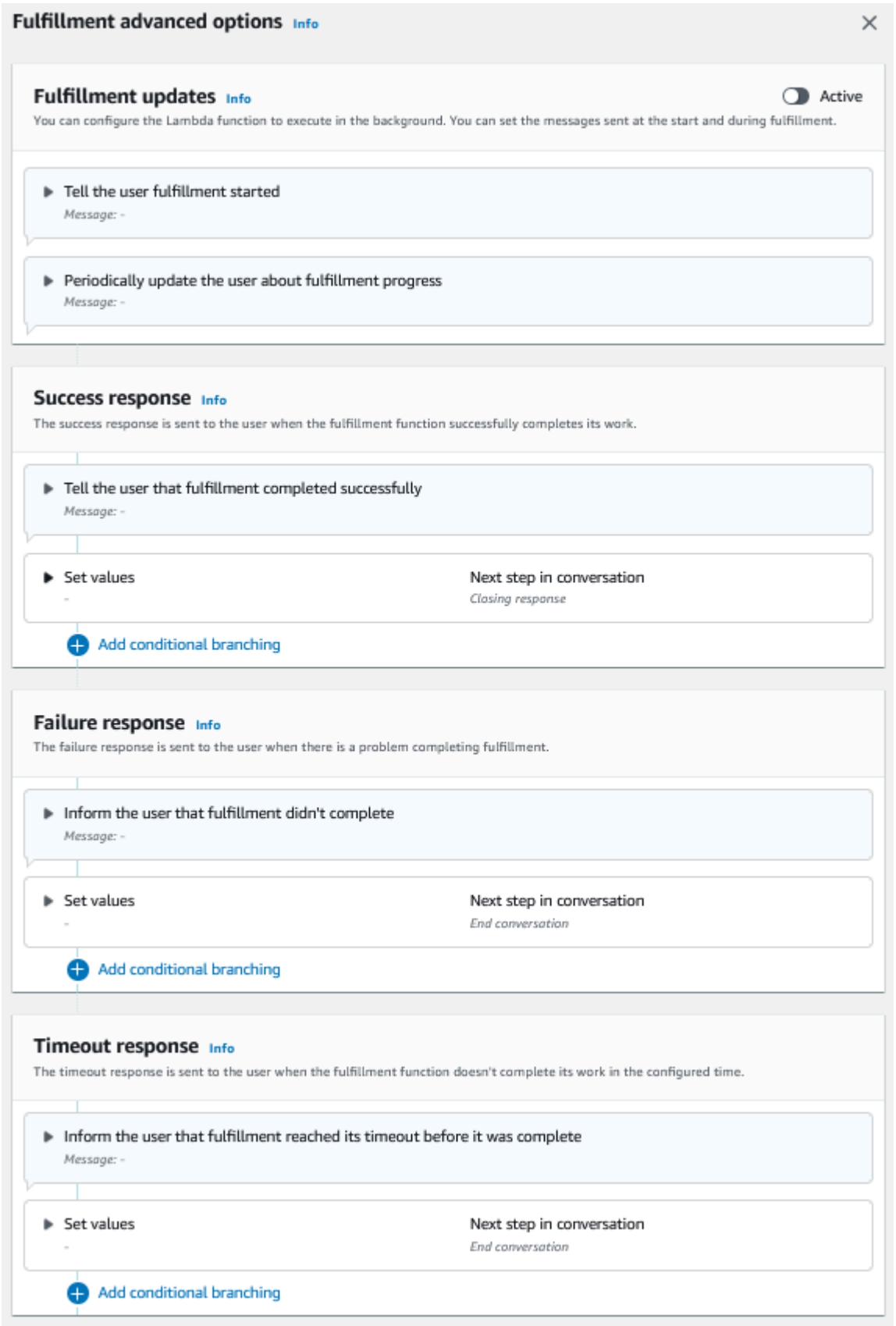
使用者針對意圖提供所有插槽值之後，Amazon Lex V2 就會滿足使用者的要求。您可以設定下列出貨選項。

- 履行代碼掛鉤— 您可以使用此選項來控制履行 Lambda 叫用。如果禁用該選項，則履行成功而不調用 Lambda 函數。
- 履行更新— 您可以啟用 Lambda 函數的履行更新，這些函數需要幾秒鐘以上的時間才能完成，以便使用者知道程序正在進行中。如需詳細資訊，請參閱設定出貨進度更新。此功能僅適用於串流交談。
- 履行回應— 您可以配置成功響應、失敗響應和超時響應。系統會根據履行 Lambda 叫用的狀態，將適當的回應傳回給使用者。

有三種可能的履行回應：

- 成功回應— Lambda 履行成功完成時傳送的訊息。
- 故障回應— 如果出貨失敗或 Lambda 因某些原因無法完成，則會傳送訊息。
- 逾時回應— 如果履行 Lambda 函數未在設定的逾時內完成，則傳送訊息。

您可以設定值、設定後續步驟，以及套用與每個回應相對應的條件，以設計交談流程。在沒有條件或明確的下一個步驟的情況下，Amazon Lex V2 會移至結束回應。


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

> **Note**
>
> 2022 年 8 月 17 日，Amazon Lex V2 發佈了與使用者對話管理方式的變更。此變更可讓您更好地控制使用者通過交談的路徑。如需詳細資訊，請參閱了解交談流程管理。在 2022 年 8 月 17 日之前建立的機器人不支援對話方塊程式碼掛接訊息、設定值、設定後續步驟和新增條件。

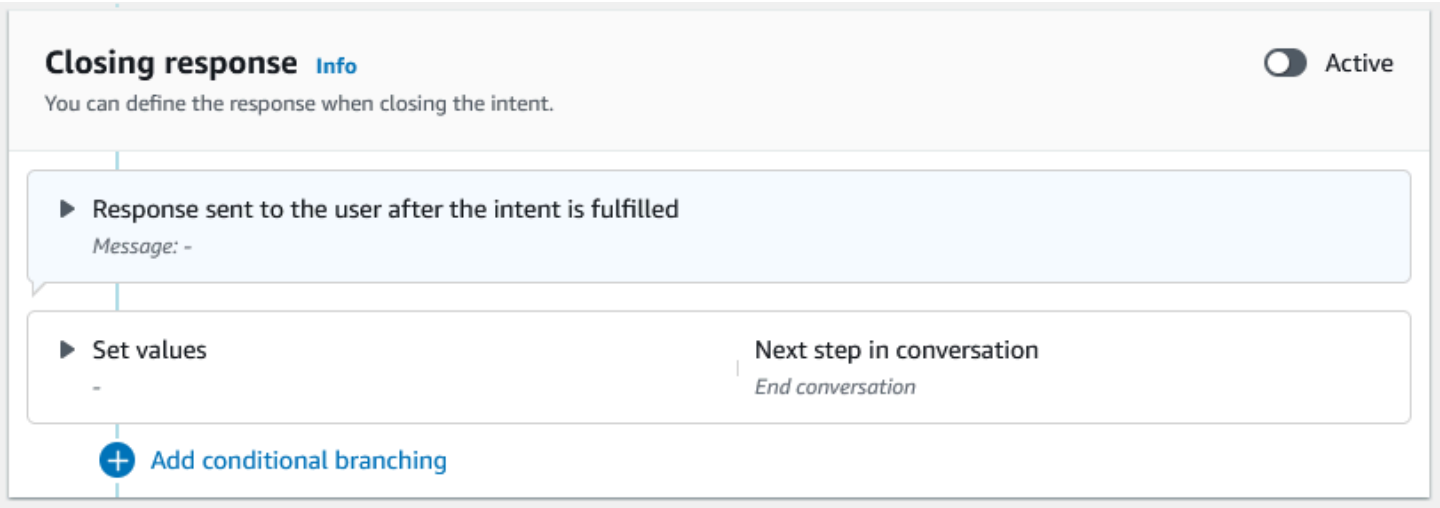
閉幕回應

結束回應會在您的使用者意圖完成後傳送給使用者。您可以使用結束回應來結束對話，也可以使用它讓使用者知道他們可以繼續進行其他意圖。例如，在旅行預訂機器人中，您可以將預訂酒店房間意圖的結束響應設置為以下內容：

```
好吧，我已經訂好了你的酒店房間 還有什麼我可以幫你的嗎？
```

您可以設定值、設定後續步驟，並在結束回應之後套用條件以設計交談路徑。在沒有條件或明確的下一個步驟的情況下，Amazon Lex V2 會結束對話。

如果您未提供結束回應，或者沒有任何條件評估為 true，Amazon Lex V2 會結束與您的機器人的對話。

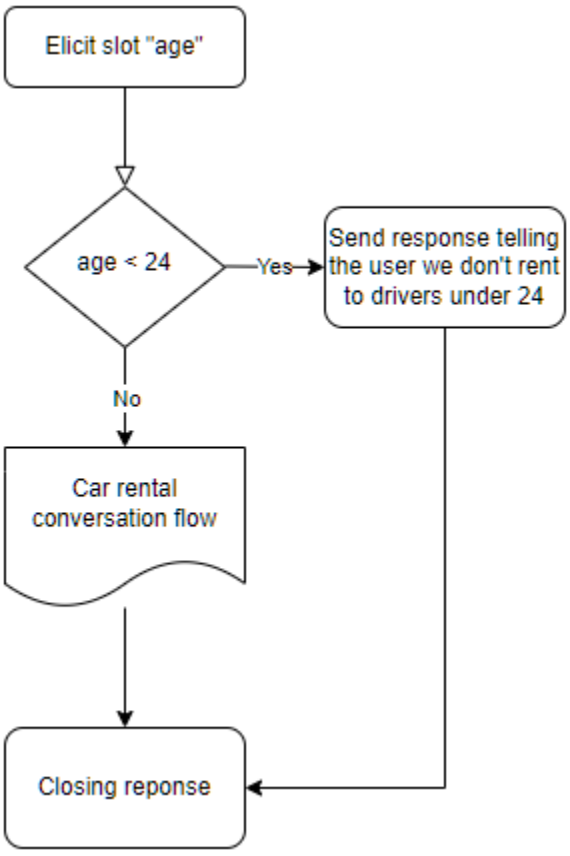


> **Note**
>
> 2022 年 8 月 17 日，Amazon Lex V2 發佈了與使用者對話管理方式的變更。此變更可讓您更好地控制使用者通過交談的路徑。如需詳細資訊，請參閱了解交談流程管理。在 2022 年 8 月 17 日之前建立的機器人不支援對話方塊程式碼掛接訊息、設定值、設定後續步驟和新增條件。

建立交談路徑

一般而言，Amazon Lex V2 會管理與使用者的交談流程。對於簡單的漫遊器，默認流程足以為您的用戶創建良好的體驗。但是，對於更複雜的漫遊器，您可能需要控制對話並將流程引導到更複雜的路徑。

例如，在預訂汽車租賃的機器人中，您可能不會租給年輕的駕駛員。在這種情況下，您可以創建一個條件來檢查驅動程序是否低於特定年齡，如果是，則跳轉到結束響應。



若要設計這類互動，您可以在交談中的每個點設定下一個步驟、評估條件、設定值，以及叫用程式碼掛接。

條件式分支可協助您透過複雜的互動為使用者建立路徑。您可以在將對話控制權傳遞給機器人的任何時候使用條件分支。例如，您可以在機器人產生第一個插槽值之前創建一個條件，您可以在引出每個插槽

值之間創建一個條件，或者您可以在機器人關閉對話之前創建一個條件。如需可新增條件的位置清單，請參閱新增意圖。

建立機器人時，Amazon Lex V2 會根據插槽的優先順序，透過交談建立預設路徑。若要自訂交談路徑，您可以在交談中的任何時間點修改下一個步驟。如需詳細資訊，請參閱 設定交談中的後續步驟。

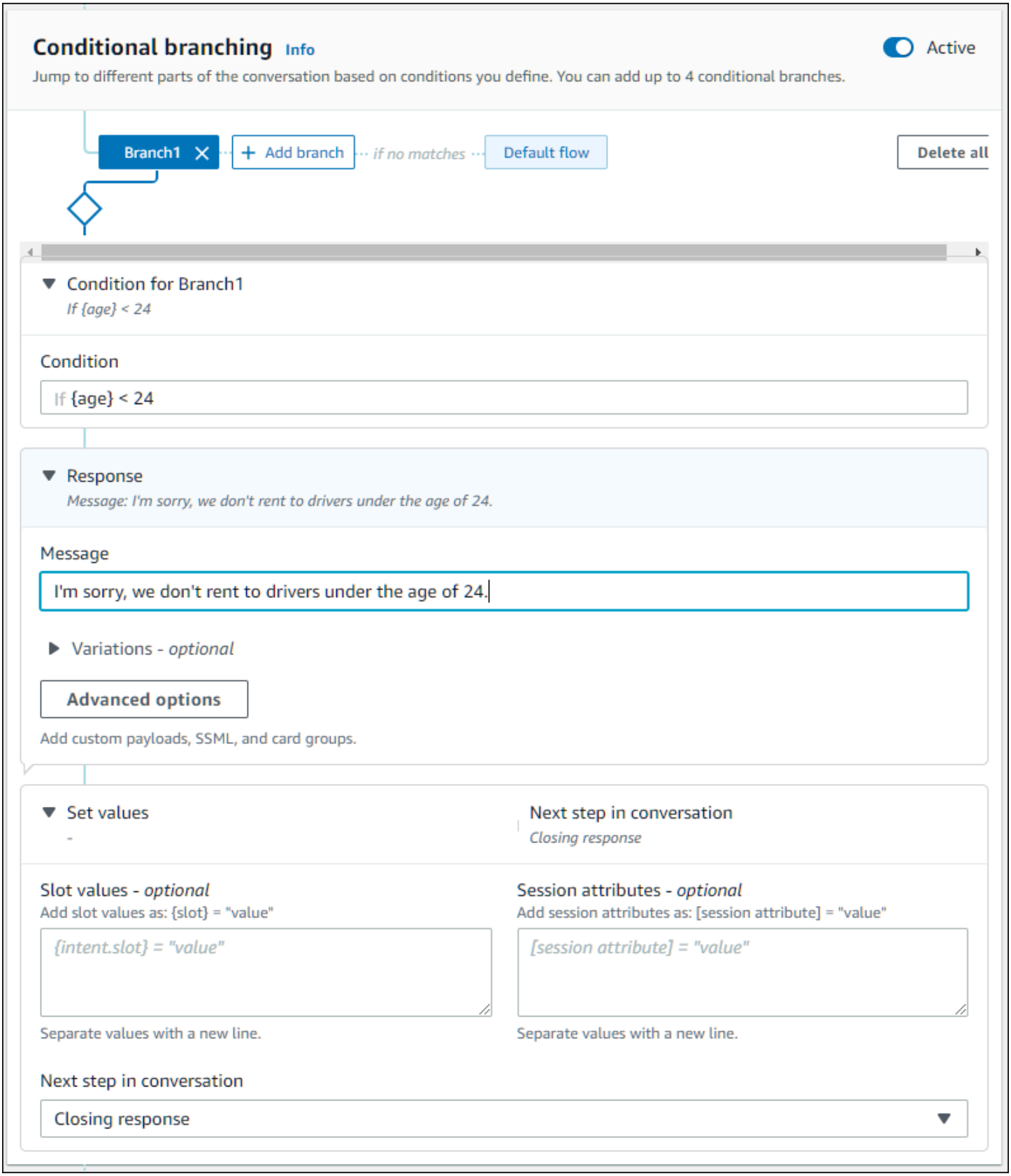
若要根據條件建立替代路徑，您可以在交談中的任何時間點使用條件式分支。例如，您可以在機器人產生第一個插槽值之前建立條件。您可以在引出每個插槽值之間創建一個條件，也可以在機器人關閉對話之前創建一個條件。如需允許您新增條件的位置清單，請參閱向分支對話添加條件。

您可以根據插槽值、工作階段屬性、輸入模式和輸入記錄，或 Amazon Kendra 的回應來設定條件。

您可以在交談中的每個點設定位置和工作階段屬性值。如需詳細資訊，請參閱 在交談期間設定值。

您也可以將下一個動作設定為對話方塊程式碼掛接，以執行 Lambda 函數。如需詳細資訊，請參閱 調用對話框代碼掛鉤。

下圖顯示在主控台中為插槽建立路徑的方式。在這個例子中，Amazon Lex V2 將引出插槽「年齡」。如果插槽的值小於 24，Amazon Lex V2 跳轉到關閉響應，否則 Amazon Lex 將遵循默認路徑。


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

> **Note**
>
> 2022 年 8 月 17 日，Amazon Lex V2 發佈了與使用者對話管理方式的變更。此變更可讓您更好地控制使用者通過交談的路徑。如需詳細資訊，請參閱 了解交談流程管理。在 2022 年 8 月 17 日之前建立的機器人不支援對話方塊程式碼掛接訊息、設定值、設定後續步驟和新增條件。

設定交談中的後續步驟

您可以在對話的每個階段設定下一個步驟，以設計對話。一般而言，Amazon Lex V2 會依照下列順序，為交談的每個階段自動設定預設後續步驟。

初始回應 → 插槽引出 → 確認（如果啟用）→ 履行（如果啟用）→ 結束回應（如果啟用）→ 結束對話

您可以修改預設的後續步驟，並根據預期的使用者經驗設計交談。您可以在交談的每個階段設定下列後續步驟：

跳轉到

- 初始回應 — 交談會從意圖開始重新啟動。設定下一步時，您可以選擇略過初始回應
- 引出一個插槽-您可以引出意圖中的任何插槽。
- 評估條件 — 您可以在對話的任何步驟評估條件和分支對話。
- 叫用對話方塊程式碼掛接 — 您可以在任何步驟叫用商務邏輯。
- 確認意圖 — 系統會提示使用者確認意圖。
- 履行意圖 — 意圖的履行將作為下一個步驟開始。
- 結束回應 — 結束回應會傳回給使用者。

切換到

- 意圖 — 您可以轉換為不同的意圖，並為此目的繼續對話。您可以選擇性地在進行轉移時略過意圖的初始回應。
- 意圖：特定槽 — 如果您已經在目前意圖中擷取了某些槽值，您可以直接在不同意圖中引出特定槽。

等待使用者輸入 — 機器人會等待使用者提供識別任何新意圖的輸入。您可以配置提示，例如「還有其他我可以幫助您的嗎？」 在設置此下一步之前。機器人將處於`ElicitIntent`對話狀態。

結束對話 — 與機器人的對話已關閉。

> **Note**
>
> 2022 年 8 月 17 日，Amazon Lex V2 發佈了與使用者對話管理方式的變更。此變更可讓您更好地控制使用者通過交談的路徑。如需詳細資訊，請參閱 了解交談流程管理。在 2022 年 8 月 17 日之前建立的機器人不支援對話方塊程式碼掛接訊息、設定值、設定後續步驟和新增條件。

在交談期間設定值

Amazon Lex V2 可讓您在交談的每個步驟中設定插槽值和工作階段屬性值。然後，您可以在交談期間使用這些值來評估條件或在意圖履行期間使用它們。

您可以設定目前意圖的槽值。如果交談中的下一個步驟是調用另一個意圖，則可以設置新意圖的插槽值。

如果指派的位置未填滿，或者無法剖析 JSON 路徑，則屬性將設定為`null`。

使用插槽值和工作階段屬性時，請使用下列語法：

- 插槽值 — 用大括號 (「{}」) 圍住插槽名稱。對於目前意圖中的槽值，您只需要使用槽名稱。例如 `{slot}`。如果您要在下一個意圖中設定值，則必須同時使用意圖名稱和槽名稱來識別槽。例如 `{intent.slot}`。

  範例：

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

  插槽的值可以是下列任何一種：

  - 一個常量字符串
  - 一個 JSON 路徑，指向 Amazon Lex 回應中的轉錄區塊 (適用於 en-US 和 en-GB)
  - 一個會話屬性

  範例：

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

> **Note**
>
> 槽值也可以設定為null。如果您需要重新引出已填滿的插槽值，則必須null先將值設定為，然後再次提示客戶輸入插槽值。如果指派的位置未填滿，或者無法剖析 JSON 路徑，則屬性將設定為null。

- 工作階段屬性 — 用方括號 (「[]」) 圍住屬性名稱。例如 [sessionAttribute]。

  範例：

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

  會話屬性的值可以是以下任何一種：

  - 一個常量字符串
  - 一個 JSON 路徑，指向 Amazon Lex 回應中的轉錄區塊 (適用於 en-US 和 en-GB)
  - 槽值參考

  > **Note**
  >
  > 如果指派的位置未填滿，或者無法剖析 JSON 路徑，則屬性將設定為null。

> **Note**
>
> 2022 年 8 月 17 日，Amazon Lex V2 發佈了與使用者對話管理方式的變更。此變更可讓您更好地控制使用者通過交談的路徑。如需詳細資訊，請參閱 了解交談流程管理。在 2022 年 8 月 17 日之前建立的機器人不支援對話方塊程式碼掛接訊息、設定值、設定後續步驟和新增條件。

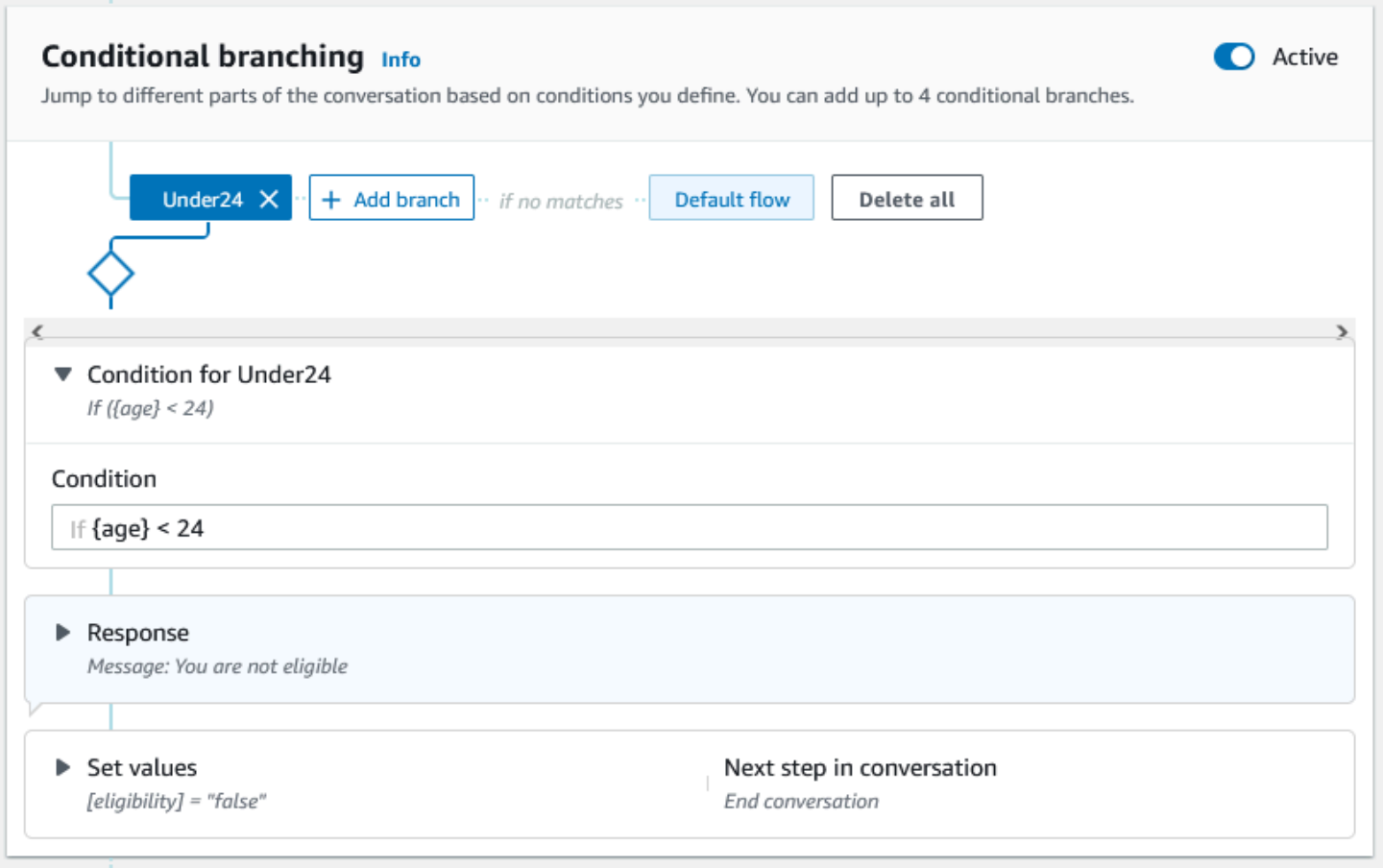
向分支對話添加條件

您可以使用條件分支來控制客戶通過與機器人交談的路徑。您可以根據插槽值、工作階段屬性、輸入模式和輸入成績單欄位的內容，或 Amazon Kendra 的回應來分支對話。

您最多可以定義四個分支。每個分支都有必須滿足的條件，Amazon Lex V2 才能跟隨該分支。如果沒有任何分支滿足其條件，則會遵循默認分支。

定義分支時，您可以定義 Amazon Lex V2 在與該分支對應的條件評估為真時應採取的動作。您可以定義下列任何動作：

- 傳送給使用者的回應。
- 要套用至狹槽的槽值。
- 目前階段作業的階段作業屬性值。
- 交談中的下一個步驟。如需詳細資訊，請參閱 建立交談路徑。



每個條件分支都有一個布林運算式，Amazon Lex V2 必須滿足該分支。您可以在條件中使用比較和布林運算子、函數和量詞運算子。例如，如果 {age} 插槽小於 24，則下列條件會傳回 true。

```
{age} < 24
```

如果 {澆頭} 多值插槽包含單詞「菠蘿」，則以下條件返回 true。

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

對於更複雜的條件，您可以將多個比較運算子與布林運算子結合。例如，如果 {make} 槽值是「本田」和 {模型} 槽值是「思域」下面的條件返回 true。使用括號來設定評估順序。

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

下列主題提供有關條件式分支運算子和函數的詳細資訊。

> **Note**
>
> 2022 年 8 月 17 日，Amazon Lex V2 發佈了與使用者對話管理方式的變更。此變更可讓您更好地控制使用者通過交談的路徑。如需詳細資訊，請參閱 了解交談流程管理。在 2022 年 8 月 17 日之前建立的機器人不支援對話方塊程式碼掛接訊息、設定值、設定後續步驟和新增條件。

主題

- 比較運算子
- 布林值運算子
- 數量詞運算子
- 函數
- 條件運算式範例

比較運算子

Amazon Lex V2 針對條件支援下列比較運算子：

- 等於 (=)
- 不等於 (! =)
- 小於 (<)
- 小於或等於 (<=)
- 大於 (>)
- 大於或等於 (>=)

使用比較運算子時，會使用下列規則。

- 左側必須是參考。例如，若要參考槽值，請使用{slotName}。若要參照工作階段屬性值，請使用[attribute]。對於輸入模式和輸入成績單，您可以使用$.inputMode和$.inputTranscript。
- 右側必須是常數且與左側相同的類型。
- 任何參考尚未設定之屬性的運算式都會被視為無效，且不會進行評估。
- 當您比較多值插槽時，使用的值是所有解譯值的逗號分隔清單。

比較以參照的槽類型為基礎。它們解決方法如下：

- 字串 — 字串會根據其 ASCII 表示法進行比較。比較不區分大小寫。
- Nu@@ mbers — 以數字為基礎的槽會從字串表示轉換成數字，然後進行比較。
- 日期/時間 — 以時間為基礎的時段會根據時間序列進行比較。較早的日期或時間被認為較小。對於持續時間，較短的週期被認為是較小的。

布林值運算子

Amazon Lex V2 支持布爾運算符結合比較運算符。它們可讓您建立類似下列內容的陳述式：

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

您可以使用下列布林運算子：

- 與 (&&)
- 或者 (
- 不 (！)

數量詞運算子

數量詞運算子會評估序列的項目，並判斷一或多個項目是否滿足條件。

- 包含 — 確定指定的值是否包含在多值插槽中，如果是，則傳回 true。例如，如果用戶在他們的比薩餅上訂購了菠蘿，則{toppings} CONTAINS "pineapple"返回 true。

函數

函數必須加上字串fn.的前綴。該函數的參數是一個插槽，會話屬性，或請求屬性的引用。Amazon Lex V2 提供兩種功能，可從插槽、工作階段屬性或要求屬性的值取得資訊。

- Fn.count () — 計算多值插槽中的值數目。

  例如，如果插槽{toppings}包含值「意大利辣香腸，菠蘿」：

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

- Fn.is_set () — 如果在目前工作階段中設定插槽、工作階段屬性或要求屬性，則值為 true。

  根據前面的例子：

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

- fn.length ()-value 是在當前會話中設置的會話屬性，插槽值或插槽屬性的值的長度。此功能不支援多值插槽或複合插槽。

  範例：

  如果插槽{credit-card-number}包含值「123456781234」：

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

條件運算式範例

以下是一些條件運算式範例。注意：$.代表 Amazon Lex JSON 響應的入口點。下列值$.將在 Amazon Lex 回應中進行剖析，以擷取值。只有在支援 ASR 轉錄分數的相同語言環境中，才支援使用對 Amazon Lex 回應中轉錄區塊的 JSON 路徑參照的條件運算式。

值類型	使用案例	條件運算式
自定義插槽	pizzaSize 插槽值等於大	{pizzaSize} = "large"
自定義插槽	pizzaSize 等於大或中	{pizzaSize} = "large"或 {pizzaSize} = "medium"
自定義插槽	使用()和的表達式 AND/OR	{pizzaType} = "pepperoni" 或{pizzaSiz

值類型	使用案例	條件運算式
		e} = "medium" 或 {pizzaSize} = "small"
自訂插槽 (多值插槽)	檢查其中一個澆頭是否是洋蔥	{toppings} CONTAINS "Onion"
自訂插槽 (多值插槽)	澆頭數量超過 3	fn.COUNT({topping}) > 2
AMAZON.AlphaNumeric	bookingID 是 ABC123	{bookingID} = "ABC123"
AMAZON.Number	年齡插槽值大於 30	{age} > 30
AMAZON.Number	年齡插槽值等於 10	{age} = 10
AMAZON.Date	dateOfBirth 1990 年之前的插槽值	{dateOfBirth} < "1990-10-01"
AMAZON.State	destinationState 插槽值等於華盛頓	{destinationState} = "washington"
AMAZON.Country	destinationCountry 插槽值不是美國	{destinationCountry} != "united states"
AMAZON.FirstName	firstName 插槽值是約翰	{firstName} = "John"
AMAZON.PhoneNumber	phoneNumber 插槽的值是 716767891932	{phoneNumer} = 716767891932
AMAZON.Percentage	檢查插槽百分比值是否大於或等於 78	{percentage} >= 78
AMAZON.EmailAddress	emailAddress 插槽值為 userA@hmail.com	{emailAddress} = "userA@hmail.com"
AMAZON.LastName	lastName插槽值是 Doe	{lastName} = "Doe"
AMAZON.City	城市插槽值等於西雅圖	{city} = "Seattle"

值類型	使用案例	條件運算式
AMAZON.Time	時間是晚上 8 點以後	{time} > "20:00"
AMAZON.StreetName	streetName 插槽值是博仁大道	{streetName} = "boren avenue"
AMAZON.Duration	travelDuration 插槽值小於 2 小時	{travelDuration} < P2H
輸入模式	輸入模式為語音	$.inputMode = "Speech"
輸入成績單	輸入成績單等於「我想要一個大披薩」	$.inputTranscript = "I want a large pizza"
階段屬性	檢查客戶訂閱類型屬性	[customer_subcription_type] = "yearly"
請求屬性	檢查已啟用重新嘗試的旗標	((retry_enabled)) = "TRUE"
Kendra 回應	Kendra 響應包含常見問題	fn.IS_SET(((x-amz-lex:kendra-search-response-question_answer-question-1))
帶有轉錄的條件表達式	使用轉錄 JSON 路徑的條件運算式	$.transcriptions[0].transcriptionConfidence < 0.8 AND $.transcriptions[1].transcriptionConfidence > 0.5

值類型	使用案例	條件運算式
設定階段屬性	使用轉錄 JSON 路徑和插槽值設置會話屬性	`[sessionAttribute] = "$.transcriptions...." AND [sessionAttribute] = "{<slotName>}"`
設定插槽值	使用會話屬性和轉錄 JSON 路徑設置插槽值	`{slotName} = [<sessionAttribute>] AND {slotName} = "$.transcriptions...."`

> **Note**
>
> `slotName`指的是 Amazon Lex 機器人中插槽的名稱。如果插槽未解析 (null)，或插槽不存在，則會在執行時間忽略指派。 `sessionAttribute`指的是由客戶在構建時設置的會話屬性的名稱。

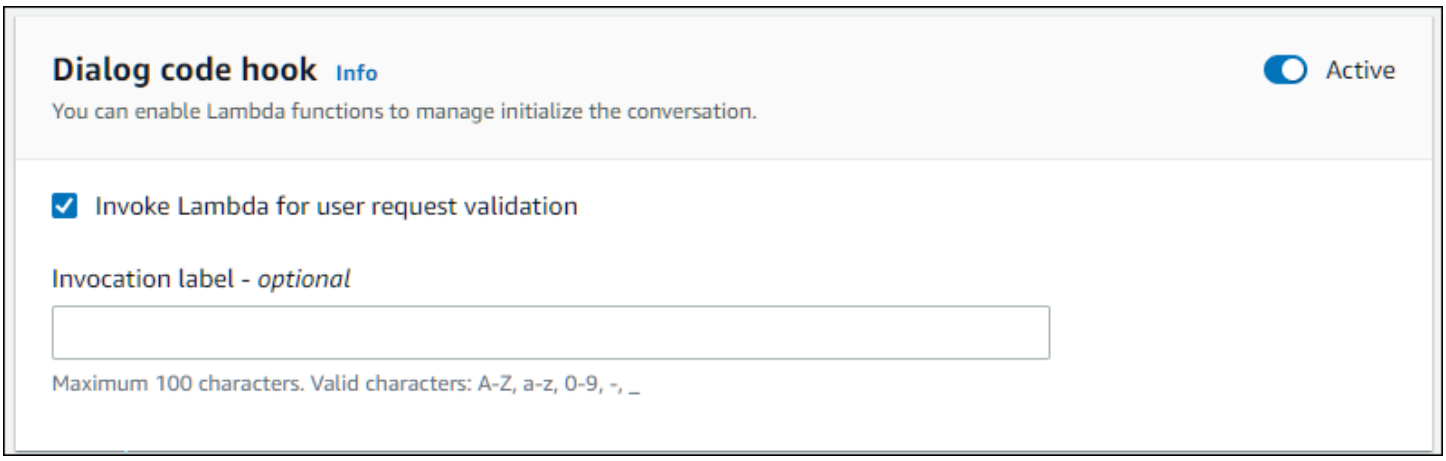
調用對話框代碼掛鉤

當 Amazon Lex 傳送訊息給使用者時，在交談中的每個步驟中，您都可以使用 Lambda 函數做為交談的下一個步驟。您可以使用函數根據交談的目前狀態來實作商務邏輯。

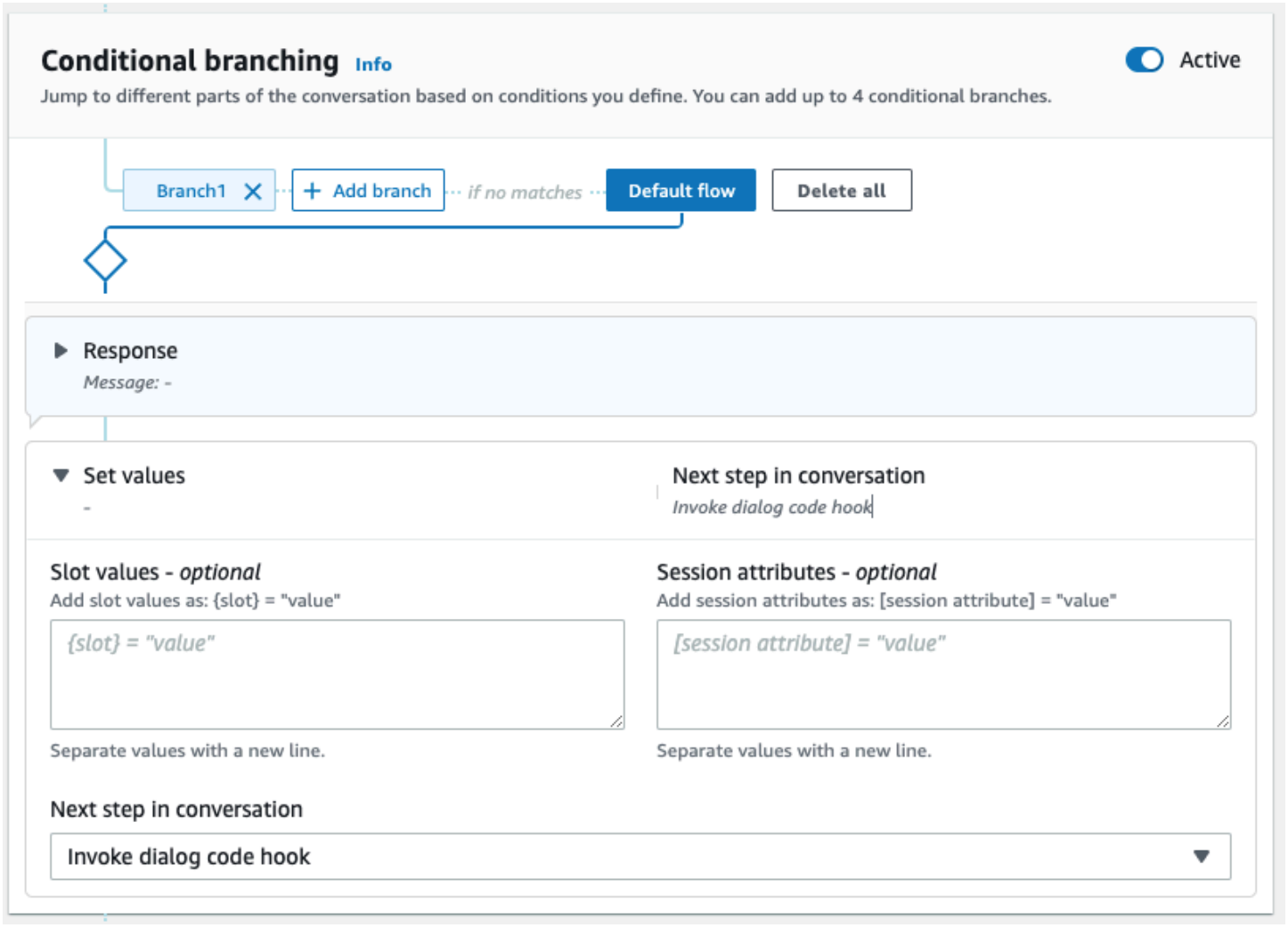
執行的 Lambda 函數與您正在使用的機器人別名相關聯。若要在意圖中的所有對話方塊程式碼掛接中叫用 Lambda 函數，您必須選取「使用 Lambda 函數來初始化和驗證意圖」。如需選擇 Lambda 函數的詳細資訊，請參閱建立 Lambda 函數並將其附加至機器人別名。

使用 Lambda 函數有兩個步驟。首先，您必須在對話中的任何時間點啟動對話方塊程式碼勾點。其次，您必須設定交談中的下一個步驟，才能使用對話方塊程式碼掛接。

下圖顯示了對話框代碼鉤激活。

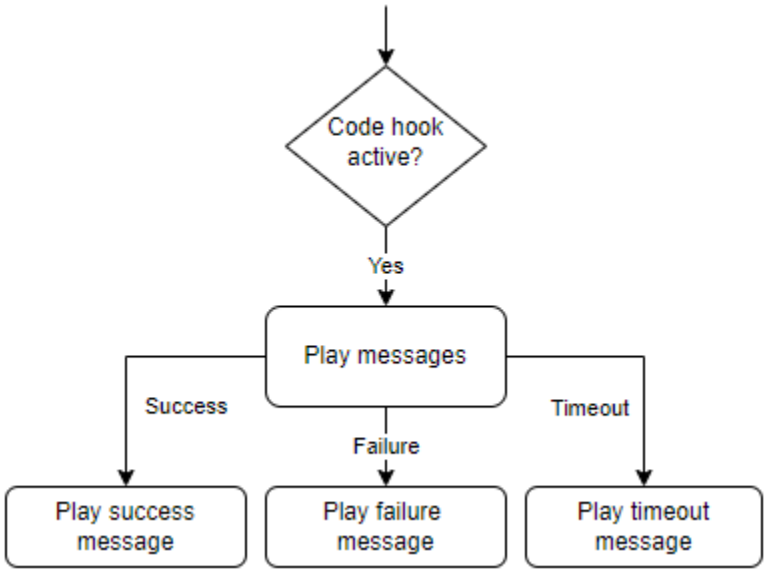


接下來，將程式碼掛接設定為交談步驟的下一個動作。您可以將交談中的下一個步驟設定為叫用對話方塊程式碼掛接來執行此操作。下圖顯示了一個條件分支，其中調用對話框代碼掛鉤是交談的默認路徑的下一個步驟。



當程式碼掛接處於作用中狀態時，您可以設定三個回應以傳回給使用者：

- 成功 — 當 Lambda 函數成功完成時傳送。
- 失敗 — 如果執行 Lambda 函數發生問題，或 Lambda 函數傳回的`intent.state`值為，則傳送`Failed`。
- 逾時 — 如果 Lambda 函數未在設定的逾時期間內完成，則傳送。



選擇 Lambda 對話方塊程式碼掛接，然後選擇進階選項，以查看對應於 Lambda 函數叫用的三個回應選項。您可以設定值、設定後續步驟，以及套用與每個回應相對應的條件，以設計交談流程。在沒有條件或明確的下一個步驟的情況下，Amazon Lex V2 會根據交談的目前狀態決定下一個步驟。

在 [進階選項] 頁面上，您也可以選擇啟用或停用 Lambda 函數叫用。啟用函數時，會使用 Lambda 叫用來叫用對話方塊程式碼掛接，然後根據 Lambda 叫用結果顯示成功、失敗或逾時訊息。停用函數時，Amazon Lex V2 不會執行 Lambda 函數，而且會像對話方塊程式碼掛接成功一樣繼續進行。

您也可以設定在此訊息叫用 Lambda 函數時傳送至 Lambda 函數的叫用標籤。您可以使用此功能來協助識別要執行的 Lambda 函數部分。

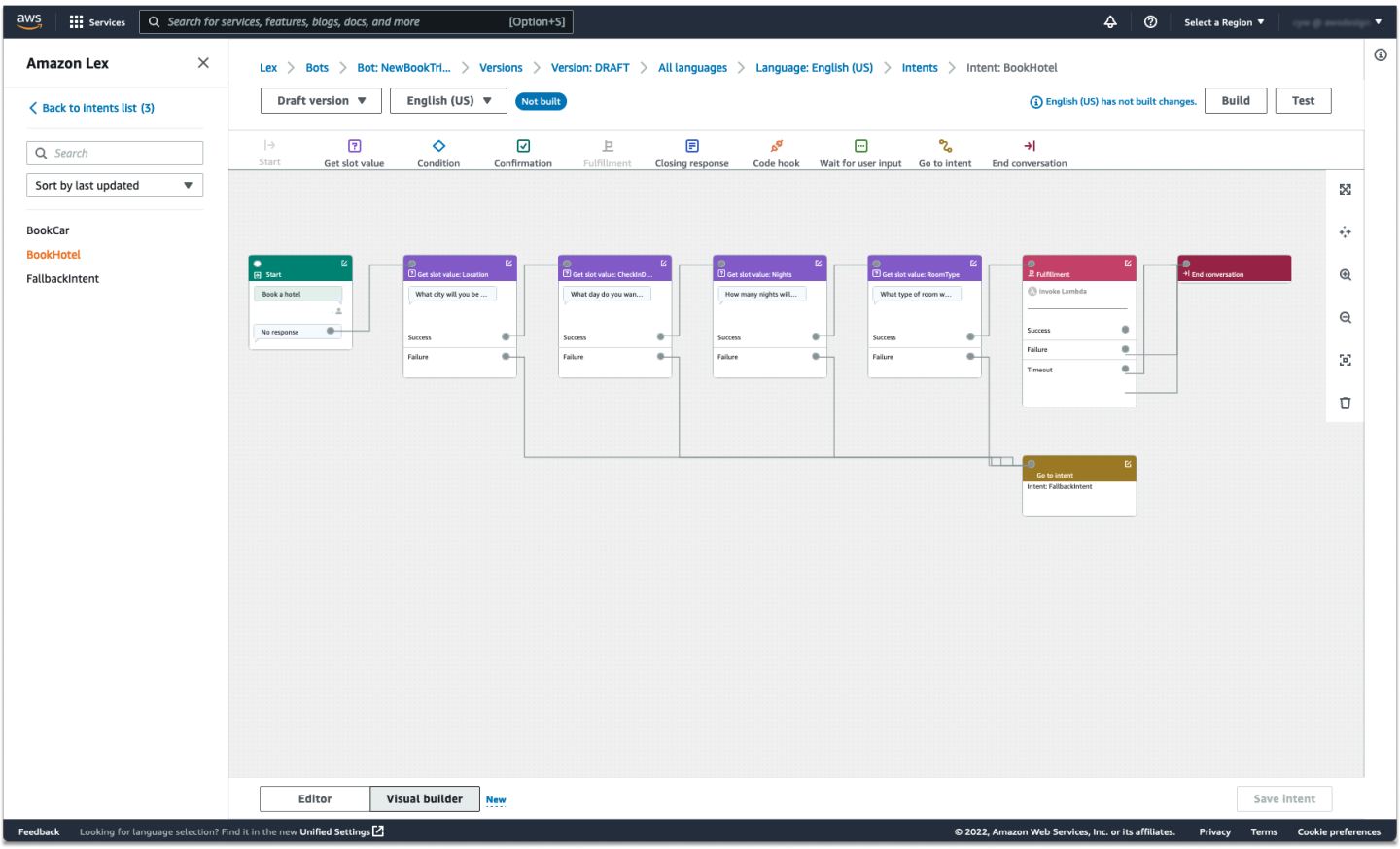
> **Note**
>
> 2022 年 8 月 17 日，Amazon Lex V2 發佈了與使用者對話管理方式的變更。此變更可讓您更好地控制使用者通過交談的路徑。如需詳細資訊，請參閱 了解交談流程管理。在 2022 年 8 月 17 日之前建立的機器人不支援對話方塊程式碼掛接訊息、設定值、設定後續步驟和新增條件。

使用視覺對話生成器

視覺對話建置器是拖放式對話建置工具，可在豐富的視覺環境中使用意圖，輕鬆設計和視覺化對話路徑。

若要存取視覺化對話產生器

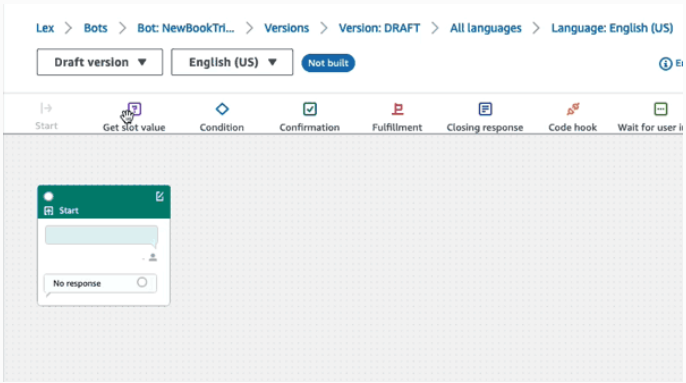
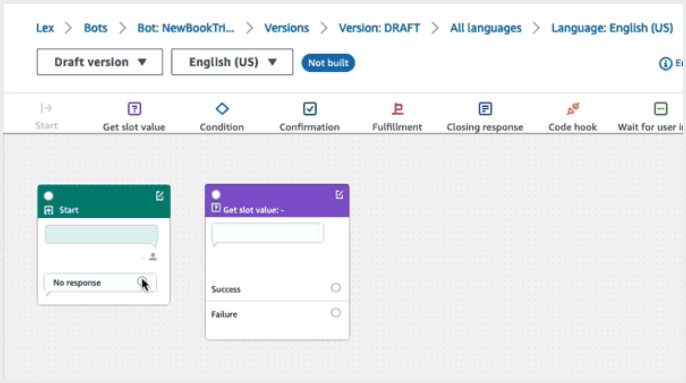
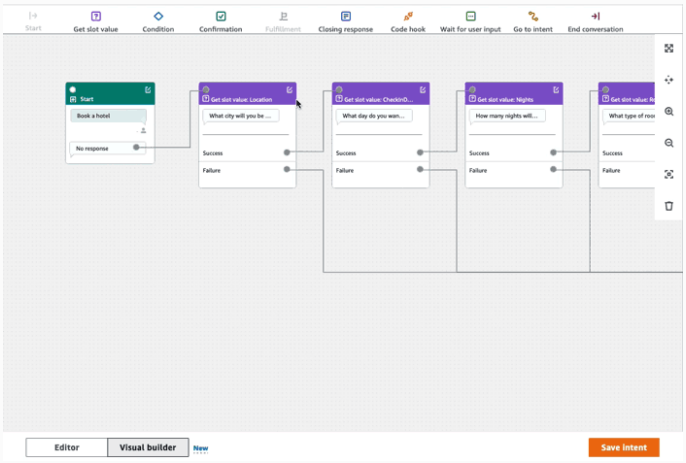
1. 在 Amazon Lex V2 主控台中，選擇機器人並選取意圖從左側導航窗格。
2. 請使用下列其中一種方式前往意圖編輯器：
   - 选择加入意圖在右上角意圖區段，然後選擇新增空白意圖或內建意圖。
   - 從中選擇意圖的名稱意圖部分。
3. 在意圖編輯器中，選取視覺生成器在屏幕底部的窗格中訪問視覺對話生成器。
4. 若要返回選單意圖編輯器介面，請選取編輯。

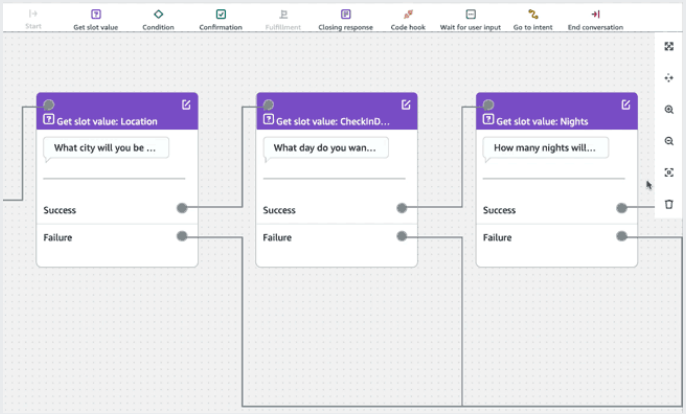
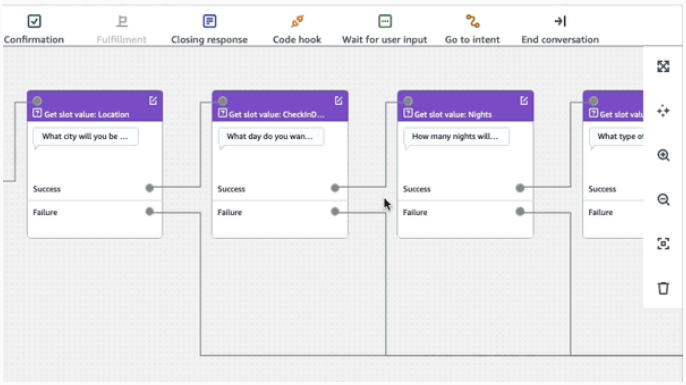
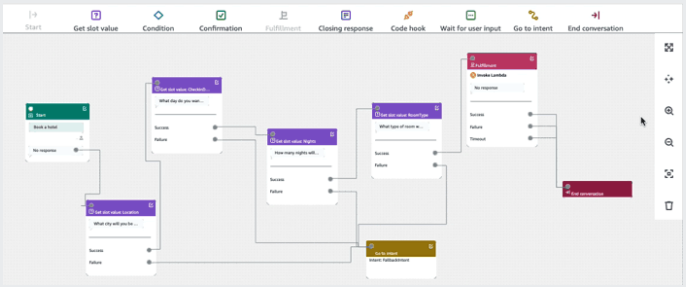


視覺化對話建立器提供了更直觀的使用者介面，能夠視覺化和修改對話流程。透過拖放區塊，您可以延伸現有流程或重新排序交談步驟。您可以開發具有複雜分支的對話流程，而無需撰寫任何 Lambda 程式碼。

此變更有助於將對話流程設計與 Lambda 中的其他商務邏輯分離。視覺化交談產生器可與現有的意圖編輯器搭配使用，並可用來建立交談流程。不過，建議您針對更複雜的交談流程使用視覺化編輯器檢視。

當您儲存意圖時，Amazon Lex V2 可以在判斷有遺漏連線時自動連線意圖、Amazon Lex V2 建議連線，或者您可以為區塊選取自己的連線。

動作	範例
將圖塊新增至工作區	
在塊之間建立連接	
開啟區塊上的組態面板	

動作	範例
縮放至佈滿	
從交談流程中刪除區塊	
自動清理工作區	

術語：

阻止— 交談流程的基本建置單位。每個塊都有一個特定的功能來處理對話的不同用例。

連接埠— 每個塊包含端口，可用於將一個塊連接到另一個塊。塊可以包含輸入端口和輸出端口。每個輸出端口代表塊的特定功能變化（例如錯誤，超時或成功）。

邊-邊緣是一個塊的輸出端口與另一塊的輸入端口之間的連接。它是交談流程中分支的一部分。

對話流程-一組由邊緣連接的塊，描述了與客戶的意圖級別的交互。

圖塊

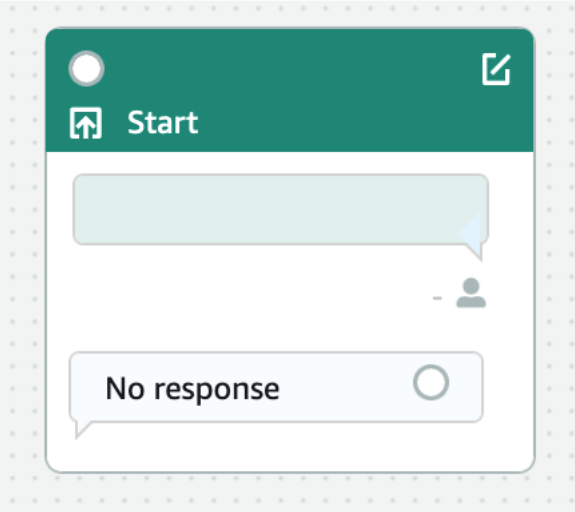
區塊是交談流程設計的建置區塊。它們代表意圖中的不同狀態，從意圖開始到用戶輸入到結束。

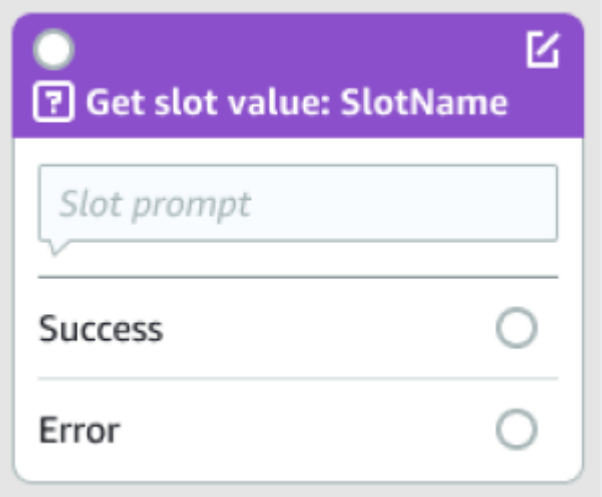
每個區塊都有一個進入點和一個或多個基於圖塊類型的出口點。當交談通過退出點進行時，可以使用對應的消息配置每個退出點。對於具有多個退出點的圖塊，退出點與節點對應的狀態有關。對於條件節點，退出點代表不同的條件。

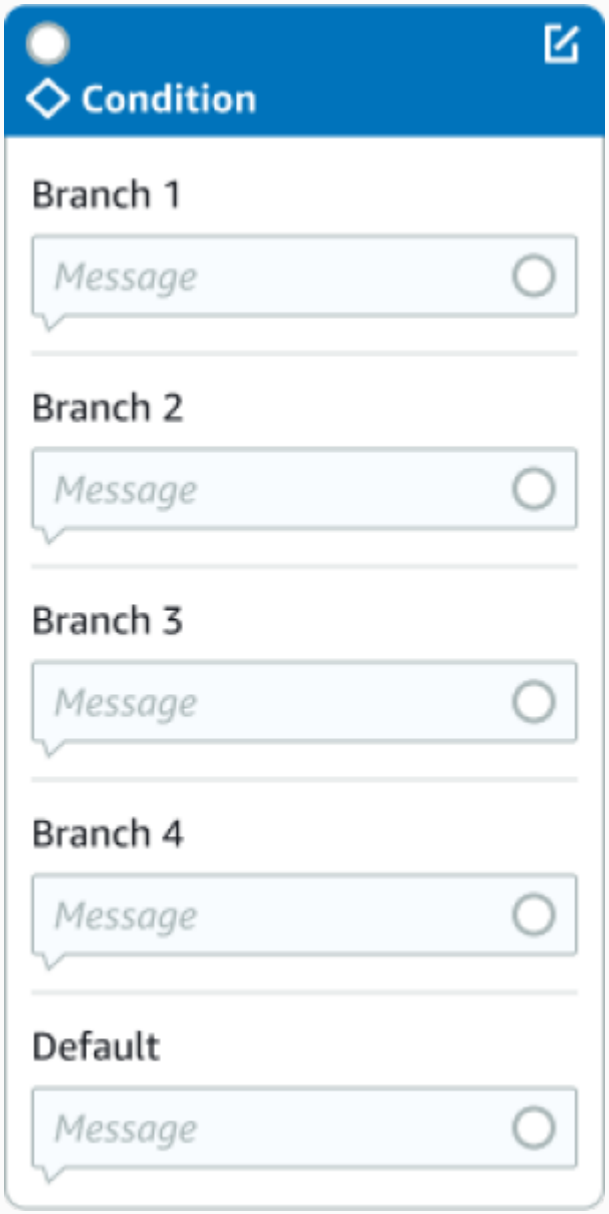
每個塊都有一個配置面板，通過單擊編輯圖標位於塊的右上角。配置面板包含可以配置為與每個塊相對應的詳細字段。

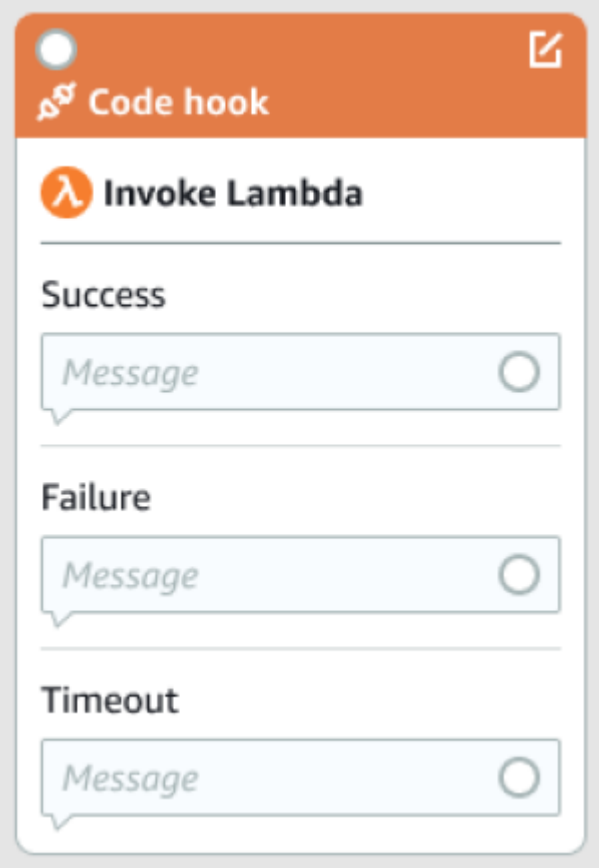
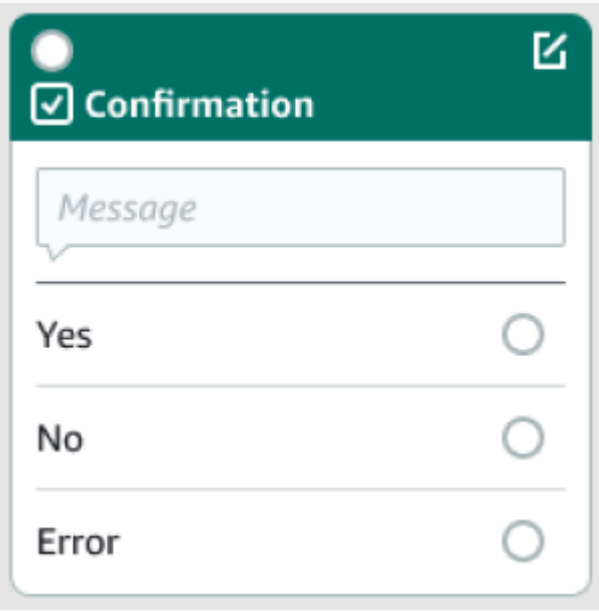
機器人提示和消息可以通過拖動新塊直接在節點上配置，也可以在右側面板中修改它們以及塊的其他屬性。

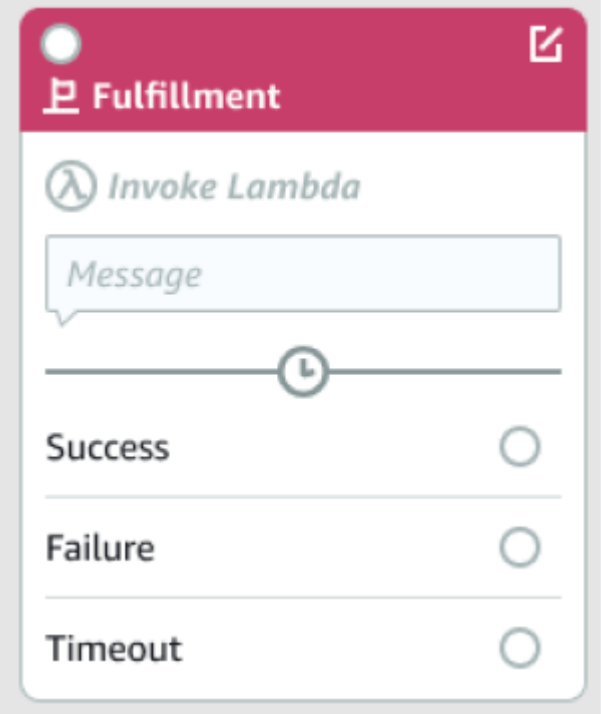
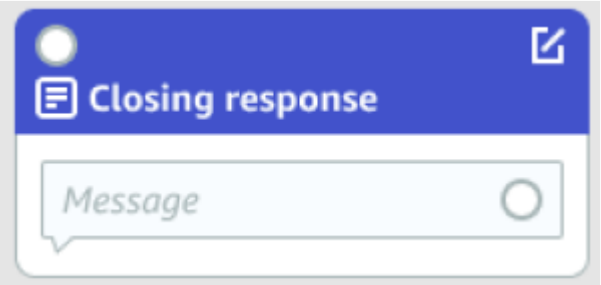
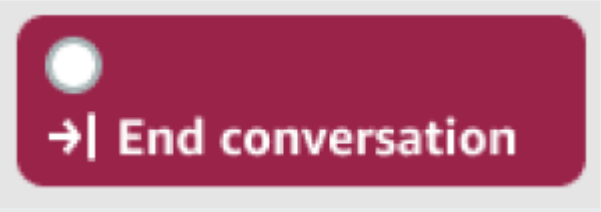
圖塊類型— 以下是您可以與視覺對話構建器一起使用的塊類型。

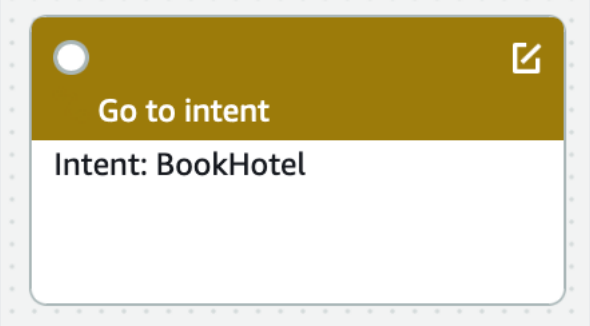
圖塊類型	封鎖
開始— 交談流程的根或第一個區塊。也可以配置此塊，以便機器人可以發送初始響應（意圖已被識別的消息）。如需詳細資訊，請參閱初步回應。	

圖塊類型	封鎖
獲取插槽值— 此區塊試圖引出單一插槽的值。此區塊具有等待客戶回應插槽引出提示的設定。如需詳細資訊，請參閱槽。	

圖塊類型	封鎖
條件— 此區塊包含條件。它最多包含 4 個自定義分支（帶條件）和一個默認分支。如需詳細資訊，請參閱向分支對話添加條件。	

圖塊類型	封鎖
對話框代碼掛鉤— 此區塊會處理對話方塊 Lambda 函數的呼叫。此區塊包含以對話方塊 Lambda 函數成功、失敗或逾時為基礎的機器人回應。如需詳細資訊，請參閱調用對話框代碼掛鉤。	
確認— 此塊在實現意圖之前查詢客戶。它包含根據客戶對確認提示說「是」或「否」的機器人回應。如需詳細資訊，請參閱確認。	

圖塊類型	封鎖
履行-此塊處理意圖的實現，通常在插槽引出之後。它可以設定為叫用 Lambda 函數，以及在履行成功或失敗時回應訊息。如需詳細資訊，請參閱實現。	
閉幕回應— 此塊允許機器人在結束對話之前用消息做出響應。如需詳細資訊，請參閱閉幕回應。	
結束對話— 此區塊表示交談流程的結束。	
等待使用者輸入— 該塊可用於捕獲客戶的輸入，並根據話語切換到另一個意圖。	

圖塊類型	封鎖
轉到意圖-該塊可用於轉到新的意圖，或直接引出該意圖的特定插槽。	

連接埠類型

所有塊都包含一個輸入端口，用於連接其父塊。對話只能從其父塊的輸出端口流向特定塊的輸入端口。但是，區塊可以包含零個、一個或多個輸出連接埠。沒有任何輸出端口的塊表示當前意圖中的對話流程的結束（GoToIntent,EndConversation,WaitForUserInput).

意圖設計規則：

- 意圖中的所有流程都以起始區塊開始。
- 與每個出口點對應的消息是可選的。
- 您可以配置塊以設置與配置面板中的每個退出點相對應的值。
- 意圖中的單一流程中只能存在單一開始、確認、履行和關閉區塊。多種條件，對話框代碼掛鉤，獲取槽值，結束對話，傳輸，等待用戶輸入塊可能存在。
- 條件區塊不能直接連接到條件區塊。這同樣適用於對話框代碼掛鉤。
- 循環流程允許使用三個區塊，但不允許使用「啟動意圖」的傳入連接器。
- 選擇性插槽沒有傳入連接器或傳出連線，主要用於擷取意圖引出期間存在的任何資料。屬於交談路徑一部分的每個其他插槽都必須是強制插槽。

區塊：

- 起始圖塊必須具有外出邊。
- 如果需要插槽，每個 get 槽值區塊都必須具有來自成功連接埠的外出邊緣。
- 如果圖塊處於活動狀態，則每個條件區塊都必須具有來自每個分支的外出邊緣。
- 一個條件圖塊不能有一個以上的父系。
- 作用中條件圖塊必須具有進入邊。

- 每個活動代碼鉤子塊必須具有來自每個端口的傳出邊緣：成功，失敗和超時。
- 活動的代碼鉤子塊必須具有傳入邊緣。
- 作用中的確認圖塊必須具有進入邊。
- 作用中的出貨區塊必須具有傳入邊緣。
- 作用中的封閉圖塊必須具有進入邊。
- 條件區塊必須至少有一個非預設分支。
- 轉到意圖塊必須指定一個意圖。

邊緣：

- 條件圖塊無法連接至另一個條件圖塊。
- 程式碼鉤子區塊無法連接到另一個程式碼鉤子區塊。
- 一個條件塊只能連接到零個或一個代碼鉤子塊。
- 連接（代碼掛鉤-> 條件-> 代碼鉤子）無效。
- 出貨區塊不能將程式碼掛接區塊設為子項。
- 條件區塊 (屬於履行區塊的子項) 不能有程式碼勾點區塊子項。
- 關閉塊不能將代碼鉤子塊作為子代碼。
- 做為封閉區塊子項的條件區塊不能有程式碼掛接區塊子項。
- 開始、確認或 get 槽值區塊在其相依性鏈中不能有一個以上的程式碼掛接區塊。

> **Note**
>
> 2022 年 8 月 17 日，Amazon Lex V2 發佈了與使用者對話管理方式的變更。此變更可讓您更好地控制使用者通過交談的路徑。如需詳細資訊，請參閱了解交談流程管理。在 2022 年 8 月 17 日之前建立的機器人不支援對話方塊程式碼掛接訊息、設定值、設定後續步驟和新增條件。

內建槽

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

您無法在目前的實作中執行以下操作：

- 從基本意圖中新增或移除範例語彙
- 設定內建意圖的槽

若要將內建意圖新增至機器人

1. 登錄到 AWS Management Console 並打開 Amazon Lex 控制台 https://console.aws.amazon.com/lex/.
2. 選擇要新增內建意圖的機器人。
3. 在左側選單中，選擇語言，然後選擇 [意圖]。
4. 選擇 [新增方式]，然後選擇 [使用內建的方式]。
5. 在內建色彩比對方式中，選擇要使用的意圖。
6. 為意圖命名，然後選擇 [新增]。
7. 使用意圖編輯器根據您的機器人的需求配置意圖。

主題

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

AMAZON.CancelIntent

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

常見的話語：

- 取消
- 沒關係
- 忘記它

AMAZON.FallbackIntent

當使用者對意圖的輸入不是機器人預期的內容時，您可以設定 Amazon Lex V2 叫用後援意圖。例如，如果使用者輸入的「我想訂購糖果」與`OrderFlowers`機器人中的意圖不符，Amazon Lex V2 會叫用後援意圖來處理回應。

當您使用主控台建立機器人或使用CreateBotLocale操作將地區設定新增至機器人時，內建的`AMAZON.FallbackIntent`意圖類型會自動新增至您的機器人。

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。

在下列情況下，Amazon Lex V2 符合後援意圖：

- 使用者對意圖的輸入不符合機器人預期的輸入
- 音訊輸入為雜訊，或文字輸入無法辨識為文字。
- 使用者的輸入不明確，而且 Amazon Lex V2 無法判斷要叫用的意圖。

以下情況會叫用備用意圖：

- 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。

您無法將以下內容新增至備用意圖：

- 表達用語
- 槽
- 一條確認提示

使用具有後援意圖的 Lambda 函數

在呼叫備用意圖時，其回應取決於對 CreateIntent 操作的 `fulfillmentCodeHook` 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。
- 呼叫別名的驗證和履行 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 CreateIntent 操作的 `fulfillmentCodeHook` 參數。

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

您可以在相同工作階段中多次叫用備用意圖。例如，假設您的 Lambda 函數使用對`ElicitIntent`話方塊動作來提示使用者不同的意圖。如果 Amazon Lex V2 在設定的嘗試次數後無法推斷使用者的意圖，則會再次叫用後援意圖。當使用者在設定的嘗試次數之後仍未回應有效的槽值，它也會叫用備用意圖。

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

AMAZON.HelpIntent

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

常見的話語：

- help
- 幫我
- 你能幫我嗎？

AMAZON.KendraSearchIntent

若要搜尋已使用 Amazon Kendra 編製索引的文件，請使用`AMAZON.KendraSearchIntent`意圖。當 Amazon Lex V2 無法判斷與使用者交談中的下一個動作時，會觸發搜尋意圖。

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

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

- 解答
- 可能回答問題的常見問題項目
- 與問題相關的文件

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

如果您設定機器人的AMAZON.KendraSearchIntent意圖，Amazon Lex V2 會在無法判斷使用者意圖的話語時呼叫意圖。如果沒有來自 Amazon Kendra 的回應，交談會依照機器人中的設定繼續進行。

> **Note**
>
> Amazon Lex V2 目前不支援插槽引出AMAZON.KendraSearchIntent期間。如果 Amazon Lex V2 無法判斷插槽的使用者話語，它會呼叫. AMAZON.FallbackIntent

當您在相同的機器人AMAZON.FallbackIntent中AMAZON.KendraSearchIntent搭配使用時，Amazon Lex V2 會使用意圖，如下所示：

1. Amazon Lex V2 調用AMAZON.KendraSearchIntent. 意圖呼叫 Amazon Kendra Query 操作。
2. 如果 Amazon Kendra 傳回回應，Amazon Lex V2 會向使用者顯示結果。
3. 如果沒有來自 Amazon Kendra 的回應，Amazon Lex V2 會重新提示用戶。下一個動作取決於來自使用者的回應。
   - 如果使用者的回應包含 Amazon Lex V2 可辨識的話語 (例如填入插槽值或確認意圖)，則與使用者的交談會按照機器人的設定繼續進行。
   - 如果使用者的回應不包含 Amazon Lex V2 可辨識的話語，Amazon Lex V2 會再次呼叫該作業。Query
4. 如果設定的重試次數後沒有回應，Amazon Lex V2 會呼叫AMAZON.FallbackIntent並結束與使用者的對話。

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

- 讓搜索意圖為您提出請求。Amazon Lex V2 調用 Amazon Kendra，並將用戶的話語作為搜索字符串。建立意圖時，您可以定義查詢篩選字串，以限制 Amazon Kendra 傳回的回應數量。Amazon Lex V2 會在查詢請求中使用篩選器。

- 使用 Lambda 函數將其他查詢參數新增至請求，以縮小搜尋結果範圍。您可以將包含 Amazon Kendra 查詢參數的kendraQueryFilterString欄位新增至delegate對話方塊動作。當您使用 Lambda 函數將查詢參數新增至請求時，它們的優先順序會高於您建立意圖時定義的查詢篩選器。
- 使用 Lambda 函數建立新的查詢。您可以建立 Amazon Lex V2 傳送的完整亞馬遜肯德拉查詢請求。您可以在 delegate 對話方塊動作的 kendraQueryRequestPayload 欄位中指定查詢。kendraQueryRequestPayload 欄位的優先順序高於 kendraQueryFilterString 欄位。

若要在建立機器人時指定queryFilterString參數，或在對話方塊 Lambda 函數中呼叫delegate動作時指定kendraQueryFilterString欄位，請指定用作 Amazon Kendra 查詢屬性篩選器的字串。如果字串不是有效的屬性篩選條件，則您會在執行時間取得 InvalidBotConfigException 例外狀況。如需有關屬性篩選器的詳細資訊，請參閱 Amazon Kendra 開發人員指南中的使用文件屬性篩選查詢。

若要控制 Amazon Lex V2 傳送至 Amazon Kendra 的查詢，您可以在 Lambda 函數的kendraQueryRequestPayload欄位中指定查詢。如果查詢無效，Amazon Lex V2 會傳回InvalidLambdaResponseException例外狀況。如需詳細資訊，請參閱 Amazon Kendra 開發人員指南中的查詢操作。

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

Amazon Kendra 搜索的 IAM 政策

若要使用AMAZON.KendraSearchIntent意圖，您必須使用提供 AWS Identity and Access Management (IAM) 政策的角色，讓 Amazon Lex V2 承擔具有呼叫 Amazon Kendra Query 意圖之權限的執行階段角色。您使用的 IAM 設定取決於您是AMAZON.KendraSearchIntent使用 Amazon Lex V2 主控台建立，還是使用 AWS 開發套件或 AWS Command Line Interface (AWS CLI) 來建立。使用主控台時，您可以選擇新增將 Amazon Kendra 呼叫到 Amazon Lex V2 服務連結角色的權限，或使用專門用於呼叫 Amazon Kendra Query 操作的角色。當您使用 AWS CLI 或 SDK 建立意圖時，您必須使用專門用於呼叫Query作業的角色。

連接許可

您可以使用主控台連接許可，將 Amazon Kendra Query 操作存取到預設的 Amazon Lex V2 服務連結角色。將許可附加到服務連結角色時，您不需要建立和管理專門用於連線至 Amazon Kendra 索引的執行時期角色。

您用來存取 Amazon Lex V2 主控台的使用者、角色或群組必須具有管理角色政策的許可。將下列 IAM 政策附加到主控台存取角色。當您授與這些許可時，角色具有變更現有的服務連結角色政策的許可。

```
{
"Version": "2012-10-17",
"Statement": [
    {
        "Effect": "Allow",
        "Action": [
            "iam:AttachRolePolicy",
            "iam:PutRolePolicy",
            "iam:GetRolePolicy"
        ],
        "Resource": "arn:aws:iam::*:role/aws-service-role/lexv2.amazonaws.com/
AWSServiceRoleForLexBots*"
    },
    {
        "Effect": "Allow",
        "Action": "iam:ListRoles",
        "Resource": "*"
    }
]
}
```

指定角色

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

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

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

Amazon Lex V2 需要用來呼叫 Amazon Kendra 的執行階段角色必須具有`kendra:Query`許可。當您使用現有的 IAM 角色獲得呼叫 Amazon Kendra `Query` 操作的許可時，該角色必須附加以下政策。

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

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

將查詢操作許可新增至角色

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

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

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

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

3. 將政策附加到您用來呼叫`Query`作業的 IAM 角色。

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

```
--role-name role-name
```

您可以選擇更新 Amazon Lex V2 服務連結角色，或使用為機器人建立時建立`AMAZON.KendraSearchIntent`的角色。下列程序顯示如何選擇要使用的 IAM 角色。

若要指定的執行階段角色 AMAZON.KendraSearchIntent

1. 登錄到 AWS Management Console 並打開 Amazon Lex 控制台 https://console.aws.amazon.com/lex/.
2. 選擇您要新增 `AMAZON.KendraSearchIntent` 的機器人。
3. 選擇 Intents (意圖) 旁邊的加號 (+)。
4. 在 Add intent (新增意圖) 中，選擇 Search existing intents (搜尋現有的意圖)。
5. 在 Search intents (搜尋意圖) 中，輸入 **`AMAZON.KendraSearchIntent`** 然後選擇 Add (新增)。
6. 在 Copy built-in intent (複製內建意圖) 中，輸入意圖的名稱，例如 **`KendraSearchIntent`**，然後選擇 Add (新增)。
7. 開啟 Amazon Kendra query (Amazon Kendra 查詢) 部分。
8. 在 IAM role (IAM 角色) 下，選擇以下其中一個選項：
   - 若要更新 Amazon Lex V2 服務連結角色，讓您的機器人能夠查詢 Amazon Kendra 索引，請選擇新增 Amazon Kendra 許可。
   - 若要使用具有呼叫 Amazon Kendra `Query` 作業之權限的角色，請選擇「使用現有角色」。

使用請求和工作階段屬性作為篩選條件

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

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

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

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

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

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

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

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

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

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

使用搜尋回應

Amazon Kendra 會從意圖的IntentClosingSetting陳述式回應中傳回搜尋的回應。除非 Lambda 函數產生結束回應訊息，否則意圖必須有closingResponse陳述式。

Amazon Kendra 有五種類型的回應。

- 下列兩個回應需要為您的 Amazon Kendra 索引設定常見問題集。如需詳細資訊，請參閱將問題和答案直接新增至索引。
  - `x-amz-lex:kendra-search-response-question_answer-question-<N>`— 來自與搜索匹配的常見問題解答的問題。
  - `x-amz-lex:kendra-search-response-question_answer-answer-<N>`— 來自與搜索匹配的常見問題解答的答案。
- 下列三個回應需要為 Amazon Kendra 索引設定資料來源。有關詳情，請參閱建立資料來源。
  - `x-amz-lex:kendra-search-response-document-<N>`— 摘錄自索引中的文件，該文件與語音文字相關。
  - `x-amz-lex:kendra-search-response-document-link-<N>`— 索引中與語音文字相關的文件 URL。
  - `x-amz-lex:kendra-search-response-answer-<N>`— 索引中文件摘錄，可回答問題。

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

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

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

使用 Lambda 函數來管理請求和回應

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

使用對話方塊程式碼掛勾建立查詢

您可以使用對話方塊程式碼掛接來建立要傳送至 Amazon Kendra 的查詢。使用對話方塊程式碼掛勾是選用的。如果您未指定對話方塊程式碼勾點，Amazon Lex V2 會根據使用者的話語建構查詢，並使用您在設定queryFilterString意圖時提供的查詢 (如果您提供的話)。

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

- kendraQueryFilterString— 使用此字串可指定 Amazon Kendra 請求的屬性篩選器。您可以使用索引中定義的任何索引欄位來篩選查詢。如需篩選字串的結構，請參閱 Amazon Kendra 開發人員指南中的使用文件屬性篩選查詢。如果指定的篩選字串無效，您會取得 InvalidLambdaResponseException 例外狀況。kendraQueryFilterString 字串會覆寫為意圖所設定的 queryFilterString 中指定的任何查詢字串。
- kendraQueryRequestPayload— 使用此字串可指定 Amazon Kendra 查詢。您的查詢可以使用 Amazon Kendra 的任何功能。如果您沒有指定有效的查詢，您會取得 InvalidLambdaResponseException 例外狀況。如需詳細資訊，請參閱 Amazon Kendra 開發人員指南中的查詢。

建立篩選器或查詢字串之後，您可以將回應傳送至 Amazon Lex V2，並將回應dialogAction欄位設定為delegate。Amazon Lex V2 會將查詢傳送至 Amazon Kendra，然後將查詢回應傳回至履行代碼勾點。

針對回應使用履行程式碼掛勾

Amazon Lex V2 將查詢傳送至 Amazon Kendra 之後，查詢回應就會傳回至AMAZON.KendraSearchIntent履行 Lambda 函數。代碼鉤子的輸入事件包含來自 Amazon Kendra 的完整響應。查詢資料的結構與 Amazon Kendra Query 作業傳回的資料結構相同。如需詳細資訊，請參閱 Amazon Kendra 開發人員指南中的查詢回應語法。

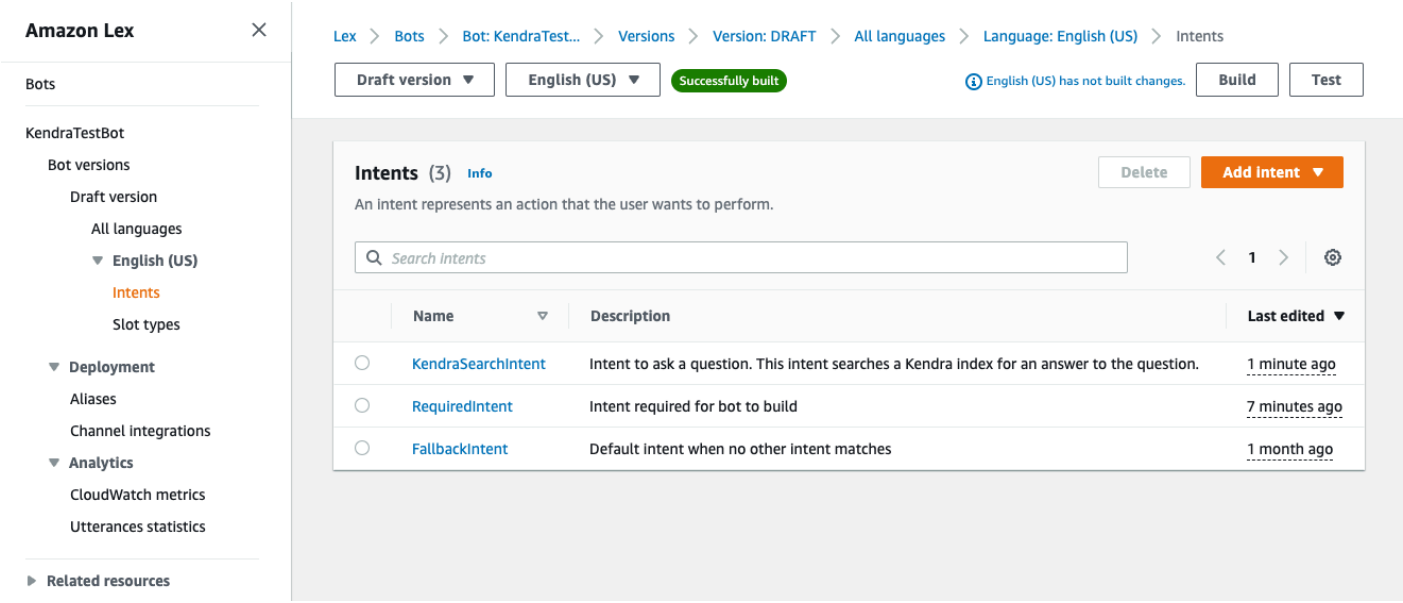
履行程式碼掛勾是選用的。如果不存在，或程式碼掛鉤未在回應中傳回訊息，Amazon Lex V2 會使用該closingResponse陳述式進行回應。

範例：為 Amazon Kendra 索引建立常見問題集機器人

此範例會建立 Amazon Lex V2 機器人，該機器人使用 Amazon Kendra 索引來提供使用者問題的答案。常見問題機器人會管理使用者的對話方塊。它使用 AMAZON.KendraSearchIntent 意圖來查詢索引，並向使用者呈現回應。以下是如何使用 Amazon Kendra 索引建立常見問題集機器人的摘要：

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。因為AMAZON.KendraSearchIntent和AMAZON.FallbackIntent是備份意圖，所以您的機器人至少需要一個其他意圖，且必須包含至少一個語音。此意圖讓您的機器人可以建置，但不會用於其他用途。因此，您的常見問題解答機器人將包含至少三個意圖，如下圖所示：



3. 將AMAZON.KendraSearchIntent意圖新增至您的機器人，並將其設定為與 Amazon Kendra 索引搭配使用。
4. 透過進行查詢來測試機器人，並驗證 Amazon Kendra 索引的結果是可回答查詢的文件。

先決條件

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 開發人員指南中的開始使用 Amazon Kendra 主控台。在此範例中，選擇範例資料集 (範例 AWS 文件) 做為您的資料來源。

若要建立常見問題集機器人：

1. 登錄到 AWS Management Console 並打開 Amazon Lex 控制台 https://console.aws.amazon.com/lex/.
2. 在導覽窗格中，選擇 Bots (機器人)。
3. 選擇建立機器人。
   a. 對於 [建立] 方法，選擇 [建立空白機器人]。
   b. 在「機器人設定」區段中，為機器人指定其用途的名稱，例如**KendraTestBot**，以及選擇性描述。該名稱在您的帳戶中必須是唯一的。

c. 在 IAM 許可區段中，選擇建立具有基本 Amazon Lex 許可的角色。這將建立具有 Amazon Lex V2 執行機器人所需許可的 AWS Identity and Access Management (IAM) 角色。

d. 在 [兒童線上隱私權保護法案 (COPPA)] 區段中，選擇 [否]。

e. 在 [閒置工作階段逾時] 和 [進階設定] 區段中，保留預設設定並選擇 [下一步]。

f. 現在，您已進入「向機器人添加語言」部分。在「語音互動」下方的選單中，選取「無」。這只是一個基於文本的應用程序。保留其餘欄位的預設設定。

g. 選擇完成。Amazon Lex V2 會建立您的機器人和呼叫的預設意圖 NewIntent，並帶您前往頁面以設定此意圖

若要成功建置機器人，您必須至少建立一個AMAZON.FallbackIntent與和分開的意圖AMAZON.KendraSearchIntent。建置 Amazon Lex V2 機器人需要此目的，但不適用於常見問答集回應。此意圖必須至少包含一個樣本語言，而言語不得適用於客戶提出的任何問題。

若要建立所需的意圖：

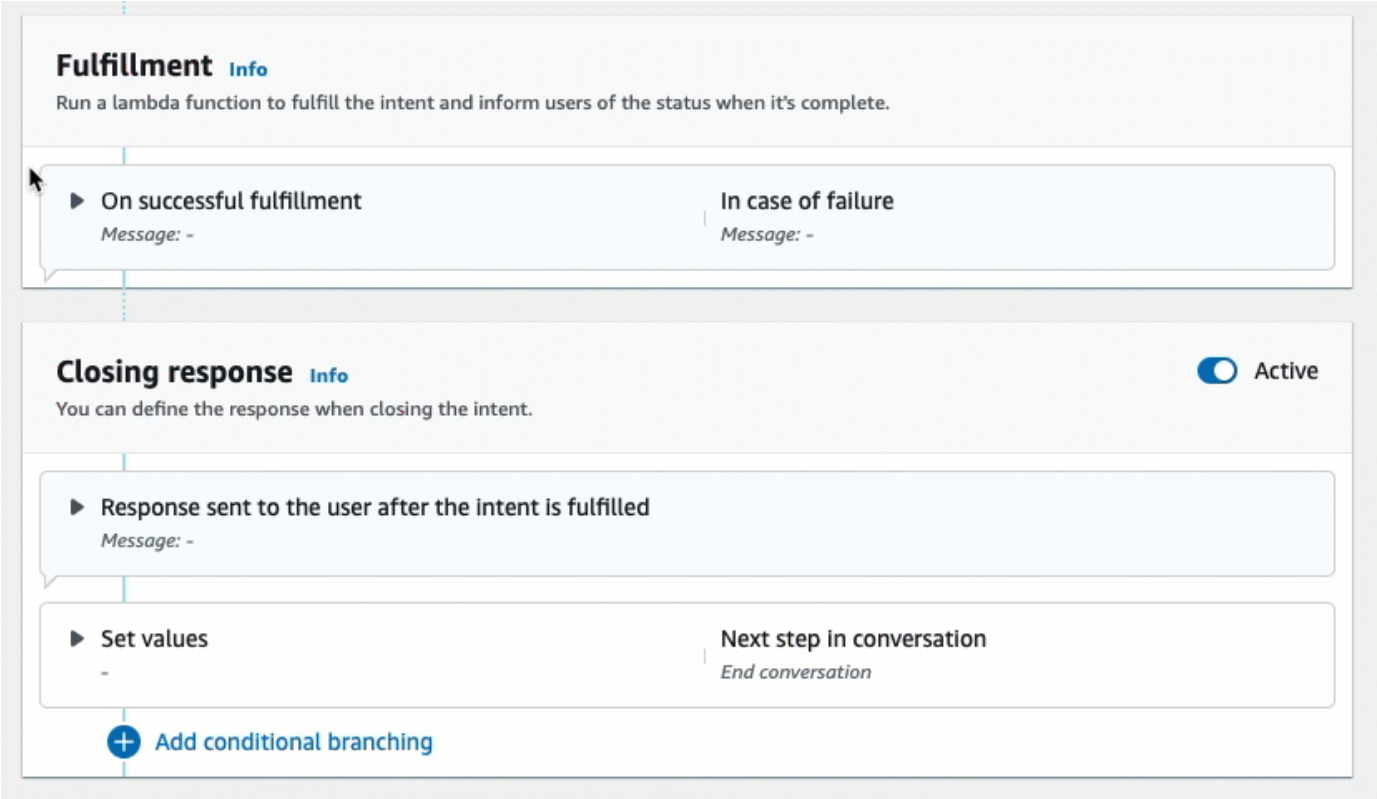
1. 在「意圖詳細資料」區段中，為意圖命名，例如**RequiredIntent**。
2. 在 [範例語音] 區段中，在 [新增語音] 旁邊的方塊中輸入語音，例如。**Required utterance**然後選擇添加語音。
3. 選擇儲存意圖。

建立意圖以搜尋 Amazon Kendra 索引及其應傳回的回應訊息。

要創建一個亞馬遜。 KendraSearchIntent 意圖和響應消息：

1. 在導覽窗格中選取 [返回意圖] 清單以返回機器人的 [意圖] 頁面。選擇 [新增意圖]，然後從下拉式選單中選取 [使用內建方式
2. 在彈出的框中，選擇「內置意圖」下的菜單。**AMAZON.KendraSearchIntent**在搜索欄中輸入，然後從列表中選擇它。
3. 為意圖命名，例如**KendraSearchIntent**。
4. 從 Amazon Kendra 索引下拉式功能表中，選擇您想要搜尋的索引。您在 [必要條件] 區段中建立的索引應該是可用的。
5. 選取新增。
6. 在意圖編輯器中，向下捲動至「出貨」區段，選取向右箭號以展開區段，並在「成功出貨時」下方的方塊中新增下列訊息：

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



如需 Amazon Kendra 搜尋回應的詳細資訊，請參閱使用搜尋回應。

7. 選擇 Save intent (儲存意圖)，然後選擇 Build (建置) 以建置機器人。機器人準備就緒時，畫面頂端的橫幅會變成綠色，並顯示成功訊息。

最後，使用主控台測試視窗來測試您的機器人的回應。

若要測試常見問題集機器人：

1. 成功建置機器人之後，請選擇 [測試]。
2. **`What is Amazon Kendra?`**在控制台測試窗口中輸入。驗證機器人是否使用連結回應。
3. 如需有關配置的詳細資訊`AMAZON.KendraSearchIntent`，請參閱AMAZON.KendraSearchIntent和KendraConfiguration。

AMAZON.PauseIntent

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

常見的話語：

- 暫停
- 暫停那個

AMAZON.QnAIntent

> **Note**
>
> 您必須符合下列先決條件，才能利用生成 AI 功能
>
> 1. 導覽至 Amazon 基岩主控台，然後註冊以存取您要使用的 Properpic Claude 模型 (如需詳細資訊，請參閱模型存取權)。如需使用 Amazon 基岩定價的相關資訊，請參閱 Amazon 基岩定價。
> 2. 為您的機器人地區設定開啟生成 AI 功能。若要這麼做，請依照中的步驟執行利用生成式 AI 最佳化機器人建立與效能。

通過使用 Amazon 基岩 FM 搜索和總結常見問題解答響應來響應客戶問題。當語音未分類到機器人中存在的任何其他意圖時，此意圖將被激活。請注意，當引出插槽值時，不會針對遺漏的話語啟動此意圖。一旦辨識出`AMAZON.QnAIntent`，就會使用指定的 Amazon 基岩模型來搜尋設定的知識庫，並回應客戶問題。

如果來自 FM 的響應不令人滿意或調用 FM 失敗，Amazon Lex V2 然後調用. `AMAZON.FallbackIntent`

> **Warning**
>
> 您無法在相同的機器人地區設定`AMAZON.KendraSearchIntent`中使用`AMAZON.QnAIntent`和。

以下是可用的知識庫選項。您必須已經建立知識庫，並編製索引其中的文件。

- OpenSearch 服務網域 — 包含索引文件。若要建立網域，請按照建立和管理 Amazon OpenSearch 服務網域中的步驟進行操作。
- Amazon Kendra 索引 — 包含索引的常見問題文件。若要建立 Amazon Kendra 索引，請依照建立索引中的步驟執行。
- Amazon 基岩知識庫 — 包含索引資料來源。若要設定知識庫，請依照建立知識庫中的步驟進行。

如果您選取此意圖，請設定下列欄位，然後選取「新增」以新增意圖。

- 基岩模型 — 選擇用於此目的的提供者和基礎模型。目前，人類克勞德 V2 和人為克勞德即時的支持。
- 知識庫 — 選擇您希望模型從中提取資訊以回答客戶問題的來源。以下是可用的來源。
  - OpenSearch— 配置以下字段。
    - 網域端點 — 提供您為網域建立的網域端點，或在網域建立後提供給您的網域端點。
    - 索引名稱 — 提供要搜尋的索引。如需詳細資訊，請參閱在 Amazon OpenSearch 服務中編製資料索引。
    - 選擇您要如何將回應傳回給客戶。
      - 確切回應 — 啟用此選項時，「回答」欄位中的值會依原樣使用機器人回應。已設定的 Amazon 基礎模型可用於按原樣選取確切的答案內容，無需任何內容合成或摘要。指定資料庫中設定的問題和答案欄位的名 OpenSearch 稱。
      - 包含欄位 — 傳回使用您指定欄位的模型產生的答案。指定資料庫中設定的最多五個欄位的名 OpenSearch 稱。使用分號 (;) 分隔欄位。
  - Amazon Kendra — 設定下列欄位。
    - Amazon Kendra 索引 — 選擇您希望機器人搜索的 Amazon Kendra 索引。
    - Amazon Kendra 篩選器 — 若要建立篩選器，請選取此核取方塊。如需 Amazon Kendra 搜尋篩選器 JSON 格式的詳細資訊，請參閱使用文件屬性篩選搜尋結果。
    - 確切回應 — 若要讓機器人傳回 Amazon Kendra 傳回的確切回應，請選取此核取方塊。否則，您選取的 Amazon 基岩模型會根據結果產生回應。

> **Note**
>
> 若要使用此功能，您必須先依照將常見問題集 (FAQ) 新增至索引中的步驟，將常見問題集問題新增至索引。

- Amazon 基岩知識庫 — 如果您選擇此選項，請指定知識庫的 ID。您可以在主控台中檢查知識庫的詳細資料頁面，或傳送GetKnowledgeBase要求來尋找 ID。

從 QnaIntent 的響應將被存儲到請求屬性，如下圖所示：

- `x-amz-lex:qnA-search-response`— 來自 QnaIntent 對問題或說話的回應。
- `x-amz-lex:qnA-search-response-source`— 指向用來產生回應的文件或文件清單。

AMAZON.RepeatIntent

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

常見的話語：

- 重複
- 再說一遍
- 重複這一點

AMAZON.ResumeIntent

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

常見的話語：

- 恢復
- 繼續
- 繼續, 去

AMAZON.StartOverIntent

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

常見的話語：

- 重新開始

- 重新開始
- 重新開始

AMAZON.StopIntent

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

常見的話語：

- stop
- off
- 閉嘴

新增插槽類型

插槽類型定義了使用者可以為您的意圖變數提供的值。您可以為每種語言定義插槽類型，以便這些值特定於該語言。例如，對於列出油漆顏色的槽類型，您可以包括英文值 red ""、法文中的 rouge "" 和西班牙文中的 rojo ""。

本主題說明如何建立自訂插槽類型，以提供意圖槽的值。您也可以針對標準值使用內建插槽類型。例如，您可以將內置插槽類型用`AMAZON.Country`於世界上的國家/地區列表。

建立槽類型

1. 登錄到AWS Management Console並打開 Amazon Lex 控制台 https://console.aws.amazon.com/lex/.
2. 從機器人清單中，選擇您要新增語言的機器人，然後選擇 [交談結構]，然後選擇 [所有語言]。
3. 選擇要新增插槽類型的語言，然後選擇插槽類型。
4. 選擇 [新增插槽類型]，為您的插槽類型命名，然後選擇 [新增]。
5. 在插槽類型編輯器中，新增插槽類型的詳細資料。

    - 槽值解析度 — 決定槽值的解析方式。如果您選擇「展開值」，Amazon Lex V2 會使用這些值做為訓練的代表值。如果您使用「限制為槽」值，則允許的插槽值會限制為您提供的值。
    - 槽類型值 — 槽的值。如果您選擇「限制為插槽值」，則可以為值新增同義字。例如，對於值「足球」，您可以添加同義詞「足球」。如果用戶在與您的機器人對話中輸入「足球」，則插槽的實際值是「足球」。

- 使用位置值作為自訂字彙 — 啟用此選項可協助改善音訊對話中插槽值和同義字的辨識度。當插槽值是常用術語時，請勿啟用此選項，例如「是」、「否」、「一」、「二」、「三」等。

6. 選擇 [儲存插槽類型]。

Amazon Lex V2 提供以下插槽類型：

主題

內建插槽類型

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

槽類型	簡短描述	支援的地區設定	
亞馬遜。 AlphaNumeric	辨識由字母和數字組成的字詞。	除了韓文 (Ko-Kr) 以外的所有語言環境	
亞馬遜城	識別代表城市的單詞。	所有語言環境	
亞馬遜。確認	識別意味著「是」，「否」，「也許」和「不知道」的單詞，並將其轉換為標準 (是/否/可能/不知道）格式。	英語 (英-美國, EN-GB, 一-歐, 恩-ZA)	
亞馬遜國家	識別代表一個國家的單詞。	所有語言環境	

槽類型	簡短描述	支援的地區設定
亞馬遜。日期	識別代表日期的單詞並將其轉換為標準格式。	所有語言環境
亞馬遜持續時間	識別代表持續時間的單詞並將其轉換為標準格式。	所有語言環境
亞馬遜。 EmailAddress	識別代表電子郵件地址的單詞，並將其轉換為標準電子郵件地址。	所有語言環境
亞馬遜。 FirstName	識別代表名字的單詞。	所有語言環境
亞馬遜。 LastName	識別代表姓氏的單詞。	所有語言環境
亞馬遜。數	識別數字單詞並將其轉換為數字。	所有語言環境
AMAZON.Percentage	辨識代表百分比的字詞，並將其轉換為數字和百分比符號 (%)。	所有語言環境
亞馬遜。 PhoneNumber	識別代表電話號碼的單詞並將其轉換為數字字符串。	所有語言環境
亚马逊州	識別代表狀態的單詞。	所有語言環境
亞馬遜。 StreetName	識別代表街道名稱的單詞。	所有語言環境

槽類型	簡短描述	支援的地區設定
亞馬遜時間	識別指示時間的單詞並將其轉換為時間格式。	所有語言環境
亞馬遜 PostalCode	識別代表英國郵政編碼的單詞，並將其轉換為標準形式。	僅限英文 (英文) (en-GB)
亞馬遜。 FreeFormInput	識別由任何單詞或字符組成的字符串。	所有語言環境

亞馬遜。 AlphaNumeric

辨識由字母和數字組成的字串，例如 **APQ123**。

此插槽類型在韓文 (K-KR) 語言環境中不可用。

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

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

AMAZON.AlphaNumeric插槽類型支援使用拼字樣式的輸入。您可以使用 spell-by-letter 和 spell-by-word 樣式來協助您的客戶輸入字母。如需詳細資訊，請參閱 使用拼字樣式擷取位置值。

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：

- 加拿大郵政編碼
- 駕照號碼
- 車輛識別碼

使用標準規則運算式。Amazon Lex V2 在規則運算式中支援下列字元：

- A-Z、a-z

- 0-9

Amazon Lex V2 也支持正則表達式中的 Unicode 字符。格式為 \u*Unicode*。使用四位數代表 Unicode 字元。例如，[\u0041-\u005A] 等同於 [A-Z]。

不支援下列規則運算式：

- 無限的重複項：*、+ 或 {x,}，沒有上限。
- 萬用字元 (.)

規則運算式的最大長度為 300 個字元。儲存在使用規則運算式的 AMAZON.AlphaNumeric slot 類型中的字串長度上限為 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 CreateSlotType 操作。當您儲存槽類型時，會驗證規則表達式。如果運算式無效，Amazon Lex V2 會傳回錯誤訊息。

當您在插槽類型中使用規則運算式時，Amazon Lex V2 會根據規則運算式檢查對該類型插槽的輸入。如果輸入與表達式相符，則會針對該槽接受值。如果輸入不相符，Amazon Lex V2 會提示使用者重複輸入。

亞馬遜城

提供本地和世界城市的列表。插槽類型可識別城市名稱的常見變化。Amazon Lex V2 不會從變體轉換為正式名稱。

範例：

- 紐約
- 雷克雅
- 東京
- 凡爾賽宮

亞馬遜。確認

此插槽類型可識別對應於 Amazon Lex V2 的「是」、「否」、「可能」和「不知道」片語和字詞的輸入詞組，並將其轉換為四個值之一。它可以用來捕獲來自用戶的確認或確認。根據最終解析值，您可以建立條件來設計多個交談路徑。

例如：

如果 {確認} =「是」，履行意圖

否則，引出另一個插槽

範例：

- 是的：是的，是的，好的，當然，我有它，我可以同意...
- 否：不，否定，Naw，算了，我會拒絕，沒辦法...
- 也許:這是可能的, 也許, 有時, 我可能, 這可能是正確的...
- 不知道：不知道，未知，不知道，不確定，誰知道...

自 2023 年 8 月 17 日起，如果有名為「確認」的現有自訂插槽類型，則必須變更名稱，以避免與內建插槽確認衝突。在 Lex 主控台的左側導覽中，移至插槽類型 (針對名為「確認」的現有自訂插槽類型)，然後更新插槽類型名稱。新插槽類型名稱不得為「確認」，這是內建確認插槽類型的保留關鍵字。

亞馬遜國家

世界各地國家的名稱。範例：

- 澳洲
- 德國
- 日本
- 美國
- 烏拉圭

亞馬遜。日期

將代表日期的單字轉換為日期格式。

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

- 映射到特定日期的語音，例如「今天」，「現在」或「11 月 25 日」，轉換為完整日期：。2020-11-25這預設為當前日期或之後的日期。
- 映射到 future 一周的話語（例如「下週」）會轉換為當週最後一天的日期。在 ISO-8601 格式中，一周從星期一開始，星期日結束。例如，如果今天是 2020-11-25，則「下週」將轉換為. 2020-11-29對映至目前或上週的日期會轉換為一週的第一天。例如，如果今天是 2020-11-25，則「上週」將轉換為. 2020-11-16
- 對應至 future 月份但不是特定日期的話語，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」將轉換為. 2020-12-31 對於映射到當前月份或上個月的日期，轉換為該月的第一天。例如，如果今天是 2020-11-25，則「本月」將映射到. 2020-11-01
- 對應至 future 年份 (但不是特定月份或日期) 的話語 (例如「明年」) 會轉換為次年的最後一天。例如，如果今天是 2020-11-25，則「明年」將轉換為. 2021-12-31 對於映射到當前年度或上一年的日期，將轉換為年份的第一天。例如，如果今天是 2020-11-25，則「去年」轉換為. 2019-01-01

亞馬遜持續時間

將指示持續時間的單字轉換為數字持續時間。

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

範例：

- 「十分鐘」：PT10M
- 「五個小時」：PT5H
- 「三天」：P3D
- 「四十五秒」：PT45S
- 「八個星期」：P8W
- 「七年」：P7Y
- 「五小時十分鐘」：PT5H10M
- 「二年三小時十分鐘」：P2YT3H10M

亞馬遜。 EmailAddress

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

AMAZON.EmailAddress插槽類型支援使用拼字樣式的輸入。您可以使用 spell-by-letter 和 spell-by-word 樣式來協助客戶輸入電子郵件地址。如需詳細資訊，請參閱 使用拼字樣式擷取位置值。

亞馬遜。 FirstName

常用的名字。此插槽類型可識別正式名稱、非正式暱稱以及由多個字組成的名稱。發送到您的意圖的名稱是用戶發送的值。Amazon Lex V2 不會從暱稱轉換為正式名稱。

對於聽起來相似但拼寫不同的名字，Amazon Lex V2 會將您的意圖傳送為單一通用表單。

AMAZON.FirstName插槽類型支援使用拼字樣式的輸入。您可以使用 spell-by-letter 和 spell-by-word 樣式來協助您的客戶輸入名稱。如需詳細資訊，請參閱 使用拼字樣式擷取位置值。

範例：

- 艾米莉
- John
- 蘇菲
- 阿尼尔·库马尔

亞馬遜。 FirstName 還返回基於原始值密切相關的名稱列表。您可以使用已解析值清單從錯別字中復原、與使用者確認名稱，或執行資料庫查詢使用者目錄中的有效名稱。

例如，輸入「John」可能會導致返回其他相關名稱，例如「約翰 J」和「約翰·保羅」。

下面顯示了AMAZON.FirstName內置插槽類型的響應格式：

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

```
}
```

亞馬遜。 LastName

常用的姓氏。對於聽起來相似且拼寫不同的名稱，Amazon Lex V2 會將您的意圖傳送為單一通用表單。

AMAZON.LastName插槽類型支援使用拼字樣式的輸入。您可以使用 spell-by-letter 和 spell-by-word 樣式來協助您的客戶輸入名稱。如需詳細資訊，請參閱 使用拼字樣式擷取位置值。

範例：

- 布羅斯基
- 大傻瓜
- 埃弗斯
- 帕雷斯
- 邊痕

亞馬遜。 LastName 還返回基於原始值密切相關的名稱列表。您可以使用已解析值清單從錯別字中復原、與使用者確認名稱，或執行資料庫查詢使用者目錄中的有效名稱。

例如，輸入「史密斯」可能會導致返回其他相關名稱，例如「史密斯」和「史密斯」。

下面顯示了AMAZON.LastName內置插槽類型的響應格式：

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

亞馬遜。數

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

Input	回應
一百二十三點四五	123.45
一百二十三點四五	123.45
點四二	0.42
點四十二	0.42
232.998	232.998
50	50
-15	-15
減去	-15

AMAZON.Percentage

將代表百分比的字詞和符號轉換成包含百分比符號 (%) 的數值。

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

Input	回應
50 百分比	50%
0.4% 百分比	0.4%
23.5%	23.5%
二十五, 百分之	25%

亞馬遜。 PhoneNumber

將代表電話號碼的數字或字詞轉換成不含標點符號的字串格式，如下所示。

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

亚马逊州

國家內的地理和政治區域的名稱。

範例：

- 巴伐利亞
- 福島縣
- 西北太平洋
- 昆士蘭
- 威爾士

亞馬遜。 StreetName

典型街道地址內的街道名稱。這只包括街道名稱，而不是門牌號碼。

範例：

- 堪培拉大道
- 前街
- 市場路

亞馬遜時間

將代表時間的單詞轉換為時間值。 AMAZON.Time可以解決確切的時間，不明確的值和時間範圍。插槽值可以解析為下列時間範圍：

- AM
- PM
- 密蘇里州（上午）
- 自動對焦 (下午)
- 電動車（夜間）
- 倪 (夜間)

當使用者輸入不明確的時間時，Amazon Lex V2 會使用 Lambda 事件的slots屬性，將不明確時間的解決方案傳遞給您的 Lambda 函數。例如，如果您的機器人提示使用者交付時間，使用者可以說「10 點鐘」來回應。但是這個時間並不明確，這可表示早上 10 點或下午 10 點。在此情況下，interpretedValue欄位中的值為null，且resolvedValues欄位包含兩種可能的時間解析度。Amazon Lex V2 將以下內容輸入到 Lambda 函數中：

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

當使用者以明確的時間回應時，Amazon Lex V2 會將時間傳送至 Lambda 事件slots屬性interpretedValue欄位中的 Lambda 函數。例如，如果您的使用者以「上午 10:00」回應交付時間的提示，Amazon Lex V2 會在 Lambda 函數中輸入下列內容：

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

```
            "10:00"
        ]
    }
}
```

當使用者以「早上」回應交付時間的提示時，Amazon Lex V2 會在 Lambda 函數中輸入下列內容：

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

如需從 Amazon Lex V2 傳送至 Lambda 函數之資料的詳細資訊，請參閱解譯輸入事件格式。

亞馬遜 PostalCode

將代表英國郵遞區號的文字轉換為英國郵遞區號的標準格式。AMAZON.UKPostalCode插槽類型會驗證郵遞區號並將其解析為一組標準化格式，但不會檢查以確定郵遞區號是否有效。您的應用程式必須驗證郵遞區號。

AMAZON.UKPostalCode插槽類型僅適用於英文 (英國) (en-GB) 地區設定。

AMAZON.UKPostalCode插槽類型支援使用拼字樣式的輸入。您可以使用 spell-by-letter 和 spell-by-word 樣式來協助您的客戶輸入字母。如需詳細資訊，請參閱 使用拼字樣式擷取位置值。

插槽類型僅識別下面列出的有效郵遞區號格式，在英國使用。有效的格式是（「A」表示一個字母，「9」代表一個數字）：

- AA9AA
- A9AA
- A9
- A99
- A9
- AA99

對於文本輸入，用戶可以輸入大寫和小寫字母的任何混合。使用者可以使用或省略郵遞區號中的空格。解析的值將始終包含郵遞區號適當位置的空格。

對於語音輸入，用戶可以說出單個字符，也可以使用雙字母發音，例如「double A」或「double 9」。它們也可以使用雙位數發音，例如「99」的「九十九」。

> **Note**
>
> 並非所有英國郵政編碼都被識別。僅支援上面列出的格式。

亞馬遜。 FreeFormInput

AMAZON.FreeFormInput可用於捕獲來自最終用戶的自由格式輸入。它識別由單詞或字符組成的字符串。解析的值是整個輸入語調。

範例：

機器人：請根據您的通話體驗提供反饋。

用戶：我得到了所有問題的答案，我能夠完成交易。

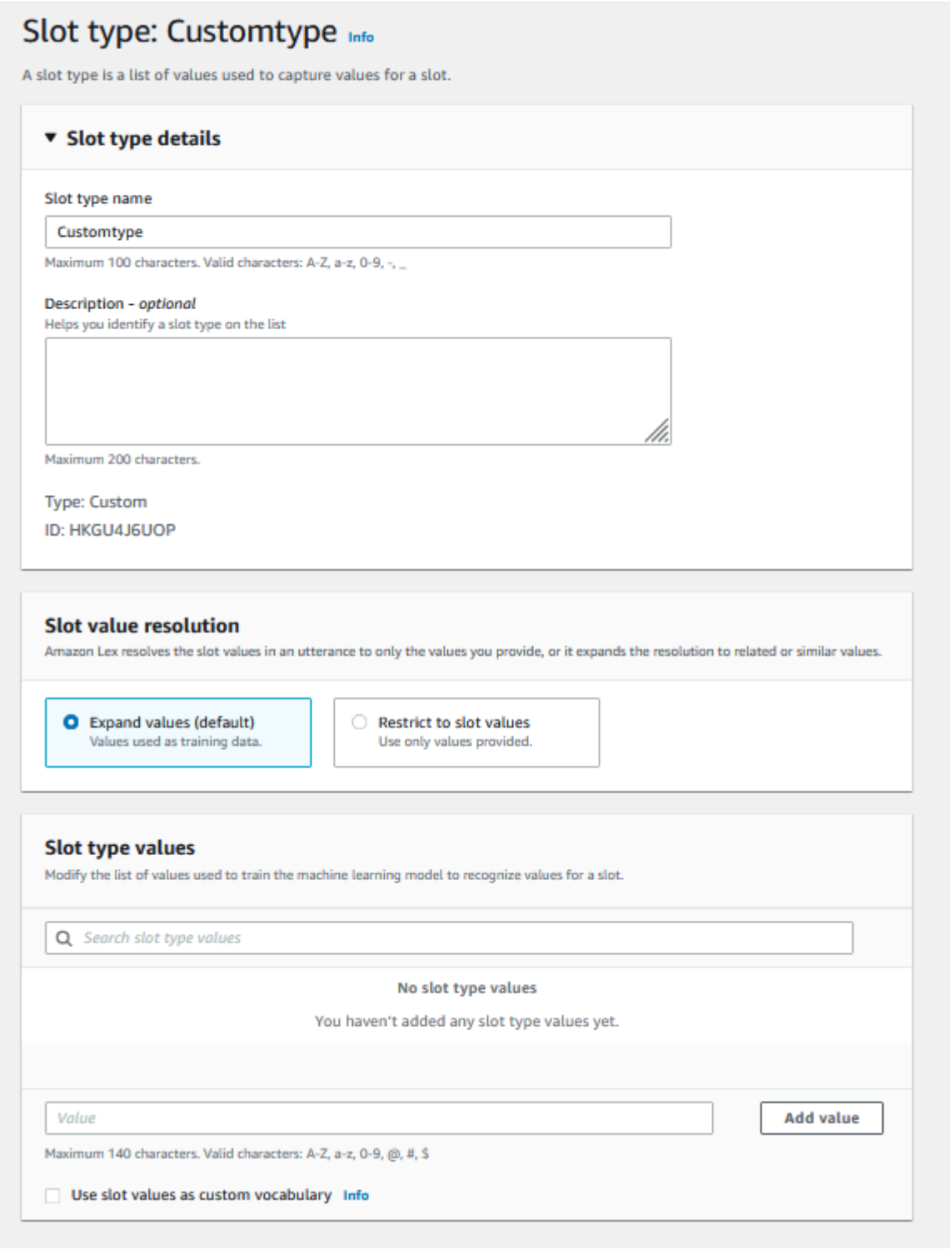
請注意：

- AMAZON.FreeFormInput可用於從最終用戶那裡捕獲自由格式輸入。
- AMAZON.FreeFormInput不能用於意圖樣本語言。
- AMAZON.FreeFormInput不能有插槽樣本語音。
- AMAZON.FreeFormInput只有在引發時才會被識別。
- AMAZON.FreeFormInput不支持等待和繼續。
- AMAZON.FreeFormInputAmazon Connect 聊天頻道目前不支持。
- 當一個AMAZON.FreeFormInput插槽被引發，FallbackIntent將不會被觸發。
- 當AMAZON.FreeFormInput插槽被引發時，將不會有意圖切換。

自訂插槽類型

對於每個意圖，您可以指定參數，指出意圖需要滿足使用者的請求的資訊。這些參數或槽，有一個類型。插槽類型是 Amazon Lex V2 用來訓練機器學習模型以辨識插槽值的值清單。例如，您可以定義名

為 Genres「喜劇」，「冒險」，「紀錄片」等值的插槽類型。您可以定義插槽類型值的同義詞。例如，您可以為值「喜劇」定義同義詞「滑稽」和「幽默」。



您可以配置槽類型以展開槽值。插槽值將用作訓練資料，且如果槽值與這些值的同義字相似，則模型會將槽解析為使用者提供的值。這是預設行為。Amazon Lex V2 會維護插槽的可能解析度清單。清單中

的每個項目都提供一個已解決的值，Amazon Lex V2 將其識別為插槽的其他可能性。解析值是符合槽值的最佳方法。該清單最多可包含五個值。

或者，您可以配置插槽類型，將解析度限制為槽值。在這種情況下，只有當模型與插槽值相同或是同義字時，才會將使用者輸入的槽值解析為現有槽值。例如，如果使用者輸入「滑稽」，它會解析為槽值「喜劇」。

當使用者輸入的值是槽類型值的同義詞時，模型會傳回該槽類型值作為清單中的第一個項目。`resolvedValues`例如，如果使用者輸入「搞笑」，則模型會在`originalValue`欄位中填入值「滑稽」，而在「已解決的 Values」欄位中的第一個項目填入「喜劇」。您可以在使用 `valueSelectionStrategy` 操作建立或更新槽類型時設定 CreateSlotType，如此一來槽值就會以解析清單中的第一個值填滿。

自訂插槽類型支援使用拼字樣式的輸入。您可以使用 spell-by-letter 和 spell-by-word 樣式來協助客戶輸入字母。如需詳細資訊，請參閱 使用拼字樣式擷取位置值。

如果您使用的是 Lambda 函數，則函數的輸入事件會包含一個名為的解析度清單`resolvedValues`。下列範例顯示輸入至 Lambda 函數的插槽區段：

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

自訂插槽類型的名稱不應與內建插槽類型相同。例如，自訂插槽類型不應以「日期」、「編號」或「確認」的保留關鍵字命名。這些關鍵字保留給內置插槽類型。如需所有內建插槽類型的清單，請參閱內建插槽類型。

語法插槽類型

使用語法插槽類型，您可以根據 SRGS 規範以 XML 格式編寫自己的語法，以便在對話中收集資訊。Amazon Lex V2 可辨識符合文法中指定規則的語音。您也可以在語法檔案中使用 ECMAScript 標籤來提供語意解譯規則。然後，Amazon Lex 會在比對發生時將標籤中設定的屬性傳回為已解析值。

您只能在英文 (澳洲)、英文 (英國) 和英文 (美國) 地區設定中建立文法位置類型。

語法插槽類型有兩個部分。第一個是使用 SRGS 規範格式編寫的語法本身。語法解釋用戶的話語。如果語音被語法接受，它被匹配，否則它被拒絕。如果一個話語匹配，它將傳遞給腳本（如果有的話）。

第二個是語法插槽類型的一部分是寫在 ECMAScript 中的可選腳本，該腳本將輸入轉換為槽類型返回的解析值。例如，您可以使用指令碼將語音數字轉換為數字。<tag>ECMAScript 語句被封閉在元素中。

下列範例是依據 SRGS 規格所採用的 XML 格式，其中顯示 Amazon Lex V2 接受的有效文法。它定義了接受卡號的語法插槽類型，並確定它們是否適用於常規或高級帳戶。如需可接受語法的詳細資訊，請參閱語法定義和主腳本格式題。

```
<grammar version="1.0" xmlns="http://www.w3.org/2001/06/grammar"
         xml:lang="en-US" tag-format="semantics/1.0" root="card_number">

   <rule id="card_number" scope="public">
        <item repeat="0-1">
            card number
        </item>
        <item>
            seven
            <tag>out.value = "7";</tag>
         </item>
         <item>
            <one-of>
               <item>
                  two four one
                  <tag> out.value = out.value + "241"; out.card_type = "premium"; </
tag>
               </item>
               <item>
                  zero zero one
                  <tag> out.value = out.value + "001"; out.card_type = "regular";</tag>
               </item>
            </one-of>
         </item>
   </rule>
```

```
</grammar>
```

上述語法只接受兩種類型的卡號：7241 或 7001。這兩者都可以選擇加上「卡號」前綴。它還包含可用於語義解釋的 ECMAScript 標籤。通過語義解釋，話語「卡號七二四一」將返回以下對象：

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

此物件會傳回為RecognizeText、RecognizeUtterance和作業傳回之resolvedValues物件中的 JSON 序列化字串。StartConversation

添加語法插槽類型

若要新增文法插槽類型

1. 將插槽類型的 XML 定義上傳到 S3 儲存貯體。記下值區名稱和檔案的路徑。

   > **Note**
   >
   > 檔案大小上限為 100 KB。

2. 登錄到AWS Management Console並打開亞馬遜萊克斯控制台 https://console.aws.amazon.com/lex/.
3. 從左側選單中選擇機器人，然後選擇要新增語法插槽類型的機器人。
4. 選擇 [檢視語言]，然後選擇要新增文法槽類型的語言。
5. 選擇檢視插槽類型。
6. 選擇 [新增插槽類型]，然後選擇 [新增語法插槽類型]。
7. 為插槽類型指定名稱，然後選擇 [新增]。
8. 選擇包含定義檔案的 S3 儲存貯體，然後輸入檔案的路徑。選擇 [儲存插槽類型]。

語法定義

本主題顯示 Amazon Lex V2 支援的 SRGS 規格各個部分。所有的規則都在 SRGS 規範中定義。如需詳細資訊，請參閱語音辨識文法規格 1.0 版 W3C 建議。

主題

- 標頭聲明
- 支援的 XML 元素
- 代幣
- 規則參考
- 序列和封裝
- 重複
- 語言
- Tags (標籤)
- 重量

本文件包含複製和衍生自 W3C 語音辨識語法規格 1.0 版 (可在 https://www.w3.org/TR/speech-grammar/ 取得) 的資料。引用文獻資訊如下：

SRGS 規格文件是 W3C 推薦書，可根據以下許可證從 W3C 獲得。

授權文字

授權

通過使用和/或複製本文檔，或與本聲明鏈接的 W3C 文檔，您（被授權人）同意您已閱讀，理解並將遵守以下條款和條件：

只要您在您使用的文件的所有副本或其部分上包括以下內容，就任何目的而在任何媒介中複製和分發本文件或本聲明所鏈接的 W3C 文檔的許可：

- 指向原始 W3C 文檔的鏈接或 URL。
- 原始作者的預先存在的版權聲明，或者如果它不存在，則通知（超文本是首選，但允許文本表示）的形式：「版權所有 © [$date-of-document] 萬維網聯盟，（麻省理工學院，ERCIM，京王，北航）。http://www.w3.org/Consortium/Legal/2015/doc-license」
- 如果它存在，W3C 文檔的狀態。

當空間允許時，應提供本通知的全文。我們要求您根據本文檔內容或其任何部分的實施而創建的任何軟件，文檔或其他項目或產品中提供作者身份歸屬。

根據本許可證，任何人都無權根據本許可證授予創建 W3C 文檔的修改或衍生物，除非如下：為了促進本文檔中規定的技術規範的實施，任何人都可以在軟件，附帶軟件的輔助材料以及軟件文檔中準備和分發本文檔的衍生作品和本文檔的一部分，前提是所有這些作品都包括以下通知。但是，明確禁止出版本文件的衍生作品以用作技術規範。

此外，「代碼組件」-清楚標記為 Web IDL 的部分中的 Web IDL; 和 W3C 定義的標記（HTML，CSS 等）和計算機編程語言代碼，明確標記為代碼示例-根據 W3C 軟件許可證進行許可。

該通知是：

「版權所有 © 2015 W3C®（麻省理工學院，埃爾西姆，京王，北航）。本軟件或文檔包括從 [W3C 文檔的標題和 URI] 複製或派生的材料。」

不承諾

本文件係依「原狀」提供，著作權持有人不作任何明示或暗示的陳述或保證，包括但不限於適銷性、特定用途適用性、不侵權或所有權之擔保；文件內容適用於任何目的，亦不會侵犯任何第三方專利、著作權、商標或其他權利。

版權持有人對於因使用文件或其內容的執行或實施而產生的任何直接、間接、特殊或相應而生的損害概不負責。

未經特定的事先書面許可，版權所有者的名稱和商標不得用於與本文件或其內容有關的廣告或宣傳。本文件中的版權所有權在任何時候均屬著作權持有人所有。

標頭聲明

下表顯示了語法插槽類型支持的頭聲明。如需詳細資訊，請參閱語音辨識文法規格第 1 版 W3C 建議中的文法標頭宣告。

宣告	規格要求	XML 格式	亞馬遜 Lex 支持	規格
文法版本	必要	4.3：`grammar`元素上的`version`屬性	必要	SRG
XML 命名空間	必要 (僅限 XML)	4.3：`grammar`元素上的`xmlns`屬性	必要	SRG
文件類型	必要 (僅限 XML)	4.3：文檔類型	建議	SRG

宣告	規格要求	XML 格式	亞馬遜 Lex 支持	規格
字元編碼	建議	4.4：XML 聲明中的encoding屬性	建議	SRG
語言	在語音模式下需要 在 DTMF 模式中忽略	4.5：grammar元素上的xml:lang屬性	在語音模式下需要 在 DTMF 模式中忽略	SRG
模式	選用	4.6：grammar元素上的mode屬性	選用	SRG
根規則	選用	4.7：grammar元素上的root屬性	必要	SRG
標籤格式	選用	4.8：grammar元素上的tag-format 屬性	支持字符串文字和 ECMAScript	SRG, 鋼鐵
基本 URI	選用	4.9：grammar元素上的xml:base屬性	選用	SRG
發音詞典	可選，允許多個	元素 lexicon	不支援	SRGS，請
中繼資料	可選，允許多個	元素 meta	必要	SRG
中繼資料	選擇性的，僅限 XML	元素 metadata	選用	SRG
Tag	可選，允許多個	元素 tag	不支援全域標籤	SRG

範例

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE grammar PUBLIC "-//W3C//DTD GRAMMAR 1.0//EN"
                  "http://www.w3.org/TR/speech-grammar/grammar.dtd">

<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xml:base="http://www.example.com/base-file-path"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US"
         version="1.0"
         mode="voice"
         root="city"
         tag-format="semantics/1.0">
```

支援的 XML 元素

亞馬遜 Lex V2 支持自定義語法的以下 XML 元素：

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

代幣

下表顯示語法插槽類型支援的記號規格。如需詳細資訊，請參閱語音辨識文法規格第 1 版 W3C 建議中的記號。

權杖類型	範例	支援？
單個非引號令牌	你好	是
單個未加引號的令牌：非字母	2	是
單引號令牌，沒有空格	"hello"	是的，當它只包含一個令牌時，刪除雙引號

權杖類型	範例	支援？
由空格分隔的兩個令牌	平安航行	是
由空格分隔的四個令牌	這是一個測試	是
單引號令牌，包括空格	「舊金山	否
標籤中的單一 XML <token>標記	<token>舊金山</token>	否（與帶空格的單引號令牌相同）

備註

- 包括空格的單引號令牌-該規範要求用雙引號括起來的單詞被視為單個標記。Amazon Lex V2 會將它們視為空格分隔的權杖。
- 中的單一 XML 記號 <token>— 規格要求以分隔的文字才<token>能代表一個記號。Amazon Lex V2 會將它們視為空格分隔的權杖。
- Amazon Lex V2 會在您的文法中找到任一使用情況時擲回驗證錯誤。

範例

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

規則參考

下表摘要說明文法文件中可能出現的各種規則參考形式。如需詳細資訊，請參閱語音辨識文法規格第 1 版 W3C 建議中的規則參考。

參考類型	XML 格式	支援
2.2.1 明確的本機規則參照	`<ruleref uri="#rulename"/>`	是

參考類型	XML 格式	支援
2.2.2 明確引用由 URI 標識的語法的命名規則	`<ruleref uri="grammarURI#rulename"/>`	否
2.2.2 隱含參考 URI 所識別之文法根規則	`<ruleref uri="grammarURI"/>`	否
2.2.2 明確引用由具有媒體類型的 URI 標識的語法的命名規則	`<ruleref uri="grammarURI#rulename" type="media-type"/>`	否
2.2.2 隱含參考由具有媒體類型的 URI 所識別的文法根規則	`<ruleref uri="grammarURI" type="media-type"/>`	否

2.2.3 特殊規則定義	`<ruleref special="NULL"/>`
	`<ruleref special="VOID"/>`
	`<ruleref special="GARBAGE"/>`
		否

備註

1. 語法 URI 是一個外部 URI。例如：`http://grammar.example.com/world-cities.grxml`。
2. 媒體類型可以是：
   - `application/srgs+xml`
   - `text/plain`

範例

```
<rule id="city" scope="public">
    <one-of>
        <item>Boston</item>
        <item>Philadelphia</item>
```

```
        <item>Fargo</item>
    </one-of>
</rule>

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

<!-- "Boston MA" -> city = Boston, state = MA -->
<rule id="city_state" scope="public">
    <ruleref uri="#city"/> <ruleref uri="#state"/>
</rule>
```

序列和封裝

下列範例顯示支援的序列。如需詳細資訊，請參閱語音辨識文法規格第 1 版 W3C 建議中的序列和封裝。

範例

```
<!-- sequence of tokens -->
this is a test

<!--sequence of rule references-->
<ruleref uri="#action"/> <ruleref uri="#object"/>

<!--sequence of tokens and rule references-->
the <ruleref uri="#object"/> is <ruleref uri="#color"/>

<!-- sequence container -->
<item>fly to <ruleref uri="#city"/> </item>
```

重複

下表顯示規則支援的重複展開。如需詳細資訊，請參閱語音辨識文法規格第 1 版 W3C 建議中的重複項目。

XML 格式 範例	Behavior (行為)	支援？
重複 = "N」 重複 = "6」	包含的表達式正好重複「n」次。「n」必須是「0」或正整數。	是
重複 = "米-N」 重複 =「4-6」	包含的膨脹在「m」和「n」次（含）之間重複。「m」和「n」必須同時為「0」或正整數，而「m」必須小於或等於「n」。	是
重複 = "m-」 重複 =「3-」	所包含的膨脹重複「m」次或更多（含）。「m」必須是「0」或正整數。例如，「3-」聲明包含的擴展可以出現三次，四次，五次或更多次。	是
重複 = "0-1」	包含的擴展是可選的。	是
<item repeat="2-4" repeat-prob="0.8">		否

語言

下面的討論適用於應用於語法的語言標識符。如需詳細資訊，請參閱語音辨識文法規格第 1 版 W3C 建議中的語言。

默認情況下，語法是一個單一的語言文檔，其中包含語言標識符在語法標題中的語言聲明中提供。除非另有聲明，否則該語法中的所有標記都將根據語法的語言進行處理。不支援文法層級語言宣告。

於下列範例中：

1. 亞馬遜萊克斯 V2 支援語言「en-US」的語法標頭宣告。
2. 不支援項目層級語言附件 (以##反白顯示)。如果語言附件與標頭宣告不同，Amazon Lex V2 就會擲回驗證錯誤。

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE grammar PUBLIC "-//W3C//DTD GRAMMAR 1.0//EN"
                  "http://www.w3.org/TR/speech-grammar/grammar.dtd">

<!-- the default grammar language is US English -->
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0">

  <!--
     single language attachment to tokens
     "yes" inherits US English language
     "oui" is Canadian French language
  -->
  <rule id="yes">
    <one-of>
      <item>yes</item>
      <item xml:lang="fr-CA">oui</item>
    </one-of>
  </rule>

  <!-- Single language attachment to an expansion -->
  <rule id="people1">
    <one-of xml:lang="fr-CA">
      <item>Michel Tremblay</item>
      <item>André Roy</item>
    </one-of>
  </rule>
</grammar>
```

Tags (標籤)

以下討論適用於為文法定義的標籤。如需詳細資訊，請參閱語音辨識文法規格第 1 版 W3C 建議中的標籤。

根據 SRGS 規範，標籤可以用下列方式定義：

1. 作為標頭聲明的一部分，如中所述標頭聲明。
2. 作為<rule>定義的一部分。

支援下列標籤格式：

- semantics/1.0(SSR, 電子印刷稿)
- semantics/1.0-literals（SISR 字符串文字）

不支援下列標籤格式：

- swi-semantics/1.0（細微差別專有）

範例

```
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xml:base="http://www.example.com/base-file-path"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US"
         version="1.0"
         mode="voice"
         root="city"
         tag-format="semantics/1.0-literals">
    <rule id="no">
        <one-of>
            <item>no</item>
            <item>nope</item>
            <item>no way</item>
        </one-of>
        <tag>no</tag>
    </rule>
</grammar>
```

重量

您可以將 weight 屬性添加到元素。權重是一個正浮點值，代表在語音辨識期間項目中片語提升的程度。如需詳細資訊，請參閱語音辨識文法規格第 1 版 W3C 建議中的權重。

權重必須大於 0 且小於或等於 10，並且只能有一個小數位。如果重量大於 0 且小於 1，則該短語會產生負面提升。如果重量大於 1 且小於或等於 10，則該短語會積極提升。1 的重量相當於完全沒有體重，並且這句話沒有提高。

為項目分配適當的權重以提高語音識別性能是一項艱鉅的任務。以下是您可以遵循的一些指定權重的提示：

- 從沒有分配項目權重的語法開始。
- 判斷語音中哪些模式經常被誤認。
- 套用不同的權重值，直到您注意到語音辨識效能有所改善，且沒有回歸。

範例 1

例如，如果您有機場的語法，並且發現紐約經常被誤認為是紐瓦克，則可以通過將紐約的權重指定為 5 來積極提升紐約。

```
<rule> id="airport">
    <one-of>
        <item>
            Boston
            <tag>out="Boston"</tag>
        </item>
        <item weight="5">
            New York
            <tag>out="New York"</tag>
        </item>
        <item>
            Newark
            <tag>out="Newark"</tag>
        </item>
    </one-of>
</rule>
```

範例 2

例如，您有一個航空公司預訂代碼的語法，開頭為英文字母，後跟三位數字。預訂代碼很可能以 B 或 D 開頭，但是您發現 B 經常被誤認為 P，D 作為 T。您可以積極提高 B 和 D。

```
<rule> id="alphabet">
    <one-of>
        <item>A<tag>out.letters+='A';</tag></item>
        <item weight="3.5">B<tag>out.letters+='B';</tag></item>
        <item>C<tag>out.letters+='C';</tag></item>
        <item weight="2.9">D<tag>out.letters+='D';</tag></item>
```

```
        <item>E<tag>out.letters+='E';</tag></item>
        <item>F<tag>out.letters+='F';</tag></item>
        <item>G<tag>out.letters+='G';</tag></item>
        <item>H<tag>out.letters+='H';</tag></item>
        <item>I<tag>out.letters+='I';</tag></item>
        <item>J<tag>out.letters+='J';</tag></item>
        <item>K<tag>out.letters+='K';</tag></item>
        <item>L<tag>out.letters+='L';</tag></item>
        <item>M<tag>out.letters+='M';</tag></item>
        <item>N<tag>out.letters+='N';</tag></item>
        <item>O<tag>out.letters+='O';</tag></item>
        <item>P<tag>out.letters+='P';</tag></item>
        <item>Q<tag>out.letters+='Q';</tag></item>
        <item>R<tag>out.letters+='R';</tag></item>
        <item>S<tag>out.letters+='S';</tag></item>
        <item>T<tag>out.letters+='T';</tag></item>
        <item>U<tag>out.letters+='U';</tag></item>
        <item>V<tag>out.letters+='V';</tag></item>
        <item>W<tag>out.letters+='W';</tag></item>
        <item>X<tag>out.letters+='X';</tag></item>
        <item>Y<tag>out.letters+='Y';</tag></item>
        <item>Z<tag>out.letters+='Z';</tag></item>
    </one-of>
</rule>
```

腳本格式

亞馬遜萊克斯 V2 支持以下 ECMAScript 功能來定義文法。

在文法中指定標籤時，Amazon Lex V2 支援下列 ECMAScript 功能。 `tag-format`必須在語法中使用 ECMAScript 標籤時發送到`semantics/1.0`。如需詳細資訊，請參閱 ECMA-262 電子印刷稿 2021 語言規範。

```
<grammar version="1.0"
xmlns="http://www.w3.org/2001/06/grammar"
xml:lang="en-US"
tag-format="semantics/1.0"
root="card_number">
```

主題

- 變數陳述式

- 表達式
- 如果聲明
- 開關語句
- 函數聲明
- 迭代語句
- 區塊陳述式
- 說明
- 不支援的陳

本文件包含來自電子印刷稿標準的材料（可在以下位置找到：https://www.ecma-international.org/publications-and-standards /標準/ ECMA-262/）。ECMAScript 語言規範文件可從 Ecma 國際根據以下許可證獲得。

授權文字

© 2020 埃克瑪國際

本文件可被複製、出版及分發予其他人士，而本文件的某些衍生作品亦可全部或部分編製、複製、出版及分發，前提是所有該等複製品及衍生作品均已包括上述版權公告及本版權許可及免責聲明。根據本版權許可和免責聲明，唯一允許的衍生作品包括：

(i) 為提供評論或解釋而納入本文件全部或部分的作品 (例如文件的註釋版本)，

(ii) 包含本文件全部或部分內容的作品，以納入提供無障礙功能的功能，

(iii) 將本文件翻譯成英文以外的語言，並翻譯成不同的格式，以及

(iv) 在標準符合標準的產品中使用本規格而進行工程，方法是實施 (例如透過全部或部分複製和貼上) 其中的功能。

但是，本文檔本身的內容不得以任何方式進行修改，包括刪除版權聲明或對 Ecma International 的引用，除非需要將其翻譯成英語以外的語言或其他格式。

Ecma 國際文件的正式版本是 Ecma 國際網站上的英文版本。如果翻譯版本與正式版本之間存在差異，則以正式版為準。

上述授予的有限權限是永久的，Ecma International 或其繼任者或受讓人不會撤銷。本文件及其中包含的資訊係依「原狀」提供，ECMA International 拒絕所有明示或暗示的保證，包括但不限於此處使用資訊不會侵犯任何所有權或對特定用途之任何暗示保證。」

變數陳述式

變量語句定義了一個或多個變量。

```
var x = 10;
var x = 10, var y = <expression>;
```

表達式

表示式類型	語法	範例	支援？
規則運算式常值	包含有效正則表達式特殊字符的字	`"^\d\.$"`	否
函數	`function functionName(parameters) { functionBody}`	`var x = function calc() { return 10; }`	否
Delete	`delete expression`	`delete obj.property;`	否
Void	`void expression`	`void (2 == '2');`	否
類型	`typeof expression`	`typeof 42;`	否
成員索引	`expression [ expressions ]`	`var fruits = ["apple"]; fruits[0];`	是
會員點	`expression . identifier`	`out.value`	是
引數	`expression (arguments)`	`new Date('1994-10-11')`	是

表示式類型	語法	範例	支援？
帖子增量	expression++	`var x=10; x++;`	是
遞減後	expression--	`var x=10; x--;`	是
預增量	++expression	`var x=10; ++x;`	是
預遞減	--expression	`var x=10; --x;`	是
一元加/一元減號	+expression / -expression	`+x / -x;`	是
位不	~ expression	`const a = 5; consol e.log( ~a );`	是
邏輯非	! expression	`!(a > 0 || b > 0)`	是
乘法	expression ('*' | '/' | '%') expression	`(x + y) * (a / b)`	是
添加劑	expression ('+' | '-') expression	`(a + b) - (a - (a + b))`	是
位移	expression ('<<' | '>>' | '>>>') expression	`(a >> b) >>> c`	是

表示式類型	語法	範例	支援？
相對	`expression ('<' | '>' | '<=' | '>=') expression`	`if (a > b) { ... }`	是
In (入)	`expression in expression`	`fruits[0] in otherFruits;`	是
平等	`expression ('==' | '!=' | '===' | '!==') expression`	`if (a == b) { ... }`	是
位元與/異或/或	`expression ('&' | '^' | '|') expression`	`a & b / a ^ b / a | b`	是
邏輯和/或	`expression ('&&' | '||') expression`	`if (a && (b ||c)) { ...}`	是
Ternary	`expression ? expression : expression`	`a > b ? obj.prop : 0`	是
指派	`expression = expression`	`out.value = "string";`	是
賦值運算符	`expression ('*=' | '/=' | '+=' | '-=' | '%=') expression`	`a *= 10;`	是

表示式類型	語法	範例	支援？
賦值按位運算符	`expression ('<<=' | '>>=' | '>>>=' | '&=' | '^=' | '|=') expression`	`a <<= 10;`	是
識別符	`identifierSequence` 其中標識符序列是有效字符序列	`fruits=[10, 20, 30];`	是
空文字	`null`	`x = null;`	是
布尔文字	`true | false`	`x = true;`	是
字串常值	`'string' / "string"`	`a = 'hello', b = "world";`	是
十進制文字	`integer [.] digits [exponent]`	`111.11 e+12`	是
十六進制字	`0 (x | X)[0-9a-fA-F]`	`0x123ABC`	是
八進制文字	`O [0-7]`	`"O51"`	是
陣列常值	`[ expression, ... ]`	`v = [a, b, c];`	是
物件常值	`{property: value, ...}`	`out = {value: 1, flag: false};`	是

表示式類型	語法	範例	支援？
括號	(expressions)	`x + (x + y)`	是

如果聲明

```
if (expressions) {
    statements;
} else {
    statements;
}
```

附註：在前面的範例中，expressionsand statements 必須是本文件所支援的其中一個。

開關語句

```
switch (expression) {
    case (expression):
     statements
     .
     .
     .
   default:
      statements
}
```

附註：在前面的範例中，expressionsand statements 必須是本文件所支援的其中一個。

函數聲明

```
function functionIdentifier([parameterList, ...]) {
     <function body>
}
```

迭代語句

迭代語句可以是以下任何一種：

```
// Do..While statement
do {
```

```
   statements
} while (expressions)


// While Loop
while (expressions) {
   statements
}

// For Loop
for ([initialization]; [condition]; [final-expression])
   statement

// For..In
for (variable in object) {
  statement
}
```

區塊陳述式

```
{
   statements
}

// Example
{
    x = 10;
   if (x > 10) {
     console.log("greater than 10");
   }
}
```

注意：在上述範例中，區塊中statements提供的必須是本文件所支援的其中一個。

說明

```
// Single Line Comments
"// <comment>"

// Multineline comments
/**
<comment>
**/
```

不支援的陳

亞馬遜萊克斯 V2 不支持以下 ECMAScript 功能。

主題

- 空白陳述式
- 繼續聲明
- 休息聲明
- 退貨聲明
- 拋出語句
- 嘗試聲明
- 調試器語句
- 標示陳述式
- 類聲明

空白陳述式

空白陳述式用於不提供任何陳述式。以下是空白陳述式的語法：

```
;
```

繼續聲明

不含標籤的 continue 陳述式支援迭代語句。不支持帶有標籤的 continue 語句。

```
// continue with label
// this allows the program to jump to a
// labelled statement (see labelled statement below)
continue <label>;
```

休息聲明

不含標籤的 break 陳述式支援迭代語句。不支援含有標籤的 break 陳述式。

```
// break with label
// this allows the program to break out of a
// labelled statement (see labelled statement below)
break <label>;
```

退貨聲明

```
return expression;
```

拋出語句

throw 語句用於拋出一個用戶定義的異常。

```
throw expression;
```

嘗試聲明

```
try {
  statements
}
catch (expression) {
  statements
}
finally {
  statements
}
```

調試器語句

調試器語句用於調用由環境提供的調試功能。

```
debugger;
```

標示陳述式

標籤化的陳述式可與 break or continue 陳述式搭配使用。

```
label:
   statements


// Example
let str = '';

loop1:
for (let i = 0; i < 5; i++) {
  if (i === 1) {
```

```
    continue loop1;
  }
  str = str + i;
}

console.log(str);
```

類聲明

```
class Rectangle {
  constructor(height, width) {
    this.height = height;
    this.width = width;
  }
}
```

產業文法

行業語法是一組 XML 文件與語法插槽類型一起使用。將互動式語音回應工作流程遷移到 Amazon Lex V2 時，您可以使用這些功能快速提供一致的終端使用者體驗。您可以從三個領域的一系列預先構建的語法中進行選擇：金融服務，保險和電信。還有一組通用的語法，您可以使用它們作為自己的語法的起點。

語法包含收集信息的規則和 ECMAScript 標籤用於語義解釋。

金融服務文法（下載）

金融服務支持以下語法：帳戶和路由號碼，信用卡和貸款號碼，信用評分，開戶和截止日期以及社會安全號碼。

帳戶號碼

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">
```

```
        <!-- Test Cases

        Grammar will support the following inputs:

            Scenario 1:
                Input: My account number is A B C 1 2 3 4
                Output: ABC1234

            Scenario 2:
                Input: My account number is 1 2 3 4 A B C
                Output: 1234ABC

            Scenario 3:
                Input: Hmm My account number is 1 2 3 4 A B C 1
                Output: 123ABC1
        -->

        <rule id="main" scope="public">
            <tag>out=""</tag>
            <item><ruleref uri="#alphanumeric"/><tag>out +=
rules.alphanumeric.alphanum;</tag></item>
            <item repeat="0-1"><ruleref uri="#alphabets"/><tag>out +=
rules.alphabets.letters;</tag></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out +=
rules.digits.numbers</tag></item>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
            <one-of>
                <item repeat="0-1">account number is</item>
                <item repeat="0-1">Account Number</item>
                <item repeat="0-1">Here is my Account Number </item>
                <item repeat="0-1">Yes, It is</item>
                <item repeat="0-1">Yes It is</item>
                <item repeat="0-1">Yes It's</item>
                <item repeat="0-1">My account Id is</item>
                <item repeat="0-1">This is the account Id</item>
                <item repeat="0-1">account Id</item>
            </one-of>
        </rule>

        <rule id="hesitation">
          <one-of>
```

```
            <item>Hmm</item>
            <item>Mmm</item>
            <item>My</item>
          </one-of>
        </rule>

        <rule id="alphanumeric" scope="public">
            <tag>out.alphanum=""</tag>
            <item><ruleref uri="#alphabets"/><tag>out.alphanum +=
rules.alphabets.letters;</tag></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out.alphanum +=
rules.digits.numbers</tag></item>
        </rule>

        <rule id="alphabets">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.letters=""</tag>
            <tag>out.firstOccurence=""</tag>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out.firstOccurence +=
rules.digits.numbers; out.letters += out.firstOccurence;</tag></item>
            <item repeat="1-">
                <one-of>
                    <item>A<tag>out.letters+='A';</tag></item>
                    <item>B<tag>out.letters+='B';</tag></item>
                    <item>C<tag>out.letters+='C';</tag></item>
                    <item>D<tag>out.letters+='D';</tag></item>
                    <item>E<tag>out.letters+='E';</tag></item>
                    <item>F<tag>out.letters+='F';</tag></item>
                    <item>G<tag>out.letters+='G';</tag></item>
                    <item>H<tag>out.letters+='H';</tag></item>
                    <item>I<tag>out.letters+='I';</tag></item>
                    <item>J<tag>out.letters+='J';</tag></item>
                    <item>K<tag>out.letters+='K';</tag></item>
                    <item>L<tag>out.letters+='L';</tag></item>
                    <item>M<tag>out.letters+='M';</tag></item>
                    <item>N<tag>out.letters+='N';</tag></item>
                    <item>O<tag>out.letters+='O';</tag></item>
                    <item>P<tag>out.letters+='P';</tag></item>
                    <item>Q<tag>out.letters+='Q';</tag></item>
                    <item>R<tag>out.letters+='R';</tag></item>
                    <item>S<tag>out.letters+='S';</tag></item>
                    <item>T<tag>out.letters+='T';</tag></item>
                    <item>U<tag>out.letters+='U';</tag></item>
                    <item>V<tag>out.letters+='V';</tag></item>
```

```
                    <item>W<tag>out.letters+='W';</tag></item>
                    <item>X<tag>out.letters+='X';</tag></item>
                    <item>Y<tag>out.letters+='Y';</tag></item>
                    <item>Z<tag>out.letters+='Z';</tag></item>
                </one-of>
            </item>
        </rule>

        <rule id="digits">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.numbers=""</tag>
            <item repeat="1-10">
                <one-of>
                    <item>0<tag>out.numbers+=0;</tag></item>
                    <item>1<tag>out.numbers+=1;</tag></item>
                    <item>2<tag>out.numbers+=2;</tag></item>
                    <item>3<tag>out.numbers+=3;</tag></item>
                    <item>4<tag>out.numbers+=4;</tag></item>
                    <item>5<tag>out.numbers+=5;</tag></item>
                    <item>6<tag>out.numbers+=6;</tag></item>
                    <item>7<tag>out.numbers+=7;</tag></item>
                    <item>8<tag>out.numbers+=8;</tag></item>
                    <item>9<tag>out.numbers+=9;</tag></item>
                </one-of>
            </item>
        </rule>
</grammar>
```

路由號碼

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="digits"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:
```

```
             Scenario 1:
                 Input: My routing number is 1 2 3 4 5 6 7 8 9
                 Output: 123456789

             Scenario 2:
                 Input: routing number 1 2 3 4 5 6 7 8 9
                 Output: 123456789

         -->

        <rule id="digits">
            <tag>out=""</tag>
            <item><ruleref uri="#singleDigit"/><tag>out += rules.singleDigit.digit;</
tag></item>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
           <one-of>
              <item repeat="0-1">My routing number</item>
              <item repeat="0-1">Routing number of</item>
              <item repeat="0-1">The routing number is</item>
           </one-of>
        </rule>

        <rule id="hesitation">
            <one-of>
               <item>Hmm</item>
               <item>Mmm</item>
               <item>My</item>
            </one-of>
          </rule>

        <rule id="singleDigit">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.digit=""</tag>
            <item repeat="16">
                <one-of>
                    <item>0<tag>out.digit+=0;</tag></item>
                    <item>zero<tag>out.digit+=0;</tag></item>
                    <item>1<tag>out.digit+=1;</tag></item>
                    <item>one<tag>out.digit+=1;</tag></item>
                    <item>2<tag>out.digit+=2;</tag></item>
```

```
                    <item>two<tag>out.digit+=2;</tag></item>
                    <item>3<tag>out.digit+=3;</tag></item>
                    <item>three<tag>out.digit+=3;</tag></item>
                    <item>4<tag>out.digit+=4;</tag></item>
                    <item>four<tag>out.digit+=4;</tag></item>
                    <item>5<tag>out.digit+=5;</tag></item>
                    <item>five<tag>out.digit+=5;</tag></item>
                    <item>6<tag>out.digit+=6;</tag></item>
                    <item>six<tag>out.digit+=5;</tag></item>
                    <item>7<tag>out.digit+=7;</tag></item>
                    <item>seven<tag>out.digit+=7;</tag></item>
                    <item>8<tag>out.digit+=8;</tag></item>
                    <item>eight<tag>out.digit+=8;</tag></item>
                    <item>9<tag>out.digit+=9;</tag></item>
                    <item>nine<tag>out.digit+=9;</tag></item>
                </one-of>
            </item>
        </rule>
</grammar>
```

信用卡號碼

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="digits"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: My credit card number is 1 2 3 4 5 6 7 8 9 1 2 3 4 5 6 7
                 Output: 1234567891234567

             Scenario 2:
                 Input: card number 1 2 3 4 5 6 7 8 9 1 2 3 4 5 6 7
                 Output: 1234567891234567
```

```
         -->

        <rule id="digits">
            <tag>out=""</tag>
            <item><ruleref uri="#singleDigit"/><tag>out += rules.singleDigit.digit;</
tag></item>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
           <one-of>
              <item repeat="0-1">My credit card number is</item>
              <item repeat="0-1">card number</item>
           </one-of>
        </rule>

        <rule id="hesitation">
            <one-of>
               <item>Hmm</item>
               <item>Mmm</item>
               <item>My</item>
            </one-of>
          </rule>

        <rule id="singleDigit">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.digit=""</tag>
            <item repeat="16">
                <one-of>
                    <item>0<tag>out.digit+=0;</tag></item>
                    <item>zero<tag>out.digit+=0;</tag></item>
                    <item>1<tag>out.digit+=1;</tag></item>
                    <item>one<tag>out.digit+=1;</tag></item>
                    <item>2<tag>out.digit+=2;</tag></item>
                    <item>two<tag>out.digit+=2;</tag></item>
                    <item>3<tag>out.digit+=3;</tag></item>
                    <item>three<tag>out.digit+=3;</tag></item>
                    <item>4<tag>out.digit+=4;</tag></item>
                    <item>four<tag>out.digit+=4;</tag></item>
                    <item>5<tag>out.digit+=5;</tag></item>
                    <item>five<tag>out.digit+=5;</tag></item>
                    <item>6<tag>out.digit+=6;</tag></item>
                    <item>six<tag>out.digit+=5;</tag></item>
```

```
                    <item>7<tag>out.digit+=7;</tag></item>
                    <item>seven<tag>out.digit+=7;</tag></item>
                    <item>8<tag>out.digit+=8;</tag></item>
                    <item>eight<tag>out.digit+=8;</tag></item>
                    <item>9<tag>out.digit+=9;</tag></item>
                    <item>nine<tag>out.digit+=9;</tag></item>
                </one-of>
            </item>
        </rule>
</grammar>
```

貸款識別碼

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">


        <!-- Test Cases

        Grammar will support the following inputs:

            Scenario 1:
                Input: My loan Id is A B C 1 2 3 4
                Output: ABC1234
        -->

        <rule id="main" scope="public">
            <tag>out=""</tag>
            <item><ruleref uri="#alphanumeric"/><tag>out +=
 rules.alphanumeric.alphanum;</tag></item>
            <item repeat="0-1"><ruleref uri="#alphabets"/><tag>out +=
 rules.alphabets.letters;</tag></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out +=
 rules.digits.numbers</tag></item>
        </rule>
```

```
        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
            <one-of>
                <item repeat="0-1">my loan number is</item>
                <item repeat="0-1">The loan number</item>
                <item repeat="0-1">The loan is </item>
                <item repeat="0-1">The number is</item>
                <item repeat="0-1">loan number</item>
                <item repeat="0-1">loan number of</item>
                <item repeat="0-1">loan Id is</item>
                <item repeat="0-1">My loan Id is</item>
            </one-of>
        </rule>

        <rule id="hesitation">
          <one-of>
             <item>Hmm</item>
             <item>Mmm</item>
             <item>My</item>
          </one-of>
        </rule>

        <rule id="alphanumeric" scope="public">
            <tag>out.alphanum=""</tag>
            <item><ruleref uri="#alphabets"/><tag>out.alphanum +=
rules.alphabets.letters;</tag></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out.alphanum +=
rules.digits.numbers</tag></item>
        </rule>

        <rule id="alphabets">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.letters=""</tag>
            <tag>out.firstOccurence=""</tag>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out.firstOccurence +=
rules.digits.numbers; out.letters += out.firstOccurence;</tag></item>
            <item repeat="1-">
                <one-of>
                    <item>A<tag>out.letters+='A';</tag></item>
                    <item>B<tag>out.letters+='B';</tag></item>
                    <item>C<tag>out.letters+='C';</tag></item>
                    <item>D<tag>out.letters+='D';</tag></item>
                    <item>E<tag>out.letters+='E';</tag></item>
                    <item>F<tag>out.letters+='F';</tag></item>
```

```
                    <item>G<tag>out.letters+='G';</tag></item>
                    <item>H<tag>out.letters+='H';</tag></item>
                    <item>I<tag>out.letters+='I';</tag></item>
                    <item>J<tag>out.letters+='J';</tag></item>
                    <item>K<tag>out.letters+='K';</tag></item>
                    <item>L<tag>out.letters+='L';</tag></item>
                    <item>M<tag>out.letters+='M';</tag></item>
                    <item>N<tag>out.letters+='N';</tag></item>
                    <item>O<tag>out.letters+='O';</tag></item>
                    <item>P<tag>out.letters+='P';</tag></item>
                    <item>Q<tag>out.letters+='Q';</tag></item>
                    <item>R<tag>out.letters+='R';</tag></item>
                    <item>S<tag>out.letters+='S';</tag></item>
                    <item>T<tag>out.letters+='T';</tag></item>
                    <item>U<tag>out.letters+='U';</tag></item>
                    <item>V<tag>out.letters+='V';</tag></item>
                    <item>W<tag>out.letters+='W';</tag></item>
                    <item>X<tag>out.letters+='X';</tag></item>
                    <item>Y<tag>out.letters+='Y';</tag></item>
                    <item>Z<tag>out.letters+='Z';</tag></item>
                </one-of>
            </item>
        </rule>

        <rule id="digits">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.numbers=""</tag>
            <item repeat="1-10">
                <one-of>
                    <item>0<tag>out.numbers+=0;</tag></item>
                    <item>1<tag>out.numbers+=1;</tag></item>
                    <item>2<tag>out.numbers+=2;</tag></item>
                    <item>3<tag>out.numbers+=3;</tag></item>
                    <item>4<tag>out.numbers+=4;</tag></item>
                    <item>5<tag>out.numbers+=5;</tag></item>
                    <item>6<tag>out.numbers+=6;</tag></item>
                    <item>7<tag>out.numbers+=7;</tag></item>
                    <item>8<tag>out.numbers+=8;</tag></item>
                    <item>9<tag>out.numbers+=9;</tag></item>
                </one-of>
            </item>
        </rule>
</grammar>
```

信用評分

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: The number is fifteen
                 Output: 15

             Scenario 2:
                 Input: My credit score is fifteen
                 Output: 15
         -->

        <rule id="main" scope="public">
            <tag>out=""</tag>
            <one-of>
              <item repeat="1"><ruleref uri="#digits"/><tag>out+= rules.digits;</tag></
item>
              <item repeat="1"><ruleref uri="#teens"/><tag>out+= rules.teens;</tag></
item>
              <item repeat="1"><ruleref uri="#above_twenty"/><tag>out+=
 rules.above_twenty;</tag></item>
            </one-of>
        </rule>

        <rule id="text">
           <one-of>
              <item repeat="0-1">Credit score is</item>
              <item repeat="0-1">Last digits are</item>
              <item repeat="0-1">The number is</item>
              <item repeat="0-1">That's</item>
              <item repeat="0-1">It is</item>
```

```
            <item repeat="0-1">My credit score is</item>
        </one-of>
    </rule>

    <rule id="digits">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <one-of>
            <item>0<tag>out=0;</tag></item>
            <item>1<tag>out=1;</tag></item>
            <item>2<tag>out=2;</tag></item>
            <item>3<tag>out=3;</tag></item>
            <item>4<tag>out=4;</tag></item>
            <item>5<tag>out=5;</tag></item>
            <item>6<tag>out=6;</tag></item>
            <item>7<tag>out=7;</tag></item>
            <item>8<tag>out=8;</tag></item>
            <item>9<tag>out=9;</tag></item>
            <item>one<tag>out=1;</tag></item>
            <item>two<tag>out=2;</tag></item>
            <item>three<tag>out=3;</tag></item>
            <item>four<tag>out=4;</tag></item>
            <item>five<tag>out=5;</tag></item>
            <item>six<tag>out=6;</tag></item>
            <item>seven<tag>out=7;</tag></item>
            <item>eight<tag>out=8;</tag></item>
            <item>nine<tag>out=9;</tag></item>
        </one-of>
    </rule>

    <rule id="teens">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <one-of>
            <item>ten<tag>out=10;</tag></item>
            <item>eleven<tag>out=11;</tag></item>
            <item>twelve<tag>out=12;</tag></item>
            <item>thirteen<tag>out=13;</tag></item>
            <item>fourteen<tag>out=14;</tag></item>
            <item>fifteen<tag>out=15;</tag></item>
            <item>sixteen<tag>out=16;</tag></item>
            <item>seventeen<tag>out=17;</tag></item>
            <item>eighteen<tag>out=18;</tag></item>
            <item>nineteen<tag>out=19;</tag></item>
            <item>10<tag>out=10;</tag></item>
            <item>11<tag>out=11;</tag></item>
```

```
                <item>12<tag>out=12;</tag></item>
                <item>13<tag>out=13;</tag></item>
                <item>14<tag>out=14;</tag></item>
                <item>15<tag>out=15;</tag></item>
                <item>16<tag>out=16;</tag></item>
                <item>17<tag>out=17;</tag></item>
                <item>18<tag>out=18;</tag></item>
                <item>19<tag>out=19;</tag></item>
            </one-of>
        </rule>

        <rule id="above_twenty">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <one-of>
                <item>twenty<tag>out=20;</tag></item>
                <item>thirty<tag>out=30;</tag></item>
                <item>forty<tag>out=40;</tag></item>
                <item>fifty<tag>out=50;</tag></item>
                <item>sixty<tag>out=60;</tag></item>
                <item>seventy<tag>out=70;</tag></item>
                <item>eighty<tag>out=80;</tag></item>
                <item>ninety<tag>out=90;</tag></item>
                <item>20<tag>out=20;</tag></item>
                <item>30<tag>out=30;</tag></item>
                <item>40<tag>out=40;</tag></item>
                <item>50<tag>out=50;</tag></item>
                <item>60<tag>out=60;</tag></item>
                <item>70<tag>out=70;</tag></item>
                <item>80<tag>out=80;</tag></item>
                <item>90<tag>out=90;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>

</grammar>
```

開戶日期

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
```

```
                              http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: I opened account on July Two Thousand and Eleven
                 Output: 07/11

             Scenario 2:
                 Input: I need account number opened on July Two Thousand and Eleven
                 Output: 07/11

         -->

        <rule id="main" scope="public">
            <tag>out=""</tag>
            <item repeat="1-10">
                <item repeat="1"><ruleref uri="#months"/><tag>out = out +
 rules.months.mon + "/";</tag></item>
                <one-of>
                    <item><ruleref uri="#thousands"/><tag>out += rules.thousands;</
tag></item>
                    <item repeat="0-1"><ruleref uri="#digits"/><tag>out +=
 rules.digits;</tag></item>
                    <item repeat="0-1"><ruleref uri="#teens"/><tag>out +=
 rules.teens;</tag></item>
                    <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out +=
 rules.above_twenty;</tag></item>
                </one-of>
            </item>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
           <one-of>
              <item repeat="0-1">I opened account on </item>
              <item repeat="0-1">I need account number opened on </item>
           </one-of>
```

```
        </rule>

        <rule id="hesitation">
            <one-of>
               <item>Hmm</item>
               <item>Mmm</item>
               <item>My</item>
            </one-of>
          </rule>
        <rule id="months">
           <item repeat="0-1"><ruleref uri="#text"/></item>
           <tag>out.mon=""</tag>
           <one-of>
               <item>january<tag>out.mon+="01";</tag></item>
               <item>february<tag>out.mon+="02";</tag></item>
               <item>march<tag>out.mon+="03";</tag></item>
               <item>april<tag>out.mon+="04";</tag></item>
               <item>may<tag>out.mon+="05";</tag></item>
               <item>june<tag>out.mon+="06";</tag></item>
               <item>july<tag>out.mon+="07";</tag></item>
               <item>august<tag>out.mon+="08";</tag></item>
               <item>september<tag>out.mon+="09";</tag></item>
               <item>october<tag>out.mon+="10";</tag></item>
               <item>november<tag>out.mon+="11";</tag></item>
               <item>december<tag>out.mon+="12";</tag></item>
               <item>jan<tag>out.mon+="01";</tag></item>
               <item>feb<tag>out.mon+="02";</tag></item>
               <item>aug<tag>out.mon+="08";</tag></item>
               <item>sept<tag>out.mon+="09";</tag></item>
               <item>oct<tag>out.mon+="10";</tag></item>
               <item>nov<tag>out.mon+="11";</tag></item>
               <item>dec<tag>out.mon+="12";</tag></item>
           </one-of>
       </rule>

        <rule id="digits">
            <one-of>
                <item>zero<tag>out=0;</tag></item>
                <item>one<tag>out=1;</tag></item>
                <item>two<tag>out=2;</tag></item>
                <item>three<tag>out=3;</tag></item>
                <item>four<tag>out=4;</tag></item>
                <item>five<tag>out=5;</tag></item>
                <item>six<tag>out=6;</tag></item>
```

```
                <item>seven<tag>out=7;</tag></item>
                <item>eight<tag>out=8;</tag></item>
                <item>nine<tag>out=9;</tag></item>
            </one-of>
        </rule>

        <rule id="teens">
            <one-of>
                <item>ten<tag>out=10;</tag></item>
                <item>eleven<tag>out=11;</tag></item>
                <item>twelve<tag>out=12;</tag></item>
                <item>thirteen<tag>out=13;</tag></item>
                <item>fourteen<tag>out=14;</tag></item>
                <item>fifteen<tag>out=15;</tag></item>
                <item>sixteen<tag>out=16;</tag></item>
                <item>seventeen<tag>out=17;</tag></item>
                <item>eighteen<tag>out=18;</tag></item>
                <item>nineteen<tag>out=19;</tag></item>
            </one-of>
        </rule>

        <rule id="thousands">
            <item>two thousand<!--<tag>out=2000;</tag>--></item>
            <item repeat="0-1">and</item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out = rules.digits;</tag></
item>
            <item repeat="0-1"><ruleref uri="#teens"/><tag>out = rules.teens;</tag></
item>
            <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out =
 rules.above_twenty;</tag></item>
        </rule>

        <rule id="above_twenty">
            <one-of>
                <item>twenty<tag>out=20;</tag></item>
                <item>thirty<tag>out=30;</tag></item>
                <item>forty<tag>out=40;</tag></item>
                <item>fifty<tag>out=50;</tag></item>
                <item>sixty<tag>out=60;</tag></item>
                <item>seventy<tag>out=70;</tag></item>
                <item>eighty<tag>out=80;</tag></item>
                <item>ninety<tag>out=90;</tag></item>
            </one-of>
```

```
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>
</grammar>
```

自動付款日期

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: I want to schedule auto pay for twenty five Dollar
                 Output: $25

             Scenario 2:
                 Input: Setup automatic payments for twenty five dollars
                 Output: $25

         -->

        <rule id="main" scope="public">
            <tag>out="$"</tag>
            <one-of>
                <item><ruleref uri="#sub_hundred"/><tag>out += rules.sub_hundred.sh;</
tag></item>
                <item><ruleref uri="#subThousands"/><tag>out += rules.subThousands;</
tag></item>
            </one-of>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
```

```
            <one-of>
               <item repeat="0-1">I want to schedule auto pay for</item>
               <item repeat="0-1">Setup automatic payments for twenty five dollars</
item>
               <item repeat="0-1">Auto pay amount of</item>
               <item repeat="0-1">Set it up for</item>
            </one-of>
         </rule>

         <rule id="hesitation">
             <one-of>
                <item>Hmm</item>
                <item>Mmm</item>
                <item>My</item>
             </one-of>
           </rule>

         <rule id="digits">
             <item repeat="0-1"><ruleref uri="#text"/></item>
             <tag>out.num = 0;</tag>
             <one-of>
                 <item>0<tag>out.num+=0;</tag></item>
                 <item>1<tag>out.num+=1;</tag></item>
                 <item>2<tag>out.num+=2;</tag></item>
                 <item>3<tag>out.num+=3;</tag></item>
                 <item>4<tag>out.num+=4;</tag></item>
                 <item>5<tag>out.num+=5;</tag></item>
                 <item>6<tag>out.num+=6;</tag></item>
                 <item>7<tag>out.num+=7;</tag></item>
                 <item>8<tag>out.num+=8;</tag></item>
                 <item>9<tag>out.num+=9;</tag></item>
                 <item>one<tag>out.num+=1;</tag></item>
                 <item>two<tag>out.num+=2;</tag></item>
                 <item>three<tag>out.num+=3;</tag></item>
                 <item>four<tag>out.num+=4;</tag></item>
                 <item>five<tag>out.num+=5;</tag></item>
                 <item>six<tag>out.num+=6;</tag></item>
                 <item>seven<tag>out.num+=7;</tag></item>
                 <item>eight<tag>out.num+=8;</tag></item>
                 <item>nine<tag>out.num+=9;</tag></item>
             </one-of>
             <item repeat="0-1"><ruleref uri="#currency"/></item>
         </rule>
```

```
        <rule id="teens">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.teen = 0;</tag>
            <one-of>
                <item>ten<tag>out.teen+=10;</tag></item>
                <item>eleven<tag>out.teen+=11;</tag></item>
                <item>twelve<tag>out.teen+=12;</tag></item>
                <item>thirteen<tag>out.teen+=13;</tag></item>
                <item>fourteen<tag>out.teen+=14;</tag></item>
                <item>fifteen<tag>out.teen+=15;</tag></item>
                <item>sixteen<tag>out.teen+=16;</tag></item>
                <item>seventeen<tag>out.teen+=17;</tag></item>
                <item>eighteen<tag>out.teen+=18;</tag></item>
                <item>nineteen<tag>out.teen+=19;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#currency"/></item>
        </rule>

        <rule id="above_twenty">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.tens = 0;</tag>
            <one-of>
                <item>twenty<tag>out.tens+=20;</tag></item>
                <item>thirty<tag>out.tens+=30;</tag></item>
                <item>forty<tag>out.tens+=40;</tag></item>
                <item>fifty<tag>out.tens+=50;</tag></item>
                <item>sixty<tag>out.tens+=60;</tag></item>
                <item>seventy<tag>out.tens+=70;</tag></item>
                <item>eighty<tag>out.tens+=80;</tag></item>
                <item>ninety<tag>out.tens+=90;</tag></item>
                <item>hundred<tag>out.tens+=100;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#currency"/></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out.tens +=
rules.digits.num;</tag></item>
        </rule>

        <rule id="currency">
            <one-of>
                <item repeat="0-1">dollars</item>
                <item repeat="0-1">Dollars</item>
                <item repeat="0-1">dollar</item>
                <item repeat="0-1">Dollar</item>
            </one-of>
```

```
        </rule>


        <rule id="sub_hundred">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.sh = 0;</tag>
            <one-of>
                <item><ruleref uri="#teens"/><tag>out.sh += rules.teens.teen;</tag></
item>
                <item>
                    <ruleref uri="#above_twenty"/><tag>out.sh +=
 rules.above_twenty.tens;</tag>
                </item>
                <item><ruleref uri="#digits"/><tag>out.sh += rules.digits.num;</tag></
item>
            </one-of>
         </rule>

        <rule id="subThousands">
            <ruleref uri="#sub_hundred"/><tag>out = (100 * rules.sub_hundred.sh);</tag>
            hundred
            <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out +=
 rules.above_twenty.tens;</tag></item>
            <item repeat="0-1"><ruleref uri="#teens"/><tag>out += rules.teens.teen;</
tag></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits.num;</
tag></item>
            <item repeat="0-1"><ruleref uri="#currency"/></item>
        </rule>
</grammar>
```

信用卡到期日

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="dateCardExpiration"
         mode="voice"
         tag-format="semantics/1.0">
```

```
        <rule id="dateCardExpiration" scope="public">
            <tag>out=""</tag>
            <item repeat="1"><ruleref uri="#months"/><tag>out = out + rules.months;</
tag></item>
            <item repeat="1"><ruleref uri="#year"/><tag>out += " " + rules.year.yr;</
tag></item>
        </rule>

        <!-- Test Cases

        Grammar will support the following inputs:

            Scenario 1:
                Input: My card expiration date is july eleven
                Output: 07 2011

            Scenario 2:
                Input: My card expiration date is may twenty six
                Output: 05 2026

        -->

        <rule id="text">
           <item repeat="0-1"><ruleref uri="#hesitation"/></item>
           <one-of>
              <item repeat="0-1">My card expiration date is </item>
              <item repeat="0-1">Expiration date is </item>
           </one-of>
        </rule>

        <rule id="hesitation">
          <one-of>
             <item>Hmm</item>
             <item>Mmm</item>
             <item>My</item>
          </one-of>
        </rule>

        <rule id="months">
           <item repeat="0-1"><ruleref uri="#text"/></item>
           <one-of>
               <item>january<tag>out="01";</tag></item>
               <item>february<tag>out="02";</tag></item>
               <item>march<tag>out="03";</tag></item>
```

```
            <item>april<tag>out="04";</tag></item>
            <item>may<tag>out="05";</tag></item>
            <item>june<tag>out="06";</tag></item>
            <item>july<tag>out="07";</tag></item>
            <item>august<tag>out="08";</tag></item>
            <item>september<tag>out="09";</tag></item>
            <item>october<tag>out="10";</tag></item>
            <item>november<tag>out="11";</tag></item>
            <item>december<tag>out="12";</tag></item>
            <item>jan<tag>out="01";</tag></item>
            <item>feb<tag>out="02";</tag></item>
            <item>aug<tag>out="08";</tag></item>
            <item>sept<tag>out="09";</tag></item>
            <item>oct<tag>out="10";</tag></item>
            <item>nov<tag>out="11";</tag></item>
            <item>dec<tag>out="12";</tag></item>
            <item>1<tag>out="01";</tag></item>
            <item>2<tag>out="02";</tag></item>
            <item>3<tag>out="03";</tag></item>
            <item>4<tag>out="04";</tag></item>
            <item>5<tag>out="05";</tag></item>
            <item>6<tag>out="06";</tag></item>
            <item>7<tag>out="07";</tag></item>
            <item>8<tag>out="08";</tag></item>
            <item>9<tag>out="09";</tag></item>
            <item>ten<tag>out="10";</tag></item>
            <item>eleven<tag>out="11";</tag></item>
            <item>twelve<tag>out="12";</tag></item>
        </one-of>
    </rule>

     <rule id="digits">
         <item repeat="0-1"><ruleref uri="#text"/></item>
         <one-of>
             <item>0<tag>out=0;</tag></item>
             <item>1<tag>out=1;</tag></item>
             <item>2<tag>out=2;</tag></item>
             <item>3<tag>out=3;</tag></item>
             <item>4<tag>out=4;</tag></item>
             <item>5<tag>out=5;</tag></item>
             <item>6<tag>out=6;</tag></item>
             <item>7<tag>out=7;</tag></item>
             <item>8<tag>out=8;</tag></item>
             <item>9<tag>out=9;</tag></item>
```

```
                <item>one<tag>out=1;</tag></item>
                <item>two<tag>out=2;</tag></item>
                <item>three<tag>out=3;</tag></item>
                <item>four<tag>out=4;</tag></item>
                <item>five<tag>out=5;</tag></item>
                <item>six<tag>out=6;</tag></item>
                <item>seven<tag>out=7;</tag></item>
                <item>eight<tag>out=8;</tag></item>
                <item>nine<tag>out=9;</tag></item>
            </one-of>
        </rule>

        <rule id="year">
          <tag>out.yr="20"</tag>
          <one-of>
              <item><ruleref uri="#teens"/><tag>out.yr += rules.teens;</tag></item>
              <item><ruleref uri="#above_twenty"/><tag>out.yr += rules.above_twenty;</
tag></item>
          </one-of>
        </rule>

        <rule id="teens">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <one-of>
                <item>ten<tag>out=10;</tag></item>
                <item>eleven<tag>out=11;</tag></item>
                <item>twelve<tag>out=12;</tag></item>
                <item>thirteen<tag>out=13;</tag></item>
                <item>fourteen<tag>out=14;</tag></item>
                <item>fifteen<tag>out=15;</tag></item>
                <item>sixteen<tag>out=16;</tag></item>
                <item>seventeen<tag>out=17;</tag></item>
                <item>eighteen<tag>out=18;</tag></item>
                <item>nineteen<tag>out=19;</tag></item>
                <item>10<tag>out=10;</tag></item>
                <item>11<tag>out=11;</tag></item>
                <item>12<tag>out=12;</tag></item>
                <item>13<tag>out=13;</tag></item>
                <item>14<tag>out=14;</tag></item>
                <item>15<tag>out=15;</tag></item>
                <item>16<tag>out=16;</tag></item>
                <item>17<tag>out=17;</tag></item>
                <item>18<tag>out=18;</tag></item>
                <item>19<tag>out=19;</tag></item>
```

```
            </one-of>
        </rule>

        <rule id="above_twenty">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <one-of>
                <item>twenty<tag>out=20;</tag></item>
                <item>thirty<tag>out=30;</tag></item>
                <item>forty<tag>out=40;</tag></item>
                <item>fifty<tag>out=50;</tag></item>
                <item>sixty<tag>out=60;</tag></item>
                <item>seventy<tag>out=70;</tag></item>
                <item>eighty<tag>out=80;</tag></item>
                <item>ninety<tag>out=90;</tag></item>
                <item>20<tag>out=20;</tag></item>
                <item>30<tag>out=30;</tag></item>
                <item>40<tag>out=40;</tag></item>
                <item>50<tag>out=50;</tag></item>
                <item>60<tag>out=60;</tag></item>
                <item>70<tag>out=70;</tag></item>
                <item>80<tag>out=80;</tag></item>
                <item>90<tag>out=90;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>
</grammar>
```

聲明日期

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:
```

```
            Scenario 1:
                Input: Show me statements from July Five Two Thousand and Eleven
                Output: 07/5/11

            Scenario 2:
                Input: Show me statements from July Sixteen Two Thousand and Eleven
                Output: 07/16/11

            Scenario 3:
                Input: Show me statements from July Thirty Two Thousand and Eleven
                Output: 07/30/11
        -->

        <rule id="main" scope="public">
            <tag>out=""</tag>
            <item>
                <item repeat="1"><ruleref uri="#months"/><tag>out = out +
 rules.months.mon + "/";</tag></item>
                <one-of>
                  <item><ruleref uri="#digits"/><tag>out += rules.digits + "/";</
tag></item>
                  <item><ruleref uri="#teens"/><tag>out += rules.teens+ "/";</tag></
item>
                  <item><ruleref uri="#above_twenty"/><tag>out += rules.above_twenty+
 "/";</tag></item>
                </one-of>
                <one-of>
                    <item><ruleref uri="#thousands"/><tag>out += rules.thousands;</
tag></item>
                    <item repeat="0-1"><ruleref uri="#digits"/><tag>out +=
 rules.digits;</tag></item>
                    <item repeat="0-1"><ruleref uri="#teens"/><tag>out +=
 rules.teens;</tag></item>
                    <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out +=
 rules.above_twenty;</tag></item>
                </one-of>
            </item>
        </rule>

        <rule id="text">
          <item repeat="0-1"><ruleref uri="#hesitation"/></item>
           <one-of>
              <item repeat="0-1">I want to see bank statements from </item>
```

```
            <item repeat="0-1">Show me statements from</item>
         </one-of>
      </rule>

      <rule id="hesitation">
         <one-of>
            <item>Hmm</item>
            <item>Mmm</item>
            <item>My</item>
         </one-of>
       </rule>

      <rule id="months">
         <tag>out.mon=""</tag>
      <item repeat="0-1"><ruleref uri="#text"/></item>
         <one-of>
           <item>january<tag>out.mon+="01";</tag></item>
           <item>february<tag>out.mon+="02";</tag></item>
           <item>march<tag>out.mon+="03";</tag></item>
           <item>april<tag>out.mon+="04";</tag></item>
           <item>may<tag>out.mon+="05";</tag></item>
           <item>june<tag>out.mon+="06";</tag></item>
           <item>july<tag>out.mon+="07";</tag></item>
           <item>august<tag>out.mon+="08";</tag></item>
           <item>september<tag>out.mon+="09";</tag></item>
           <item>october<tag>out.mon+="10";</tag></item>
           <item>november<tag>out.mon+="11";</tag></item>
           <item>december<tag>out.mon+="12";</tag></item>
           <item>jan<tag>out.mon+="01";</tag></item>
           <item>feb<tag>out.mon+="02";</tag></item>
           <item>aug<tag>out.mon+="08";</tag></item>
           <item>sept<tag>out.mon+="09";</tag></item>
           <item>oct<tag>out.mon+="10";</tag></item>
           <item>nov<tag>out.mon+="11";</tag></item>
           <item>dec<tag>out.mon+="12";</tag></item>
         </one-of>
      </rule>

       <rule id="digits">
           <one-of>
               <item>zero<tag>out=0;</tag></item>
               <item>one<tag>out=1;</tag></item>
               <item>two<tag>out=2;</tag></item>
               <item>three<tag>out=3;</tag></item>
```

```
                <item>four<tag>out=4;</tag></item>
                <item>five<tag>out=5;</tag></item>
                <item>six<tag>out=6;</tag></item>
                <item>seven<tag>out=7;</tag></item>
                <item>eight<tag>out=8;</tag></item>
                <item>nine<tag>out=9;</tag></item>
            </one-of>
        </rule>

        <rule id="teens">
            <one-of>
                <item>ten<tag>out=10;</tag></item>
                <item>eleven<tag>out=11;</tag></item>
                <item>twelve<tag>out=12;</tag></item>
                <item>thirteen<tag>out=13;</tag></item>
                <item>fourteen<tag>out=14;</tag></item>
                <item>fifteen<tag>out=15;</tag></item>
                <item>sixteen<tag>out=16;</tag></item>
                <item>seventeen<tag>out=17;</tag></item>
                <item>eighteen<tag>out=18;</tag></item>
                <item>nineteen<tag>out=19;</tag></item>
            </one-of>
        </rule>

        <rule id="thousands">
            <item>two thousand</item>
            <item repeat="0-1">and</item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out = rules.digits;</
tag></item>
            <item repeat="0-1"><ruleref uri="#teens"/><tag>out = rules.teens;</tag></
item>
            <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out =
 rules.above_twenty;</tag></item>
        </rule>

        <rule id="above_twenty">
            <one-of>
                <item>twenty<tag>out=20;</tag></item>
                <item>thirty<tag>out=30;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>
```

```
 </grammar>
```

交易日期

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: My last incorrect transaction date is july twenty three
                 Output: 07/23

             Scenario 2:
                 Input: My last incorrect transaction date is july fifteen
                 Output: 07/15

         -->

        <rule id="main" scope="public">
            <tag>out=""</tag>
            <item repeat="1-10">
                <item><ruleref uri="#months"/><tag>out= rules.months.mon + "/";</tag></
item>
                <one-of>
                    <item><ruleref uri="#digits"/><tag>out+= rules.digits;</tag></item>
                    <item><ruleref uri="#teens"/><tag>out+= rules.teens;</tag></item>
                    <item><ruleref uri="#above_twenty"/><tag>out+=
 rules.above_twenty;</tag></item>
                </one-of>
            </item>
        </rule>

        <rule id="text">
```

```
        <item repeat="0-1"><ruleref uri="#hesitation"/></item>
      <one-of>
         <item repeat="0-1">My last incorrect transaction date is</item>
         <item repeat="0-1">It is</item>
      </one-of>
   </rule>
   <rule id="hesitation">
       <one-of>
          <item>Hmm</item>
          <item>Mmm</item>
          <item>My</item>
       </one-of>
     </rule>

   <rule id="months">
      <item repeat="0-1"><ruleref uri="#text"/></item>
      <tag>out.mon=""</tag>
      <one-of>
          <item>january<tag>out.mon+="01";</tag></item>
          <item>february<tag>out.mon+="02";</tag></item>
          <item>march<tag>out.mon+="03";</tag></item>
          <item>april<tag>out.mon+="04";</tag></item>
          <item>may<tag>out.mon+="05";</tag></item>
          <item>june<tag>out.mon+="06";</tag></item>
          <item>july<tag>out.mon+="07";</tag></item>
          <item>august<tag>out.mon+="08";</tag></item>
          <item>september<tag>out.mon+="09";</tag></item>
          <item>october<tag>out.mon+="10";</tag></item>
          <item>november<tag>out.mon+="11";</tag></item>
          <item>december<tag>out.mon+="12";</tag></item>
          <item>jan<tag>out.mon+="01";</tag></item>
          <item>feb<tag>out.mon+="02";</tag></item>
          <item>aug<tag>out.mon+="08";</tag></item>
          <item>sept<tag>out.mon+="09";</tag></item>
          <item>oct<tag>out.mon+="10";</tag></item>
          <item>nov<tag>out.mon+="11";</tag></item>
          <item>dec<tag>out.mon+="12";</tag></item>
      </one-of>
   </rule>

   <rule id="digits">
       <item repeat="0-1"><ruleref uri="#text"/></item>
       <one-of>
           <item>0<tag>out=0;</tag></item>
```

```
            <item>1<tag>out=1;</tag></item>
            <item>2<tag>out=2;</tag></item>
            <item>3<tag>out=3;</tag></item>
            <item>4<tag>out=4;</tag></item>
            <item>5<tag>out=5;</tag></item>
            <item>6<tag>out=6;</tag></item>
            <item>7<tag>out=7;</tag></item>
            <item>8<tag>out=8;</tag></item>
            <item>9<tag>out=9;</tag></item>
            <item>first<tag>out=01;</tag></item>
            <item>second<tag>out=02;</tag></item>
            <item>third<tag>out=03;</tag></item>
            <item>fourth<tag>out=04;</tag></item>
            <item>fifth<tag>out=05;</tag></item>
            <item>sixth<tag>out=06;</tag></item>
            <item>seventh<tag>out=07;</tag></item>
            <item>eighth<tag>out=08;</tag></item>
            <item>ninth<tag>out=09;</tag></item>
            <item>one<tag>out=1;</tag></item>
            <item>two<tag>out=2;</tag></item>
            <item>three<tag>out=3;</tag></item>
            <item>four<tag>out=4;</tag></item>
            <item>five<tag>out=5;</tag></item>
            <item>six<tag>out=6;</tag></item>
            <item>seven<tag>out=7;</tag></item>
            <item>eight<tag>out=8;</tag></item>
            <item>nine<tag>out=9;</tag></item>
        </one-of>
    </rule>

    <rule id="teens">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <one-of>
            <item>ten<tag>out=10;</tag></item>
            <item>tenth<tag>out=10;</tag></item>
            <item>eleven<tag>out=11;</tag></item>
            <item>twelve<tag>out=12;</tag></item>
            <item>thirteen<tag>out=13;</tag></item>
            <item>fourteen<tag>out=14;</tag></item>
            <item>fifteen<tag>out=15;</tag></item>
            <item>sixteen<tag>out=16;</tag></item>
            <item>seventeen<tag>out=17;</tag></item>
            <item>eighteen<tag>out=18;</tag></item>
            <item>nineteen<tag>out=19;</tag></item>
```

```
                <item>tenth<tag>out=10;</tag></item>
                <item>eleventh<tag>out=11;</tag></item>
                <item>twelveth<tag>out=12;</tag></item>
                <item>thirteenth<tag>out=13;</tag></item>
                <item>fourteenth<tag>out=14;</tag></item>
                <item>fifteenth<tag>out=15;</tag></item>
                <item>sixteenth<tag>out=16;</tag></item>
                <item>seventeenth<tag>out=17;</tag></item>
                <item>eighteenth<tag>out=18;</tag></item>
                <item>nineteenth<tag>out=19;</tag></item>
            </one-of>
        </rule>

        <rule id="above_twenty">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <one-of>
                <item>twenty<tag>out=20;</tag></item>
                <item>thirty<tag>out=30;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>
</grammar>
```

轉賬金額

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: I want to transfer twenty five Dollar
                 Output: $25
```

```
             Scenario 2:
                 Input: transfer twenty five dollars
                 Output: $25

         -->

        <rule id="main" scope="public">
            <tag>out="$"</tag>
            <one-of>
                <item><ruleref uri="#sub_hundred"/><tag>out += rules.sub_hundred.sh;</
tag></item>
                <item><ruleref uri="#subThousands"/><tag>out += rules.subThousands;</
tag></item>
            </one-of>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
           <one-of>
              <item repeat="0-1">I want to transfer</item>
              <item repeat="0-1">transfer</item>
              <item repeat="0-1">make a transfer for</item>
           </one-of>
        </rule>

        <rule id="hesitation">
            <one-of>
               <item>Hmm</item>
               <item>Mmm</item>
               <item>My</item>
            </one-of>
          </rule>

        <rule id="digits">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.num = 0;</tag>
            <one-of>
                <item>0<tag>out.num+=0;</tag></item>
                <item>1<tag>out.num+=1;</tag></item>
                <item>2<tag>out.num+=2;</tag></item>
                <item>3<tag>out.num+=3;</tag></item>
                <item>4<tag>out.num+=4;</tag></item>
                <item>5<tag>out.num+=5;</tag></item>
```

```
            <item>6<tag>out.num+=6;</tag></item>
            <item>7<tag>out.num+=7;</tag></item>
            <item>8<tag>out.num+=8;</tag></item>
            <item>9<tag>out.num+=9;</tag></item>
            <item>one<tag>out.num+=1;</tag></item>
            <item>two<tag>out.num+=2;</tag></item>
            <item>three<tag>out.num+=3;</tag></item>
            <item>four<tag>out.num+=4;</tag></item>
            <item>five<tag>out.num+=5;</tag></item>
            <item>six<tag>out.num+=6;</tag></item>
            <item>seven<tag>out.num+=7;</tag></item>
            <item>eight<tag>out.num+=8;</tag></item>
            <item>nine<tag>out.num+=9;</tag></item>
        </one-of>
        <item repeat="0-1"><ruleref uri="#currency"/></item>
    </rule>

    <rule id="teens">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <tag>out.teen = 0;</tag>
        <one-of>
            <item>ten<tag>out.teen+=10;</tag></item>
            <item>eleven<tag>out.teen+=11;</tag></item>
            <item>twelve<tag>out.teen+=12;</tag></item>
            <item>thirteen<tag>out.teen+=13;</tag></item>
            <item>fourteen<tag>out.teen+=14;</tag></item>
            <item>fifteen<tag>out.teen+=15;</tag></item>
            <item>sixteen<tag>out.teen+=16;</tag></item>
            <item>seventeen<tag>out.teen+=17;</tag></item>
            <item>eighteen<tag>out.teen+=18;</tag></item>
            <item>nineteen<tag>out.teen+=19;</tag></item>
        </one-of>
        <item repeat="0-1"><ruleref uri="#currency"/></item>
    </rule>

    <rule id="above_twenty">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <tag>out.tens = 0;</tag>
        <one-of>
            <item>twenty<tag>out.tens+=20;</tag></item>
            <item>thirty<tag>out.tens+=30;</tag></item>
            <item>forty<tag>out.tens+=40;</tag></item>
            <item>fifty<tag>out.tens+=50;</tag></item>
            <item>sixty<tag>out.tens+=60;</tag></item>
```

```
                <item>seventy<tag>out.tens+=70;</tag></item>
                <item>eighty<tag>out.tens+=80;</tag></item>
                <item>ninety<tag>out.tens+=90;</tag></item>
                <item>hundred<tag>out.tens+=100;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#currency"/></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out.tens +=
 rules.digits.num;</tag></item>
        </rule>

        <rule id="currency">
            <one-of>
                <item repeat="0-1">dollars</item>
                <item repeat="0-1">Dollars</item>
                <item repeat="0-1">dollar</item>
                <item repeat="0-1">Dollar</item>
            </one-of>
        </rule>


        <rule id="sub_hundred">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.sh = 0;</tag>
            <one-of>
                <item><ruleref uri="#teens"/><tag>out.sh += rules.teens.teen;</tag></
item>
                <item>
                    <ruleref uri="#above_twenty"/><tag>out.sh +=
 rules.above_twenty.tens;</tag>
                </item>
                <item><ruleref uri="#digits"/><tag>out.sh += rules.digits.num;</tag></
item>
            </one-of>
         </rule>

        <rule id="subThousands">
            <ruleref uri="#sub_hundred"/><tag>out = (100 * rules.sub_hundred.sh);</tag>
            hundred
            <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out +=
 rules.above_twenty.tens;</tag></item>
            <item repeat="0-1"><ruleref uri="#teens"/><tag>out += rules.teens.teen;</
tag></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits.num;</
tag></item>
```

```
            <item repeat="0-1"><ruleref uri="#currency"/></item>
        </rule>
</grammar>
```

社會安全號碼

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <rule id="main" scope="public">
             <tag>out=""</tag>
             <ruleref uri="#digits"/><tag>out += rules.digits.numbers;</tag>
         </rule>

         <rule id="digits">
             <tag>out.numbers=""</tag>
             <item repeat="1-12">
                 <one-of>
                     <item>0<tag>out.numbers+=0;</tag></item>
                     <item>1<tag>out.numbers+=1;</tag></item>
                     <item>2<tag>out.numbers+=2;</tag></item>
                     <item>3<tag>out.numbers+=3;</tag></item>
                     <item>4<tag>out.numbers+=4;</tag></item>
                     <item>5<tag>out.numbers+=5;</tag></item>
                     <item>6<tag>out.numbers+=6;</tag></item>
                     <item>7<tag>out.numbers+=7;</tag></item>
                     <item>8<tag>out.numbers+=8;</tag></item>
                     <item>9<tag>out.numbers+=9;</tag></item>
                     <item>zero<tag>out.numbers+=0;</tag></item>
                     <item>one<tag>out.numbers+=1;</tag></item>
                     <item>two<tag>out.numbers+=2;</tag></item>
                     <item>three<tag>out.numbers+=3;</tag></item>
                     <item>four<tag>out.numbers+=4;</tag></item>
                     <item>five<tag>out.numbers+=5;</tag></item>
                     <item>six<tag>out.numbers+=6;</tag></item>
                     <item>seven<tag>out.numbers+=7;</tag></item>
```

```
                    <item>eight<tag>out.numbers+=8;</tag></item>
                    <item>nine<tag>out.numbers+=9;</tag></item>
                    <item>dash</item>
                </one-of>
            </item>
        </rule>
</grammar>
```

保險文法（下載）

保險域名支持以下語法：索賠和保單號碼，駕駛執照和車牌號碼，到期日期，開始日期和續期日期，索賠和保單金額。

申請識別碼

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="digits"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: My claim number is One Five Four Two
                 Output: 1542

             Scenario 2:
                 Input: Claim number One Five Four Four
                 Output: 1544

         -->

        <rule id="digits">
            <tag>out="";</tag>
            <item><ruleref uri="#singleDigit"/><tag>out += rules.singleDigit.digit;</
tag></item>
        </rule>
```

```
<rule id="text">
    <item repeat="0-1"><ruleref uri="#hesitation"/></item>
    <one-of>
        <item repeat="0-1">My claim number is</item>
        <item repeat="0-1">Claim number</item>
        <item repeat="0-1">This is for claim</item>
    </one-of>
</rule>

<rule id="hesitation">
    <one-of>
       <item>Hmm</item>
       <item>Mmm</item>
       <item>My</item>
    </one-of>
  </rule>

<rule id="singleDigit">
    <item repeat="0-1"><ruleref uri="#text"/></item>
    <tag>out.digit=""</tag>
    <item repeat="1-10">
        <one-of>
            <item>0<tag>out.digit+=0;</tag></item>
            <item>zero<tag>out.digit+=0;</tag></item>
            <item>1<tag>out.digit+=1;</tag></item>
            <item>one<tag>out.digit+=1;</tag></item>
            <item>2<tag>out.digit+=2;</tag></item>
            <item>two<tag>out.digit+=2;</tag></item>
            <item>3<tag>out.digit+=3;</tag></item>
            <item>three<tag>out.digit+=3;</tag></item>
            <item>4<tag>out.digit+=4;</tag></item>
            <item>four<tag>out.digit+=4;</tag></item>
            <item>5<tag>out.digit+=5;</tag></item>
            <item>five<tag>out.digit+=5;</tag></item>
            <item>6<tag>out.digit+=6;</tag></item>
            <item>six<tag>out.digit+=5;</tag></item>
            <item>7<tag>out.digit+=7;</tag></item>
            <item>seven<tag>out.digit+=7;</tag></item>
            <item>8<tag>out.digit+=8;</tag></item>
            <item>eight<tag>out.digit+=8;</tag></item>
            <item>9<tag>out.digit+=9;</tag></item>
            <item>nine<tag>out.digit+=9;</tag></item>
        </one-of>
```

```
            </item>
        </rule>
</grammar>
```

政策 ID

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">


        <!-- Test Cases

        Grammar will support the following inputs:

            Scenario 1:
                Input: My policy number is A B C 1 2 3 4
                Output: ABC1234

            Scenario 2:
                Input: This is the policy number 1 2 3 4 A B C
                Output: 1234ABC

            Scenario 3:
                Input: Hmm My policy number is 1 2 3 4 A B C 1
                Output: 123ABC1
        -->

        <rule id="main" scope="public">
            <tag>out=""</tag>
            <item><ruleref uri="#alphanumeric"/><tag>out +=
 rules.alphanumeric.alphanum;</tag></item>
            <item repeat="0-1"><ruleref uri="#alphabets"/><tag>out +=
 rules.alphabets.letters;</tag></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out +=
 rules.digits.numbers</tag></item>
            <item repeat="0-1"><ruleref uri="#thanks"/></item>
```

```
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
            <one-of>
                <item repeat="0-1">My policy number is</item>
                <item repeat="0-1">This is the policy number</item>
                <item repeat="0-1">Policy number</item>
                <item repeat="0-1">Yes, It is</item>
                <item repeat="0-1">Yes It is</item>
                <item repeat="0-1">Yes It's</item>
                <item repeat="0-1">My policy Id is</item>
                <item repeat="0-1">This is the policy Id</item>
                <item repeat="0-1">Policy Id</item>
            </one-of>
        </rule>

        <rule id="hesitation">
          <one-of>
             <item>Hmm</item>
             <item>Mmm</item>
             <item>My</item>
          </one-of>
        </rule>

        <rule id="thanks">
            <one-of>
               <item>Thanks</item>
               <item>I think</item>
            </one-of>
          </rule>

        <rule id="alphanumeric" scope="public">
            <tag>out.alphanum=""</tag>
            <item><ruleref uri="#alphabets"/><tag>out.alphanum +=
rules.alphabets.letters;</tag></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out.alphanum +=
rules.digits.numbers</tag></item>
        </rule>

        <rule id="alphabets">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.letters=""</tag>
            <tag>out.firstOccurence=""</tag>
```

```
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out.firstOccurence +=
rules.digits.numbers; out.letters += out.firstOccurence;</tag></item>
            <item repeat="1-">
                <one-of>
                    <item>A<tag>out.letters+='A';</tag></item>
                    <item>B<tag>out.letters+='B';</tag></item>
                    <item>C<tag>out.letters+='C';</tag></item>
                    <item>D<tag>out.letters+='D';</tag></item>
                    <item>E<tag>out.letters+='E';</tag></item>
                    <item>F<tag>out.letters+='F';</tag></item>
                    <item>G<tag>out.letters+='G';</tag></item>
                    <item>H<tag>out.letters+='H';</tag></item>
                    <item>I<tag>out.letters+='I';</tag></item>
                    <item>J<tag>out.letters+='J';</tag></item>
                    <item>K<tag>out.letters+='K';</tag></item>
                    <item>L<tag>out.letters+='L';</tag></item>
                    <item>M<tag>out.letters+='M';</tag></item>
                    <item>N<tag>out.letters+='N';</tag></item>
                    <item>O<tag>out.letters+='O';</tag></item>
                    <item>P<tag>out.letters+='P';</tag></item>
                    <item>Q<tag>out.letters+='Q';</tag></item>
                    <item>R<tag>out.letters+='R';</tag></item>
                    <item>S<tag>out.letters+='S';</tag></item>
                    <item>T<tag>out.letters+='T';</tag></item>
                    <item>U<tag>out.letters+='U';</tag></item>
                    <item>V<tag>out.letters+='V';</tag></item>
                    <item>W<tag>out.letters+='W';</tag></item>
                    <item>X<tag>out.letters+='X';</tag></item>
                    <item>Y<tag>out.letters+='Y';</tag></item>
                    <item>Z<tag>out.letters+='Z';</tag></item>
                </one-of>
            </item>
        </rule>

        <rule id="digits">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.numbers=""</tag>
            <item repeat="1-10">
                <one-of>
                    <item>0<tag>out.numbers+=0;</tag></item>
                    <item>1<tag>out.numbers+=1;</tag></item>
                    <item>2<tag>out.numbers+=2;</tag></item>
                    <item>3<tag>out.numbers+=3;</tag></item>
                    <item>4<tag>out.numbers+=4;</tag></item>
```

```
                    <item>5<tag>out.numbers+=5;</tag></item>
                    <item>6<tag>out.numbers+=6;</tag></item>
                    <item>7<tag>out.numbers+=7;</tag></item>
                    <item>8<tag>out.numbers+=8;</tag></item>
                    <item>9<tag>out.numbers+=9;</tag></item>
                </one-of>
            </item>
        </rule>
</grammar>
```

駕照號碼

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="digits"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: My drivers license number is One Five Four Two
                 Output: 1542

             Scenario 2:
                 Input: driver license number One Five Four Four
                 Output: 1544

         -->

        <rule id="digits">
            <tag>out=""</tag>
            <item><ruleref uri="#singleDigit"/><tag>out += rules.singleDigit.digit;</
tag></item>
        </rule>

        <rule id="text">
```

```
        <item repeat="0-1"><ruleref uri="#hesitation"/></item>
        <one-of>
            <item repeat="0-1">My drivers license number is</item>
            <item repeat="0-1">My drivers license id is</item>
            <item repeat="0-1">Driver license number</item>
        </one-of>
    </rule>

    <rule id="hesitation">
        <one-of>
           <item>Hmm</item>
           <item>Mmm</item>
           <item>My</item>
        </one-of>
      </rule>

    <rule id="singleDigit">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <tag>out.digit=""</tag>
        <item repeat="1-10">
            <one-of>
                <item>0<tag>out.digit+=0;</tag></item>
                <item>zero<tag>out.digit+=0;</tag></item>
                <item>1<tag>out.digit+=1;</tag></item>
                <item>one<tag>out.digit+=1;</tag></item>
                <item>2<tag>out.digit+=2;</tag></item>
                <item>two<tag>out.digit+=2;</tag></item>
                <item>3<tag>out.digit+=3;</tag></item>
                <item>three<tag>out.digit+=3;</tag></item>
                <item>4<tag>out.digit+=4;</tag></item>
                <item>four<tag>out.digit+=4;</tag></item>
                <item>5<tag>out.digit+=5;</tag></item>
                <item>five<tag>out.digit+=5;</tag></item>
                <item>6<tag>out.digit+=6;</tag></item>
                <item>six<tag>out.digit+=5;</tag></item>
                <item>7<tag>out.digit+=7;</tag></item>
                <item>seven<tag>out.digit+=7;</tag></item>
                <item>8<tag>out.digit+=8;</tag></item>
                <item>eight<tag>out.digit+=8;</tag></item>
                <item>9<tag>out.digit+=9;</tag></item>
                <item>nine<tag>out.digit+=9;</tag></item>
            </one-of>
        </item>
    </rule>
```

```
</grammar>
```

車牌號碼

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">


        <!-- Test Cases

        Grammar will support the following inputs:

            Scenario 1:
                Input: my license plate is A B C D 1 2
                Output: ABCD12

            Scenario 2:
                Input: license plate number A B C 1 2 3 4
                Output: ABC1234

            Scenario 3:
                Input: my plates say A F G K 9 8 7 6 Thanks
                Output: AFGK9876
        -->

        <rule id="main" scope="public">
            <tag>out.licenseNum=""</tag>
            <item><ruleref uri="#alphabets"/><tag>out.licenseNum +=
 rules.alphabets.letters;</tag></item>
            <item repeat="0-1"><ruleref uri="#thanks"/></item>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
            <one-of>
                <item repeat="0-1">my license plate is</item>
```

```
            <item repeat="0-1">license plate number</item>
            <item repeat="0-1">my plates say</item>
        </one-of>
    </rule>

    <rule id="hesitation">
      <one-of>
         <item>Hmm</item>
         <item>Mmm</item>
         <item>My</item>
      </one-of>
    </rule>

    <rule id="thanks">
        <one-of>
           <item>Thanks</item>
           <item>I think</item>
        </one-of>
      </rule>

    <rule id="alphabets">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <tag>out.letters=""</tag>
        <tag>out.firstOccurence=""</tag>
        <item repeat="3-4">
            <one-of>
                <item>A<tag>out.letters+='A';</tag></item>
                <item>B<tag>out.letters+='B';</tag></item>
                <item>C<tag>out.letters+='C';</tag></item>
                <item>D<tag>out.letters+='D';</tag></item>
                <item>E<tag>out.letters+='E';</tag></item>
                <item>F<tag>out.letters+='F';</tag></item>
                <item>G<tag>out.letters+='G';</tag></item>
                <item>H<tag>out.letters+='H';</tag></item>
                <item>I<tag>out.letters+='I';</tag></item>
                <item>J<tag>out.letters+='J';</tag></item>
                <item>K<tag>out.letters+='K';</tag></item>
                <item>L<tag>out.letters+='L';</tag></item>
                <item>M<tag>out.letters+='M';</tag></item>
                <item>N<tag>out.letters+='N';</tag></item>
                <item>O<tag>out.letters+='O';</tag></item>
                <item>P<tag>out.letters+='P';</tag></item>
                <item>Q<tag>out.letters+='Q';</tag></item>
                <item>R<tag>out.letters+='R';</tag></item>
```

```
                    <item>S<tag>out.letters+='S';</tag></item>
                    <item>T<tag>out.letters+='T';</tag></item>
                    <item>U<tag>out.letters+='U';</tag></item>
                    <item>V<tag>out.letters+='V';</tag></item>
                    <item>W<tag>out.letters+='W';</tag></item>
                    <item>X<tag>out.letters+='X';</tag></item>
                    <item>Y<tag>out.letters+='Y';</tag></item>
                    <item>Z<tag>out.letters+='Z';</tag></item>
                </one-of>
            </item>
    <item repeat="0-1"><ruleref uri="#digits"/><tag>out.firstOccurence +=
 rules.digits.numbers; out.letters += out.firstOccurence;</tag></item>
        </rule>

        <rule id="digits">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.numbers=""</tag>
            <item repeat="2-4">
                <one-of>
                    <item>0<tag>out.numbers+=0;</tag></item>
                    <item>1<tag>out.numbers+=1;</tag></item>
                    <item>2<tag>out.numbers+=2;</tag></item>
                    <item>3<tag>out.numbers+=3;</tag></item>
                    <item>4<tag>out.numbers+=4;</tag></item>
                    <item>5<tag>out.numbers+=5;</tag></item>
                    <item>6<tag>out.numbers+=6;</tag></item>
                    <item>7<tag>out.numbers+=7;</tag></item>
                    <item>8<tag>out.numbers+=8;</tag></item>
                    <item>9<tag>out.numbers+=9;</tag></item>
                </one-of>
            </item>
        </rule>
</grammar>
```

信用卡到期日

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="dateCardExpiration"
```

```
         mode="voice"
         tag-format="semantics/1.0">

        <rule id="dateCardExpiration" scope="public">
            <tag>out=""</tag>
            <item repeat="1"><ruleref uri="#months"/><tag>out = out + rules.months;</
tag></item>
            <item repeat="1"><ruleref uri="#year"/><tag>out += " " + rules.year.yr;</
tag></item>
            <item repeat="0-1"><ruleref uri="#thanks"/></item>
        </rule>

        <!-- Test Cases

        Grammar will support the following inputs:

            Scenario 1:
                Input: My card expiration date is july eleven
                Output: 07 2011

            Scenario 2:
                Input: My card expiration date is may twenty six
                Output: 05 2026

        -->

        <rule id="text">
           <item repeat="0-1"><ruleref uri="#hesitation"/></item>
           <one-of>
              <item repeat="0-1">My card expiration date is </item>
           </one-of>
        </rule>

        <rule id="hesitation">
          <one-of>
             <item>Hmm</item>
             <item>Mmm</item>
             <item>My</item>
          </one-of>
        </rule>

        <rule id="thanks">
            <one-of>
               <item>Thanks</item>
```

```
            <item>I think</item>
          </one-of>
        </rule>

      <rule id="months">
         <item repeat="0-1"><ruleref uri="#text"/></item>
         <one-of>
            <item>january<tag>out="01";</tag></item>
            <item>february<tag>out="02";</tag></item>
            <item>march<tag>out="03";</tag></item>
            <item>april<tag>out="04";</tag></item>
            <item>may<tag>out="05";</tag></item>
            <item>june<tag>out="06";</tag></item>
            <item>july<tag>out="07";</tag></item>
            <item>august<tag>out="08";</tag></item>
            <item>september<tag>out="09";</tag></item>
            <item>october<tag>out="10";</tag></item>
            <item>november<tag>out="11";</tag></item>
            <item>december<tag>out="12";</tag></item>
            <item>jan<tag>out="01";</tag></item>
            <item>feb<tag>out="02";</tag></item>
            <item>aug<tag>out="08";</tag></item>
            <item>sept<tag>out="09";</tag></item>
            <item>oct<tag>out="10";</tag></item>
            <item>nov<tag>out="11";</tag></item>
            <item>dec<tag>out="12";</tag></item>
            <item>1<tag>out="01";</tag></item>
            <item>2<tag>out="02";</tag></item>
            <item>3<tag>out="03";</tag></item>
            <item>4<tag>out="04";</tag></item>
            <item>5<tag>out="05";</tag></item>
            <item>6<tag>out="06";</tag></item>
            <item>7<tag>out="07";</tag></item>
            <item>8<tag>out="08";</tag></item>
            <item>9<tag>out="09";</tag></item>
            <item>ten<tag>out="10";</tag></item>
            <item>eleven<tag>out="11";</tag></item>
            <item>twelve<tag>out="12";</tag></item>
         </one-of>
      </rule>

      <rule id="digits">
          <item repeat="0-1"><ruleref uri="#text"/></item>
          <one-of>
```

```
                <item>0<tag>out=0;</tag></item>
                <item>1<tag>out=1;</tag></item>
                <item>2<tag>out=2;</tag></item>
                <item>3<tag>out=3;</tag></item>
                <item>4<tag>out=4;</tag></item>
                <item>5<tag>out=5;</tag></item>
                <item>6<tag>out=6;</tag></item>
                <item>7<tag>out=7;</tag></item>
                <item>8<tag>out=8;</tag></item>
                <item>9<tag>out=9;</tag></item>
                <item>one<tag>out=1;</tag></item>
                <item>two<tag>out=2;</tag></item>
                <item>three<tag>out=3;</tag></item>
                <item>four<tag>out=4;</tag></item>
                <item>five<tag>out=5;</tag></item>
                <item>six<tag>out=6;</tag></item>
                <item>seven<tag>out=7;</tag></item>
                <item>eight<tag>out=8;</tag></item>
                <item>nine<tag>out=9;</tag></item>
            </one-of>
        </rule>

        <rule id="year">
          <tag>out.yr="20"</tag>
          <one-of>
              <item><ruleref uri="#teens"/><tag>out.yr += rules.teens;</tag></item>
              <item><ruleref uri="#above_twenty"/><tag>out.yr += rules.above_twenty;</
tag></item>
          </one-of>
        </rule>

        <rule id="teens">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <one-of>
                <item>ten<tag>out=10;</tag></item>
                <item>eleven<tag>out=11;</tag></item>
                <item>twelve<tag>out=12;</tag></item>
                <item>thirteen<tag>out=13;</tag></item>
                <item>fourteen<tag>out=14;</tag></item>
                <item>fifteen<tag>out=15;</tag></item>
                <item>sixteen<tag>out=16;</tag></item>
                <item>seventeen<tag>out=17;</tag></item>
                <item>eighteen<tag>out=18;</tag></item>
                <item>nineteen<tag>out=19;</tag></item>
```

```
                <item>10<tag>out=10;</tag></item>
                <item>11<tag>out=11;</tag></item>
                <item>12<tag>out=12;</tag></item>
                <item>13<tag>out=13;</tag></item>
                <item>14<tag>out=14;</tag></item>
                <item>15<tag>out=15;</tag></item>
                <item>16<tag>out=16;</tag></item>
                <item>17<tag>out=17;</tag></item>
                <item>18<tag>out=18;</tag></item>
                <item>19<tag>out=19;</tag></item>
            </one-of>
        </rule>

        <rule id="above_twenty">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <one-of>
                <item>twenty<tag>out=20;</tag></item>
                <item>thirty<tag>out=30;</tag></item>
                <item>forty<tag>out=40;</tag></item>
                <item>fifty<tag>out=50;</tag></item>
                <item>sixty<tag>out=60;</tag></item>
                <item>seventy<tag>out=70;</tag></item>
                <item>eighty<tag>out=80;</tag></item>
                <item>ninety<tag>out=90;</tag></item>
                <item>20<tag>out=20;</tag></item>
                <item>30<tag>out=30;</tag></item>
                <item>40<tag>out=40;</tag></item>
                <item>50<tag>out=50;</tag></item>
                <item>60<tag>out=60;</tag></item>
                <item>70<tag>out=70;</tag></item>
                <item>80<tag>out=80;</tag></item>
                <item>90<tag>out=90;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>
</grammar>
```

保單到期日，日/月/年

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
        xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                            http://www.w3.org/TR/speech-grammar/grammar.xsd"
        xml:lang="en-US" version="1.0"
        root="main"
        mode="voice"
        tag-format="semantics/1.0">

        <!-- Test Cases

        Grammar will support the following inputs:

            Scenario 1:
                Input: My policy expired on July Five Two Thousand and Eleven
                Output: 07/5/11

            Scenario 2:
                Input: My policy will expire on July Sixteen Two Thousand and Eleven
                Output: 07/16/11

            Scenario 3:
                Input: My policy expired on July Thirty Two Thousand and Eleven
                Output: 07/30/11
        -->

        <rule id="main" scope="public">
            <tag>out=""</tag>
            <item>
                <item repeat="1"><ruleref uri="#months"/><tag>out = out +
 rules.months.mon + "/";</tag></item>
                <one-of>
                  <item><ruleref uri="#digits"/><tag>out += rules.digits + "/";</
tag></item>
                  <item><ruleref uri="#teens"/><tag>out += rules.teens+ "/";</tag></
item>
                  <item><ruleref uri="#above_twenty"/><tag>out += rules.above_twenty+
 "/";</tag></item>
                </one-of>
                <one-of>
                    <item><ruleref uri="#thousands"/><tag>out += rules.thousands;</
tag></item>
                    <item repeat="0-1"><ruleref uri="#digits"/><tag>out +=
 rules.digits;</tag></item>
                    <item repeat="0-1"><ruleref uri="#teens"/><tag>out +=
 rules.teens;</tag></item>
```

```
                    <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out +=
rules.above_twenty;</tag></item>
                </one-of>
            </item>
        </rule>

        <rule id="text">
          <item repeat="0-1"><ruleref uri="#hesitation"/></item>
           <one-of>
              <item repeat="0-1">My policy expired on</item>
              <item repeat="0-1">My policy will expire on</item>
           </one-of>
        </rule>

        <rule id="hesitation">
           <one-of>
              <item>Hmm</item>
              <item>Mmm</item>
              <item>My</item>
           </one-of>
         </rule>

        <rule id="months">
           <tag>out.mon=""</tag>
    <item repeat="0-1"><ruleref uri="#text"/></item>
           <one-of>
             <item>january<tag>out.mon+="01";</tag></item>
             <item>february<tag>out.mon+="02";</tag></item>
             <item>march<tag>out.mon+="03";</tag></item>
             <item>april<tag>out.mon+="04";</tag></item>
             <item>may<tag>out.mon+="05";</tag></item>
             <item>june<tag>out.mon+="06";</tag></item>
             <item>july<tag>out.mon+="07";</tag></item>
             <item>august<tag>out.mon+="08";</tag></item>
             <item>september<tag>out.mon+="09";</tag></item>
             <item>october<tag>out.mon+="10";</tag></item>
             <item>november<tag>out.mon+="11";</tag></item>
             <item>december<tag>out.mon+="12";</tag></item>
             <item>jan<tag>out.mon+="01";</tag></item>
             <item>feb<tag>out.mon+="02";</tag></item>
             <item>aug<tag>out.mon+="08";</tag></item>
             <item>sept<tag>out.mon+="09";</tag></item>
             <item>oct<tag>out.mon+="10";</tag></item>
             <item>nov<tag>out.mon+="11";</tag></item>
```

```
             <item>dec<tag>out.mon+="12";</tag></item>
           </one-of>
        </rule>

         <rule id="digits">
             <one-of>
                 <item>zero<tag>out=0;</tag></item>
                 <item>one<tag>out=1;</tag></item>
                 <item>two<tag>out=2;</tag></item>
                 <item>three<tag>out=3;</tag></item>
                 <item>four<tag>out=4;</tag></item>
                 <item>five<tag>out=5;</tag></item>
                 <item>six<tag>out=6;</tag></item>
                 <item>seven<tag>out=7;</tag></item>
                 <item>eight<tag>out=8;</tag></item>
                 <item>nine<tag>out=9;</tag></item>
             </one-of>
         </rule>

         <rule id="teens">
             <one-of>
                 <item>ten<tag>out=10;</tag></item>
                 <item>eleven<tag>out=11;</tag></item>
                 <item>twelve<tag>out=12;</tag></item>
                 <item>thirteen<tag>out=13;</tag></item>
                 <item>fourteen<tag>out=14;</tag></item>
                 <item>fifteen<tag>out=15;</tag></item>
                 <item>sixteen<tag>out=16;</tag></item>
                 <item>seventeen<tag>out=17;</tag></item>
                 <item>eighteen<tag>out=18;</tag></item>
                 <item>nineteen<tag>out=19;</tag></item>
             </one-of>
         </rule>

         <rule id="thousands">
             <item>two thousand</item>
             <item repeat="0-1">and</item>
             <item repeat="0-1"><ruleref uri="#digits"/><tag>out = rules.digits;</
tag></item>
             <item repeat="0-1"><ruleref uri="#teens"/><tag>out = rules.teens;</tag></
item>
             <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out =
 rules.above_twenty;</tag></item>
         </rule>
```

```
        <rule id="above_twenty">
            <one-of>
                <item>twenty<tag>out=20;</tag></item>
                <item>thirty<tag>out=30;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>
 </grammar>
```

保單續保日期、月份 / 年

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: I renewed my policy on July Two Thousand and Eleven
                 Output: 07/11

             Scenario 2:
                 Input: My policy will renew on July Two Thousand and Eleven
                 Output: 07/11

         -->

        <rule id="main" scope="public">
            <tag>out=""</tag>
            <item repeat="1-10">
                <item repeat="1"><ruleref uri="#months"/><tag>out = out +
 rules.months.mon + "/";</tag></item>
                <one-of>
```

```
                    <item><ruleref uri="#thousands"/><tag>out += rules.thousands;</
tag></item>
                    <item repeat="0-1"><ruleref uri="#digits"/><tag>out +=
 rules.digits;</tag></item>
                    <item repeat="0-1"><ruleref uri="#teens"/><tag>out +=
 rules.teens;</tag></item>
                    <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out +=
 rules.above_twenty;</tag></item>
                </one-of>
            </item>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
            <one-of>
                <item repeat="0-1">My policy will renew on</item>
                <item repeat="0-1">My policy was renewed on</item>
                <item repeat="0-1">Renew policy on</item>
                <item repeat="0-1">I renewed my policy on</item>
            </one-of>
        </rule>

        <rule id="hesitation">
            <one-of>
               <item>Hmm</item>
               <item>Mmm</item>
               <item>My</item>
            </one-of>
          </rule>

        <rule id="months">
           <item repeat="0-1"><ruleref uri="#text"/></item>
           <tag>out.mon=""</tag>
           <one-of>
               <item>january<tag>out.mon+="01";</tag></item>
               <item>february<tag>out.mon+="02";</tag></item>
               <item>march<tag>out.mon+="03";</tag></item>
               <item>april<tag>out.mon+="04";</tag></item>
               <item>may<tag>out.mon+="05";</tag></item>
               <item>june<tag>out.mon+="06";</tag></item>
               <item>july<tag>out.mon+="07";</tag></item>
               <item>august<tag>out.mon+="08";</tag></item>
               <item>september<tag>out.mon+="09";</tag></item>
               <item>october<tag>out.mon+="10";</tag></item>
```

```
            <item>november<tag>out.mon+="11";</tag></item>
            <item>december<tag>out.mon+="12";</tag></item>
            <item>jan<tag>out.mon+="01";</tag></item>
            <item>feb<tag>out.mon+="02";</tag></item>
            <item>aug<tag>out.mon+="08";</tag></item>
            <item>sept<tag>out.mon+="09";</tag></item>
            <item>oct<tag>out.mon+="10";</tag></item>
            <item>nov<tag>out.mon+="11";</tag></item>
            <item>dec<tag>out.mon+="12";</tag></item>
        </one-of>
    </rule>

    <rule id="digits">
        <one-of>
            <item>zero<tag>out=0;</tag></item>
            <item>one<tag>out=1;</tag></item>
            <item>two<tag>out=2;</tag></item>
            <item>three<tag>out=3;</tag></item>
            <item>four<tag>out=4;</tag></item>
            <item>five<tag>out=5;</tag></item>
            <item>six<tag>out=6;</tag></item>
            <item>seven<tag>out=7;</tag></item>
            <item>eight<tag>out=8;</tag></item>
            <item>nine<tag>out=9;</tag></item>
        </one-of>
    </rule>

    <rule id="teens">
        <one-of>
            <item>ten<tag>out=10;</tag></item>
            <item>eleven<tag>out=11;</tag></item>
            <item>twelve<tag>out=12;</tag></item>
            <item>thirteen<tag>out=13;</tag></item>
            <item>fourteen<tag>out=14;</tag></item>
            <item>fifteen<tag>out=15;</tag></item>
            <item>sixteen<tag>out=16;</tag></item>
            <item>seventeen<tag>out=17;</tag></item>
            <item>eighteen<tag>out=18;</tag></item>
            <item>nineteen<tag>out=19;</tag></item>
        </one-of>
    </rule>

    <rule id="thousands">
        <item>two thousand<!--<tag>out=2000;</tag>--></item>
```

```
            <item repeat="0-1">and</item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out = rules.digits;</tag></
item>
            <item repeat="0-1"><ruleref uri="#teens"/><tag>out = rules.teens;</tag></
item>
            <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out =
 rules.above_twenty;</tag></item>
        </rule>

        <rule id="above_twenty">
            <one-of>
                <item>twenty<tag>out=20;</tag></item>
                <item>thirty<tag>out=30;</tag></item>
                <item>forty<tag>out=40;</tag></item>
                <item>fifty<tag>out=50;</tag></item>
                <item>sixty<tag>out=60;</tag></item>
                <item>seventy<tag>out=70;</tag></item>
                <item>eighty<tag>out=80;</tag></item>
                <item>ninety<tag>out=90;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>
</grammar>
```

保單開始日期

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: I bought my policy on july twenty three
```

```
                Output: 07/23

            Scenario 2:
                Input: My policy started on july fifteen
                Output: 07/15

         -->

        <rule id="main" scope="public">
            <tag>out=""</tag>
            <item repeat="1-10">
                <item><ruleref uri="#months"/><tag>out= rules.months.mon + "/";</tag></
item>
                <one-of>
                    <item><ruleref uri="#digits"/><tag>out+= rules.digits;</tag></item>
                    <item><ruleref uri="#teens"/><tag>out+= rules.teens;</tag></item>
                    <item><ruleref uri="#above_twenty"/><tag>out+=
 rules.above_twenty;</tag></item>
                </one-of>
            </item>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
             <one-of>
                <item repeat="0-1">I bought my policy on</item>
                <item repeat="0-1">I bought policy on</item>
                <item repeat="0-1">My policy started on</item>
             </one-of>
          </rule>

          <rule id="hesitation">
             <one-of>
                <item>Hmm</item>
                <item>Mmm</item>
                <item>My</item>
             </one-of>
           </rule>

        <rule id="months">
           <item repeat="0-1"><ruleref uri="#text"/></item>
           <tag>out.mon=""</tag>
           <one-of>
               <item>january<tag>out.mon+="01";</tag></item>
```

```
            <item>february<tag>out.mon+="02";</tag></item>
            <item>march<tag>out.mon+="03";</tag></item>
            <item>april<tag>out.mon+="04";</tag></item>
            <item>may<tag>out.mon+="05";</tag></item>
            <item>june<tag>out.mon+="06";</tag></item>
            <item>july<tag>out.mon+="07";</tag></item>
            <item>august<tag>out.mon+="08";</tag></item>
            <item>september<tag>out.mon+="09";</tag></item>
            <item>october<tag>out.mon+="10";</tag></item>
            <item>november<tag>out.mon+="11";</tag></item>
            <item>december<tag>out.mon+="12";</tag></item>
            <item>jan<tag>out.mon+="01";</tag></item>
            <item>feb<tag>out.mon+="02";</tag></item>
            <item>aug<tag>out.mon+="08";</tag></item>
            <item>sept<tag>out.mon+="09";</tag></item>
            <item>oct<tag>out.mon+="10";</tag></item>
            <item>nov<tag>out.mon+="11";</tag></item>
            <item>dec<tag>out.mon+="12";</tag></item>
        </one-of>
    </rule>

    <rule id="digits">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <one-of>
            <item>0<tag>out=0;</tag></item>
            <item>1<tag>out=1;</tag></item>
            <item>2<tag>out=2;</tag></item>
            <item>3<tag>out=3;</tag></item>
            <item>4<tag>out=4;</tag></item>
            <item>5<tag>out=5;</tag></item>
            <item>6<tag>out=6;</tag></item>
            <item>7<tag>out=7;</tag></item>
            <item>8<tag>out=8;</tag></item>
            <item>9<tag>out=9;</tag></item>
            <item>first<tag>out=01;</tag></item>
            <item>second<tag>out=02;</tag></item>
            <item>third<tag>out=03;</tag></item>
            <item>fourth<tag>out=04;</tag></item>
            <item>fifth<tag>out=05;</tag></item>
            <item>sixth<tag>out=06;</tag></item>
            <item>seventh<tag>out=07;</tag></item>
            <item>eighth<tag>out=08;</tag></item>
            <item>ninth<tag>out=09;</tag></item>
            <item>one<tag>out=1;</tag></item>
```

```
            <item>two<tag>out=2;</tag></item>
            <item>three<tag>out=3;</tag></item>
            <item>four<tag>out=4;</tag></item>
            <item>five<tag>out=5;</tag></item>
            <item>six<tag>out=6;</tag></item>
            <item>seven<tag>out=7;</tag></item>
            <item>eight<tag>out=8;</tag></item>
            <item>nine<tag>out=9;</tag></item>
        </one-of>
    </rule>

    <rule id="teens">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <one-of>
            <item>ten<tag>out=10;</tag></item>
            <item>tenth<tag>out=10;</tag></item>
            <item>eleven<tag>out=11;</tag></item>
            <item>twelve<tag>out=12;</tag></item>
            <item>thirteen<tag>out=13;</tag></item>
            <item>fourteen<tag>out=14;</tag></item>
            <item>fifteen<tag>out=15;</tag></item>
            <item>sixteen<tag>out=16;</tag></item>
            <item>seventeen<tag>out=17;</tag></item>
            <item>eighteen<tag>out=18;</tag></item>
            <item>nineteen<tag>out=19;</tag></item>
            <item>tenth<tag>out=10;</tag></item>
            <item>eleventh<tag>out=11;</tag></item>
            <item>twelveth<tag>out=12;</tag></item>
            <item>thirteenth<tag>out=13;</tag></item>
            <item>fourteenth<tag>out=14;</tag></item>
            <item>fifteenth<tag>out=15;</tag></item>
            <item>sixteenth<tag>out=16;</tag></item>
            <item>seventeenth<tag>out=17;</tag></item>
            <item>eighteenth<tag>out=18;</tag></item>
            <item>nineteenth<tag>out=19;</tag></item>
        </one-of>
    </rule>

    <rule id="above_twenty">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <one-of>
            <item>twenty<tag>out=20;</tag></item>
            <item>thirty<tag>out=30;</tag></item>
        </one-of>
```

```
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>
</grammar>
```

索償金額

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: I want to make a claim of one hundre ten dollars
                 Output: $110

             Scenario 2:
                 Input: Requesting claim of Two hundred dollars
                 Output: $200

         -->

        <rule id="main" scope="public">
            <tag>out="$"</tag>
            <one-of>
                <item><ruleref uri="#sub_hundred"/><tag>out += rules.sub_hundred.sh;</
tag></item>
                <item><ruleref uri="#subThousands"/><tag>out += rules.subThousands;</
tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#thanks"/></item>
        </rule>

        <rule id="text">
```

```
    <item repeat="0-1"><ruleref uri="#hesitation"/></item>
    <one-of>
       <item repeat="0-1">I want to place a claim for</item>
       <item repeat="0-1">I want to make a claim of</item>
       <item repeat="0-1">I assess damage of</item>
       <item repeat="0-1">Requesting claim of</item>
    </one-of>
</rule>

<rule id="hesitation">
    <one-of>
       <item>Hmm</item>
       <item>Mmm</item>
       <item>My</item>
    </one-of>
  </rule>

  <rule id="thanks">
      <one-of>
         <item>Thanks</item>
         <item>I think</item>
      </one-of>
    </rule>

<rule id="digits">
    <item repeat="0-1"><ruleref uri="#text"/></item>
    <tag>out.num = 0;</tag>
    <one-of>
        <item>0<tag>out.num+=0;</tag></item>
        <item>1<tag>out.num+=1;</tag></item>
        <item>2<tag>out.num+=2;</tag></item>
        <item>3<tag>out.num+=3;</tag></item>
        <item>4<tag>out.num+=4;</tag></item>
        <item>5<tag>out.num+=5;</tag></item>
        <item>6<tag>out.num+=6;</tag></item>
        <item>7<tag>out.num+=7;</tag></item>
        <item>8<tag>out.num+=8;</tag></item>
        <item>9<tag>out.num+=9;</tag></item>
        <item>one<tag>out.num+=1;</tag></item>
        <item>two<tag>out.num+=2;</tag></item>
        <item>three<tag>out.num+=3;</tag></item>
        <item>four<tag>out.num+=4;</tag></item>
        <item>five<tag>out.num+=5;</tag></item>
        <item>six<tag>out.num+=6;</tag></item>
```

```
                <item>seven<tag>out.num+=7;</tag></item>
                <item>eight<tag>out.num+=8;</tag></item>
                <item>nine<tag>out.num+=9;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#currency"/></item>
        </rule>

        <rule id="teens">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.teen = 0;</tag>
            <one-of>
                <item>ten<tag>out.teen+=10;</tag></item>
                <item>eleven<tag>out.teen+=11;</tag></item>
                <item>twelve<tag>out.teen+=12;</tag></item>
                <item>thirteen<tag>out.teen+=13;</tag></item>
                <item>fourteen<tag>out.teen+=14;</tag></item>
                <item>fifteen<tag>out.teen+=15;</tag></item>
                <item>sixteen<tag>out.teen+=16;</tag></item>
                <item>seventeen<tag>out.teen+=17;</tag></item>
                <item>eighteen<tag>out.teen+=18;</tag></item>
                <item>nineteen<tag>out.teen+=19;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#currency"/></item>
        </rule>

        <rule id="above_twenty">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.tens = 0;</tag>
            <one-of>
                <item>twenty<tag>out.tens+=20;</tag></item>
                <item>thirty<tag>out.tens+=30;</tag></item>
                <item>forty<tag>out.tens+=40;</tag></item>
                <item>fifty<tag>out.tens+=50;</tag></item>
                <item>sixty<tag>out.tens+=60;</tag></item>
                <item>seventy<tag>out.tens+=70;</tag></item>
                <item>eighty<tag>out.tens+=80;</tag></item>
                <item>ninety<tag>out.tens+=90;</tag></item>
                <item>hundred<tag>out.tens+=100;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#currency"/></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out.tens +=
rules.digits.num;</tag></item>
        </rule>
```

```
        <rule id="currency">
            <one-of>
                <item repeat="0-1">dollars</item>
                <item repeat="0-1">Dollars</item>
                <item repeat="0-1">dollar</item>
                <item repeat="0-1">Dollar</item>
            </one-of>
        </rule>


        <rule id="sub_hundred">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.sh = 0;</tag>
            <one-of>
                <item><ruleref uri="#teens"/><tag>out.sh += rules.teens.teen;</tag></
item>
                <item>
                    <ruleref uri="#above_twenty"/><tag>out.sh +=
 rules.above_twenty.tens;</tag>
                </item>
                <item><ruleref uri="#digits"/><tag>out.sh += rules.digits.num;</tag></
item>
            </one-of>
         </rule>

        <rule id="subThousands">
            <ruleref uri="#sub_hundred"/><tag>out = (100 * rules.sub_hundred.sh);</tag>
            hundred
            <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out +=
 rules.above_twenty.tens;</tag></item>
            <item repeat="0-1"><ruleref uri="#teens"/><tag>out += rules.teens.teen;</
tag></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits.num;</
tag></item>
            <item repeat="0-1"><ruleref uri="#currency"/></item>
        </rule>
</grammar>
```

保費金額

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Premium amounts
             Scenario 1:
                 Input: The premium for one hundre ten dollars
                 Output: $110

             Scenario 2:
                 Input: RPremium amount of Two hundred dollars
                 Output: $200

         -->

        <rule id="main" scope="public">
            <tag>out="$"</tag>
            <one-of>
                <item><ruleref uri="#sub_hundred"/><tag>out += rules.sub_hundred.sh;</
tag></item>
                <item><ruleref uri="#subThousands"/><tag>out += rules.subThousands;</
tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#thanks"/></item>
        </rule>

        <rule id="text">
           <item repeat="0-1"><ruleref uri="#hesitation"/></item>
           <one-of>
              <item repeat="0-1">A premium of</item>
              <item repeat="0-1">Premium amount of</item>
              <item repeat="0-1">The premium for</item>
              <item repeat="0-1">Insurance premium for</item>
           </one-of>
        </rule>

        <rule id="hesitation">
```

```
        <one-of>
           <item>Hmm</item>
           <item>Mmm</item>
           <item>My</item>
        </one-of>
      </rule>

      <rule id="thanks">
          <one-of>
             <item>Thanks</item>
             <item>I think</item>
          </one-of>
        </rule>

    <rule id="digits">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <tag>out.num = 0;</tag>
        <one-of>
            <item>0<tag>out.num+=0;</tag></item>
            <item>1<tag>out.num+=1;</tag></item>
            <item>2<tag>out.num+=2;</tag></item>
            <item>3<tag>out.num+=3;</tag></item>
            <item>4<tag>out.num+=4;</tag></item>
            <item>5<tag>out.num+=5;</tag></item>
            <item>6<tag>out.num+=6;</tag></item>
            <item>7<tag>out.num+=7;</tag></item>
            <item>8<tag>out.num+=8;</tag></item>
            <item>9<tag>out.num+=9;</tag></item>
            <item>one<tag>out.num+=1;</tag></item>
            <item>two<tag>out.num+=2;</tag></item>
            <item>three<tag>out.num+=3;</tag></item>
            <item>four<tag>out.num+=4;</tag></item>
            <item>five<tag>out.num+=5;</tag></item>
            <item>six<tag>out.num+=6;</tag></item>
            <item>seven<tag>out.num+=7;</tag></item>
            <item>eight<tag>out.num+=8;</tag></item>
            <item>nine<tag>out.num+=9;</tag></item>
        </one-of>
        <item repeat="0-1"><ruleref uri="#currency"/></item>
    </rule>

    <rule id="teens">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <tag>out.teen = 0;</tag>
```

```
            <one-of>
                <item>ten<tag>out.teen+=10;</tag></item>
                <item>eleven<tag>out.teen+=11;</tag></item>
                <item>twelve<tag>out.teen+=12;</tag></item>
                <item>thirteen<tag>out.teen+=13;</tag></item>
                <item>fourteen<tag>out.teen+=14;</tag></item>
                <item>fifteen<tag>out.teen+=15;</tag></item>
                <item>sixteen<tag>out.teen+=16;</tag></item>
                <item>seventeen<tag>out.teen+=17;</tag></item>
                <item>eighteen<tag>out.teen+=18;</tag></item>
                <item>nineteen<tag>out.teen+=19;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#currency"/></item>
        </rule>

        <rule id="above_twenty">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.tens = 0;</tag>
            <one-of>
                <item>twenty<tag>out.tens+=20;</tag></item>
                <item>thirty<tag>out.tens+=30;</tag></item>
                <item>forty<tag>out.tens+=40;</tag></item>
                <item>fifty<tag>out.tens+=50;</tag></item>
                <item>sixty<tag>out.tens+=60;</tag></item>
                <item>seventy<tag>out.tens+=70;</tag></item>
                <item>eighty<tag>out.tens+=80;</tag></item>
                <item>ninety<tag>out.tens+=90;</tag></item>
                <item>hundred<tag>out.tens+=100;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#currency"/></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out.tens +=
rules.digits.num;</tag></item>
        </rule>

        <rule id="currency">
            <one-of>
                <item repeat="0-1">dollars</item>
                <item repeat="0-1">Dollars</item>
                <item repeat="0-1">dollar</item>
                <item repeat="0-1">Dollar</item>
            </one-of>
        </rule>
```

```
        <rule id="sub_hundred">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.sh = 0;</tag>
            <one-of>
                <item><ruleref uri="#teens"/><tag>out.sh += rules.teens.teen;</tag></
item>
                <item>
                    <ruleref uri="#above_twenty"/><tag>out.sh +=
 rules.above_twenty.tens;</tag>
                </item>
                <item><ruleref uri="#digits"/><tag>out.sh += rules.digits.num;</tag></
item>
            </one-of>
         </rule>

        <rule id="subThousands">
            <ruleref uri="#sub_hundred"/><tag>out = (100 * rules.sub_hundred.sh);</tag>
            hundred
            <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out +=
 rules.above_twenty.tens;</tag></item>
            <item repeat="0-1"><ruleref uri="#teens"/><tag>out += rules.teens.teen;</
tag></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits.num;</
tag></item>
            <item repeat="0-1"><ruleref uri="#currency"/></item>
        </rule>
</grammar>
```

政策數量

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:
```

```
             Scenario 1:
                 Input: The number is one
                 Output: 1

             Scenario 2:
                 Input: I want policy for ten
                 Output: 10

         -->

        <rule id="main" scope="public">
            <tag>out=""</tag>
            <one-of>
              <item repeat="1"><ruleref uri="#digits"/><tag>out+= rules.digits;</tag></
item>
              <item repeat="1"><ruleref uri="#teens"/><tag>out+= rules.teens;</tag></
item>
              <item repeat="1"><ruleref uri="#above_twenty"/><tag>out+=
 rules.above_twenty;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#thanks"/></item>
        </rule>

        <rule id="text">
           <one-of>
              <item repeat="0-1">I want policy for</item>
              <item repeat="0-1">I want to order policy for</item>
              <item repeat="0-1">The number is</item>
           </one-of>
        </rule>

        <rule id="thanks">
            <one-of>
               <item>Thanks</item>
               <item>I think</item>
            </one-of>
          </rule>

        <rule id="digits">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <one-of>
                <item>0<tag>out=0;</tag></item>
                <item>1<tag>out=1;</tag></item>
```

```
            <item>2<tag>out=2;</tag></item>
            <item>3<tag>out=3;</tag></item>
            <item>4<tag>out=4;</tag></item>
            <item>5<tag>out=5;</tag></item>
            <item>6<tag>out=6;</tag></item>
            <item>7<tag>out=7;</tag></item>
            <item>8<tag>out=8;</tag></item>
            <item>9<tag>out=9;</tag></item>
            <item>one<tag>out=1;</tag></item>
            <item>two<tag>out=2;</tag></item>
            <item>three<tag>out=3;</tag></item>
            <item>four<tag>out=4;</tag></item>
            <item>five<tag>out=5;</tag></item>
            <item>six<tag>out=6;</tag></item>
            <item>seven<tag>out=7;</tag></item>
            <item>eight<tag>out=8;</tag></item>
            <item>nine<tag>out=9;</tag></item>
        </one-of>
    </rule>

    <rule id="teens">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <one-of>
            <item>ten<tag>out=10;</tag></item>
            <item>eleven<tag>out=11;</tag></item>
            <item>twelve<tag>out=12;</tag></item>
            <item>thirteen<tag>out=13;</tag></item>
            <item>fourteen<tag>out=14;</tag></item>
            <item>fifteen<tag>out=15;</tag></item>
            <item>sixteen<tag>out=16;</tag></item>
            <item>seventeen<tag>out=17;</tag></item>
            <item>eighteen<tag>out=18;</tag></item>
            <item>nineteen<tag>out=19;</tag></item>
            <item>10<tag>out=10;</tag></item>
            <item>11<tag>out=11;</tag></item>
            <item>12<tag>out=12;</tag></item>
            <item>13<tag>out=13;</tag></item>
            <item>14<tag>out=14;</tag></item>
            <item>15<tag>out=15;</tag></item>
            <item>16<tag>out=16;</tag></item>
            <item>17<tag>out=17;</tag></item>
            <item>18<tag>out=18;</tag></item>
            <item>19<tag>out=19;</tag></item>
        </one-of>
```

```
        </rule>

        <rule id="above_twenty">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <one-of>
                <item>twenty<tag>out=20;</tag></item>
                <item>thirty<tag>out=30;</tag></item>
                <item>forty<tag>out=40;</tag></item>
                <item>fifty<tag>out=50;</tag></item>
                <item>sixty<tag>out=60;</tag></item>
                <item>seventy<tag>out=70;</tag></item>
                <item>eighty<tag>out=80;</tag></item>
                <item>ninety<tag>out=90;</tag></item>
                <item>20<tag>out=20;</tag></item>
                <item>30<tag>out=30;</tag></item>
                <item>40<tag>out=40;</tag></item>
                <item>50<tag>out=50;</tag></item>
                <item>60<tag>out=60;</tag></item>
                <item>70<tag>out=70;</tag></item>
                <item>80<tag>out=80;</tag></item>
                <item>90<tag>out=90;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>

</grammar>
```

電信語法（[下載](#)）

電信支持以下語法：電話號碼，序列號，SIM 卡號碼，美國郵政編碼，信用卡到期日期，計劃開始，續訂和到期日期，服務開始日期，設備數量和賬單金額。

電話號碼

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="digits"
         mode="voice"
         tag-format="semantics/1.0">
```

```
        <!-- Test Cases

        Grammar will support 10-12 digits number and here are couple of examples of
 valid inputs:

            Scenario 1:
                Input: Mmm My phone number is two zero one two five two six seven
 eight five
                Output: 2012526785

            Scenario 2:
                Input: My phone number is two zero one two five two six seven eight
 five
                Output: 2012526785

        -->

       <rule id="digits">
           <tag>out=""</tag>
           <item><ruleref uri="#singleDigit"/><tag>out += rules.singleDigit.digit;</
tag></item>
       </rule>

       <rule id="text">
           <item repeat="0-1"><ruleref uri="#hesitation"/></item>
          <one-of>
             <item repeat="0-1">My phone number is</item>
             <item repeat="0-1">Phone number is</item>
             <item repeat="0-1">It is</item>
             <item repeat="0-1">Yes, it's</item>
             <item repeat="0-1">Yes, it is</item>
             <item repeat="0-1">Yes it is</item>
          </one-of>
       </rule>

       <rule id="hesitation">
           <one-of>
              <item>Hmm</item>
              <item>Mmm</item>
              <item>My</item>
           </one-of>
         </rule>
```

```
        <rule id="singleDigit">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.digit=""</tag>
            <item repeat="10-12">
                <one-of>
                    <item>0<tag>out.digit+=0;</tag></item>
                    <item>zero<tag>out.digit+=0;</tag></item>
                    <item>1<tag>out.digit+=1;</tag></item>
                    <item>one<tag>out.digit+=1;</tag></item>
                    <item>2<tag>out.digit+=2;</tag></item>
                    <item>two<tag>out.digit+=2;</tag></item>
                    <item>3<tag>out.digit+=3;</tag></item>
                    <item>three<tag>out.digit+=3;</tag></item>
                    <item>4<tag>out.digit+=4;</tag></item>
                    <item>four<tag>out.digit+=4;</tag></item>
                    <item>5<tag>out.digit+=5;</tag></item>
                    <item>five<tag>out.digit+=5;</tag></item>
                    <item>6<tag>out.digit+=6;</tag></item>
                    <item>six<tag>out.digit+=5;</tag></item>
                    <item>7<tag>out.digit+=7;</tag></item>
                    <item>seven<tag>out.digit+=7;</tag></item>
                    <item>8<tag>out.digit+=8;</tag></item>
                    <item>eight<tag>out.digit+=8;</tag></item>
                    <item>9<tag>out.digit+=9;</tag></item>
                    <item>nine<tag>out.digit+=9;</tag></item>
                </one-of>
            </item>
        </rule>
</grammar>
```

序列號

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="digits"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases
```

```
         Grammar will support the following inputs:

             Scenario 1:
                 Input: My serial number is 1 2 3 4 5 6 7 8 9 1 2 3 4 5 6
                 Output: 123456789123456

             Scenario 2:
                 Input: Device Serial number 1 2 3 4 5 6 7 8 9 1 2 3 4 5 6
                 Output: 123456789123456

         -->

        <rule id="digits">
            <tag>out=""</tag>
            <item><ruleref uri="#singleDigit"/><tag>out += rules.singleDigit.digit;</
tag></item>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
            <one-of>
                <item repeat="0-1">My serial number is</item>
                <item repeat="0-1">Device Serial number</item>
                <item repeat="0-1">The number is</item>
                <item repeat="0-1">The IMEI number is</item>
            </one-of>
        </rule>

        <rule id="hesitation">
            <one-of>
               <item>Hmm</item>
               <item>Mmm</item>
               <item>My</item>
            </one-of>
          </rule>

        <rule id="singleDigit">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.digit=""</tag>
            <item repeat="15">
                <one-of>
                    <item>0<tag>out.digit+=0;</tag></item>
                    <item>zero<tag>out.digit+=0;</tag></item>
```

```
                    <item>1<tag>out.digit+=1;</tag></item>
                    <item>one<tag>out.digit+=1;</tag></item>
                    <item>2<tag>out.digit+=2;</tag></item>
                    <item>two<tag>out.digit+=2;</tag></item>
                    <item>3<tag>out.digit+=3;</tag></item>
                    <item>three<tag>out.digit+=3;</tag></item>
                    <item>4<tag>out.digit+=4;</tag></item>
                    <item>four<tag>out.digit+=4;</tag></item>
                    <item>5<tag>out.digit+=5;</tag></item>
                    <item>five<tag>out.digit+=5;</tag></item>
                    <item>6<tag>out.digit+=6;</tag></item>
                    <item>six<tag>out.digit+=5;</tag></item>
                    <item>7<tag>out.digit+=7;</tag></item>
                    <item>seven<tag>out.digit+=7;</tag></item>
                    <item>8<tag>out.digit+=8;</tag></item>
                    <item>eight<tag>out.digit+=8;</tag></item>
                    <item>9<tag>out.digit+=9;</tag></item>
                    <item>nine<tag>out.digit+=9;</tag></item>
                </one-of>
            </item>
        </rule>
</grammar>
```

SIM 卡號碼

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">


        <!-- Test Cases

        Grammar will support the following inputs:

            Scenario 1:
                Input: My SIM number is A B C 1 2 3 4
                Output: ABC1234
```

```
            Scenario 2:
                Input: My SIM number is 1 2 3 4 A B C
                Output: 1234ABC

            Scenario 3:
                Input: My SIM number is 1 2 3 4 A B C 1
                Output: 123ABC1
        -->

        <rule id="main" scope="public">
            <tag>out=""</tag>
            <item><ruleref uri="#alphanumeric"/><tag>out +=
rules.alphanumeric.alphanum;</tag></item>
            <item repeat="0-1"><ruleref uri="#alphabets"/><tag>out +=
rules.alphabets.letters;</tag></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out +=
rules.digits.numbers</tag></item>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
            <one-of>
                <item repeat="0-1">My SIM number is</item>
                <item repeat="0-1">SIM number is</item>
            </one-of>
        </rule>

        <rule id="hesitation">
          <one-of>
             <item>Hmm</item>
             <item>Mmm</item>
             <item>My</item>
          </one-of>
        </rule>

        <rule id="alphanumeric" scope="public">
            <tag>out.alphanum=""</tag>
            <item><ruleref uri="#alphabets"/><tag>out.alphanum +=
rules.alphabets.letters;</tag></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out.alphanum +=
rules.digits.numbers</tag></item>
        </rule>
```

```
        <rule id="alphabets">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.letters=""</tag>
            <tag>out.firstOccurence=""</tag>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out.firstOccurence +=
rules.digits.numbers; out.letters += out.firstOccurence;</tag></item>
            <item repeat="1-">
                <one-of>
                    <item>A<tag>out.letters+='A';</tag></item>
                    <item>B<tag>out.letters+='B';</tag></item>
                    <item>C<tag>out.letters+='C';</tag></item>
                    <item>D<tag>out.letters+='D';</tag></item>
                    <item>E<tag>out.letters+='E';</tag></item>
                    <item>F<tag>out.letters+='F';</tag></item>
                    <item>G<tag>out.letters+='G';</tag></item>
                    <item>H<tag>out.letters+='H';</tag></item>
                    <item>I<tag>out.letters+='I';</tag></item>
                    <item>J<tag>out.letters+='J';</tag></item>
                    <item>K<tag>out.letters+='K';</tag></item>
                    <item>L<tag>out.letters+='L';</tag></item>
                    <item>M<tag>out.letters+='M';</tag></item>
                    <item>N<tag>out.letters+='N';</tag></item>
                    <item>O<tag>out.letters+='O';</tag></item>
                    <item>P<tag>out.letters+='P';</tag></item>
                    <item>Q<tag>out.letters+='Q';</tag></item>
                    <item>R<tag>out.letters+='R';</tag></item>
                    <item>S<tag>out.letters+='S';</tag></item>
                    <item>T<tag>out.letters+='T';</tag></item>
                    <item>U<tag>out.letters+='U';</tag></item>
                    <item>V<tag>out.letters+='V';</tag></item>
                    <item>W<tag>out.letters+='W';</tag></item>
                    <item>X<tag>out.letters+='X';</tag></item>
                    <item>Y<tag>out.letters+='Y';</tag></item>
                    <item>Z<tag>out.letters+='Z';</tag></item>
                </one-of>
            </item>
        </rule>

        <rule id="digits">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.numbers=""</tag>
            <item repeat="1-10">
                <one-of>
                    <item>0<tag>out.numbers+=0;</tag></item>
```

```
                    <item>1<tag>out.numbers+=1;</tag></item>
                    <item>2<tag>out.numbers+=2;</tag></item>
                    <item>3<tag>out.numbers+=3;</tag></item>
                    <item>4<tag>out.numbers+=4;</tag></item>
                    <item>5<tag>out.numbers+=5;</tag></item>
                    <item>6<tag>out.numbers+=6;</tag></item>
                    <item>7<tag>out.numbers+=7;</tag></item>
                    <item>8<tag>out.numbers+=8;</tag></item>
                    <item>9<tag>out.numbers+=9;</tag></item>
                </one-of>
            </item>
        </rule>
</grammar>
```

美國郵政編碼

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="digits"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support 5 digits code and here are couple of examples of valid
 inputs:

             Scenario 1:
                 Input: Mmmm My zipcode is umm One Oh Nine Eight Seven
                 Output: 10987

             Scenario 2:
                 Input: My zipcode is One Oh Nine Eight Seven
                 Output: 10987

         -->

        <rule id="digits">
            <tag>out=""</tag>
```

```
            <item><ruleref uri="#singleDigit"/><tag>out += rules.singleDigit.digit;</
tag></item>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
           <one-of>
              <item repeat="0-1">My zipcode is</item>
              <item repeat="0-1">Zipcode is</item>
              <item repeat="0-1">It is</item>
           </one-of>
        </rule>

        <rule id="hesitation">
            <one-of>
               <item>Hmm</item>
               <item>Mmm</item>
               <item>My</item>
            </one-of>
          </rule>

        <rule id="singleDigit">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.digit=""</tag>
            <item repeat="5">
                <one-of>
                    <item>0<tag>out.digit+=0;</tag></item>
                    <item>zero<tag>out.digit+=0;</tag></item>
                    <item>Oh<tag>out.digit+=0;</tag></item>
                    <item>1<tag>out.digit+=1;</tag></item>
                    <item>one<tag>out.digit+=1;</tag></item>
                    <item>2<tag>out.digit+=2;</tag></item>
                    <item>two<tag>out.digit+=2;</tag></item>
                    <item>3<tag>out.digit+=3;</tag></item>
                    <item>three<tag>out.digit+=3;</tag></item>
                    <item>4<tag>out.digit+=4;</tag></item>
                    <item>four<tag>out.digit+=4;</tag></item>
                    <item>5<tag>out.digit+=5;</tag></item>
                    <item>five<tag>out.digit+=5;</tag></item>
                    <item>6<tag>out.digit+=6;</tag></item>
                    <item>six<tag>out.digit+=5;</tag></item>
                    <item>7<tag>out.digit+=7;</tag></item>
                    <item>seven<tag>out.digit+=7;</tag></item>
                    <item>8<tag>out.digit+=8;</tag></item>
```

```
                    <item>eight<tag>out.digit+=8;</tag></item>
                    <item>9<tag>out.digit+=9;</tag></item>
                    <item>nine<tag>out.digit+=9;</tag></item>
                </one-of>
            </item>
        </rule>
</grammar>
```

信用卡到期日

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="dateCardExpiration"
         mode="voice"
         tag-format="semantics/1.0">

        <rule id="dateCardExpiration" scope="public">
            <tag>out="";</tag>
            <item repeat="1"><ruleref uri="#months"/><tag>out = out + rules.months;</
tag></item>
            <item repeat="1"><ruleref uri="#year"/><tag>out += " " + rules.year.yr;</
tag></item>
        </rule>

        <!-- Test Cases

        Grammar will support the following inputs:

            Scenario 1:
                Input: My card expiration date is july eleven
                Output: 07 2011

            Scenario 2:
                Input: My card expiration date is may twenty six
                Output: 05 2026

        -->

        <rule id="text">
```

```
      <item repeat="0-1"><ruleref uri="#hesitation"/></item>
      <one-of>
         <item repeat="0-1">My card expiration date is </item>
      </one-of>
   </rule>

   <rule id="hesitation">
     <one-of>
        <item>Hmm</item>
        <item>Mmm</item>
        <item>My</item>
     </one-of>
   </rule>

   <rule id="months">
      <item repeat="0-1"><ruleref uri="#text"/></item>
      <one-of>
          <item>january<tag>out="01";</tag></item>
          <item>february<tag>out="02";</tag></item>
          <item>march<tag>out="03";</tag></item>
          <item>april<tag>out="04";</tag></item>
          <item>may<tag>out="05";</tag></item>
          <item>june<tag>out="06";</tag></item>
          <item>july<tag>out="07";</tag></item>
          <item>august<tag>out="08";</tag></item>
          <item>september<tag>out="09";</tag></item>
          <item>october<tag>out="10";</tag></item>
          <item>november<tag>out="11";</tag></item>
          <item>december<tag>out="12";</tag></item>
          <item>jan<tag>out="01";</tag></item>
          <item>feb<tag>out="02";</tag></item>
          <item>aug<tag>out="08";</tag></item>
          <item>sept<tag>out="09";</tag></item>
          <item>oct<tag>out="10";</tag></item>
          <item>nov<tag>out="11";</tag></item>
          <item>dec<tag>out="12";</tag></item>
          <item>1<tag>out="01";</tag></item>
          <item>2<tag>out="02";</tag></item>
          <item>3<tag>out="03";</tag></item>
          <item>4<tag>out="04";</tag></item>
          <item>5<tag>out="05";</tag></item>
          <item>6<tag>out="06";</tag></item>
          <item>7<tag>out="07";</tag></item>
          <item>8<tag>out="08";</tag></item>
```

```
                <item>9<tag>out="09";</tag></item>
                <item>ten<tag>out="10";</tag></item>
                <item>eleven<tag>out="11";</tag></item>
                <item>twelve<tag>out="12";</tag></item>
            </one-of>
        </rule>

        <rule id="digits">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <one-of>
                <item>0<tag>out=0;</tag></item>
                <item>1<tag>out=1;</tag></item>
                <item>2<tag>out=2;</tag></item>
                <item>3<tag>out=3;</tag></item>
                <item>4<tag>out=4;</tag></item>
                <item>5<tag>out=5;</tag></item>
                <item>6<tag>out=6;</tag></item>
                <item>7<tag>out=7;</tag></item>
                <item>8<tag>out=8;</tag></item>
                <item>9<tag>out=9;</tag></item>
                <item>one<tag>out=1;</tag></item>
                <item>two<tag>out=2;</tag></item>
                <item>three<tag>out=3;</tag></item>
                <item>four<tag>out=4;</tag></item>
                <item>five<tag>out=5;</tag></item>
                <item>six<tag>out=6;</tag></item>
                <item>seven<tag>out=7;</tag></item>
                <item>eight<tag>out=8;</tag></item>
                <item>nine<tag>out=9;</tag></item>
            </one-of>
        </rule>

        <rule id="year">
          <tag>out.yr="20"</tag>
          <one-of>
              <item><ruleref uri="#teens"/><tag>out.yr += rules.teens;</tag></item>
              <item><ruleref uri="#above_twenty"/><tag>out.yr += rules.above_twenty;</
tag></item>
          </one-of>
        </rule>

        <rule id="teens">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <one-of>
```

```
            <item>ten<tag>out=10;</tag></item>
            <item>eleven<tag>out=11;</tag></item>
            <item>twelve<tag>out=12;</tag></item>
            <item>thirteen<tag>out=13;</tag></item>
            <item>fourteen<tag>out=14;</tag></item>
            <item>fifteen<tag>out=15;</tag></item>
            <item>sixteen<tag>out=16;</tag></item>
            <item>seventeen<tag>out=17;</tag></item>
            <item>eighteen<tag>out=18;</tag></item>
            <item>nineteen<tag>out=19;</tag></item>
            <item>10<tag>out=10;</tag></item>
            <item>11<tag>out=11;</tag></item>
            <item>12<tag>out=12;</tag></item>
            <item>13<tag>out=13;</tag></item>
            <item>14<tag>out=14;</tag></item>
            <item>15<tag>out=15;</tag></item>
            <item>16<tag>out=16;</tag></item>
            <item>17<tag>out=17;</tag></item>
            <item>18<tag>out=18;</tag></item>
            <item>19<tag>out=19;</tag></item>
        </one-of>
    </rule>

    <rule id="above_twenty">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <one-of>
            <item>twenty<tag>out=20;</tag></item>
            <item>thirty<tag>out=30;</tag></item>
            <item>forty<tag>out=40;</tag></item>
            <item>fifty<tag>out=50;</tag></item>
            <item>sixty<tag>out=60;</tag></item>
            <item>seventy<tag>out=70;</tag></item>
            <item>eighty<tag>out=80;</tag></item>
            <item>ninety<tag>out=90;</tag></item>
            <item>20<tag>out=20;</tag></item>
            <item>30<tag>out=30;</tag></item>
            <item>40<tag>out=40;</tag></item>
            <item>50<tag>out=50;</tag></item>
            <item>60<tag>out=60;</tag></item>
            <item>70<tag>out=70;</tag></item>
            <item>80<tag>out=80;</tag></item>
            <item>90<tag>out=90;</tag></item>
        </one-of>
```

```
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>
</grammar>
```

計劃到期日，日/月/年

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: My plan expires on July Five Two Thousand and Eleven
                 Output: 07/5/11

             Scenario 2:
                 Input: My plan will expire on July Sixteen Two Thousand and Eleven
                 Output: 07/16/11

             Scenario 3:
                 Input: My plan will expire on July Thirty Two Thousand and Eleven
                 Output: 07/30/11
         -->

         <rule id="main" scope="public">
             <tag>out=""</tag>
             <item>
                 <item repeat="1"><ruleref uri="#months"/><tag>out = out +
 rules.months.mon + "/";</tag></item>
                 <one-of>
                   <item><ruleref uri="#digits"/><tag>out += rules.digits + "/";</
tag></item>
```

```
                    <item><ruleref uri="#teens"/><tag>out += rules.teens+ "/";</tag></
item>
                    <item><ruleref uri="#above_twenty"/><tag>out += rules.above_twenty+
 "/";</tag></item>
                </one-of>
                <one-of>
                    <item><ruleref uri="#thousands"/><tag>out += rules.thousands;</
tag></item>
                    <item repeat="0-1"><ruleref uri="#digits"/><tag>out +=
 rules.digits;</tag></item>
                    <item repeat="0-1"><ruleref uri="#teens"/><tag>out +=
 rules.teens;</tag></item>
                    <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out +=
 rules.above_twenty;</tag></item>
                </one-of>
            </item>
        </rule>

        <rule id="text">
          <item repeat="0-1"><ruleref uri="#hesitation"/></item>
          <one-of>
               <item repeat="0-1">My plan expires on</item>
               <item repeat="0-1">My plan expired on</item>
               <item repeat="0-1">My plan will expire on</item>
           </one-of>
        </rule>

        <rule id="hesitation">
           <one-of>
              <item>Hmm</item>
              <item>Mmm</item>
              <item>My</item>
           </one-of>
         </rule>

        <rule id="months">
           <tag>out.mon=""</tag>
        <item repeat="0-1"><ruleref uri="#text"/></item>

           <one-of>
             <item>january<tag>out.mon+="01";</tag></item>
             <item>february<tag>out.mon+="02";</tag></item>
             <item>march<tag>out.mon+="03";</tag></item>
             <item>april<tag>out.mon+="04";</tag></item>
```

```
            <item>may<tag>out.mon+="05";</tag></item>
            <item>june<tag>out.mon+="06";</tag></item>
            <item>july<tag>out.mon+="07";</tag></item>
            <item>august<tag>out.mon+="08";</tag></item>
            <item>september<tag>out.mon+="09";</tag></item>
            <item>october<tag>out.mon+="10";</tag></item>
            <item>november<tag>out.mon+="11";</tag></item>
            <item>december<tag>out.mon+="12";</tag></item>
            <item>jan<tag>out.mon+="01";</tag></item>
            <item>feb<tag>out.mon+="02";</tag></item>
            <item>aug<tag>out.mon+="08";</tag></item>
            <item>sept<tag>out.mon+="09";</tag></item>
            <item>oct<tag>out.mon+="10";</tag></item>
            <item>nov<tag>out.mon+="11";</tag></item>
            <item>dec<tag>out.mon+="12";</tag></item>
        </one-of>
    </rule>

    <rule id="digits">
        <one-of>
            <item>zero<tag>out=0;</tag></item>
            <item>one<tag>out=1;</tag></item>
            <item>two<tag>out=2;</tag></item>
            <item>three<tag>out=3;</tag></item>
            <item>four<tag>out=4;</tag></item>
            <item>five<tag>out=5;</tag></item>
            <item>six<tag>out=6;</tag></item>
            <item>seven<tag>out=7;</tag></item>
            <item>eight<tag>out=8;</tag></item>
            <item>nine<tag>out=9;</tag></item>
        </one-of>
    </rule>

    <rule id="teens">
        <one-of>
            <item>ten<tag>out=10;</tag></item>
            <item>eleven<tag>out=11;</tag></item>
            <item>twelve<tag>out=12;</tag></item>
            <item>thirteen<tag>out=13;</tag></item>
            <item>fourteen<tag>out=14;</tag></item>
            <item>fifteen<tag>out=15;</tag></item>
            <item>sixteen<tag>out=16;</tag></item>
            <item>seventeen<tag>out=17;</tag></item>
            <item>eighteen<tag>out=18;</tag></item>
```

```
                <item>nineteen<tag>out=19;</tag></item>
            </one-of>
        </rule>

        <rule id="thousands">
            <item>two thousand</item>
            <item repeat="0-1">and</item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out = rules.digits;</
tag></item>
            <item repeat="0-1"><ruleref uri="#teens"/><tag>out = rules.teens;</tag></
item>
            <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out =
 rules.above_twenty;</tag></item>
        </rule>

        <rule id="above_twenty">
            <one-of>
                <item>twenty<tag>out=20;</tag></item>
                <item>thirty<tag>out=30;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>
 </grammar>
```

計劃續約日期，月份/年

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: My plan will renew on July Two Thousand and Eleven
```

```
                Output: 07/11

             Scenario 2:
                Input: Renew plan on July Two Thousand and Eleven
                Output: 07/11

         -->

        <rule id="main" scope="public">
            <tag>out=""</tag>
            <item repeat="1-10">
                <item repeat="1"><ruleref uri="#months"/><tag>out = out +
 rules.months.mon + "/";</tag></item>
                <one-of>
                    <item><ruleref uri="#thousands"/><tag>out += rules.thousands;</
tag></item>
                    <item repeat="0-1"><ruleref uri="#digits"/><tag>out +=
 rules.digits;</tag></item>
                    <item repeat="0-1"><ruleref uri="#teens"/><tag>out +=
 rules.teens;</tag></item>
                    <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out +=
 rules.above_twenty;</tag></item>
                </one-of>
            </item>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
            <one-of>
                <item repeat="0-1">My plan will renew on</item>
                <item repeat="0-1">My plan was renewed on</item>
                <item repeat="0-1">Renew plan on</item>
            </one-of>
        </rule>

        <rule id="hesitation">
            <one-of>
               <item>Hmm</item>
               <item>Mmm</item>
               <item>My</item>
            </one-of>
          </rule>

        <rule id="months">
```

```
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <tag>out.mon=""</tag>
        <one-of>
            <item>january<tag>out.mon+="01";</tag></item>
            <item>february<tag>out.mon+="02";</tag></item>
            <item>march<tag>out.mon+="03";</tag></item>
            <item>april<tag>out.mon+="04";</tag></item>
            <item>may<tag>out.mon+="05";</tag></item>
            <item>june<tag>out.mon+="06";</tag></item>
            <item>july<tag>out.mon+="07";</tag></item>
            <item>august<tag>out.mon+="08";</tag></item>
            <item>september<tag>out.mon+="09";</tag></item>
            <item>october<tag>out.mon+="10";</tag></item>
            <item>november<tag>out.mon+="11";</tag></item>
            <item>december<tag>out.mon+="12";</tag></item>
            <item>jan<tag>out.mon+="01";</tag></item>
            <item>feb<tag>out.mon+="02";</tag></item>
            <item>aug<tag>out.mon+="08";</tag></item>
            <item>sept<tag>out.mon+="09";</tag></item>
            <item>oct<tag>out.mon+="10";</tag></item>
            <item>nov<tag>out.mon+="11";</tag></item>
            <item>dec<tag>out.mon+="12";</tag></item>
        </one-of>
    </rule>

    <rule id="digits">
        <one-of>
            <item>zero<tag>out=0;</tag></item>
            <item>one<tag>out=1;</tag></item>
            <item>two<tag>out=2;</tag></item>
            <item>three<tag>out=3;</tag></item>
            <item>four<tag>out=4;</tag></item>
            <item>five<tag>out=5;</tag></item>
            <item>six<tag>out=6;</tag></item>
            <item>seven<tag>out=7;</tag></item>
            <item>eight<tag>out=8;</tag></item>
            <item>nine<tag>out=9;</tag></item>
        </one-of>
    </rule>

    <rule id="teens">
        <one-of>
            <item>ten<tag>out=10;</tag></item>
            <item>eleven<tag>out=11;</tag></item>
```

```
                <item>twelve<tag>out=12;</tag></item>
                <item>thirteen<tag>out=13;</tag></item>
                <item>fourteen<tag>out=14;</tag></item>
                <item>fifteen<tag>out=15;</tag></item>
                <item>sixteen<tag>out=16;</tag></item>
                <item>seventeen<tag>out=17;</tag></item>
                <item>eighteen<tag>out=18;</tag></item>
                <item>nineteen<tag>out=19;</tag></item>
            </one-of>
        </rule>

        <rule id="thousands">
            <item>two thousand</item>
            <item repeat="0-1">and</item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out = rules.digits;</tag></
item>
            <item repeat="0-1"><ruleref uri="#teens"/><tag>out = rules.teens;</tag></
item>
            <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out =
 rules.above_twenty;</tag></item>
        </rule>

        <rule id="above_twenty">
            <one-of>
                <item>twenty<tag>out=20;</tag></item>
                <item>thirty<tag>out=30;</tag></item>
                <item>forty<tag>out=40;</tag></item>
                <item>fifty<tag>out=50;</tag></item>
                <item>sixty<tag>out=60;</tag></item>
                <item>seventy<tag>out=70;</tag></item>
                <item>eighty<tag>out=80;</tag></item>
                <item>ninety<tag>out=90;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>
</grammar>
```

計劃開始日期，月/日

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: My plan will start on july twenty three
                 Output: 07/23

             Scenario 2:
                 Input: My plan will start on july fifteen
                 Output: 07/15

         -->

        <rule id="main" scope="public">
            <tag>out=""</tag>
            <item repeat="1-10">
                <item><ruleref uri="#months"/><tag>out= rules.months.mon + "/";</tag></
item>
                <one-of>
                    <item><ruleref uri="#digits"/><tag>out+= rules.digits;</tag></item>
                    <item><ruleref uri="#teens"/><tag>out+= rules.teens;</tag></item>
                    <item><ruleref uri="#above_twenty"/><tag>out+=
 rules.above_twenty;</tag></item>
                </one-of>
            </item>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
             <one-of>
                <item repeat="0-1">My plan started on</item>
                <item repeat="0-1">My plan will start on</item>
                <item repeat="0-1">I paid it on</item>
                <item repeat="0-1">I paid bill for</item>
             </one-of>
          </rule>
```

```
  <rule id="hesitation">
     <one-of>
        <item>Hmm</item>
        <item>Mmm</item>
        <item>My</item>
     </one-of>
   </rule>

<rule id="months">
   <item repeat="0-1"><ruleref uri="#text"/></item>
   <tag>out.mon=""</tag>
   <one-of>
       <item>january<tag>out.mon+="01";</tag></item>
       <item>february<tag>out.mon+="02";</tag></item>
       <item>march<tag>out.mon+="03";</tag></item>
       <item>april<tag>out.mon+="04";</tag></item>
       <item>may<tag>out.mon+="05";</tag></item>
       <item>june<tag>out.mon+="06";</tag></item>
       <item>july<tag>out.mon+="07";</tag></item>
       <item>august<tag>out.mon+="08";</tag></item>
       <item>september<tag>out.mon+="09";</tag></item>
       <item>october<tag>out.mon+="10";</tag></item>
       <item>november<tag>out.mon+="11";</tag></item>
       <item>december<tag>out.mon+="12";</tag></item>
       <item>jan<tag>out.mon+="01";</tag></item>
       <item>feb<tag>out.mon+="02";</tag></item>
       <item>aug<tag>out.mon+="08";</tag></item>
       <item>sept<tag>out.mon+="09";</tag></item>
       <item>oct<tag>out.mon+="10";</tag></item>
       <item>nov<tag>out.mon+="11";</tag></item>
       <item>dec<tag>out.mon+="12";</tag></item>
   </one-of>
</rule>

<rule id="digits">
    <item repeat="0-1"><ruleref uri="#text"/></item>
    <one-of>
        <item>0<tag>out=0;</tag></item>
        <item>1<tag>out=1;</tag></item>
        <item>2<tag>out=2;</tag></item>
        <item>3<tag>out=3;</tag></item>
        <item>4<tag>out=4;</tag></item>
        <item>5<tag>out=5;</tag></item>
```

```
            <item>6<tag>out=6;</tag></item>
            <item>7<tag>out=7;</tag></item>
            <item>8<tag>out=8;</tag></item>
            <item>9<tag>out=9;</tag></item>
            <item>first<tag>out=01;</tag></item>
            <item>second<tag>out=02;</tag></item>
            <item>third<tag>out=03;</tag></item>
            <item>fourth<tag>out=04;</tag></item>
            <item>fifth<tag>out=05;</tag></item>
            <item>sixth<tag>out=06;</tag></item>
            <item>seventh<tag>out=07;</tag></item>
            <item>eighth<tag>out=08;</tag></item>
            <item>ninth<tag>out=09;</tag></item>
            <item>one<tag>out=1;</tag></item>
            <item>two<tag>out=2;</tag></item>
            <item>three<tag>out=3;</tag></item>
            <item>four<tag>out=4;</tag></item>
            <item>five<tag>out=5;</tag></item>
            <item>six<tag>out=6;</tag></item>
            <item>seven<tag>out=7;</tag></item>
            <item>eight<tag>out=8;</tag></item>
            <item>nine<tag>out=9;</tag></item>
        </one-of>
    </rule>

    <rule id="teens">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <one-of>
            <item>ten<tag>out=10;</tag></item>
            <item>tenth<tag>out=10;</tag></item>
            <item>eleven<tag>out=11;</tag></item>
            <item>twelve<tag>out=12;</tag></item>
            <item>thirteen<tag>out=13;</tag></item>
            <item>fourteen<tag>out=14;</tag></item>
            <item>fifteen<tag>out=15;</tag></item>
            <item>sixteen<tag>out=16;</tag></item>
            <item>seventeen<tag>out=17;</tag></item>
            <item>eighteen<tag>out=18;</tag></item>
            <item>nineteen<tag>out=19;</tag></item>
            <item>tenth<tag>out=10;</tag></item>
            <item>eleventh<tag>out=11;</tag></item>
            <item>twelveth<tag>out=12;</tag></item>
            <item>thirteenth<tag>out=13;</tag></item>
            <item>fourteenth<tag>out=14;</tag></item>
```

```
                <item>fifteenth<tag>out=15;</tag></item>
                <item>sixteenth<tag>out=16;</tag></item>
                <item>seventeenth<tag>out=17;</tag></item>
                <item>eighteenth<tag>out=18;</tag></item>
                <item>nineteenth<tag>out=19;</tag></item>
            </one-of>
        </rule>

        <rule id="above_twenty">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <one-of>
                <item>twenty<tag>out=20;</tag></item>
                <item>thirty<tag>out=30;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>
</grammar>
```

服務開始日期，月/日

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: My plan starts on july twenty three
                 Output: 07/23

             Scenario 2:
                 Input: I want to activate on july fifteen
                 Output: 07/15
```

```
         -->

        <rule id="main" scope="public">
            <tag>out=""</tag>
            <item repeat="1-10">
                <item><ruleref uri="#months"/><tag>out= rules.months.mon + "/";</tag></
item>
                <one-of>
                    <item><ruleref uri="#digits"/><tag>out+= rules.digits;</tag></item>
                    <item><ruleref uri="#teens"/><tag>out+= rules.teens;</tag></item>
                    <item><ruleref uri="#above_twenty"/><tag>out+=
 rules.above_twenty;</tag></item>
                </one-of>
            </item>
        </rule>

        <rule id="text">
            <item repeat="0-1"><ruleref uri="#hesitation"/></item>
             <one-of>
                <item repeat="0-1">My plan starts on</item>
                <item repeat="0-1">I want to start my plan on</item>
                <item repeat="0-1">Activation date of</item>
                <item repeat="0-1">Start activation on</item>
                <item repeat="0-1">I want to activate on</item>
                <item repeat="0-1">Activate plan starting</item>
                <item repeat="0-1">Starting</item>
                <item repeat="0-1">Start on</item>
             </one-of>
          </rule>

          <rule id="hesitation">
             <one-of>
                <item>Hmm</item>
                <item>Mmm</item>
                <item>My</item>
             </one-of>
           </rule>

        <rule id="months">
           <item repeat="0-1"><ruleref uri="#text"/></item>
           <tag>out.mon=""</tag>
           <one-of>
               <item>january<tag>out.mon+="01";</tag></item>
               <item>february<tag>out.mon+="02";</tag></item>
```

```
            <item>march<tag>out.mon+="03";</tag></item>
            <item>april<tag>out.mon+="04";</tag></item>
            <item>may<tag>out.mon+="05";</tag></item>
            <item>june<tag>out.mon+="06";</tag></item>
            <item>july<tag>out.mon+="07";</tag></item>
            <item>august<tag>out.mon+="08";</tag></item>
            <item>september<tag>out.mon+="09";</tag></item>
            <item>october<tag>out.mon+="10";</tag></item>
            <item>november<tag>out.mon+="11";</tag></item>
            <item>december<tag>out.mon+="12";</tag></item>
            <item>jan<tag>out.mon+="01";</tag></item>
            <item>feb<tag>out.mon+="02";</tag></item>
            <item>aug<tag>out.mon+="08";</tag></item>
            <item>sept<tag>out.mon+="09";</tag></item>
            <item>oct<tag>out.mon+="10";</tag></item>
            <item>nov<tag>out.mon+="11";</tag></item>
            <item>dec<tag>out.mon+="12";</tag></item>
        </one-of>
    </rule>

    <rule id="digits">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <one-of>
            <item>0<tag>out=0;</tag></item>
            <item>1<tag>out=1;</tag></item>
            <item>2<tag>out=2;</tag></item>
            <item>3<tag>out=3;</tag></item>
            <item>4<tag>out=4;</tag></item>
            <item>5<tag>out=5;</tag></item>
            <item>6<tag>out=6;</tag></item>
            <item>7<tag>out=7;</tag></item>
            <item>8<tag>out=8;</tag></item>
            <item>9<tag>out=9;</tag></item>
            <item>first<tag>out=01;</tag></item>
            <item>second<tag>out=02;</tag></item>
            <item>third<tag>out=03;</tag></item>
            <item>fourth<tag>out=04;</tag></item>
            <item>fifth<tag>out=05;</tag></item>
            <item>sixth<tag>out=06;</tag></item>
            <item>seventh<tag>out=07;</tag></item>
            <item>eighth<tag>out=08;</tag></item>
            <item>ninth<tag>out=09;</tag></item>
            <item>one<tag>out=1;</tag></item>
            <item>two<tag>out=2;</tag></item>
```

```
            <item>three<tag>out=3;</tag></item>
            <item>four<tag>out=4;</tag></item>
            <item>five<tag>out=5;</tag></item>
            <item>six<tag>out=6;</tag></item>
            <item>seven<tag>out=7;</tag></item>
            <item>eight<tag>out=8;</tag></item>
            <item>nine<tag>out=9;</tag></item>
        </one-of>
    </rule>

    <rule id="teens">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <one-of>
            <item>ten<tag>out=10;</tag></item>
            <item>tenth<tag>out=10;</tag></item>
            <item>eleven<tag>out=11;</tag></item>
            <item>twelve<tag>out=12;</tag></item>
            <item>thirteen<tag>out=13;</tag></item>
            <item>fourteen<tag>out=14;</tag></item>
            <item>fifteen<tag>out=15;</tag></item>
            <item>sixteen<tag>out=16;</tag></item>
            <item>seventeen<tag>out=17;</tag></item>
            <item>eighteen<tag>out=18;</tag></item>
            <item>nineteen<tag>out=19;</tag></item>
            <item>tenth<tag>out=10;</tag></item>
            <item>eleventh<tag>out=11;</tag></item>
            <item>twelveth<tag>out=12;</tag></item>
            <item>thirteenth<tag>out=13;</tag></item>
            <item>fourteenth<tag>out=14;</tag></item>
            <item>fifteenth<tag>out=15;</tag></item>
            <item>sixteenth<tag>out=16;</tag></item>
            <item>seventeenth<tag>out=17;</tag></item>
            <item>eighteenth<tag>out=18;</tag></item>
            <item>nineteenth<tag>out=19;</tag></item>
        </one-of>
    </rule>

    <rule id="above_twenty">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <one-of>
            <item>twenty<tag>out=20;</tag></item>
            <item>thirty<tag>out=30;</tag></item>
        </one-of>
```

```
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>
</grammar>
```

設備數量

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

             Scenario 1:
                 Input: The number is one
                 Output: 1

             Scenario 2:
                 Input: It is ten
                 Output: 10

         -->

        <rule id="main" scope="public">
            <tag>out=""</tag>
            <one-of>
              <item repeat="1"><ruleref uri="#digits"/><tag>out+= rules.digits;</tag></
item>
              <item repeat="1"><ruleref uri="#teens"/><tag>out+= rules.teens;</tag></
item>
              <item repeat="1"><ruleref uri="#above_twenty"/><tag>out+=
 rules.above_twenty;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#thanks"/></item>
        </rule>
```

```
<rule id="text">
    <item repeat="0-1"><ruleref uri="#hesitation"/></item>
   <one-of>
      <item repeat="0-1">It is</item>
      <item repeat="0-1">The number is</item>
      <item repeat="0-1">Order</item>
      <item repeat="0-1">I want to order</item>
      <item repeat="0-1">Total equipment</item>
   </one-of>
</rule>

<rule id="hesitation">
    <one-of>
       <item>Hmm</item>
       <item>Mmm</item>
       <item>My</item>
    </one-of>
  </rule>

<rule id="thanks">
    <one-of>
       <item>Thanks</item>
       <item>I think</item>
    </one-of>
  </rule>

<rule id="digits">
    <item repeat="0-1"><ruleref uri="#text"/></item>
    <one-of>
        <item>0<tag>out=0;</tag></item>
        <item>1<tag>out=1;</tag></item>
        <item>2<tag>out=2;</tag></item>
        <item>3<tag>out=3;</tag></item>
        <item>4<tag>out=4;</tag></item>
        <item>5<tag>out=5;</tag></item>
        <item>6<tag>out=6;</tag></item>
        <item>7<tag>out=7;</tag></item>
        <item>8<tag>out=8;</tag></item>
        <item>9<tag>out=9;</tag></item>
        <item>one<tag>out=1;</tag></item>
        <item>two<tag>out=2;</tag></item>
        <item>three<tag>out=3;</tag></item>
        <item>four<tag>out=4;</tag></item>
```

```
            <item>five<tag>out=5;</tag></item>
            <item>six<tag>out=6;</tag></item>
            <item>seven<tag>out=7;</tag></item>
            <item>eight<tag>out=8;</tag></item>
            <item>nine<tag>out=9;</tag></item>
        </one-of>
    </rule>

    <rule id="teens">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <one-of>
            <item>ten<tag>out=10;</tag></item>
            <item>eleven<tag>out=11;</tag></item>
            <item>twelve<tag>out=12;</tag></item>
            <item>thirteen<tag>out=13;</tag></item>
            <item>fourteen<tag>out=14;</tag></item>
            <item>fifteen<tag>out=15;</tag></item>
            <item>sixteen<tag>out=16;</tag></item>
            <item>seventeen<tag>out=17;</tag></item>
            <item>eighteen<tag>out=18;</tag></item>
            <item>nineteen<tag>out=19;</tag></item>
            <item>10<tag>out=10;</tag></item>
            <item>11<tag>out=11;</tag></item>
            <item>12<tag>out=12;</tag></item>
            <item>13<tag>out=13;</tag></item>
            <item>14<tag>out=14;</tag></item>
            <item>15<tag>out=15;</tag></item>
            <item>16<tag>out=16;</tag></item>
            <item>17<tag>out=17;</tag></item>
            <item>18<tag>out=18;</tag></item>
            <item>19<tag>out=19;</tag></item>
        </one-of>
    </rule>

    <rule id="above_twenty">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <one-of>
            <item>twenty<tag>out=20;</tag></item>
            <item>thirty<tag>out=30;</tag></item>
            <item>forty<tag>out=40;</tag></item>
            <item>fifty<tag>out=50;</tag></item>
            <item>sixty<tag>out=60;</tag></item>
            <item>seventy<tag>out=70;</tag></item>
            <item>eighty<tag>out=80;</tag></item>
```

```
                <item>ninety<tag>out=90;</tag></item>
                <item>20<tag>out=20;</tag></item>
                <item>30<tag>out=30;</tag></item>
                <item>40<tag>out=40;</tag></item>
                <item>50<tag>out=50;</tag></item>
                <item>60<tag>out=60;</tag></item>
                <item>70<tag>out=70;</tag></item>
                <item>80<tag>out=80;</tag></item>
                <item>90<tag>out=90;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>

</grammar>
```

賬單金額

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <!-- Test Cases

         Grammar will support the following inputs:

                 Input: I want to make a payment of one hundred ten dollars
                 Output: $110

         -->

        <rule id="main" scope="public">
            <tag>out="$"</tag>
            <one-of>
                <item><ruleref uri="#sub_hundred"/><tag>out += rules.sub_hundred.sh;</
tag></item>
```

```
                <item><ruleref uri="#subThousands"/><tag>out += rules.subThousands;</
tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#thanks"/></item>
        </rule>

        <rule id="text">
           <item repeat="0-1"><ruleref uri="#hesitation"/></item>
           <one-of>
              <item repeat="0-1">I want to make a payment for</item>
              <item repeat="0-1">I want to make a payment of</item>
              <item repeat="0-1">Pay a total of</item>
              <item repeat="0-1">Paying</item>
              <item repeat="0-1">Pay bill for </item>
           </one-of>
        </rule>

        <rule id="hesitation">
            <one-of>
               <item>Hmm</item>
               <item>Mmm</item>
               <item>My</item>
            </one-of>
          </rule>

          <rule id="thanks">
              <one-of>
                 <item>Thanks</item>
                 <item>I think</item>
              </one-of>
           </rule>

        <rule id="digits">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.num = 0;</tag>
            <one-of>
                <item>0<tag>out.num+=0;</tag></item>
                <item>1<tag>out.num+=1;</tag></item>
                <item>2<tag>out.num+=2;</tag></item>
                <item>3<tag>out.num+=3;</tag></item>
                <item>4<tag>out.num+=4;</tag></item>
                <item>5<tag>out.num+=5;</tag></item>
                <item>6<tag>out.num+=6;</tag></item>
                <item>7<tag>out.num+=7;</tag></item>
```

```
            <item>8<tag>out.num+=8;</tag></item>
            <item>9<tag>out.num+=9;</tag></item>
            <item>one<tag>out.num+=1;</tag></item>
            <item>two<tag>out.num+=2;</tag></item>
            <item>three<tag>out.num+=3;</tag></item>
            <item>four<tag>out.num+=4;</tag></item>
            <item>five<tag>out.num+=5;</tag></item>
            <item>six<tag>out.num+=6;</tag></item>
            <item>seven<tag>out.num+=7;</tag></item>
            <item>eight<tag>out.num+=8;</tag></item>
            <item>nine<tag>out.num+=9;</tag></item>
        </one-of>
        <item repeat="0-1"><ruleref uri="#currency"/></item>
    </rule>

    <rule id="teens">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <tag>out.teen = 0;</tag>
        <one-of>
            <item>ten<tag>out.teen+=10;</tag></item>
            <item>eleven<tag>out.teen+=11;</tag></item>
            <item>twelve<tag>out.teen+=12;</tag></item>
            <item>thirteen<tag>out.teen+=13;</tag></item>
            <item>fourteen<tag>out.teen+=14;</tag></item>
            <item>fifteen<tag>out.teen+=15;</tag></item>
            <item>sixteen<tag>out.teen+=16;</tag></item>
            <item>seventeen<tag>out.teen+=17;</tag></item>
            <item>eighteen<tag>out.teen+=18;</tag></item>
            <item>nineteen<tag>out.teen+=19;</tag></item>
        </one-of>
        <item repeat="0-1"><ruleref uri="#currency"/></item>
    </rule>

    <rule id="above_twenty">
        <item repeat="0-1"><ruleref uri="#text"/></item>
        <tag>out.tens = 0;</tag>
        <one-of>
            <item>twenty<tag>out.tens+=20;</tag></item>
            <item>thirty<tag>out.tens+=30;</tag></item>
            <item>forty<tag>out.tens+=40;</tag></item>
            <item>fifty<tag>out.tens+=50;</tag></item>
            <item>sixty<tag>out.tens+=60;</tag></item>
            <item>seventy<tag>out.tens+=70;</tag></item>
            <item>eighty<tag>out.tens+=80;</tag></item>
```

```
                <item>ninety<tag>out.tens+=90;</tag></item>
                <item>hundred<tag>out.tens+=100;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#currency"/></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out.tens +=
 rules.digits.num;</tag></item>
        </rule>

        <rule id="currency">
            <one-of>
                <item repeat="0-1">dollars</item>
                <item repeat="0-1">Dollars</item>
                <item repeat="0-1">dollar</item>
                <item repeat="0-1">Dollar</item>
            </one-of>
        </rule>


        <rule id="sub_hundred">
            <item repeat="0-1"><ruleref uri="#text"/></item>
            <tag>out.sh = 0;</tag>
            <one-of>
                <item><ruleref uri="#teens"/><tag>out.sh += rules.teens.teen;</tag></
item>
                <item>
                    <ruleref uri="#above_twenty"/><tag>out.sh +=
 rules.above_twenty.tens;</tag>
                </item>
                <item><ruleref uri="#digits"/><tag>out.sh += rules.digits.num;</tag></
item>
            </one-of>
         </rule>

        <rule id="subThousands">
            <ruleref uri="#sub_hundred"/><tag>out = (100 * rules.sub_hundred.sh);</tag>
            hundred
            <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out +=
 rules.above_twenty.tens;</tag></item>
            <item repeat="0-1"><ruleref uri="#teens"/><tag>out += rules.teens.teen;</
tag></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits.num;</
tag></item>
            <item repeat="0-1"><ruleref uri="#currency"/></item>
        </rule>
```

```
</grammar>
```

通用語法（下載）

我們提供以下通用語法：字母數字，貨幣，日期（mm/dd/yy），數字，問候語，猶豫和代理。

英數字元

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">


        <!-- Test Cases

            Scenario 1:
                Input: A B C 1 2 3 4
                Output: ABC1234

            Scenario 2:
                Input: 1 2 3 4 A B C
                Output: 1234ABC

            Scenario 3:
                Input: 1 2 3 4 A B C 1
                Output: 123ABC1
        -->

        <rule id="main" scope="public">
            <tag>out=""</tag>
            <item><ruleref uri="#alphanumeric"/><tag>out +=
 rules.alphanumeric.alphanum;</tag></item>
            <item repeat="0-1"><ruleref uri="#alphabets"/><tag>out +=
 rules.alphabets.letters;</tag></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out +=
 rules.digits.numbers</tag></item>
        </rule>
```

```
        <rule id="alphanumeric" scope="public">
            <tag>out.alphanum=""</tag>
            <item><ruleref uri="#alphabets"/><tag>out.alphanum +=
rules.alphabets.letters;</tag></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out.alphanum +=
rules.digits.numbers</tag></item>
        </rule>

        <rule id="alphabets">
            <tag>out.letters=""</tag>
            <tag>out.firstOccurence=""</tag>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out.firstOccurence +=
rules.digits.numbers; out.letters += out.firstOccurence;</tag></item>
            <item repeat="1-">
                <one-of>
                    <item>A<tag>out.letters+='A';</tag></item>
                    <item>B<tag>out.letters+='B';</tag></item>
                    <item>C<tag>out.letters+='C';</tag></item>
                    <item>D<tag>out.letters+='D';</tag></item>
                    <item>E<tag>out.letters+='E';</tag></item>
                    <item>F<tag>out.letters+='F';</tag></item>
                    <item>G<tag>out.letters+='G';</tag></item>
                    <item>H<tag>out.letters+='H';</tag></item>
                    <item>I<tag>out.letters+='I';</tag></item>
                    <item>J<tag>out.letters+='J';</tag></item>
                    <item>K<tag>out.letters+='K';</tag></item>
                    <item>L<tag>out.letters+='L';</tag></item>
                    <item>M<tag>out.letters+='M';</tag></item>
                    <item>N<tag>out.letters+='N';</tag></item>
                    <item>O<tag>out.letters+='O';</tag></item>
                    <item>P<tag>out.letters+='P';</tag></item>
                    <item>Q<tag>out.letters+='Q';</tag></item>
                    <item>R<tag>out.letters+='R';</tag></item>
                    <item>S<tag>out.letters+='S';</tag></item>
                    <item>T<tag>out.letters+='T';</tag></item>
                    <item>U<tag>out.letters+='U';</tag></item>
                    <item>V<tag>out.letters+='V';</tag></item>
                    <item>W<tag>out.letters+='W';</tag></item>
                    <item>X<tag>out.letters+='X';</tag></item>
                    <item>Y<tag>out.letters+='Y';</tag></item>
                    <item>Z<tag>out.letters+='Z';</tag></item>
                </one-of>
            </item>
        </rule>
```

```
        <rule id="digits">
            <tag>out.numbers=""</tag>
            <item repeat="1-10">
                <one-of>
                    <item>0<tag>out.numbers+=0;</tag></item>
                    <item>1<tag>out.numbers+=1;</tag></item>
                    <item>2<tag>out.numbers+=2;</tag></item>
                    <item>3<tag>out.numbers+=3;</tag></item>
                    <item>4<tag>out.numbers+=4;</tag></item>
                    <item>5<tag>out.numbers+=5;</tag></item>
                    <item>6<tag>out.numbers+=6;</tag></item>
                    <item>7<tag>out.numbers+=7;</tag></item>
                    <item>8<tag>out.numbers+=8;</tag></item>
                    <item>9<tag>out.numbers+=9;</tag></item>
                </one-of>
            </item>
        </rule>
</grammar>
```

貨幣

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

        <rule id="main" scope="public">
            <tag>out="$"</tag>
            <one-of>
                <item><ruleref uri="#sub_hundred"/><tag>out += rules.sub_hundred.sh;</
tag></item>
                <item><ruleref uri="#subThousands"/><tag>out += rules.subThousands;</
tag></item>
            </one-of>
        </rule>

        <rule id="digits">
```

```
            <tag>out.num = 0;</tag>
            <one-of>
                <item>0<tag>out.num+=0;</tag></item>
                <item>1<tag>out.num+=1;</tag></item>
                <item>2<tag>out.num+=2;</tag></item>
                <item>3<tag>out.num+=3;</tag></item>
                <item>4<tag>out.num+=4;</tag></item>
                <item>5<tag>out.num+=5;</tag></item>
                <item>6<tag>out.num+=6;</tag></item>
                <item>7<tag>out.num+=7;</tag></item>
                <item>8<tag>out.num+=8;</tag></item>
                <item>9<tag>out.num+=9;</tag></item>
                <item>one<tag>out.num+=1;</tag></item>
                <item>two<tag>out.num+=2;</tag></item>
                <item>three<tag>out.num+=3;</tag></item>
                <item>four<tag>out.num+=4;</tag></item>
                <item>five<tag>out.num+=5;</tag></item>
                <item>six<tag>out.num+=6;</tag></item>
                <item>seven<tag>out.num+=7;</tag></item>
                <item>eight<tag>out.num+=8;</tag></item>
                <item>nine<tag>out.num+=9;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#currency"/></item>
        </rule>

        <rule id="teens">
            <tag>out.teen = 0;</tag>
            <one-of>
                <item>ten<tag>out.teen+=10;</tag></item>
                <item>eleven<tag>out.teen+=11;</tag></item>
                <item>twelve<tag>out.teen+=12;</tag></item>
                <item>thirteen<tag>out.teen+=13;</tag></item>
                <item>fourteen<tag>out.teen+=14;</tag></item>
                <item>fifteen<tag>out.teen+=15;</tag></item>
                <item>sixteen<tag>out.teen+=16;</tag></item>
                <item>seventeen<tag>out.teen+=17;</tag></item>
                <item>eighteen<tag>out.teen+=18;</tag></item>
                <item>nineteen<tag>out.teen+=19;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#currency"/></item>
        </rule>

        <rule id="above_twenty">
            <tag>out.tens = 0;</tag>
```

```
            <one-of>
                <item>twenty<tag>out.tens+=20;</tag></item>
                <item>thirty<tag>out.tens+=30;</tag></item>
                <item>forty<tag>out.tens+=40;</tag></item>
                <item>fifty<tag>out.tens+=50;</tag></item>
                <item>sixty<tag>out.tens+=60;</tag></item>
                <item>seventy<tag>out.tens+=70;</tag></item>
                <item>eighty<tag>out.tens+=80;</tag></item>
                <item>ninety<tag>out.tens+=90;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#currency"/></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out.tens +=
 rules.digits.num;</tag></item>
        </rule>

        <rule id="currency">
            <one-of>
                <item repeat="0-1">dollars</item>
                <item repeat="0-1">Dollars</item>
                <item repeat="0-1">dollar</item>
                <item repeat="0-1">Dollar</item>
            </one-of>
        </rule>


        <rule id="sub_hundred">
            <tag>out.sh = 0;</tag>
            <one-of>
                <item><ruleref uri="#teens"/><tag>out.sh += rules.teens.teen;</tag></
item>
                <item>
                    <ruleref uri="#above_twenty"/><tag>out.sh +=
 rules.above_twenty.tens;</tag>
                </item>
                <item><ruleref uri="#digits"/><tag>out.sh += rules.digits.num;</tag></
item>
            </one-of>
         </rule>

        <rule id="subThousands">
            <ruleref uri="#sub_hundred"/><tag>out = (100 * rules.sub_hundred.sh);</tag>
            hundred
            <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out +=
 rules.above_twenty.tens;</tag></item>
```

```
            <item repeat="0-1"><ruleref uri="#teens"/><tag>out += rules.teens.teen;</
tag></item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits.num;</
tag></item>
        </rule>
</grammar>
```

日期，DD/毫米

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

        <rule id="main" scope="public">
            <tag>out="";</tag>
            <item repeat="1-10">
                <one-of>
                    <item><ruleref uri="#digits"/><tag>out += rules.digits + " ";</
tag></item>
                    <item><ruleref uri="#teens"/><tag>out += rules.teens+ " ";</tag></
item>
                    <item><ruleref uri="#above_twenty"/><tag>out += rules.above_twenty+
 " ";</tag></item>
                </one-of>
                <item><ruleref uri="#months"/><tag>out = out + rules.months;</tag></
item>
            </item>
        </rule>

        <rule id="months">
           <one-of>
             <item>january<tag>out="january";</tag></item>
             <item>february<tag>out="february";</tag></item>
             <item>march<tag>out="march";</tag></item>
             <item>april<tag>out="april";</tag></item>
             <item>may<tag>out="may";</tag></item>
             <item>june<tag>out="june";</tag></item>
```

```
            <item>july<tag>out="july";</tag></item>
            <item>august<tag>out="august";</tag></item>
            <item>september<tag>out="september";</tag></item>
            <item>october<tag>out="october";</tag></item>
            <item>november<tag>out="november";</tag></item>
            <item>december<tag>out="december";</tag></item>
            <item>jan<tag>out="january";</tag></item>
            <item>feb<tag>out="february";</tag></item>
            <item>aug<tag>out="august";</tag></item>
            <item>sept<tag>out="september";</tag></item>
            <item>oct<tag>out="october";</tag></item>
            <item>nov<tag>out="november";</tag></item>
            <item>dec<tag>out="december";</tag></item>
        </one-of>
    </rule>

     <rule id="digits">
         <one-of>
             <item>0<tag>out=0;</tag></item>
             <item>1<tag>out=1;</tag></item>
             <item>2<tag>out=2;</tag></item>
             <item>3<tag>out=3;</tag></item>
             <item>4<tag>out=4;</tag></item>
             <item>5<tag>out=5;</tag></item>
             <item>6<tag>out=6;</tag></item>
             <item>7<tag>out=7;</tag></item>
             <item>8<tag>out=8;</tag></item>
             <item>9<tag>out=9;</tag></item>
             <item>first<tag>out=1;</tag></item>
             <item>second<tag>out=2;</tag></item>
             <item>third<tag>out=3;</tag></item>
             <item>fourth<tag>out=4;</tag></item>
             <item>fifth<tag>out=5;</tag></item>
             <item>sixth<tag>out=6;</tag></item>
             <item>seventh<tag>out=7;</tag></item>
             <item>eighth<tag>out=8;</tag></item>
             <item>ninth<tag>out=9;</tag></item>
             <item>one<tag>out=1;</tag></item>
             <item>two<tag>out=2;</tag></item>
             <item>three<tag>out=3;</tag></item>
             <item>four<tag>out=4;</tag></item>
             <item>five<tag>out=5;</tag></item>
             <item>six<tag>out=6;</tag></item>
             <item>seven<tag>out=7;</tag></item>
```

```
                <item>eight<tag>out=8;</tag></item>
                <item>nine<tag>out=9;</tag></item>
            </one-of>
        </rule>

        <rule id="teens">
            <one-of>
                <item>ten<tag>out=10;</tag></item>
                <item>tenth<tag>out=10;</tag></item>
                <item>eleven<tag>out=11;</tag></item>
                <item>twelve<tag>out=12;</tag></item>
                <item>thirteen<tag>out=13;</tag></item>
                <item>fourteen<tag>out=14;</tag></item>
                <item>fifteen<tag>out=15;</tag></item>
                <item>sixteen<tag>out=16;</tag></item>
                <item>seventeen<tag>out=17;</tag></item>
                <item>eighteen<tag>out=18;</tag></item>
                <item>nineteen<tag>out=19;</tag></item>
                <item>tenth<tag>out=10;</tag></item>
                <item>eleventh<tag>out=11;</tag></item>
                <item>twelveth<tag>out=12;</tag></item>
                <item>thirteenth<tag>out=13;</tag></item>
                <item>fourteenth<tag>out=14;</tag></item>
                <item>fifteenth<tag>out=15;</tag></item>
                <item>sixteenth<tag>out=16;</tag></item>
                <item>seventeenth<tag>out=17;</tag></item>
                <item>eighteenth<tag>out=18;</tag></item>
                <item>nineteenth<tag>out=19;</tag></item>
            </one-of>
        </rule>

        <rule id="above_twenty">
            <one-of>
                <item>twenty<tag>out=20;</tag></item>
                <item>thirty<tag>out=30;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>
</grammar>
```

日期，毫米/年

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

        <rule id="main" scope="public">
            <tag>out="";</tag>
            <item repeat="1-10">
                <item repeat="1"><ruleref uri="#months"/><tag>out = out +
 rules.months.mon + " ";</tag></item>
                <one-of>
                    <item><ruleref uri="#thousands"/><tag>out += rules.thousands;</
tag></item>
                    <item repeat="0-1"><ruleref uri="#digits"/><tag>out +=
 rules.digits;</tag></item>
                    <item repeat="0-1"><ruleref uri="#teens"/><tag>out +=
 rules.teens;</tag></item>
                    <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out +=
 rules.above_twenty;</tag></item>
                </one-of>
            </item>
        </rule>

        <rule id="months">
           <tag>out.mon="";</tag>
           <one-of>
               <item>january<tag>out.mon+="january";</tag></item>
               <item>february<tag>out.mon+="february";</tag></item>
               <item>march<tag>out.mon+="march";</tag></item>
               <item>april<tag>out.mon+="april";</tag></item>
               <item>may<tag>out.mon+="may";</tag></item>
               <item>june<tag>out.mon+="june";</tag></item>
               <item>july<tag>out.mon+="july";</tag></item>
               <item>august<tag>out.mon+="august";</tag></item>
               <item>september<tag>out.mon+="september";</tag></item>
               <item>october<tag>out.mon+="october";</tag></item>
               <item>november<tag>out.mon+="november";</tag></item>
```

```
            <item>december<tag>out.mon+="december";</tag></item>
            <item>jan<tag>out.mon+="january";</tag></item>
            <item>feb<tag>out.mon+="february";</tag></item>
            <item>aug<tag>out.mon+="august";</tag></item>
            <item>sept<tag>out.mon+="september";</tag></item>
            <item>oct<tag>out.mon+="october";</tag></item>
            <item>nov<tag>out.mon+="november";</tag></item>
            <item>dec<tag>out.mon+="december";</tag></item>
        </one-of>
    </rule>

    <rule id="digits">
        <one-of>
            <item>zero<tag>out=0;</tag></item>
            <item>one<tag>out=1;</tag></item>
            <item>two<tag>out=2;</tag></item>
            <item>three<tag>out=3;</tag></item>
            <item>four<tag>out=4;</tag></item>
            <item>five<tag>out=5;</tag></item>
            <item>six<tag>out=6;</tag></item>
            <item>seven<tag>out=7;</tag></item>
            <item>eight<tag>out=8;</tag></item>
            <item>nine<tag>out=9;</tag></item>
        </one-of>
    </rule>

    <rule id="teens">
        <one-of>
            <item>ten<tag>out=10;</tag></item>
            <item>eleven<tag>out=11;</tag></item>
            <item>twelve<tag>out=12;</tag></item>
            <item>thirteen<tag>out=13;</tag></item>
            <item>fourteen<tag>out=14;</tag></item>
            <item>fifteen<tag>out=15;</tag></item>
            <item>sixteen<tag>out=16;</tag></item>
            <item>seventeen<tag>out=17;</tag></item>
            <item>eighteen<tag>out=18;</tag></item>
            <item>nineteen<tag>out=19;</tag></item>
        </one-of>
    </rule>

    <!--  <rule id="singleDigit">
        <item><ruleref uri="#digits"/><tag>out += rules.digits;</tag></item>
    </rule> -->
```

```
        <rule id="thousands">
            <!-- <item>
                <ruleref uri="#digits"/>
                <tag>out = (1000 * rules.digits);</tag>
                thousand
            </item> -->
            <item>two thousand<tag>out=2000;</tag></item>
            <item repeat="0-1">and</item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
            <item repeat="0-1"><ruleref uri="#teens"/><tag>out += rules.teens;</tag></
item>
            <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out +=
 rules.above_twenty;</tag></item>
        </rule>

        <rule id="above_twenty">
            <one-of>
                <item>twenty<tag>out=20;</tag></item>
                <item>thirty<tag>out=30;</tag></item>
                <item>forty<tag>out=40;</tag></item>
                <item>fifty<tag>out=50;</tag></item>
                <item>sixty<tag>out=60;</tag></item>
                <item>seventy<tag>out=70;</tag></item>
                <item>eighty<tag>out=80;</tag></item>
                <item>ninety<tag>out=90;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>

</grammar>
```

日期，日/月/年

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
```

```
         mode="voice"
         tag-format="semantics/1.0">

         <rule id="main" scope="public">
             <tag>out=""</tag>
             <item repeat="1-10">
                 <one-of>
                   <item><ruleref uri="#digits"/><tag>out += rules.digits + " ";</
tag></item>
                   <item><ruleref uri="#teens"/><tag>out += rules.teens+ " ";</tag></
item>
                   <item><ruleref uri="#above_twenty"/><tag>out += rules.above_twenty+
 " ";</tag></item>
                 </one-of>
                 <item repeat="1"><ruleref uri="#months"/><tag>out = out +
 rules.months.mon + " ";</tag></item>
                 <one-of>
                     <item><ruleref uri="#thousands"/><tag>out += rules.thousands;</
tag></item>
                     <item repeat="0-1"><ruleref uri="#digits"/><tag>out +=
 rules.digits;</tag></item>
                     <item repeat="0-1"><ruleref uri="#teens"/><tag>out +=
 rules.teens;</tag></item>
                     <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out +=
 rules.above_twenty;</tag></item>
                 </one-of>
             </item>
         </rule>

         <rule id="months">
            <tag>out.mon=""</tag>
            <one-of>
                <item>january<tag>out.mon+="january";</tag></item>
                <item>february<tag>out.mon+="february";</tag></item>
                <item>march<tag>out.mon+="march";</tag></item>
                <item>april<tag>out.mon+="april";</tag></item>
                <item>may<tag>out.mon+="may";</tag></item>
                <item>june<tag>out.mon+="june";</tag></item>
                <item>july<tag>out.mon+="july";</tag></item>
                <item>august<tag>out.mon+="august";</tag></item>
                <item>september<tag>out.mon+="september";</tag></item>
                <item>october<tag>out.mon+="october";</tag></item>
                <item>november<tag>out.mon+="november";</tag></item>
                <item>december<tag>out.mon+="december";</tag></item>
```

```
            <item>jan<tag>out.mon+="january";</tag></item>
            <item>feb<tag>out.mon+="february";</tag></item>
            <item>aug<tag>out.mon+="august";</tag></item>
            <item>sept<tag>out.mon+="september";</tag></item>
            <item>oct<tag>out.mon+="october";</tag></item>
            <item>nov<tag>out.mon+="november";</tag></item>
            <item>dec<tag>out.mon+="december";</tag></item>
        </one-of>
    </rule>

    <rule id="digits">
        <one-of>
            <item>zero<tag>out=0;</tag></item>
            <item>one<tag>out=1;</tag></item>
            <item>two<tag>out=2;</tag></item>
            <item>three<tag>out=3;</tag></item>
            <item>four<tag>out=4;</tag></item>
            <item>five<tag>out=5;</tag></item>
            <item>six<tag>out=6;</tag></item>
            <item>seven<tag>out=7;</tag></item>
            <item>eight<tag>out=8;</tag></item>
            <item>nine<tag>out=9;</tag></item>
        </one-of>
    </rule>

    <rule id="teens">
        <one-of>
            <item>ten<tag>out=10;</tag></item>
            <item>eleven<tag>out=11;</tag></item>
            <item>twelve<tag>out=12;</tag></item>
            <item>thirteen<tag>out=13;</tag></item>
            <item>fourteen<tag>out=14;</tag></item>
            <item>fifteen<tag>out=15;</tag></item>
            <item>sixteen<tag>out=16;</tag></item>
            <item>seventeen<tag>out=17;</tag></item>
            <item>eighteen<tag>out=18;</tag></item>
            <item>nineteen<tag>out=19;</tag></item>
        </one-of>
    </rule>

    <rule id="thousands">
        <item>two thousand<tag>out=2000;</tag></item>
        <item repeat="0-1">and</item>
```

```
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
            <item repeat="0-1"><ruleref uri="#teens"/><tag>out += rules.teens;</tag></
item>
            <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out +=
 rules.above_twenty;</tag></item>
        </rule>

        <rule id="above_twenty">
            <one-of>
                <item>twenty<tag>out=20;</tag></item>
                <item>thirty<tag>out=30;</tag></item>
                <item>forty<tag>out=40;</tag></item>
                <item>fifty<tag>out=50;</tag></item>
                <item>sixty<tag>out=60;</tag></item>
                <item>seventy<tag>out=70;</tag></item>
                <item>eighty<tag>out=80;</tag></item>
                <item>ninety<tag>out=90;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>

 </grammar>
```

數字，數字

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="digits"
         mode="voice"
         tag-format="semantics/1.0">

        <rule id="digits">
            <tag>out=""</tag>
            <item><ruleref uri="#singleDigit"/><tag>out += rules.singleDigit.digit;</
tag></item>
        </rule>
```

```
        <rule id="singleDigit">
            <tag>out.digit=""</tag>
            <item repeat="1-10">
                <one-of>
                    <item>0<tag>out.digit+=0;</tag></item>
                    <item>zero<tag>out.digit+=0;</tag></item>
                    <item>1<tag>out.digit+=1;</tag></item>
                    <item>one<tag>out.digit+=1;</tag></item>
                    <item>2<tag>out.digit+=2;</tag></item>
                    <item>two<tag>out.digit+=2;</tag></item>
                    <item>3<tag>out.digit+=3;</tag></item>
                    <item>three<tag>out.digit+=3;</tag></item>
                    <item>4<tag>out.digit+=4;</tag></item>
                    <item>four<tag>out.digit+=4;</tag></item>
                    <item>5<tag>out.digit+=5;</tag></item>
                    <item>five<tag>out.digit+=5;</tag></item>
                    <item>6<tag>out.digit+=6;</tag></item>
                    <item>six<tag>out.digit+=6;</tag></item>
                    <item>7<tag>out.digit+=7;</tag></item>
                    <item>seven<tag>out.digit+=7;</tag></item>
                    <item>8<tag>out.digit+=8;</tag></item>
                    <item>eight<tag>out.digit+=8;</tag></item>
                    <item>9<tag>out.digit+=9;</tag></item>
                    <item>nine<tag>out.digit+=9;</tag></item>
                </one-of>
            </item>
        </rule>
</grammar>
```

序數

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

        <rule id="main" scope="public">
            <tag>out=""</tag>
```

```
            <one-of>
              <item repeat="1"><ruleref uri="#digits"/><tag>out+= rules.digits;</tag></
item>
              <item repeat="1"><ruleref uri="#teens"/><tag>out+= rules.teens;</tag></
item>
              <item repeat="1"><ruleref uri="#above_twenty"/><tag>out+=
 rules.above_twenty;</tag></item>
            </one-of>
        </rule>

        <rule id="digits">
            <one-of>
                <item>0<tag>out=0;</tag></item>
                <item>1<tag>out=1;</tag></item>
                <item>2<tag>out=2;</tag></item>
                <item>3<tag>out=3;</tag></item>
                <item>4<tag>out=4;</tag></item>
                <item>5<tag>out=5;</tag></item>
                <item>6<tag>out=6;</tag></item>
                <item>7<tag>out=7;</tag></item>
                <item>8<tag>out=8;</tag></item>
                <item>9<tag>out=9;</tag></item>
                <item>one<tag>out=1;</tag></item>
                <item>two<tag>out=2;</tag></item>
                <item>three<tag>out=3;</tag></item>
                <item>four<tag>out=4;</tag></item>
                <item>five<tag>out=5;</tag></item>
                <item>six<tag>out=6;</tag></item>
                <item>seven<tag>out=7;</tag></item>
                <item>eight<tag>out=8;</tag></item>
                <item>nine<tag>out=9;</tag></item>
            </one-of>
        </rule>

        <rule id="teens">
            <one-of>
                <item>ten<tag>out=10;</tag></item>
                <item>eleven<tag>out=11;</tag></item>
                <item>twelve<tag>out=12;</tag></item>
                <item>thirteen<tag>out=13;</tag></item>
                <item>fourteen<tag>out=14;</tag></item>
                <item>fifteen<tag>out=15;</tag></item>
                <item>sixteen<tag>out=16;</tag></item>
                <item>seventeen<tag>out=17;</tag></item>
```

```
                <item>eighteen<tag>out=18;</tag></item>
                <item>nineteen<tag>out=19;</tag></item>
                <item>10<tag>out=10;</tag></item>
                <item>11<tag>out=11;</tag></item>
                <item>12<tag>out=12;</tag></item>
                <item>13<tag>out=13;</tag></item>
                <item>14<tag>out=14;</tag></item>
                <item>15<tag>out=15;</tag></item>
                <item>16<tag>out=16;</tag></item>
                <item>17<tag>out=17;</tag></item>
                <item>18<tag>out=18;</tag></item>
                <item>19<tag>out=19;</tag></item>
            </one-of>
        </rule>

        <rule id="above_twenty">
            <one-of>
                <item>twenty<tag>out=20;</tag></item>
                <item>thirty<tag>out=30;</tag></item>
                <item>forty<tag>out=40;</tag></item>
                <item>fifty<tag>out=50;</tag></item>
                <item>sixty<tag>out=60;</tag></item>
                <item>seventy<tag>out=70;</tag></item>
                <item>eighty<tag>out=80;</tag></item>
                <item>ninety<tag>out=90;</tag></item>
                <item>20<tag>out=20;</tag></item>
                <item>30<tag>out=30;</tag></item>
                <item>40<tag>out=40;</tag></item>
                <item>50<tag>out=50;</tag></item>
                <item>60<tag>out=60;</tag></item>
                <item>70<tag>out=70;</tag></item>
                <item>80<tag>out=80;</tag></item>
                <item>90<tag>out=90;</tag></item>
            </one-of>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
        </rule>

</grammar>
```

客服人員

```
<?xml version="1.0" encoding="UTF-8" ?>
```

```
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <rule id="main" scope="public">
             <tag>out=""</tag>
             <ruleref uri="#text"/><tag>out = rules.text</tag>
         </rule>

         <rule id="text">
           <one-of>
              <item>Can I talk to the agent<tag>out="You will be trasnfered to the
 agent in a while"</tag></item>
              <item>talk to an agent<tag>out="You will be trasnfered to the agent in a
 while"</tag></item>
           </one-of>
         </rule>
</grammar>
```

問候

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <rule id="main" scope="public">
             <tag>out=""</tag>
             <ruleref uri="#text"/><tag>out = rules.text</tag>
         </rule>

         <rule id="text">
           <one-of>
```

```
            <item>hey<tag>out="Greeting"</tag></item>
            <item>hi<tag>out="Greeting"</tag></item>
            <item>Hi<tag>out="Greeting"</tag></item>
            <item>Hey<tag>out="Greeting"</tag></item>
            <item>Hello<tag>out="Greeting"</tag></item>
            <item>hello<tag>out="Greeting"</tag></item>
          </one-of>
        </rule>
</grammar>
```

猶豫

```
<?xml version="1.0" encoding="UTF-8" ?>
<grammar xmlns="http://www.w3.org/2001/06/grammar"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.w3.org/2001/06/grammar
                             http://www.w3.org/TR/speech-grammar/grammar.xsd"
         xml:lang="en-US" version="1.0"
         root="main"
         mode="voice"
         tag-format="semantics/1.0">

         <rule id="main" scope="public">
             <tag>out=""</tag>
             <ruleref uri="#text"/><tag>out = rules.text</tag>
         </rule>

         <rule id="text">
           <one-of>
             <item>Hmm<tag>out="Waiting for your input"</tag></item>
             <item>Mmm<tag>out="Waiting for your input"</tag></item>
             <item>Can you please wait<tag>out="Waiting for your input"</tag></item>
           </one-of>
         </rule>
</grammar>
```

複合槽類型

複合插槽是兩個或多個插槽的組合，可在單一使用者輸入中擷取多個資訊片段。例如，您可以將機器人設定為透過要求輸入「城市和州/省或郵遞區號」來引出位置。相比之下，當對話配置為使用單獨的插槽類型時，會產生嚴格的對話體驗（「什麼是城市？」 其次是「什麼是郵遞區號？」）。使用複合插槽，您可以通過單個插槽捕獲所有信息。複合插槽是稱為子插槽的組合，例如城市、州和郵遞區號。

您可以使用可用的 Amazon Lex 插槽類型 (內建插件) 和您自己的插槽 (自訂插槽) 的組合。您可以設計邏輯運算式來擷取所需子槽內的資訊。例如：城市和州或郵遞區號。

複合插槽類型僅適用於 en-US。

建立複合槽類型

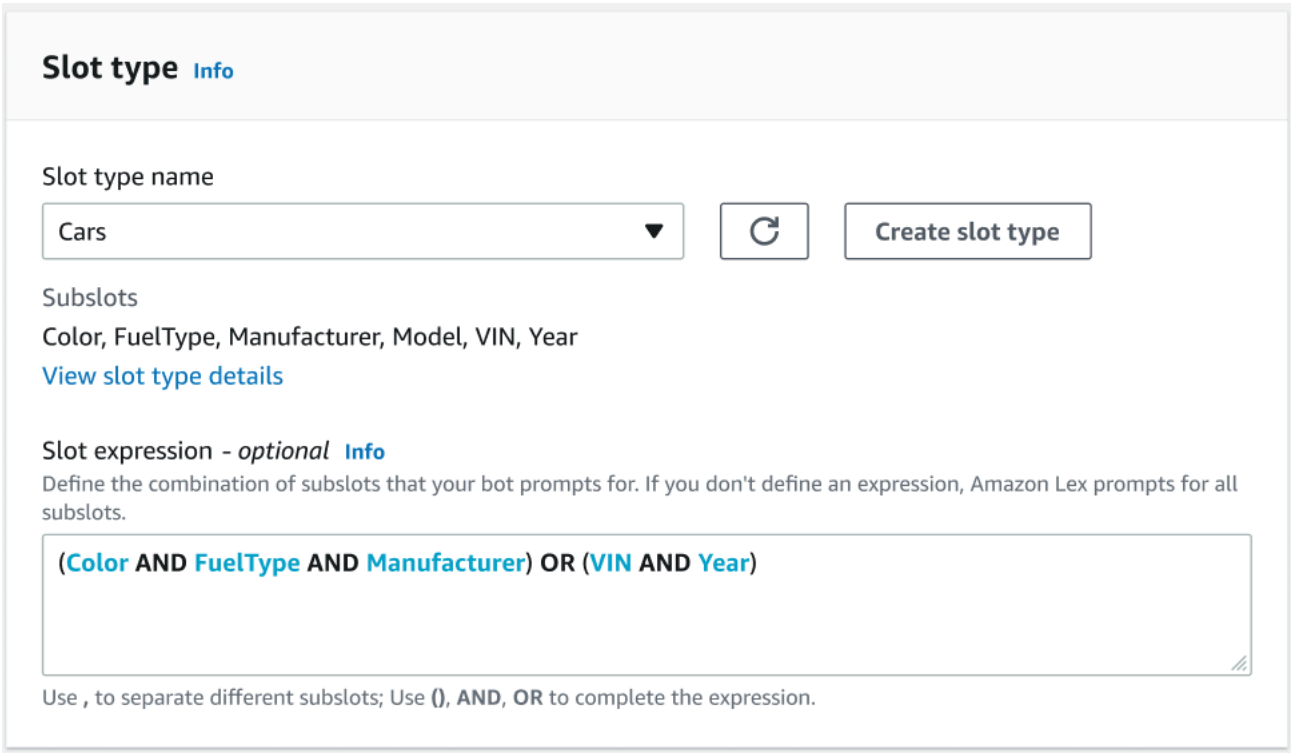
若要在複合槽中使用子槽，您必須先配置複合槽類型。若要這麼做，請使用新增插槽類型主控台步驟或 API 作業。選擇複合槽類型的名稱和描述之後，您必須提供子槽的資訊。如需新增插槽類型的詳細資訊，請參閱 新增插槽類型

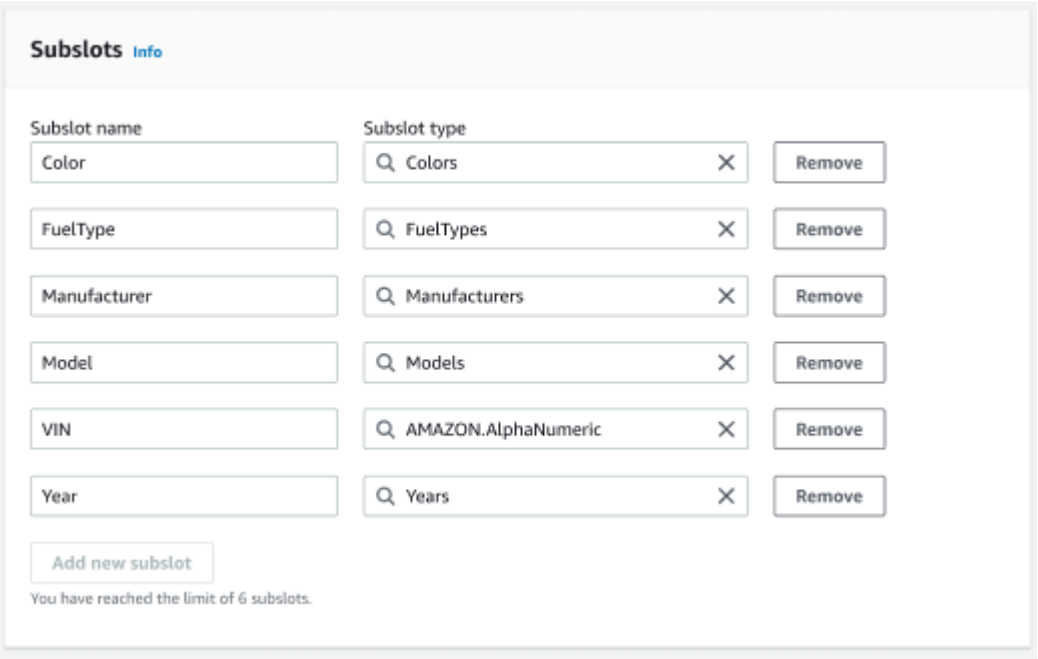
子槽

複合插槽類型需要基礎插槽 (稱為子槽) 的組態。如果您希望在一個請求中從客戶那裡獲取多個信息，請配置子插槽的組合。例如：城市、州和郵遞區號。您最多可以為複合插槽加入 6 個子槽。

奇異槽類型的槽可用於將子槽增加至複材槽類型。但是，您不能使用複合槽類型作為子槽的插槽類型。

下列影像是複合插槽「Car」的圖例，這是子槽的組合：顏色、、製造商FuelType、型號、VIN 和年份。





表示式建置

若要驅動複合槽的履行，您可以選擇性地使用表示式建置器。使用運算式建置器，您可以設計邏輯槽運算式，以所需的順序擷取所需的子槽值。做為布林運算式的一部分，您可以使用 AND 和 OR 等運算子。根據設計的表達式，當滿足所需的子槽時，複合槽被視為已滿足。

使用複合槽類型

對於某些意圖，您可能希望捕獲不同的插槽作為單個插槽的一部分。例如，汽車維修調度機器人可能有以下語調的意圖：

```
My car is a {car}
```

意圖預計 {car} 複合插槽包含一個插槽的列表，其中包含汽車的細節。例如，「2021 年白色豐田凱美瑞」。

複合插槽與多值插槽不同。複合槽由多個插槽組成，每個槽都有自己的值。然而，多值槽是一個可以包含值列表的單數槽。有關多值槽的更多信息，請參閱，在插槽中使用多個值

對於複合插槽，Amazon Lex 會針對RecognizeText或RecognizeUtterance作業的回應傳回每個子插槽的值。以下是為話語返回的插槽信息：「我想從機器人為我的「2021 年白色豐田凱美瑞」安排服務。CarService

```
"slots": {
    "CarType": {
        "value": {
```

```
            "originalValue": "White Toyota Camry 2021",
            "interpretedValue": "White Toyota Camry 2021",
            "resolvedValues": [
                "white Toyota Camry 2021"
            ]
        },
        "subSlots": {
            "Color": {
                "value": {
                    "originalValue": "White",
                    "interpretedValue": "White",
                    "resolvedValues": [
                        "white"
                    ]
                },
                "shape": "Scalar"
            },
            "Manufacturer": {
                "value": {
                    "originalValue": "Toyota",
                    "interpretedValue": "Toyota",
                    "resolvedValues": [
                        "Toyota"
                    ]
                },
                "shape": "Scalar"
            },
            "Model": {
                "value": {
                    "originalValue": "Camry",
                    "interpretedValue": "Camry",
                    "resolvedValues": [
                       "Camry"
                    ]
                },
                "shape": "Scalar"
            },
            "Year": {
                "value": {
                    "originalValue": "2021",
                    "interpretedValue": "2021",
                    "resolvedValues": [
                        "2021"
                    ]
```

```
                },
                "shape": "Scalar"
            }
        }
    },
    ...
}
```

複合插槽可以引導在第一回合或第 n 回合的對話。根據提供的輸入值，複合槽可引出剩餘所需的子槽。

複合槽一律會傳回每個子插槽的值。當語音不包含給定子插槽的可識別值時，沒有針對該特定子插槽返回任何響應。

複合插槽可與文本和語音輸入一起使用。

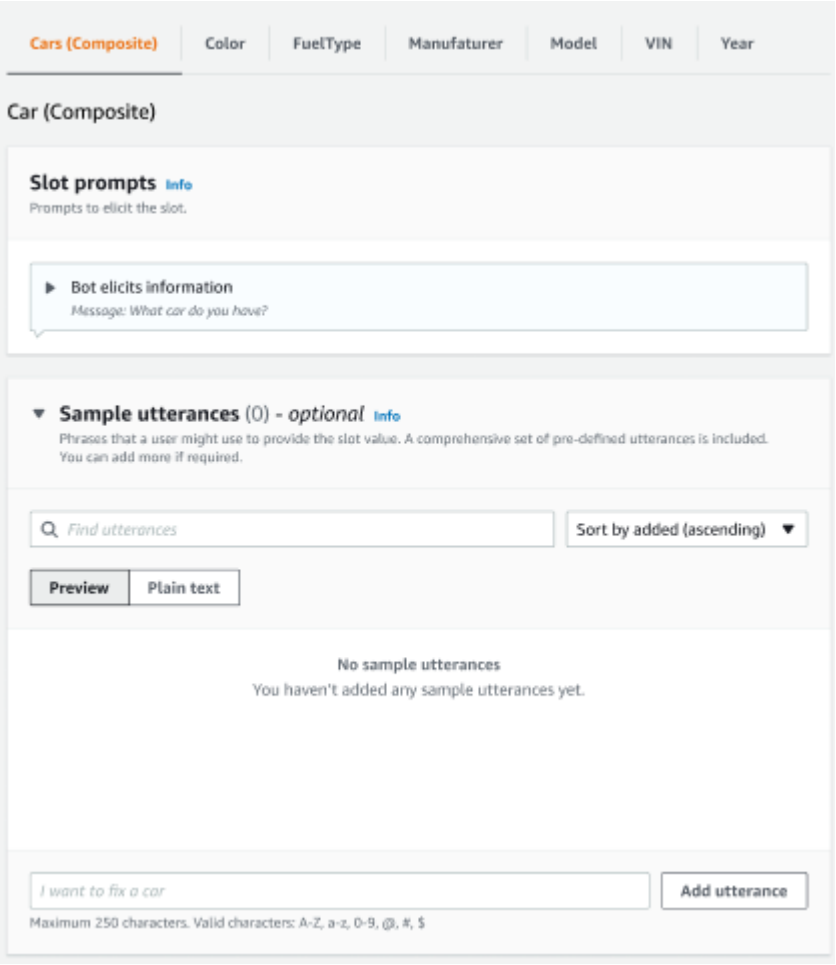
將狹槽新增至意圖時，複合槽只能作為自訂槽類型使用。

您可以在提示中使用複合槽。例如，您可以設定意圖的確認提示。

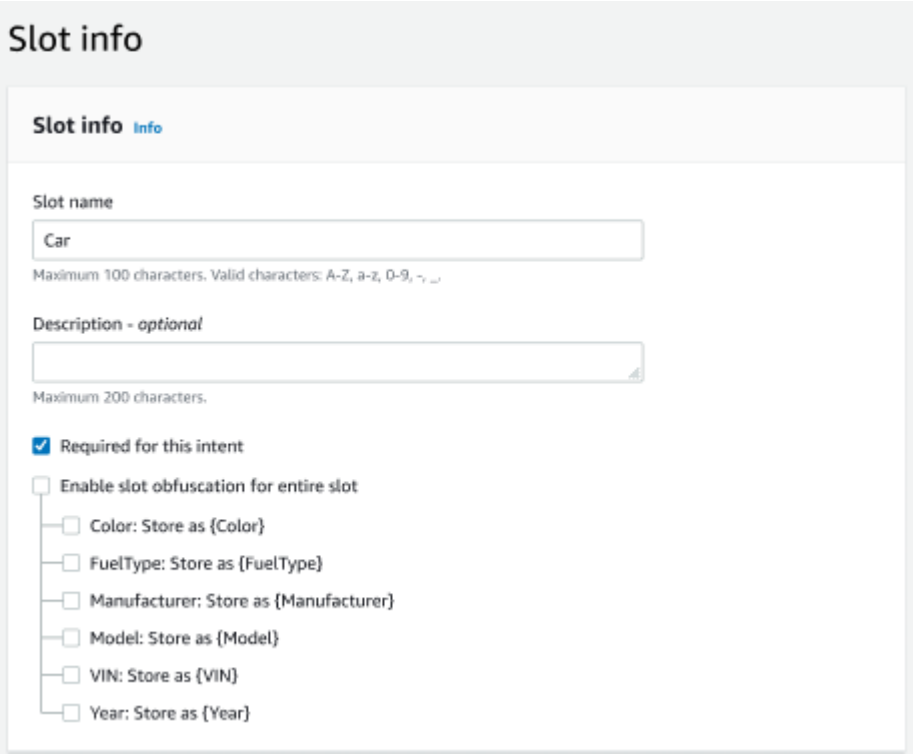
```
Would you like me to schedule service for your 2021 White Toyota Camry?
```

當 Amazon Lex 向用戶發送提示時，它會發送「您希望我為您的 2021 年白色豐田凱美瑞安排服務嗎？」

每個子插槽都配置為一個插槽。您可以加入插槽提示來引出子槽和範例語音。您可以啟用等待並繼續子槽以及預設值。如需詳細資訊，請參閱 使用預設插槽值



您可以使用插槽混淆來遮罩交談記錄中的整個複合插槽。請注意，插槽模糊化會在複合插槽層級套用，啟用時，屬於複合插槽的子插槽值會模糊化。當您混淆槽值時，每個槽值的值都會被槽的名稱取代。如需詳細資訊，請參閱隱藏交談記錄中的插槽值。



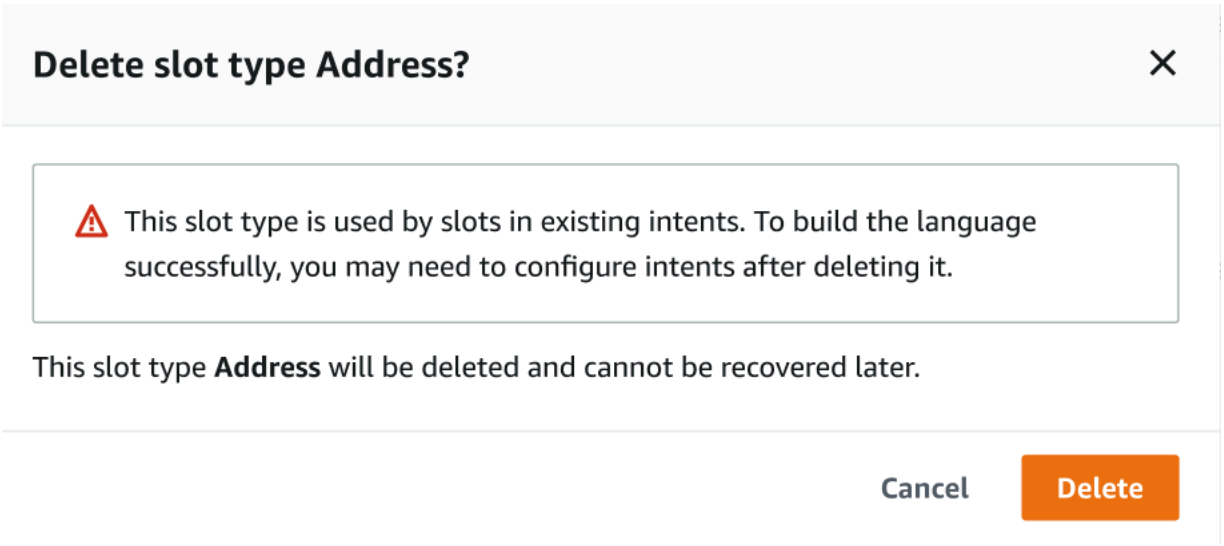
編輯複合槽類型

您可以從複合槽組態中編輯子槽，以修改子槽名稱及插槽類型。但是，當意圖正在使用複合槽時，您必須先編輯意圖，然後才能修改子槽。

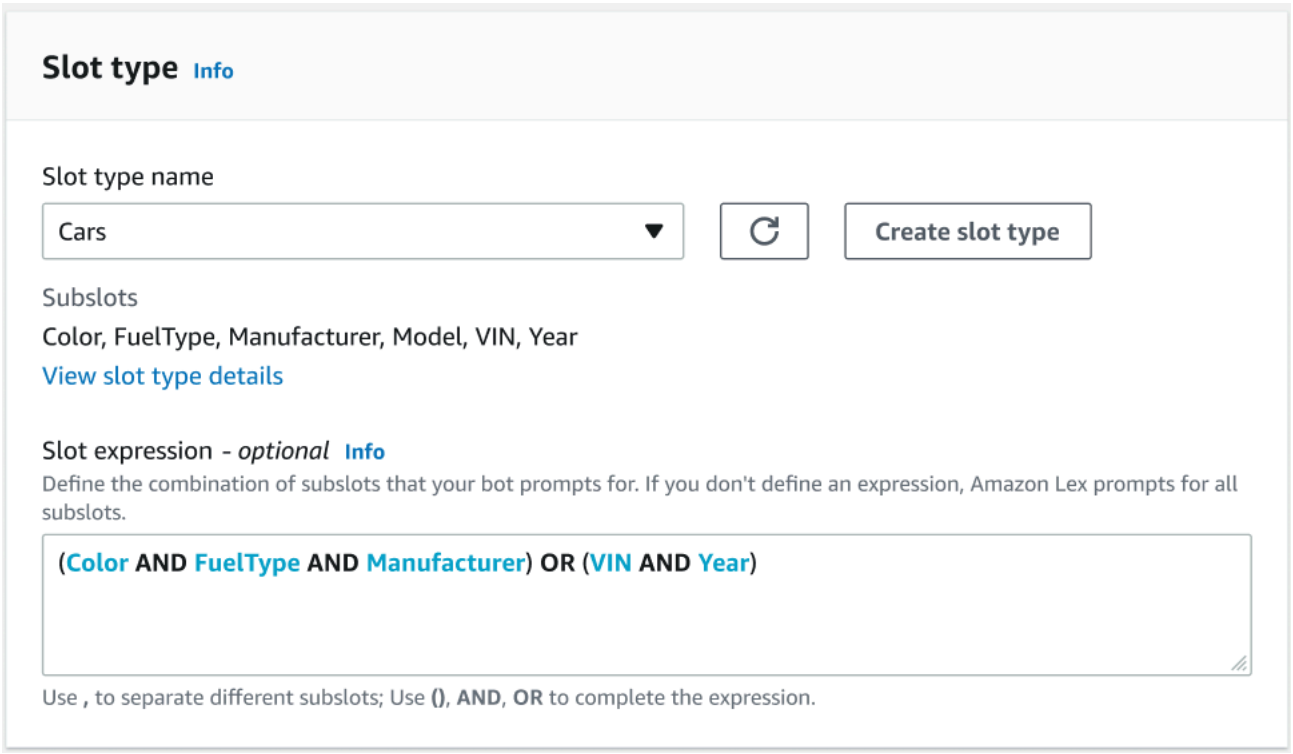
> Existing intents use this slot type. To build the language successfully, you may need to configure those intents after editing sub slots.

刪除複合槽類型

您可以從複合插槽組態中刪除子槽。請注意，當在意圖中使用子槽時，子槽仍然會從該意圖中移除。



運算式建置器中的 slot 運算式會提供警示，以通知已刪除的子槽。



使用主控台測試機器人

Amazon Lex V2 主控台包含一個測試視窗，可用來測試與機器人之間的互動。您可以使用測試視窗與機器人進行測試對話，並查看應用程式從機器人收到的回應。

您可以使用機器人執行兩種類型的測試。第一個快速測試可讓您使用建立機器人時使用的確切詞組來測試您的機器人。例如，如果您在意圖中添加了「我想摘花」的話語，則可以使用該確切的短語測試機器人。

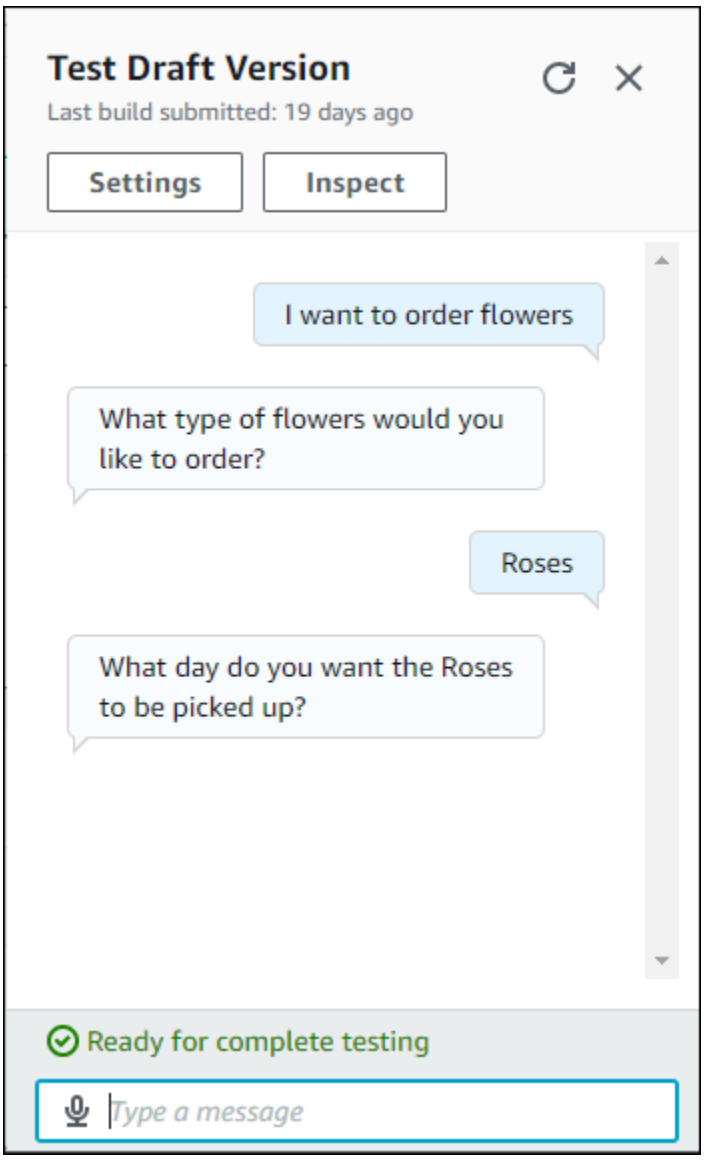
第二種類型是完整的測試，可讓您使用與您設定的話語相關的詞組來測試您的機器人。例如，您可以使用短語「我可以點鮮花」與您的機器人開始對話。

您可以使用特定的別名和語言來測試機器人。如果您正在測試機器人的開發版本，則使用`TestBotAlias`別名進行測試。

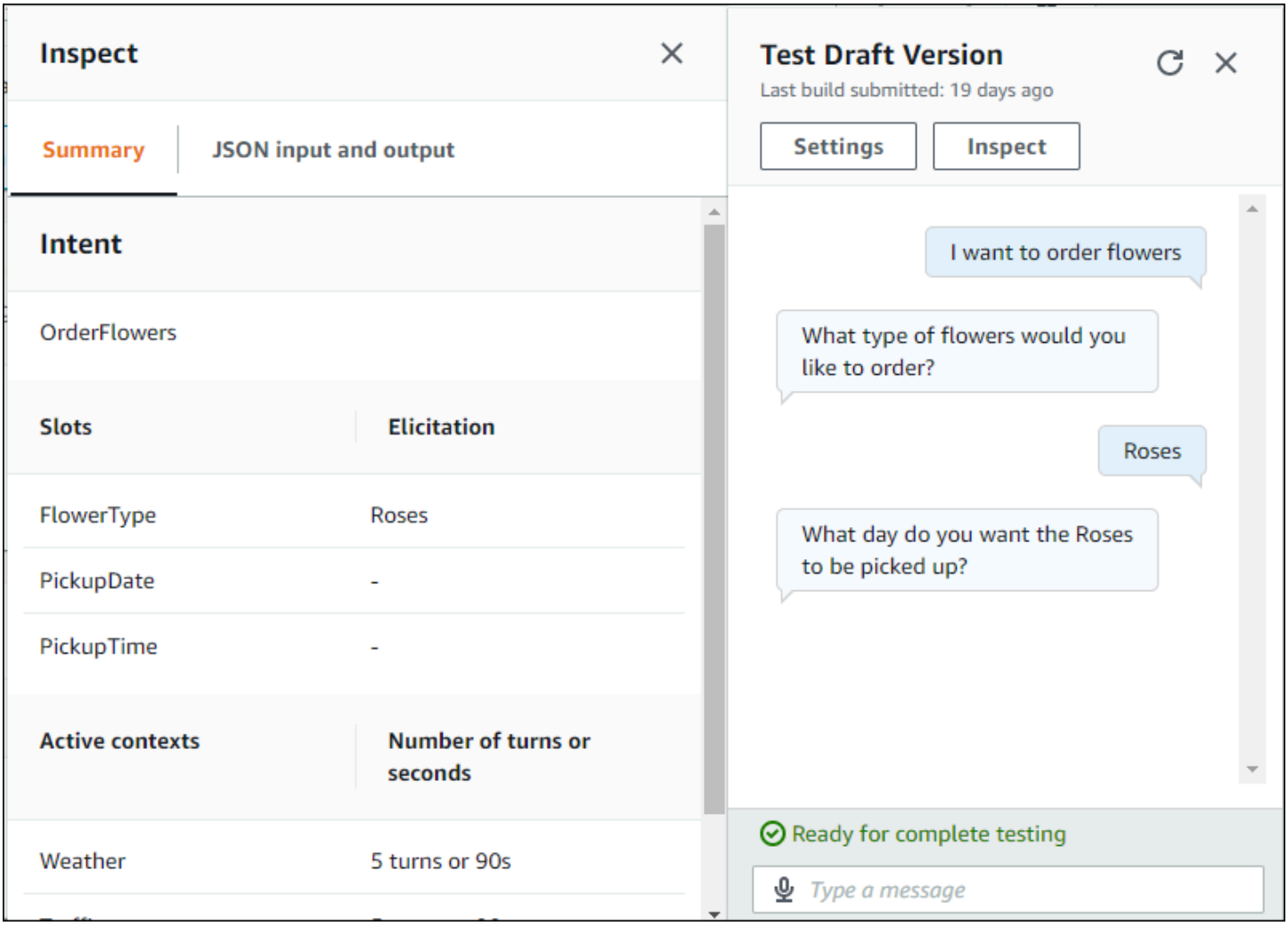
開啟測試視窗

1. 登錄到AWS Management Console並打開亞馬遜萊克斯控制台 https://console.aws.amazon.com/lex/.
2. 從機器人列表中選擇要測試的機器人。
3. 從左側選單中選擇「別名」。
4. 從別名清單中，選擇要測試的別名。
5. 從語言中，選擇要測試之語言的圓形按鈕，然後選擇測試。

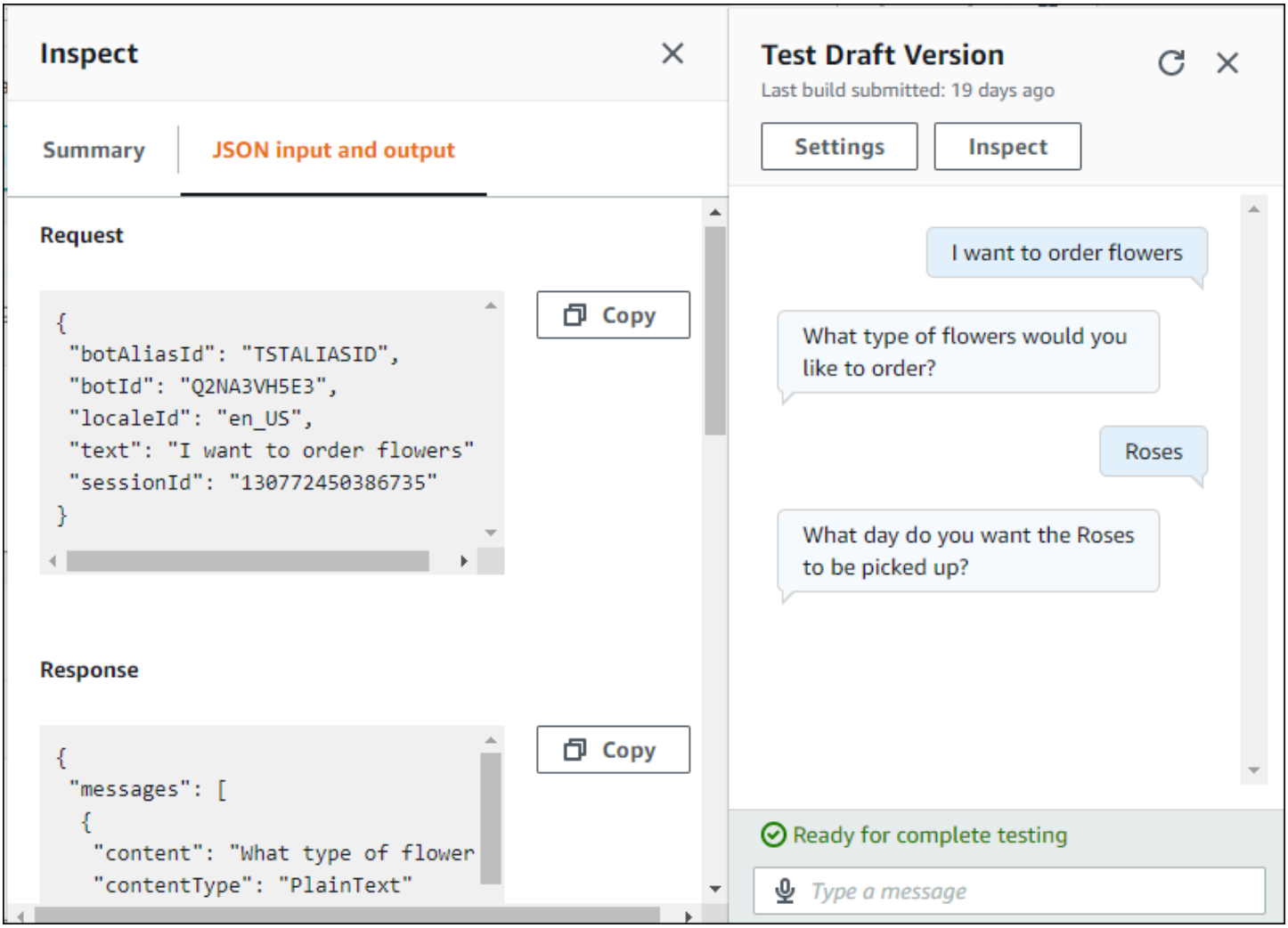
選擇「測試」之後，測試視窗會在主控台中開啟。您可以使用測試視窗與您的機器人互動，如下圖所示。



除了對話之外，您也可以在測試視窗中選擇「檢查」，以查看機器人傳回的回應。第一個檢視會顯示從機器人傳回至測試視窗的資訊摘要。



您也可以使用測試檢查視窗來查看機器人與測試視窗之間傳送的 JSON 結構。您可以看到來自測試視窗的要求，以及來自 Amazon Lex V2 的回應。


Inspect
Summary
JSON input and output
Request
Copy
{
"botAliasId": "TSTALIASID",
"botId": "Q2NA3VH5E3",
"localeId": "en_US",
"text": "I want to order flowers"
"sessionId": "130772450386735"
}
Response
Copy
{
"messages": [
{
"content": "What type of flower
"contentType": "PlainText"
Test Draft Version
Last build submitted: 19 days ago
Settings
Inspect
I want to order flowers
What type of flowers would you like to order?
Roses
What day do you want the Roses to be picked up?
Ready for complete testing
Type a message

利用生成式 AI 最佳化機器人建立與效能

> **Note**
>
> 這些功能使用生成式 AI。當您使用服務時，請記住，它可能會給出不正確或不適當的回應。如需詳細資訊，請參閱 AWS 責任 AI 政策。
> 由 Amazon 基岩提供支援： AWS 實作自動濫用偵測。由於 Amazon Lex V2 生成式人工智慧功能建置在 Amazon 基岩上，因此使用者會繼承 Amazon 基岩中實作的控制項，以強制執行人工智慧的安全性、安全性和負責任的使用。

利用 Amazon 基岩的生成人工智慧功能，自動化並加速 Amazon Lex V2 機器人建置程序。您可以在 Amazon 基岩的幫助下執行以下過程。

- 建立新的機器人，並使用自然語言說明有效地填入相關意圖和插槽類型。
- 針對您的機器人意圖自動產生範例語彙。
- 提高機器人的插槽分辨率性能。
- 建立意圖以協助回答客戶的問題。

您可以透過主控台或 API 為 Amazon Lex V2 啟動生成式人工智慧功能。

> **Note**
>
> 您必須符合下列先決條件，才能利用生成 AI 功能
>
> 1. 導覽至 Amazon 基岩主控台，然後註冊以存取您要使用的 Properpic Claude 模型 (如需詳細資訊，請參閱模型存取權)。如需使用 Amazon 基岩定價的相關資訊，請參閱 Amazon 基岩定價。
> 2. 為您的機器人地區設定開啟生成 AI 功能。若要這麼做，請依照中的步驟執行利用生成式 AI 最佳化機器人建立與效能。

Using the console

1. 登入 AWS Management Console 並開啟 Amazon Lex V2 主控台，網址為 https://console.aws.amazon.com/lexv2/home。

2. 在機器人中選取您要為其開啟生成 AI 功能的機器人和地區設定。
3. 在「生成 AI 組態」區段中，選取「設定」。
4. 針對您要啟動的每個功能切換 [已啟用] 按鈕。選取要用於該特徵的模型和版本。啟用某項功能可能會產生額外費用。如需使用 Amazon 基岩定價的相關資訊，請參閱 Amazon 基岩定價。若要深入瞭解某項功能，請從下列清單中選取對應的主題。開啟您要啟動的功能後，選取 [儲存]。會出現綠色的成功橫幅，以確認功能已開啟。

Using the API

1. 若要為新機器人啟用生成式 AI 功能，請使用此CreateBot作業來建立新的機器人。
2. 發送CreateBotLocale請求，根據需要修改`generativeAISettings`對象。如果您要為現有機器人啟用這些功能，UpdateBotLocale請改為傳送請求。

   a. 若要啟用描述性 Bot 產生器的使用，請修改`descriptiveBotBuilder`物件。指定要在`modelArn`欄位中使用的基礎模型，並將`enabled`值設定為`True`。

   b. 若要啟用改善插槽解析度，請修改`slotResolutionImprovement`物件。指定要在`modelArn`欄位中使用的基礎模型，並將`enabled`值設定為`True`。

   c. 若要啟用範例語音產生，請修改物件`sampleUtteranceGeneration`。指定要在`modelArn`欄位中使用的基礎模型，並將`enabled`值設定為`True`。

主題

- 使用描述性機器人構建器
- 話語生成
- 使用輔助插槽解析度
- 亚马逊

使用描述性機器人構建器

> **Note**
>
> 您必須符合下列先決條件，才能利用生成 AI 功能
>
> 1. 導覽至 Amazon 基岩主控台，然後註冊以存取您要使用的 Properpic Claude 模型 (如需詳細資訊，請參閱模型存取權)。如需使用 Amazon 基岩定價的相關資訊，請參閱 Amazon 基岩定價。

2. 為您的機器人地區設定開啟生成 AI 功能。若要這麼做，請依照中的步驟執行利用生成式 AI 最佳化機器人建立與效能。

描述性機器人產生器可讓您利用 Amazon Bedrock 對大型語言模型的存取，以提高機器人建立程序的效率。您使用自然語言提供提示，其中包括機器人的目的及其應執行的動作。Amazon Lex V2 利用 Amazon 基岩的功能，根據您的描述為您的機器人產生相關的意圖和插槽類型。一旦您選擇了想要保留的意圖和插槽類型，您就可以對機器人進行迭代，以將其修改為您的特定用例。描述性的機器人構建器可讓您避免手動為機器人創建意圖和插槽類型，從而節省您的時間。

英文地區設定提供描述性 Bot 建置器 (請參閱中表格開頭的en_語言環境)。亞馬遜萊克斯 V2 支援的語言和語言環境

在您建立機器人之前，請執行下列動作。

1. 檢閱中的步驟，以檢查您的角色是否具有正確的權限使用自然語言描述建立機器人所需的權限。
2. 決定要使用的描述。您可以參考以取機器人說明範例得機器人範例說明。

通過使用自然語言來描述機器人應該能夠做什麼來創建一個機器人。Amazon Lex V2 會叫用 Amazon 基岩模型，以產生適合您機器人使用案例的意圖和插槽類型。您可以使用主控台或 API 建立機器人。

Console

使用描述性機器人構建器創建機器人

1. 登入AWS Management Console並開啟 Amazon Lex V2 主控台，網址為 https://console.aws.amazon.com/lexv2/home。
2. 在機器人頁面中，選取建立機器人。
3. 對於「建立」方法，請選擇「描述性機器人建置器-GenAI」。
4. 為您的機器人提供名稱和可選說明、設定 IAM 許可，以及選擇您的機器人是否受 COPPA 約束。然後選擇下一步。
5. 選取要建立機器人的語言、機器人的語音，以及意圖分類的可信度閾值 (如需詳細資訊，請參閱使用意圖信賴度分數。
6. 在「描述性機器人產生器-GenAI」下，為您要建立的機器人提供描述。您的描述應該既詳細又精確，以幫助為您的機器人產生適當且足夠的意圖。包括動作清單，以改善意圖建立程序。

7. 在選擇模型下選擇一個模型提供者和型號。
8. 若要在其他地區設定中建立機器人，請選擇新增其他語言。完成新增語言後，請選取 [完成]。Amazon Lex V2 會建立您的機器人，描述性機器人產生意圖和插槽。產生語言環境後，橫幅會從藍色變成綠色。選取「檢閱」以查看產生的方式與插槽類型。

> **Note**
>
> 描述性 Bot 產生器目前僅提供英文語言環境。不過，您可以在建立機器人之後，將機器人複製到非英文地區設定。

查看生成的意圖和插槽類型並將其添加到您的機器人

1. 如果有足夠的意圖和插槽類型適用於您的機器人的使用案例，您可以查看生成的意圖。
   a. 檢閱產生的意圖。
      i. 選擇意圖旁邊的核取方塊，將其從要新增至機器人的意圖清單中移除。
      ii. 選擇意圖名稱以檢視針對意圖產生的「範例語音」和「槽」。
      iii. 依預設，會選取所有語音和槽。選擇核取方塊以從意圖中移除該項目。選取「新增至選取項目」(Add to Selection) 來保持核取項目在意
   b. 檢閱產生的插槽類型。
      i. 選擇插槽類型旁邊的核取方塊，將其從要新增至機器人的意圖清單中移除。
      ii. 將其添加到機器人後，您可以將其添加到插槽類型中
2. 如果您對意圖和插槽類型感到滿意，請選擇頁面頂部的添加意圖和插槽類型，以將意圖和插槽類型添加到您的機器人中。
3. 資源新增完畢後，會出現綠色的成功橫幅。轉到意圖和插槽類型以編輯生成的並添加更多值。
4. 如果「已產生的意圖」和「已產生的位置」類型大多不適用於您要建立的機器人，請執行以下步驟。
   a. 在描述性機器人構建器詳細信息部分中選擇新一代。
   b. 重新撰寫提示並選取 [重新產生] 以產生新的對應方式和插槽類型。如果您使用不同的模型，結果會有所不同。

> **Important**
>
> 不能保證會產生相同的意圖和插槽。每次重新生成意圖和槽類型時，您都會收取費用。

API

使用自然語言描述建立機器人

當您透過 API 使用描述性機器人建置器時，它會在 Amazon S3 儲存貯體的 .zip 檔案中建立機器人定義。您可以下載此檔案，然後將機器人定義匯入 Amazon Lex V2，以建立您的機器人。

1. 傳送要CreateBot求以建立新機器人。然後傳送CreateBotLocale要求以建立機器人的語言環境。
2. 傳送要StartBotResourceGeneration求，指定機器人的 ID、版本和地區設定。您可以使DRAFT用機器人版本。在欄位中提供您的提generationInputPrompt示。您的描述應該既詳細又精確，以幫助為您的機器人產生適當且足夠的意圖。包括動作清單，以改善意圖建立程序。
3. 注意回應generationId中的。
4. 使用您在響應中收generationId到的發送DescribeBotResourceGeneration請StartBotResourceGeneration求。包括機器人 ID、版本和地區設定。
5. 如果DescribeBotResourceGeneration回應generationStatus中的是Complete，也會填入generatedBotLocaleUrl欄位。按照下載物件中的步驟，使用此 Amazon S3 URI 下載機器人定義。

檢查生成的機器人定義並將其導入

1. 依照下載物件generationStatus中的步驟，使用DescribeBotResourceGeneration回應中的 Amazon S3 URI 下載機器人定義。
2. 您可以編輯檔案，直接修改機器人特定使用案例所產生的內容。您也可以傳送另一個StartBotResourceGeneration要求來重新產生意圖和插槽。

> **Important**
>
> 不能保證會產生相同的意圖和插槽。每次重新生成意圖和槽類型時，您都會收取費用。

3. 若要匯入機器人定義，請依照中的步驟執行匯入中。
4. 匯入之後，您可以使用、和UpdateSlotType操作來修改產生的UpdateIntent意圖和槽。UpdateSlot

若要列出有關機器人地區設定之所有產生項目的中繼資料，請使用此ListBotResourceGenerations作業。使用`DescribeBotResourcGeneration`請求中的任何傳回`generationId`值來擷取產生的機器人定義的 Amazon S3 URI。

主題

- 機器人說明範例
- 使用自然語言描述建立機器人所需的權限

機器人說明範例

Industry	範例提示
金融服務	我們是一種金融卡服務，可幫助用戶在收到新卡時執行任務，例如激活卡，發送電子郵件或發送 PIN 碼，驗證新卡（使用郵政編碼）。我們也會協助他們完成與現有信用卡相關的工作，例如查詢信用卡優惠、報失信用卡、申請新卡、重設信用卡密碼或支付卡帳單。
食品服務	我想要一個機器人來幫助客戶訂購食物（使用項目 ID，數量，大小），檢查訂單狀態並取消訂單。使用訂單 ID 編製訂單索引。
航空	我們是一個航空公司域名，可幫助用戶預訂機票，查看預訂詳細信息，獲取預訂航班的收據，查詢航班狀態，重新安排預訂的航班，提出航班

Industry	範例提示
	詳細信息以及取消預訂的航班。如果有助於支援網域說明中的功能，您也可以產生其他意圖。
保險	目標：我們是一家銷售汽車，家居和年金保險的保險公司。我想要一個可以檢查索賠狀態，提出索賠，支付保單和取消政策的機器人。我希望機器人至少具有以下意圖和插槽：身份驗證-政策_id，last4SSN 政策類型：汽車，家庭，年金政策狀態：檢查餘額，檢查到期日，檢查覆蓋付款：一次性付款，分期付款，金額
車輛管理	我們正在構建一個拖車查找機器人，該機器人可以幫助已拖曳汽車的城市中的駕駛員找到汽車的位置。該機器人應該詢問汽車被拖的地址或位置，以及有關車輛的詳細信息，例如車牌和品牌，型號和汽車年份。機器人應回覆拖曳停車場的位置以及營業時間。
旅行	我是旅行社，我想要一個機器人來幫助我的客戶預訂迪士尼之旅。迪士尼在世界各地有幾個公園可供選擇，還有可以預訂的酒店，餐飲和特殊娛樂場所。機器人的用戶應該能夠修改或取消他們的預訂。預訂必須至少包括公園，日期和酒店。包括餐飲或娛樂是可選的，可以在以後添加或更改。

使用自然語言描述建立機器人所需的權限

- 若要在 Amazon Lex V2 主控台上存取此功能，請確保您的主控台角色具有`bedrock:ListFoundationModels`權限。
- 與機器人關聯的 IAM 角色應具有`bedrock:InvokeModel`權限。當您透過 Amazon Lex 主控台啟用此功能時，政策會自動新增至機器人角色，前提是您的機器人使用 Amazon Lex 產生的服務連結角色。

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
      {
          "Effect": "Allow",
          "Action": [
              "bedrock:InvokeModel"
          ],
          "Resource": [
              "arn:aws:bedrock:region::foundation-model/model-id"
          ]
      }
  ]
}
```

話語生成

> **Note**
>
> 您必須符合下列先決條件，才能利用生成 AI 功能
>
> 1. 導覽至 Amazon 基岩主控台，然後註冊以存取您要使用的 Properpic Claude 模型 (如需詳細資訊，請參閱模型存取權)。如需使用 Amazon 基岩定價的相關資訊，請參閱 Amazon 基岩定價。
> 2. 為您的機器人地區設定開啟生成 AI 功能。若要這麼做，請依照中的步驟執行利用生成式 AI 最佳化機器人建立與效能。

使用語音產生可根據您的意圖自動建立範例語音。Amazon Lex V2 不會手動輸入範例語音，而是根據意圖名稱、說明和現有的樣本語言來為您產生範例語音，這樣您就可以減少探索和撰寫自己的範例語音所花費的時間和精力。在 Amazon Lex V2 產生語音之後，您可以編輯和刪除語音內容。使用此工具可加速建立意圖辨識程序的範例語音。

若要允許產生話語，請遵循中的步驟利用生成式 AI 最佳化機器人建立與效能來啟動生成 AI 功能。

您可以使用控制台或 API 生成語音。

Console

1. 導航到機器人中任何意圖的「示例語言」部分（在「視覺化對話構建器」中，它位於「開始」塊中）。
2. 選取「產生語音」按鈕以產生 5 個範例語音。如果您的意圖具有超過 25 個範例語音，「產生話語」按鈕會停用。
3. 產生的話語會以綠色橫幅顯示，以區分產生的話語與現有的話語。
4. 將游標暫留在語音上，以顯示編輯、刪除和排序產生的話語的選項。

API

1. 傳送要GenerateBotElement求，填入您要產生範例語言的意圖和機器人 ID、版本和地區設定。
2. 回應會傳回物件清單，每個 SampleUtterance物件都包含產生的語音。
3. 要將語音添加到意圖中，請發送UpdateIntent請求並將語音添加到字段中。`sampleUtterances`

主題

- 語音產生的權限

語音產生的權限

若要在 Amazon Lex V2 主控台上存取此功能，請確保您的主控台角色具有`bedrock:ListFoundationModels`並具有`bedrock:InvokeModel`許可。

使用輔助插槽解析度

> **Note**
>
> 您必須符合下列先決條件，才能利用生成 AI 功能
>
> 1. 導覽至 Amazon 基岩主控台，然後註冊以存取您要使用的 Properpic Claude 模型 (如需詳細資訊，請參閱模型存取權)。如需使用 Amazon 基岩定價的相關資訊，請參閱 Amazon 基岩定價。

2. 為您的機器人地區設定開啟生成 AI 功能。若要這麼做，請依照中的步驟執行利用生成式 AI 最佳化機器人建立與效能。

您可以通過使用輔助插槽分辨率來提高機器人對話流程中某些內置插槽的準確性。輔助插槽解析使用 Amazon 基岩大型語言模型 (LLM) 來改善對某些內建插槽的辨識，進而改善插槽引出期間客戶回應的解釋。對於無法正常解決的話語，Amazon Lex 將嘗試使用 Amazon 基岩第二次解決這些語音。

輔助插槽解析度可讓您使用 Amazon 基礎模型的強大功能，以提高下列內建插槽的準確性：

- `AMAZON.Alphanumeric`沒有正則表達式
- `AMAZON.City`
- `AMAZON.Country`
- `AMAZON.Date`
- `AMAZON.Number`
- `AMAZON.PhoneNumber`
- `AMAZON.Confirmation`

您可以針對使用上述所列內建插槽的任何意圖啟用輔助插槽解析度。輔助插槽解析不適用於上面未列出的自訂插槽或 Amazon 內建插槽。

在 Amazon Lex 機器人中啟用協助插槽解析後，您可以使用交談日誌和指標，收集有關準確性改進的資料。

- 對話日誌-如果使用 Amazon 基岩來解析插槽`Bedrock`，則解釋將具有 `interpretationSource` as。
- CloudWatch 度量-量度將在量度中 CloudWatch 列出的維度下發佈。若要進一步了解，請參閱使用 Amazon 監控 Amazon Lex CloudWatch。

若要使用描述性 Bot 產生器，請遵循中的步驟，確保您的 IAM 角色具有適當的許可協助插槽解析的權限。

主題

- 輔助插槽解析度範例
- 在生成式 AI 配置畫面中啟用輔助插槽解析度
- 在插槽設定中啟用輔助插槽解析度

- 協助插槽解析的權限

輔助插槽解析度範例

以下是一些輔助插槽解析度能夠智慧地將使用者話語解析為值的範例。

亞馬遜。數

Vertical (垂直)	插槽類型	slotName	插槽提示	發聲	已解決的值
旅行	亞馬遜。數	numberOfNights留下來	你在旅行中住了多少晚？	一整星期七晚	7
銀行業	亞馬遜。數	numberOfPeopleOnTheAccount	帳戶中有多少人？	我和我的妻子	2
旅行	亞馬遜。數	numberOfStops	有多少站？	曾經在日本。一旦在洛杉磯。	2

亞馬遜。 AlphaNumeric

Vertical (垂直)	插槽類型	slotName	插槽提示	發聲	已解決的值
汽車租賃	亞馬遜。英數字	交易 ID	您的交易 ID 是什麼？	我相信這是阿爾法威士忌迴聲八三四九羅密歐朱麗葉.	AWE8349RJ
旅行	亞馬遜。英數字	確認碼	您的預訂確認號碼是什麼？	確認號碼為 BLT2UE。	布爾特

亞馬遜。日期

Vertical (垂直)	插槽類型	slotName	插槽提示	發聲	已解決的值	目前日期
汽車租賃	亞馬遜。日期	杜埃達特	租賃合約何時到期？	租約是在下個月的 1 日。	2023-12-01	2023-11-09
旅行	亞馬遜。日期	返回日期	你什麼時候回來？	今天晚些時候大約 7.	2023-11-09	2023-11-09

亞馬遜。 PhoneNumber

Vertical (垂直)	插槽類型	slotName	插槽提示	發聲	已解決的值
保險	亞馬遜。PhoneNumber	保單持有人	保單持有人的電話號碼是什麼？	保單持有人的電話號碼為 123-456-7890。	1234567890
零售	亞馬遜。PhoneNumber	電話查詢	您的電話號碼是什麼，我可以找到您的帳戶？	我認為這是在 413-570-9617 之下，讓我仔細檢查一下。	4135709617

亞馬遜國家

Vertical (垂直)	插槽類型	slotName	插槽提示	發聲	已解決的值
旅行	亞馬遜國家	本地文化	您的原籍國是什麼？	我是印度人。	印度

Vertical (垂直)	插槽類型	slotName	插槽提示	發聲	已解決的值
銀行業	亞馬遜國家	國家行程	您將使用借記卡前往哪些國家？	我將前往新德里。	印度

亚马逊城市

Vertical (垂直)	槽類型	意圖	問題	回應	已解決的值
保險	亚马逊城市	policyHolderCity	保單持有人居住在哪個城市？	我住在斯普林菲爾德	斯普林菲爾德
旅行	亚马逊城市	目的地城市	您要前往哪個城市？	我要去东京。	東京

亞馬遜。確認

Vertical (垂直)	插槽類型	slotName	插槽提示	發聲	已解決的值
保險	亞馬遜。確認	政策已過期	保單是否已過期？	是的，不幸的是它已經過期。	是
銀行業	亞馬遜。確認	哈斯投資	您有任何投資嗎？	我還沒有投資任何東西。	否

在生成式 AI 配置畫面中啟用輔助插槽解析度

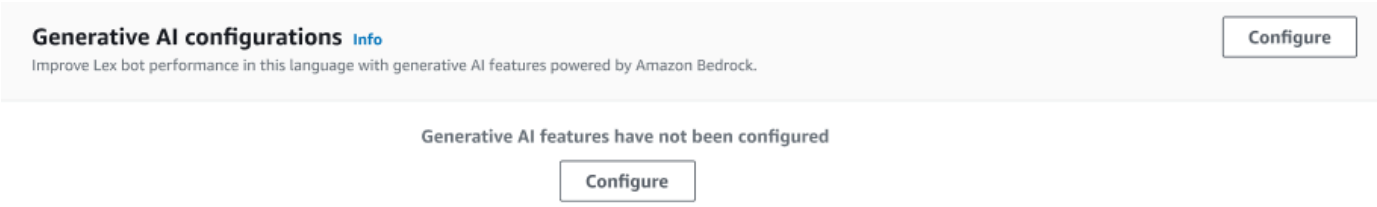
您可以瀏覽至生成 AI 畫面，為支援的內建插槽啟用輔助插槽解析度。

如果插槽是支援的內建插槽，您可以選擇在插槽層級啟動輔助插槽解析度。

1. 登入AWS Management Console並開啟 Amazon Lex V2 主控台，網址為 https://console.aws.amazon.com/lexv2/home。
2. 在「機器人」下的導航窗格中，選擇要用於輔助插槽解析的機器人。
3. 選取您要啟用的機器人的語言英文 (美國)。
4. 前往畫面上的「生成 AI 設定」區段。
5. 如果功能尚未啟用，請選取 [前往 Amazon 基岩] 以註冊並啟用此功能。

> **Note**
>
> 如果您無法存取 Amazon 基岩基礎模型，您應該看到「前往 Amazon 基岩」。按一下「前往 Amazon 基岩」以前往 Amazon 基岩頁面，您可以在其中註冊以存取基礎模型。輔助插槽分辨率目前支持克勞德 V2 和克勞德即時 V1。我們建議使用克勞德 V2 以獲得最佳效果。

6. 如果您已經可以存取基岩基礎模型，您應該會看到「設定」按鈕。按一下此按鈕即可前往生成 AI 組態頁面，以啟動 Lex 中的生成式 AI 功能。



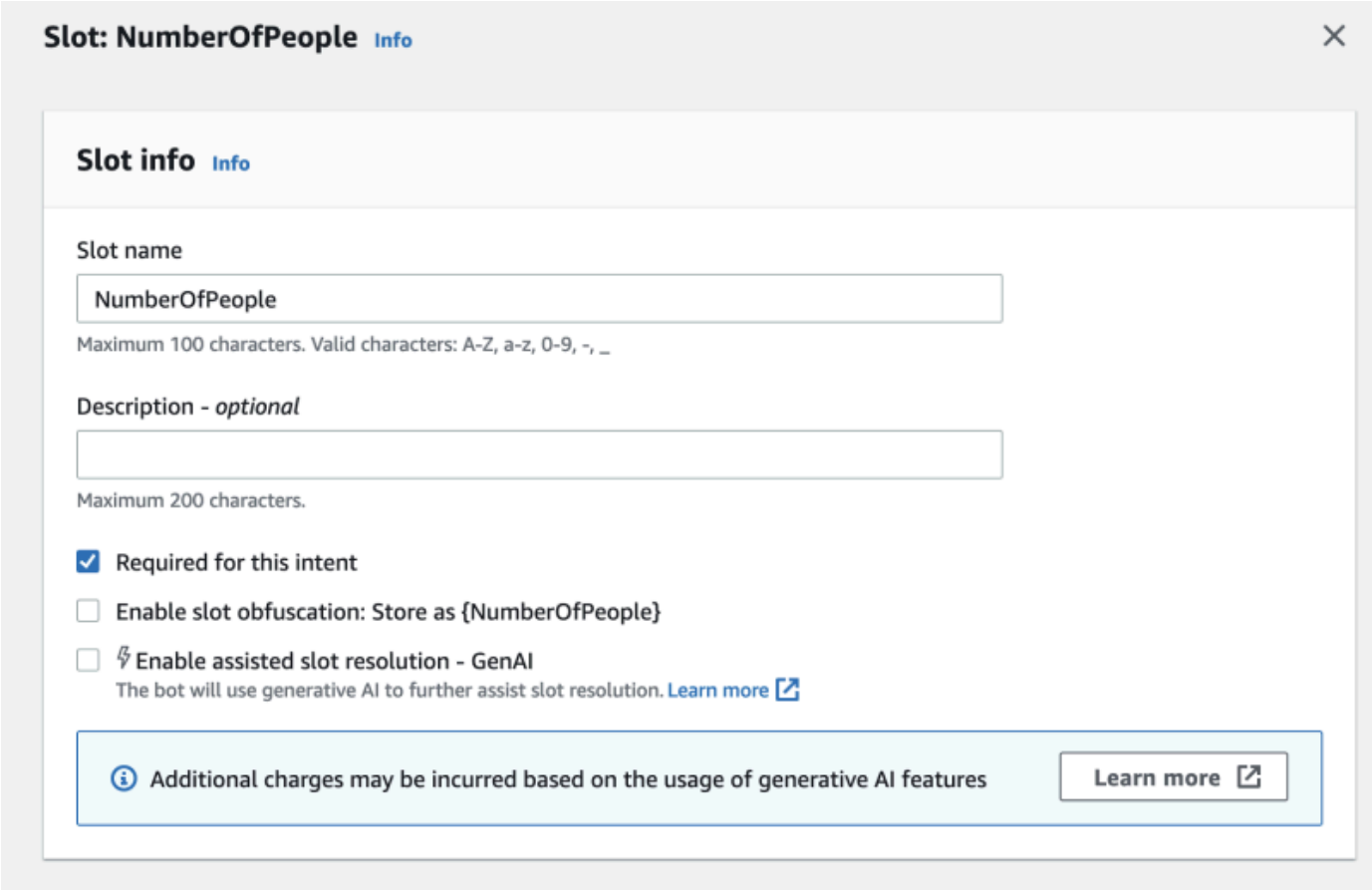
7. 在方塊的右上角，將滑桿向右移動以選擇 [已啟用] 設定。
8. 選擇啟用按鈕以啟用所選插槽的輔助插槽解析度。
9. 您可以從清單中選取插槽並選取 [停用] 按鈕來停用輔助插槽解析度。

在插槽設定中啟用輔助插槽解析度

您可以為支援的內建插槽啟用輔助插槽解析度，方法是導覽至具有插槽的每個意圖的插槽層級。插槽必須是上面列出的其中一個支援的內建插槽，才能選擇啟動輔助插槽解析度。如果插槽沒有啟動輔助插槽解析度的選項，則該選項將顯示為灰色。

> **Note**
>
> 您必須先啟動生成 AI 面板上的輔助插槽解析功能，才能啟動個別插槽的功能。

1. 登入 AWS 管理主控台，然後開啟 Amazon Lex V2 主控台，網址為 https://console.aws.amazon.com/lexv2/home。
2. 在「機器人」下的導航窗格中，選擇要用於輔助插槽解析的機器人。
3. 在 [所有語言] 下，選取 [英文 (美國)] 展開清單。
4. 在左側面板中，選擇「意圖」以檢視所選機器人中的意圖清單。
5. 在「意圖」畫面中，選擇包含您要修改之插槽的意圖。
6. 選取意圖名稱以檢視該意圖的狹槽。
7. 選取 [插槽] 區段中的 [進階選項] 按鈕。
8. 選取 [啟用輔助插槽解析度] 核取方塊以啟用此功能。



9. 選擇屏幕右下角的更新插槽按鈕。這將為您選擇的插槽激活輔助插槽分辨率。

您可以透過進行 API 呼叫，為支援的內建插槽啟用輔助插槽解析度。

- 請遵循中的步驟，為您的機器人地區設定啟用輔助插槽解析。利用生成式 AI 最佳化機器人建立與效能

- 傳送要UpdateSlot求，指定您要啟用輔助插槽解析的插槽。在slotResolutionSetting欄位中，將slotResolutionStrategy值設定為EnhancedFallback。若要建立啟用輔助插槽解析度的新插槽，CreateSlot請改為傳送要求。

協助插槽解析的權限

- 若要在 Amazon Lex V2 主控台上存取此功能，請確保您的主控台角色具有bedrock:ListFoundationModels權限。
- 與機器人關聯的 IAM 角色應具有bedrock:InvokeModel權限。當您透過 Amazon Lex 主控台啟用此功能時，政策會自動新增至機器人角色，前提是您的機器人使用 Amazon Lex 產生的服務連結角色。

```
{
  "Version": "2012-10-17",
  "Statement": [
      {
          "Effect": "Allow",
          "Action": [
              "bedrock:InvokeModel"
          ],
          "Resource": [
              "arn:aws:bedrock:Region::foundation-model/modelId"
          ]
      }
  ]
}
```

亚马逊

> **Note**
>
> 您必須符合下列先決條件，才能利用生成 AI 功能
>
> 1. 導覽至 Amazon 基岩主控台，然後註冊以存取您要使用的 Properpic Claude 模型 (如需詳細資訊，請參閱模型存取權)。如需使用 Amazon 基岩定價的相關資訊，請參閱 Amazon 基岩定價。

2. 為您的機器人地區設定開啟生成 AI 功能。若要這麼做，請依照中的步驟執行利用生成式 AI 最佳化機器人建立與效能。

您可以利用 Amazon 基岩 FM 在機器人對話中回答客戶的問題。Amazon Lex V2 提供您可以新增至機器人的內建內建功AMAZON.QnAIntent能。此意圖透過辨識客戶問題並從下列知識商店搜尋答案 (例如)，藉此利用 Amazon Bedrock 的生成式 AI 功能。**Can you provide me details on the baggage limits for my international flight?**此功能可減少使用 Amazon Lex V2 意圖中的工作導向對話來設定問題和答案的需求。此意圖也會根據交談歷程記錄辨識後續問題 (例如**What about domestic flight?**)，並相應地提供答案。

請遵循中的步驟，確保您的 IAM 角色具有存取的適當許可的權限 AMAZON.QnAIntent。AMAZON.QnAIntent

若要利用，AMAZON.QnAIntent您必須設定下列其中一個知識庫。

- Amazon OpenSearch 服務資料庫 — 如需詳細資訊，請參閱建立和管理 Amazon OpenSearch 服務網域。
- Amazon Kendra 索引 — 如需詳細資訊，請參閱建立索引。
- Amazon 基岩知識庫 — 如需詳細資訊，請參閱建立知識庫。

您可以使用下列AMAZON.QnAIntent兩種方式之一來設定：

使用生成 AI 組態進行設定

1. 在 Amazon Lex V2 主控台中，從左側導覽窗格中選取「機器人」，然後從「機器人」區段選擇要為其新增意圖的機器人。
2. 在左側導覽窗格中，選取您要新增意圖的語言。
3. 在「生成 AI 組態」區段中，選取「設定」。
4. 在 QnA 組態區段中，選取建立 QnA 意圖。

通過向機器人添加內置意圖進行設置

1. 在 Amazon Lex V2 主控台中，從左側導覽窗格中選取「機器人」，然後從「機器人」區段選擇要為其新增意圖的機器人。
2. 在左側導覽窗格中，選取您要新增意圖的語言下方的 [意圖]。

3. 選取 [新增意圖]，然後從下拉式功能表中選擇 [使用內建的
4. 如需有關的組態的更多詳細資訊AMAZON.QnAIntent，請參閱AMAZON.QnAIntent。

> **Note**
>
> 當語音未分類到機器人中存在的任何其他意圖時，將啟動。AMAZON.QnAIntent當語音未分類到機器人中存在的任何其他意圖時，此意圖將被激活。請注意，當引出插槽值時，不會針對遺漏的話語啟動此意圖。識別後，就會AMAZON.QnAIntent使用指定的 Amazon 基岩模型來搜尋設定的知識庫並回應客戶問題。

主題

- 的權限 AMAZON.QnAIntent

的權限 AMAZON.QnAIntent

若要在 Amazon Lex V2 主控台上存取此功能，請確保您的主控台角色具有bedrock:ListFoundationModels許可。

與機器人關聯的 IAM 角色應具有的以下所需許可AMAZON.QnAIntent。機器人角色應具有呼叫的權限bedrock:InvokeModel。您還應該為您在機器人中指定的每個數據存儲附上一個聲明AMAZON.QnAIntent（請參閱下面政策中的Permissions to access Amazon Kendra indexPermissions to access OpenSearch Service index、和Permissions to access knowledge base in Amazon Bedrock聲明）。當您透過 Amazon Lex 主控台啟用此功能時，政策會自動新增至機器人角色，前提是您的機器人使用 Amazon Lex 產生的服務連結角色。

```
{
  "Version": "2012-10-17",
  "Statement": [
      {
          "Sid": "Permissions to invoke Amazon Bedrock foundation models",
          "Effect": "Allow",
          "Action": [
              "bedrock:InvokeModel"
          ],
          "Resource": [
              "arn:aws:bedrock:region::foundation-model/model-id"
          ]
```

```
      },
      {
        "Sid": "Permissions to access Amazon Kendra index",
        "Effect": "Allow",
        "Action": [
            "kendra:Query",
            "kendra:Retrieve"
        ],
        "Resource": [
            "arn:aws:kendra:region:account-id:index/kendra-index"
        ]
    },
    {
        "Sid": "Permissions to access OpenSearch Service index",
        "Effect": "Allow",
        "Action": [
            "es:ESHttpGet",
            "es:ESHttpPost"
        ],
        "Resource": [
            "arn:aws:es:region:account-id:domain/domain-name/index-name/_search"
        ]
    },
    {
        "Sid": "Permissions to access knowledge base in Amazon Bedrock",
        "Effect": "Allow",
        "Action": [
            "bedrock:Retrieve"
        ],
        "Resource": [
            "arn:aws:bedrock:region:account-id:knowledge-base/knowledge-base"
        ]
    }
  ]
}
```

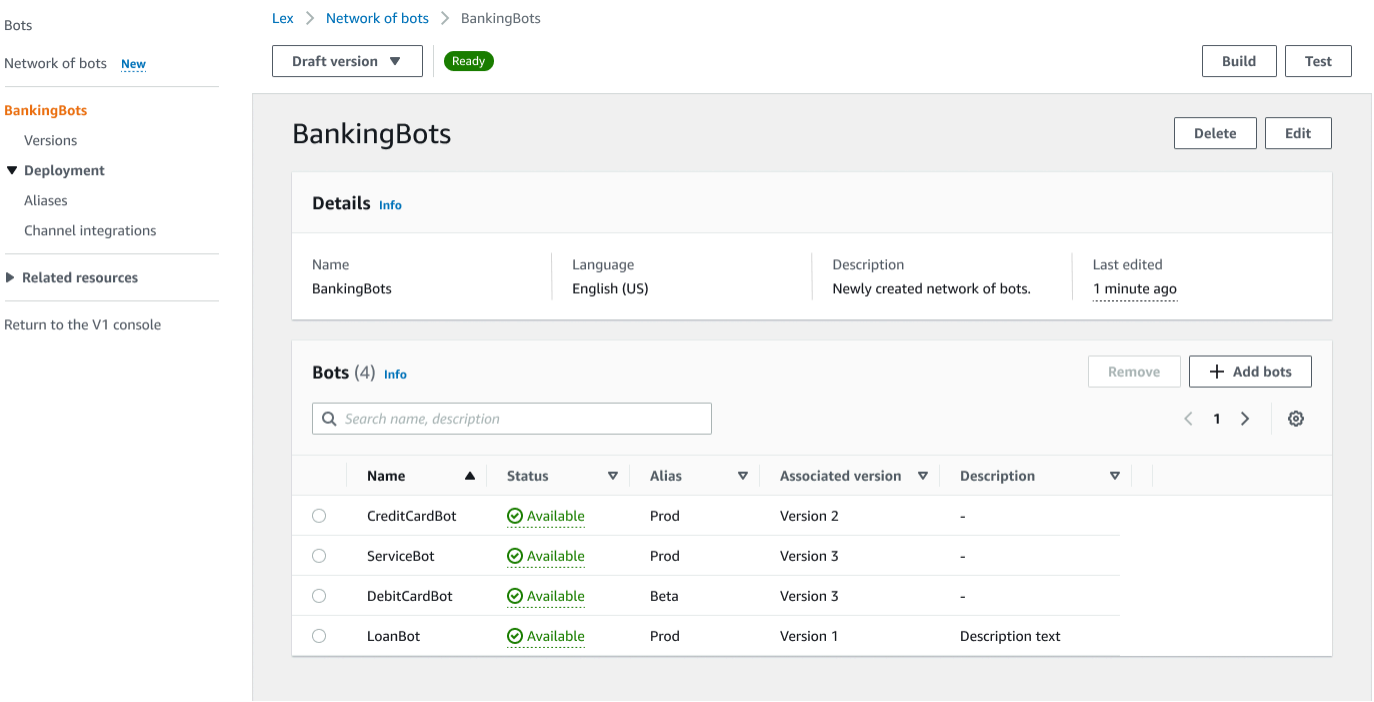
建立機器人網路

機器人網路讓企業能夠跨多個機器人提供統一的使用者體驗。透過機器人網路，企業可以將多個機器人新增至單一網路，以實現靈活且獨立的機器人生命週期管理。網路會向使用者公開單一整合介面，並根據使用者輸入將要求路由至適當的機器人。

團隊可以通過在改進的機器人部署到生產環境中時，通過維護並將機器人添加到網絡中，來共同創建機器人網絡以滿足各種業務需求。開發人員可以將多個機器人整合到單一網路中，以簡化並加速部署和改進。

機器人網路目前僅提供 en-US 語言版本。

> **Note**
>
> 目前，機器人網絡僅限於一個帳戶。您無法從其他帳戶新增機器人。



建立機器人網路

登入AWS Management Console並開啟亞馬遜萊克斯 V2 主控台，網址為 https://console.aws.amazon.com/lexv2/home。從側面菜單中選擇機器人網絡。您必須建立至少一個機器人才能建立機器人網路。

步驟 1：配置機器人設置網絡

1. 在「詳細資料」區段中，輸入您的網路名稱，並為其提供選擇性描述。
2. 在 IAM 許可區段中，選擇提供 Amazon Lex V2 存取其他AWS服務 AWS Identity and Access Management (例如 Amazon) 權限的 (IAM) 角色CloudWatch。您可以讓 Amazon Lex V2 建立角色，也可以選擇具有CloudWatch許可的現有角色。如需詳細資訊，請參閱 Amazon Lex V2 的身分識別和存取管理。
3. 在《兒童線上隱私保護法》(COPPA) 區段中，選擇適當的回應。DataPrivacy如需詳細資訊，請參閱。
4. 在閒置工作階段逾時區段中，選擇 Amazon Lex V2 在使用者開啟的情況下保持工作階段的持續時間。Amazon Lex V2 會在工作階段期間維護工作階段變數，以便您的機器人可以使用相同的變數繼續交談。如需詳細資訊，請參閱設定工作階段逾時。
5. 在「新增語言設定」區段中，選擇機器人與使用者互動的語音。您可以在語音樣本中輸入片語，然後選擇播放以聆聽聲音。
6. 在「進階設定」區段中，選擇性地新增可協助識別機器人的標籤。標籤可用於控制存取和監視資源。如需詳細資訊，請參閱標記資源。
7. 選擇「下一步」以創建機器人網絡並轉到添加漫遊器。

步驟 2：新增機器人

1. 在 [機器人] 區段中，選取 [+ 新增機器人]。
2. 將彈出一個添加機器人模式。從「機器人」下拉式功能表中選擇要新增的機器人，並從別名下拉式選單中選擇要使用的機器人別名。

   別名必須指向機器人的編號版本，而不是草稿版本。您最多可以添加 5 個機器人。一個機器人可以添加到多達 25 個不同的網絡。
3. 選擇 + 添加機器人以將更多漫遊器添加到您的網絡中。若要移除機器人，請在您要移除的機器人旁選取 [移除]。添加完漫遊器後，請選擇「保存」以關閉強制回應。
4. 選取 [儲存] 以完成網路建立。

管理您的機器人網路

創建您的機器人網絡後，您將被帶到一個頁面，您可以在其中管理和構建網絡。或者，您可以通過在側邊菜單中選擇機器人網絡，然後選擇要管理的網絡名稱來訪問此頁面。

1. 若要編輯網路的資訊，請選取 [詳細資料] 區段上方的 [編輯]。若要刪除網路，請選取「詳細資料」區段上方的刪除。
2. 在「機器人」部分中，您可以通過選擇 + 添加漫遊器來添加更多機器人。如果您導覽至 Amazon Lex V2 主控台側邊功能表中的「機器人」頁面，也可以新增機器人。切換您要新增的機器人旁邊的選項按鈕，然後從動作下拉式功能表中選擇新增至機器人網路。

   從彈出的模式中的機器人網絡下拉菜單中，選擇要添加機器人的網絡。然後從「機器人別名」下拉式選單中選擇您要使用的機器人別名。選取 [新增]，將機器人新增至您選擇的網路。
3. 您可以通過切換機器人旁邊的單選按鈕並選擇刪除來從網絡中刪除漫遊器。
4. 完成網路設定後，請選取右上角的 [建置] 以建立網路。建置可能需要幾分鐘的時間。如果建置成功，頁面頂端會出現綠色的成功橫幅。
5. 建立網絡後，您可以選擇右上角的「測試」，以便在右下角顯示聊天窗口。您可以使用此聊天窗口與網絡的機器人進行交談，並確保正確配置了對話流程和轉換。

> **Note**
>
> 如果您在網路中新增、移除或更新機器人，則必須重建網路。

版本

您可以創建不同版本的機器人網絡。若要管理版本，請從 Amazon Lex V2 主控台的側邊功能表選擇您的網路，然後選取版本。

1. 選取 [建立版本] 以建立機器人網路的新版本。您可以添加一個可選的描述。選擇「建立」以建立版本。
2. 當您切換機器人網路版本旁邊的選項按鈕時，您可以選取將別名與版本相關聯，將別名指向此版本。
3. 若要管理網路的某個版本，請在「版本」區段中選取版本名稱。在下面的頁面中，您可以編輯版本的詳細信息，並管理版本及其關聯別名中的機器人。

別名

您可以使用別名來部署網路。若要管理別名，請從 Amazon Lex V2 主控台的側邊功能表選擇您的網路，然後選取別名。

1. 選取「建立別名」以建立新別名。
2. 在別名詳細資料區段中為別名指定名稱和選用說明。您可以選擇要將別名與「與版本關聯」區段產生關聯的版本，並在「標籤」區段中新增標籤。選擇「建立」以建立別名。
3. 若要管理網路的別名，請在「別名」區段中選取別名的名稱。在下一頁中，您可以編輯別名的詳細資訊，並管理其標籤、頻道整合和資源型政策。您也可以檢視其與網路版本之間的關聯歷史記錄。

頻道整合

若要將機器人網路與簡訊平台整合，請從 Amazon Lex V2 主控台的側邊功能表選擇您的機器人網路。然後選取 [頻道整合]。

1. 選取 [新增頻道]，將您的網路與新頻道整合。
2. 在「平台」區段中，在「選取平台」中選擇您要部署機器人的平台。將會建立 IAM 角色。從 KMS 金鑰下的下拉式選單中選擇金鑰，以保護您的資訊。
3. 在整合設定通道中，輸入名稱和選擇性說明。從下拉式功能表中選擇別名。
4. 從平台獲取您的帳戶 SID 和身份驗證令牌，並填寫帳戶 SID 和身份驗證令牌字段。如需詳細資訊，請參閱整合機器人。
5. 選取「建立」以完成頻道整合。

> **Note**
>
> 目前在亞馬遜 Connect 語音或聊天中無法使用機器人網路。

部署機器人

建立並測試您的機器人之後，就可以進行部署，以便與客戶互動。在本節中，瞭解如何在進行更新時建立機器人的版本。當機器人準備好進行部署時，請使用別名來指向不同版本的機器人。瞭解如何將您的機器人與訊息平台、行動應用程式和網站整合。

主題

版本控制和別名

Amazon Lex V2 支援建立機器人和機器人網路的版本和別名，以便您控制用戶端應用程式使用的實作。版本充當您作品的編號快照。您可以將別名指向您希望提供給客戶的機器人版本。在建立版本之間，您可以繼續更新機器人的Draft版本，而不會影響使用者體驗。

版本

Amazon Lex V2 支援建立機器人版本，因此您可以控制用戶端應用程式使用的實作。版本是工作的編號快照，您可以建立用於工作流程的不同部分，例如開發、測試版部署和生產環境。

草案版

當您建立 Amazon Lex V2 機器人時，只有一個版本，即Draft版本。

Draft是你的機器人的工作副本。您只能更新Draft版本，直到您創建第一個版本，Draft是您擁有的唯一機器人版本。

您的機器人Draft版本與TestBotAlias. 只TestBotAlias能用於手動測試。Amazon Lex V2 會限制您可以對機器人TestBotAlias別名發出的執行時間請求數量。

建立版本

當您對 Amazon Lex V2 機器人進行版本時，您會建立機器人的編號快照，以便您可以使用機器人建立版本時的存在狀態。一旦您建立了數字版本，當您繼續處理應用程式草稿版本時，它將保持不變。

建立版本時，您可以選擇要包含在版本中的語言環境。您不需要選擇機器人中的所有語言環境。此外，當您建立版本時，您可以從以前的版本中選擇語言環境。例如，如果您有三個Draft版本的機器人，則可以從版本中選擇一個地區設定，在建立第四版時從第二版中選擇一個地區設定。

如果您從版本中刪除地區設定，則不會從編號Draft版本中刪除該語言環境。

如果機器人版本在六個月內未使用，Amazon Lex V2 會將該版本標示為非作用中。當某個版本處於非作用中狀態時，您無法對機器人使用執行階段作業。若要讓機器人處於作用中狀態，請重建與版本相關聯的所有語言。

更新 Amazon Lex V2 機器人

您只能更新 Amazon Lex V2 機器人的Draft版本。版本無法變更。您可以在主控台中或使用CreateBotVersion作業更新資源之後，隨時建立新版本。

刪除 Amazon Lex V2 機器人或版本

Amazon Lex V2 支援使用主控台或其中一個 API 操作刪除機器人或版本：

- DeleteBot
- DeleteBotVersion

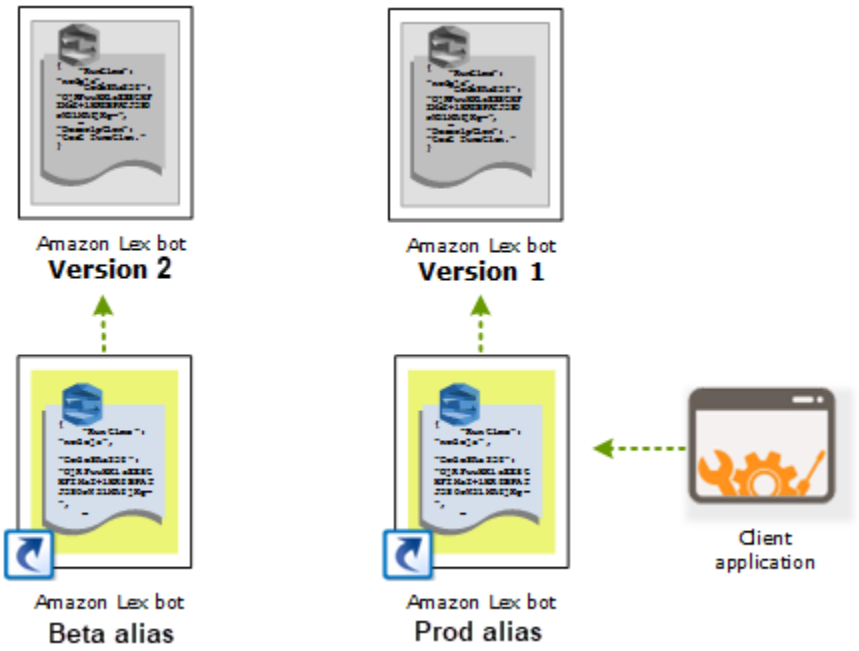
別名

Amazon Lex V2 機器人支援別名。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，您可以建立版本 2 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。

別名是指向特定版本的 Amazon Lex V2 機器人的指標。利用別名以讓用戶端應用程式能夠使用特定版本的機器人，而無需由應用程式追蹤其為哪個版本。

當您建立機器人時，Amazon Lex V2 會建立名為TestBotAlias的別名，供您用來測試機器人。TestBotAlias別名始終與您的機器人Draft版本相關聯。您應該只使用TestBotAlias別名進行測試，Amazon Lex V2 會限制您可以對別名發出的執行時間請求數量。

下列範例顯示 Amazon Lex V2 機器人的兩個版本，版本 1 和第 2 版。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 UpdateBotAlias 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。

當您將用戶端應用程式設定為呼叫 Amazon Lex Runtime V2 API 以讓客戶與您的機器人互動時，您可以使用指向您希望客戶使用的版本的別名。

> **Note**
>
> 雖然您可以在主控台中測試機器人的Draft版本，但我們建議您在將機器人與用戶端應用程式整合時，先建立版本並建立指向該版本的別名。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 V2 會針對所有進行中的工作階段使用目前版本。新工作階段會使用新版本。

使用 Java 應用程式與 Amazon Lex V2 機器人互動

AWS SDK for Java2.0 提供了一個接口，您可以使用 Java 應用程序與機器人進行交互。使用適用於 Java 的開發套件建立用戶端應用程式。

下列應用程式會與您在中建立的 OrderFlowers 機器人互動練習 1：從範例建立機器人。它使用LexRuntimeV2Client來自 SDK for Java 來調用RecognizeText操作以與機器人進行對話。

交談輪輪輪輪輪：

```
User : I would like to order flowers
Bot  : What type of flowers would you like to order?
User : 1 dozen roses
Bot  : What day do you want the dozen roses to be picked up?
User : Next Monday
Bot  : At what time do you want the dozen roses to be picked up?
User : 5 in the evening
Bot  : Okay, your dozen roses will be ready for pickup by 17:00 on 2021-01-04.  Does
 this sound okay?
User : Yes
Bot  : Thanks.
```

如需在用戶端應用程式和 Amazon Lex V2 機器人之間傳送的 JSON 結構，請參閱練習 2：檢閱交談流程。

若要執行應用程式，您必須提供以項目資訊：

- BID ID — 建立機器人時向其指派的 ID ID 類似於如果 ID 類似於如果 ID 您可以在機器人設定頁面上的 Amazon Lex V2 主控台中查看機器人識別碼。
- botAliasId — 建立機器人別名時指派給機器人別名 ID 類似於如果。您可以在「別名」頁面上的 Amazon Lex V2 主控台中看到機器人別名 ID。如果您在清單中看不到別名 ID，請選擇右上角的齒輪圖示，然後開啟「別名 ID」。

- LocaleID — 您用於機器人的地區設定識別碼。如需地區設定的清單，請參閱亞馬遜萊克斯 V2 支援的語言和語言環境。
- accessKey 和 secretKey — 您帳戶的驗證金鑰。如果還沒有一組金鑰，請使用AWS Identity and Access Management主控台建立。
- sessionId — 與 Amazon Lex V2 機器人進行工作階段的識別碼。在這種情況下，代碼使用隨機的 UUID。
- 區域 — 如果您的機器人不在美國東部 (維吉尼亞北部) 區域，請確保您變更區域。

應用程式使用呼叫的函數`getRecognizeTextRequest`來建立對機器人的個別要求。此函數會使用必要的參數建立要求，以傳送至 Amazon Lex V2。

```
package com.lex.recognizetext.sample;

import software.amazon.awssdk.auth.credentials.AwsBasicCredentials;
import software.amazon.awssdk.auth.credentials.AwsCredentialsProvider;
import software.amazon.awssdk.auth.credentials.StaticCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.lexruntimev2.LexRuntimeV2Client;
import software.amazon.awssdk.services.lexruntimev2.model.RecognizeTextRequest;
import software.amazon.awssdk.services.lexruntimev2.model.RecognizeTextResponse;

import java.net.URISyntaxException;
import java.util.UUID;


/**
 * This is a sample application to interact with a bot using RecognizeText API.
 */
public class OrderFlowersSampleApplication {

    public static void main(String[] args) throws URISyntaxException,
 InterruptedException {
        String botId = "";
        String botAliasId = "";
        String localeId = "en_US";
        String accessKey = "";
        String secretKey = "";
        String sessionId = UUID.randomUUID().toString();
        Region region = Region.US_EAST_1; // pick an appropriate region
```

```
        AwsBasicCredentials awsCreds = AwsBasicCredentials.create(accessKey,
secretKey);
        AwsCredentialsProvider awsCredentialsProvider =
StaticCredentialsProvider.create(awsCreds);

        LexRuntimeV2Client lexV2Client = LexRuntimeV2Client
                .builder()
                .credentialsProvider(awsCredentialsProvider)
                .region(region)
                .build();

        // utterance 1
        String userInput = "I would like to order flowers";
        RecognizeTextRequest recognizeTextRequest = getRecognizeTextRequest(botId,
botAliasId, localeId, sessionId, userInput);
        RecognizeTextResponse recognizeTextResponse =
lexV2Client.recognizeText(recognizeTextRequest);

        System.out.println("User : " + userInput);
        recognizeTextResponse.messages().forEach(message -> {
            System.out.println("Bot : " + message.content());
        });

        // utterance 2
        userInput = "1 dozen roses";
        recognizeTextRequest = getRecognizeTextRequest(botId, botAliasId, localeId,
sessionId, userInput);
        recognizeTextResponse = lexV2Client.recognizeText(recognizeTextRequest);

        System.out.println("User : " + userInput);
        recognizeTextResponse.messages().forEach(message -> {
            System.out.println("Bot : " + message.content());
        });

        // utterance 3
        userInput = "next monday";
        recognizeTextRequest = getRecognizeTextRequest(botId, botAliasId, localeId,
sessionId, userInput);
        recognizeTextResponse = lexV2Client.recognizeText(recognizeTextRequest);

        System.out.println("User : " + userInput);
        recognizeTextResponse.messages().forEach(message -> {
            System.out.println("Bot : " + message.content());
        });
```

```
        // utterance 4
        userInput = "5 in evening";
        recognizeTextRequest = getRecognizeTextRequest(botId, botAliasId, localeId,
 sessionId, userInput);
        recognizeTextResponse = lexV2Client.recognizeText(recognizeTextRequest);

        System.out.println("User : " + userInput);
        recognizeTextResponse.messages().forEach(message -> {
            System.out.println("Bot : " + message.content());
        });

        // utterance 5
        userInput = "Yes";
        recognizeTextRequest = getRecognizeTextRequest(botId, botAliasId, localeId,
 sessionId, userInput);
        recognizeTextResponse = lexV2Client.recognizeText(recognizeTextRequest);

        System.out.println("User : " + userInput);
        recognizeTextResponse.messages().forEach(message -> {
            System.out.println("Bot : " + message.content());
        });
    }

    private static RecognizeTextRequest getRecognizeTextRequest(String botId, String
 botAliasId, String localeId, String sessionId, String userInput) {
        RecognizeTextRequest recognizeTextRequest = RecognizeTextRequest.builder()
                .botAliasId(botAliasId)
                .botId(botId)
                .localeId(localeId)
                .sessionId(sessionId)
                .text(userInput)
                .build();
        return recognizeTextRequest;
    }
}
```

全球彈性

> **Note**
>
> 此功能僅適用於在美國東部 (維吉尼亞北部) 和美國西部 (奧勒岡) 區域建立的 Amazon Connect 和 Amazon Lex V2 執行個體。
> 若要存取此功能，請聯絡您的 Amazon Connect 解決方案架構師或技術客戶經理。

全域復原能力可讓您在次要區域複製機器人。透過兩個區域中使用者機器人的自動複製功能，可讓次要區域啟用。如果發生區域中斷，您將擁有一個備份區域。一旦啟用全域恢復能力，建立的新機器人就會在第二個 AWS 區域中複製。

啟用此功能後，您可以近乎即時地在配對 AWS 區域自動複寫 Amazon Lex V2 機器人及其資源、版本和別名。使用此功能，您可以監控原始機器人和複本機器人的版本號碼，以確保機器人複本與原始機器人保持同步。當您啟用複製時，您可以啟動您要在其中複製機器人的預先決定 AWS 區域 (區域以預先決定的配對為基礎)。來源區域中來源機器人的任何更新都會自動更新為第二個區域中的複製機器人。

> **Note**
>
> 啟動全域彈性後，只有在功能啟動後建立的機器人、版本和別名才會在複製區域中複製。先前建立的機器人、版本和別名將不會出現在複製區域中。識別的第二個區域是唯讀的，並以預先確定的配對。機器人更新僅限於最初建立機器人的區域。

有關使用全域恢復能力的其他資訊：

- 全域彈性目前僅適用於預先確定的區域對。

us-east-1	us-west-2
eu-west-2	eu-central-1

- 您可以建立任何 Amazon Lex V2 機器人的複本。啟用全域彈性之後，您必須為機器人建立新版本和新別名。
- 在全域復原功能中啟用的別名只能與啟用全域恢復功能的版本相關聯。

限制:

- 全域備援不會複寫使用 LLM (例如 CPFAQ 和話語產生) 的插槽建立的機器人。
- 全球彈性不會複寫機器人網路，但是屬於機器人網路一部分的任何機器人仍然可以個別複製。

主題

複製機器人和管理機器人複本的權限

如果 IAM 角色已附加AmazonLexFullAccess政策，則可以建立和管理機器人複本。

如果您想要建立具有最低權限的全域彈性角色，請使用下列原則，其中包含下列陳述式。

- 針對機器人複寫存取 Amazon Lex V2 服務連結角色的權限。
- 允許 Amazon Lex V2 為您的機器人複寫建立服務連結角色的許可。
- 呼叫機器人複寫 API 的權限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "GetReplicationSLR",
            "Effect": "Allow",
            "Action": [
                "iam:GetRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/replication.lexv2.amazonaws.com/
AWSServiceRoleForLexV2Replication*"
            ]
        },
        {
            "Sid": "CreateReplicationSLR",
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole",
            ],
            "Resource": [
```

```
                "arn:aws:iam::*:role/aws-service-role/replication.lexv2.amazonaws.com/
AWSServiceRoleForLexV2Replication*"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "lexv2.amazonaws.com"
                }
            }
        },
        {
            "Sid": "AllowBotReplicaActions",
            "Effect": "Allow",
            "Action": [
                "lex:CreateBotReplica",
                "lex:DescribeBotReplica",
                "lex:ListBotReplica",
                "lex:ListBotVersionReplicas",
                "lex:ListBotAliasReplicas",
                "lex:DeleteBotReplica"
            ],
            "Resource": [
                "arn:aws:lex:*:*:bot/*",
                "arn:aws:lex:*:*:bot-alias/*"
            ]
        }
    ]
}
```

您可以透過如下方式修改權限來進一步限制權限。

- 以特定機器人或機器人別名 ID 取代 *，以將權限限制為特定機器人或機器人別名。
- 使用lex BotReplica動作子集將角色限制為特定動作。

如需範例，請參閱 允許使用者建立和檢視機器人複本，但不能刪除它們。

部署全球彈性

全域備援資訊面板

您可以在「全域備援」面板中存取下列資訊：

- 來源詳細資料 — 有關機器人來源區域、複本類型、啟用複製日期和上次建立版本的資訊。使用此資訊來追蹤機器人的反覆運算。
- 複寫詳細資料 — 建立機器人複本後，您可以追蹤複製區域、複本類型、上次版本同步日期，以及上次複製的版本。使用此資訊來追蹤機器人複本的同步。
- 來源區域 — 啟用「全域備援」的區域。您可以在來源區域中進行變更，以便在兩個區域中複製機器人。
- 複本類型 — 指出機器人是唯讀還是能夠根據區域讀取和寫入。
- 複本區域 — 用來複寫來源機器人以獲得全域恢復能力的次要區域。全球彈性目前僅適用於 IAD/PDX 和 LDN/FRA 區域配對。
- 啟用複寫日期 — 啟用機器人複本的日期和時間。
- 上次建立的版本 — 與來源區域中複本相關聯的最後一個機器人版本。

啟用全球彈性

> **Note**
>
> 此功能僅適用於在美國東部 (維吉尼亞北部) 和美國西部 (奧勒岡) 區域建立的 Amazon Connect 和 Amazon Lex V2 執行個體。
> 若要存取此功能，請聯絡您的 Amazon Connect 解決方案架構師或技術客戶經理。

在 Amazon Lex V2 主控台中啟用全域備援之前，您必須確保啟用機器人複寫的使用者具有建立服務連結角色 (SLR) 的權限。呼叫時 CreateReplica ，全域恢復能力將使用這些 FAS 認證在啟用的帳戶中建立 SLR。如需在 Amazon Lex V2 中設定全球彈性單反相機的詳細資訊，請參閱 AWS 受管政策：。AmazonLexFullAccess

啟動全域恢復能力並為第二個區域設定機器人複寫：

1. 登入 AWS 管理主控台，然後開啟 Amazon Lex 主控台，網址為 https://console.aws.amazon.com/lex/。
2. 從左側導覽面板的「機器人」導覽中選擇您要複製的機器人。
3. 選擇「部署」>「全域恢復」。
4. 選擇窗口右上角的「創建副本」按鈕以創建機器人的草稿版本。

> **Note**
>
> 檢查以確定您在次要區域中沒有與您要複製的機器人相同名稱的任何機器人。（您的機器人必須具有唯一命名）。

5. 移至「全域備援」，按一下「建立複本」-此動作會建立 Bot 的草稿版本。(除了檢閱狀態或查看 future 組建的詳細資料外，您不需要返回 [全域備援] 索引標籤)。

> **Note**
>
> 您也可以移至別名並選取為啟用全域復原的機器人建立新別名，以便在全域恢復能力中的複寫建立別名機器人。只有在啟用複製後建立的別名才會被複製。

6. 移至別名-為啟用全域彈性的機器人建立新別名。只有在啟用複製後建立的別名才會複製。
7. 轉到版本-為啟用全球彈性的機器人創建新版本。只有在啟用複製之後建立的版本才會複製。

> **Note**
>
> 客戶仍然可以完全控制其資源型政策和複製機器人的標籤。Lambda 函數和 CloudWatch 日誌群組必須使用相同的識別碼在這兩個區域中部署。使用者不必在複本區域中再次關聯 lambda 函數。

停用全域恢復能力

您可以隨時選取停用全域復原功能按鈕來停用全域復原。此動作可防止您的來源機器人以及與其相關聯的任何別名和版本在其他區域中複製。

使用具有全域復原能力的 API

您可以使用下列 API 在全域備援中進行 API 呼叫。您可以在 Amazon Lex V2 API 指南中找到有關全球彈性 API 和 Amazon Lex V2 的其他資訊。

- CreateBotReplica

  啟用全域恢復能力並建立複製的機器人。需要 `replicaRegion`。

  如需詳細資訊，請參閱 Lex API 指南CreateBotReplica中的。

- DeleteBotReplica

  停用全域恢復能力並刪除複製的機器人。需要 `replicaRegion` 和 `botId`。

  如需詳細資訊，請參閱 Lex API 指南DeleteBotReplica中的。

- ListBotReplicas

  列出次要區域中複製的機器人。需要 `botId`。

  如需詳細資訊，請參閱 Lex API 指南ListBotReplicas中的。

- DescribeBotReplica

  複製機器人的資訊摘要。需要 `replicaRegion` 和 `botId`。

  如需詳細資訊，請參閱 Lex API 指南DescribeBotReplica中的。

將 Amazon Lex V2 機器人與簡訊平台整合

本節說明如何將 Amazon Lex V2 機器人與臉書、鬆弛和 Twilio 簡訊平台整合。如果您還沒有 Amazon Lex V2 機器人，請建立一個機器人。在本主題中，我們假設您正在使用您在其中建立的機器人練習 1：從範例建立機器人。但是，您可以使用任何機器人。

> **Note**
>
> 儲存您的臉書、鬆弛或 Twilio 組態時，Amazon Lex V2 會使用一個 AWS KMS key 來加密資訊。當您第一次建立這些簡訊平台的通道時，Amazon Lex V2 會在您的 AWS 帳戶中建立預設的客戶受管金鑰 (`aws/lex`)，或者您也可以選取自己的客戶受管金鑰。Amazon Lex V2 僅支援對稱金鑰。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 V2 時，它會包含平台特定資訊，做為 Lambda 函數的請求屬性。使用此屬性可自訂機器人的行為方式。如需詳細資訊，請參閱 設定請求屬性。

通用請求屬性

屬性	描述
x-amz-lex: 頻道:平台	下列其中一值： • `Facebook`

屬性	描述
	• Slack • Twilio

整合 Amazon Lex V2 機器人與臉書信使

您可以託管您的 Amazon Lex V2 機器人在 Facebook 信使。當您這樣做時，Facebook 用戶可以與您的機器人進行交互以實現意圖。

開始之前，您需要在 https://developers.facebook.com 註冊一個 Facebook 開發人員帳戶。

您需要執行以下步驟：

主題

步驟 1：建立應用程式

在 Facebook 開發人員入口網站上，建立 Facebook 應用程式和 Facebook 粉絲專頁。

建立臉書應用程式

1. 打開 https://developers.facebook.com/apps
2. 選擇 Create App (建立應用程式)。
3. 在「建立 App」頁面中，選擇「商家」，然後選擇「下一步」。
4. 在 [新增應用程式名稱]、[應用程式連絡人電子郵件] 和 [企業帳戶] 欄位中，為您的應用程式做出適當的選 選擇建立應用程式以繼續。
5. 從「新增產品至您的應用程式」中，從 Messenger 動態磚選擇「設定」。
6. 在「存取權杖」區段中，選擇「新增或移除頁面」。
7. 選擇要搭配 App 使用的頁面，然後選擇「下一步」。

8. 針對應用程式允許執行的動作，保留預設值，然後選擇 [完成]。
9. 在確認頁面上，選擇 OK (確定)。
10. 在「存取權杖」區段中，選擇「產生權杖」，然後複製權杖。您可以在 Amazon Lex V2 主控台中輸入此權杖。
11. 從左側選單中選擇「設定」，然後選擇「基本」。
12. 對於應用程式密碼，請選擇顯示，然後複製密碼。您可以在 Amazon Lex V2 主控台中輸入此權杖。

下一步驟

第 2 步：集成臉書信使與 Amazon Lex V2 機器人

第 2 步：集成臉書信使與 Amazon Lex V2 機器人

在這一步你鏈接你的 Amazon Lex V2 機器人與 Facebook。

1. 登入，AWS Management Console並在 https://console.aws.amazon.com/lex/ 開啟 Amazon Lex 主控台。
2. 從機器人清單中，選擇您建立的 Amazon Lex V2 機器人。
3. 在左側選單中，選擇 [頻道整合]，然後選擇 [新增頻道]。
4. 在「建立頻道」中，執行下列作業：

    a. 在「平台」中，選擇臉書。

    b. 針對身分識別原則，請選擇保護頻道資訊的AWS KMS金鑰。預設金鑰由 Amazon Lex V2 提供。

    c. 若為配置，請提供名稱和選用描述。選擇指向您要使用之機器人版本的別名，並選擇頻道支援的語言。

    d. 對於其他組態，請輸入以下內容：

    - 別名 — 識別呼叫 Amazon Lex V2 之應用程式的字串。您可以使用任何字串。記錄此字符串，然後在 Facebook 開發者控制台中輸入它。
    - 頁面訪問令牌-您從 Facebook 開發人員控制台複製的頁面訪問令牌。
    - 應用程式密鑰 — 您從 Facebook 開發人員控制台複製的密鑰。

    e. 選擇 Create (建立)

    f. Amazon Lex V2 會顯示您機器人的通道清單。從清單中，選擇您剛建立的頻道。

g. 在回呼 URL 中，記錄回呼 URL。您可以在 Facebook 開發人員控制台中輸入此 URL。

下一步驟

步驟 3：完成整合

步驟 3：完成整合

在此步驟中，請使用 Facebook 開發人員主控台完成與 Amazon Lex V2 的整合。

完成臉書信使整合

1. 打開 https://developers.facebook.com/apps
2. 從應用程序列表中，選擇要與 Facebook 即時通集成的應用程序。
3. 在左側選單中，選擇「即時通」，然後選擇「設定」。
4. 在「網路掛鉤」區段中：
   a. 選擇新增回呼 URL。
   b. 在編輯回呼 URL 中，輸入下列內容：
      - 回呼網址 — 輸入您從 Amazon Lex V2 主控台錄製的回呼網址。
      - 驗證權杖 — 輸入您在 Amazon Lex V2 主控台中輸入的別名。
   c. 選擇驗證並儲存。
   d. 在頁面旁邊的 Webhook 下選擇新增子選項。
   e. 在彈出的視窗中，選擇，`messages`然後按一下 [儲存]。

下一步驟

步驟 4：測試整合

步驟 4：測試整合

現在，您可以從 Facebook 信使與您的 Amazon Lex V2 機器人開始對話。

為了測試 Facebook 信使和 Amazon Lex V2 機器人之間的集成

1. 在步驟 1 中開啟您與機器人相關聯的 Facebook 頁面。

2. 在 Messenger 視窗中，使用中提供的測試語音練習 1：從範例建立機器人。

整合 Amazon Lex V2 機器人

本主題提供將 Amazon Lex V2 機器人與 Slack 簡訊應用程式整合的指示。您會執行以下步驟：

主題

- 步驟 1：註冊
- 步驟 2：建立 Slack 應用程式
- 步驟 3：整合 Slack 應用程式與 Amazon Lex V2 機器人
- 步驟 4：完整的 Slack 整合
- 步驟 5：測試整合

步驟 1：註冊

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，您將創建一個 Slack 應用程序，任何 Slack 團隊都可以安裝該應用程序。

下一步驟

步驟 2：建立 Slack 應用程式

步驟 2：建立 Slack 應用程式

請在本節執行以下動作：

1. 在 Slack API 主控台中建立 Slack 應用程式。
2. 設定應用程式以將互動式訊息新增至您的機器人。

在本節末尾，您將獲得應用程序憑據（客戶端 ID，客戶端密鑰和驗證令牌）。在下一個步驟中，您會使用此資訊在 Amazon Lex V2 主控台中整合機器人。

若要建立鬆弛應用程式

1. 請登入 Slack API 主控台，網址為 https://api.slack.com。
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。在 Amazon Lex 主控台新增機器人通道關聯之後，您將會更新此 URL。

     - 選擇 Save Changes (儲存變更)。
4. 在左側功能表的 Settings (設定) 中，選擇在 Basic Information (基本資訊)。記錄以下應用程式登入資料：

   - 用戶端 ID
   - 用戶端密碼
   - 驗證符記

下一步驟

步驟 3：整合 Slack 應用程式與 Amazon Lex V2 機器人

步驟 3：整合 Slack 應用程式與 Amazon Lex V2 機器人

在本節中，將您建立的 Slack 應用程式與您使用通道整合建立的 Amazon Lex V2 機器人進行整合。

1. 登入，AWS Management Console並開啟位於 https://console.aws.amazon.com/lex/ 的 Amazon Lex (Amazon Lex) 主控台。
2. 從機器人清單中，選擇您建立的 Amazon Lex V2 機器人。
3. 在左側選單中，選擇 [頻道整合]，然後選擇 [新增頻道]。
4. 在 [執行下列作業]，執行下列作業：

   a. 針對「平台」，選擇「鬆弛」。

   b. 針對身分識別原則，請選擇保護頻道資訊的AWS KMS金鑰。Amazon Lex V2 提供的預設金鑰。

c. 對於整合配置，提供名稱和選用描述。選擇指向您要使用之機器人版本的別名，並選擇頻道支援的語言。

> **Note**
>
> 如果您的機器人有多種語言版本，您必須為每種語言建立不同的管道和不同的應用程式。

d. 對於其他組態，請輸入以下內容：

- 用戶端 ID — 輸入 Slack 中的用戶端 ID。
- 用戶端密碼 — 輸入 Slack 的用戶端密碼。
- 驗證令牌 — 輸入 Slack 的驗證令牌。
- 成功頁面 URL — Slack 在驗證使用者時應開啟的頁面 URL。這個項目一般會保持空白。

5. 選擇「建立」以建立頻道。
6. Amazon Lex V2 會顯示您機器人的通道清單。從清單中，選擇您建立的頻道。
7. 在回呼網址中，記錄端點和 OAuth 端點。

下一步驟

步驟 4：完整的 Slack 整合

步驟 4：完整的 Slack 整合

在本節中，請使用 Slack API 主控台來完成與 Slack 應用程式的整合。

1. 請登入 Slack API 主控台，網址為 https://api.slack.com。選擇您在步驟 2：建立 Slack 應用程式中建立的應用程式。
2. 依下列方式更新 OAuth 與許可功能：

a. 在左側功能表中，選擇 OAuth 與許可。

b. 在「重新導向網址」區段中，新增 Amazon Lex 在上一個步驟中提供的 OAuth 端點。選擇 Add (新增)，然後選擇Save URLs (儲存 URL)。

c. 在「機器人權杖範圍」區段中，使用「新增 OAuth 範圍」按鈕新增兩個權限。以下列文字篩選清單：

- **`chat:write`**

- **`team:read`**

3. 將請求 URL 值更新到 Amazon Lex 在上一步中提供的端點，以更新互動性和捷徑功能。輸入您在步驟 3 中儲存的端點，然後選擇「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端點。
   - 在「訂閱機器人事件」區段中，選取「新增機器人使用者事件」並新增**`message.im`**機器人事件，以啟用終端使用者與 Slack 機器人之間的直接訊息。
   - 儲存變更。
5. 啟用從「訊息」標籤傳送訊息，如下所示：

   - 從左側選單中選擇「應用程式首頁」。
   - 在 「顯示分頁」區段中，選擇「允許使用者從訊息」標籤傳送 Slash 命令和訊息。
6. 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。如果您已通過多個工作區的驗證，請先從下拉式清單中選擇右上角的正確工作區。接下來，選擇允許授權機器人回應消息。

> **Note**
>
> 如果您稍後對 Slack 應用程式設定進行任何變更，則必須重做此子步驟。

下一步驟

步驟 5：測試整合

步驟 5：測試整合

現在，使用瀏覽器視窗來測試 Slack 與 Amazon Lex V2 機器人的整合。

若要測試您的 Slack 應用程式

1. 啟動 Slack。在左側選單的「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。
2. 與您的 Slack 應用程式聊天。您的機器人會回應訊息。

   如果您使用建立機器人練習 1：從範例建立機器人，則可以使用該練習中的範例對話。

整合亞 Amazon Lex V2 機器人與 Twilio 簡訊

本主題提供將 Amazon Lex V2 機器人與 Twilio 簡易訊息服務 (SMS) 整合的指示。您會執行以下步驟：

主題

步驟 1：建立簡訊帳戶

註冊 Twilio 帳戶，並記錄以下帳戶資訊：

- ACCOUNT SID
- AUTH TOKEN

如需註冊說明，請參閱 https://www.twilio.com/console。

下一步驟

步驟 2：將 Twilio 訊息服務端點與 Amazon Lex V2 機器人整合

步驟 2：將 Twilio 訊息服務端點與 Amazon Lex V2 機器人整合

1. 登入，AWS Management Console並在 https://console.aws.amazon.com/lex/ 開 Amazon Lex 主控台。
2. 從機器人清單中，選擇您建立的 Amazon Lex V2 機器人。
3. 在左側選單中，選擇 [頻道整合]，然後選擇 [新增頻道]。
4. 在 [，執行下列作業：

    a. 針對「平台」，選擇「Twilio」。

    b. 針對身分識別原則，請選擇保護頻道資訊的AWS KMS金鑰。預設金鑰由 Amazon Lex V2 提供。

    c. 對於 [，提供選用描述]，提供通道名稱和選用描述。選擇指向您要使用之機器人版本的別名。

    d. 對於其他配置，請從 Twilio 儀表板輸入帳戶 SID 和驗證令牌。

5. 選擇 建立 。
6. 從頻道清單中，選擇您剛建立的頻道。
7. 複製回呼網址。

下一步驟

步驟 3：完成整合

步驟 3：完成整合

使用 Twilio 主控台完成 Amazon Lex V2 機器人與 Twilio 簡訊的整合。

1. 開啟主控台，網址為 https://www.twilio.com/console。
2. 從左側選單中選擇「所有產品與服務」，然後選擇「電話號碼」。
3. 如果您有電話號碼，請選擇它。如果您沒有電話號碼，請選擇「購買號碼」即可取得電話號碼。
4. 在「訊息」區段的「傳入訊息」中，輸入來自 Amazon Lex V2 主控台的回呼 URL。
5. 選擇 儲存 。

下一步驟

步驟 4：測試整合

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。使用您的手機傳送訊息至 Twilio 號碼。

如果您使用建立機器人練習 1：從範例建立機器人，則可以使用該練習中的範例對話。

將 Amazon Lex V2 機器人與聯絡中心整合

您可以將 Amazon Lex V2 機器人與您的聯絡中心整合，以使用 Amazon Lex V2 串流 API 啟用自助式使用案例。將這些機器人用作電話語音上的互動式語音回應 (IVR) 代理程式，或作為整合至您聯絡中心的文字型聊天機器人。如需串流 API 的詳細資訊，請參閱串流事件到 Amazon Lex V2 機器人。

使用串流 API，您可以啟用下列功能：

- 中斷 (「剔除」) — 呼叫者可以在提示完成之前中斷機器人並回答問題。如需詳細資訊，請參閱 讓您的機器人被您的使用者中斷。

- 等待並繼續 — 呼叫者可以指示機器人在通話過程中需要時間來檢索其他信息，例如信用卡號碼或預訂 ID 等待。如需詳細資訊，請參閱 讓機器人等待使用者提供更多資訊。
- DTMF 支援 — 來電者可以透過語音或 DTMF 互換提供資訊。
- SSML 支援 — 您可以使用 SSML 標籤設定 Amazon Lex V2 機器人提示，以更好地控制文字產生語音。如需詳細資訊，請參閱 Amazon Polly 開發人員指南中的從 SSML 文件產生語音。
- 可設定的逾時 — 您可以設定客戶在 Amazon Lex V2 收集其語音輸入 (例如回答「是」或「否」問題) 之前等待客戶完成講話的時間長度，或提供日期或信用卡號碼。如需詳細資訊，請參閱 設定擷取使用者輸入的逾時。
- 履行進度更新 — 您可以將機器人設定為根據商務邏輯執行期間的履行狀態回應多則訊息，以達成意圖履行。您可以設定機器人在履行開始和完成時回應訊息，並為長時間執行的 Lambda 函數提供定期更新。如需詳細資訊，請參閱 設定出貨進度更新。

Amazon Chime SDK

使用 Amazon Chime 開發套件，將即時音訊、視訊、螢幕共用和簡訊功能新增至您的 Web 或行動應用程式。Amazon Chime SDK 提供公用交換電話網路 (PSTN) 音訊服務，讓您可以建立具有某個AWS Lambda功能的自訂電話語音應用程式。

Amazon Chime 聲 PSTN 音頻與 Amazon Lex V2 集成。您可以使用此整合來存取 Amazon Lex V2 機器人，做為客服中心的互動式語音回應 (IVR) 系統，以進行音訊互動。在下列情況下，使用 PSTN 音訊服務來整合 Amazon Lex V2。

聯絡中心整合 — 您可以使用 Amazon Chime 語音連接器和 Amazon Chime SDK PSTN 音訊服務來存取 Amazon Lex V2 機器人。在任何使用工作階段初始通訊協定 (SIP) 進行語音通訊的客服中心應用程式中使用它們。此整合可透過 SIP 支援，將自然語言語音對話體驗新增至您現有的內部部署或雲端式聯絡中心。如需支援的聯絡中心平台清單，請參閱 Amazon Chime 語音連接器資源。

下圖顯示使用 SIP 和 Amazon Lex V2 的聯絡中心之間的整合。

直接電話語音支援 — 您可以建立自訂的 IVR 解決方案，使用 Amazon Chime 開發套件中佈建的電話號碼直接存取 Amazon Lex V2 機器人。

如需詳細資訊，請參閱《Amazon Chime 開發人員指南》中的下列主題。

- 使用 Amazon Chime 語音連接器進行 SIP 整合
- 使用 Amazon Chime 開發套件 PSTN 音訊服務
- 將 Amazon Chime 聲 PSTN 音頻與 Amazon Lex V2 集成

當 Amazon Chime 開發套件傳送請求至 Amazon Lex V2 時，它會將平台特定資訊納入您的 Lambda 函數和交談日誌中。使用此資訊來判斷將流量傳送至您的機器人的聯絡中心應用程式。

常用	值
x-amz-lex: 頻道:平台	`Amazon Chime SDK PSTN Audio`

Amazon Connect

Amazon Connect 是全通道雲端聯絡中心。您可以通過幾個步驟設置聯絡中心，添加位於任何地方的代理，並開始與客戶互動。如需詳細資訊，請參閱 Amazon Connect 管理員指南中的開始使用 Amazon Connect。

您可以使用全通道通訊，為客戶建立個人化體驗。例如，您可以根據客戶偏好和估計的等待時間提供聊天和語音聯繫。同時，代理商可以從一個界面處理所有客戶。例如，他們可以與客戶聊天，並在路由到他們的任務時建立或回應任務。

您可以使用 Amazon Connect 與客戶進行音訊互動，或使用 Amazon Connect 聊天進行純文字互動。

如需詳細資訊，請參閱 Amazon Connect 管理員指南中的以下主題。

- 什麼是 Amazon Connect
- 添加 Amazon Lex V2 機器人
- Amazon Connect 獲取客戶輸入聯繫人塊

當聯絡中心傳送請求至 Amazon Lex V2 時，它會包含平台特定資訊，做為 Lambda 函數和交談日誌的請求屬性。使用此資訊來判斷哪個聯絡中心應用程式將流量傳送至您的機器人。

通用請求屬性

屬性	值
x-amz-lex: 頻道:平台	下列其中一值：

屬性	值
	• Connect • Connect Chat

創世基雲

Genesys 雲端是一套雲端服務，適用於企業通訊、協作和聯絡中心管理。Genesys Cloud 建立在基礎上， AWS 並使用分散式雲端環境，為工作中的組織提供安全存取。

有關更多信息，請參閱 Genesys 雲網站上的以下頁面。

- 關於創惟雲端聯絡中心
- 關於 Amazon Lex V2 集成

當聯絡中心傳送請求至 Amazon Lex V2 時，會包含平台特定資訊，做為 Lambda 函數和交談日誌的請求屬性。使用此資訊來判斷哪個聯絡中心應用程式將流量傳送至您的機器人。

通用請求屬性

屬性	值
x-amz-lex: 頻道:平台	• Genesys Cloud

進一步了解

- 利用 Amazon Lex 和 Genesys 雲端為您的聯絡中心提供動力

管理對話

建立機器人之後，您可以將用戶端應用程式與 Amazon Lex V2 執行階段作業整合，以便與機器人進行對話。

當使用者與您的機器人開始對話時，Amazon Lex V2 會建立工作階段。工作階段會封裝應用程式與機器人之間交換的資訊。如需詳細資訊，請參閱使用亞馬遜萊克斯 V2 API 管理工作階段。

典型的對話涉及用戶和機器人之間的來回流程。例如：

```
User : I'd like to make an appointment
Bot : What type of appointment would you like to schedule?
User : dental
Bot : When should I schedule your dental appointment?
User : Tomorrow
Bot : At what time do you want to schedule the dental appointment on 2021-01-01?
User : 9 am
Bot : 09:00 is available, should I go ahead and book your appointment?
User : Yes
Bot : Thank you. Your appointment has been set successfully.
```

當您使用RecognizeText或RecognizeUtterance作業時，您必須在用戶端應用程式中管理交談。當您使用StartConversation作業時，Amazon Lex V2 會為您管理交談。

若要管理對話，您必須將使用者話語傳送至機器人，直到對話達到邏輯結束為止。目前的交談會以工作階段狀態擷取。工作階段狀態會在每個使用者說話之後更新。會話狀態包含交談的當前狀態，並由機器人在響應中返回。給每個用戶的話語。

交談可以處於下列任一狀態：

- ElicitIntent— 表示機器人尚未確定使用者的意圖。
- ElicitSlot— 表示機器人已偵測到使用者的意圖，並正在收集必要的資訊以達成意圖。
- ConfirmIntent— 表示機器人正在等待用戶確認收集的信息是否正確。
- 已關閉 — 表示使用者的意圖已完成，且與機器人的對話達到邏輯結束。

使用者可以在第一個意圖完成後指定新的意圖。如需詳細資訊，請參閱管理交談內容。

意圖可以具有下列其中一種狀態：

- InProgress— 表示機器人正在收集完成意圖所需的資訊。這與`ElicitSlot`交談狀態結合。
- 等待中 — 表示使用者要求機器人在機器人要求提供特定位置的資訊時等候。
- 已履行 — 表示與意圖相關聯的 Lambda 函數中的商務邏輯已成功執行。
- ReadyForFulfillment— 指出機器人已收集完成意圖所需的所有資訊，且用戶端應用程式可執行履行商務邏輯。
- 失敗 — 指示意圖失敗。

請參閱下列主題，了解如何使用 Amazon Lex V2 API 管理機器人和使用者之間的交談內容和工作階段。

主題

管理交談內容

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

主題

設定意圖上下文

您可以根據上下文有 Amazon Lex 觸發意圖。上下文是一種狀態變數，可在您定義機器人時與意圖相關聯。當您使用主控台或使用CreateIntent作業建立意圖時，可以配置意圖的前後關聯。您只能在英文 (美國) (en-US) 地區設定中使用上下文。

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

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

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

您也可以針對意圖使用 Lambda 函數來設定交談的內容。Amazon Lex 的輸出內容會傳送至 Lambda 函數輸入事件。Lambda 函數可以在其回應中傳送內容。如需詳細資訊，請參閱使用AWS Lambda函數啟用自訂邏輯。

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

輸出上下文

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

每個前後關聯都有一份在工作階段中維護的參數清單。參數是履行意圖的槽值。您可以使用這些參數為其他意圖預先填入插槽值。如需詳細資訊，請參閱 使用預設插槽值。

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

定義輸出內容時，您還可以定義其存留時間、內容包含在 Amazon Lex 回應中的時間長度或圈數。轉向是從您的應用程序向亞馬遜 Lex 提出的一個請求。一旦回合次數或時間過期，上下文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：

- 根據上下文更改應用程序的行為。例如，旅遊應用程式對於上下文「已完成的書籍」可能會有不同的動作，而不是「租賃酒店」。

- 將輸出內容傳回 Amazon Lex 做為下一個語音的輸入內容。如果 Amazon Lex 將語音識別為嘗試引出意圖，它會使用上下文來限制可傳回給具有指定上下文的意圖。

輸入上下文

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

您可以使用控制台或`CreateIntent`操作設置意圖響應的輸入上下文。一個意圖可以有多個輸入上下文。

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

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

使用預設插槽值

當您使用預設值時，您可以指定插槽值的來源，以便在使用者的輸入未提供插槽時填入新意圖。此來源可以是先前的對話方塊、要求或工作階段屬性，或是您在建置時設定的固定值。

您可以使用以下內容作為預設值的來源。

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

當您使用此CreateIntent操作將槽加入至意圖時，您可以加入預設值的清單。預設值會以列出的順序使用。例如，假設您的意圖具有下列定義的槽：

```
"slots": [
    {
        "botId": "string",
        "defaultValueSpec": {
            "defaultValueList": [
                {
                    "defaultValue": "#book-car-fulfilled.startDate"
```

```
                },
                {
                    "defaultValue": "[reservationStartDate]"
                }
            ]
        },
        Other slot configuration settings
    }
]
```

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

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

如果插槽使用預設值，即使是必要的，也不會產生插槽。

設定階段屬性

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

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

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

工作階段屬性會在工作階段期間持續保留。Amazon Lex 將它們存放在加密的資料存放區中，直到工作階段結束為止 用戶端可以在要求中建立工作階段屬性，方法是呼叫RecognizeText或RecognizeUtterance作業，並將`sessionAttributes`欄位設定為值。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未在 Amazon Lex 的請求中包含sessionAttribute欄位，就會使用儲存的屬性值。

例如，假設您有兩個工作階段屬性，{"x": "1", "y": "2"}。如果用戶端在未指定sessionAttributes欄位的情況下呼叫RecognizeText或RecognizeUtterance作業，Amazon Lex 會呼叫具有已存工作階段屬性的 Lambda 函數 ({"x": 1, "y": 2})。如果 Lambda 函數未傳回工作階段屬性，Amazon Lex 會將儲存的工作階段屬性傳回給用戶端應用程式。

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

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

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

針對RecognizeText作業，您可以使用sessionState結構sessionAttributes欄位將對映插入要求主體中，如下所示：

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

對於該RecognizeUtterance操作，您需要 base64 對映進行編碼，然後將其作為x-amz-lex-session-state標題的一部分發送。

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 RecognizeUtterance 標頭，或在 requestAttributes 請求中使用 RecognizeText 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 RecognizeUtterance 或 RecognizeText 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。

設定使用者定義請求

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 RecognizeUtterance 標頭，或在 requestAttributes 請求中 RecognizeText 欄位傳送資訊。

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

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

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

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

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

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

設定工作階段逾時

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

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

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

在意圖之間共用資訊

Amazon Lex 支援在意圖之間共用資訊。若要在意圖之間共用，請使用輸出內容或工作階段屬性。

若要使用輸出前後關聯，您可以在建立或更新方式時定義輸出前後關聯。當意圖達成時，Amazon Lex V2 的回應會包含意圖中的內容和位置值做為內容參數。您可以在後續意圖或應用程式程式碼或 Lambda 函數中使用這些參數做為預設值。

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

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

設定複雜屬性

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

```
{
   "cities": [
      {
         "city": {
            "name": "New York",
            "state": "New York",
            "pop": "8537673"
         }
      },
      {
         "city": {
            "name": "Los Angeles",
            "state": "California",
            "pop": "3976322"
         }
      },
      {
         "city": {
            "name": "Chicago",
            "state": "Illinois",
            "pop": "2704958"
         }
      }
   ]
}
```

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

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

```
// To convert an object to a string.
var jsonString = JSON.stringify(object, null, 2);
// To convert a string to an object.
var obj = JSON.parse(JSON string);
```

若要隨`RecognizeUtterance`作業傳送屬性，您必須先對屬性進行 base64 編碼，然後再將屬性新增至要求標頭，如下列JavaScript程式碼所示：

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

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

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

使用亞馬遜萊克斯 V2 API 管理工作階段

當使用者與您的機器人開始對話時，Amazon Lex V2 會建立工作階段。應用程式和 Amazon Lex V2 之間交換的資訊會構成交談的工作階段狀態。當您提出要求時，工作階段會由您指定的識別碼來識別。如需有關工作階段識別碼的詳細資訊，請參閱RecognizeText或RecognizeUtterance作業中的sessionId欄位。

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。

有三種方法可以更新工作階段狀態。

- 將工作階段資訊傳遞為RecognizeText或RecognizeUtterance作業呼叫的一部分內嵌。
- 使用 Lambda 函數RecognizeText搭配在每次交談後呼叫的或RecognizeUtterance作業。如需詳細資訊，請參閱使用AWS Lambda函數啟用自訂邏輯。另一種方法是在應用程式中使用 Amazon Lex V2 執行階段 API 來變更工作階段狀態。
- 使用可讓您管理與機器人對話的工作階段資訊的作業。作業包括PutSession作業、GetSession作業和DeleteSession作業。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。此作業會傳回工作階段的目前狀態，包括與使用者的對話狀態、已設定的任何工作階段屬性以及目前意圖的插槽值，以及 Amazon Lex V2 識別為符合使用者話語的可能意圖的任何其他意圖。

此 PutSession 操作可讓您直接運用目前的工作階段狀態。您可以設定工作階段，包括機器人接下來將執行的對話動作類型，以及 Amazon Lex V2 傳送給使用者的訊息。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Delegate為讓 Amazon Lex V2 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。

來自 PutSession 操作的回應包含如 RecognizeUtterance 操作的相同資訊。正如您會對 RecognizeUtterance 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。

工作階段作業與您的履行 Lambda 函數搭配使用。例如，如果您的 Lambda 函數傳回Failed為履行狀態，您可以使用該PutSession作業將對話方塊動作類型設定fulfillmentState為ReadyForFulfillment或重試履行步驟。close

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。

開始新的工作階段

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 會根據您的歡迎意圖提示回應，以開始與使用者進行對話。

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 將使用新的意圖與使用者開始對話。

恢復先前的意圖

若要繼續先前的意圖，請使用GetSession作業取得意圖的狀態、執行所需的互動，然後使用該PutSession作業將意圖設定為其先前的對話方塊狀態。

- 呼叫 GetSession 操作。存儲意圖的狀態。
- 執行另一個互動，例如實現不同的意圖。
- 使用已儲存之前意圖的資訊，呼叫PutSession作業。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。

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。如果您想要變更 Amazon Lex 用來引出插槽messageFormat值的訊息，您可以選擇設定message和欄位。

使用AWS Lambda函數啟用自訂邏輯

透過AWS Lambda功能，您可以透過定義的自訂函數更好地控制 Amazon Lex V2 機器人的行為。

Amazon Lex V2 會針對每種語言的每個機器人別名使用一個 Lambda 函數，而不是針對每個意圖使用一個

若要將 Lambda 函數與您的 Amazon Lex V2 機器人整合，請執行下列步驟：

1. 決定輸入事件中要從哪些欄位中擷取資訊，以便在 Lambda 函數中使用。
2. 決定您要操作及從 Lambda 函數傳回回應中的哪些欄位。
3. AWS Lambda使用您選擇的編程語言創建一個函數並編寫腳本。
4. 請確定函數傳回與回應格式相符的結構。
5. 部署 Lambda 函數。
6. 將 Lambda 函數與 Amazon Lex V2 機器人別名與主控台或 API 作業建立關聯。
7. 選取您要透過主控台或 API 作業叫用 Lambda 函數的交談階段。
8. 建置您的 Amazon Lex V2 機器人，並測試 Lambda 函數是否如預期運作。在 Amazon 的幫助下調試您的功能 CloudWatch。

主題

解譯輸入事件格式

將 Lambda 函數整合到 Amazon Lex V2 機器人的第一個步驟是了解 Amazon Lex V2 事件中的欄位，並從這些欄位判斷撰寫指令碼時要使用的資訊。下列 JSON 物件顯示傳遞至 Lambda 函數之 Amazon Lex V2 事件的一般格式：

> **Note**
>
> 輸入格式可能會變更而不對messageVersion. 如果存在新字段，您的代碼不應該拋出錯誤。

```
{
    "messageVersion": "1.0",
    "invocationSource": "DialogCodeHook | FulfillmentCodeHook",
    "inputMode": "DTMF | Speech | Text",
    "responseContentType": "audio/mpeg | audio/ogg | audio/pcm | text/plain;
 charset=utf-8",
    "sessionId": string,
    "inputTranscript": string,
    "invocationLabel": string,
    "bot": {
        "id": string,
        "name": string,
        "localeId": string,
        "version": string,
        "aliasId": string,
        "aliasName": string
    },
    "interpretations": [
        {
            "interpretationSource": "Bedrock | Lex",
            "intent": {
                // see ## for details about the structure
            },
            "nluConfidence": number,
            "sentimentResponse": {
                "sentiment": "MIXED | NEGATIVE | NEUTRAL | POSITIVE",
                "sentimentScore": {
                    "mixed": number,
                    "negative": number,
                    "neutral": number,
                    "positive": number
                }
            }
        },
        ...
    ],
    "proposedNextState": {
```

```
        "dialogAction": {
            "slotToElicit": string,
            "type": "Close | ConfirmIntent | Delegate | ElicitIntent | ElicitSlot"
        },
        "intent": {
            // see ## for details about the structure
        },
        "prompt": {
            "attempt": string
        }
    },
    "requestAttributes": {
        string: string,
        ...
    },
    "sessionState": {
        // see ###### for details about the structure
    },
    "transcriptions": [
        {
            "transcription": string,
            "transcriptionConfidence": number,
            "resolvedContext": {
                "intent": string
            },
            "resolvedSlots": {
                slot name: {
                    // see # for details about the structure
                },
                ...
            }
        },
        ...
    ]
}
```

輸入事件中的每個字段描述如下：

messageVersion

訊息版本，可識別進入 Lambda 函數的事件資料格式，以及 Lambda 函數回應的預期格式。

> **Note**
>
> 您在定義意圖時設定此值。在目前的實作中，Amazon Lex V2 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。

invocationSource

呼叫 Lambda 函數的程式碼掛接。可能的值如下：

`DialogCodeHook`-Amazon Lex V2 在用戶輸入後調用 Lambda 函數。

`FulfillmentCodeHook`— Amazon Lex V2 在填滿所有必要的插槽後呼叫 Lambda 函數，且意圖已準備好履行。

輸入模式

使用者說話的模式。可能的值如下：

`DTMF`— 用戶使用按鍵式鍵盤（雙音多頻）輸入語音。

`Speech`— 用戶說話.

`Text`— 使用者輸入的話語。

responseContentType

機器人對使用者回應的模式。 `text/plain; charset=utf-8`表示已寫入最後一個語音，而以開頭的值則`audio`表示朗讀最後一個話語。

sessionId

用於交談的英數字工作階段識別碼。

inputTranscript

來自用戶的輸入的轉錄。

- 對於文字輸入，這是使用者輸入的文字。對於 DTMF 輸入，這是使用者輸入的金鑰。
- 對於語音輸入，這是 Amazon Lex V2 將使用者語音轉換成的文字，以叫用意圖或填滿插槽。

調用標籤

指出叫用 Lambda 函數之回應的值。您可以設定初始回應、位置和確認回應的呼叫標籤。

機器人

處理要求之機器人的相關資訊，包含下列欄位：

- id — 建立機器人時指派給機器人的識別碼。您可以在機器人設定頁面上的 Amazon Lex V2 主控台中查看機器人識別碼。
- name — 您在建立機器人時給予機器人的名稱。
- LocaleID — 您用於機器人的地區設定識別碼。如需地區設定的清單，請參閱亞馬遜萊克斯 V2 支援的語言和語言環境。
- version — 處理請求的機器人版本。
- AliaSid — 當您建立機器人別名時，指派給機器人別名的識別碼。您可以在「別名」頁面上的 Amazon Lex V2 主控台中看到機器人別名 ID。如果您在清單中看不到別名 ID，請選擇右上角的齒輪圖示，然後開啟「別名 ID」。
- 別名 — 您為機器人提供別名的名稱。

解釋

Amazon Lex V2 認為可能符合使用者話語的意圖相關資訊清單。每個項目都是一個結構，提供有關語音與意圖的比對資訊，格式如下：

```
{
    "intent": {
        // see ## for details about the structure
    },
    "interpretationSource": "Bedrock | Lex",
    "nluConfidence": number,
    "sentimentResponse": {
        "sentiment": "MIXED | NEGATIVE | NEUTRAL | POSITIVE",
        "sentimentScore": {
            "mixed": number,
            "negative": number,
            "neutral": number,
            "positive": number
        }
```

```
    }
}
```

結構內的欄位如下所示：

- 意圖 — 包含意圖相關資訊的結構。如需結構的意圖詳細資訊，請參閱。
- NLUCESS — 指出 Amazon Lex V2 對於意圖符合使用者意圖的信心程度有多大的分數。
- 情 sentimentResponse — 回應情緒的分析，包含下列欄位：
  - 情緒 — 指出話語的情緒是POSITIVE、NEGATIVENEUTRAL、或。MIXED
  - 情緒分數 — 將每個情緒對應至數字的結構，指出 Amazon Lex V2 對語言傳達該情緒的信心程度。
- 解釋來源 — 指出插槽是否由 Amazon Lex 或 Amazon 基岩解決。

proposedNextState

如果 Lambda 函數將dialogAction的設定sessionState為Delegate，則會出現此欄位，並顯示 Amazon Lex V2 針對交談的下一個步驟提出的建議。否則，下一個狀態取決於您從 Lambda 函數回應中傳回的設定。只有在下列兩個陳述式都成立時，才會出現此結構：

1. 該invocationSource值是 DialogCodeHook
2. type的預測dialogAction是ElicitSlot。

您可以使用此信息runtimeHints在對話中的正確點添加。使用執行時期提示改善位置值的辨識如需詳細資訊，請參閱。 proposedNextState是包含下列欄位的結構：

的結構proposedNextState如下：

```
"proposedNextState": {
    "dialogAction": {
        "slotToElicit": string,
        "type": "Close | ConfirmIntent | Delegate | ElicitIntent | ElicitSlot"
    },
    "intent": {
        // see ## for details about the structure
    },
    "prompt": {
        "attempt": string
    }
```

```
}
```

- dialogAction — 包含 Amazon Lex V2 提議的下一個步驟的相關資訊。結構中的字段如下：
  - slotToElicit— 按照 Amazon Lex V2 建議接下來引出的插槽。此欄位只有在是時才會type顯示ElicitSlot。
  - 類型-Amazon Lex V2 提出的對話中的下一個步驟。可能的值如下：

    Delegate— Amazon Lex V2 確定下一個動作.

    ElicitIntent— 下一個動作是向使用者引出意圖。

    ElicitSlot— 下一個動作是向使用者引出插槽值。

    Close— 結束意圖履行程序，並指出不會有使用者的回應。

    ConfirmIntent— 下一個動作是詢問使用者插槽是否正確，並且意圖已準備好履行。
- 意圖-機器人確定用戶嘗試履行的意圖。如需結構的意圖詳細資訊，請參閱。
- prompt — 包含欄位的結構attempt，此結構對應至指定 Amazon Lex V2 提示使用者輸入下一個插槽的次數的值。可能的值是Initial針對第一次嘗試和Retry1、Retry2、Retry3Retry4、和後續嘗試Retry5的值。

requestAttributes

包含客戶端在請求中發送的特定請求屬性的結構。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。

工作階段狀態

使用者與您的 Amazon Lex V2 機器人之間交談的目前狀態。如需結構的工作階段狀態詳細資訊，請參閱。

抄錄

Amazon Lex V2 認為可能符合使用者話語的轉錄清單。如需詳細資訊，請參閱使用語音轉錄信心分數。每個項目都是具有以下格式的對象，其中包含有關一種可能轉錄的信息：

```
{
    "transcription": string,
    "transcriptionConfidence": number,
```

```
    "resolvedContext": {
        "intent": string
    },
    "resolvedSlots": {
        slot name: {
            // see # for details about the structure
        },
        ...
    }
}
```

這些欄位說明如下：

- 轉錄 — Amazon Lex V2 認為可能與使用者音訊話語相符的轉錄。
- 轉錄信心 — 指出 Amazon Lex V2 對意圖符合使用者意圖的信心程度的分數。
- 已解決的前後關聯 — 包含欄位的結構`intent`，該欄位會對應至語音相關意圖。
- 已解決的插槽 — 一種結構，其鍵是由語音解析的每個插槽的名稱。每個插槽名稱都映射到一個包含有關該插槽信息的結構。如需結構的槽詳細資訊，請參閱。

準備回應格式

將 Lambda 函數整合到 Amazon Lex V2 機器人的第二個步驟是瞭解 Lambda 函數回應中的欄位，並決定要操作的參數。下列 JSON 物件顯示傳回給 Amazon Lex V2 的 Lambda 回應的一般格式：

```
{
    "sessionState": {
        // see ###### for details about the structure
    },
    "messages": [
        {
            "contentType": "CustomPayload | ImageResponseCard | PlainText | SSML",
            "content": string,
            "imageResponseCard": {
                "title": string,
                "subtitle": string,
                "imageUrl": string,
                "buttons": [
                    {
                        "text": string,
                        "value": string
```

```
                    },
                    ...
                ]
            }
        },
        ...
    ],
    "requestAttributes": {
        string: string,
        ...
    }
}
```

響應中的每個字段描述如下：

工作階段狀態

您要傳回的使用者與 Amazon Lex V2 機器人之間的交談狀態。如需結構的工作階段狀態詳細資訊，請參閱。此欄位永遠為必填欄位。

messages

Amazon Lex V2 在下一輪交談時傳回給客戶的訊息清單。如果contentType您提供的是PlainTextCustomPayload、或SSML，請在content欄位中撰寫您要傳回給客戶的訊息。如果contentType您提供的是ImageResponseCard，請在該imageResponseCard字段中提供卡的詳細信息。如果您不提供訊息，Amazon Lex V2 會使用建立機器人時定義的適當訊息。

如果是ElicitIntent或，則此messages欄位dialogAction.type為必要欄位ConfirmIntent。

清單中的每個項目都是下列格式的結構，其中包含要傳回給使用者之訊息的相關資訊。請見此處範例：

```
{
    "contentType": "CustomPayload | ImageResponseCard | PlainText | SSML",
    "content": string,
    "imageResponseCard": {
        "title": string,
        "subtitle": string,
        "imageUrl": string,
        "buttons": [
            {
                "text": string,
                "value": string
            },
```

```
            ...
        ]
    }
}
```

下面提供了每個字段的描述：

- 內容類型 — 要使用的訊息類型。

  CustomPayload— 您可以自訂的回應字串，以包含應用程式的資料或中繼資料。

  ImageResponseCard— 帶有客戶可以選擇按鈕的圖像。ImageResponseCard如需詳細資訊，請參閱。

  PlainText— 純文字字串。

  SSML— 包含語音合成標記語言的字串，可自訂音訊回應。
- 內容 — 要傳送給使用者的訊息。如果訊息類型為PlainText、或CustomPayload，請使用此欄位SSML。
- imageResponseCard— 包含要顯示給使用者的回應卡定義。如果訊息類型為，請使用此欄位ImageResponseCard。對映至包含下列欄位的結構：
  - 「標題」— 響應卡的標題。
  - 副標題 — 提示使用者選擇按鈕。
  - imageUrl — 卡片影像的連結。
  - button — 包含按鈕相關資訊的結構清單。每個結構都包含一個text欄位，其中包含要顯示的文字，以及一個value欄位，其中包含要傳送至 Amazon Lex V2 的值 (如果客戶選取該按鈕)。您最多可以包含三個按鈕。

requestAttributes

一種結構，其中包含針對客戶的響應的請求特定屬性。如需更多資訊，請參閱設定請求屬性。此欄位為選用欄位。

響應中的必填字段

至少，Lambda 回應必須包含一個sessionState物件。在其中，提供一個dialogAction對象並指定type字段。視您提供typedialogAction的欄位而定，Lambda 回應可能還有其他必要欄位。這些要求描述如下，以及最少的工作示例：

委派代表

委派可讓 Amazon Lex V2 決定下一個步驟。不需要其他欄位。

```
{
    "sessionState": {
        "dialogAction": {
            "type": "Delegate"
    }
}
```

ElicitIntent

ElicitIntent提示客戶表達意圖。您必須在messages欄位中包含至少一則訊息，才能提示意圖引出。

```
{
    "sessionState": {
        "dialogAction": {
            "type": "ElicitIntent"
    },
    "messages": [
        {
            "contentType": PlainText,
            "content": "How can I help you?"
        }
    ]
}
```

ElicitSlot

ElicitSlot提示客戶提供插槽值。您必須在物件的slotToElicit欄位中包含插槽的名dialogAction稱。您還必須在sessionState物件intent中包含name的。

```
{`
    "sessionState": {
        "dialogAction": {
            "slotToElicit": "OriginCity",
            "type": "ElicitSlot"
        },
        "intent": {
            "name": "BookFlight"
        }
```

```
    }
}
```

ConfirmIntent

ConfirmIntent確認客戶的插槽值，以及意圖是否已準備好可以實現。您必須在sessionState物件intent中包含name的，以及slots要確認的。您還必須在messages字段中包含至少一條消息，以要求用戶確認插槽值。您的訊息應提示「是」或「否」回應。如果使用者回應「是」，Amazon Lex V2 會confirmationState將意圖設定為Confirmed。如果使用者回應「否」，Amazon Lex V2 會confirmationState將意圖設定為Denied。

```
{
    "sessionState": {
        "dialogAction": {
            "type": "ConfirmIntent"
        },
        "intent": {
            "name": "BookFlight",
            "slots": {
                "DepartureDate": {
                    "value": {
                        "originalValue": "tomorrow",
                        "interpretedValue": "2023-05-09",
                        "resolvedValues": [
                            "2023-05-09"
                        ]
                     }
                },
                "DestinationCity": {
                    "value": {
                        "originalValue": "sf",
                        "interpretedValue": "sf",
                        "resolvedValues": [
                            "sf"
                        ]
                    }
                },
                "OriginCity": {
                    "value": {
                        "originalValue": "nyc",
                        "interpretedValue": "nyc",
                        "resolvedValues": [
                            "nyc"
```

```
                        ]
                    }
                }
            }
        }
    },
    "messages": [
        {
            "contentType": PlainText,
            "content": "Okay, you want to fly from {OriginCity} to \
            {DestinationCity} on {DepartureDate}. Is that correct?"
        }
    ]
}
```

Close (關閉)

Close 會結束意圖的履行程序，並指出使用者不需要進一步的回應。您必須在namestatesessionState物件intent中包括和。相容的意圖狀態為FailedFulfilled、和InProgress。

```
"sessionState": {
    "dialogAction": {
        "type": "Close"
    },
    "intent": {
        "name": "BookFlight",
        "state": "Failed | Fulfilled | InProgress"
    }
}
```

Lambda 事件和回應中的常見結構

在 Lambda 回應中，有許多結構會重複發生。本節提供有關這些通用結構的詳細資訊。

意圖

```
"intent": {
    "confirmationState": "Confirmed | Denied | None",
    "name": string,
    "slots": {
```

```
        // see # for details about the structure
    },
    "state": "Failed | Fulfilled | FulfillmentInProgress | InProgress |
 ReadyForFulfillment | Waiting",
    "kendraResponse": {
        // Only present when intent is KendraSearchIntent. For details, see
// https://docs.aws.amazon.com/kendra/latest/dg/API_Query.html#API_Query_ResponseSyntax
       }
}
```

該字intent段映射到具有以下字段的對象：

確認狀態

指出使用者是否已確認意圖的位置，且意圖已準備好履行。可能的值如下：

Confirmed— 使用者確認槽值是否正確。

Denied— 使用者指示插槽值不正確。

None— 使用者尚未到達確認階段。

name

意圖的名稱。

slots

有關實現意圖所需的插槽的信息。如需結構的槽詳細資訊，請參閱。

state

表示意圖的履行狀態。可能的值如下：

Failed— 機器人無法實現意圖。

Fulfilled— 機器人已完成意圖的履行。

FulfillmentInProgress— 機器人正處於實現意圖的中間。

InProgress— 機器人正處於引出實現意圖所必需的插槽值的中間。

ReadyForFulfillment— 機器人已經為意圖引出了所有插槽值，並準備好實現意圖。

Waiting— 機器人正在等待用戶的響應（僅限於流式對話）。

kendraResponse

包含 Kendra 搜尋查詢結果的相關資訊。僅當意圖為時，此欄位才會顯示KendraSearchIntent。如需詳細資訊，請參閱 Kendra 的查詢 API 呼叫中的回應語法。

槽

該slots字段存在於intent結構中，並映射到一個結構，該結構的鍵是該意圖的插槽的名稱。如果插槽不是多值插槽 (請參閱以在插槽中使用多個值取得更多詳細資訊)，則會將其對映至具有下列格式的結構。請注意，shape是Scalar.

```
{
    slot name: {
        "shape": "Scalar",
        "value": {
            "originalValue": string,
            "interpretedValue": string,
            "resolvedValues": [
                string,
                ...
            ]
        }
    }
}
```

如果插槽是多值插槽，它所對應的物件會包含另一個名為的欄位values，該欄位會對應至結構清單，每個欄位都包含組成多值插槽之插槽的相關資訊。清單中每個物件的格式與一般槽所對映之物件的格式相符。請注意，shape是List，但下面shape的零組件槽的values是Scalar。

```
{
    slot name: {
    "shape": "List",
    "value": {
        "originalValue": string,
        "interpretedValue": string,
        "resolvedValues": [
            string,
            ...
        ]
    },
```

```
    "values": [
        {
            "shape": "Scalar",
            "value": {
                "originalValue": string,
                "interpretedValue": string,
                "resolvedValues": [
                    string,
                    ...
                ]
            }
        },
        {
            "shape": "Scalar",
            "value": {
                "originalValue": string,
                "interpretedValue": string,
                "resolvedValues": [
                    string,
                    ...
                ]
            }
        },
        ...
    ]
}
```

插槽物件中的欄位說明如下：

shape

插槽的形狀。`List`如果插槽中有多個值 (請參閱以在插槽中使用多個值取得更多詳細資訊)，`Scalar`否則為此值。

value

物件，其中包含使用者為插槽提供的值和 Amazon Lex 解釋的相關資訊，格式如下：

```
{
    "originalValue": string,
    "interpretedValue": string,
    "resolvedValues": [
        string,
```

```
        ...
    ]
}
```

這些欄位說明如下：

- 原始值 — Amazon Lex 判斷使用者對插槽的回應部分與插槽值相關。
- 解譯值 — Amazon Lex 為插槽決定的值，在使用者輸入的情況下。
- 已解決的值 — Amazon Lex 判定為使用者輸入的可能解析度的值清單。

values

物件清單，其中包含組成多值槽之槽的相關資訊。每個物件的格式與一般插槽的格式相符，並使用上述value欄位的shape和欄位。 values只有在槽由多個值組成時才會出現 (如在插槽中使用多個值需詳細資訊，請參閱)。下面的 JSON 對象顯示了兩個組件插槽：

```
"values": [
    {
        "shape": "Scalar",
        "value": {
            "originalValue": string,
            "interpretedValue": string,
            "resolvedValues": [
                string,
                ...
            ]
        }
    },
    {
        "shape": "Scalar",
        "value": {
            "originalValue": string,
            "interpretedValue": string,
            "resolvedValues": [
                string,
                ...
            ]
        }
    },
    ...
]
```

工作階段狀態

此sessionState欄位會對應至包含與使用者之交談狀態相關資訊的物件。顯示在對象中的實際字段取決于對話框操作的類型。響應中的必填字段如需 Lambda 回應中的必要欄位，請參閱。sessionState物件的格式如下：

```
"sessionState": {
    "activeContexts": [
        {
            "name": string,
            "contextAttributes": {
                string: string
            },
            "timeToLive": {
                "timeToLiveInSeconds": number,
                "turnsToLive": number
            }
        },
        ...
    ],
    "sessionAttributes": {
        string: string,
        ...
    },
    "runtimeHints": {
        "slotHints": {
            intent name: {
                slot name: {
                    "runtimeHintValues": [
                        {
                            "phrase": string
                        },
                        ...
                    ]
                },
                ...
            },
            ...
        }
    },
    "dialogAction": {
        "slotElicitationStyle": "Default | SpellByLetter | SpellByWord",
        "slotToElicit": string,
```

```
        "type": "Close | ConfirmIntent | Delegate | ElicitIntent | ElicitSlot"
    },
    "intent": {
        // see ## for details about the structure
    },
    "originatingRequestId": string
}
```

這些欄位說明如下：

主動式環境

物件清單，其中包含使用者在工作階段中使用之前後關聯的相關資訊。使用上下文來促進和控制意圖識別。如需前後關聯的詳細資訊，請參閱設定意圖上下文。每個物件的格式如下：

```
{
    "name": string,
    "contextAttributes": {
        string: string
    },
    "timeToLive": {
        "timeToLiveInSeconds": number,
        "turnsToLive": number
    }
}
```

這些欄位說明如下：

- 名稱 — 前後關聯的名稱。
- 上下文屬性 — 物件，其中包含前後關聯的屬性名稱以及它們所對應的值。
- timeToLive— 物件，指定前後關聯保持使用中的時間長度。此物件可以包含下列一個或兩個欄位：
  - timeToLiveInSeconds— 前後關聯保持使用中狀態的秒數。
  - turnsToLive— 前後關聯保持使用中的迴轉數。

sessionAttributes

代表會話特定上下文信息的鍵/值對的映射。如需詳細資訊，請參閱設定階段屬性。物件的格式如下：

```
{
```

```
    string: string,
    ...
}
```

執行階段提示

為客戶可能用於插槽的片語提供提示，以改善音訊辨識度。您在提示中提供的值可以提高這些值的音頻識別，而不是聽起來相似的單詞。runtimeHints物件的格式如下：

```
{
    "slotHints": {
        intent name: {
            slot name: {
                "runtimeHintValues": [
                    {
                        "phrase": string
                    },
                    ...
                ]
            },
            ...
        },
        ...
    }
}
```

該字slotHints段映射到一個對象，該對象的字段是機器人中的意圖的名稱。每個意圖名稱都映射到一個對象，該對象的字段是該意圖的插槽的名稱。每個插槽名稱映射到具有單個字段的結構runtimeHintValues，該字段是對象列表。每個物件都包含對應至提示的phrase欄位。

dialogAction

決定 Amazon Lex V2 要採取的下一個動作。物件的格式如下：

```
{
    "slotElicitationStyle": "Default | SpellByLetter | SpellByWord",
    "slotToElicit": string,
    "type": "Close | ConfirmIntent | Delegate | ElicitIntent | ElicitSlot"
}
```

這些欄位說明如下：

- slotElicitationStyle— 決定 Amazon Lex V2 如何解譯來自使用者的音訊輸入 (如果dialogAction是ElicitSlot)。type如需詳細資訊，請參閱使用拼字樣式擷取位置值。可能的值如下：

  Default— Amazon Lex V2 會以預設方式解譯音訊輸入，以滿足插槽。

  SpellByLetter— Amazon Lex V2 會偵聽使用者對插槽值的拼寫。

  SpellByWord— Amazon Lex V2 會使用與每個字母相關聯的單字 (例如「a in apple」) 偵聽使用者對插槽值的拼寫。
- slotToElicit— 定義要從使用者引出的插槽 (如果type的dialogAction是)。ElicitSlot
- type — 定義機器人應該執行的動作。可能的值如下：

  Delegate— 讓 Amazon Lex V2 確定下一個步驟。

  ElicitIntent— 提示客戶表達意圖。

  ConfirmIntent— 確認客戶的位置值，以及意圖是否已準備好履行。

  ElicitSlot— 提示客戶為意圖提供插槽值。

  Close— 結束意圖履行程序。

意圖

如意圖需intent欄位的結構，請參閱。

originatingRequestId

要求的唯一識別碼。此欄位對於 Lambda 回應而言是選擇性的。

建立 Lambda 函數並將其附加至機器人別名

建立 Lambda 函數

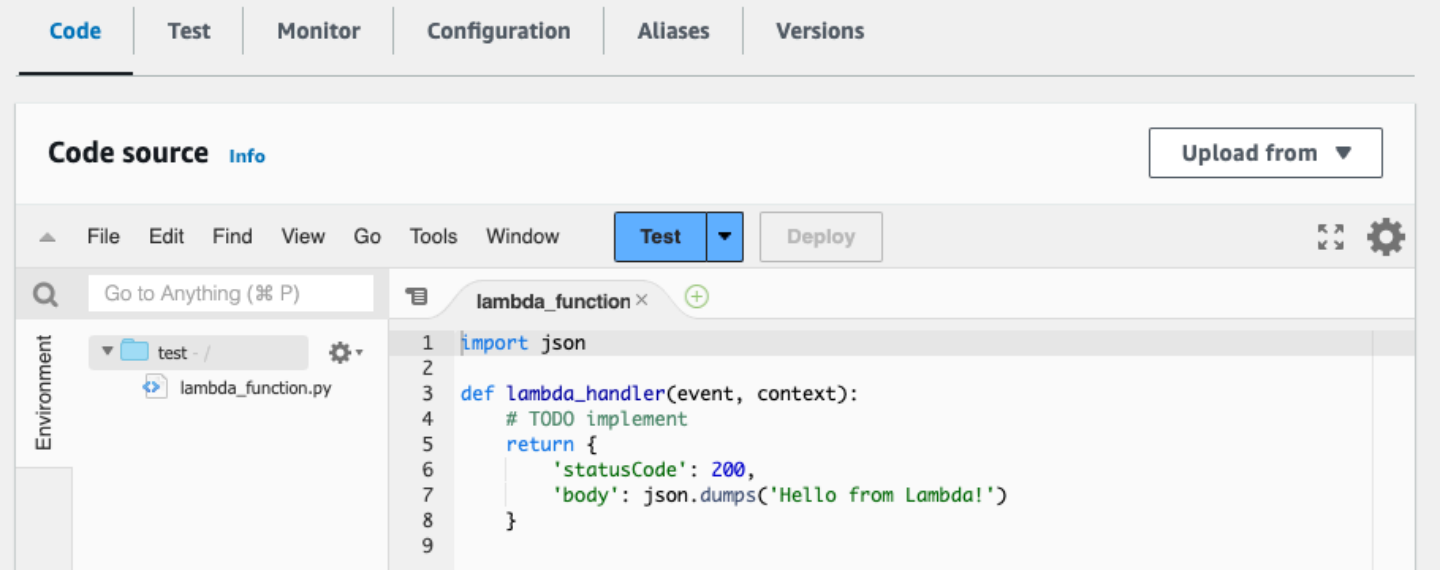
若要為您的 Amazon Lex V2 機器人建立 Lambda 函數，請AWS Lambda從您的機器人存取AWS Management Console並建立新函數。有關更多詳細信息，請參閱AWS Lambda開發人員指南AWS Lambda。

1. 請登入 AWS Management Console，並開啟位於 https://console.aws.amazon.com/lambda/ 的 AWS Lambda 主控台。

2. 選擇左側邊欄中的「功能」。
3. 選取 Create function (建立函數)。
4. 您可以選取從頭開始撰寫最少程式碼、使用藍圖從清單中選取常見使用案例的範例程式碼，或選取容器映像選取要為您的函數部署的容器映像。如果您從頭開始選取「作者」，請繼續執行下列步驟：

   a. 給你的函數一個有意義的函數名稱來描述它的作用。
   b. 從「執行階段」下的下拉式功能表中選擇一種語言，以將函數寫入。
   c. 為您的函數選取指令集架構。
   d. 根據預設，Lambda 會建立具有基本權限的角色。若要使用現有角色或使用AWS原則範本建立角色，請展開 [變更預設執行角色] 功能表，然後選取選項。
   e. 展開 [進階設定] 功能表以設定更多選項。
5. 選取 Create function (建立函數)。

下圖顯示了從頭開始創建新函數時所看到的內容：



Lambda 處理常式函數會因您使用的語言而有所不同。它最低限度地需要一個 `event` JSON 對象作為參數。您可以看到在 Amazon Lex V2 提供的欄位解譯輸入事件格式。`event`修改處理常式函數，最終傳回符合中所述格式的 `response` JSON 物件準備回應格式。

完成寫入函數後，請選取 [部署] 以允許使用函數。

請記住，您最多可以將每個機器人別名與一個 Lambda 函數建立關聯。不過，您可以在 Lambda 程式碼中為機器人定義所需的任意數量的函數，並在 Lambda 處理常式函數中呼叫這些函數。例如，雖然

相同機器人別名中的所有意圖都必須呼叫相同的 Lambda 函數，但您可以建立路由器函數，為每個意圖啟動個別函數。以下是您可以在應用程式中使用或修改的範例路由器功能：

```
import os
import json
import boto3

# reuse client connection as global
client = boto3.client('lambda')

def router(event):
    intent_name = event['sessionState']['intent']['name']
    fn_name = os.environ.get(intent_name)
    print(f"Intent: {intent_name} -> Lambda: {fn_name}")
    if (fn_name):
        # invoke lambda and return result
        invoke_response = client.invoke(FunctionName=fn_name, Payload =
 json.dumps(event))
        print(invoke_response)
        payload = json.load(invoke_response['Payload'])
        return payload
    raise Exception('No environment variable for intent: ' + intent_name)

def lambda_handler(event, context):
    print(event)
    response = router(event)
    return response
```

添加和調用一個 Lambda 函數

若要呼叫 Amazon Lex V2 機器人中的 Lambda 函數，您必須先將函數附加到機器人別名，然後在機器人叫用函數的交談中設定點。您可以透過主控台或 API 作業來執行這些步驟。

您可以在與使用者交談的下列時間點使用 Lambda 函數：

- 在意圖被識別後的初始響應。例如，在用戶說他們要訂購比薩餅之後。
- 從用戶引出插槽值之後。例如，在用戶告訴機器人他們想要訂購的比薩餅大小之後。
- 在每次重試之間引出一個插槽。例如，如果客戶不使用公認的比薩餅尺寸。
- 確認意圖時。例如，在確認披薩訂單時。
- 為了實現一個意圖。例如，要放置一個比薩餅的訂單。
- 在實現意圖之後，並在您的機器人關閉對話之前。例如，切換到訂購飲料的意圖。

主題

- 使用主控台
- 使用 API 作業

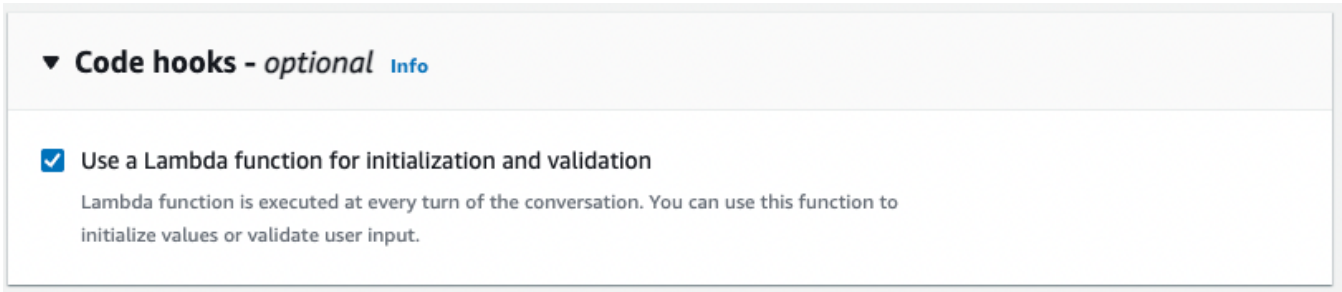
使用主控台

將 Lambda 函數附加至機器人別名

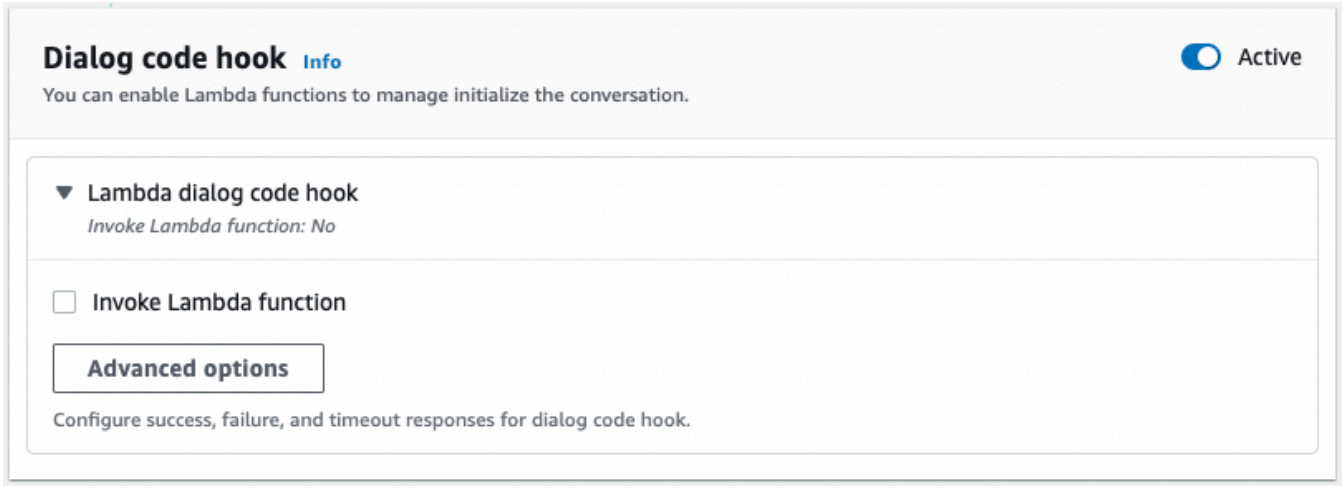
1. 登錄到AWS Management Console並打開 Amazon Lex 控制台 https://console.aws.amazon.com/lex/.
2. 從左側面板中選擇機器人，然後從機器人列表中選擇要附加 Lambda 函數的機器人的名稱。
3. 在左側面板中，選取「部署」功能表下的「別名」。
4. 從別名清單中，選擇要附加 Lambda 函數的別名名稱。
5. 在「語言」面板中，選取您要使用 Lambda 函數的語言。如果面板中沒有顯示語言，請選取「管理別名中的語言」以新增語言。
6. 在 [來源] 下拉式功能表中，選擇您要附加的 Lambda 函數名稱。
7. 在 Lambda 函數版本或別名下拉式功能表中，選擇您要使用的 Lambda 函數的版本或別名。然後選取 Save (儲存)。相同的 Lambda 函數用於機器人所支援的語言中的所有意圖。

設定意圖以叫用 Lambda 函數

1. 選取機器人之後，在您要叫用 Lambda 函數之機器人語言下方的左側功能表中選取 [意圖]。
2. 選擇您要呼叫 Lambda 函數以開啟意圖編輯器的意圖。
3. 設定 Lambda 程式碼掛接有兩個選項：
    1. 若要在交談的每個步驟之後叫用 Lambda 函數，請捲動至意圖編輯器底部的程式碼掛接區段，然後選取 [使用 Lambda 函數進行初始化和驗證] 核取方塊，如下圖所示：



2. 或者，在要叫用 Lambda 函數的交談階段中，使用對話方塊程式碼掛接區段。「對話方塊程式碼掛接」區段顯示如下：



有兩種方法可以控制 Amazon Lex V2 呼叫程式碼掛接以進行回應的方式：

- 切換「作用中」按鈕，將其標示為作用中或非作用中。當程式碼掛接處於作用中狀態時，Amazon Lex V2 會呼叫程式碼掛接。當程式碼掛接處於非作用中狀態時，Amazon Lex V2 不會執行程式碼掛接。
- 展開 Lambda 對話方塊程式碼勾點區段，然後選取叫用 Lambda 函數核取方塊，將其標示為啟用或停用。您只能在程式碼掛接標示為作用中時啟用或停用程式碼掛接。標記為啟用時，代碼掛鉤將正常運行。停用程式碼時，不會呼叫程式碼掛接，Amazon Lex V2 就像程式碼掛接成功傳回一樣。若要在對話方塊程式碼掛接成功、失敗或逾時之後設定回應，請選取 [進階選項]

您可以在下列交談階段叫用 Lambda 程式碼掛接：

- 若要呼叫函數作為初始回應，請捲動至「初始回應」區段，展開「回應」旁邊的箭號以確認使用者的要求，然後選取進階選項。在彈出的菜單底部找到對話框代碼掛鉤部分。
- 若要在插槽引出後叫用函數，請捲動至「插槽」區段，展開相關「提示插槽」旁邊的箭頭，然後選取「進階選項」。在彈出的菜單底部附近找到對話框代碼掛鉤部分默認值上方。

  您也可以在每次引出後調用該函數。若要這麼做，請展開「插槽提示」區段中的機器人會產生資訊，選取更多提示選項，然後選取每次引出後叫用 Lambda 程式碼掛接旁邊的核取方塊。
- 若要呼叫功能進行意圖確認，請捲動至「確認」區段，展開「提示」以確認意圖旁邊的箭頭，然後選取「進階」選項。在彈出的菜單底部找到對話框代碼掛鉤部分。

- 若要叫用意圖履行的函數，請捲動至「履行」區段。切換「作用中」按鈕，將程式碼掛接設定為作用中。展開 [成功出貨時] 旁邊的箭頭，然後選取 [進階選項]。選取「履行 Lambda 程式碼勾點」區段下的「使用 Lambda 函數進行履行」旁邊的核取方塊，將程式碼掛接設定為啟用。

4. 設定要呼叫 Lambda 函數的交談階段後，再次建置機器人以測試函數。

使用 API 作業

將 Lambda 函數附加至機器人別名

如果您要建立新的機器人別名，請使用此CreateBotAlias作業連接 Lambda 函數。若要將 Lambda 函數附加至現有的機器人別名，請使用此UpdateBotAlias作業。修改botAliasLocaleSettings欄位以包含正確的設定：

```
{
    "botAliasLocaleSettings" : {
        locale: {
            "codeHookSpecification": {
                "lambdaCodeHook": {
                    "codeHookInterfaceVersion": "1.0",
                    "lambdaARN": "arn:aws:lambda:region:account-id:function:function-
name"
                }
            },
            "enabled": true
        },
        ...
    }
}
```

1. 此botAliasLocaleSettings欄位會對應至物件，其索引鍵為您要附加 Lambda 函數的地區設定。支援的語言和地區設定如需支援的地區設定清單，以及有效金鑰的代碼，請參閱。

2. 若要尋找 Lambda 函數的功能，請在 https://console.aws.amazon.com/lambda/home 開啟AWS Lambda主控台，選取左側邊列中的函數，然後選取要與機器人別名建立關聯的函數。lambdaARN在功能概述的右側，找到功能 ARN **lambdaARN** 下的。它應該包含一個地區，帳戶 ID 和函數的名稱。

3. 若要允許 Amazon Lex V2 叫用別名的 Lambda 函數，請將enabled欄位設定為true。

設定意圖以叫用 Lambda 函數

若要在意圖期間設定 Lambda 函數叫用，請在建立新意圖時使用CreateIntent作UpdateIntent業；如果要在現有意圖中叫用函數，請使用作業。控制意圖作業中 Lambda 函數叫用的欄位為dialogCodeHook、initialResponseSettingintentConfirmationSetting、和fulfillmentCodeHook。

如果您在插槽引出期間呼叫函數，請在建立新插槽時使用此CreateSlot作業，或者在現有插槽中叫用函數的UpdateSlot作業。控制插槽作業中 Lambda 函數叫用的欄位是valueElicitationSetting物件slotCaptureSetting的欄位。

1. 若要將 Lambda 對話方塊程式碼掛接設定為在每次交談後執行，請將enabled欄位中下列DialogCodeHookSettings物件的dialogCodeHook欄位設定為true：

   ```
   "dialogCodeHook": {
       "enabled": boolean
   }
   ```

2. 或者，您可以將 Lambda 對話方塊程式碼掛接設定為僅在交談中的特定點執行，方法是修改與您要叫用函數之交談階段對應的結構內的codeHook和/或elicitationCodeHook欄位。若要使用 Lambda 對話方塊程式碼掛接來實現意圖，請使用CreateIntent或UpdateIntent作業中的fulfillmentCodeHook欄位。這三種類型的代碼掛接的結構和用法如下：

代碼掛鉤

此codeHook欄位會定義程式碼掛接的設定，以便在交談中的指定階段執行。它是具有以下結構的DialogCodeHookInvocationSetting對象：

```
"codeHook": {
    "active": boolean,
    "enableCodeHookInvocation": boolean,
    "invocationLabel": string,
    "postCodeHookSpecification": PostDialogCodeHookInvocationSpecification object,
}
```

- 將 Amazon Lex V2 true 的active欄位變更為，以呼叫交談中該點的程式碼勾點。
- 將 Amazon Lex V2 true 的enableCodeHookInvocation欄位變更為，以允許程式碼掛接正常執行。如果您加上標記false，Amazon Lex V2 就會像程式碼掛接成功傳回一樣。
- invocationLabel指示從中叫用程式碼掛接的對話方塊步驟。

- 使用此postCodeHookSpecification欄位可指定程式碼掛接成功、失敗或逾時後所發生的動作和訊息。

elicitationCodeHook

此elicitationCodeHook欄位定義程式碼掛接的設定，以便在需要重新引導一個或多個插槽時執行。如果插槽引出失敗或意圖確認被拒絕，則可能會發生這種情況。該elicitationCodeHook字段是具有以下結構的ElicitationCodeHookInvocationSetting對象：

```
"elicitationCodeHook": {
    "enableCodeHookInvocation": boolean,
    "invocationLabel": string
}
```

- 將 Amazon Lex V2 true 的enableCodeHookInvocation欄位變更為，以允許程式碼掛接正常執行。如果您加上標記false，Amazon Lex V2 就會像程式碼掛接成功傳回一樣。
- invocationLabel指示從中叫用程式碼掛接的對話方塊步驟。

fulfillmentCodeHook

此fulfillmentCodeHook欄位定義程式碼掛接要執行以達成意圖的設定。它會對應至下列FulfillmentCodeHookSettings物件：

```
"fulfillmentCodeHook": {
    "active": boolean,
    "enabled": boolean,
    "fulfillmentUpdatesSpecification": FulfillmentUpdatesSpecification object,
    "postFulfillmentStatusSpecification": PostFulfillmentStatusSpecification object
}
```

- 將 Amazon Lex V2 true 的active欄位變更為，以呼叫交談中該點的程式碼勾點。
- 將 Amazon Lex V2 true 的enabled欄位變更為，以允許程式碼掛接正常執行。如果您加上標記false，Amazon Lex V2 就會像程式碼掛接成功傳回一樣。
- 使用此fulfillmentUpdatesSpecification欄位可指定在完成意圖期間顯示為更新使用者的訊息，以及與使用者相關聯的時間。
- 使用此postFulfillmentStatusSpecification欄位可指定程式碼掛接成功、失敗或逾時後發生的訊息和動作。

您可以將active和enableCodeHookInvocation/欄位設定為，在交談中的下列enabled位置叫用 Lambda 程式碼掛接true：

初始回應期間

若要在辨識出意圖之後在初始回應中叫用 Lambda 函數，請使用CreateIntent或UpdateIntent作業initialResponse欄位中的codeHook結構。該initialResponse字段映射到以下InitialResponseSetting對象：

```
"initialResponse": {
    "codeHook": {
        "active": boolean,
        "enableCodeHookInvocation": boolean,
        "invocationLabel": string,
        "postCodeHookSpecification": PostDialogCodeHookInvocationSpecification object,
    },
    "initialResponse": FulfillmentUpdatesSpecification object,
    "nextStep": PostFulfillmentStatusSpecification object,
    "conditional": ConditionalSpecification object
}
```

插槽引出後或插槽重新引出期間

若要在引出插槽值之後叫用 Lambda 函數，請使用slotCaptureSettingCreateSlot或UpdateSlot作業valueElicitation欄位內的欄位。該slotCaptureSetting字段映射到以下SlotCaptureSetting對象：

```
"slotCaptureSetting": {
    "captureConditional": ConditionalSpecification object,
    "captureNextStep": DialogState object,
    "captureResponse": ResponseSpecification object,
    "codeHook": {
        "active": true,
        "enableCodeHookInvocation": true,
        "invocationLabel": string,
        "postCodeHookSpecification": PostDialogCodeHookInvocationSpecification object,
    },
    "elicitationCodeHook": {
        "enableCodeHookInvocation": boolean,
        "invocationLabel": string
    },
    "failureConditional": ConditionalSpecification object,
    "failureNextStep": DialogState object,
```

```
    "failureResponse": ResponseSpecification object
}
```

- 若要在插槽引出成功之後叫用 Lambda 函數，請使用欄位codeHook。
- 若要在插槽引出失敗且 Amazon Lex V2 嘗試重試插槽引出後呼叫 Lambda 函數，請使用欄位。elicitationCodeHook

意圖確認或拒絕之後

若要在確認意圖時叫用 Lambda 函數，請使用CreateIntent或UpdateIntent作業的intentConfirmationSetting欄位。該intentConfirmation字段映射到以下IntentConfirmationSetting對象：

```
"intentConfirmationSetting": {
    "active": boolean,
    "codeHook": {
        "active": boolean,
        "enableCodeHookInvocation": boolean,
        "invocationLabel": string,
        "postCodeHookSpecification": PostDialogCodeHookInvocationSpecification object,
    },
    "confirmationConditional": ConditionalSpecification object,
    "confirmationNextStep": DialogState object,
    "confirmationResponse": ResponseSpecification object,
    "declinationConditional": ConditionalSpecification object,
    "declinationNextStep": FulfillmentUpdatesSpecification object,
    "declinationResponse": PostFulfillmentStatusSpecification object,
    "elicitationCodeHook": {
        "enableCodeHookInvocation": boolean,
        "invocationLabel": string,
    },
    "failureConditional": ConditionalSpecification object,
    "failureNextStep": DialogState object,
    "failureResponse": ResponseSpecification object,
    "promptSpecification": PromptSpecification object
}
```

- 若要在使用者確認意圖及其插槽之後叫用 Lambda 函數，請使用codeHook欄位。
- 若要在使用者拒絕意圖確認後呼叫 Lambda 函數，而 Amazon Lex V2 嘗試重試插槽引出，請使用欄位。elicitationCodeHook

在意圖履行期間

若要叫用 Lambda 函數來達成意圖，請在CreateIntent或UpdateIntent作業中使用`fulfillmentCodeHook`欄位。該`fulfillmentCodeHook`字段映射到以下FulfillmentCodeHookSettings對象：

```
{
    "active": boolean,
    "enabled": boolean,
    "fulfillmentUpdatesSpecification": FulfillmentUpdatesSpecification object,
    "postFulfillmentStatusSpecification": PostFulfillmentStatusSpecification object
}
```

3. 設定要呼叫 Lambda 函數的交談階段後，請使用`BuildBotLocale`作業重建機器人以測試函數。

對 Lambda 函數進行除錯

Amazon CloudWatch 日誌是用來追蹤 API 呼叫和指標的工具，您可以使用這些工具來協助偵錯 Lambda 函數。當您在主控台或使用 API 呼叫測試機器人時，會 CloudWatch 記錄交談的每個步驟。如果您在 Lambda 程式碼中使用列印函數，也 CloudWatch 會顯示它。

若要檢視 Lambda 函數的 CloudWatch 記錄

1. 請登入AWS Management Console並開啟 CloudWatch 主控台，網址為 https://console.aws.amazon.com/cloudwatch/。
2. 在左側工具列的 [記錄] 功能表下，選取 [記錄群組]。
3. 選取您的 Lambda 函數日誌群組，格式應該是`/aws/lambda/`*`function-name`*。
4. 日誌流的列表包含一個與機器人的每個會話的日誌。選擇要檢視的記錄資料流。
5. 在記錄事件清單中，選取時間戳記旁的向右箭頭以展開該事件的詳細資訊。您從 Lambda 程式碼列印的任何內容都會顯示為記錄事件。使用此資訊來偵錯您的程式碼。
6. 偵錯程式碼之後，請記得部署 Lambda 函數，如果您使用主控台，請在重新測試機器人行為之前重新載入測試視窗。

自訂機器人互動

瞭解下列功能，您可以透過展開和調整使用者的預設行為來自訂機器人與使用者的互動：

主題

分析使用者話語的情緒

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。

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

您可以使用主控台或使用 Amazon Lex API 為機器人啟用情緒分析。您可以針對機器人的別名啟用情緒分析。在亞馬遜萊克斯控制台上：

1. 選擇別名。
2. 在詳細資訊中，選擇編輯。
3. 選擇 [啟用情緒分析以開啟或關閉情緒分析]。
4. 選擇 Confirm (確認) 以儲存變更。

如果您正在使用 API，請在 `detectSentiment` 欄位設定為 `true` 的情況下呼叫 CreateBotAlias 操作。

啟用情緒分析時，RecognizeText和RecognizeUtterance作業的回應會傳回`interpretations`結構`sentimentResponse`中名為的欄位，其中包含其他中繼資料。`sentimentResponse` 欄位有兩個欄位，`sentiment` 和 `sentimentScore`，其中包含情緒分析的結果。如果您使用 Lambda 函數，則`sentimentResponse`欄位會包含在傳送至函數的事件資料中。

以下是 `sentimentResponse` 欄位傳回 `RecognizeText` 或 `RecognizeUtterance` 回應一部分的範例。

```
sentimentResponse {
    "sentimentScore": {
        "mixed": 0.030585512690246105,
        "positive": 0.94992071056365967,
        "neutral": 0.0141543131828308,
        "negative": 0.00893945890665054
    },
    "sentiment": "POSITIVE"
}
```

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

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。

使用可信度分數

Amazon Lex V2 有兩個步驟可用來判斷使用者所說的內容。第一個是自動語音辨識 (ASR)，會建立使用者音訊說話的文字記錄。第二個，自然語言理解 (NLU) 決定了用戶的話語，以識別用戶的意圖或插槽的價值的含義。

依預設，亞馬遜萊克斯 V2 會傳回 ASR 和 NLU 最有可能的結果。有時候，Amazon Lex V2 可能很難判斷最可能的結果。在這種情況下，它會傳回數個可能的結果以及可信度分數，指出結果正確的可能性。可信度分數是 Amazon Lex V2 提供的分級，顯示其對結果的相對可信度。置信度分數的範圍從 0.0 到 1.0。

您可以使用您的網域知識和可信度分數，以協助判斷 ASR 或 NLU 結果的正確解釋。

ASR (或轉錄) 可信度分數是 Amazon Lex V2 對特定轉錄正確性有多大信心的評分。NLU (或意圖) 可信度分數是 Amazon Lex V2 對於最上層轉錄指定意圖正確的信心程度評分。使用最適合您應用程式的信賴度分數。

主題

- 使用意圖信賴度分數
- 使用語音轉錄信心分數

使用意圖信賴度分數

當使用者發出說話時，Amazon Lex V2 會使用自然語言理解 (NLU) 來瞭解使用者的要求並傳回正確的意圖。依預設，Amazon Lex V2 會傳回您的機器人所定義的最可能意圖。

在某些情況下，Amazon Lex V2 可能很難判斷最可能的意圖。例如，使用者可能會發出不明確的話語，或者可能有兩個相似的意圖。為了幫助確定正確的意圖，您可以在解釋列表中結合您的領域知識與 NLU 信賴度分數。可信度分數是 Amazon Lex V2 提供的分級，顯示意圖是正確意圖的信心程度。

若要判斷解譯中兩個意圖之間的差異，您可以比較其信心分數。例如，如果一個意圖的信賴度分數為 0.95，而另一個意圖的分數為 0.65，則第一個意圖可能是正確的。但是，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糊，您可能可以在應用程序中使用域知識進行區分。

您也可以使用可信度分數來建立測試應用程式，以判斷意圖話語的變更是否會影響機器人的行為。例如，您可以使用一組語音來獲得機器人意圖的可信度分數，然後使用新的話語更新意圖。然後，您可以檢查可信度分數，以查看是否有所改善。

Amazon Lex V2 傳回的可信度分數是比較值。您不應該依賴它們作為絕對分數。這些值可能會根據對亞馬遜 Lex V2 的改進而改變。

Amazon Lex V2 會在每個回應中傳回最可能的意圖和最多 4 個替代意圖，以及其在`interpretations`結構中的相關分數。下列 JSON 程式碼會顯示RecognizeText作業回應中的`interpretations`結構：

```
   "interpretations": [
      {
         "intent": {
            "confirmationState": "string",
            "name": "string",
            "slots": {
               "string" : {
                  "value": {
                     "interpretedValue": "string",
                     "originalValue": "string",
                     "resolvedValues": [ "string" ]
                  }
               }
            },
            "state": "string"
         },
         "nluConfidence": number
```

```
    }
  ]
```

亞馬遜。 FallbackIntent

在兩種情況下，亞馬遜 Lex V2 返回AMAZON.FallbackIntent為最高意圖：

1. 如果所有可能的意圖的置信度分數小於置信閾值。您可以使用預設臨界值，也可以設定自己的臨界值。如果您已設定完成AMAZON.KendraSearchIntent設定，Amazon Lex V2 也會在此情況下傳回它。
2. 如果解釋信心高AMAZON.FallbackIntent於所有其他意圖的解釋信心。

請注意，亞馬遜 Lex V2 不會顯示AMAZON.FallbackIntent.

設定和變更可信度閾值

信賴閾值必須是介於 0.00 和 1.00 之間的數字。您可以透過下列方式設定機器人中每種語言的臨界值：

使用亞馬遜萊克斯 V2 控制台

- 要在使用「添加語言」向機器人添加語言時設置閾值，可以在「可信度分數閾值」面板中插入所需的值。
- 若要更新臨界值，您可以在「語言詳細資料」面板中選取機器人語言的「編輯」。然後在「可信度分數閾值」面板中插入所需的值。

使用 API 作業

- 若要設定臨界值，請設定CreateBotLocale作業的nluIntentConfidenceThreshold參數。
- 若要更新可信度閾值，請設定UpdateBotLocale作業的nluIntentConfidenceThreshold參數。

會話管理

若要變更 Amazon Lex V2 在與使用者交談中使用的意圖，您可以使用對話方塊程式碼掛接 Lambda 函數的回應，或者在自訂應用程式中使用工作階段管理 API。

使用拉姆達函數

當您使用 Lambda 函數時，Amazon Lex V2 會以包含函數輸入的 JSON 結構呼叫該函數。JSON 結構包含一個名為的欄位currentIntent，其中包含 Amazon Lex V2 已識別為使用者說話的最可能意圖

的意圖。JSON 結構也包含一個alternativeIntents欄位，其中包含最多四個可滿足使用者意圖的額外意圖。每個意圖都包含一個名為nluIntentConfidenceScore的欄位，其中包含 Amazon Lex V2 指派給意圖的可信度分數。

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

如需詳細資訊，請參閱使用AWS Lambda函數啟用自訂邏輯。

使用工作階段管理 API

若要使用與目前意圖不同的意圖，請使用此PutSession作業。例如，如果您決定第一個替代方案比 Amazon Lex V2 選擇的意圖更可取，則可以使用該PutSession操作來變更意圖，以便使用者與之互動的下一個意圖就是您選取的意圖。

如需詳細資訊，請參閱 使用亞馬遜萊克斯 V2 API 管理工作階段。

使用語音轉錄信心分數

當使用者發出語音說話時，Amazon Lex V2 會使用自動語音辨識 (ASR) 來轉錄使用者的請求，然後再進行解譯。根據預設，Amazon Lex V2 會使用最有可能的音訊轉錄進行解譯。

在某些情況下，可能會有多個可能的音頻轉錄。例如，使用者可能會發出含糊不清的聲音，例如「我的名字是 John」，可能會被理解為「我的名字是胡安」。在這種情況下，您可以使用消歧義技術或將您的領域知識與轉錄置信度分數相結合，以幫助確定轉錄列表中的哪些轉錄是正確的轉錄。

Amazon Lex V2 包含最上層轉錄和最多兩個替代轉錄，供使用者在向 Lambda 程式碼掛接函數的請求中輸入。每個轉錄都包含一個可信度分數，表明它是正確的轉錄。每個轉錄還包括從用戶輸入推斷的任何插槽值。

您可以比較兩個轉錄的可信度分數，以確定它們之間是否存在模糊性。例如，如果一個轉錄的置信度分數為 0.95，而另一個轉錄的可信度分數為 0.65，則第一個轉錄可能是正確的，並且它們之間的模糊性很低。如果兩個轉錄的信心分數為 0.75 和 0.72，則它們之間的模糊性很高。您可以使用您的領域知識來區分它們。

例如，如果兩個可信度分數為 0.75 和 0.72 的記錄單中推斷的插槽值是「John」和「Juan」，您可以查詢資料庫中的使用者是否存在這些名稱，並排除其中一項轉錄。如果「John」不是資料庫中的使用者，而且「Juan」是，您可以使用對話方塊程式碼掛接將名字的推斷位置值變更為「Juan」。

Amazon Lex V2 傳回的可信度分數是比較值。不要依賴它們作為絕對分數。這些值可能會根據對 Amazon Lex V2 的改進而改變。

音訊轉錄置信度分數僅提供英文 (GB) (en_GB) 和英文 (美國) (en_US) 語言。信賴度分數僅支援 8 kHz 音訊輸入。Amazon Lex V2 主控台上測試視窗的音訊輸入不會提供轉錄可信度分數，因為它使用 16 kHz 音訊輸入。

> **Note**
>
> 您必須先重建機器人，才能將音訊轉錄置信度分數用於現有機器人。現有版本的機器人不支援轉錄可信度分數。您必須創建一個新版本的機器人才能使用它們。

您可以針對多個交談設計模式使用信賴度分數：

- 如果最高可信度分數因為環境嘈雜或訊號品質不佳而降至臨界值以下，您可以提示使用者提示相同的問題，以擷取品質較佳的音訊。
- 如果多個轉錄對於插槽值具有類似的可信度分數，例如「John」和「Juan」，您可以將這些值與預先存在的資料庫進行比較，以消除輸入，或者提示使用者選取兩個值之一。例如，「為約翰說 1，或說 2 代表胡安。」
- 如果您的商務邏輯需要根據替代文字本中的特定關鍵字進行意圖切換，且信賴度分數接近頂端成績單，您可以使用對話方塊程式碼掛接 Lambda 函數或使用工作階段管理作業來變更意圖。如需詳細資訊，請參閱工作階段管理。

Amazon Lex V2 會傳送下列 JSON 結構，並將使用者輸入至 Lambda 程式碼掛接函數的最多三個轉錄檔：

```
"transcriptions": [
    {
        "transcription": "string",
        "rawTranscription": "string",
        "transcriptionConfidence": "number",
        },
        "resolvedContext": {
            "intent": "string"
        },
        "resolvedSlots": {
            "string": {
                "shape": "List",
                "value": {
                    "originalValue": "string",
```

```
                    "resolvedValues": [
                        "string"
                    ]
                },
                "values": [
                    {
                        "shape": "Scalar",
                        "value": {
                            "originalValue": "string",
                            "resolvedValues": [
                                "string"
                            ]
                        }
                    },
                    {
                        "shape": "Scalar",
                        "value": {
                            "originalValue": "string",
                            "resolvedValues": [
                                "string"
                            ]
                        }
                    }
                ]
            }
        }
    }
]
```

JSON 結構包含轉錄文本，為話語解析的意圖以及語言中檢測到的任何插槽的值。對於文字使用者輸入，轉錄會包含單一成績單，信賴度分數為 1.0。

成績單的內容取決於對話的轉向和公認的意圖。

對於第一回合，意圖引出，Amazon Lex V2 決定了前三個轉錄。對於頂部轉錄，它返回轉錄中的意圖和任何推斷的插槽值。

在隨後的轉彎中，槽引出，結果取決於每個轉錄的推斷意圖，如下所示。

- 如果頂部成績單的推斷意圖與上一回合相同，並且所有其他成績單具有相同的意圖，那麼
  - 所有成績單都包含推斷的位置值。

- 如果頂級成績單的推斷意圖與上一回合不同，並且所有其他成績單都有先前的意圖，那麼
  - 頂部成績單包含新意圖的推斷插槽值。
  - 其他成績單具有先前意圖的意圖和推斷的位置值，以供先前的意圖使用。

- 如果頂部成績單的推斷意圖與前一回合不同，則一個成績單與前一個意圖相同，而一個成績單是不同的意圖，那麼
  - 頂部成績單包含新推斷的意圖和任何推斷出來的插槽值。
  - 具有先前推斷意圖的成績單包含針對該意圖推斷的位置值。
  - 具有不同意圖的成績單沒有推斷的意圖名稱，也沒有推斷的插槽值。

- 如果頂級成績單的推斷意圖與上一回合不同，並且所有其他成績單具有不同的意圖，那麼
  - 頂部成績單包含新推斷的意圖和任何推斷出來的插槽值。
  - 其他成績單不包含推斷的意圖，也沒有推斷的插槽值。

- 如果前兩個成績單的推斷意圖與前一回合相同且不同，而第三個成績單是不同的意圖，那麼
  - 前兩個成績單中包含新推斷的意圖和任何推斷出來的插槽值。
  - 第三個記錄沒有意圖名稱，也沒有解析的槽值。

工作階段管理

若要變更 Amazon Lex V2 在與使用者交談中使用的意圖，請使用對話方塊程式碼掛接 Lambda 函數的回應。或者，您可以在自訂應用程式中使用工作階段管理 API。

使用 Lambda 函數

當您使用 Lambda 函數時，Amazon Lex V2 會以包含函數輸入的 JSON 結構呼叫該函數。JSON 結構包含一個名為的欄位`transcriptions`，其中包含 Amazon Lex V2 針對話語所決定的可能轉錄。該`transcriptions`欄位包含一到三個可能的轉錄，每個都有可信度分數。

若要使用替代轉錄中的意圖，請在 Lambda 函數的`ConfirmIntent`或對`ElicitSlot`話方塊動作中指定該意圖。若要使用替代轉錄中的插槽值，請在 Lambda 函數回應的`intent`欄位中設定值。如需詳細資訊，請參閱使用AWS Lambda函數啟用自訂邏輯。

範例程式碼

下列程式碼範例是 Python Lambda 函數，它使用音訊轉錄來改善使用者的交談體驗。

若要使用範例程式碼，您必須具備：

- 具有一種語言的機器人，可以是英文 (GB) (en_GB) 或英文 (美國) (en_US)。
- 一個意圖， OrderBirthStone. 確定已在意圖定義的 [程式碼掛接] 區段中選取 [使用 Lambda 函數進行初始化和驗證]。
- 意圖應該有兩個插槽，「BirthMonth」和「名稱」兩種類型AMAZON.AlphaNumeric。
- 定義了 Lambda 函數的別名。如需詳細資訊，請參閱建立 Lambda 函數並將其附加至機器人別名。

```
import time
import os
import logging

logger = logging.getLogger()
logger.setLevel(logging.DEBUG)


# --- Helpers that build all of the responses ---

def elicit_slot(session_attributes, intent_request, slots, slot_to_elicit, message):
    return {
        'sessionState': {
            'dialogAction': {
                'type': 'ElicitSlot',
                'slotToElicit': slot_to_elicit
            },
            'intent': {
                'name': intent_request['sessionState']['intent']['name'],
                'slots': slots,
                'state': 'InProgress'
            },
            'sessionAttributes': session_attributes,
            'originatingRequestId': 'e3ab4d42-fb5f-4cc3-bb78-caaf6fc7cccd'
        },
        'sessionId': intent_request['sessionId'],
        'messages': [message],
        'requestAttributes': intent_request['requestAttributes'] if 'requestAttributes'
 in intent_request else None
```

```
    }


def close(intent_request, session_attributes, fulfillment_state, message):
    intent_request['sessionState']['intent']['state'] = fulfillment_state
    return {
        'sessionState': {
            'sessionAttributes': session_attributes,
            'dialogAction': {
                'type': 'Close'
            },
            'intent': intent_request['sessionState']['intent'],
            'originatingRequestId': '3ab4d42-fb5f-4cc3-bb78-caaf6fc7cccd'
        },
        'messages': [message],
        'sessionId': intent_request['sessionId'],
        'requestAttributes': intent_request['requestAttributes'] if 'requestAttributes'
 in intent_request else None
    }


def delegate(intent_request, session_attributes):
    return {
        'sessionState': {
            'dialogAction': {
                'type': 'Delegate'
            },
            'intent': intent_request['sessionState']['intent'],
            'sessionAttributes': session_attributes,
            'originatingRequestId': 'abc'
        },
        'sessionId': intent_request['sessionId'],
        'requestAttributes': intent_request['requestAttributes'] if 'requestAttributes'
 in intent_request else None
    }


def get_session_attributes(intent_request):
    sessionState = intent_request['sessionState']
    if 'sessionAttributes' in sessionState:
        return sessionState['sessionAttributes']

    return {}
```

```
def get_slots(intent_request):
    return intent_request['sessionState']['intent']['slots']


""" --- Functions that control the behavior of the bot --- """


def order_birth_stone(intent_request):
    """
    Performs dialog management and fulfillment for ordering a birth stone.
    Beyond fulfillment, the implementation for this intent demonstrates the following:
    1) Use of N best transcriptions to re prompt user when confidence for top
 transcript is below a threshold
    2) Overrides resolved slot for birth month from a known fixed list if the top
 transcript
    is not accurate.
    """

    transcriptions = intent_request['transcriptions']

    if intent_request['invocationSource'] == 'DialogCodeHook':
        # Disambiguate if there are multiple transcriptions and the top transcription
        # confidence is below a threshold (0.8 here)
        if len(transcriptions) > 1 and transcriptions[0]['transcriptionConfidence'] <
 0.8:
            if transcriptions[0]['resolvedSlots'] is not {} and 'Name' in
 transcriptions[0]['resolvedSlots'] and \
                    transcriptions[0]['resolvedSlots']['Name'] is not None:
                return prompt_for_name(intent_request)
            elif transcriptions[0]['resolvedSlots'] is not {} and 'BirthMonth' in
 transcriptions[0]['resolvedSlots'] and \
                    transcriptions[0]['resolvedSlots']['BirthMonth'] is not None:
                return validate_month(intent_request)

    return continue_conversation(intent_request)


def prompt_for_name(intent_request):
    """
    If the confidence for the name is not high enough, re prompt the user with the
 recognized names
    so it can be confirmed.
    """
```

```
    resolved_names = []
    for transcription in intent_request['transcriptions']:
        if transcription['resolvedSlots'] is not {} and 'Name' in
 transcription['resolvedSlots'] and \
                transcription['resolvedSlots']['Name'] is not None:
            resolved_names.append(transcription['resolvedSlots']['Name']['value']
['originalValue'])
    if len(resolved_names) > 1:
        session_attributes = get_session_attributes(intent_request)
        slots = get_slots(intent_request)
        return elicit_slot(session_attributes, intent_request, slots, 'Name',
                           {'contentType': 'PlainText',
                            'content': 'Sorry, did you say your name is {} ?'.format("
 or ".join(resolved_names))})
    else:
        return continue_conversation(intent_request)


def validate_month(intent_request):
    """
    Validate month from an expected list, if not valid looks for other transcriptions
 and to see if the month
    recognized there has an expected value. If there is, replace with that and if not
 continue conversation.
    """

    expected_months = ['january', 'february', 'march']
    resolved_months = []
    for transcription in intent_request['transcriptions']:
        if transcription['resolvedSlots'] is not {} and 'BirthMonth' in
 transcription['resolvedSlots'] and \
                transcription['resolvedSlots']['BirthMonth'] is not None:
            resolved_months.append(transcription['resolvedSlots']['BirthMonth']
['value']['originalValue'])

    for resolved_month in resolved_months:
        if resolved_month in expected_months:
            intent_request['sessionState']['intent']['slots']['BirthMonth']
['resolvedValues'] = [resolved_month]
            break

    return continue_conversation(intent_request)
```

```
def continue_conversation(event):
    session_attributes = get_session_attributes(event)

    if event["invocationSource"] == "DialogCodeHook":
        return delegate(event, session_attributes)


# --- Intents ---


def dispatch(intent_request):
    """
    Called when the user specifies an intent for this bot.
    """

    logger.debug('dispatch sessionId={},
 intentName={}'.format(intent_request['sessionId'],

 intent_request['sessionState']['intent']['name']))

    intent_name = intent_request['sessionState']['intent']['name']

    # Dispatch to your bot's intent handlers
    if intent_name == 'OrderBirthStone':
        return order_birth_stone(intent_request)

    raise Exception('Intent with name ' + intent_name + ' not supported')


# --- Main handler ---


def lambda_handler(event, context):
    """
    Route the incoming request based on intent.
    The JSON body of the request is provided in the event slot.

    """
    # By default, treat the user request as coming from the America/New_York time
 zone.
    os.environ['TZ'] = 'America/New_York'
    time.tzset()
    logger.debug('event={}'.format(event))
```

```
    return dispatch(event)
```

使用工作階段管理 API

若要使用與目前意圖不同的意圖，請使用此PutSession作業。例如，如果您決定第一個替代方法比 Amazon Lex V2 選擇的意圖更好，則可以使用此PutSession操作來變更意圖。這樣，用戶與之交互的下一個意圖將是您選擇的意圖。

您也可以使用此PutSession作業來變更結intent構中的位置值，以使用替代轉錄中的值。

如需詳細資訊，請參閱 使用亞馬遜萊克斯 V2 API 管理工作階段。

自訂語音轉錄

機器人的預設行為有時可能會導致語音轉錄不正確。以下功能可幫助您的機器人識別較不常見或容易混淆的單詞或名稱。

主題

- 使用自訂詞彙改善語音辨識
- 使用執行時期提示改善位置值的辨識
- 使用拼字樣式擷取位置值

使用自訂詞彙改善語音辨識

您可以透過以特定語言建立自訂詞彙，向 Amazon Lex V2 提供有關如何處理與機器人音訊交談的詳細資訊。自訂字彙是您希望 Amazon Lex V2 在音訊輸入中辨識的特定片語清單。這些通常是 Amazon Lex V2 無法辨識的適當名詞或特定於網域的字詞。

例如，假設您有技術支援機器人。您可以將「備份」添加到自定義詞彙中，以幫助機器人將音頻正確轉錄為「備份」，即使音頻聽起來像「收拾」。自定義詞彙還可以幫助識別音頻中罕見的單詞，例如金融服務的「償付能力」或適當的名詞，例如「Cognito」或「Monitron」。

自訂詞彙基礎

- 自定義詞彙適用於將音頻輸入到機器人的轉錄。您必須提供範例語彙，才能辨識意圖或槽值。
- 自訂字彙是特定語言所獨有的。您必須為每種語言個別設定自訂字彙。自訂詞彙僅支援英文 (英國) 和英文 (美國) 語言。

- Amazon Lex V2 支援的客服中心整合功能可使用自訂字彙。Amazon Lex V2 主控台中的測試視窗支援在 2022 年 7 月 31 日或之後建置的所有 Amazon Lex V2 機器人的自訂字彙。如果您在測試視窗中遇到自訂字彙問題，請重新建置機器人，然後再試一次。

Amazon Lex V2 使用自訂字彙來引出意圖和插槽。相同的自訂字彙檔案用於意圖和插槽。您可以在新增插槽類型時，選擇性地關閉插槽的自訂詞彙功能。

引出意圖 — 您可以創建自定義詞彙來引出意圖。當您的機器人確定用戶的意圖時，這些短語用於轉錄。例如，如果您在自訂字彙中設定了「備份」一詞，Amazon Lex V2 會將使用者輸入的內容轉錄為「可以請您備份我的相片嗎？」 -即使音頻聽起來像「你能收拾我的照片。」 您可以透過設定 0、1、2 或 3 的權重來指定每個片語的提升程度。您也可以透過新增`displayAs`欄位，為最終語音轉換文字輸出中的片語指定替代表示法。

用於在意圖引出期間改善轉錄的自定義詞彙短語不會影響引出插槽時的轉錄。如需建立自訂字彙以引出意圖的詳細資訊，請參閱。創建引出意圖和插槽的自定義詞彙

引出自定義插槽 — 您可以使用自定義詞彙來提高音頻對話的插槽識別。若要提升 Amazon Lex V2 機器人辨識插槽值的能力，請建立自訂位置並將插槽值新增至自訂插槽，然後選擇「使用位置值作為自訂詞彙」。插槽值的範例包括產品名稱、型錄或適當名詞。您不應在自訂詞彙中使用常見的字詞或片語，例如「yes」和「no」。

新增插槽值之後，當機器人期待自訂插槽的輸入時，這些值會用於改善位置辨識度。引出意圖時，這些值不用於轉錄。如需詳細資訊，請參閱新增插槽類型。

建立自訂字彙的最佳作法

引出意圖

- 自訂詞彙最適合用來將特定字組或詞組設成目標。只有在 Amazon Lex V2 無法輕易辨識的情況下，才能在自訂字彙中新增單字。
- 根據字詞在轉錄中無法辨識的頻率，以及單字在輸入中的罕見程度，來決定要提供字詞的權重。難以發音的單詞需要更高的權重。
- 使用代表性的測試集來確定權重是否合適。您可以開啟交談記錄中的音訊記錄來收集音訊測試集。
- 避免在自定義詞彙中使用「on」，「it」，「to」，「是」，「否」之類的短詞。

引出一個自訂欄位

- 將值新增至您預期可辨識的自訂槽類型。為自訂插槽類型新增所有可能的插槽值，無論插槽值有多普遍或稀有。
- 僅當自訂槽類型包含目錄值或實體 (例如產品名稱或共同基金) 清單時，才啟用此選項。
- 如果插槽類型用於擷取一般詞組，例如「是」、「否」、「我不知道」、「也許」或「一」、「二」、「三」等一般字詞，請停用此選項。
- 將插槽值和同義詞的數量限制為 500 或更少，以獲得最佳效能。

輸入首字母縮寫或其他單字，其字母應該以單一字母的形式發音，並以句點和空格分隔。除非它們是短語的一部分，否則不要使用個別字母，例如「J·P· 摩根」或「A.W. S.」 您可以使用大寫或小寫字母來定義縮寫。

創建引出意圖和插槽的自定義詞彙

您可以使用 Amazon Lex V2 主控台建立和管理自訂詞彙，也可以使用 Amazon Lex V2 API 操作。有兩種方法可以通過控制台創建自定義詞彙：

主控台

在控制台中導入自定義詞彙：

1. 打開亞馬遜萊克斯 V2 控制台 https://console.aws.amazon.com/lexv2/home
2. 從機器人列表中，選擇要添加自定義詞彙的機器人。
3. 在機器人詳細資料頁面的 [新增語言] 區段中，選擇 [檢視語言]。
4. 從語言清單中，選擇您要新增自訂字彙的語言。

直接通過控制台創建新的自定義詞彙：

1. 在語言詳細資訊頁面的「自訂詞彙」區段中，按一下「建立」。這將打開一個沒有自定義詞彙的編輯窗口。
2. 視需要新增詞組DisplayAs、和重量的輸入。您可以透過更新項目的欄位或從清單中刪除項目，進一步對新增的項目進行內嵌編輯。
3. 點擊保存。請注意：新的自定義詞彙僅在您單擊「保存」後保存在您的機器人中。
4. 您可以繼續在此頁面中進行內嵌編輯，完成後按一下「儲存」。
5. 此頁面還允許您從右上角的下拉菜單中導入，導出和刪除自定義詞彙文件。

API

使用 **ListCustomVocabularyItems** API 來檢視自訂字彙項目：

1. 使用此ListCustomVocabularyItems作業可檢視自訂字彙項目。請求主體將如下所示：

```
{
   "maxResults": number,
   "nextToken": "string"
}
```

2. 請注意，maxResults和nextToken是請求主體的可選字段。
3. 來自ListCustomVocabularyItems操作的響應如下所示：

```
{
   "botId": "string",
   "botVersion": "string",
   "localeId": "string",
   "customVocabularyItems": [
        {
            "itemId": "string",
            "phrase": "string",
            "weight": number,
            "displayAs": "string"
        }
   ]
}
```

使用 **BatchCreateCustomVocabularyItem** API 建立新的自訂字彙項目：

1. 如果您的機器人地區設定尚未建立自訂字彙，請依照步驟使StartImport用建立自訂字彙。
2. 建立自訂字彙之後，使用此BatchCreateCustomVocabularyItem作業建立新的自訂字彙項目。請求主體將如下所示：

```
{
   "customVocabularyItemList": [
        {
            "phrase": "string",
            "weight": number,
            "displayAs": "string"
        }
```

```
    ]
}
```

3. 請注意，weight和displayAs是請求主體的可選字段。

4. 來自BatchCreateCustomVocabularyItem意志的響應如下所示：

```
{
    "botId": "string",
    "botVersion": "string",
    "localeId": "string",
    "errors": [
        {
            "itemId": "string",
            "errorMessage": "string",
            "errorCode": "string"
        }
    ],
    "resources": [
        {
            "itemId": "string",
            "phrase": "string",
            "weight": number,
            "displayAs": "string"
        }
    ]
}
```

5. 由於這是一個批處理操作，如果其中一個項目無法創建請求將不會失敗。錯誤清單會包含該特定項目作業失敗原因的相關資訊。資源清單將包含已成功建立的所有項目。

6. 對於BatchCreateCustomVocabularyItem，您可以期望看到以下類型的錯誤：

   - RESOURCE_DOES_NOT_EXIST：自定義詞彙不存在。在呼叫此作業之前，請遵循建立自訂字彙的步驟。
   - DUPLICATE_INPUT：輸入列表包含重複的短語。
   - RESOURCE_ALREADY_EXISTS：該條目的給定短語已存在於您的自定義詞彙中。
   - INTERNAL_SERVER_FAILURE：處理您的請求時後端出現錯誤。這可能表示服務中斷或其他問題。

使用 **BatchDeleteCustomVocabularyItem** API 刪除現有的自訂字彙項目：

1. 如果您的機器人的語言環境尚未建立自訂字彙，請依照使用建立自訂字彙StartImport來建立字彙的步驟。
2. 建立自訂字彙之後，使用此BatchDeleteCustomVocabularyItem作業刪除現有的自訂字彙項目。請求主體將如下所示：

```
{
   "customVocabularyItemList": [
        {
            "itemId": "string"
        }
   ]
}
```

3. 來自BatchDeleteCustomVocabularyItem意志的響應如下所示：

```
{
    "botId": "string",
    "botVersion": "string",
    "localeId": "string",
    "errors": [
        {
            "itemId": "string",
            "errorMessage": "string",
            "errorCode": "string"
        }
    ],
    "resources": [
        {
            "itemId": "string",
            "phrase": "string",
            "weight": number,
            "displayAs": "string"
        }
    ]
}
```

4. 由於這是一個批處理操作，如果其中一個項目無法刪除，請求將不會失敗。錯誤清單會包含該特定項目作業失敗原因的相關資訊。資源清單將包含已成功刪除的所有項目。
5. 對於BatchDeleteCustomVocabularyItem，您可以期望看到以下類型的錯誤：

- RESOURCE_DOES_NOT_EXIST：您嘗試刪除的自定義詞彙條目不存在。
- INTERNAL_SERVER_FAILURE：處理您的請求時後端出現錯誤。這可能表示服務中斷或其他問題。

使用 **BatchUpdateCustomVocabularyItem** API 更新現有的自訂字彙項目：

1. 如果您的機器人的語言環境尚未建立自訂字彙，請依照使用建立自訂字彙StartImport來建立自訂字彙的步驟進行操作。
2. 建立自訂字彙之後，使用此BatchUpdateCustomVocabularyItem作業更新現有的自訂字彙項目。請求主體將如下所示：

```
{
   "customVocabularyItemList": [
        {
            "itemId": "string",
            "phrase": "string",
            "weight": number,
            "displayAs": "string"
        }
   ]
}
```

3. 請注意，weight和displayAs是請求主體的可選字段。
4. 來自BatchUpdateCustomVocabularyItem意志的響應如下所示：

```
{
    "botId": "string",
    "botVersion": "string",
    "localeId": "string",
    "errors": [
        {
            "itemId": "string",
            "errorMessage": "string",
            "errorCode": "string"
        }
    ],
    "resources": [
        {
            "itemId": "string",
            "phrase": "string",
```

```
            "weight": number,
            "displayAs": "string"
        }
    ]
}
```

5. 由於這是一個批處理操作，如果其中一個項目無法刪除，請求將不會失敗。錯誤清單會包含該特定項目作業失敗原因的相關資訊。資源清單將包含所有已成功更新的項目。
6. 對於BatchUpdateCustomVocabularyItem，您可以期望看到以下類型的錯誤：

   - RESOURCE_DOES_NOT_EXIST：您嘗試更新的自定義詞彙條目不存在。
   - DUPLICATE_INPUT：輸入清單包含重複的項目 MID。
   - RESOURCE_ALREADY_EXISTS：該條目的給定短語已存在於您的自定義詞彙中。
   - INTERNAL_SERVER_FAILURE：處理您的請求時後端出現錯誤。這可能表示服務中斷或其他問題。

建立自訂字彙檔

自訂字彙檔案是以 Tab 字元分隔的值清單，其中包含要辨識的片語、提供增強功能的權重，以及用來取代語音記錄中片語的displayAs欄位。具有較高提升值的短語出現在音頻輸入中時，更有可能使用它們。

自訂字彙檔案必須命名**CustomVocabulary.tsv**，且必須先以 zip 檔案壓縮，才能匯入該檔案。壓縮檔案的大小必須小於 300 MB。自定義詞彙中的短語數量上限為 500。

- 短語 1—4 個應該被識別的單詞。用空格分隔片語中的單詞。檔案中不能有重複的片語。片語欄位為必填欄位。
- weight — 提升片語辨識度的程度。該值是一個整數 0、1、2 或 3。如果未指定線寬，預設值為 1。根據轉錄中無法識別單詞的頻率以及單詞在輸入中的罕見程度來決定權重。權重 0 表示不會套用任何提升，且該項目將僅用於使用displayAs欄位執行替換。
- DisplayAS — 定義您希望片語在轉錄輸出中的外觀方式。這是自訂字彙中的可選欄位。

自訂字彙檔案必須包含標題列，其中標題為「片語」、「重量」和「DisplayAs」。標題可以是任何順序，但必須遵循上述命名法。

下列範例為自訂字彙檔案。用於分隔片語，權重和顯示所需的製表符由文本「[TAB]」表示。如果您使用此範例，請以定位字元取代文字。

```
phrase[TAB]weight[TAB]displayAs
Newcastle[TAB]2
Hobart[TAB]2[TAB]Hobart, Australia
U. Dub[TAB]1[TAB]University of Washington, Seattle
W. S. U.[TAB]3
Issaquah
Kennewick
```

使用執行時期提示改善位置值的辨識

透過執行階段提示，您可以根據情境為 Amazon Lex V2 提供一組插槽值，以便在音訊交談中獲得更好的辨識度，並改善插槽解析度。您可以使用執行階段提示來提供在執行階段成為解析位置值的候選片語清單。

例如，如果使用者與航班預訂機器人互動經常前往舊金山、雅加達、首爾和莫斯科，您可以在引出目的地時，使用這四個城市的清單來設定執行時提示，以提升經常旅行城市的辨識度。

執行階段提示僅提供英文 (美國) 和英文 (英國) 語言。它們可以與以下插槽類型一起使用：

- 自訂插槽類型
- 亞馬遜城
- 亞馬遜國家
- 亞馬遜。 FirstName
- 亞馬遜。 LastName
- 亚马逊州
- 亞馬遜。 StreetName

運行時提示基礎

- 只有在從使用者引出插槽值時，才會使用執行階段提示。
- 當您使用執行階段提示時，提示的值會優先於類似的值。例如，對於食品訂購機器人，您可以將菜單項列表設置為運行時提示，同時引出自定義插槽中的食物項目，以使其更喜歡「魚片」而不是聽起來相似的「伙計」。
- 如果使用者輸入與執行時期提示中提供的值不同，原始使用者輸入將用於插槽。
- 對於自訂插槽類型，作為執行階段提示提供的值將用於解析插槽，即使它們在建立機器人時不屬於自訂位置的一部分。

- 執行階段提示僅支援 8 kHz 音訊輸入。它們可透過 Amazon Lex V2 支援的客服中心整合功能來使用。Amazon Lex V2 主控台上測試視窗的音訊輸入不會提供執行階段提示，因為它使用 16 kHz 音訊輸入。

> **Note**
>
> 您必須先重建機器人，才能將執行階段提示與現有機器人搭配使用。現有版本的機器人不支援執行階段提示。您必須創建一個新版本的機器人才能使用它們。

您可以使用PutSession、RecognizeText或StartConversation作業，將執行階段提示傳送至 Amazon Lex V2。RecognizeUtterance您也可以使用 Lambda 函數新增執行階段提示。

您可以在交談開始時傳送執行階段提示，以針對機器人中使用的每個插槽設定提示，也可以在交談期間傳送提示做為工作階段狀態的一部分。`runtimeHints`屬性會將插槽對應至該欄位的提示。

將執行階段提示傳送至 Amazon Lex V2 之後，它們會在交談的每一回合保留，直到工作階段結束為止。如果您傳送 null `runtimeHints` 結構，則會使用現有的提示。您可以通過以下方式修改提示：

- 向機器人發送新`runtimeHints`結構。新結構的內容會取代現有結構。
- 向機器人發送一個空的`runtimeHints`結構。這會清除機器人的執行階段提示。

在上下文中添加插槽值

當您的應用程式具有使用者下一個可能話語的相關資訊時，提供預期的位置值作為執行階段提示，為您的機器人新增上下文。將 Lambda 對話方塊程式碼掛接新增至您的機器人 (如使用AWS Lambda函數啟用自訂邏輯需詳細資訊，請參閱)，並使用中的proposedNextState欄位解譯輸入事件格式來決定您應包含哪些執行階段提示，以改善與使用者之間的對話。

例如，在銀行應用程序中，您可以為特定用戶生成帳戶暱稱列表，然後在引出用戶要訪問的帳戶時使用該列表。

當您有協助機器人解譯使用者輸入的內容時，會在交談開始時傳送執行階段提示。例如，如果您有使用者的電話號碼，您可以使用此資訊來查詢使用者，以便在您引導使用者名稱來驗證其認證時，可以使用`PutSession`或`StartConversation`作業將姓名提示傳送給機器人。

在交談期間，您可能會從一個插槽值收集資訊，以協助處理另一個插槽值。例如，在汽車護理應用程序中，當您擁有用戶的帳戶號碼時，您可以進行查找以查找客戶擁有的汽車，並將其作為提示傳遞給另一個插槽。

輸入首字母縮略詞，或其他字母應該單獨發音的單字，並以句點和空格分隔。除非它們是短語的一部分，否則請勿使用個別字母，例如「J.P. 摩根」或「A.W.S」。您可以使用大寫或小寫字母來定義縮寫。

將提示新增至插槽

若要將執行階段提示新增至插槽，您可以使用屬於runtimeHints結構一部分的sessionState結構。以下是結runtimeHints構的範例。它提供了兩個插槽的提示，FirstNameLastName"" 和 "MakeAppointment" 意圖。

```
{
    "sessionState": {
        "intent": {},
        "activeContexts": [],
        "dialogAction": {},
        "originatingRequestId": {},
        "sessionAttributes": {},
        "runtimeHints": {
            "slotHints": {
                "MakeAppointment": {
                    "FirstName": {
                        "runtimeHintValues": [
                            {
                                "phrase": "John"
                            },
                            {
                                "phrase": "Mary"
                            }
                        ]
                    },
                    "LastName": {
                        "runtimeHintValues": [
                            {
                                "phrase": "Stiles"
                            },
                            {
                                "phrase": "Major"
                            }
                        ]
                    }
                }
            }
        }
```

```
    }
}
```

您也可以使用 Lambda 函數在交談期間新增執行時期提示。若要新增執行時期提示，請將`runtimeHints`結構新增至 Lambda 函數傳送至 Amazon Lex V2 之回應的工作階段狀態。如需詳細資訊，請參閱準備回應格式。

您必須在請求`slotName`中指定有效`intentName`的，否則 Amazon Lex V2 會傳回執行階段錯誤。

使用拼字樣式擷取位置值

Amazon Lex V2 提供內建插槽，可擷取使用者特定資訊，例如名字、姓氏、電子郵件地址或英數字元識別碼。例如，您可以使用`AMAZON.LastName`插槽擷取姓氏，例如「傑克遜」或「加西亞」。不過，Amazon Lex V2 可能會與難以發音或在地區設定中不常見的姓氏混淆，例如「秀蘭」。要捕獲此類名稱，您可以要求用戶通過字母或按單詞樣式拼寫提供拼寫輸入。

Amazon Lex V2 提供三種插槽引導樣式供您使用。當您設定插槽引出樣式時，它會改變 Amazon Lex V2 解譯使用者輸入的方式。

按字母拼寫 — 使用這種風格，您可以指示機器人聆聽拼寫而不是整個短語。例如，要捕獲「秀蘭」之類的姓氏，您可以告訴用戶一次拼出他們的姓氏一個字母。機器人將捕獲拼寫並將字母解析為單詞。例如，如果使用者說「x i u l a n」，則機器人會將姓氏擷取為「xiulan」。

按字拼寫 — 在語音對話中，尤其是使用電話，有幾個字母，例如「t」，「b」，「p」，這聽起來很相似。擷取英數字元值或拼字名稱時，會產生不正確的值，您可以提示使用者提供識別字詞以及字母。例如，如果對預訂 ID 的請求的語音回應是「abp123」，則您的機器人可能會改為識別「ab b 123」這個短語。如果這是一個不正確的值，你可以要求用戶提供輸入作為「a 在 alpha b 中，如在男孩 p 中，如彼得一二三」。機器人會將輸入解析為「abp123」。

使用逐字拼字時，您可以使用下列格式：

- 「如在」（一如蘋果）
- 「為」（蘋果的一個）
- 「喜歡」（像蘋果一樣）

默認-這是使用單詞發音進行插槽捕獲的自然風格。例如，它可以自然地捕獲諸如「約翰·斯泰爾斯」之類的名稱。如果未指定插槽引出樣式，則機器人會使用默認樣式。對於`AMAZON.AlphaNumeric`和`AMAZON.UKPostal`代碼插槽類型，默認樣式支持通過字母輸入拼寫。

如果使用字母和單詞混合說出名稱「秀蘭」，例如「x 如 x 射線 i u l 中的獅子 a n」，則必須將插槽引出樣式設置為樣式。 spell-by-word spell-by-letter 風格不會識別它。

您應該創建一個語音界面，以自然的對話風格捕獲插槽值，以獲得更好的體驗。對於未使用自然樣式正確擷取的輸入，您可以重新提示使用者，並將插槽引出樣式設定為或。 spell-by-letter spell-by-word

您可以使用 spell-by-word 英文 (美國)、英文 (英國) 和英文 (澳洲) 語言的下列插槽類型和 spell-by-letter 樣式：

- 亞馬遜。 AlphaNumeric
- 亞馬遜。 EmailAddress
- 亞馬遜。 FirstName
- 亞馬遜。 LastName
- 亞馬遜 PostalCode
- 自訂插槽類型

啟用拼字

當您從使 spell-by-letter 用者引出插槽 spell-by-word 時，可以在執行階段啟用和。您可以使用PutSession、、RecognizeText或StartConversation操作來設定拼字樣式。RecognizeUtterance您也可以啟 spell-by-word 用 spell-by-letter 和使用 Lambda 函數。

您可以在上述其中一個 API 作業的`dialogAction`要求中，或在設定 Lambda 回應時，使用`sessionState`欄位的欄位來設定拼字樣式 (如準備回應格式需詳細資訊，請參閱)。只有當對話方塊動作類型為，`ElicitSlot`且要引出的槽為其中一種支援的槽類型時，您才能設定樣式。

下列 JSON 程式碼顯示設定為使用此 spell-by-word 樣式的`dialogAction`欄位：

```
"dialogAction": {
    "slotElicitationStyle": "SpellByWord",
    "slotToElicit": "BookingId",
    "type": "ElicitSlot"
}
```

`slotElicitationStyle` 欄位可以設定為 `SpellByLetter`、`SpellByWord` 或 `Default`。如果您未指定值，則值會設定為`Default`。

> **Note**
>
> 您無法透過主控台啟用 spell-by-letter 或 spell-by-word 引出樣式。

範例程式碼

如果第一次嘗試解析不起作用的插槽值，通常會執行更改拼寫樣式。下列程式碼範例是 Python Lambda 函數，它會在第二次嘗試解析插槽時使用該 spell-by-word 樣式。

若要使用範例程式碼，您必須具備：

- 具有一種語言的機器人，英語（GB）（en_GB）。
- 一個意圖，「CheckAccount」與一個樣本語言，「我想檢查我的帳戶」。確定已在意圖定義的 [程式碼掛接] 區段中選取 [使用 Lambda 函數進行初始化和驗證]。
- 意圖應該有一個`AMAZON.UKPostalCode`內置類型的插槽 PostalCode「」。
- 定義了 Lambda 函數的別名。如需詳細資訊，請參閱 建立 Lambda 函數並將其附加至機器人別名。

```
import json
import time
import os
import logging

logger = logging.getLogger()
logger.setLevel(logging.DEBUG)

# --- Helpers that build all of the responses ---

def get_slots(intent_request):
    return intent_request['sessionState']['intent']['slots']

def get_session_attributes(intent_request):
    sessionState = intent_request['sessionState']
    if 'sessionAttributes' in sessionState:
        return sessionState['sessionAttributes']
    return {}

def get_slot(intent_request, slotName):
    slots = get_slots(intent_request)
    if slots is not None and slotName in slots and slots[slotName] is not None:
```

```
        logger.debug('resolvedValue={}'.format(slots[slotName]['value']
['resolvedValues']))
        return slots[slotName]['value']['resolvedValues']
    else:
        return None

def elicit_slot(session_attributes, intent_request, slots, slot_to_elicit,
 slot_elicitation_style, message):
    return {'sessionState': {'dialogAction': {'type': 'ElicitSlot',
                                              'slotToElicit': slot_to_elicit,
                                              'slotElicitationStyle':
 slot_elicitation_style
                                              },
                             'intent': {'name': intent_request['sessionState']
['intent']['name'],
                                        'slots': slots,
                                        'state': 'InProgress'
                                        },
                             'sessionAttributes': session_attributes,
                             'originatingRequestId': 'REQUESTID'
                             },
            'sessionId': intent_request['sessionId'],
            'messages': [ message ],
            'requestAttributes': intent_request['requestAttributes']
            if 'requestAttributes' in intent_request else None
            }

def build_validation_result(isvalid, violated_slot, slot_elicitation_style,
 message_content):
    return {'isValid': isvalid,
            'violatedSlot': violated_slot,
            'slotElicitationStyle': slot_elicitation_style,
            'message': {'contentType': 'PlainText',
            'content': message_content}
            }

def GetItemInDatabase(postal_code):
    """
    Perform database check for transcribed postal code. This is a no-op
    check that shows that postal_code can't be found in the database.
    """
    return None

def validate_postal_code(intent_request):
```

```
    postal_code = get_slot(intent_request, 'PostalCode')

    if GetItemInDatabase(postal_code) is None:
        return build_validation_result(
            False,
            'PostalCode',
            'SpellByWord',
            "Sorry, I can't find your information. " +
            "To try again, spell out your postal " +
            "code using words, like a as in apple."
        )
    return {'isValid': True}

def check_account(intent_request):
    """
    Performs dialog management and fulfillment for checking an account
    with a postal code. Besides fulfillment, the implementation for this
    intent demonstrates the following:
    1) Use of elicitSlot in slot validation and re-prompting.
    2) Use of sessionAttributes to pass information that can be used to
        guide a conversation.
    """
    slots = get_slots(intent_request)
    postal_code = get_slot(intent_request, 'PostalCode')
    session_attributes = get_session_attributes(intent_request)

    if intent_request['invocationSource'] == 'DialogCodeHook':
        # Validate the PostalCode slot. If any aren't valid,
        # re-elicit for the value.
        validation_result = validate_postal_code(intent_request)
        if not validation_result['isValid']:
            slots[validation_result['violatedSlot']] = None
            return elicit_slot(
                session_attributes,
                intent_request,
                slots,
                validation_result['violatedSlot'],
                validation_result['slotElicitationStyle'],
                validation_result['message']
            )

        return close(
            intent_request,
```

```
            session_attributes,
            'Fulfilled',
            {'contentType': 'PlainText',
             'content': 'Thanks'
             }
        )

def close(intent_request, session_attributes, fulfillment_state, message):
    intent_request['sessionState']['intent']['state'] = fulfillment_state
    return {
        'sessionState': {
            'sessionAttributes': session_attributes,
            'dialogAction': {
                'type': 'Close'
            },
            'intent': intent_request['sessionState']['intent'],
            'originatingRequestId': 'xxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx'
        },
        'messages': [ message ],
        'sessionId': intent_request['sessionId'],
        'requestAttributes': intent_request['requestAttributes'] if 'requestAttributes'
 in intent_request else None
    }

# --- Intents ---

def dispatch(intent_request):
    """
    Called when the user specifies an intent for this bot.
    """
    intent_name = intent_request['sessionState']['intent']['name']
    response = None

    # Dispatch to your bot's intent handlers
    if intent_name == 'CheckAccount':
        response = check_account(intent_request)

    return response

# --- Main handler ---

def lambda_handler(event, context):
    """
    Route the incoming request based on the intent.
```

```
    The JSON body of the request is provided in the event slot.
    """

    # By default, treat the user request as coming from
    # Eastern Standard Time.
    os.environ['TZ'] = 'America/New_York'
    time.tzset()

    logger.debug('event={}'.format(json.dumps(event)))
    response = dispatch(event)
    logger.debug("response={}".format(json.dumps(response)))

    return response
```

監控機器人效能

監控對於維持 Amazon Lex V2 聊天機器人的可靠性、可用性和效能非常重要。本主題說明如何使用交談日誌來監控使用者和聊天機器人之間的交談，並使用話語統計資料判斷機器人偵測和遺漏的話語，以及如何使用 Amazon CloudWatch Logs 和 AWS CloudTrail 監控 Amazon Lex V2。同時也說明 Amazon Lex V2 執行階段和通道關聯指標。

使用這些工具和指標來了解您可以採取哪些方向和行動來提高機器人的性能。

主題

使用分析衡量業務績效

使用 Analytics，您可以使用與機器人與客戶互動的成功和失敗率相關的指標來評估機器人的效能。您也可以視覺化機器人與客戶之間的對話流程模式。Analytics 將這些指標匯總為圖形和圖表，從而簡化您的體驗。Analytics 提供的工具可協助您篩選結果，以識別涉及意圖、時段、話語和對話的問題和問題。您可以使用這些數據迭代和改進您的機器人，以創建更好的客戶體驗。

> **Note**
>
> 若要讓使用者存取 Analytics，必須將包含分析 API 權限的AWS 受管政策：AmazonLexFullAccess或自訂政策附加至其 IAM 角色。如管理分析的存取權限需如何使用自訂原則處理使用者權限的詳細資訊，請參閱。如果連接至客戶的 IAM 角色，則會出現錯誤訊息，顯示您需要新增至使用者 IAM 角色的遺失許可，以便他們能夠存取 Analytics 儀表板。AWS 受管政策： AmazonLexReadOnly

若要存取分析

1. 登入 AWS Management Console 並開啟 Amazon Lex V2 主控台，網址為 https://console.aws.amazon.com/lexv2/home。

2. 在「機器人」下的導覽窗格中，選取您要在分析中檢視的機器人。
3. 在「分析」下方選取您要檢視的區段。

主題

關鍵定義

本主題提供可協助您解譯機器人分析的關鍵定義。這些定義與您的機器人在四種情況下的效能有關：意圖、角子機、對話和話語。下列欄位與許多效能測量結果相關：

- `Intent`物件的`state`欄位。
- `dialogAction`物件內物件的`type`欄位。SessionState

意圖

Amazon Lex V2 以下列方式對意圖進行分類：

- 成功 — 機器人成功實現了意圖。下列其中一種情況為真：
  - 意圖`state`是`ReadyForFulfillment`和`type`的`dialogAction`是`Close`。
  - 意圖`state`是`Fulfilled`和`type`的`dialogAction`是`Close`。
- 失敗 — 機器人無法達成意圖。意圖狀態。下列其中一種情況為真：
  - 意圖`state`是`Failed`和`type`的 `dialogAction` `Close` (例如，使用者拒絕確認提示)。
  - 在意圖完成之`AMAZON.FallbackIntent`前，機器人會切換至。
- 切換 — 在原始意圖被歸類為成功或失敗之前，機器人會識別不同的意圖並切換到該意圖。
- 掉線 — 客戶在意圖被歸類為成功或失敗之前沒有回應。

槽

亞馬遜 Lex V2 通過以下方式對插槽進行分類：

- 成功 — 機器人填滿了插槽，並成功轉換到另一個插槽或確認步驟。
- 失敗 — 即使達到最大重試次數，機器人也無法填滿插槽。
- 掉線 — 客戶在插槽被歸類為成功或失敗之前，不會回應或切換到其他意圖。

对话

當客戶對 Amazon Lex V2 進行執行階段呼叫時，他們會提供一個`sessionId`和 Amazon Lex V2 產生的 `originatingRequestId`. 如果客戶未在您為機器人設定的工作階段逾時 (`idleSessionTTLInSeconds`) 內回應，工作階段就會過期。如果客戶使用相同的方式返回工作階段`sessionId`，Amazon Lex V2 會產生一個新的工作階段`originatingRequestId`。

對於分析，對話是 a `sessionId` 和的唯一組合`originatingRequestId`。Amazon Lex V2 會以下列方式將對話分類：

- 成功 — 對話中的最終意圖被歸類為成功。
- 失敗 — 交談中的最終意圖失敗。如果 Amazon Lex V2 預設為AMAZON.FallbackIntent.
- 已捨棄 — 客戶在對話被歸類為成功或失敗之前不會回應。

表達用語

Amazon Lex V2 會以下列方式對話語進行分類：

- 偵測到 — Amazon Lex V2 將語音識別為嘗試叫用針對機器人設定的意圖。
- 錯過了-Amazon Lex V2 無法識別話語。

篩選結果

在每個頁面的頂端，您可以篩選機器人分析的結果。
用於分析的篩選選項。

您可以依下列參數進行篩選：

- 時間 — 您可以依相對或絕對時間範圍篩選結果。當您選取開始和結束時間時，Amazon Lex V2 會擷取在開始時間之後開始並在結束時間之前結束的對話。

- 相對範圍 — 選取 1d 可查看過去一天的結果，過去一週為 1w，或選取 1m 查看過去一個月的結果。

  如需更多選項，請選取「自訂」，並在「相對範圍」選單中選擇持續時間。若要進一步控制持續時間，請選取「自訂範圍」，在「持續時間」欄位中輸入數字，然後從下拉式選單中選擇「時間單位」。
- 絕對範圍 — 選取 [自訂]，然後選擇 [絕對範圍] 選單，以篩選指定時間範圍內的交談。您可以在日曆上選擇開始和結束日期，也可以使用 YYYY/MM/DD 格式輸入。

> **Note**
>
> 您可以看到結果的時間長度上限為 30 天。

- 機器人篩選器 — 若要依地區設定、別名和機器人版本進行篩選，請選取標示為 [所有語言環境]、[所有別名] 和 [所有版本] 的下拉式功能表。
- 模式 — 選取齒輪圖示，然後選擇「模式」下拉式功能表，以選擇是否要顯示「語音」或「文字」的結果。
- 頻道 — 選取齒輪圖示，然後選擇 [頻道] 下拉式功能表，選擇要顯示結果的頻道。如需通道整合的詳細資訊，請參閱將 Amazon Lex V2 機器人與簡訊平台整合和 Amazon Connect 聯絡中心

概觀：您的機器人效能摘要

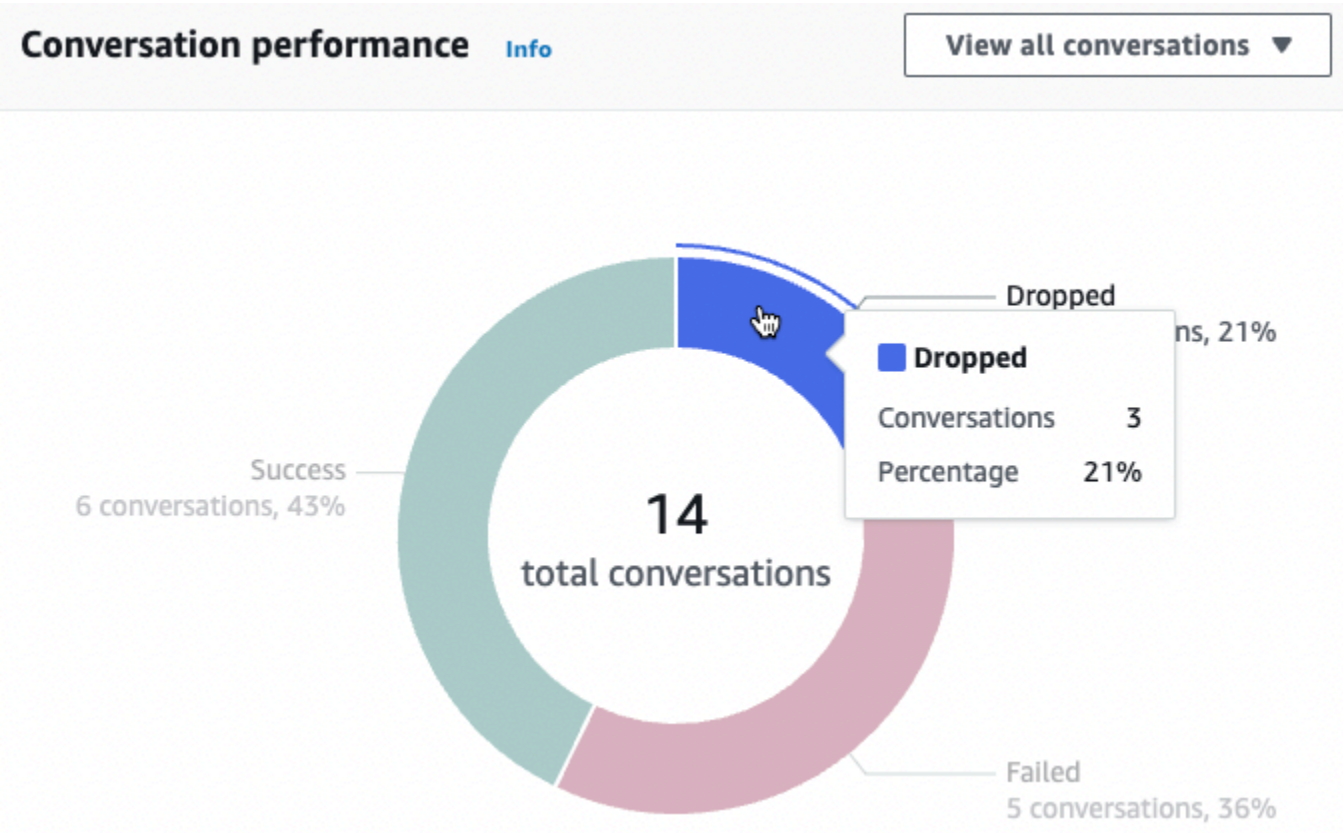
概觀頁面總結了您的機器人在對話、說話識別和意圖使用方面的表現。概觀包含以下部分：

- 對話效能
- 语音识别率
- 對話效能記錄
- 排名前 5 位的使用意圖
- 前 5 名失敗的意圖

對話效能

使用此圖表可追蹤分類為成功、失敗和捨棄之交談的數目和百分比。若要存取交談清單，請選取 [檢視所有交談] 以顯示下拉式功能表。您可以選擇檢視與機器人之間的所有使用者對話清單，或篩選具有特定結果 (成功、失敗或捨棄) 的對話。這些連結會帶您前往 [交談] 儀表板的 [交談] 子區段。如需詳細資訊，請參閱 对话。

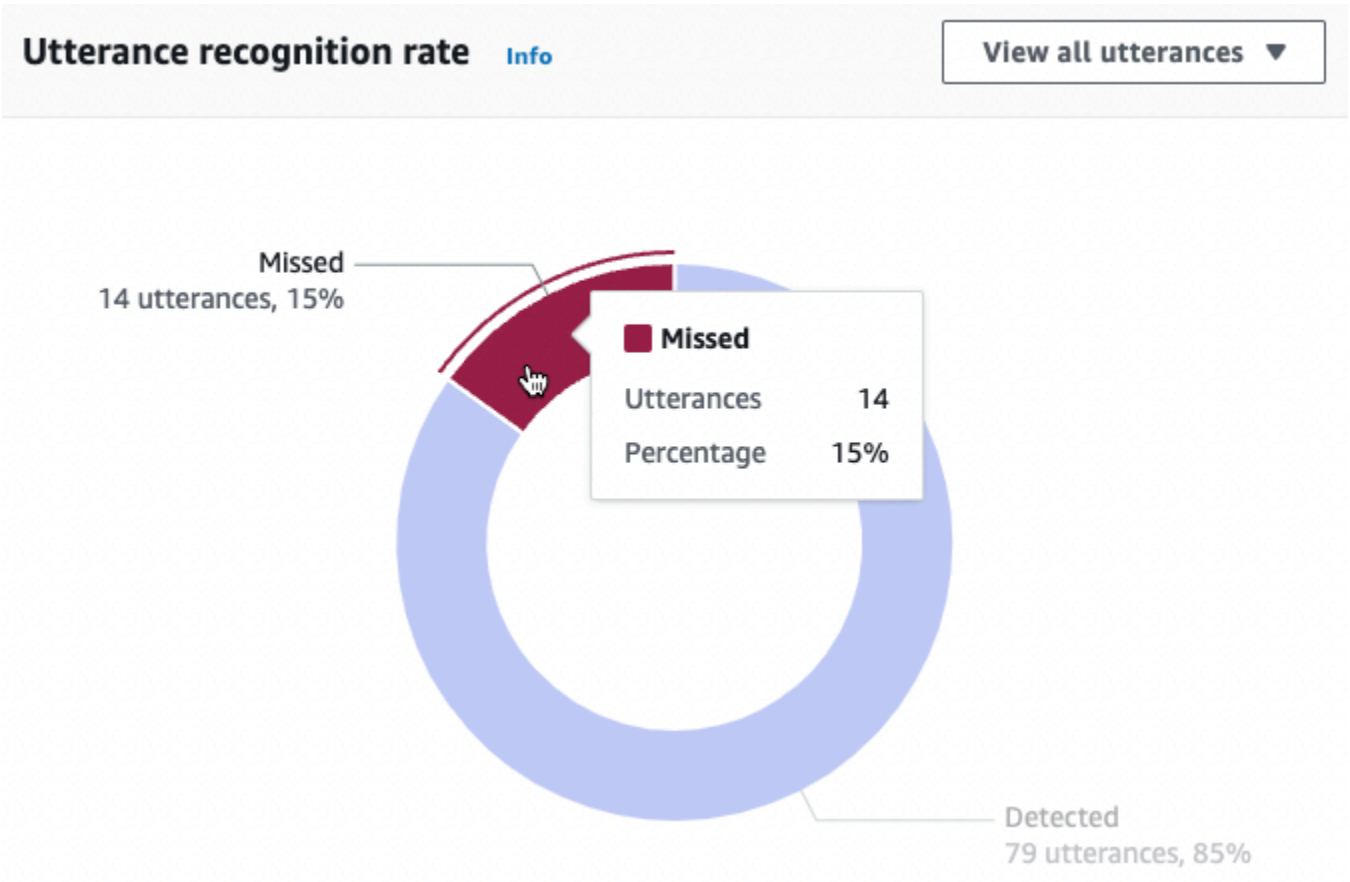
若要顯示含有該結果之交談計數和百分比的方塊，請將游標暫留在圖表的某個區段上，如下圖所示。



语音识别率

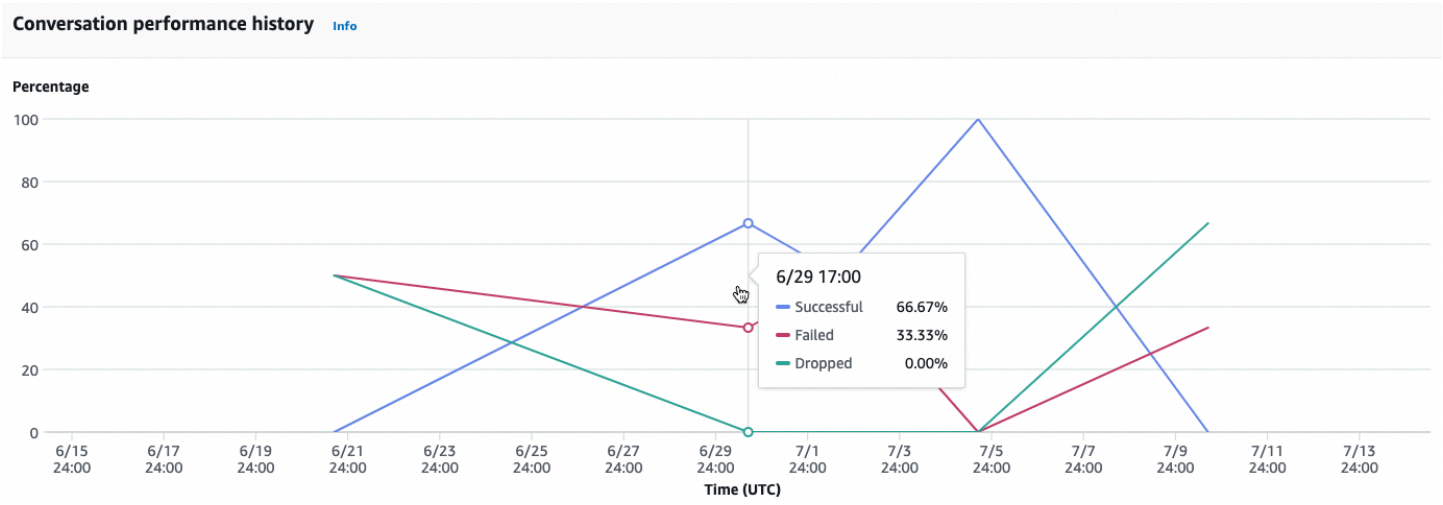
使用此圖表來追蹤您的機器人偵測到和遺漏的話語數和百分比。若要存取話語清單，請選取 [檢視話語] 以顯示下拉式選單。您可以選擇檢視所有使用者話語的清單，或篩選具有特定結果 (遺漏或偵測到) 的話語。這些連結會帶您前往「效能」儀表板的「語音辨識」子區段。如需詳細資訊，請參閱檢視要導覽至的語言。语言识别

若要顯示含有語音計數和百分比的方塊，請將游標暫留在圖表的某個區段上，如下圖所示。



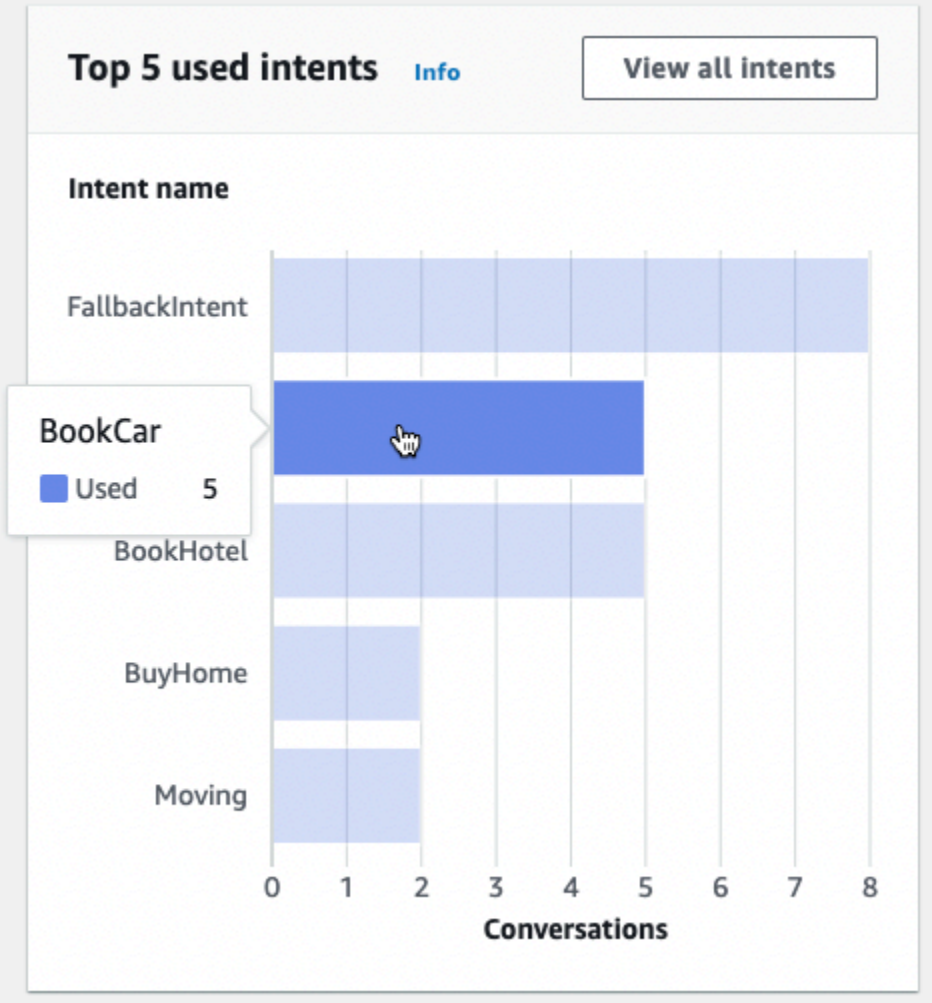
對話效能記錄

使用此圖表可追蹤在篩選器中設定的時間範圍內，分類為成功、失敗和捨棄的交談百分比。若要在時間間隔內查看具有特定結果的交談百分比，請將游標暫留在該間隔上，如下圖所示。



排名前 5 位的使用意圖

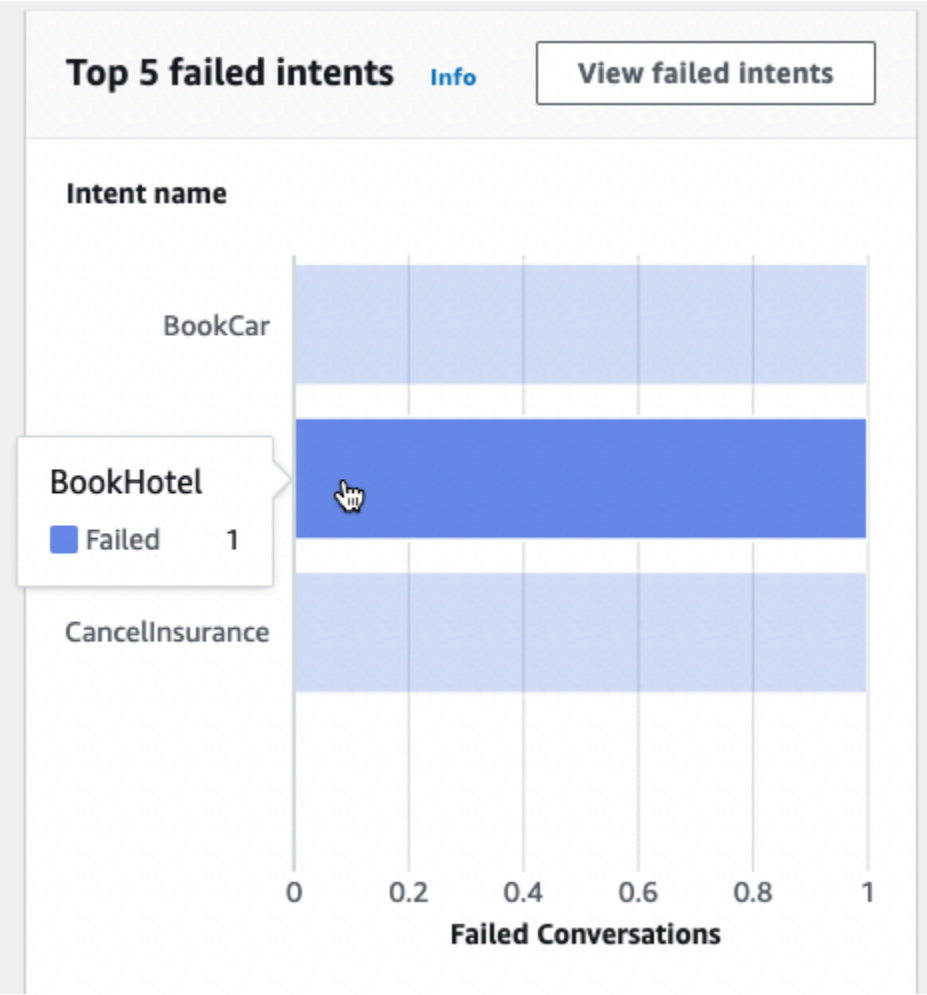
使用此圖表可識別客戶搭配機器人使用的前五個意圖。將游標暫留在列上，以查看您的機器人辨識出該意圖的次數，如下圖所示。



選取 [檢視所有意圖] 以導覽至 [效能] 儀表板的 [意圖效能] 子區段，您可以在其中檢視機器人在履行意圖時效能的指標。如需詳細資訊，請參閱 意圖性能。

前 5 名失敗的意圖

使用此圖表可識別您的機器人未能完成的前五個意圖 (請參閱以意圖取得失敗意圖的定義)。將游標暫留在列上，以查看您的機器人無法達成該意圖的次數，如下圖所示。



選取 [檢視失敗的意圖] 以導覽至 [效能] 儀表板的 [意圖效能] 子區段，您可以在其中檢視機器人未能履行之意圖的量度。如需詳細資訊，請參閱 意圖性能。

對話儀表板：您的機器人對話摘要

對話儀表板可視化客戶與您的機器人對話的指標（对话如需對話的定義，請參閱）。

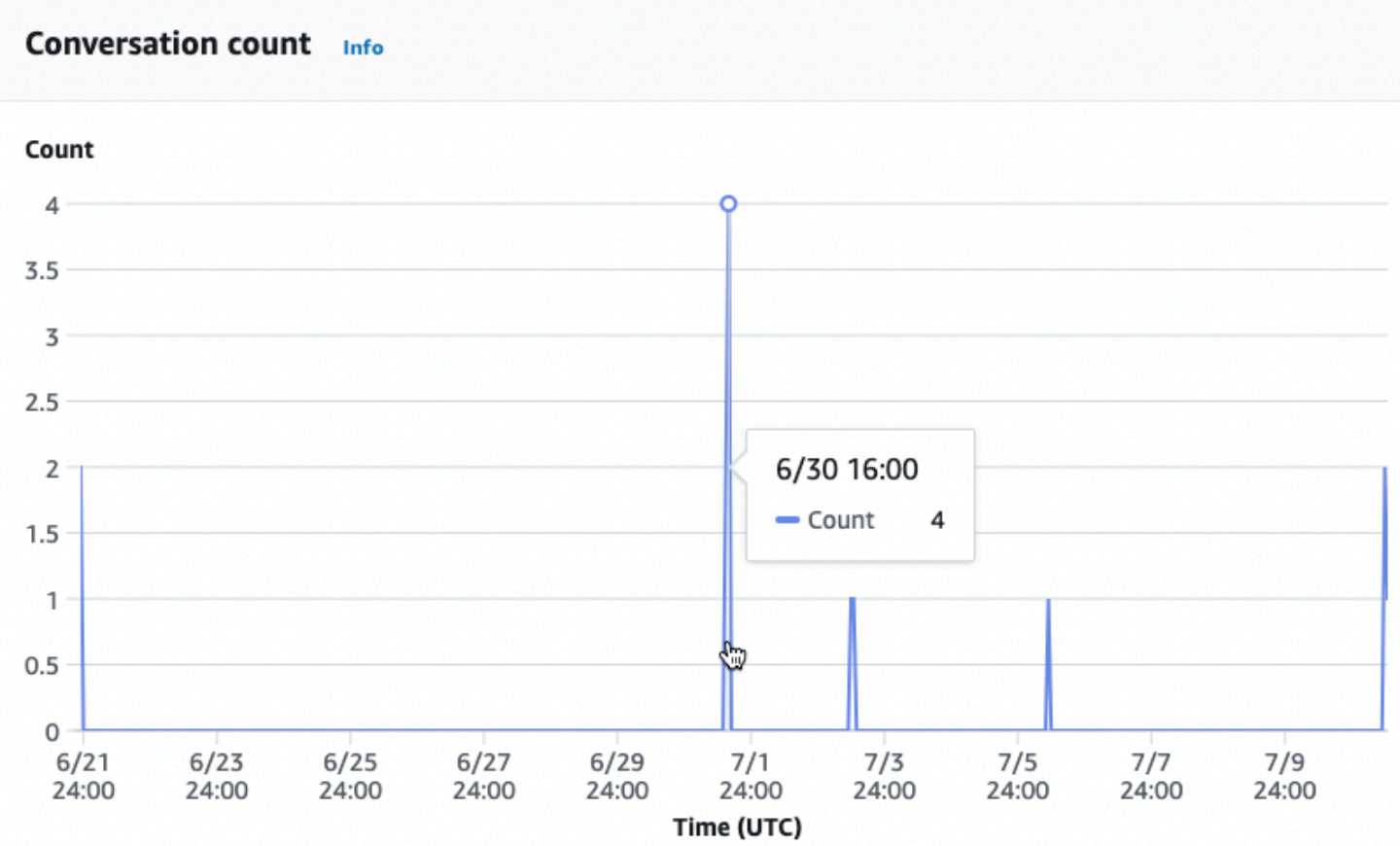
摘要包含以下有關使用者與您的機器人對話的資訊。數字是根據篩選器設定計算的。

- 對話總數 — 與機器人的對話總數。
- 平均對話持續時間 — 使用者與機器人對話的平均時間 (以分鐘和秒為單位)。格式為毫米：SS。
- 每個交談的平均回合次數 — 交談持續的平均回合次數。

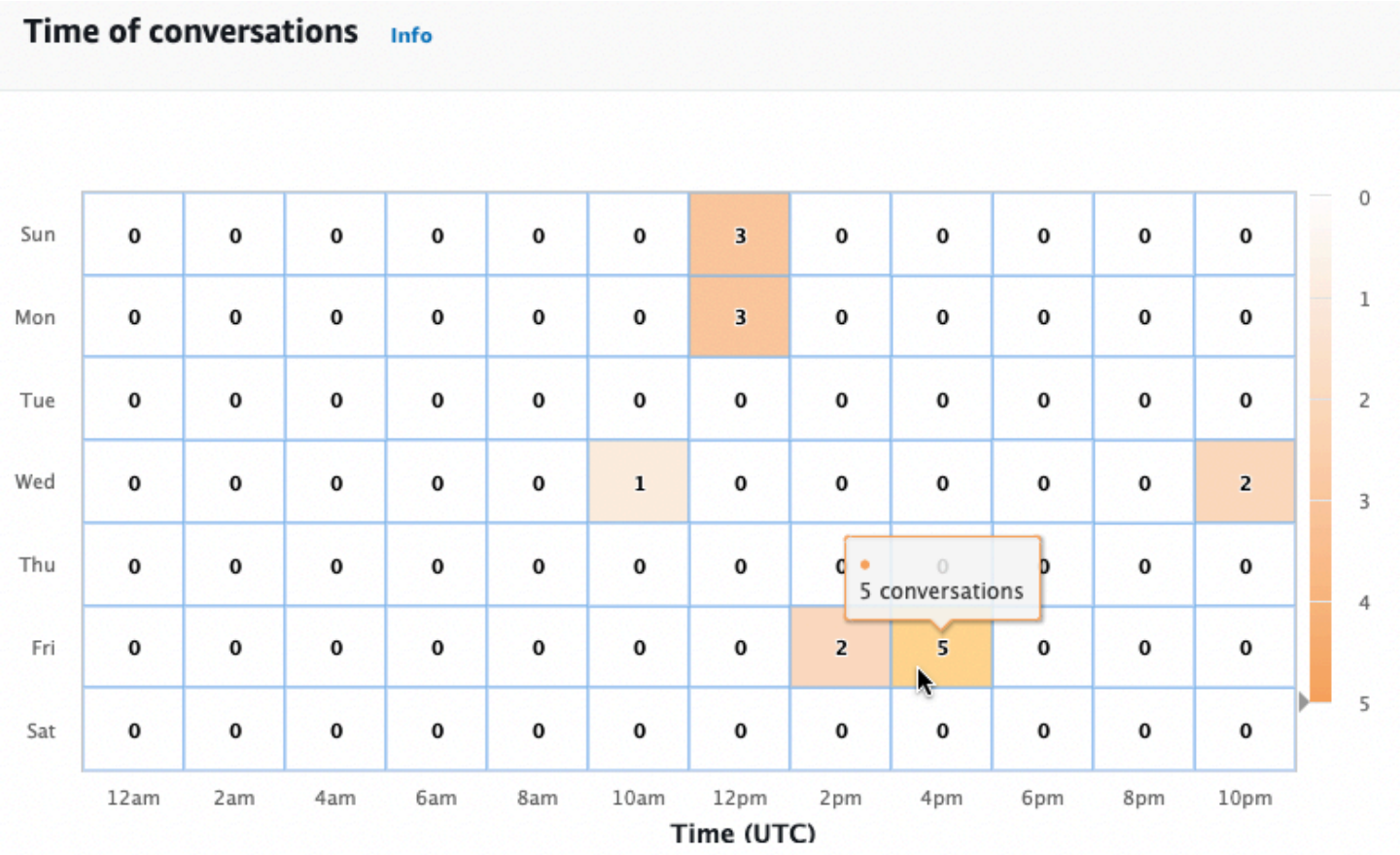
[交談] 計數和 [訊息計數] 區段各包含一個圖表，其中分別顯示您在篩選器中指定的時間範圍內的交談和訊息數目。將滑鼠游標暫留在一段時間上，即可查看該區段中對話或訊息的數量。時間區段的大小取決於您指定的時間範圍：

- 少於 1 週 — 顯示每小時的計數。
- 1 週或更長時間 — 顯示每一天的計數。

請參閱下圖中懸停行為的範例。



[交談時間] 區段會顯示在您在篩選器中指定的時間範圍內，在一週的每兩個小時間隔內，機器人與客戶之間發生的交談次數。更深色陰影的單元格表示發生更多對話的時間。將滑鼠游標暫留在儲存格上，即可顯示從該時段開始 2 小時內的交談次數。例如，下圖中的動作顯示 UTC 下午 4:00 到下午 6:00 之間發生的交談數目。



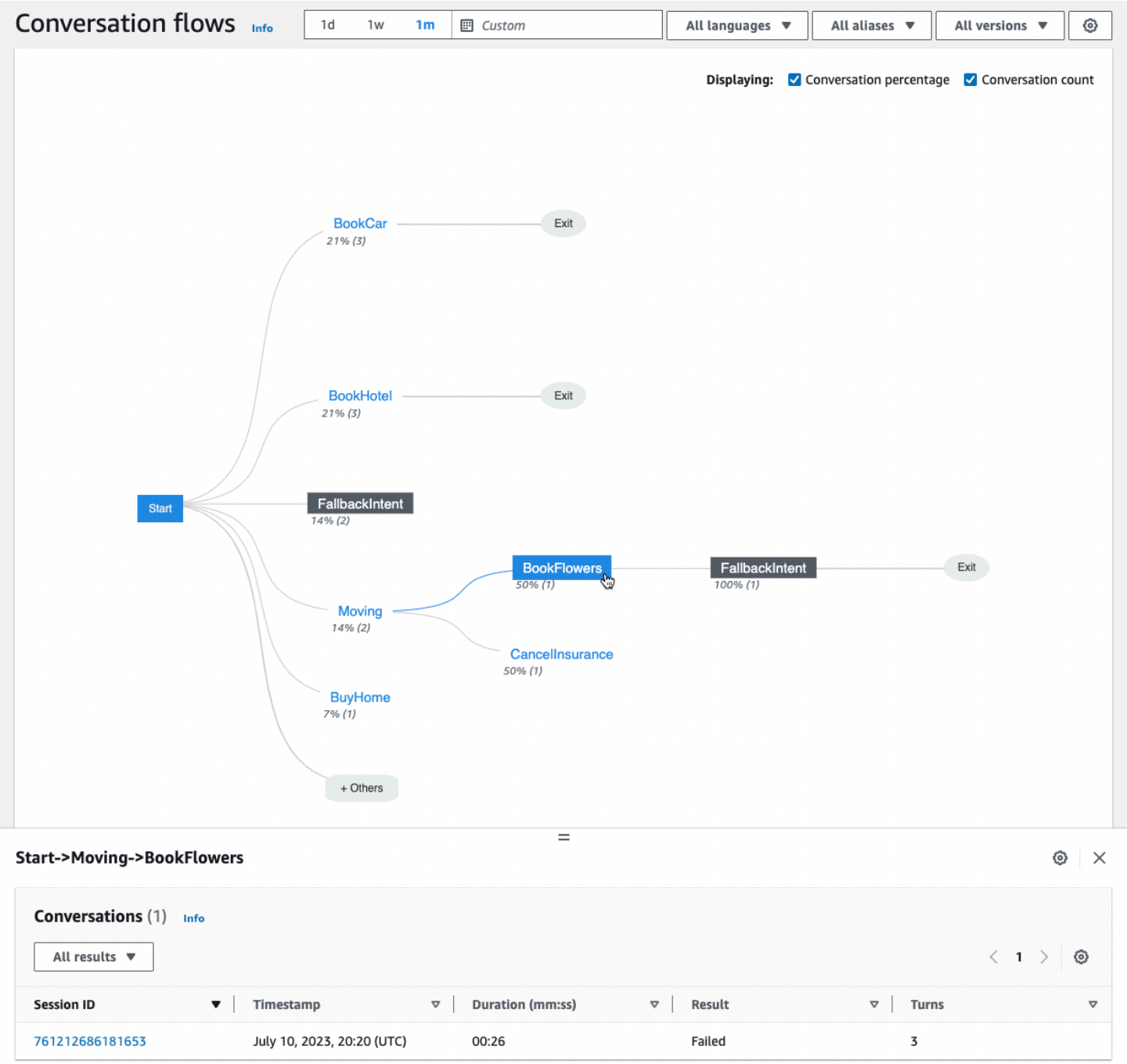
[交談] 儀表板包含兩個工具：[交談] 流程和 [交談] 在左側導覽窗格的 [交談] 儀表板下選取工具，即可存取工具。

對話流程

使用對話流程，以視覺化方式呈現客戶與您的機器人對話時所採取的意圖順序。每個意圖的下方是在交談中該時間點叫用該意圖的交談百分比和計數。您可以通過選擇頂部的對話百分比或對話計數來切換百分比並計數。根據預設，交談中該時間點的五個最常見的意圖會以頻率的遞減順序顯示。選取 [+ 其他] 以顯示所有色彩比對方式。

選擇一個意圖以擴展到新的分支列，該列顯示在交談中該點採取的意圖列表，按降序頻率排序。

當您選取交談流程中的節點時，您可以展開下面的視窗，以顯示遵循該意圖順序的交談清單。選擇與交談對應的工作階段 ID，以檢視有關該交談的詳細資料。下圖顯示了一個交談流程和一個展開的交談視窗在底部。



对话

對話工具會顯示您機器人的對話清單。您可以選取要依該欄的遞增或遞減順序排序的欄。

若要依結果篩選交談，請選取所有結果，然後選擇 [成功]、[失敗] 或 [已捨棄]。

若要依持續時間篩選對話

1. 選擇標記為按持續時間過濾對話的搜索欄
2. 使用下列其中一種方式定義篩選器：
   - 使用預先定義的選項。
     a. 選取持續時間。
     b. 在 = (等於)、> (大於) 和 < (小於) 運算子之間進行選擇。
     c. 選擇時間長度。
   - 以「持續時間 {operator} {數字} 秒」格式輸入輸入。例如，若要搜尋持續超過 30 秒的所有對話，請輸入**`Duration > 30 sec`**。指定時間長度 (以秒為單位)。

若要檢視工作階段的詳細資訊，包括中繼資料、意圖使用情況和成績單，請選取交談的工作階段 ID。

> **Note**
>
> 因為交談是 a `sessionId` 和的唯一組合`originatingRequestId`，因此相同內容`sessionId`可能會出現在表格中多次。

「詳細資訊」區段包含下列中繼資料：

- 時間戳記 — 指定交談的日期和開始時間。時間是 HH: mm: ss 格式。
- 持續時間 — 指定交談以 mm: ss 格式持續的時間長度。持續時間不包括會話超時持續時間（`idleSessionTTLInSeconds`）。
- 結果 — 指定是否將交談分類為成功、失敗或中斷。对话如需這些結果的詳細資訊，請參閱。
- 模式 — 指定交談是`SpeechText`、還是 DTMF (按下觸控音鍵盤)。由多種模式組成的對話是`Multimode`。
- 頻道 — 指定發生交談的通道 (如果適用的話)。請參閱將 Amazon Lex V2 機器人與簡訊平台整合。
- 語言 — 指定機器人的語言。

機器人在對話中引起的意圖顯示在「詳細資料」下方。選取「移至意圖」以在意圖編輯器中移至該意圖。選取「貼齊至文字記錄」，自動將成績單捲動至機器人提出意圖的第一個例項。

選取意圖名稱旁邊的右箭頭，以檢視針對意圖而產生的插槽詳細資訊，包括插槽名稱、機器人針對每個插槽產生的值，以及機器人嘗試引出每個插槽的次數。

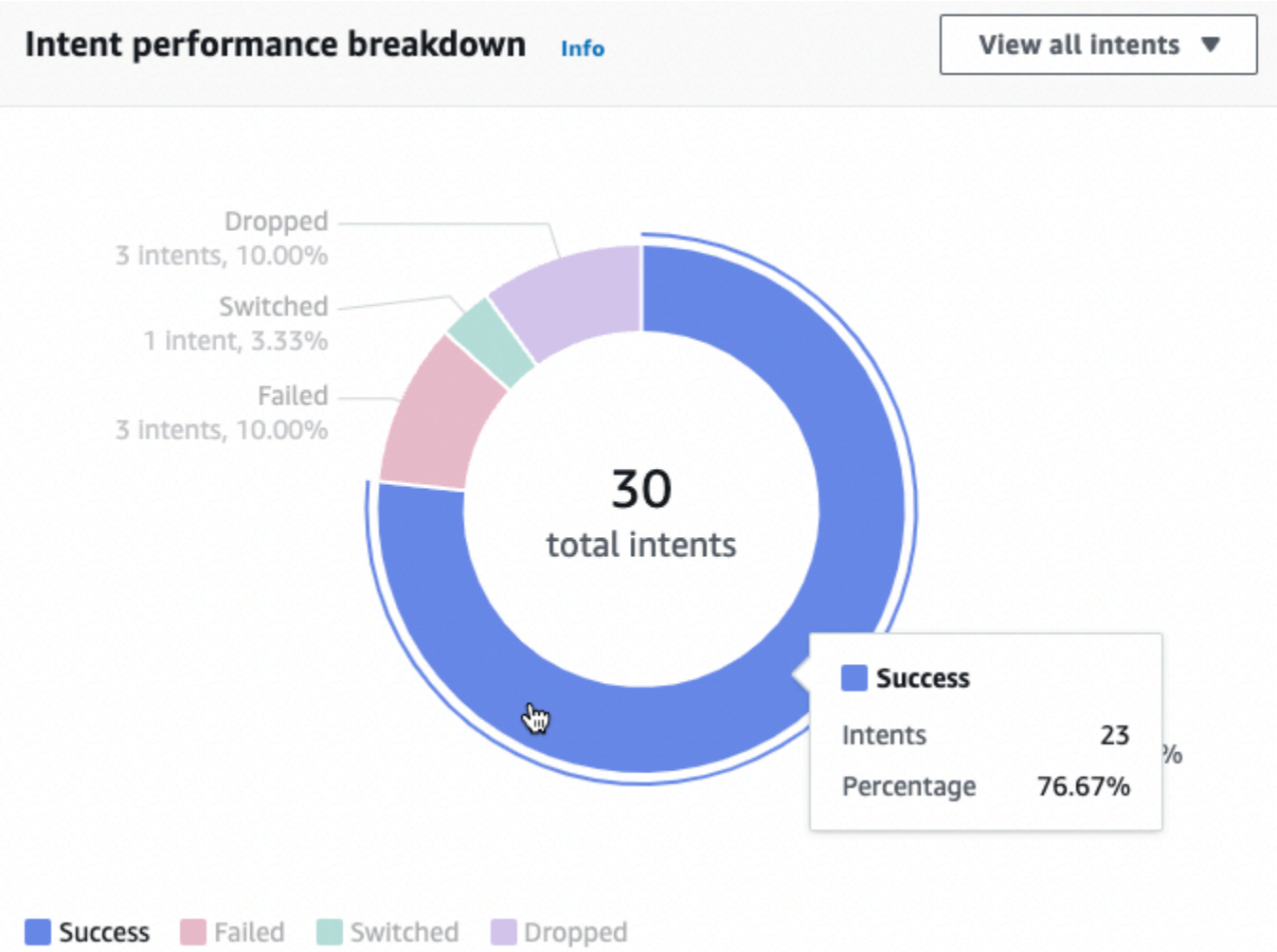
成績單可讓您查看對話話語和機器人在引出意圖和插槽時的行為。使用者話語會顯示在左側，而機器人的話語則顯示在右側。使用此工作階段中標示為篩選文字單的搜尋列來尋找成績單中的文字。在「顯示」旁邊：每個對話回合下都會顯示三條資訊，您可以選擇是否顯示：

- 時間戳記 — 指定說話的時間。
- 意圖狀態 — 指定機器人在話語期間引出的意圖和意圖的結果 (如果適用)。下列意圖狀態是可能的：
  - 呼叫意圖：#### — 機器人已識別客戶正在呼叫的意圖。
  - 切換意圖：#### — 機器人已根據語言切換到不同的意圖。
  - ####：成功 — 機器人已滿足意圖。
- 插槽狀態 — 指定機器人在話語期間引出的位置 (如果適用)，以及客戶提供的值。

效能儀表板：機器人意圖和話語指標的摘要

在性能儀表板中，您可以查看有關機器人意圖實現性能和說話識別的詳細信息。

「意圖效能明細」區段會顯示您的機器人叫用意圖的總次數，並細分意圖被分類為成功、失敗、刪除和切換的次數和百分比。意圖如需這些定義的說明，請參閱。將滑鼠游標暫留在圖表的某個區段上，以顯示包含該結果之交談計數和百分比的方塊，如下圖所示。



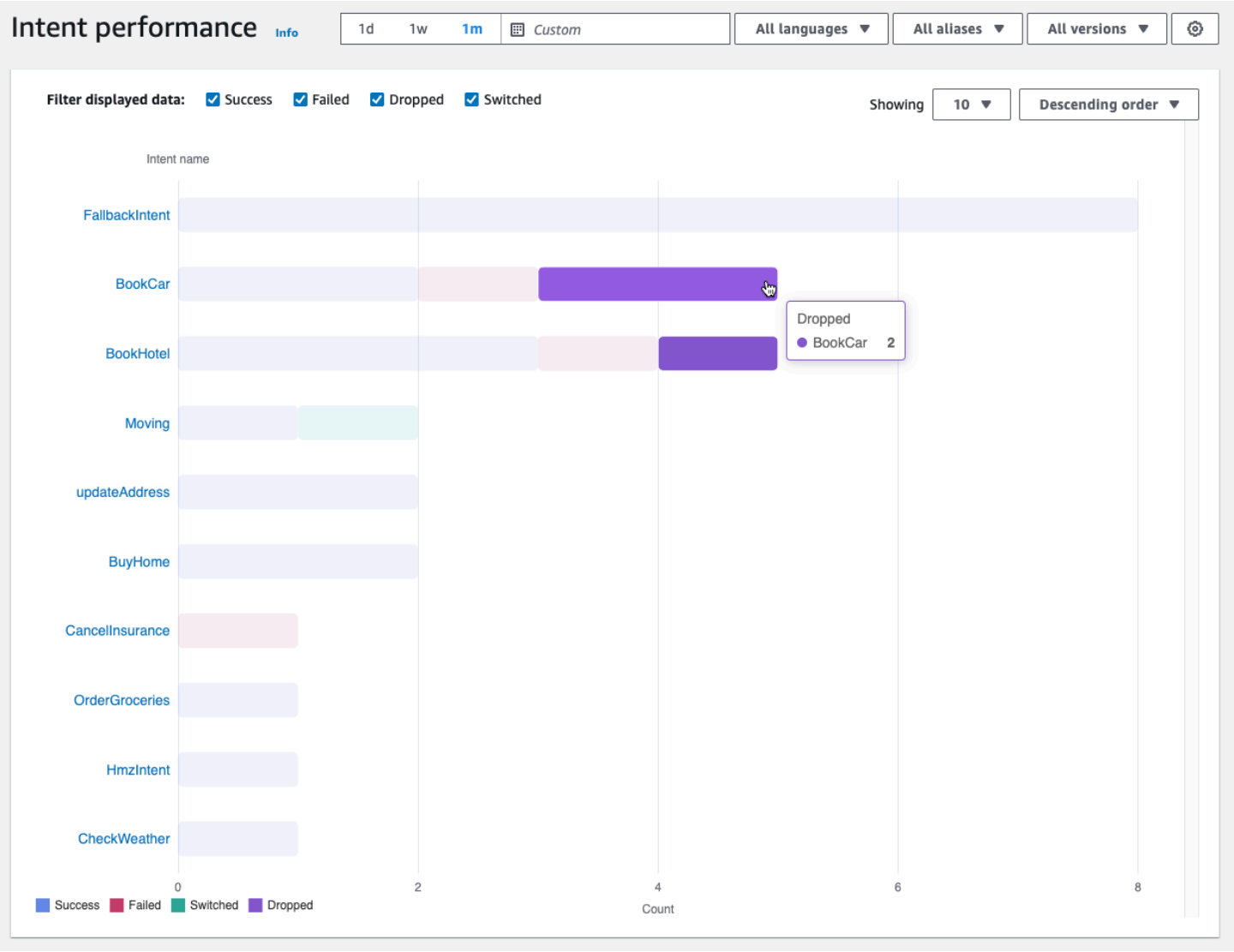
選取 [檢視所有意圖] 以顯示下拉式選單，您可以從中選擇檢視機器人提出的意圖清單。您也可以選擇檢視具有特定結果 (成功、失敗、中斷或切換) 的意圖。這些連結會帶您前往「效能」儀表板的「意圖」效能子區段。如需詳細資訊，請參閱 意圖性能。

「語音辨識」部分總結了遺漏和偵測到的語音數量。選取 [檢視詳細資料] 以導覽至機器人的話語清單。選擇「遺漏話語」下的數字，以查看遺漏的話語清單以及「偵測到的話語」下方的數字，以查看機器人偵測到的話語清單。如需詳細資訊，請參閱 语言识别。

在左側側邊欄的「效能」儀表板下，選取「意圖效能」和「話語辨識」，即可檢視機器人中意圖和話語的詳細資料。

意圖性能

此儀表板總結了與機器人搭配使用之意圖的效能 (依頻率降序排列)。每個意圖旁邊的列會以視覺化方式分類為成功、失敗、捨棄和切換意圖的次數。意圖如需這些定義的說明，請參閱。將滑鼠游標暫留在列的某個區段上，以查看使用該意圖與該結果的交談次數，如下圖所示：



> **Note**
>
> 儀表板會顯示一組篩選器設定的前 1,000 個結果。若要取得更具目標性的結果，請設定精細篩選設定。

在圖表頂端，您可以使用「成功」、「失敗」、「已刪除」和「已切換」核取方塊切換要檢視的意圖狀態。

選取「顯示」右側的下拉式功能表，可調整要顯示的對應方式數目，以及是以頻率的遞增還是遞減順序顯示方式。

選取意圖名稱以瀏覽顯示三個圖表的頁面：意圖效能明細、插槽效能及意圖切換。

「意圖效能明細」區段會顯示機器人使用意圖的總次數，並細分意圖履行被歸類為成功、失敗、刪除和切換的次數和百分比。意圖如需這些定義的說明，請參閱。將滑鼠游標暫留在圖表區段上，即可查看意圖履行產生該結果的計數和次數百分比。

「插槽效能」段落會顯示屬於目前意圖之插槽的測量結果。若要依欄排序，請選取該欄一次以遞增順序排序，選取兩次以遞減順序排序。您可以使用搜尋列尋找特定位置，或使用頁碼按鈕來瀏覽插槽。

> **Note**
>
> 儀表板會顯示一組篩選器設定的前 1,000 個結果。若要取得更有針對性的結果，請設定精細篩選設定。

「意圖切換」區段會列出機器人從目前意圖切換到另一個目標的執行個體，其中包含下列資訊：

- 階段 — 機器人切換意圖的對話階段。
- 意圖切換到 — 機器人切換目前意圖的目的。
- 階段作業計數 —「階段」和「意圖」切換至組合的工作階段數目。

> **Note**
>
> 儀表板會顯示一組篩選器設定的前 1,000 個結果。若要取得更有針對性的結果，請設定精細篩選設定。

语言识别

此頁面列出了您的機器人錯過和檢測到的所有話語，並提供了一些工具，讓您可以在意圖中添加示例語句以幫助訓練您的機器人。表達用語如需這些定義的說明，請參閱。使用頂端的索引標籤可在遺漏的話語清單與偵測到的話語清單之間切換。

> **Note**
>
> 儀表板會顯示一組篩選器設定的前 1,000 個結果。若要取得更具目標性的結果，請設定精細篩選設定。

要將語音添加到意圖中：

1. 選擇您要新增為意圖範例語音的話語旁邊的核取方塊。
2. 選取 [新增至意圖]，然後在 [意圖] 下方的下拉式功能表中選擇要新增語音的意圖。
3. 選取新增。

使用 API 進行分析

本節說明您用來擷取機器人分析的 API 作業。

> **Note**
>
> 若要使用ListUtterance指標 ListUtteranceAnalyticsData，您的 IAM 角色必須具有執行「ListAggregated語音」操作的許可，該操作可提供對語言相關分析的存取權。如檢視話語統計資料需要套用至 IAM 角色的詳細資訊和 IAM 政策，請參閱。

- 下列 API 作業會擷取機器人的摘要指標：
  - ListSession指標
  - ListIntent指標
  - ListIntentStageMetrics
  - ListUtterance指標
- 下列 API 作業會擷取工作階段和話語的中繼資料清單：
  - ListSessionAnalyticsData
  - ListUtteranceAnalyticsData
- P ListIntentath s 作業會擷取客戶與機器人交談時所採取之意圖順序的量度。

篩選結果

分析 API 請求會要求您指定`startTime`和`endTime`。API 會傳回在. 之後開始*`startTime`*和結束的工作階段、意圖階段或語音。 `endTime`

`filters`是分析 API 請求中的可選欄位。它會對應至「濾鏡」、「AnalyticsSession濾鏡」或 AnalyticsUtterance「AnalyticsIntent濾鏡」物件的清單。AnalyticsIntentStageFilter在每個物件中，使用欄位建立要篩選依據的運算式。例如，如果您將下列篩選器新增至清單，則機器人會搜尋超過 30 秒的交談。

```
{
```

```
    "name": "Duration",
    "operator": "GT",
    "value": "30 sec",
}
```

擷取機器人的指標

使用ListSessionMetrics、ListIntentMetricsListIntentStageMetrics、和ListUtteranceMetrics作業擷取工作階段、意圖、意圖階段和語音的摘要量度。

對於這些作業，請填寫下列必要欄位：

- 提供startTime和，endTime以定義您要擷取結果的時間範圍。
- 指定您要計算的量度、「量度」metrics、「AnalyticsSession量度」或AnalyticsUtterance「AnalyticsIntent量度」物件的清單。AnalyticsIntentStageMetric在每個物件中，使用name欄位指定要計算statistic欄位的量度，以指定是否要計算SumAverage、或Max數字，以及使用order欄位來指定是按結果排序Ascending還是按Descending順序排序。

> **Note**
>
> metrics和binBy物件都包含order欄位。您只能在兩個物件的其order中一個中指定排序。

請求中的其餘字段是可選的。您可以使用下列方式篩選和組織結果：

- 篩選結果 — 使用filters欄位來篩選結果。如需詳細資訊，請參閱篩選結果。
- 依類別分組結果 — 指定groupBy欄位、包含單一「結果」、「AnalyticsSession結果」或AnalyticsUtterance「AnalyticsIntent結果」物件的清單。AnalyticsIntentStageResult在物件中，指定含有要群組結果所依據之類別的name欄位。

  如果您在要求中指定groupBy欄位，則回應中的results物件會包含 Key groupByKeys、Key 或 AnalyticsSessionGroupByAnalyticsUtteranceGroupByKey 物件的清單，每個物件都包含您在name要求中指定的物件，以及value欄位中該類別的成員。AnalyticsIntent GroupBy AnalyticsIntentStageGroupByKey
- 依時間排列結果 — 指定binBy欄位，此欄位包含單一AnalyticsBinBySpecification物件的清單。在物件中，指定ConversationStartTime要在交談開始時收納結果的name欄位，或者在發生話語時UtteranceTimestamp將結果分隔起來。指定要在欄位中儲存結果的時間間隔，以及是否要在interval欄位中依時間排序Ascending或時間Descending順序。order

如果您在要求中指定binBy欄位，回應中的results物件會包含binKeys一個 AnalyticsBinKey 物件清單，其中包含您在要求中指name定的金鑰物件清單，以及在value欄位中定義該 bin 的時間間隔。

> **Note**
>
> metrics和binBy物件都包含order欄位。您只能在兩個物件的其order中一個中指定排序。

使用下列欄位來處理回應的顯示：

- 在欄位中指定介於 1 到 1,000 之間的數maxResults字，以限制在單一回應中傳回的結果數目。
- 如果結果數大于您在maxResults字段中指定的數目，那么響應會包含一個nextToken。再次提出要求，但在nextToken欄位中使用此值可傳回下一批結果。

如果您正在使用ListUtteranceMetrics，則可以指定要在attributes欄位中傳回的屬性。此欄位對應至包含單一AnalyticsUtterance屬性物件的清單。LastUsedIntent在name欄位中指定，以傳回 Amazon Lex V2 在發言時使用的意圖。

在回應中，results欄位會對應至「結果」、「AnalyticsSession結果」或 AnalyticsUtterance「AnalyticsIntent結果」物件的清單。AnalyticsIntentStageResult每個物件都包含一個metrics欄位，此欄位會傳回您要求之測量結果的摘要統計值，以及透過您指定之方法建立的任何資料桶或群組。

擷取機器人中工作階段和語音的中繼資料

使用ListSessionAnalyticsData和作ListUtteranceAnalyticsData業擷取有關個別工作階段和話語的中繼資料。

填寫必要endTime欄位startTime和欄位，以定義您要擷取結果的時間範圍。

請求中的其餘字段是可選的。若要篩選和排序結果：

- 篩選結果 — 使用filters欄位來篩選結果。如需詳細資訊，請參閱篩選結果。
- 排序結果 — 使用包含SessionDataSortBy或物件的sortBy欄位UtteranceDataSortBy對結果進行排序。指定要在欄位中排序依據的值，以及是在name欄位中依據排序Ascending還是按Descending順序order排序。

使用下列欄位來處理回應的顯示：

- 在欄位中指定介於 1 到 1,000 之間的數maxResults字，以限制在單一回應中傳回的結果數目。
- 如果結果數大于您在maxResults字段中指定的數目，那么響應會包含一個nextToken。再次提出要求，但在nextToken欄位中使用此值可傳回下一批結果。

在回應中，sessions或utterances欄位會對應至SessionSpecification或UtteranceSpecification物件清單。每個物件都包含單一工作階段或話語的中繼資料。

擷取機器人中工作階段和語音的中繼資料

使用「ListIntent路徑」作業可擷取客戶與機器人交談之意圖順序的相關指標。

對於此操作，請填寫以下必填字段：

- 提供startTime和，endTime以定義您要擷取結果的時間範圍。
- 提供intentPath定義您要擷取度量的意圖順序。使用正斜線分隔路徑中的對應方式。例如，在intentPath欄位中填入，**/BookCar/BookHotel**以查看使用者以該順序呼叫BookCar和BookHotel意圖次數的詳細資訊。

使用選擇性filters欄位來篩選結果。如需詳細資訊，請參閱篩選結果。

檢視話語統計資料

您可以使用話語統計信息來確定用戶發送給您的機器人的話語。您可以看到 Amazon Lex V2 成功偵測到的語音，以及看不到的話語。您可以使用此信息來幫助調整您的機器人。

例如，如果您發現使用者正在傳送 Amazon Lex V2 遺失的語音，您可以將語音新增到意圖中。意圖的草稿版本會以新的話語進行更新，您可以在將其部署到機器人之前對其進行測試。

當 Amazon Lex V2 將語音識別為嘗試叫用針對機器人設定的意圖時，就會偵測到語音。當 Amazon Lex V2 無法識別語音並調用語音時，會錯過一個話語。AMAZON.FallbackIntent

可以使用 ListUtteranceMetrics API 和 API 查看語音統計信息。ListAggregatedUtterance

在下列情況下，不會使用 ListUtteranceMetrics API 產生話語統計資料：

- 使用主控台建立機器人時，[兒童線上隱私權保護法案] 設定設定為 [是]，或者當機器人透過CreateBot作業建立時，childDirected欄位設定為 true。

該 ListUtteranceMetrics API 提供了額外的功能，包括：

- 可用的更多資訊，例如偵測到語音的對應意圖。
- 更多的濾波功能（包括通道和模式）。
- 更長的保留日期範圍（30 天）。
- 即使您已選擇退出資料儲存，也可以使用 API。遺漏和偵測到的話語的主控台功能將取決ListUtteranceMetrics於 API。

在下列情況下，不會使用 ListAggregatedUtterance API 產生話語統計資料：

- 使用主控台建立機器人時，[兒童線上隱私權保護法案] 設定設定為 [是]，或者當機器人透過CreateBot作業建立時，childDirected欄位設定為 true。
- 您正在使用一個或多個插槽的插槽混淆。
- 您選擇不參與改善 Amazon Lex 的活動。

該 ListAggregatedUtterance API 提供的功能包括：

- 較少的詳細資訊可用 (沒有對應的語音意圖)。
- 有限的濾波功能（不包括通道和模式）。
- 短保留日期範圍（15 天）。

使用話語統計資料，您可以查看是否偵測到或遺漏特定語音，以及上次在機器人互動中使用該語音的時間。

當使用者與您的機器人互動時，Amazon Lex V2 會持續存放語音。您可以使用控制台或ListAggregatedUtterances操作查詢統計信息。它的數據保留時間為 15 天，如果用戶選擇退出數據存儲，則不可用。您可以使用DeleteUtterances作業或選擇退出資料儲存來刪除語音。如果您 AWS 關閉帳戶，所有語音都會被刪除。儲存的語音會使用由伺服器管理的金鑰加密。

當您刪除機器人版本時，該版本的話語統計最多可使用 30 天ListUtteranceMetrics，並使用 15 天。ListAggregatedUtterances您無法在 Amazon Lex V2 主控台中查看已刪除版本的統計資料。要查看已刪除版本的統計信息，可以同時使用ListAggregatedUtterances和ListUtteranceMetrics操作。

對於ListAggregatedUtterances和 ListUtteranceMetrics API，話語是由語音文本彙總的。例如，客戶使用「我要訂購比薩餅」一詞的所有執行個體都會在回應中彙總到同一行中。當您使用該RecognizeUtterance操作時，使用的文本是輸入成績單。

若要使用ListAggregatedUtterances和 ListUtteranceMetrics API，請將下列原則套用至角色。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ListAggregatedUtterancesPolicy",
            "Effect": "Allow",
            "Action": "lex:ListAggregatedUtterances",
            "Resource": "*"
        }
    ]
}
```

管理分析的存取權限

若要提供使用者對分析的存取權，請將政策附加至 IAM 角色，以允許該角色呼叫 API 作業進行分析。您可以附加AWS 受管政策： AmazonLexFullAccess到 IAM 角色以提供對 Amazon Lex API 操作的完整存取權，或者建立僅允許分析許可的自訂政策，並將其附加到 IAM 角色。

若要建立包含分析權限的自訂原則

1. 如果您需要先建立 IAM 角色，請按照建立角色將許可委派給 IAM 使用者中的步驟進行操作。
2. 請遵循建立 IAM 政策中的步驟，使用下列 JSON 物件建立政策。若要啟用 IAM 角色特定機器人的分析存取權，請將每個機器人的 ARN 新增至Resource欄位。將*##*、*## ID # BOT ID* 取代為與機器人對應的值。您也可以使用您選擇的名稱取代陳述式識別碼。*AnalyticsActions*

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AnalyticsActions",
            "Effect": "Allow",
            "Action": [
                "lex:ListAggregatedUtterances",
                "lex:ListIntentMetrics",
```

```
                "lex:ListSessionAnalyticsData",
                "lex:ListIntentPaths",
                "lex:ListIntentStageMetrics",
                "lex:ListSessionMetrics"
            ],
            "Resource": [
                "arn:aws:lex:region:account-id:bot/BOTID"
            ]
        }
    ]
}
```

3. 按照新增和移除 IAM 身分許可中的步驟，將您建立的政策附加到要授予分析許可的角色。
4. 該角色現在應具有檢視您指定機器人分析的權限。

啟用交談記錄

使用對話記錄來儲存使用者與機器人的對話。檢閱這些日誌以識別機器人與使用者互動的問題，並利用這些見解修改機器人的行為。本節還描述了如何混淆插槽值以保護用戶的隱私。

主題

- 使用交談記錄記錄
- 隱藏交談記錄中的插槽值
- 選擇性交談記錄擷取

使用交談記錄記錄

您可以啟用「對話日誌」來存放機器人互動。您可以使用這些日誌來檢閱機器人的效能，以及疑難排解對話問題。您可以記錄RecognizeText作業的文字。您可以記錄RecognizeUtterance操作的文本和音頻。通過啟用對話日誌，您可以獲得用戶與機器人進行的對話的詳細視圖。

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。文字記錄會在記錄中儲存文字輸入、音訊輸入的文字 CloudWatch 記錄和關聯的中繼資料。音訊日誌會將音訊輸入存放在 Amazon S3 中。您可以使用 AWS KMS 客戶管理的 CMK 來啟用文字和音訊日誌的加密。

若要設定記錄，請使用主控台或CreateBot別名或UpdateBot別名作業。啟用別名的交談日誌後，對該別名使用RecognizeText或RecognizeUtterance操作會記錄已設定的日誌日 CloudWatch 誌群組或 S3 儲存貯體中的文字或音訊語音。

主題

交談記錄的 IAM 政策

視您選取的記錄類型而定，Amazon Lex V2 需要使用 Amazon CloudWatch 日誌和 Amazon Simple Storage Service (S3) 貯體來存放日誌的許可。您必須建立 AWS Identity and Access Management 角色和許可，才能讓 Amazon Lex V2 存取這些資源。

建立對話日誌的 IAM 角色和政策

若要啟用交談日誌，您必須授與 CloudWatch 日誌和 Amazon S3 的寫入權限。如果您為 S3 物件啟用物件加密，則需要授與用於加密物件之 AWS KMS 金鑰的存取權限。

您可以使用 IAM 主控台、IAM API 或 AWS Command Line Interface 建立角色和政策。這些指示使 AWS CLI 用建立角色和原則。

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

建立交談記錄的 IAM 角色

1. 在稱為 **LexConversationLogsAssumeRolePolicyDocument.json** 的目前目錄中建立一個文件、將下列程式碼新增至其中，然後儲存它。本政策文件將 Amazon Lex V2 新增為該角色的受信任實體。這可讓 Amazon Lex 擔任角色，將日誌傳遞到針對交談日誌設定的資源。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "lexv2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

2. 在中 AWS CLI，執行下列命令以建立交談記錄的 IAM 角色。

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

接下來，建立政策並將其附加到可讓 Amazon Lex V2 寫入 CloudWatch 日誌的角色。

若要建立 IAM 政策，將對話文字記錄到記 CloudWatch 錄

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

```
{
  "Version": "2012-10-17",
  "Statement": [
      {
          "Effect": "Allow",
          "Action": [
              "logs:CreateLogStream",
              "logs:PutLogEvents"
```

```
            ],
            "Resource": "arn:aws:logs:region:account-id:log-group:log-group-name:*"
        }
    ]
}
```

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

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

3. 將政策附加到您為交談記錄建立的 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 V2 寫入儲存貯體的政策。

若要為 S3 儲存貯體的音訊記錄建立 IAM 政策

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

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

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

```
aws iam create-policy \
```

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

3. 將此政策附加到您為對話日誌建立的角色。

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

授與傳遞 IAM 角色的權限

當您使用主控台 AWS Command Line Interface、或 AWS SDK 來指定用於交談日誌的 IAM 角色時，指定交談日誌 IAM 角色的使用者必須具有將角色傳遞給 Amazon Lex V2 的權限。若要允許使用者將角色傳遞給 Amazon Lex V2，您必須將`PassRole`權限授與使用者的 IAM 使用者、角色或群組。

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

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::account-id:role/role-name",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "lexv2.amazonaws.com"
                },
                "StringLike": {
                    "iam:AssociatedResourceARN": "arn:aws:lex:region:account-
id:bot:bot-name:bot-alias"
                }
            }
        }
    ]
}
```

設定交談記錄

您可以使用主控台或或UpdateBotAlias作業的conversationLogSettings欄位來啟用和停用CreateBotAlias交談記錄。您可以開啟或關閉音訊日誌、文字日誌或兩者。記錄會在新的機器人工作階段開始。對於作用中的工作階段，不會反映日誌設定的變更。

若要存放文字日誌，請在您的 AWS 帳戶中使用 Amazon CloudWatch 日誌日誌群組。您可以使用任何有效的日誌群組。日誌群組必須與 Amazon Lex V2 機器人位於相同的區域。如需有關建立 CloudWatch 日誌日誌群組的詳細資訊，請參閱 Amazon 日誌使用指南中的使用 CloudWatch 日誌群組和日誌串流。

若要存放音訊日誌，請在您的 AWS 帳戶中使用 Amazon S3 儲存貯體。您可以使用任何有效的 S3 儲存貯體。儲存貯體必須與 Amazon Lex V2 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 簡單儲存服務入門指南中的建立儲存貯體。

當您使用主控台管理交談日誌時，主控台會更新您的服務角色，以便它可以存取日誌群組和 S3 儲存貯體。

如果您未使用主控台，則必須提供 IAM 角色與政策，讓 Amazon Lex V2 寫入已設定的日誌群組或儲存貯體。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，如下列範例所示。如需詳細資訊，請參閱 建立對話日誌的 IAM 角色和政策。

```
aws iam create-service-linked-role \
    --aws-service-name lexv2.amazon.aws.com \
    --custom-suffix suffix
```

您用來啟用交談記錄的 IAM 角色必須具有iam:PassRole權限。下列原則應附加至角色：

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

啟用交談記錄

使用主控台開啟日誌

1. 打開 Amazon Lex V2 控制台 https://console.aws.amazon.com/lexv2。
2. 從清單中選擇一個機器人。
3. 從左側選單中選擇「別名」。
4. 在別名清單中，選擇您要設定交談記錄的別名。
5. 在 [交談記錄] 區段中，選擇 [管理交談記錄]。
6. 對於文字日誌，請選擇啟用，然後輸入 Amazon CloudWatch 日誌日誌群組名稱。
7. 對於音訊日誌，請選擇啟用，然後輸入 S3 儲存貯體資訊。
8. 選用。若要加密音訊記錄，請選擇要用於加密的 AWS KMS 金鑰。
9. 選擇 Save (儲存) 以開始記錄對話。如有必要，Amazon Lex V2 會更新您的服務角色，並具有存取 CloudWatch 日誌日誌群組和所選 S3 儲存貯體的許可。

停用交談記錄

使用主控台關閉日誌

1. 打開 Amazon Lex V2 控制台 https://console.aws.amazon.com/lexv2。
2. 從清單中選擇一個機器人。
3. 從左側選單中選擇「別名」。
4. 在別名清單中，選擇您要設定交談記錄的別名。
5. 在 [交談記錄] 區段中，選擇 [管理交談記錄]。
6. 停用文字記錄、音訊記錄或兩者皆可關閉記錄功能。
7. 選擇 Save (儲存) 以停止記錄對話。

在 Amazon 日誌中查看文本 CloudWatch 日誌

Amazon Lex V2 會將您對話的文字日誌儲存在 Amazon CloudWatch 日誌中。若要檢視記錄檔，請使用 CloudWatch 記錄主控台或 API。如需詳細資訊，請參閱 Amazon Logs 使用者指南中的使用篩選器模式和CloudWatch 日誌見解查詢語法搜尋 CloudWatch 日誌資料。

若要使用 Amazon Lex V2 主控台檢視日誌

1. 打開 Amazon Lex V2 控制台 https://console.aws.amazon.com/lexv2。
2. 從清單中選擇一個機器人。
3. 從左側功能表中選擇「分析」，然後選擇CloudWatch 指標。
4. 在指標頁面上檢視機器人的CloudWatch 指標。

您也可以使用主 CloudWatch 控台或 API 來檢視您的記錄項目。若要尋找日誌項目，請導覽至您針對別名設定的日誌群組。您可以在 Amazon Lex V2 主控台或使用DescribeBot別名作業，找到日誌的日誌串流前置詞。

您可以在多個記錄串流中找到使用者語音的記錄項目。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。記錄資料流中的項目包含下列資訊：

訊息版

訊息結構描述版本。

機器人

有關客戶正在與之互動的機器人的詳細信息。

messages

機器人傳回給使用者的回應。

语言上下文

有關處理此語音的資訊。

- `runtimeHints`—runtime 上下文用於轉錄和解釋用戶的輸入。如需詳細資訊，請參閱 使用執行時期提示改善位置值的辨識。
- `slotElicitationStyle`用於解譯使用者輸入的槽引出樣式。如需詳細資訊，請參閱 使用拼字樣式擷取位置值。

工作階段狀態

使用者與機器人之間交談的目前狀態。如需詳細資訊，請參閱 管理對話。

解釋

Amazon Lex V2 確定的意圖清單可以滿足使用者的話語。 使用可信度分數。

解釋來源

表示插槽是否由 Amazon Lex 或 Amazon 基岩解決。價值觀：Lex | 基岩

sessionId

具有交談之使用者工作階段的識別碼。

inputTranscript

來自用戶的輸入的轉錄。

- 對於文字輸入，這是使用者輸入的文字。對於 DTMF 輸入，這是使用者輸入的金鑰。
- 對於語音輸入，這是 Amazon Lex V2 將使用者語音轉換成的文字，以叫用意圖或填滿插槽。

原始 InputTranscript

在套用任何文字處理之前，使用者輸入的原始成績單。注意：文字處理僅適用於 en-US 和 en-GB 地區設定。

抄錄

用戶輸入的潛在轉錄列表。如需詳細資訊，請參閱 使用語音轉錄信心分數。

原始转录

使用語音轉錄信心得分。如需詳細資訊，請參閱 使用語音轉錄信心分數。

失誤行為

指出 Amazon Lex V2 是否能夠辨識使用者的話語。

requestId

Amazon Lex V2 為使用者輸入產生的請求識別碼。

timestamp

使用者輸入的時間戳記。

开发者覆盖

指出是否已使用對話方塊程式碼掛接更新交談流程。如需使用對話方塊程式碼掛接的詳細資訊，請參閱〈〉使用AWS Lambda函數啟用自訂邏輯。

輸入模式

指示輸入的類型。可以是音訊、DTMF 或文字。

requestAttributes

處理用戶的輸入時使用的請求屬性。

音訊屬性

如果音訊交談記錄已啟用且使用者輸入為音訊格式，則包括音訊輸入的總持續時間、語音的持續時間和音訊中的靜音持續時間。它還包括指向音頻文件的鏈接。

駁船

指出使用者輸入是否中斷先前的機器人回應。

回應原因

產生回應的原因。可為下列其中一個：

- UtteranceResponse— 響應用戶輸入
- StartTimeout-服務器生成的響應時，用戶沒有提供輸入
- StillWaitingResponse— 當用戶請求機器人等待時，服務器生成的響應
- FulfillmentInitiated— 服務器生成的響應，履行即將啟動
- FulfillmentStartedResponse— 服務器生成的響應，履行已經開始
- FulfillmentUpdateResponse-在履行過程中定期服務器生成的響應
- FulfillmentCompletedResponse— 服務器在履行完成時生成的響應。

operationName

用來與機器人互動的 API。可以是PutSession、RecognizeText、或其RecognizeUtterance中之一StartConversation。

```
{
    "message-version": "2.0",
    "bot": {
        "id": "string",
        "name": "string",
        "aliasId": "string",
        "aliasName": "string",
        "localeId": "string",
        "version": "string"
    },
    "messages": [
        {
```

```
            "contentType": "PlainText | SSML | CustomPayload | ImageResponseCard",
            "content": "string",
            "imageResponseCard": {
                "title": "string",
                "subtitle": "string",
                "imageUrl": "string",
                "buttonsList": [
                    {
                        "text": "string",
                        "value": "string"
                    }
                ]
            }
        }
    ],
    "utteranceContext": {
        "activeRuntimeHints": {
            "slotHints": {
                "string": {
                    "string": {
                        "runtimeHintValues": [
                            {
                                "phrase": "string"
                            },
                            {
                                "phrase": "string"
                            }
                        ]
                    }
                }
            }
        },
        "slotElicitationStyle": "string"
    },
    "sessionState": {
        "dialogAction": {
            "type": "Close | ConfirmIntent | Delegate | ElicitIntent | ElicitSlot",
            "slotToElicit": "string"
        },
        "intent": {
            "name": "string",
            "slots": {
                "string": {
                    "value": {
```

```
                        "interpretedValue": "string",
                        "originalValue": "string",
                        "resolvedValues": [ "string" ]
                    }
                 },
                "string": {
                    "shape": "List",
                    "value": {
                        "originalValue": "string",
                        "interpretedValue": "string",
                        "resolvedValues": [ "string" ]
                    },
                    "values": [
                        {
                            "shape": "Scalar",
                            "value": {
                                "originalValue": "string",
                                "interpretedValue": "string",
                                "resolvedValues": [ "string" ]
                            }
                        },
                        {
                            "shape": "Scalar",
                            "value": {
                                "originalValue": "string",
                                "interpretedValue": "string",
                                "resolvedValues": [ "string" ]
                            }
                        }
                    ]
                }
            },
            "kendraResponse": {
                // Only present when intent is KendraSearchIntent. For details, see
                // https://docs.aws.amazon.com/kendra/latest/dg/
API_Query.html#API_Query_ResponseSyntax
                },
            "state": "InProgress | ReadyForFulfillment | Fulfilled | Failed",
            "confirmationState": "Confirmed | Denied | None"
        },
        "originatingRequestId": "string",
        "sessionAttributes": {
            "string": "string"
        },
```

```
        "runtimeHints": {
            "slotHints": {
                "string": {
                    "string": {
                        "runtimeHintValues": [
                            {
                                "phrase": "string"
                            },
                            {
                                "phrase": "string"
                            }
                        ]
                    }
                }
            }
        }
    },
   "dialogEventLogs": [
        {
   // only for conditional
     "conditionalEvaluationResult":[
      // all the branches until true

     {
     "conditionalBranchName": "string",
     "expressionString": "string",
     "evaluatedExpression": "string",
     "evaluationResult": "true | false"
     }
    ],
  "dialogCodeHookInvocationLabel": "string",
  "response": "string",
  "nextStep": {
        "dialogAction": {
            "type": "Close | ConfirmIntent | Delegate | ElicitIntent | ElicitSlot",
            "slotToElicit": "string"
        },
       "intent": {
                          "name": "string",
           "slots": {
               }
        }
       }
    ]
```

```
    "interpretations": [
        {
            "interpretationSource": "Bedrock | Lex",
            "nluConfidence": "string",
            "intent": {
                "name": "string",
                "slots": {
                    "string": {
                        "value": {
                            "originalValue": "string",
                            "interpretedValue": "string",
                            "resolvedValues": [ "string" ]
                        }
                    },
                    "string": {
                        "shape": "List",
                        "value": {
                            "interpretedValue": "string",
                            "originalValue": "string",
                            "resolvedValues": [ "string" ]
                        },
                        "values": [
                            {
                                "shape": "Scalar",
                                "value": {
                                    "interpretedValue": "string",
                                    "originalValue": "string",
                                    "resolvedValues": [ "string" ]
                                }
                            },
                            {
                                "shape": "Scalar",
                                "value": {
                                    "interpretedValue": "string",
                                    "originalValue": "string",
                                    "resolvedValues": [ "string" ]
                                }

                            }
                        ]
                    }
                },
                "kendraResponse": {
```

```
                    // Only present when intent is KendraSearchIntent. For details, see

                    // https://docs.aws.amazon.com/kendra/latest/dg/
API_Query.html#API_Query_ResponseSyntax
                    },
                "state": "InProgress | ReadyForFulfillment | Fulfilled | Failed",
                "confirmationState": "Confirmed | Denied | None"
                },
            "sentimentResponse": {
                "sentiment": "string",
                "sentimentScore": {
                    "positive": "string",
                    "negative": "string",
                    "neutral": "string",
                    "mixed": "string"
                }
            }
        }
    ],
    "sessionId": "string",
    "inputTranscript": "string",
    "rawInputTranscript": "string",
    "transcriptions": [
        {
            "transcription": "string",
            "rawTranscription": "string",
            "transcriptionConfidence": "number",
            },
            "resolvedContext": {
                "intent": "string"
            },
            "resolvedSlots": {
                "string": {
                    "name": "slotName",
                    "shape": "List",
                    "value": {
                        "originalValue": "string",
                        "resolvedValues": [
                            "string"
                        ]
                    }
                }
            }
        }
```

```
    ],
    "missedUtterance": "bool",
    "requestId": "string",
    "timestamp": "string",
    "developerOverride": "bool",
    "inputMode": "DTMF | Speech | Text",
    "requestAttributes": {
        "string": "string"
    },
    "audioProperties": {
        "contentType": "string",
        "s3Path": "string",
        "duration": {
            "total": "integer",
            "voice": "integer",
            "silence": "integer"
        }
    },
    "bargeIn": "string",
    "responseReason": "string",
    "operationName": "string"
}
```

記錄項目的內容取決於交易的結果以及機器人和要求的組態。

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

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

訪問 Amazon S3 中的音頻日誌

Amazon Lex V2 會將您交談的音訊日誌存放在 S3 儲存貯體中。

您可以使用 Amazon S3 主控台或 API 存取音訊日誌。您可以在 Amazon Lex V2 主控台或DescribeBotAlias作業回應的conversationLogSettings欄位中看到音訊檔案的 S3 物件 key prefix。

使用 CloudWatch 指標監視交談記錄狀態

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

Amazon Lex V2 在AWS/Lex命名空間中為交談日誌提供四個指標：

- ConversationLogsAudioDeliverySuccess
- ConversationLogsAudioDeliveryFailure
- ConversationLogsTextDeliverySuccess
- ConversationLogsTextDeliveryFailure

成功指標顯示 Amazon Lex V2 已成功將您的音訊或文字日誌寫入目的地。

失敗指標顯示 Amazon Lex V2 無法將音訊或文字日誌傳遞到指定的目的地。通常，這是組態錯誤。當您的失敗指標高於零時，請檢查下列情況：

- 請確定 Amazon Lex V2 是 IAM 角色的受信任實體。
- 對於文字記錄，請確定記錄 CloudWatch 檔記錄群組存在。對於音訊記錄，確定 S3 儲存貯體存在。
- 確定 Amazon Lex V2 用來存取 CloudWatch 日誌日誌群組或 S3 儲存貯體的 IAM 角色具有日誌群組或儲存貯體的寫入權限。
- 確定 S3 儲存貯體與 Amazon Lex V2 機器人位於相同的區域，且屬於您的帳戶。

隱藏交談記錄中的插槽值

Amazon Lex V2 可讓您混淆或隱藏插槽的內容，以便看不到內容。若要保護擷取為槽值的敏感資料，您可以啟用槽混淆以遮罩這些記錄值。

當您選擇混淆插槽值時，Amazon Lex V2 會以交談日誌中插槽的名稱取代插槽的值。對於稱為 `full_name` 的槽，槽值將被混淆，如下所示：

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

如果語音包含括號字元 ({})，Amazon Lex V2 會使用兩個反斜線 (\\) 逸出括號字元。例如，文字 `{John Stiles}` 會被混淆，如下所示：

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

對話日誌中的槽值會被混淆。`RecognizeText`和`RecognizeUtterance`作業的回應中仍可使用插槽值，而且位置值可供您的驗證和履行 Lambda 函數使用。如果您是在提示或回應中使用槽值，則這些槽值不會在對話日誌中混淆。

在對話的第一個回合中，Amazon Lex V2 如果辨識出插槽和插槽值，就會混淆插槽值。如果沒有辨識任何插槽值，Amazon Lex V2 就不會混淆話語。

在第二個和之後的回合中，Amazon Lex V2 知道要引出的插槽，以及是否應該混淆插槽值。如果 Amazon Lex V2 辨識出插槽值，該值就會被混淆。如果 Amazon Lex V2 無法辨識某個值，則整個語音都會混淆。遺漏表達用語中的任何槽值都不會被混淆。

Amazon Lex V2 也不會混淆存放在請求或工作階段屬性中的插槽值。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 V2 不會混淆音頻中的插槽值。它的確會混淆音訊記錄中的槽值。

您可以使用主控台或使用 Amazon Lex V2 API 來選擇要混淆的插槽。在主控台中，於槽設定中選擇 Slot obfuscation (槽混淆)。如果您使用的是 API，請在呼叫CreateSlot或UpdateSlot作業`DEFAULT_OBFUSCATION`時將插槽的`obfuscationSetting`欄位設定為。

選擇性交談記錄擷取

選擇性交談記錄擷取可讓使用者選取如何使用即時交談中的文字和音訊資料擷取交談記錄。

若要啟用和擷取選擇性交談記錄擷取功能的輸出，您必須在 Amazon Lex V2 主控台中啟用該功能，並在 API 設定中啟用必要的工作階段屬性，以便從日誌擷取選取的輸出。

您可以選取下列選項來擷取選擇性交談記錄檔：

- 只有文字
- 只有音訊
- 文字和音訊

您可以擷取交談的特定部分，並選擇是否擷取交談記錄的音訊、文字或兩者。

> **Note**
>
> 選擇性交談記錄擷取僅適用於 Amazon Lex V2。

主題

管理選擇性交談記錄擷取

您可以使用 Lex 主控台啟用選擇性交談記錄擷取設定，並選擇要啟用選擇性交談記錄擷取的插槽。

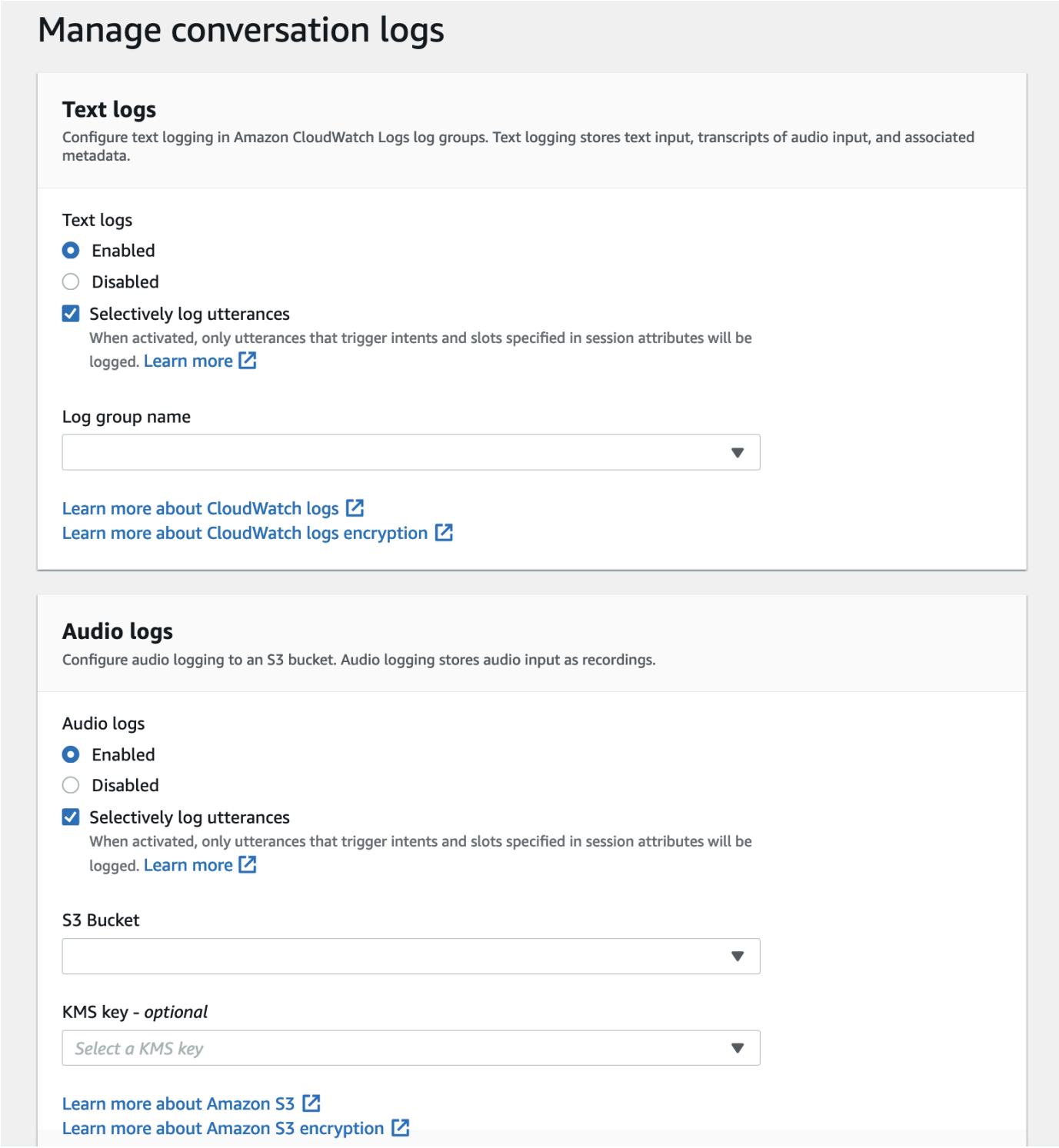
在 Amazon Lex V2 主控台中啟用選擇性交談記錄擷取：

1. 登入 AWS Management Console 並開啟 Amazon Lex V2 主控台，網址為 https://console.aws.amazon.com/lexv2/home。
2. 從左側面板中選擇機器人，然後選擇要啟用選擇性對話日誌捕獲的機器人。使用現有的機器人或建立新的機器人。
3. 在左側面板的「部署」區段下選擇所選機器人的別名。
4. 選擇機器人的別名，然後選擇管理對話記錄。

5. 在「管理交談記錄」面板中，對於「文字記錄」，請選取選項按鈕來選擇要啟用還是停用文字記錄。如果您針對文字記錄選擇 [啟用]，則必須輸入記錄群組名稱，或從下拉式功能表中選擇現有的記錄群組名稱。如果您要選擇性地記錄文字檔，請選取 [選擇性記錄語彙] 核取方塊。

   > **Note**
   >
   > 在建置時BotAlias間設定中選取交談記錄設定 (文字和/或音訊) 中的 [選擇性記錄語彙] 核取方塊，以啟用文字和/或音訊記錄。您必須設定 CloudWatch 日誌群組和 Amazon S3 儲存貯體，才能選取此選項。

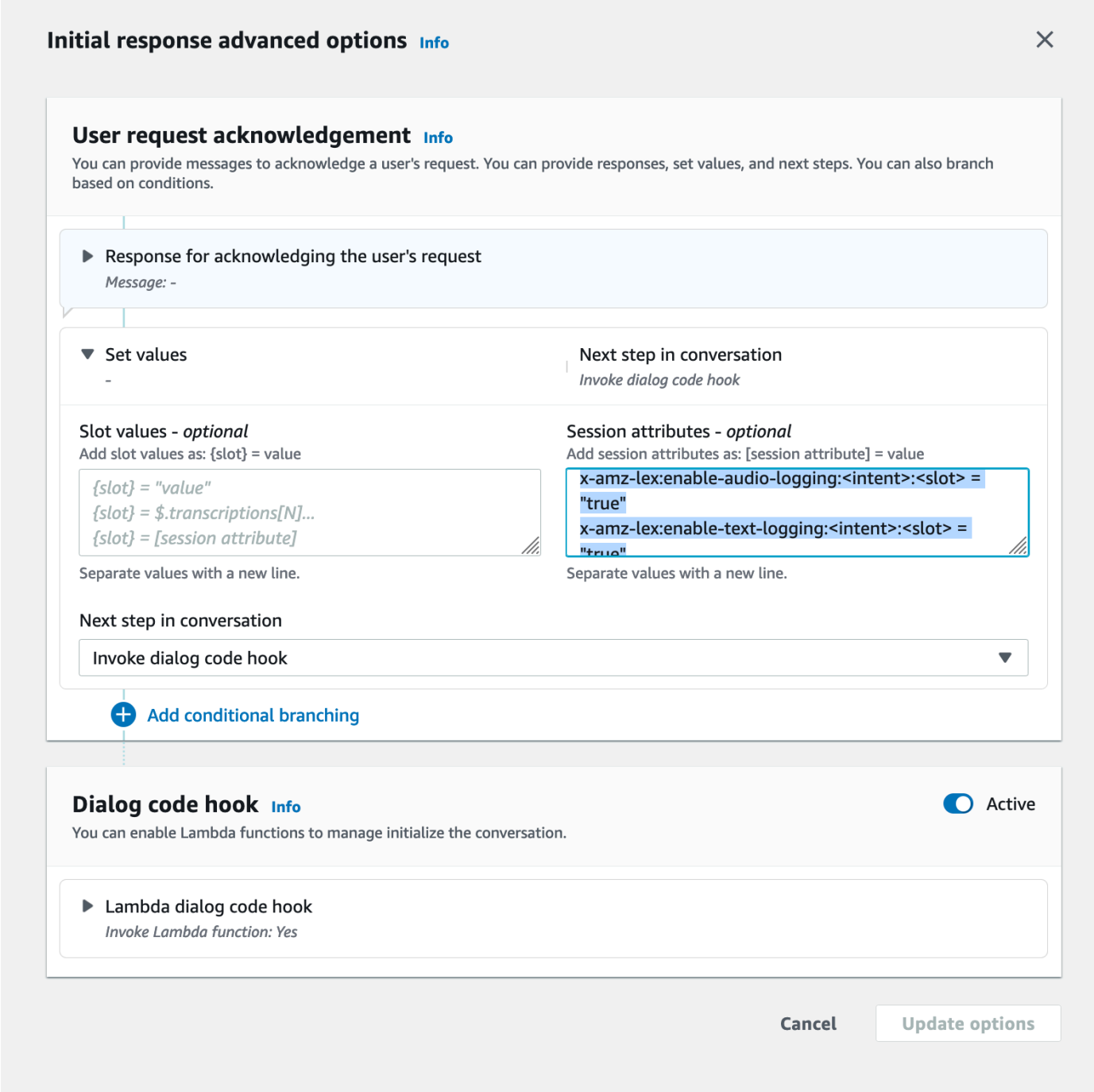
6. 在 [音訊記錄] 區段中，選取選項按鈕，選擇是否啟用或停用音訊記錄。如果您選擇為音訊日誌啟用，則需要指定 Amazon S3 儲存貯體位置和 (選用) 用於加密音訊資料的 KMS 金鑰。如果您要選擇性地記錄音訊檔案，請選取選擇性記錄語音的核取方塊。



7. 選取面板右下角的 [儲存]，以儲存您的選擇性交談記錄擷取設定。

在 Lex 主控台中啟用選擇性交談記錄擷取：

1. 轉到「意圖」，然後選擇「意圖」名稱，「初始響應」，「高級設置」，「設置值」，「會話屬性」。
2. 根據您要啟用選擇性交談記錄擷取的意圖和插槽，將下列屬性設定為：
   - `x-amz-lex:enable-audio-logging:intent:slot = "true"`
   - `x-amz-lex:enable-text-logging:intent:slot = "true"`



> **Note**
>
> 設定x-amz-lex:enable-audio-logging:*intent*:*slot* = "true"為擷取僅包含對話中特定插槽的話語。##若要記錄語音，工作階段屬性中至少必須允許一個運算式，並將啟用記錄旗標設定為。true##和#的值也可"*"以是。如果插槽和/或意圖值是"*"，則表示的任何插槽和/或意圖值都"*"將與其相符。與類似x-amz-lex:enable-audio-logging，x-amz-lex:enable-text-logging將使用名為的新工作階段屬性來控制文字記錄。

3. 選擇更新選項並構建機器人以包含更新的設置。

> **Note**
>
> 您的 IAM 角色必須具有存取權限，才能讓您將資料寫入 Amazon S3 儲存貯體，並使用 KMS 金鑰加密資料。Lex 將使用 Lex 許可更新您的 IAM 角色，以存取 CloudWatch 日誌日誌群組和選取的 Amazon S3 儲存貯體。

使用選擇性交談記錄擷取的準則：

您只能在 [交談記錄] 設定中啟用文字和/或音訊記錄時，才能啟用文字和/或音訊記錄的選擇性交談記錄擷取。藉由啟用文字和/或音訊記錄的選擇性交談記錄擷取功能，您可以停用交談中所有意圖和插槽的記錄。若要針對特定意圖和插槽產生文字和/或音訊記錄，您必須將這些意圖和插槽的文字或/和音訊選擇性交談記錄擷取工作階段屬性設定為「true」。

- 如果已啟用選擇性交談記錄擷取，且沒有前置詞為 x-amz-lex:enable-audio-logging 的工作階段屬性出現，則預設會停用所有語音的記錄功能。在以下情況下，這種情況也是如此：啟用文本日誌記錄 x-amz-lex。
- 如果 session 屬性中至少有一個表達式允許，話語日誌將專門為文本和/或音頻對話段存儲。
- 工作階段屬性中定義的文字和/或音訊的選擇性交談記錄擷取的組態只有在機器人別名的交談記錄設定中啟用文字和/或音訊的選擇性交談記錄擷取時才會生效；否則，工作階段屬性將會被忽略。
- 啟用選擇性交談記錄擷取時，未使用工作階段屬性啟用記錄的任何位置值、解釋和轉譯，都會在產生的文字記錄中模糊化。 SessionState
- 產生音訊和/或文字記錄的決定是透過比對機器人所產生的插槽與選擇性交談記錄擷取工作階段屬性進行評估，除了意圖引導回合，使用者可以提供插槽值以及意圖引出。在意圖引出回合中，當前回合填充的插槽與選擇性交談日誌捕獲會話屬性匹配。

- 被視為填充的槽是從車削結束時的作業階段狀態衍生出來的。因此，對話方塊 Codehook Lambda 對工作階段狀態中的插槽所做的任何變更都會影響選擇性交談記錄擷取的行為。
- 在意圖引出轉向中，如果用戶給出了多個插槽值，則只有在文本/音頻會話屬性允許對此回合中填充的所有插槽進行記錄時才會生成文本和/或音頻日誌。
- 建議的作業方法是在工作階段開始時設定選擇性交談記錄擷取工作階段屬性，並避免在工作階段期間對其進行修改。
- 如果有任何插槽包含敏感資料，您應始終啟用插槽模糊化。

選擇性交談記錄擷取範例

以下是選擇性交談記錄擷取的商業使用案例範例。

使用案例：

一家金融科技公司利用 Amazon Lex V2 機器人來支援他們的 IVR 系統，讓使用者能夠支付帳單。為了滿足合規性和審計要求，他們必須保留用戶提供的授權同意的錄音。但是，啟用一般音訊記錄是不可行的，因為這會使它們不相容，因為無法混淆音訊記錄檔中的敏感插槽 CardNumber，例如 CVV 和其他資訊。相反，它們可以為音訊記錄啟用選擇性交談記錄擷取，並將 session 屬性設定為僅產生具有授權同意的話語的音訊記錄。

BotAlias 設定：

- 文字記錄已啟用：真
- 文字記錄檔選擇性記錄已啟用：false
- 音頻日誌啟用：真
- 音訊記錄選擇性記錄已啟用：true

階段作業屬性：

```
x-amz-lex:enable-audio-logging:PayBill:AuthorizationConsent = "true"
```

樣本對話：

- 用戶（音頻輸入）：「我想用賬單號碼 35XU68 支付賬單。」
- 博特：「什麼是美元的到期金額？」
- 使用者（音訊輸入）：「235。」
- Bot：「你的信用卡號碼是多少？」

- 使用者 (音訊輸入)：「9239829722200348。」
- 機器人：「您正在使用以 0348 結尾的信用卡號碼支付 235 美元。請說「我授權支付 235 美元。」
- 用戶（音頻輸入）：「我授權支付 235 美元。」
- Bot：「您的帳單已支付。」

交談記錄輸出：

在此情況下，將為所有回合產生文字記錄。但是，只有在PayBill意圖中的AuthorizationConsent插槽被引發時，才會在特定回合錄製音頻日誌，並且不會為任何其他轉彎生成音頻日誌。

監控操作指標

Amazon AWS CloudTrail 是 CloudWatch 與 Amazon Lex V2 整合的兩種 AWS 服務，可協助您監控使用者與機器人的互動。使用這些服務來記錄動作、傳送近乎即時的資料，以及在符合條件時設定通知和自動化動作。

主題

- 使用 Amazon 測量操作指標 CloudWatch
- 檢視事件 AWS CloudTrail

使用 Amazon 測量操作指標 CloudWatch

您可以使用來監控 Amazon Lex V2 CloudWatch，這會收集原始資料並將其處理為可讀且近乎即時的指標。這些統計資料會保留 15 個月，以便您存取歷史資訊，並更清楚 Web 應用程式或服務的執行效能。您也可以設定留意特定閾值的警示，當滿足這些閾值時傳送通知或採取動作。如需詳細資訊，請參閱 Amazon CloudWatch 使用者指南。

Amazon Lex V2 服務會在AWS/Lex命名空間中報告下列指標。

指標	描述
AssistedSlotResolutionModelAccessDeniedErrorCount	Amazon Lex V2 被拒絕訪問 Amazon 基岩的次數 RecognizeUtterance 和StartConversation 作業的有效維度： • BotId,, BotAliasId, LocaleId, 操作, InputMode, ModelType, 模型 • BotId,, BotVersion, LocaleId, 操作, InputMode, ModelType, 模型

指標	描述
	有效的尺寸RecognizeText ： • BotId, BotAliasId,, LocaleId, 操作 ModelType, 模型 • BotId, BotVersion,, LocaleId, 操作 ModelType, 模型 單位：計數
AssistedSlotResolutionModelInvocationCount	Amazon 基岩被調用的次數。 RecognizeUtterance 和StartConversation 作業的有效維度： • BotId,, BotAliasId, LocaleId, 操作, InputMode, ModelType, 模型 • BotId,, BotVersion, LocaleId, 操作, InputMode, ModelType, 模型 有效的尺寸RecognizeText ： • BotId, BotAliasId,, LocaleId, 操作 ModelType, 模型 • BotId, BotVersion,, LocaleId, 操作 ModelType, 模型 單位：計數
AssistedSlotResolutionModelSystemErrorCount	調用 Amazon 基岩時 5xx 發生的次數。 RecognizeUtterance 和StartConversation 作業的有效維度： • BotId,, BotAliasId, LocaleId, 操作, InputMode, ModelType, 模型 • BotId,, BotVersion, LocaleId, 操作, InputMode, ModelType, 模型 有效的尺寸RecognizeText ： • BotId, BotAliasId,, LocaleId, 操作 ModelType, 模型 • BotId, BotVersion,, LocaleId, 操作 ModelType, 模型 單位：計數

指標	描述
AssistedSlotResolutionModelThrottlingErrorCount	Amazon Lex 被亞馬遜基岩限制的次數。 RecognizeUtterance 和StartConversation 作業的有效維度： • BotId,, BotAliasId, LocaleId, 操作, InputMode, ModelType, 模型 • BotId,, BotVersion, LocaleId, 操作, InputMode, ModelType, 模型 有效的尺寸RecognizeText ： • BotId, BotAliasId,, LocaleId, 操作 ModelType, 模型 • BotId, BotVersion,, LocaleId, 操作 ModelType, 模型 單位：計數
AssistedSlotResolutionResolvedSlotCount	Amazon 基岩返回插槽值的次數。 RecognizeUtterance 和StartConversation 作業的有效維度： • BotId,, BotAliasId, LocaleId, 操作, InputMode, ModelType, 模型 • BotId,, BotVersion, LocaleId, 操作, InputMode, ModelType, 模型 有效的尺寸RecognizeText ： • BotId, BotAliasId,, LocaleId, 操作 ModelType, 模型 • BotId, BotVersion,, LocaleId, 操作 ModelType, 模型 單位：計數
KendraIndexAccessError	Amazon Lex V2 無法存取您的 Amazon Kendra 索引的次數。 • 操作 BotId、 BotAliasId、 LocaleId 單位：計數

指標	描述
KendraLatency	Amazon Kendra 回應來自的請求所AMAZON.KendraSearchIntent 花費的時間量。 有效維度： • 操作 BotId、 BotVersion、 LocaleId • 操作 BotId、 BotAliasId、 LocaleId 單位：毫秒
KendraSuccess	Amazon Lex V2 無法存取您的 Amazon Kendra 索引的次數。 有效維度： • 操作 BotId、 BotVersion、 LocaleId • 操作 BotId、 BotAliasId、 LocaleId 單位：計數
KendraSystemErrors	Amazon Lex V2 無法查詢亞 Amazon Kendra 索引的次數。 有效維度： • 操作 BotId、 BotAliasId、 InputMode、 LocaleId 單位：計數
KendraThrottledEvents	Amazon Kendra 限制請求的次數。AMAZON.KendraSearchIntent 有效維度： • 操作、 BotId、 BotAliasId、 InputMode。 LocaleId 單位：計數

指標	描述
RuntimeConcurrency	指定時間範圍內同時連線的數目。 RuntimeConcurrency 報告為StatisticSet . RecognizeUtterance 或StartConversation 作業的有效維度： • 操作 BotId、 BotVersion、 InputMode、 LocaleId • 操作 BotId、 BotAliasId、 InputMode、 LocaleId 其他操作的有效尺寸： • 操作 BotId、 BotVersion、 LocaleId • 操作 BotId、 BotAliasId、 LocaleId 單位：計數
RuntimeInvalidLambdaResponses	指定期間內無效 AWS Lambda 回應的數目。 有效維度： • 操作 BotId、 BotAliasId、 InputMode、 LocaleId 單位：計數
RuntimeLambdaErrors	指定期間內的 Lambda 執行階段錯誤數目。 有效維度： • 操作 BotId、 BotAliasId、 InputMode、 LocaleId 單位：計數

指標	描述
RuntimePollyErrors	指定時間段內無效的 Amazon Polly 回應數目。 有效維度： • 操作 BotId、 BotAliasId、 InputMode、 LocaleId 單位：計數
RuntimeRequestCount	指定期間內的執行階段要求數目。 RecognizeUtterance 和StartConversation 作業的有效維度： • 操作 BotId、 BotVersion、 InputMode、 LocaleId • 操作 BotId、 BotAliasId、 InputMode、 LocaleId 其他操作的有效尺寸： • 操作 BotId、 BotVersion、 LocaleId • 操作 BotId、 BotAliasId、 LocaleId 單位：計數
RuntimeRequestLength	與 Amazon Lex V2 機器人的交談總長度。僅適用於StartConversation操作。 有效維度： • BotAlias識別碼、 BotId、 LocaleId、作業 • BotId、 BotAliasId、 LocaleId、作業 單位：毫秒

指標	描述
RuntimeSuccessfulRequestLatency **Important** 此量度是RuntimeSuccessfulRequestLatency ，而不是RuntimeSuccessfulRequestLatency 。	提出要求到回應之間，成功要求的延遲時間。 RecognizeUtterance 和StartConversation 作業的有效維度： • 操作 BotId、 BotVersion、 InputMode、 LocaleId • 操作 BotId、 BotAliasId、 InputMode、 LocaleId 其他操作的有效尺寸： • 操作 BotId、 BotVersion、 LocaleId • 操作 BotId、 BotAliasId、 LocaleId 單位：毫秒
RuntimeSystemErrors	指定期間內的系統錯誤數量。系統錯誤的回應碼範圍是 500 到 599。 RecognizeUtterance 和StartConversation 作業的有效維度： • 操作 BotId、 BotVersion、 InputMode、 LocaleId • 操作 BotId、 BotAliasId、 InputMode、 LocaleId 其他操作的有效尺寸： • 操作 BotId、 BotVersion、 LocaleId • 操作 BotId、 BotAliasId、 LocaleId 單位：計數

指標	描述
RuntimeThrottledEvents	節流事件的數目。當 Amazon Lex V2 收到的請求數量超過您帳戶設定的每秒交易限制時，會對事件進行調節。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。若要請求增加，請參閱 AWS 服務限制。 RecognizeUtterance 和StartConversation 作業的有效維度： • 操作 BotId、 BotVersion、 InputMode、 LocaleId • 操作 BotId、 BotAliasId、 InputMode、 LocaleId 其他操作的有效尺寸： • 操作 BotId、 BotVersion、 LocaleId • 操作 BotId、 BotAliasId、 LocaleId 單位：計數
RuntimeUserErrors	指定期間內的使用者錯誤數量。使用者錯誤的回應碼範圍是 400 到 499。 RecognizeUtterance 和StartConversation 作業的有效維度： • 操作 BotId、 BotVersion、 InputMode、 LocaleId • 操作 BotId、 BotAliasId、 InputMode、 LocaleId 其他操作的有效尺寸： • 操作 BotId、 BotVersion、 LocaleId • 操作 BotId、 BotAliasId、 LocaleId 單位：計數

Amazon Lex V2 指標支援下列維度。

維度	描述
Operation	產生項目的 Amazon Lex V2 作業名稱 DeleteSession — RecognizeText RecognizeUtterance StartConversation GetSession PutSession 、、、、、、。
BotId	機器人的字母數字唯一識別碼。
BotAliasId	機器人別名的英數字元唯一識別碼。
BotVersion	機器人的數字版本。
InputMode	機器人的輸入類型 — 語音、文字或 DTMF。
LocaleId	機器人地區設定的識別碼，例如 en-US 或 fr-CA。
Model	表示 Amazon 基岩大語言模型的模型 ID。
ModelType	表示從 Amazon 基岩叫用的大型語言模型類型。

檢視事件 AWS CloudTrail

Amazon Lex V2 與這項服務整合在一起 AWS CloudTrail，可提供 Amazon Lex V2 中使用者、角色或 AWS 服務所採取的動作記錄的服務。 CloudTrail 擷取 Amazon Lex V2 的 API 呼叫做為事件。擷取的呼叫包括來自 Amazon Lex V2 主控台的呼叫，以及對 Amazon Lex V2 API 作業的程式碼呼叫。如果您建立追蹤，您可以啟用持續交付 CloudTrail 事件到 Amazon S3 儲存貯體，包括 Amazon Lex V2 的事件。如果您未設定追蹤，您仍然可以在 [事件歷程記錄] 中檢視 CloudTrail主控台中最近的事件。使用收集的資訊 CloudTrail，您可以判斷向 Amazon Lex V2 發出的請求、提出請求的來源 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，請參閱使AWS CloudTrail 用者指南。

Amazon Lex V2 信息 CloudTrail

CloudTrail 在您創建 AWS 帳戶時，您的帳戶已啟用。當 Amazon Lex V2 中發生活動時，該活動會與事件歷史記錄中的其他 AWS 服務 CloudTrail事件一起記錄在事件中。您可以在帳戶中查看，搜索和下載最近的事 AWS 件。如需詳細資訊，請參閱使用 CloudTrail 事件歷程記錄檢視事件。

如需 AWS 帳戶中持續的事件記錄 (包括 Amazon Lex V2 的事件)，請建立追蹤。追蹤可 CloudTrail 將日誌檔交付到 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。追蹤記錄來自 AWS 分區中所有區域的事件，並將日誌檔傳送到您指定的 Amazon S3 儲存貯體。此外，您還可以設定其他 AWS 服務，以進一步分析 CloudTrail 記錄中收集的事件資料並採取行動。如需詳細資訊，請參閱下列內容：

- 建立追蹤的概觀
- CloudTrail 支援的服務與整合
- 設定 CloudTrail 的 Amazon SNS 通知
- 從多個區域接收 CloudTrail 日誌文件並從多個帳戶接收 CloudTrail 日誌文件

Amazon Lex V2 支援記錄模型建置 API V2 中列出的所有動作。

每一筆事件或日誌專案都會包含產生請求者的資訊。身分資訊可協助您判斷下列事項：

- 請求是以根使用者登入資料還是 AWS Identity and Access Management IAM 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暫時安全憑證。
- 請求是否由其他 AWS 服務提出。

如需詳細資訊，請參閱CloudTrail 使 userIdentity 元素。

了解 Amazon Lex V2 日誌檔項目

追蹤是一種組態，可讓事件以日誌檔的形式傳遞到您指定的 Amazon S3 儲存貯體。 CloudTrail 記錄檔包含一或多個記錄項目。事件代表來自任何來源的單一請求，包括有關請求的操作，動作的日期和時間，請求參數等信息。 CloudTrail 日誌文件不是公共 API 調用的有序堆棧跟踪，因此它們不會以任何特定順序顯示。

下列範例顯示示範「CreateBot別名」動作的 CloudTrail 記錄項目。

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "ID of caller:temporary credentials",
        "arn": "arn:aws:sts::111122223333:assumed-role/role name/role ARN",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
```

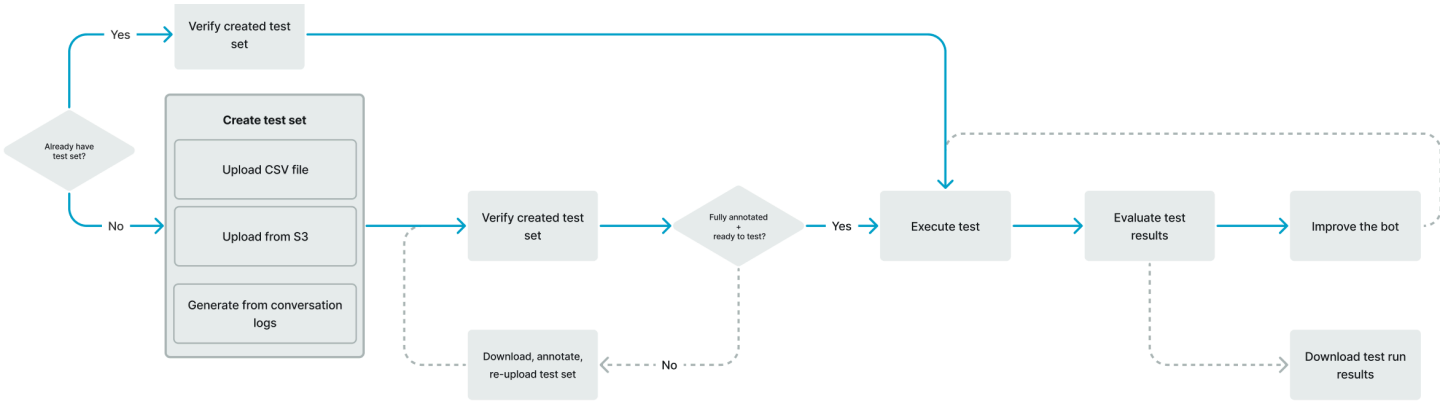
```
            "sessionIssuer": {
                "type": "Role",
                "principalId": "ID of caller",
                "arn": "arn:aws:iam::111122223333:role/role name",
                "accountId": "111122223333",
                "userName": "role name"
            },
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "creation date"
            }
        }
    },
    "eventTime": "event timestamp",
    "eventSource": "lex.amazonaws.com",
    "eventName": "CreateBotAlias",
    "awsRegion": "Region",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "user agent",
    "requestParameters": {
        "botAliasLocaleSettingsMap": {
            "en_US": {
                "enabled": true
            }
        },
        "botId": "bot ID",
        "botAliasName": "bot aliase name",
        "botVersion": "1"
    },
    "responseElements": {
        "botAliasLocaleSettingsMap": {
            "en_US": {
                "enabled": true
            }
        },
        "botAliasId": "bot alias ID",
        "botAliasName": "bot alias name",
        "botId": "bot ID",
        "botVersion": "1",
        "creationDateTime": creation timestamp
    },
    "requestID": "unique request ID",
    "eventID": "unique event ID",
```

```
    "readOnly": false,
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

使用測試工作台評估機器人效能

若要改善機器人效能，您可以大規模評估機器人的效能。測試評估的結果會顯示在簡單的表格和圖表中。

您可以使用「測試工作台」來建立使用現有轉錄資料的參考測試集。您可以在部署之前測試機器人以評估效能，並大規模檢視測試結果明細。



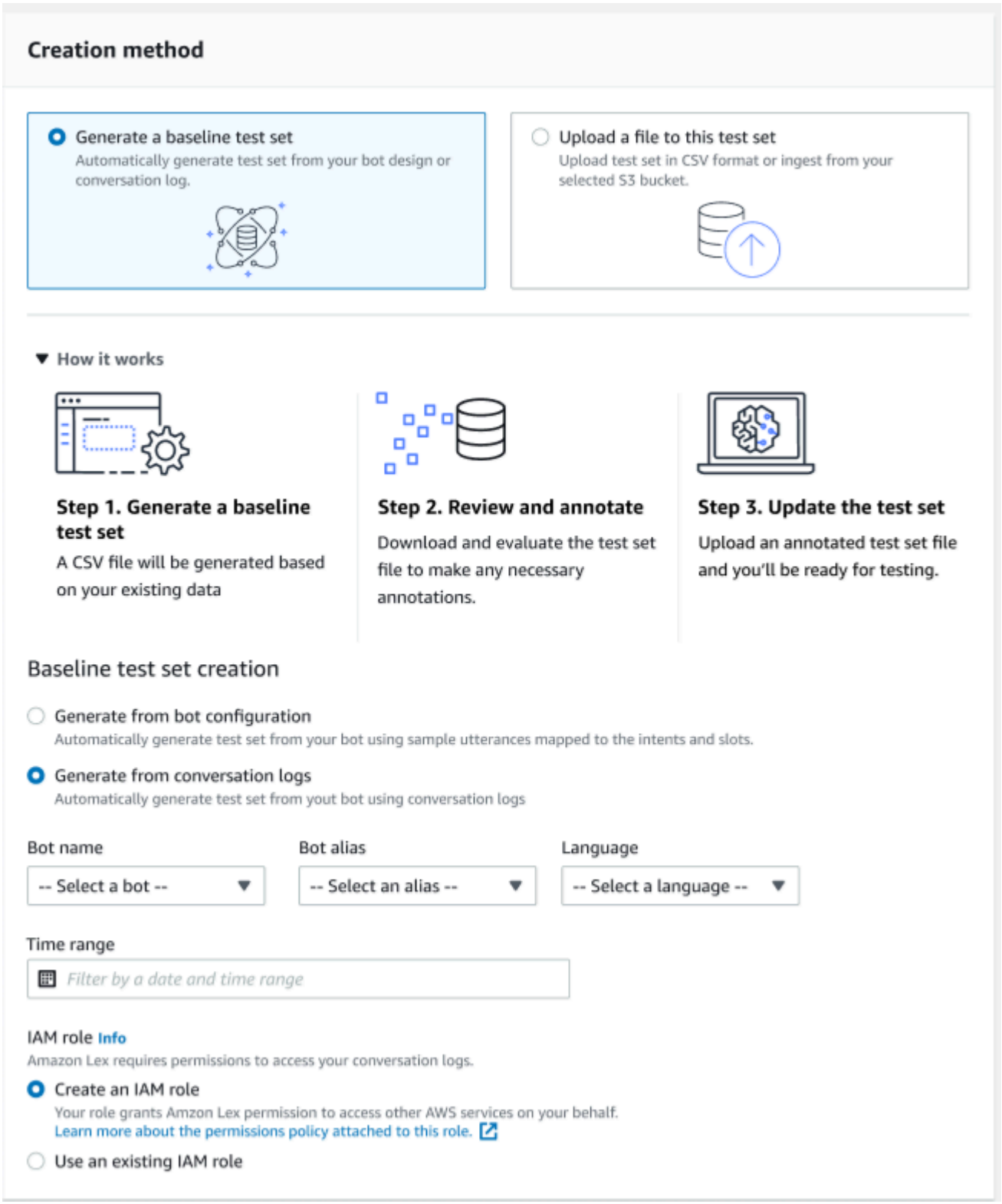
使用者可以使用測試工作台來建立機器人的基準效能。這涵蓋了單一輸入或交談形式的話語的意圖和插槽效能。成功載入測試集後，您可以對現有的預生產或生產機器人執行測試集。測試工作台可協助您找出改善插槽填充與意圖分類的機會。

主題

- 產生測試集
- 管理測試集
- 執行測試
- 測試集覆蓋範圍
- 檢視測試結果
- 測試結果詳情

產生測試集

您可以建立測試集來評估機器人的效能。上傳 CSV 檔案格式的測試集，或從交談記錄產生測試集，以產生測試集。測試集可以包含音頻或文本輸入。



如果測試集產生驗證錯誤，請移除測試集並以其他測試集資料清單取代，或使用試算表編輯程式編輯 CSV 檔案中的資料。

若要建立測試集：

1. 登錄到 AWS Management Console 並打開 Amazon Lex 控制台 https://console.aws.amazon.com/lex/.
2. 從左側面板中選擇測試工作台。
3. 從「測試工作台」下的選項中選取「測試集」。
4. 選取主控台上的 [建立測試集] 按鈕。
5. 在 [詳細資料] 中，輸入測試集名稱和選用說明。
6. 選取產生基準測試集。
7. 選取 [從交談記錄檔產生]。
8. 從下拉式功能表中選取「機器人名稱」、「機器人別名」和「語言」。
9. 如果您要從交談記錄產生基準測試，請視需要選擇 [時間範圍] 和 [IAM 角色]。您可以使用基本 Amazon Lex V2 許可建立角色，或使用現有的角色。
10. 為您正在創建的測試集選擇音頻或文本的模式。注意：「測試工作台」最多可以導入 50k 的文本文件，最多可導入 5 小時的音頻。
11. 選取 Amazon S3 位置來存放測試結果，並新增選用的 KMS 金鑰來加密輸出記錄。
12. 選取建立。

若要上傳 CSV 檔案格式的現有測試集，或更新測試集：

1. 從左側面板中選擇測試工作台。
2. 從「測試工作台」下的選項中選取「測試集」。
3. 在主控台上選取 [將檔案上傳至此測試集]。
4. 選擇從 Amazon S3 儲存貯體上傳或從您的電腦上傳。注意：您可以上傳從範本建立的 CSV 檔案。按一下 CSV 範本以下載包含範本的 zip 檔案。
5. 選擇「建立具有基本 Amazon Lex 權限的角色」或「使用角色 ARN 的現有角色」。
6. 為您正在創建的測試集選擇音頻或文本的模式。注意：「測試工作台」最多可以導入 50k 的文本文件，最多可導入 5 小時的音頻。
7. 選取 Amazon S3 位置來存放測試結果，並新增選用的 KMS 金鑰來加密輸出記錄。
8. 選取建立。

如果操作成功，確認消息將指示測試集已準備好測試，並且狀態將顯示「準備測試」。

建立成功測試集的秘訣

- 您可以在主控台中為測試工作台建立 IAM 角色，也可以設定 IAM 角色 step-by-step。有關詳情，請參閱為測試工作台建立 IAM 角色。
- 在執行測試之前，請使用「驗證差異」按鈕來驗證測試集和機器人定義是否有任何不一致。如果測試集中使用的意圖和插槽命名約定與機器人一致，請繼續執行測試。如果識別出任何異常，請修訂測試集、更新測試集，然後選擇「驗證差異」。再次重複此序列，直到沒有發現任何不一致之處，然後執行測試。
- 「測試工作台」可以在「預期輸出插槽」欄中使用不同的插槽值格式進行測試。對於任何內建插槽，您可以選擇使用者輸入中提供的值 (例如，日期 = 明天)，或提供其絕對解析值 (例如，日期 = 2023-03-21)。如需有關內建插槽及其絕對值的詳細資訊，請參閱內建插槽。
- 為確保「預期的輸出插槽」欄中的一致性和可讀性，請遵循 SlotValue「SlotName =」(例如，AppointmentType = 清理) 的慣例，並在等號前後加上空格。
- 如果機器人包含複合插槽，則在「預期輸出插槽」中定義插槽名稱的子插槽，並以句點分隔（例如，「Car.Color」）。沒有其他語法和標點符號將起作用。
- 如果機器人包含多值插槽，則「預期輸出插槽」中會提供多個插槽值，並以逗號分隔（「FlowerType = 玫瑰，百合花」）。沒有其他語法和標點符號將起作用。
- 請確定測試集是從有效的交談記錄建立的。
- 槽：槽值將位於 CSV 格式的意圖欄之後的同一欄中。
- 來自使用者回合的 DTMF 輸入會解譯為預期的轉錄，而不會列出 Amazon S3 位置。

在測試集中創建測試用例

測試工作台結果取決於機器人定義及其對應的測試集。您可以使用機器人定義中的資訊產生測試集，以精確找出需要改進的區域。創建一個測試數據集，其中包含您懷疑（或知道）將具有挑戰性的示例，考慮到當前的機器人設計以及您對客戶對話的了解來正確解釋。

根據生產機器人的知識，定期檢閱您的意圖。繼續新增並調整機器人的範例語調和位置值。請考慮使用可用的選項 (例如執行階段提示) 來改善插槽解析度。機器人的設計和開發是一個連續循環的迭代過程。

以下是一些最佳化測試集的其他提示：

- 選擇測試集中具有常用意圖和插槽的最常見使用案例。

- 探索客戶可以參考您的意圖和位置的不同方式。這可以包括語句，問題和命令形式的用戶輸入，這些形式從最小到擴展的長度不同。
- 包括具有不同數量插槽的用戶輸入。
- 包括機器人支援的自訂插槽值的常用同義詞或縮寫（例如，「根管道」、「管道」或「RC」）。
- 包括內置插槽值的變化（例如，「明天」，「盡快」或「第二天」）。
- 通過收集可能被誤解的用戶輸入（例如，「墨水」，「腳踝」或「錨」）來檢查機器人語言模式的穩健性。

從 CSV 檔案建立測試集

您可以使用 CSV 試算表編輯器直接輸入值，從 Amazon Lex V2 主控台提供的 CSV 檔案範本建立測試集。測試集是一個逗號分隔值 (CSV) 檔案，由單一使用者的話語和多回合對話組成，記錄在下列欄位中：

- 行號 — 此欄是一個遞增計數器，可追蹤要測試的填滿列總數。
- 交談 # — 此欄會追蹤對話中的回合次數。對於單一輸入，此欄可以保留空白，填入「-」或「N/A」。對於對話，對話中的每個回合將被分配相同的對話號碼。
- 來源 — 此欄位設定為「使用者」或「代理程式」。對於單個輸入，它將始終設置為「用戶」。
- 輸入 — 此欄包含使用者說話或機器人提示。
- 預期的輸出色彩比對方式 — 此欄會擷取輸入中符合的意圖。
- 意圖預期的輸出插槽 1 — 此欄會擷取使用者輸入中產生的第一個插槽。對於使用者輸入中的每個插槽，測試集應包含一個名為「預期輸出插槽 X」的欄。

具有單個輸入的測試集示例：

行編號	對話 #	來源	輸入	預期的輸出比對	預期輸出插槽 1	預期輸出插槽 2
1		使用者	明天預約清潔	MakeAppointment	AppointmentType = 清潔	日期 = 明天
2	N/A	使用者	4 月 15 日預約清潔預約	MakeAppointment	AppointmentType = 清潔	日期 = 4 月 15 日

行編號	對話 #	來源	輸入	預期的輸出比對	預期輸出插槽 1	預期輸出插槽 2
3	N/A	使用者	12 月第一次預約	MakeAppointment	日期 = 十二月第一	
4	N/A	使用者	預約清潔	MakeAppointment	AppointmentType = 清潔	
1		使用者	你能幫我預約嗎？	MakeAppointment		

帶有對話的測試集示例

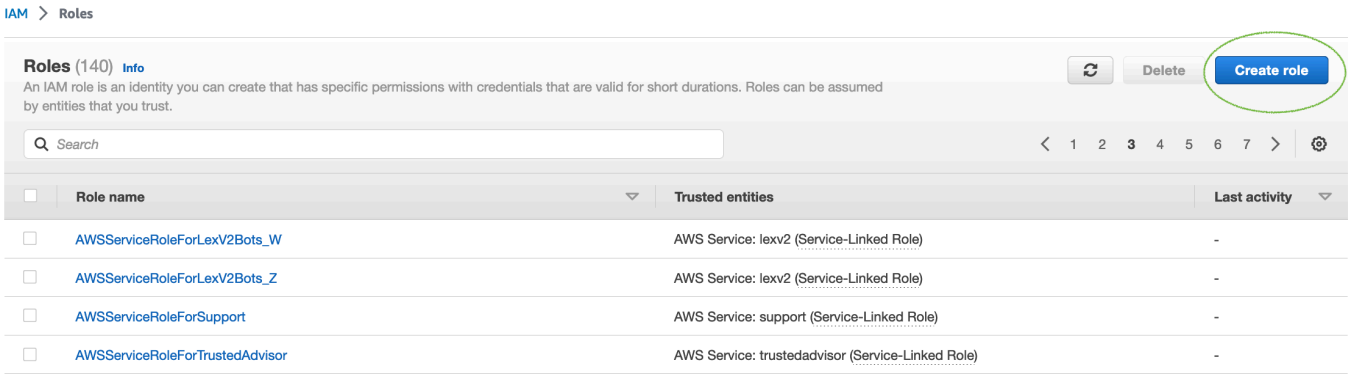
行編號	對話 #	來源	輸入	預期的輸出比對	預期輸出插槽 1	預期輸出插槽 2	預期輸出插槽 3
1	1	使用者	預約	MakeAppointment			
2	1	代理程式	您想安排什麼類型的預約？	MakeAppointment			
3	1	使用者	清潔	MakeAppointment	AppointmentType = 清潔		
4	1	代理程式	我應該何時安排您的預約？	MakeAppointment			
5	1	使用者	tomorrow	MakeAppointment		日期 = 明天	

行編號	對話 #	來源	輸入	預期的輸出比對	預期輸出插槽 1	預期輸出插槽 2	預期輸出插槽 3
6	2	使用者	今天預訂根管任命	MakeAppointment	AppointmentType = 根管	日期 = 今天	
7	2	代理程式	我應該在什麼時候安排你的預約？	MakeAppointment			
8	2	使用者	上午十一時	MakeAppointment			時間 = 上午十一時

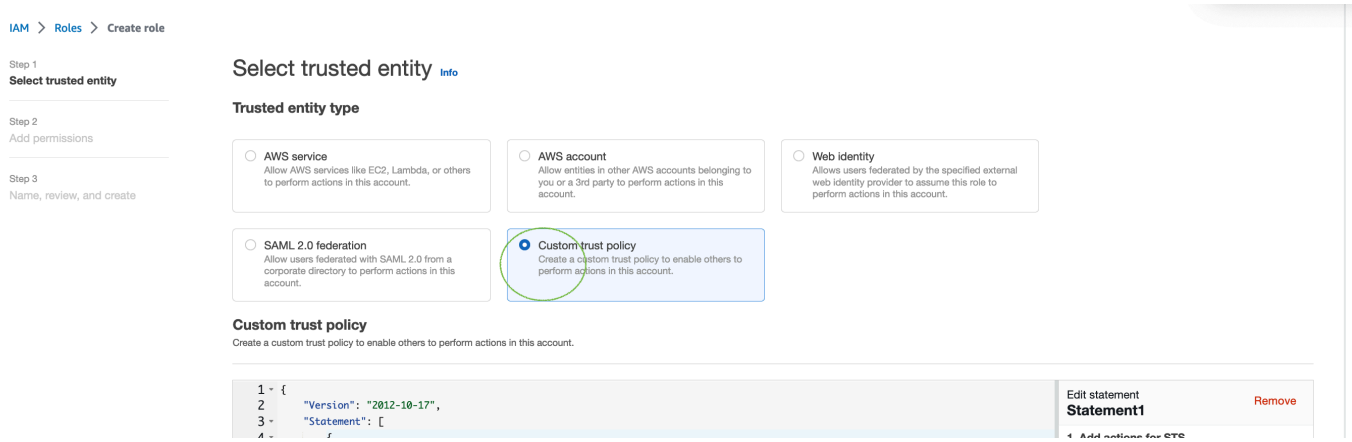
為測試工作台建立 IAM 角色

若要為測試工作台建立 IAM 角色

1. 按照建立 IAM 使用者中的步驟建立可用於存取測試工作台主控台的 IAM 使用者。
2. 選取 Create role (建立角色) 按鈕。



3. 選取 [自訂信任原則] 選項。



4. 在下方輸入信任原則，然後按 [下一步]。

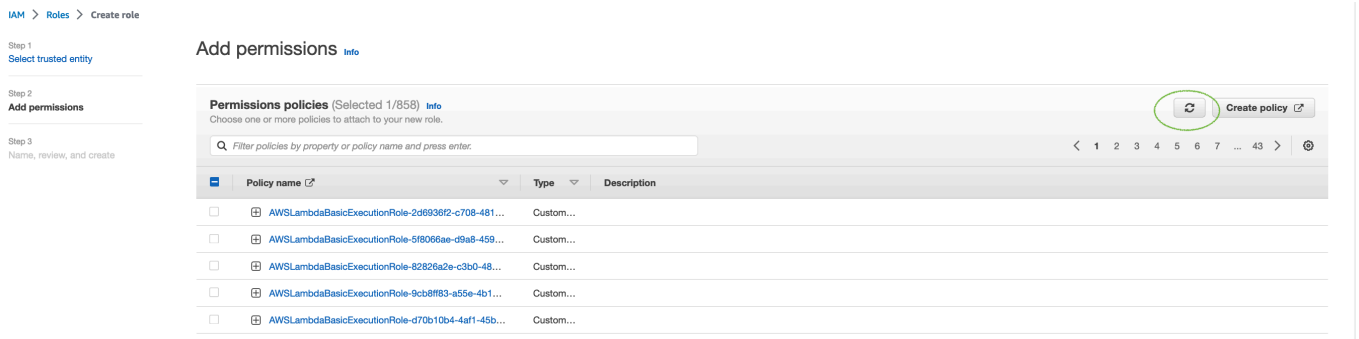
```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "sid4",
      "Effect": "Allow",
      "Principal": {
        "Service": "lexv2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

5. 選取 [建立原則] 按鈕。
6. 一個新的選項卡將在您的瀏覽器中打開，您可以在其中輸入以下策略，然後單擊下一步：標籤按鈕。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:*"
            ],
            "Resource": "*"
        },
        {
```

```
            "Effect": "Allow",
            "Action": [
                "logs:FilterLogEvents"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "lex:*"
            ],
            "Resource": "*"
        }
    ]
}
```

7. 輸入策略名稱，例如 'LexTestWorkbenchPolicy'，然後按一下「建立策略」。
8. 按一下 [重新整理] 按鈕，返回瀏覽器中的上一個索引標籤和重新整理原則清單，如下所示。



9. 輸入您在第 6 個步驟中使用的策略名稱，在策略清單中搜尋，然後選擇策略。
10. 選取 [下一步] 按鈕。
11. 輸入角色名稱，然後按一下「建立角色」按鈕。
12. 在測試工作台的 Amazon Lex V2 主控台中出現提示時，選擇您的新 IAM 角色。

管理測試集

您可以從測試集視窗下載、更新和刪除測試集。或者，您可以使用可用的測試集清單來編輯或手動註解測試集檔案。然後，由於錯誤或其他輸入問題，請再次上傳以重試驗證。

若要從測試集記錄下載測試集檔案：

1. 從測試集清單中選取測試集的名稱。

2. 在測試集記錄視窗中，選取 [測試輸入] 區段中畫面右側的 [下載] 按鈕。
3. 如果視窗頂端有關於測試集的任何驗證錯誤詳細資訊，請選取 [下載] 按鈕。該文件將保存到您的「下載」文件夾中。您可以從測試集 CSV 檔案中的錯誤訊息修正測試集中的驗證錯誤。尋找驗證步驟中識別的錯誤、修正或移除該行，然後上傳檔案以重試驗證步驟。
4. 如果您成功下載測試集，則會出現綠色橫幅訊息。

若要從測試集清單下載測試集：

1. 從測試集清單中，選取您要下載之測試集項目旁邊的選項按鈕。
2. 從右上角的「操作」菜單中，選擇「下載」。
3. 綠色橫幅訊息會指出您是否已成功下載測試集。該文件將保存到您的「下載」文件夾中。

檢視測試驗證錯誤

您可以更正報告驗證錯誤的測試集。當測試集尚未準備好進行測試時，會產生這些驗證錯誤。測試工作台可以顯示測試集輸入 CSV 檔案中哪些必要資料行沒有預期格式的值。

若要檢視測試驗證錯誤：

1. 從測試集清單中，選取報告您要檢視之驗證錯誤狀態之測試集的名稱。測試集的名稱是使用中的連結，可帶您前往有關測試集的詳細資訊。
2. 測試集記錄會在畫面頂端顯示驗證錯誤詳細資料。選擇「檢視明細」以查看「驗證錯誤」的報表。
3. 從錯誤報告視窗中，檢閱「行號」和「錯誤類型」，以查看發生錯誤的位置。如需冗長的錯誤清單，您可以選擇下載錯誤報告。
4. 將測試集輸入 CSV 檔案中列出的錯誤與原始測試檔案進行比較，以更正任何問題，然後再次上傳測試集。

下表列出了帶有案例的輸入 CSV 驗證錯誤消息。

案例	錯誤訊息	備註
測試集檔案大小超過	「測試集」檔案大小超過 200 MB。請提供較小的檔案，然後再試一次。	

案例	錯誤訊息	備註
測試集超過最大記錄	輸入文件的記錄超過了 200,000 支持的最大數量。	
上傳空白的測試集	匯入的測試集是空的。提供非空白的測試集，然後再次嘗試您的請求。	
空欄標題名稱	列標題行：在列號 5 中找到空列名稱。	
無法辨識的欄標頭名稱	列標題行：無法識別列號 2 中的列名「虛擬」。	
重複的列標題名稱	列標題行：找到多個相同或等效的列「S3 音頻鏈接」和「S3 音頻鏈接」。移除或重新命名其中一欄。	
多值欄名稱超過限制	列標題行：「預期輸出插槽」的列數超過支持的最大計數：6。請移除「預期輸出插槽」的部分欄，然後再試一次。	多值欄支援的最大欄數為 6。
文本或音頻相關的列標題不存在	找不到文字或音訊對話的欄。對於文字對話，請使用 {'文字輸入'} 欄。對於音訊交談，請使用 {'S3 音訊連結 '、' 預期轉錄 '} 欄。	音訊必要欄：{'S3 音訊連結 '、' 預期轉錄 '} 文字必要欄：{' 文字輸入 '}
存在與文本和音頻相關的列標題	找到文本和音頻對話的列。您可以使用 {'文本輸入'} 列進行文本對話，或使用 {'S3 音頻鏈接 ' , ' 預期的轉錄 '} 列進行音頻對話。	音訊必要欄：{'S3 音訊連結 '、' 預期轉錄 '} 文字必要欄：{' 文字輸入 '}
缺少強制列	找不到必要欄 ["預期的輸出色彩比對方式"]。	必要欄：{"行 #」、「來源」、「預期的輸出色彩比對方式"}

案例	錯誤訊息	備註
在沒有標題的列中找到數據	在列號 8 中找到列號 6 的資料，但對應的欄沒有欄標題。	
找不到必要欄的資料	Row=12：找不到必要欄的值：{"來源」、「預期的輸出色彩比對方式"}	
找到重複的交談 ID	在行號 39 處以前的對話中看到了對話號碼 '19'。」 確保兩個對話沒有提供相同的對話號碼，您可以通過確保對話號碼的所有行都組合在一起來做到這一點。	
提供的交談 ID 無效	在「交談編號」欄中找到無效的值「測試對話」。對於使用者列，此欄的值必須是數值或 N/A (亦即不適用)。	
為行號提供的非數值	在「行號」欄中找到非數值「測試行」。它的值必須是數字。	
在代理程式列中找不到交談識別碼	找不到「交談 #」欄的值。必須為代理程式列提供此代理程式。	
在代理程式列中找到非數字交談 ID	在「交談編號」欄中找到非數值「測試對話」。代理程式資料列的值必須是數值。	
無效的 S3 位置	提供的值 '儲存區/資料夾' 無效。有效的格式為 S3：//<bucketName>/<keyName>。	
S3 儲存貯體名稱無效	提供了無效的 s3 存儲桶名稱「測試桶」。檢查值區名稱。	

案例	錯誤訊息	備註
S3 音頻位置是文件夾	提供的音訊位置 'S3://儲存貯體/資料夾 '無效。它指向 S3 文件夾。	
意圖名稱無效	意圖 '意圖 @name' 中存在無效字元。檢查意圖名稱。	正則表達式檢查：^ ([0-9A-Z] [_-] ？ ） +$
插槽名稱無效	插槽「@Name」插槽中出現無效的字元。檢查插槽名稱。	正則表達式：^ ([0-9A-Z] [_-] ？ ） +$ 它不應該以點（ 。 ）開頭或結束
為父插槽提供的插槽值	插槽值被提供給子插槽「地址。城市」以及父插槽「地址」。應該僅針對子槽提供值。	CST 中的父插槽不應該有插槽值
上下文名稱中的無效字符	前後關聯名稱 '環境 @1' 中出現無效字元。檢查前後關聯名稱。	正則表達式:^ ([A-ZA-Z] _?) +$
無效的插槽拼字樣式	提供的值「測試」無效。確保它們都是大寫字母。有效值為 ["預設"、"SpellBy字母"、"SpellBy單字"]。	支援的值 [預設值」、「SpellBy字母」、「SpellBy單字」
參與者或來源必須是代理人或使用者	提供的值 'bot' 無效。有效值為 ["代理程式"、"使用者"]。	支持的枚舉：「代理」，「用戶」
行號不應為十進制	提供的值 '10.1' 無效。它應該是一個沒有任何分數的有效數字。	
交談號碼不應為十進位	提供的值 '10.1' 無效。它應該是一個沒有任何分數的有效數字。	

案例	錯誤訊息	備註
行號應在範圍內	提供無效的值 '92233720 368547758071'。它應該大於或等於 1，並且小於或等於 9223372036854775807。	
條形列只接受布爾值	提供的值「測試」無效。它應該是一個有效的布爾值，如「真」或「假」。或者，可以使用「是」和「否」。	可能的值：「真」、「真」、「T」、「是」、「是」、「假」、「F」、「否」、「否」、「N」、「
預期的插槽，會話屬性，請求屬性應該用等於（=）分隔	值 '插槽:插槽值' 沒有 '='。這樣的值應以 '<key>=<value>' 格式的鍵值對提供。	例如：slotName = 插槽類型
預期的插槽，會話屬性，請求屬性應該有鍵值對	'= 插槽值' 之前沒有 '=' 之前的鍵。這樣的值應以 '<key>=<value>' 格式的鍵值對提供。	例如：slotName = 插槽類型
結尾引號無效	在「萊尼的漢堡」中找到不正確的引用。它以引號字符 `"` 開頭，但不以相同的引號字符結束。	例如：`「萊尼的漢堡」，KFC`
中間引號無效	在「萊尼的」漢堡，KFC` 中發現不正確的引用。它包含引號字符 `"`其內容裡面。包含單引號的值應該用雙引號包裝，反之亦然。	正確的例子：`「萊尼的漢堡」，KFC`
必要的引號	`key = Lenny 的漢堡 `包含單引號或雙引號，但沒有被包裹在引號內。包含單引號的值應包裝在雙引號內，反之亦然。	

案例	錯誤訊息	備註
重複鍵在列中重複	關鍵字 `key1` 在兩欄中重複：「階段作業屬性 3」和「階段作業屬性 1」。	
運行時提示格式無效	無效的金鑰BookFlight。汽車。「`提供給執行階段提示。對於運行時提示，密鑰應該是格式<intentName>。 <slotName>。	如果 '。' 必須出現在鍵的中間，則無法從此類密鑰中提取意圖名稱和插槽名稱。這種不正確格式的例子：BookFlight「，」。 BookFlight. 汽車」,「BookFlight. 汽車。」
運行時提示鍵中的意圖名稱無效	在執行階段提示中找到無效的意圖「意圖 @name」。檢查意圖名稱。	正則表達式檢查：^ ([0-9A-Z] [_-]？ ） +$
運行時提示鍵中的插槽名稱無效	在執行階段提示的「插槽 @Name」中找到無效的插槽名稱。檢查插槽名稱。	正則表達式：^ ([0-9A-Z] [_-]？ ） +$ 它不應該以點（。）開頭或結束

刪除測試集

您可以輕鬆地從測試集清單中刪除測試集。

若要刪除測試集：

1. 從左側菜單轉到測試集列表以查看測試集列表。
2. 從測試集清單中，選取您要刪除的測試集。
3. 轉到右上角的「操作」下拉菜單，然後選擇「刪除」。
4. 會出現一則訊息，確認已刪除測試集。

編輯測試集詳細資料

您可以在測試集清單中編輯測試集名稱和詳細資料。您可以稍後新增或更新名稱或詳細資料。但是，在使用機器人或轉錄數據運行測試之前，您必須更新測試集。

若要編輯測試集詳細資訊：

1. 從左側菜單轉到測試集列表以查看測試集列表。
2. 從測試集清單中，選取要編輯之測試集的核取方塊。
3. 轉到右上角的「操作」下拉菜單，然後選擇 「編輯詳細信息」。
4. 會出現一則訊息，確認測試集已成功編輯。

更新測試集

您可以更新、更正、修改或刪除測試集中的項目，以最佳化基準結果，或更正測試集中可能發生的其他錯誤

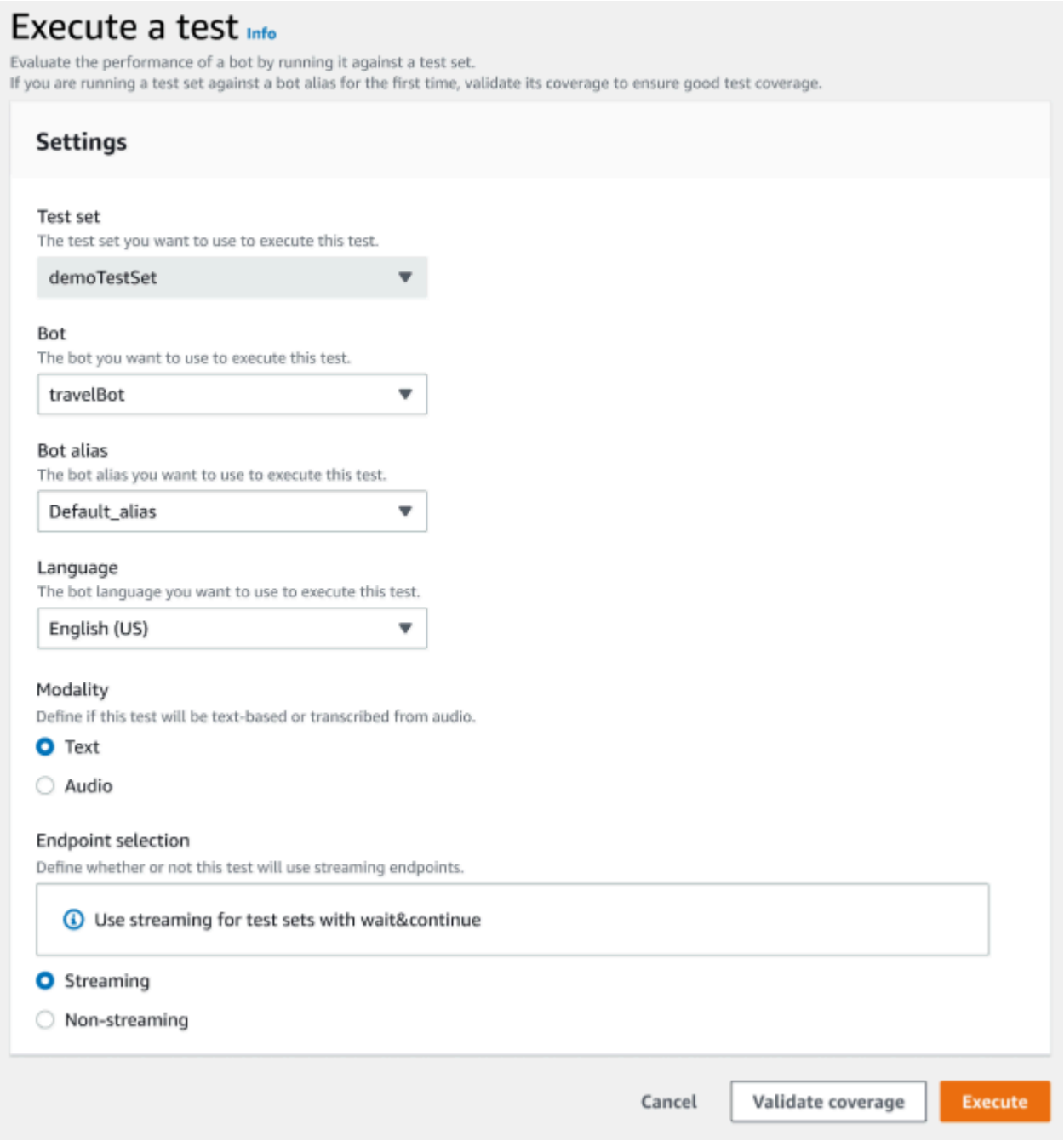
您可以下載測試集並修正驗證錯誤，然後再上傳更正的測試集。請參閱檢視測試驗證錯誤。

若要更新測試集：

1. 從測試集記錄中，選擇右上角的「更新測試集」按鈕。
2. 選擇要從 Amazon S3 帳戶上傳的檔案，或從電腦上傳 CSV 測試檔案。注意：更新測試集將覆蓋現有的數據。
3. 選取「更新」按鈕。
4. 會出現一則訊息，確認測試集已成功更新。注意：此操作可能需要幾分鐘的時間，具體取決於測試集的複雜性和大小。
5. 會出現一則訊息，確認測試集已成功更新，且「狀態」會顯示「準備測試」。

執行測試

若要執行測試集，您必須選擇適當的機器人來針對測試集執行測試。您可以從「測試集」下的下拉式功能表中選擇 AWS 帳戶中的機器人。此操作將根據已驗證的測試數據測試您選擇的機器人，以根據測試集中的基準數據報告性能指標。



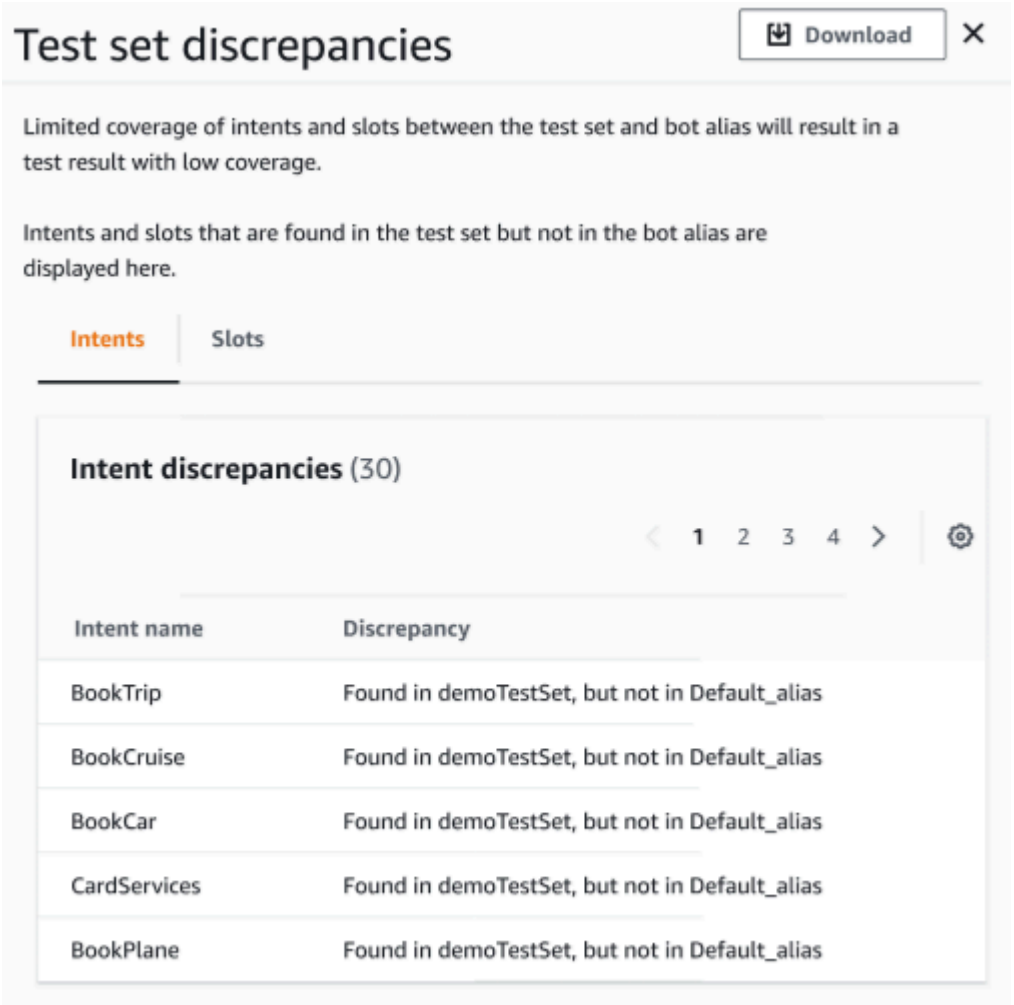
若要在「測試工作台」中執行測試

1. 在「測試集記錄」頁面中，選擇「執行測試」。
2. 選取您要在測試中使用的測試集。
3. 從「機器人」下拉式功能表中選取要在測試中使用的機器人名稱。
4. 從機器人別名下拉式功能表中選擇機器人別名 (如果適用)。
5. 從「語言」選項中，選擇英文版本。

6. 選取「文字」或「音訊」做為「模式」類型。
7. 選擇您的 Amazon S3 位置。 (僅限音訊)
8. 為您的機器人選取您的端點選項。 (僅限串流)
9. 選取「驗證涵蓋範圍」按鈕，以確認您的測試已準備好執行。如果驗證步驟中存在任何錯誤，請檢閱先前的參數並進行更正。
10. 選取執行以執行測試。
11. 會出現一則訊息，確認測試已成功執行。

測試集覆蓋範圍

測試集與機器人之間的意圖和時段涵蓋範圍有限，可能會導致預期的效能衡量。我們建議您在執行測試之前檢閱測試集涵蓋範圍。



若要複查驗證範圍

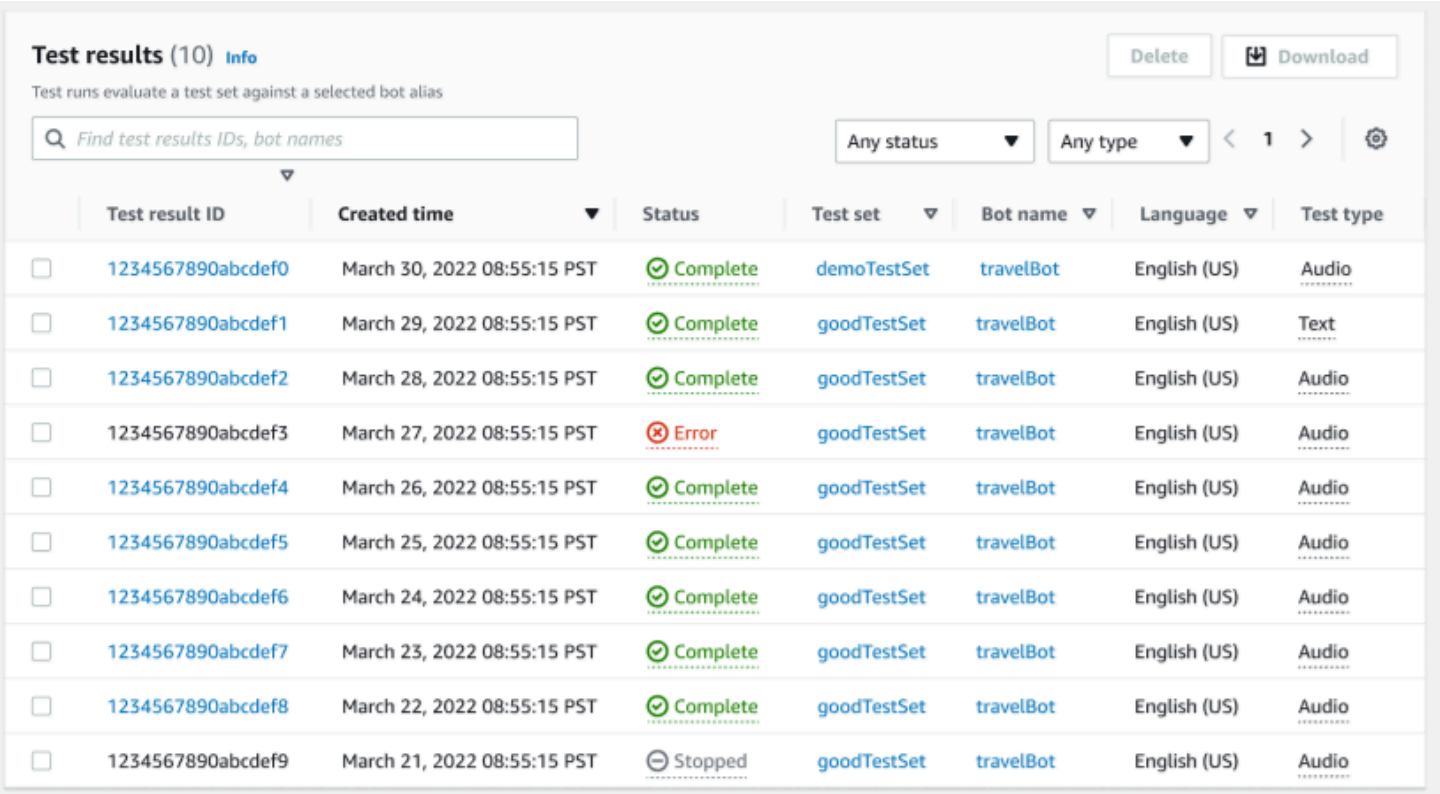
1. 在測試集記錄中，選擇「驗證涵蓋範圍」按鈕。
2. 該消息表明它正在驗證測試集和所選機器人之間的覆蓋範圍。
3. 操作完成後，該消息指示覆蓋驗證成功。
4. 選擇視窗底部的「檢視明細」按鈕。
5. 透過為每個選項選擇索引標籤，檢視意圖和插槽的測試集差異。您可以選擇 [下載] 按鈕，將這些資料下載為 CSV 格式。
6. 檢閱測試集資料、機器人意圖和位置的驗證結果。識別問題並在您的機器人測試集架構中進行變更，以改善結果。對 CSV 檔案進行變更後，上傳編輯過的測試集和機器人以執行測試。注意：驗證涵蓋範圍針對測試集運行，而不是針對機器人。不會涵蓋機器人中但不存在於測試集中的意圖。

檢視測試結果

解譯測試工作台中的測試結果，以判斷您的機器人與客戶之間的對話可能會失敗的位置，或要求客戶多次嘗試實現意圖。

透過在測試結果中找到這些問題，您可以使用與即時機器人轉錄值更一致的不同訓練資料或話語來改善意圖效能，從而最佳化機器人的效能。

您可以取得有效能差異的意圖和插槽的詳細檢視。一旦您確定了具有差異的意圖或插槽，您可以進一步向下鑽研並檢閱話語和對話流程。



若要檢視測試結果：

1. 從左側功能表移至測試集清單，選取 [測試工作台] 下的 [測試結果] 選項。注意：如果測試結果成功，則表示「完成狀態」。
2. 選取您要檢閱之測試結果的測試結果 ID。

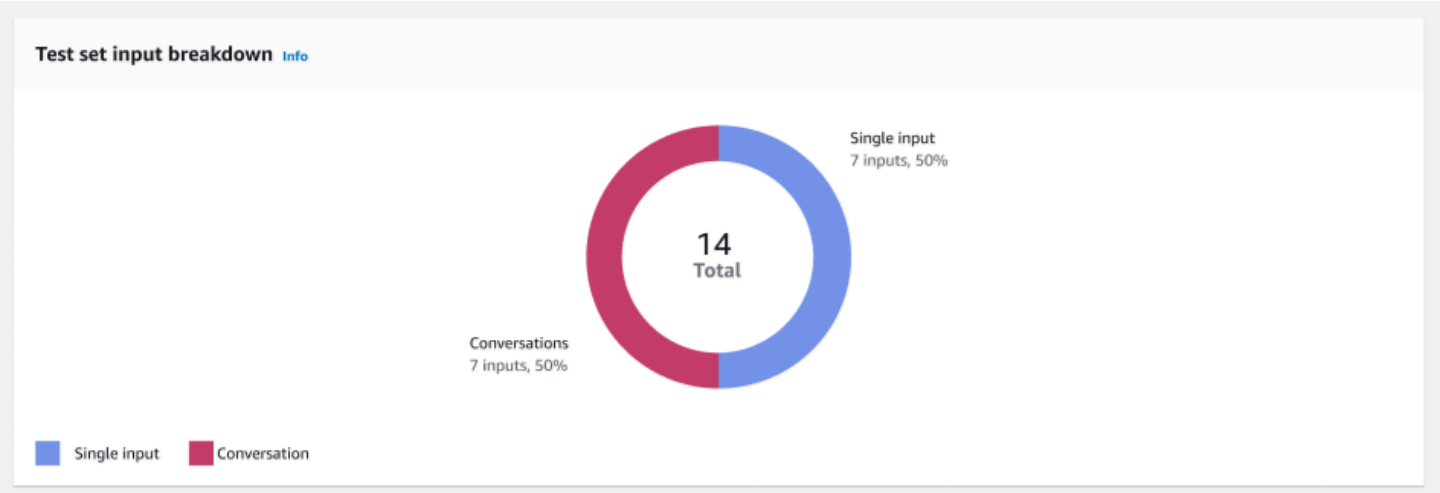
測試結果詳情

測試結果會顯示測試集詳細資料、使用的意圖，以及使用的插槽。它還提供了整體測試集輸入細分，包括整體結果，交談結果，意圖和插槽結果。

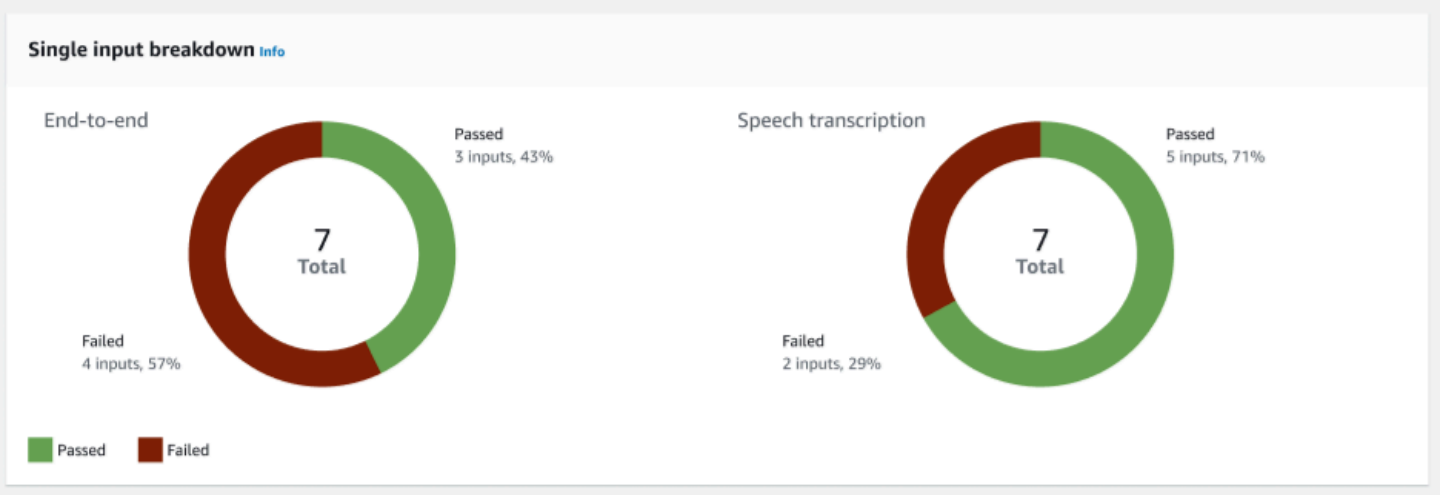
測試結果包括所有與測試相關的信息，例如：

- 測試詳細信息元
- 整體結果
- 對話結果
- 意圖和插槽結果
- 詳細結果

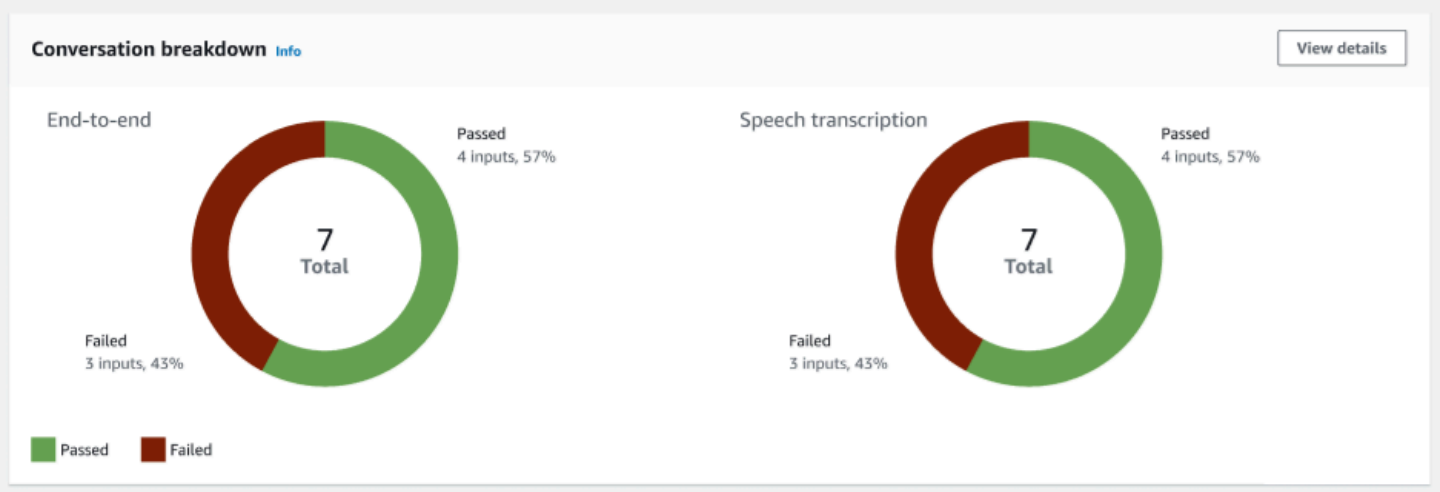
整體結果標籤：



測試集輸入細分 — 此圖表顯示測試集中交談數和單一輸入語音的明細。

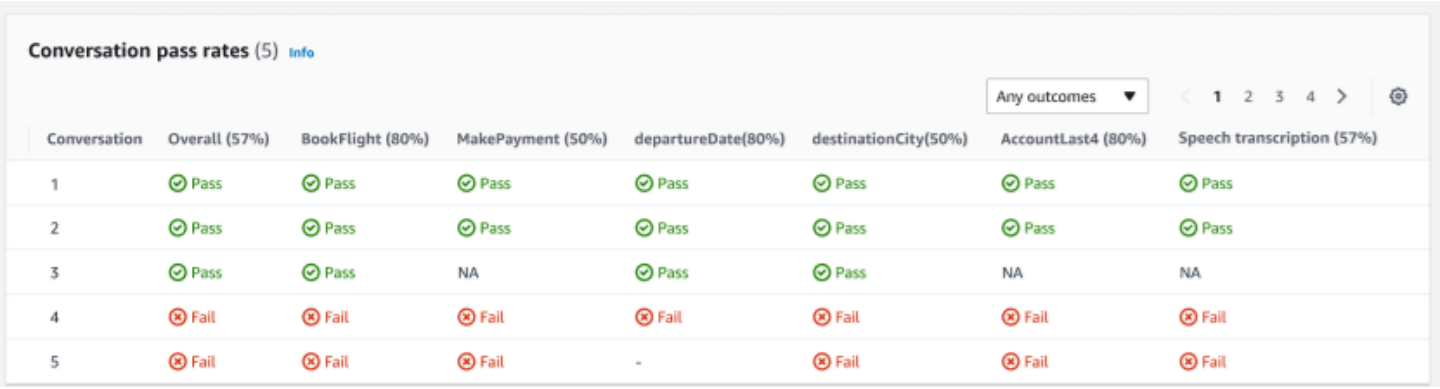


單一輸入劃分 — 顯示兩個圖表，其中包括 end-to-end 交談和語音轉錄。通過和失敗的輸入數會在每個圖表上顯示。注意：只有音訊測試集才會顯示語音轉錄圖表。



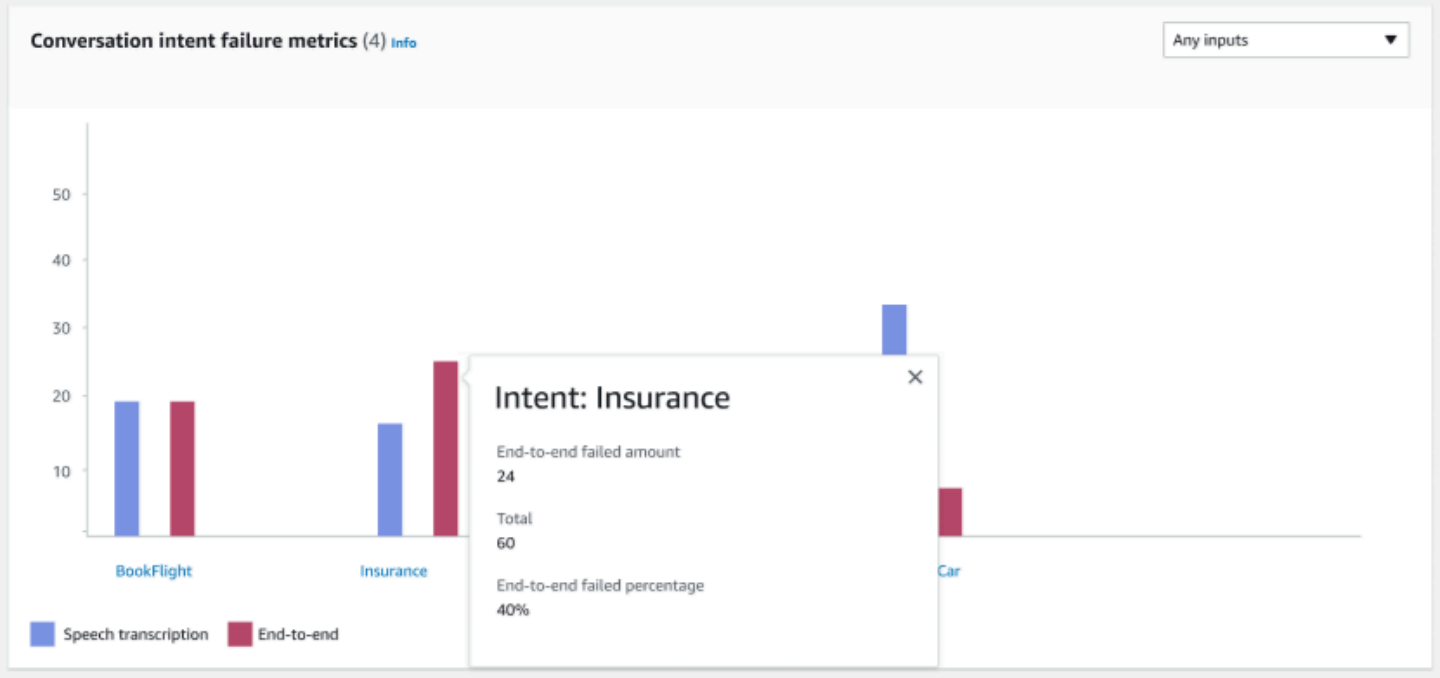
對話劃分 — 顯示兩個圖表，其中包括交 end-to-end 談和語音轉錄。通過和失敗的輸入數會在每個圖表上顯示。注意：只有音訊測試集才會顯示語音轉錄圖表。

交談結果標籤：



Conversation	Overall (57%)	BookFlight (80%)	MakePayment (50%)	departureDate(80%)	destinationCity(50%)	AccountLast4 (80%)	Speech transcription (57%)
1	Pass	Pass	Pass	Pass	Pass	Pass	Pass
2	Pass	Pass	Pass	Pass	Pass	Pass	Pass
3	Pass	Pass	NA	Pass	Pass	NA	NA
4	Fail	Fail	Fail	Fail	Fail	Fail	Fail
5	Fail	Fail	Fail	-	Fail	Fail	Fail

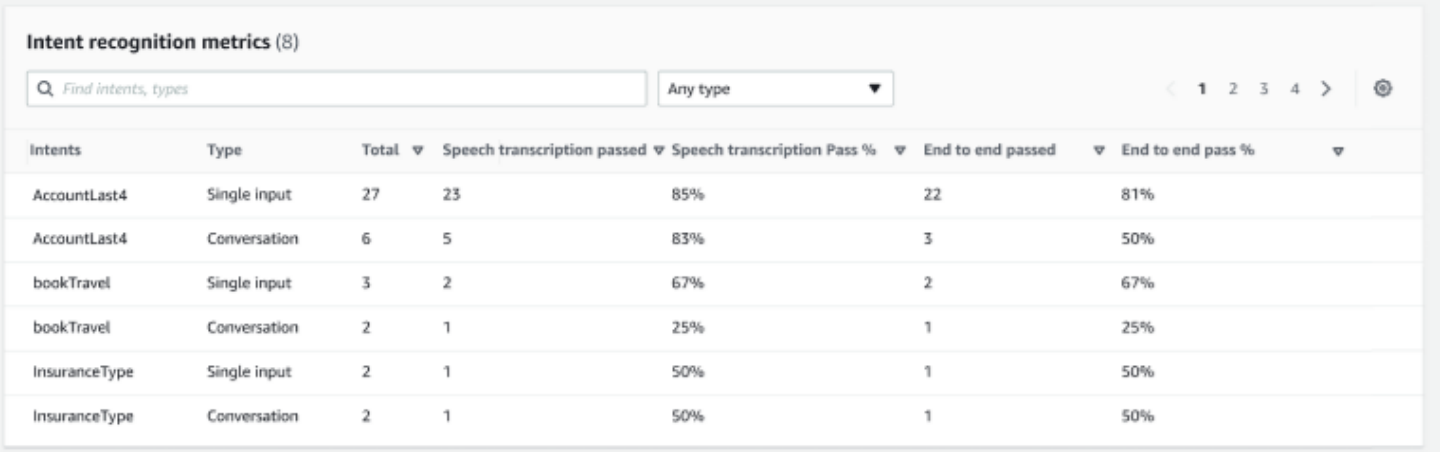
交談通過率 — 交談通過率表用於查看測試集中每個對話中使用的意圖和插槽。您可以檢視哪個意圖或插槽失敗，以及每個意圖和插槽的通過百分比，以視覺化交談失敗的位置。



交談意圖失敗度量 — 此度量顯示測試集中前 5 個效能最差的意圖。此面板會根據機器人的對話記錄或轉錄，顯示意圖成功或失敗的百分比或數目。一個成功的意圖並不意味著整個對話是成功的。這些度量僅適用於意圖的值，無論哪個意圖之前或之後。

交談插槽失敗度量 — 此度量顯示測試集中效能最差的 5 個插槽。指出意圖中每個插槽的成功率。條形圖顯示意圖中每個插槽的語音轉錄和 end-to-end 對話。

意圖和插槽結果標籤：

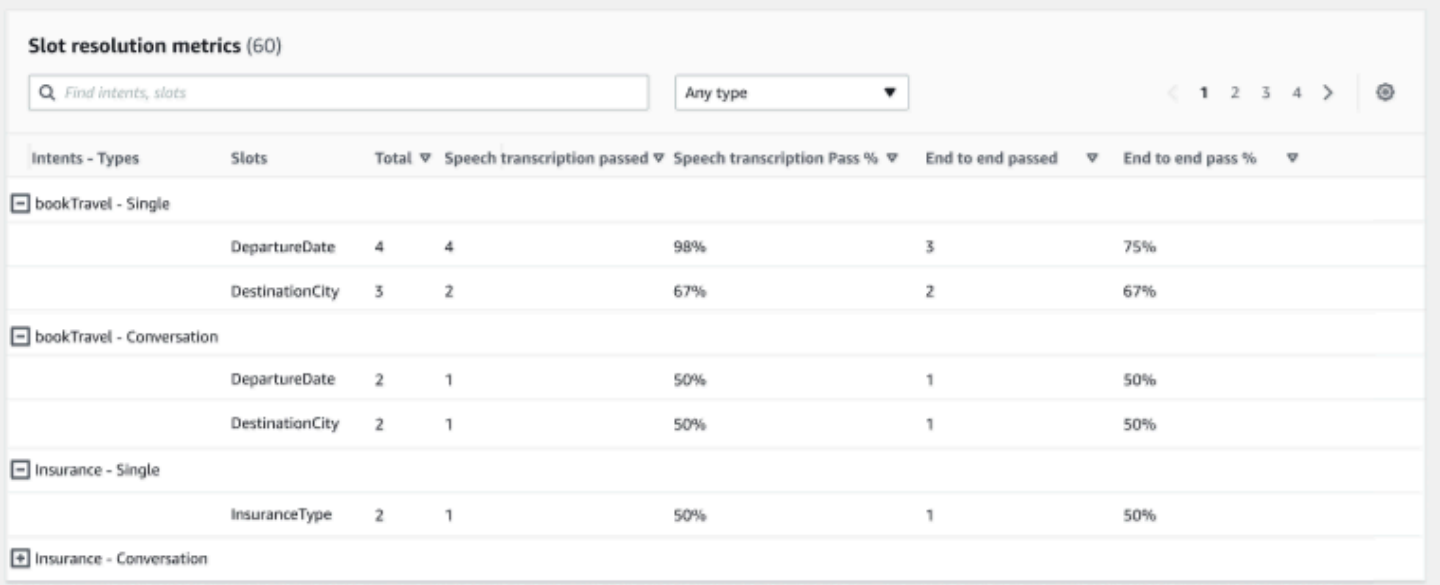


Intent recognition metrics (8)

Find intents, types | Any type | 1 2 3 4

Intents	Type	Total	Speech transcription passed	Speech transcription Pass %	End to end passed	End to end pass %
AccountLast4	Single input	27	23	85%	22	81%
AccountLast4	Conversation	6	5	83%	3	50%
bookTravel	Single input	3	2	67%	2	67%
bookTravel	Conversation	2	1	25%	1	25%
InsuranceType	Single input	2	1	50%	1	50%
InsuranceType	Conversation	2	1	50%	1	50%

意圖辨識度量 — 顯示成功辨識多少意圖的表格。顯示語音轉錄和 end-to-end 交談的合格率。



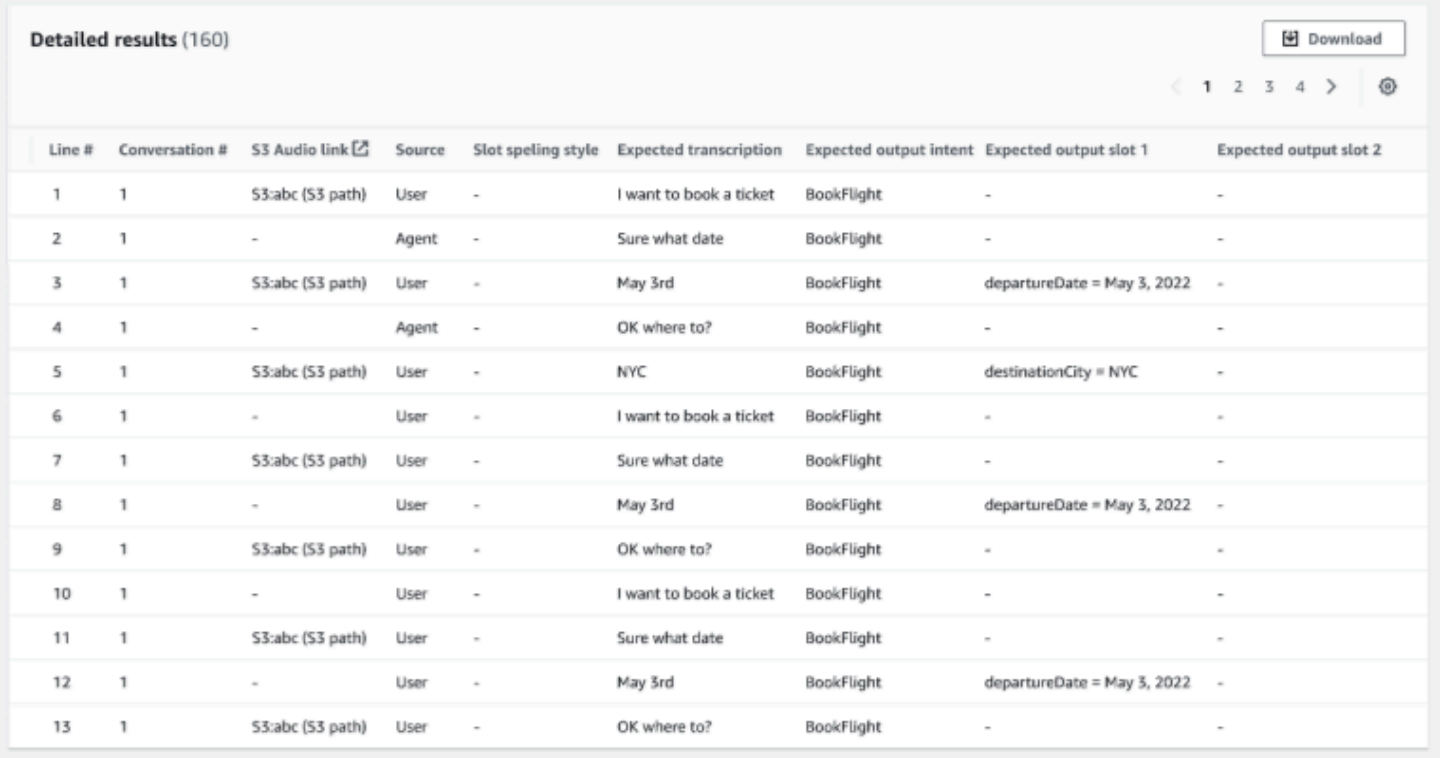
Slot resolution metrics (60)

Find intents, slots | Any type | 1 2 3 4

Intents - Types	Slots	Total	Speech transcription passed	Speech transcription Pass %	End to end passed	End to end pass %
bookTravel - Single						
	DepartureDate	4	4	98%	3	75%
	DestinationCity	3	2	67%	2	67%
bookTravel - Conversation						
	DepartureDate	2	1	50%	1	50%
	DestinationCity	2	1	50%	1	50%
Insurance - Single						
	InsuranceType	2	1	50%	1	50%
Insurance - Conversation						

插槽解析度量 — 分別顯示意圖和插槽，以及交談或單一輸入中使用之每個意圖的每個插槽的成功和失敗率。顯示語音轉錄和 end-to-end 交談的合格率。

詳細結果標籤：



Detailed results (160)

Download

1 2 3 4

Line #	Conversation #	S3 Audio link	Source	Slot speling style	Expected transcription	Expected output intent	Expected output slot 1	Expected output slot 2
1	1	S3:abc (S3 path)	User	-	I want to book a ticket	BookFlight	-	-
2	1	-	Agent	-	Sure what date	BookFlight	-	-
3	1	S3:abc (S3 path)	User	-	May 3rd	BookFlight	departureDate = May 3, 2022	-
4	1	-	Agent	-	OK where to?	BookFlight	-	-
5	1	S3:abc (S3 path)	User	-	NYC	BookFlight	destinationCity = NYC	-
6	1	-	User	-	I want to book a ticket	BookFlight	-	-
7	1	S3:abc (S3 path)	User	-	Sure what date	BookFlight	-	-
8	1	-	User	-	May 3rd	BookFlight	departureDate = May 3, 2022	-
9	1	S3:abc (S3 path)	User	-	OK where to?	BookFlight	-	-
10	1	-	User	-	I want to book a ticket	BookFlight	-	-
11	1	S3:abc (S3 path)	User	-	Sure what date	BookFlight	-	-
12	1	-	User	-	May 3rd	BookFlight	departureDate = May 3, 2022	-
13	1	S3:abc (S3 path)	User	-	OK where to?	BookFlight	-	-

詳細結果 — 在交談記錄上顯示詳細的表格，其中包含使用者和代理程式的話語，以及每個插槽的預期輸出和預期的轉錄。您可以選取 [下載] 按鈕來下載此報告。

下表列出具有案例的結果失敗錯誤訊息。

案例	錯誤訊息	動作
意圖不匹配	預期的 BookFlight 意圖，但它是 BookHotel 意圖。	跳過對話中的其他回合
插槽引出不匹配	預計離開日期插槽將被引發，但它是機艙類型。	跳過對話中的其他回合
插槽值不符	預期與實際插槽值不相符。	在對話中繼續其他回合
B ack-to-back 代理程式提示遺失	預期機器人在此回合中會傳回代理程式提示，但未收到。	跳過對話中的其他回合
轉錄不匹配	預期的轉錄與實際轉錄不匹配。	在對話中繼續其他回合

案例	錯誤訊息	動作
未引發可選插槽	預計將在下一回合引出 CabinType 插槽，但在此之前實現了當前的意圖。	跳過對話中的其他回合
插槽無法辨識	預計離開日期插槽在此回合中未被識別。	跳過對話中的其他回合
額外的 back-to-back 代理提示	預計用戶轉向，但它是代理提示	跳過對話中的其他回合

串流事件到 Amazon Lex V2 機器人

您可以使用 Amazon Lex V2 串流 API 在 Amazon Lex V2 機器人和應用程式之間啟動雙向串流。啟動串流可讓機器人管理機器人與使用者之間的對話。機器人會回應使用者輸入，而不需要您撰寫程式碼來處理使用者的回應。機器人可以：

- 在播放提示時處理來自用戶的中斷。如需詳細資訊，請參閱讓您的機器人被您的使用者中斷。
- 等待用戶提供輸入。例如，機器人可以等待用戶收集信用卡信息。如需詳細資訊，請參閱讓機器人等待使用者提供更多資訊。
- 在同一個串流中同時取得雙音多頻 (DTMF) 和音訊輸入。
- 處理使用者輸入中的暫停，比從應用程式管理交談時要好。

Amazon Lex V2 機器人不僅會回應從應用程式傳送的資料，還會將交談狀態的相關資訊傳送至您的應用程式。您可以使用此資訊來變更您的應用程式對客戶的回應方式。

Amazon Lex V2 機器人也會監控機器人與應用程式之間的連線。它可以確定連接是否已超時。

若要使用 API 開始串流至 Amazon Lex V2 機器人，請參閱開始串流至機器人。

當您從應用程式開始串流至 Amazon Lex V2 機器人時，可以將機器人設定為接受使用者的音訊輸入或文字輸入。您還可以選擇用戶是否接收音頻或文本以響應他們的輸入。

如果您已將 Amazon Lex V2 機器人設定為接受使用者的音訊輸入，則無法輸入文字。如果您已將機器人設定為接受文字輸入，使用者只能使用書面文字與其通訊。

當 Amazon Lex V2 機器人接受串流音訊輸入時，機器人會決定使用者何時開始說話以及停止說話的時間。它處理來自用戶的任何暫停或任何中斷。它還可以在同一個流中採取 DTMF（雙音多頻）輸入和語音輸入。這有助於用戶更自然地與機器人進行交互。您可以向使用者顯示歡迎訊息和提示。您也可以讓使用者中斷這些訊息和提示。

當您啟動雙向串流時，Amazon Lex V2 會使用 HTTP/2 通訊協定。您的應用程式和機器人會在單一串流中作為一系列事件交換資料。事件可為下列其中之一：

- 來自使用者的文字、音訊或 DTMF 輸入。
- 從應用程式傳送至 Amazon Lex V2 機器人的訊號。其中包括訊息的音訊播放已完成，或使用者已與工作階段中斷連線的指示。

如需事件的詳細資訊，請參閱開始串流至機器人。如需如何編碼事件的詳細資訊，請參閱事件串流事件。

主題

- 開始串流至機器人
- 事件串流事件
- 讓您的機器人被您的使用者中斷
- 讓機器人等待使用者提供更多資訊
- 設定出貨進度更新
- 設定擷取使用者輸入的逾時

開始串流至機器人

您可以使用此StartConversation作業在應用程式中的使用者和 Amazon Lex V2 機器人之間啟動串流。應用程式的POST要求會在您的應用程式和 Amazon Lex V2 機器人之間建立連線。這可讓您的應用程式和機器人透過事件開始互相交換資訊。

只有下列 SDK 才支援此`StartConversation`作業：

- 適用於 C++ 的 AWS 開發套件
- AWS SDK for Java 第 2 版
- JavaScript v3 適用的 AWS 開發套件
- 適用於 Ruby V3 的 AWS 開發套件

您的應用程式必須傳送至 Amazon Lex V2 機器人的第一個事件是 ConfigurationEvent. 此事件包含回應類型格式等資訊。下列是您可以在組態事件中使用的參數：

- responseContentType— 決定機器人是否以文字或語音回應使用者輸入。
- 工作階段狀態 — 與機器人串流工作階段相關的資訊，例如預先決定的意圖或對話狀態。
- 歡迎訊息 — 指定在使用者與機器人交談開始時為使用者播放的歡迎訊息。這些消息在用戶提供任何輸入之前播放。若要啟用歡迎使用訊息，您也必須指定`sessionState`和`dialogAction`參數的值。
- 停用播放 — 決定機器人在開始接聽呼叫者輸入之前是否應等待來自用戶端的提示。默認情況下，播放被激活，因此該字段的值為`false`。
- 要 requestAttributes — 提供請求的其他資訊。

如需如何為上述參數指定值的詳細資訊，請參閱StartConversation作業的資ConfigurationEvent料類型。

機器人與應用程式之間的每個串流只能有一個設定事件。在您的應用程式傳送設定事件之後，機器人可以從您的應用程式接收其他通訊。

如果您已指定使用者使用音訊與 Amazon Lex V2 機器人通訊，您的應用程式可以在該交談期間將下列事件傳送給機器人：

- AudioInputEvent— 包含具有 320 個位元組大小上限的音訊區塊。您的應用程式必須使用多個音訊輸入事件，才能將訊息從伺服器傳送至機器人。串流中的每個音訊輸入事件都必須具有相同的音訊格式。
- DTMFInputEvent — 將 DTMF 輸入傳送至機器人。每個 DTMF 按鍵都對應於一個事件。
- PlaybackCompletionEvent— 通知伺服器已播放使用者輸入的回應給他們。如果您要傳送音訊回應給使用者，則必須使用播放完成事件。如果您`disablePlayback`的配置事件是`true`，則無法使用此功能。
- DisconnectionEvent— 通知機器人用戶已斷開與對話的連接。

如果您已指定使用者使用文字與機器人通訊，您的應用程式可以在該對話期間將下列事件傳送給機器人：

- TextInputEvent— 從您的應用程序發送到機器人的文本。您在文字輸入事件中，最多可有 512 個字元。
- PlaybackCompletionEvent— 通知伺服器已播放使用者輸入的回應給他們。如果您要向使用者播放音訊，則必須使用此事件。如果您`disablePlayback`的配置事件是`true`，則無法使用此功能。
- DisconnectionEvent— 通知機器人用戶已斷開與對話的連接。

您必須以正確的格式對傳送至 Amazon Lex V2 機器人的每個事件進行編碼。如需詳細資訊，請參閱事件串流事件。

每個事件都有事件 ID。若要協助疑難排解串流中可能發生的任何問題，請為每個輸入事件指派唯一的事件 ID。然後，您可以使用機器人疑難排解任何處理失敗。

Amazon Lex V2 也使用時間戳記為每個事件。除了事件 ID 之外，您還可以使用這些時間戳記來協助疑難排解任何網路傳輸問題。

在使用者與 Amazon Lex V2 機器人之間的交談期間，機器人可以傳送下列輸出事件以回應使用者：

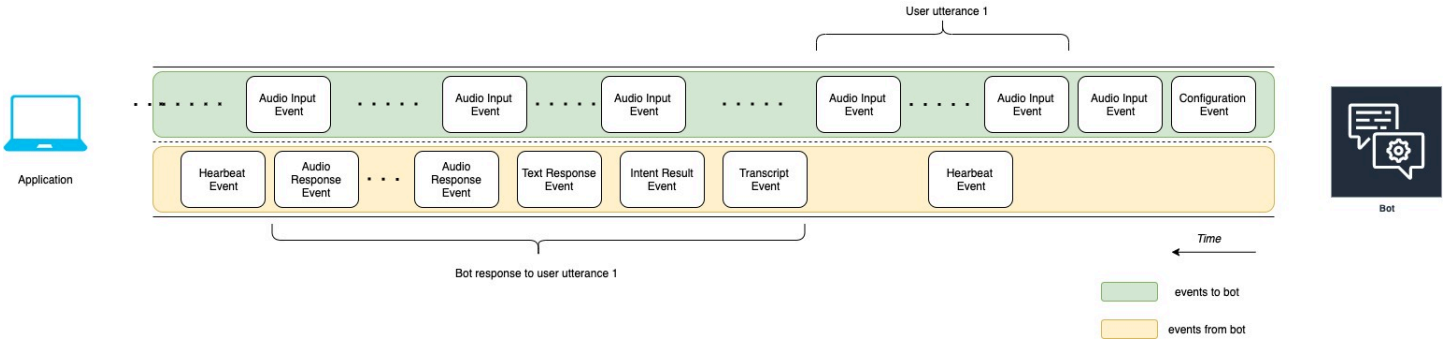
- IntentResultEvent— 包含 Amazon Lex V2 根據使用者說話所決定的意圖。每個內部結果事件包括：
  - 輸入模式 — 使用者說話的類型。有效值為 Speech、DTMF 或 Text。
  - 解釋 — Amazon Lex V2 根據使用者的話語決定的解釋。
  - requestAttributes-如果您尚未使用 lambda 函數修改請求屬性，則這些屬性與在交談開始時傳遞的屬性相同。
  - sessionId — 用於交談的工作階段識別碼。
  - 工作階段狀態 — 使用者與 Amazon Lex V2 的工作階段狀態。
- TranscriptEvent— 如果使用者為您的應用程式提供輸入，則此事件會包含使用者對機器人說話的文字記錄。TranscriptEvent如果沒有使用者輸入，您的應用程式將不會收到。

  傳送至應用程式的成績單事件值取決於您是否已將音訊 (語音和 DMTF) 或文字指定為交談模式：
  - 語音輸入的文字記錄 — 如果使用者正在與機器人交談，文字記錄事件就是使用者音訊的轉錄。這是從用戶開始說話到他們結束說話的時間內所有演講的成績單。
  - DTMF 輸入的文字記錄 — 如果使用者在鍵盤上輸入，謄本事件會包含使用者在其輸入中按下的所有數字。
  - 文字輸入的成績單 — 如果使用者提供文字輸入，則成績單事件會包含使用者輸入中的所有文字。
- TextResponseEvent— 包含文字格式的機器人回應。默認情況下返回文本響應。如果您已將 Amazon Lex V2 設定為傳回音訊回應，則會使用此文字來產生音訊回應。每個文字回應事件都包含機器人傳回給使用者的訊息物件陣列。
- AudioResponseEvent— 包含從中產生的文字合成的音訊回應TextResponseEvent。若要接收音訊回應事件，您必須設定 Amazon Lex V2 以提供音訊回應。所有音頻響應事件都具有相同的音頻格式。每個事件包含不超過 100 個字節的音頻塊。Amazon Lex V2 會傳送bytes欄位設定為的空白音訊區塊，null以指示音訊回應事件結束至您的應用程式。
- PlaybackInterruptionEvent— 當使用者中斷機器人已傳送至您應用程式的回應時，Amazon Lex V2 會觸發此事件以停止回應的播放。
- HeartbeatEvent— Amazon Lex V2 會定期傳回此事件，以防止應用程式與機器人之間的連線逾時。

音訊交談事件的時間序列

下圖顯示使用者和 Amazon Lex V2 機器人之間的串流音訊交談。該應用程序會持續將音頻流式傳輸到機器人，並且機器人從音頻中查找用戶輸入。在此範例中，使用者和機器人的名稱都是。每個圖表都對應於一個用戶的話語和機器人對該話語的響應。

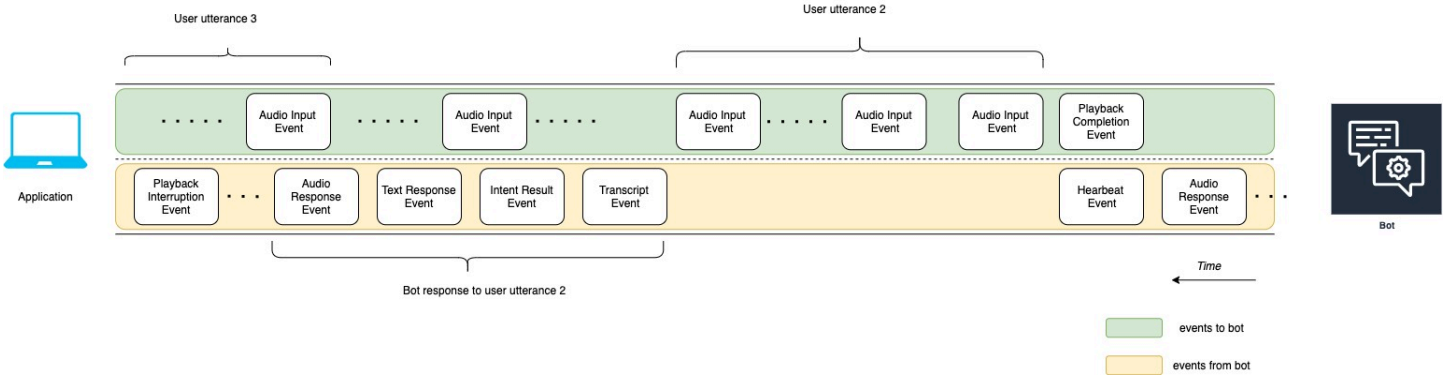
下圖顯示了應用程序和機器人之間的對話開始。串流從時間零 (t0) 開始。



下列清單說明前述圖表的事件。

- t0：應用程序將配置事件發送給機器人以啟動流。
- t1：應用程序流式傳輸音頻數據。此資料會分解為應用程式中的一系列輸入事件。
- t2：對於用戶說話 1，機器人會在用戶開始說話時檢測到音頻輸入事件。
- t2：使用者說話時，機器人會傳送活動訊號事件以維持連線。它會間歇性地傳送這些事件，以確保連線不會逾時。
- t3：機器人檢測到用戶的話語結束。
- t4：機器人會向應用程式傳回成績單事件，其中包含使用者演講的文字記錄。這是 Bot 回應使用者話語 1 的開始。
- t5：機器人發送意圖結果事件以指示用戶想要執行的操作。
- t6：機器人開始在文字回應事件中以文字形式提供回應。
- t7：機器人向應用程序發送一系列音頻響應事件以供用戶播放。
- t8：機器人會傳送另一個活動訊號事件，以間歇性地維護連線。

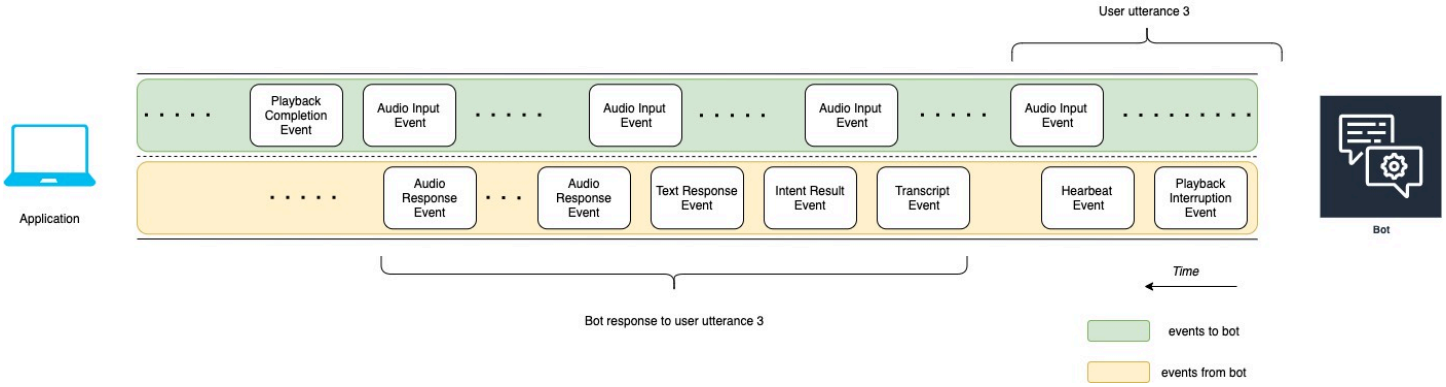
下圖是上一個圖表的延續。它顯示應用程序向機器人發送播放完成事件，以指示它已停止播放用戶的音頻響應。應用程式會向使用者播放 Bot 對使用者說話 1 的回應。使用者以使用者的話語 2 回應 Bot 回應使用者話語 1。



下列清單說明前述圖表的事件：

- t10：應用程序發送一個播放完成事件，以指示它已完成向用戶播放機器人的消息。
- t11：應用程序將用戶響應作為用戶的話語 2 發送回機器人。
- t12：對於 Bot 對使用者說話 2 的回應，機器人會等待使用者停止說話，然後開始提供音訊回應。
- t13：當機器人向應用程序發送 Bot 響應用程序時，機器人會檢測到用戶話語 3 的開始。機器人會停止 Bot 對使用者話語 2 的回應，並傳送播放中斷事件。
- t14：機器人向應用程序發送播放中斷事件，以表明用戶已中斷提示。

下圖顯示了機器人對用戶話語 3 的響應，並且在機器人響應用戶話語後繼續對話。



使用 API 開始串流事事件

當您開始串流至 Amazon Lex V2 機器人時，您可以完成下列任務：

1. 建立與伺服器的初始連線。
2. 設定安全登入資料和機器人詳細資料。機器人詳細資料包括機器人是否需要 DTMF 和音訊輸入，還是文字輸入。
3. 將事件發送到服務器。這些事件是來自用戶的文本數據或音頻數據。
4. 處理從伺服器傳送的事件。在此步驟中，您可以決定機器人輸出是以文字還是語音方式呈現給使用者。

下列程式碼範例會初始化與 Amazon Lex V2 機器人和本機電腦的串流交談。您可以修改程式碼以符合您的需求。

下列程式碼是範例要求，使用AWS SDK for Java來啟動與機器人的連線，並設定機器人詳細資料和認證。

```
package com.lex.streaming.sample;

import software.amazon.awssdk.auth.credentials.AwsBasicCredentials;
import software.amazon.awssdk.auth.credentials.AwsCredentialsProvider;
import software.amazon.awssdk.auth.credentials.StaticCredentialsProvider;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.lexruntimev2.LexRuntimeV2AsyncClient;
import software.amazon.awssdk.services.lexruntimev2.model.ConversationMode;
import software.amazon.awssdk.services.lexruntimev2.model.StartConversationRequest;

import java.net.URISyntaxException;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;

/**
 * The following code creates a connection with the Amazon Lex bot and configures the
 bot details and credentials.
 * Prerequisite: To use this example, you must be familiar with the Reactive streams
 programming model.
 * For more information, see
 * https://github.com/reactive-streams/reactive-streams-jvm.
 * This example uses AWS SDK for Java for Amazon Lex V2.
 * <p>
 * The following sample application interacts with an Amazon Lex bot with the streaming
 API. It uses the Audio
 * conversation mode to return audio responses to the user's input.
 * <p>
 * The code in this example accomplishes the following:
 * <p>
 * 1. Configure details about the conversation between the user and the Amazon Lex bot.
 These details include the conversation mode and the specific bot the user is speaking
 with.
 * 2. Create an events publisher that passes the audio events to the Amazon Lex bot
 after you establish the connection. The code we provide in this example tells your
 computer to pick up the audio from
 * your microphone and send that audio data to Amazon Lex.
 * 3. Create a response handler that handles the audio responses from the Amazon Lex
 bot and plays back the audio to you.
 */
public class LexBidirectionalStreamingExample {
```

```
    public static void main(String[] args) throws URISyntaxException,
InterruptedException {
        String botId = "";
        String botAliasId = "";
        String localeId = "";
        String accessKey = "";
        String secretKey = "";
        String sessionId = UUID.randomUUID().toString();
        Region region = Region.region_name; // Choose an AWS Region where the Amazon
Lex Streaming API is available.

        AwsCredentialsProvider awsCredentialsProvider = StaticCredentialsProvider
                .create(AwsBasicCredentials.create(accessKey, secretKey));

        // Create a new SDK client. You need to use an asynchronous client.
        System.out.println("step 1: creating a new Lex SDK client");
        LexRuntimeV2AsyncClient lexRuntimeServiceClient =
LexRuntimeV2AsyncClient.builder()
                .region(region)
                .credentialsProvider(awsCredentialsProvider)
                .build();


        // Configure the bot, alias and locale that you'll use to have a conversation.
        System.out.println("step 2: configuring bot details");
        StartConversationRequest.Builder startConversationRequestBuilder =
StartConversationRequest.builder()
                .botId(botId)
                .botAliasId(botAliasId)
                .localeId(localeId);

        // Configure the conversation mode of the bot. By default, the
        // conversation mode is audio.
        System.out.println("step 3: choosing conversation mode");
        startConversationRequestBuilder =
startConversationRequestBuilder.conversationMode(ConversationMode.AUDIO);

        // Assign a unique identifier for the conversation.
        System.out.println("step 4: choosing a unique conversation identifier");
        startConversationRequestBuilder =
startConversationRequestBuilder.sessionId(sessionId);

        // Start the initial request.
```

```
        StartConversationRequest startConversationRequest =
startConversationRequestBuilder.build();

        // Create a stream of audio data to the Amazon Lex bot. The stream will start
after the connection is established with the bot.
        EventsPublisher eventsPublisher = new EventsPublisher();

        // Create a class to handle responses from bot. After the server processes the
user data you've streamed, the server responds
        // on another stream.
        BotResponseHandler botResponseHandler = new
BotResponseHandler(eventsPublisher);

        // Start a connection and pass in the publisher that streams the audio and
process the responses from the bot.
        System.out.println("step 5: starting the conversation ...");
        CompletableFuture<Void> conversation =
lexRuntimeServiceClient.startConversation(
                startConversationRequest,
                eventsPublisher,
                botResponseHandler);

        // Wait until the conversation finishes. The conversation finishes if the
dialog state reaches the "Closed" state.
        // The client stops the connection. If an exception occurs during the
conversation, the
        // client sends a disconnection event.
        conversation.whenComplete((result, exception) -> {
            if (exception != null) {
                eventsPublisher.disconnect();
            }
        });

        // The conversation finishes when the dialog state is closed and last prompt
has been played.
        while (!botResponseHandler.isConversationComplete()) {
            Thread.sleep(100);
        }

        // Randomly sleep for 100 milliseconds to prevent JVM from exiting.
        // You won't need this in your production code because your JVM is
        // likely to always run.
        // When the conversation finishes, the following code block stops publishing
more data and informs the Amazon Lex bot that there is no more data to send.
```

```
        if (botResponseHandler.isConversationComplete()) {
            System.out.println("conversation is complete.");
            eventsPublisher.stop();
        }
    }
}
```

下列程式碼是使用將事件傳送AWS SDK for Java至機器人的範例要求。此範例中的程式碼會使用電腦上的麥克風來傳送音訊事件。

```
package com.lex.streaming.sample;

import org.reactivestreams.Publisher;
import org.reactivestreams.Subscriber;
import
 software.amazon.awssdk.services.lexruntimev2.model.StartConversationRequestEventStream;

/**
 * You use the Events publisher to send events to the Amazon Lex bot. When you
 establish a connection, the bot uses the
 * subscribe() method and enables the events publisher starts sending events to
 * your computer. The bot uses the "request" method of the subscription to make more
 requests. For more information on the request method, see https://github.com/reactive-
streams/reactive-streams-jvm.
 */
public class EventsPublisher implements Publisher<StartConversationRequestEventStream>
 {

    private AudioEventsSubscription audioEventsSubscription;

    @Override
    public void subscribe(Subscriber<? super StartConversationRequestEventStream>
 subscriber) {
        if (audioEventsSubscription == null) {

            audioEventsSubscription = new AudioEventsSubscription(subscriber);
            subscriber.onSubscribe(audioEventsSubscription);

        } else {
```

```
            throw new IllegalStateException("received unexpected subscription
 request");
        }
    }

    public void disconnect() {
        if (audioEventsSubscription != null) {
            audioEventsSubscription.disconnect();
        }
    }

    public void stop() {
        if (audioEventsSubscription != null) {
            audioEventsSubscription.stop();
        }
    }

    public void playbackFinished() {
        if (audioEventsSubscription != null) {
            audioEventsSubscription.playbackFinished();
        }
    }
}
```

下列程式碼是使用來處理機器人回應的範例要求。AWS SDK for Java此範例中的程式碼會將 Amazon Lex V2 設定為播放音訊回應給您。

```
package com.lex.streaming.sample;

import javazoom.jl.decoder.JavaLayerException;
import javazoom.jl.player.advanced.AdvancedPlayer;
import javazoom.jl.player.advanced.PlaybackEvent;
import javazoom.jl.player.advanced.PlaybackListener;
import software.amazon.awssdk.core.async.SdkPublisher;
import software.amazon.awssdk.services.lexruntimev2.model.AudioResponseEvent;
import software.amazon.awssdk.services.lexruntimev2.model.DialogActionType;
import software.amazon.awssdk.services.lexruntimev2.model.IntentResultEvent;
import software.amazon.awssdk.services.lexruntimev2.model.PlaybackInterruptionEvent;
import software.amazon.awssdk.services.lexruntimev2.model.StartConversationResponse;
```

```
import
 software.amazon.awssdk.services.lexruntimev2.model.StartConversationResponseEventStream;
import
 software.amazon.awssdk.services.lexruntimev2.model.StartConversationResponseHandler;
import software.amazon.awssdk.services.lexruntimev2.model.TextResponseEvent;
import software.amazon.awssdk.services.lexruntimev2.model.TranscriptEvent;

import java.io.IOException;
import java.io.UncheckedIOException;
import java.util.concurrent.CompletableFuture;

/**
 * The following class is responsible for processing events sent from the Amazon Lex
 bot. The bot sends multiple audio events,
 * so the following code concatenates those audio events and uses a publicly available
 Java audio player to play out the message to
 * the user.
 */
public class BotResponseHandler implements StartConversationResponseHandler {

    private final EventsPublisher eventsPublisher;

    private boolean lastBotResponsePlayedBack;
    private boolean isDialogStateClosed;
    private AudioResponse audioResponse;


    public BotResponseHandler(EventsPublisher eventsPublisher) {
        this.eventsPublisher = eventsPublisher;
        this.lastBotResponsePlayedBack = false;// At the start, we have not played back
 last response from bot.
        this.isDialogStateClosed = false; // At the start, the dialog state is open.
    }

    @Override
    public void responseReceived(StartConversationResponse startConversationResponse) {
        System.out.println("successfully established the connection with server.
 request id:" + startConversationResponse.responseMetadata().requestId()); // would
 have 2XX, request id.
    }

    @Override
    public void onEventStream(SdkPublisher<StartConversationResponseEventStream>
 sdkPublisher) {
```

```
        sdkPublisher.subscribe(event -> {
            if (event instanceof PlaybackInterruptionEvent) {
                handle((PlaybackInterruptionEvent) event);
            } else if (event instanceof TranscriptEvent) {
                handle((TranscriptEvent) event);
            } else if (event instanceof IntentResultEvent) {
                handle((IntentResultEvent) event);
            } else if (event instanceof TextResponseEvent) {
                handle((TextResponseEvent) event);
            } else if (event instanceof AudioResponseEvent) {
                handle((AudioResponseEvent) event);
            }
        });
    }

    @Override
    public void exceptionOccurred(Throwable throwable) {
        System.err.println("got an exception:" + throwable);
    }

    @Override
    public void complete() {
        System.out.println("on complete");
    }

    private void handle(PlaybackInterruptionEvent event) {
        System.out.println("Got a PlaybackInterruptionEvent: " + event);
    }

    private void handle(TranscriptEvent event) {
        System.out.println("Got a TranscriptEvent: " + event);
    }


    private void handle(IntentResultEvent event) {
        System.out.println("Got an IntentResultEvent: " + event);
        isDialogStateClosed =
DialogActionType.CLOSE.equals(event.sessionState().dialogAction().type());
    }

    private void handle(TextResponseEvent event) {
        System.out.println("Got an TextResponseEvent: " + event);
        event.messages().forEach(message -> {
```

```
            System.out.println("Message content type:" + message.contentType());
            System.out.println("Message content:" + message.content());
        });
    }

    private void handle(AudioResponseEvent event) {//Synthesize speech
        // System.out.println("Got a AudioResponseEvent: " + event);
        if (audioResponse == null) {
            audioResponse = new AudioResponse();
            //Start an audio player in a different thread.
            CompletableFuture.runAsync(() -> {
                try {
                    AdvancedPlayer audioPlayer = new AdvancedPlayer(audioResponse);

                    audioPlayer.setPlayBackListener(new PlaybackListener() {
                        @Override
                        public void playbackFinished(PlaybackEvent evt) {
                            super.playbackFinished(evt);

                            // Inform the Amazon Lex bot that the playback has
finished.
                            eventsPublisher.playbackFinished();
                            if (isDialogStateClosed) {
                                lastBotResponsePlayedBack = true;
                            }
                        }
                    });
                    audioPlayer.play();
                } catch (JavaLayerException e) {
                    throw new RuntimeException("got an exception when using audio
player", e);
                }
            });
        }

        if (event.audioChunk() != null) {
            audioResponse.write(event.audioChunk().asByteArray());
        } else {
            // The audio audio prompt has ended when the audio response has no
            // audio bytes.
            try {
                audioResponse.close();
                audioResponse = null;  // Prepare for the next audio prompt.
            } catch (IOException e) {
```

```
                throw new UncheckedIOException("got an exception when closing the audio
 response", e);
            }
        }
    }

    // The conversation with the Amazon Lex bot is complete when the bot marks the
 Dialog as DialogActionType.CLOSE
    // and any prompt playback is finished. For more information, see
    // https://docs.aws.amazon.com/lexv2/latest/dg/API_runtime_DialogAction.html.
    public boolean isConversationComplete() {
        return isDialogStateClosed && lastBotResponsePlayedBack;
    }

}
```

若要設定機器人以回應含音訊的輸入事件，您必須先從 Amazon Lex V2 訂閱音訊事件，然後設定機器人為使用者的輸入事件提供音訊回應。

下列程式碼是從 Amazon Lex V2 訂閱音訊事件的AWS SDK for Java範例。

```
package com.lex.streaming.sample;

import org.reactivestreams.Subscriber;
import org.reactivestreams.Subscription;
import software.amazon.awssdk.core.SdkBytes;
import software.amazon.awssdk.services.lexruntimev2.model.AudioInputEvent;
import software.amazon.awssdk.services.lexruntimev2.model.ConfigurationEvent;
import software.amazon.awssdk.services.lexruntimev2.model.DisconnectionEvent;
import software.amazon.awssdk.services.lexruntimev2.model.PlaybackCompletionEvent;
import
 software.amazon.awssdk.services.lexruntimev2.model.StartConversationRequestEventStream;

import javax.sound.sampled.AudioFormat;
import javax.sound.sampled.AudioInputStream;
import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.DataLine;
import javax.sound.sampled.LineUnavailableException;
import javax.sound.sampled.TargetDataLine;
import java.io.IOException;
```

```
import java.io.UncheckedIOException;
import java.nio.ByteBuffer;
import java.util.Arrays;
import java.util.concurrent.BlockingQueue;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.LinkedBlockingQueue;
import java.util.concurrent.atomic.AtomicLong;

public class AudioEventsSubscription implements Subscription {
    private static final AudioFormat MIC_FORMAT = new AudioFormat(8000, 16, 1, true,
 false);
    private static final String AUDIO_CONTENT_TYPE = "audio/lpcm; sample-rate=8000;
 sample-size-bits=16; channel-count=1; is-big-endian=false";
    //private static final String RESPONSE_TYPE = "audio/pcm; sample-rate=8000";
    private static final String RESPONSE_TYPE = "audio/mpeg";
    private static final int BYTES_IN_AUDIO_CHUNK = 320;
    private static final AtomicLong eventIdGenerator = new AtomicLong(0);

    private final AudioInputStream audioInputStream;
    private final Subscriber<? super StartConversationRequestEventStream> subscriber;
    private final EventWriter eventWriter;
    private CompletableFuture eventWriterFuture;


    public AudioEventsSubscription(Subscriber<? super
 StartConversationRequestEventStream> subscriber) {
        this.audioInputStream = getMicStream();
        this.subscriber = subscriber;
        this.eventWriter = new EventWriter(subscriber, audioInputStream);
        configureConversation();
    }

    private AudioInputStream getMicStream() {
        try {
            DataLine.Info dataLineInfo = new DataLine.Info(TargetDataLine.class,
 MIC_FORMAT);
            TargetDataLine targetDataLine = (TargetDataLine)
 AudioSystem.getLine(dataLineInfo);

            targetDataLine.open(MIC_FORMAT);
            targetDataLine.start();

            return new AudioInputStream(targetDataLine);
        } catch (LineUnavailableException e) {
```

```
            throw new RuntimeException(e);
        }
    }

    @Override
    public void request(long demand) {
        // If a thread to write events has not been started, start it.
        if (eventWriterFuture == null) {
            eventWriterFuture = CompletableFuture.runAsync(eventWriter);
        }
        eventWriter.addDemand(demand);
    }

    @Override
    public void cancel() {
        subscriber.onError(new RuntimeException("stream was cancelled"));
        try {
            audioInputStream.close();
        } catch (IOException e) {
            throw new UncheckedIOException(e);
        }
    }

    public void configureConversation() {
        String eventId = "ConfigurationEvent-" +
String.valueOf(eventIdGenerator.incrementAndGet());

        ConfigurationEvent configurationEvent = StartConversationRequestEventStream
                .configurationEventBuilder()
                .eventId(eventId)
                .clientTimestampMillis(System.currentTimeMillis())
                .responseContentType(RESPONSE_TYPE)
                .build();

        System.out.println("writing config event");
        eventWriter.writeConfigurationEvent(configurationEvent);
    }

    public void disconnect() {

        String eventId = "DisconnectionEvent-" +
String.valueOf(eventIdGenerator.incrementAndGet());

        DisconnectionEvent disconnectionEvent = StartConversationRequestEventStream
```

```
                .disconnectionEventBuilder()
                .eventId(eventId)
                .clientTimestampMillis(System.currentTimeMillis())
                .build();

        eventWriter.writeDisconnectEvent(disconnectionEvent);

        try {
            audioInputStream.close();
        } catch (IOException e) {
            throw new UncheckedIOException(e);
        }

    }
    //Notify the subscriber that we've finished.
    public void stop() {
        subscriber.onComplete();
    }

    public void playbackFinished() {
        String eventId = "PlaybackCompletion-" +
String.valueOf(eventIdGenerator.incrementAndGet());

        PlaybackCompletionEvent playbackCompletionEvent =
StartConversationRequestEventStream
                .playbackCompletionEventBuilder()
                .eventId(eventId)
                .clientTimestampMillis(System.currentTimeMillis())
                .build();

        eventWriter.writePlaybackFinishedEvent(playbackCompletionEvent);
    }

    private static class EventWriter implements Runnable {
        private final BlockingQueue<StartConversationRequestEventStream> eventQueue;
        private final AudioInputStream audioInputStream;
        private final AtomicLong demand;
        private final Subscriber subscriber;

        private boolean conversationConfigured;

        public EventWriter(Subscriber subscriber, AudioInputStream audioInputStream) {
            this.eventQueue = new LinkedBlockingQueue<>();
```

```
            this.demand = new AtomicLong(0);
            this.subscriber = subscriber;
            this.audioInputStream = audioInputStream;
        }

        public void writeConfigurationEvent(ConfigurationEvent configurationEvent) {
            eventQueue.add(configurationEvent);
        }

        public void writeDisconnectEvent(DisconnectionEvent disconnectionEvent) {
            eventQueue.add(disconnectionEvent);
        }

        public void writePlaybackFinishedEvent(PlaybackCompletionEvent
playbackCompletionEvent) {
            eventQueue.add(playbackCompletionEvent);
        }

        void addDemand(long l) {
            this.demand.addAndGet(l);
        }

        @Override
        public void run() {
            try {

                while (true) {
                    long currentDemand = demand.get();

                    if (currentDemand > 0) {
                        // Try to read from queue of events.
                        // If nothing is in queue at this point, read the audio events
directly from audio stream.
                        for (long i = 0; i < currentDemand; i++) {

                            if (eventQueue.peek() != null) {
                                subscriber.onNext(eventQueue.take());
                                demand.decrementAndGet();
                            } else {
                                writeAudioEvent();
                            }
                        }
                    }
                }
```

```
            } catch (InterruptedException e) {
                throw new RuntimeException("interrupted when reading data to be sent to
server");
            } catch (Exception e) {
                e.printStackTrace();
            }
        }

        private void writeAudioEvent() {
            byte[] bytes = new byte[BYTES_IN_AUDIO_CHUNK];

            int numBytesRead = 0;
            try {
                numBytesRead = audioInputStream.read(bytes);
                if (numBytesRead != -1) {
                    byte[] byteArrayCopy = Arrays.copyOf(bytes, numBytesRead);

                    String eventId = "AudioEvent-" +
String.valueOf(eventIdGenerator.incrementAndGet());

                    AudioInputEvent audioInputEvent =
StartConversationRequestEventStream
                            .audioInputEventBuilder()

.audioChunk(SdkBytes.fromByteBuffer(ByteBuffer.wrap(byteArrayCopy)))
                            .contentType(AUDIO_CONTENT_TYPE)
                            .clientTimestampMillis(System.currentTimeMillis())
                            .eventId(eventId).build();

                    //System.out.println("sending audio event:" + audioInputEvent);
                    subscriber.onNext(audioInputEvent);
                    demand.decrementAndGet();
                    //System.out.println("sent audio event:" + audioInputEvent);
                } else {
                    subscriber.onComplete();
                    System.out.println("audio stream has ended");
                }

            } catch (IOException e) {
                System.out.println("got an exception when reading from audio stream");
                System.err.println(e);
                subscriber.onError(e);
            }
        }
```

```
    }
}
```

下列AWS SDK for Java範例會將 Amazon Lex V2 機器人設定為對輸入事件提供音訊回應。

```
package com.lex.streaming.sample;

import java.io.IOException;
import java.io.InputStream;
import java.io.UncheckedIOException;
import java.util.Optional;
import java.util.concurrent.LinkedBlockingQueue;
import java.util.concurrent.TimeUnit;

public class AudioResponse extends InputStream{

    // Used to convert byte, which is signed in Java, to positive integer (unsigned)
    private static final int UNSIGNED_BYTE_MASK = 0xFF;
    private static final long POLL_INTERVAL_MS = 10;

    private final LinkedBlockingQueue<Integer> byteQueue = new LinkedBlockingQueue<>();

    private volatile boolean closed;

    @Override
    public int read() throws IOException {
        try {
            Optional<Integer> maybeInt;
            while (true) {
                maybeInt = Optional.ofNullable(this.byteQueue.poll(POLL_INTERVAL_MS,
 TimeUnit.MILLISECONDS));

                // If we get an integer from the queue, return it.
                if (maybeInt.isPresent()) {
                    return maybeInt.get();
                }

                // If the stream is closed and there is nothing queued up, return -1.
                if (this.closed) {
                    return -1;
```

```
                }
            }
        } catch (InterruptedException e) {
            throw new IOException(e);
        }
    }

    /**
     * Writes data into the stream to be offered on future read() calls.
     */
    public void write(byte[] byteArray) {
        // Don't write into the stream if it is already closed.
        if (this.closed) {
            throw new UncheckedIOException(new IOException("Stream already closed when
 attempting to write into it."));
        }

        for (byte b : byteArray) {
            this.byteQueue.add(b & UNSIGNED_BYTE_MASK);
        }
    }

    @Override
    public void close() throws IOException {
        this.closed = true;
        super.close();
    }
}
```

事件串流事件

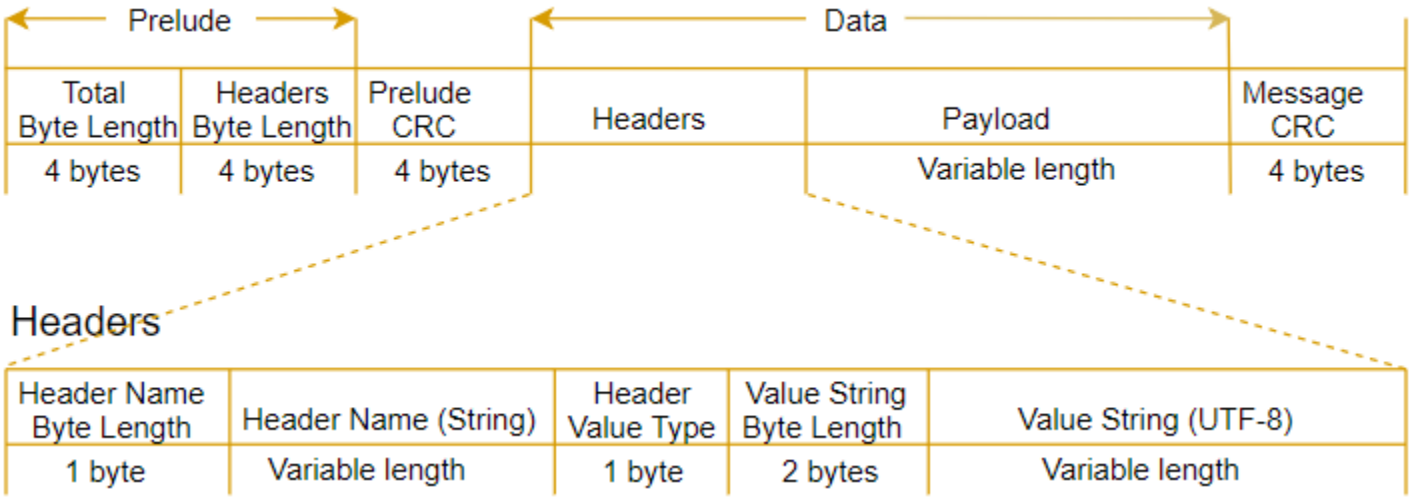
事件串流編碼會在用戶端與伺服器之間使用訊息來提供雙向通訊。傳送至 Amazon Lex V2 串流服務的資料框架會以這種格式編碼。來自 Amazon Lex V2 的響應也使用這種編碼。

每則訊息包含兩個部分：序言和資料。前奏部分包含消息的總字節長度和所有標頭的組合字節長度。data 部分包含標題和有效負載。

每個區段以 4 位元組由大到小的整數 CRC 檢查總和結束。消息 CRC 校驗和包括前奏部分和數據部分。Amazon Lex V2 使用 CRC32（通常稱為 GZIP CRC32）來計算兩個 CRC。如需 CRC32 的詳細資訊，請參閱 GZIP 檔案格式規格 4.3 版。

總訊息額外負荷 (包括序言和兩個檢查總和) 為 16 位元組。

下圖顯示構成訊息和標題的元件。每則訊息有多個標頭。



每則訊息都包含下列元件：

- 序言： 一律為 8 位元組的固定大小，兩個欄位每一個為 4 位元組。
  - 前 4 位元組：總位元組長度。這是整個訊息的大端序整數位元組長度，包括 4 位元組長度欄位本身。
  - 後 4 位元組：標頭位元組長度。這是訊息標頭部分的大端序整數位元組長度，不含 4 位元組長度欄位本身。
- 序言 CRC：訊息序言部分的 4 位元組 CRC 檢查總和，不含 CRC 本身。序曲與訊息 CRC 具有獨立的 CRC，以確保 Amazon Lex V2 可以立即偵測到損毀的位元組長度資訊，而不會造成緩衝區溢位等錯誤。
- 標頭：註釋訊息的中繼資料，例如訊息類型、內容類型等等。訊息有多個標頭。標頭是索引鍵值組，其中索引鍵是 UTF-8 字串。標頭可以任何順序出現在訊息的標頭部分中，而且任何指定標頭只能出現一次。如需必要標頭類型，請參閱以下各節。
- 承載：音訊或文字內容傳送至 Amazon Lex。
- 訊息 CRC：從訊息開頭到檢查總和開頭的 4 位元組 CRC 檢查總和。這包括除了 CRC 本身之外的消息中的所有內容。

每個標頭都包含下列元件。每個框架有多個標頭。

- 標頭名稱位元組長度：標頭名稱的位元組長度。
- 標頭名稱：指出標頭類型的標頭名稱。如需有效值，請參閱下列框架描述。

- 標頭值類型：指出標頭值類型的列舉。
- 數值字串位元組長度： 標頭值字串的位元組長度。
- 標頭值：標頭字串的值。此欄位的有效值取決於標頭類型。如需有效值，請參閱下列框架描述。

讓您的機器人被您的使用者中斷

當您在 Amazon Lex V2 機器人與應用程式之間啟動雙向音訊串流時，您可以設定機器人在傳回提示時接聽使用者輸入。有了這個，用戶可以在機器人完成播放之前中斷提示。您可以在使用者可能已知道問題答案的情況下使用此設定，例如當系統提示他們提供 CVV 代碼時。

機器人會知道使用者在偵測到使用者輸入時何時中斷提示，然後您的應用程式可以傳回`PlaybackCompletion`事件。當用戶中斷一個機器人時，機器人會發送一個`PlaybackInterruptionEvent`.

根據預設，使用者可以中斷機器人正在串流到您的應用程式的任何提示。您可以在 Amazon Lex V2 主控台中變更此設定。

您可以透過編輯位置來變更使用者回應提示的方式。插槽是意圖的一部分，它是用戶為您提供所需信息的手段。每個插槽都有一個提示，要求用戶向您提供該信息。若要進一步了解插槽，請參閱運作方式。

要改變使用者是否可以中斷提示 (主控台)

1. 在 Amazon Lex V2 主控台登入AWS Management Console並開啟 Amazon Lex V2 主控台。
2. 在機器人下，選擇一個機器人。
3. 在 [語言] 下，選取機器人的語言。
4. 選擇 [檢視方式]。
5. 選擇 意圖。
6. 對於「插槽」，請選擇插槽。
7. 在進階選項下，選擇插槽提示。
8. 選擇更多提示選項。
9. 選取或取消選取使用者可在讀取提示時中斷提示。

您可以透過建立具有兩個插槽的機器人，並指定使用者無法中斷一個插槽的提示來測試此功能。如果您中斷可中斷的提示，機器人會傳送播放中斷事件。如果您中斷不間斷，提示會繼續播放。

讓機器人等待使用者提供更多資訊

當您從 Amazon Lex V2 機器人啟動雙向串流至應用程式時，可以將機器人設定為等待使用者提供其他資訊。在某些情況下，使用者可能尚未準備好回應提示。例如，使用者可能尚未準備好提供他們的信用卡資訊，因為他們的錢包位於另一個房間。

透過使用 Amazon Lex V2 機器人的「等待並繼續」行為，使用者可以說出「等待一秒鐘」之類的詞組，讓機器人等待他們尋找資訊並提供資訊。當您啟用此行為時，機器人會定期向使用者傳送提醒，以提供資訊。它不會傳回成績單事件，因為沒有要轉錄的使用者話語。

Amazon Lex V2 機器人會自動管理串流交談。您不必撰寫任何額外的程式碼來啟用此功能。當機器人系統提示使用者等待時，`IntentDialogAction`是`Waiting`和`type`的`ElicitSlot`。`state`您可以使用此資訊來協助您根據需求自訂應用程式。例如，您可以將應用程式設定為在使用者尋找其信用卡時播放音樂。

您可以啟用個別插槽的等待和繼續行為。若要進一步了解插槽，請參閱運作方式。

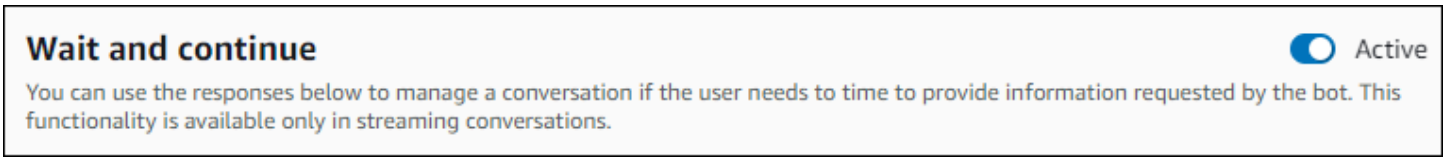
啟用等待並繼續

1. 在 Amazon Lex V2 主控台登入AWS Management Console並開啟 Amazon Lex V2 主控台。
2. 在機器人下，選擇一個機器人。
3. 在 [語言] 下，選取機器人的語言。
4. 選擇 [檢視方式]。
5. 選擇 意圖。
6. 在插槽下，選擇一個插槽。
7. 在「高級選項」下，選擇「等待並繼續」。
8. 在「等待並繼續」下，指定下列欄位：
   - 當用戶希望機器人等待時的響應-這是機器人在用戶要求等待其他信息時響應的方式。
   - 如果用戶需要機器人繼續等待，則響應 — 這是機器人發送的響應，提醒用戶它仍在等待信息。您可以變更機器人提醒使用者的頻率。
   - 用戶想要繼續時的響應 — 當用戶獲得請求的信息時，這是機器人的響應。

對於每個機器人響應，您可以提供響應的多種變體，並隨機向用戶顯示一個。您也可以選擇是否可以由使用者中斷這些回應。

若要測試等待和繼續功能，請將機器人設定為等待使用者輸入，然後開始串流至 Amazon Lex V2 機器人。如需串流至機器人的相關資訊，請參閱使用 API 開始串流事件。

您可能需要關閉等待並繼續回應。使用作用中切換來設定是否要使用等待和繼續回應。



設定出貨進度更新

呼叫意圖的履行 Lambda 函數時，機器人在函數完成之前不會傳送回應。如果 Lambda 函數需要幾秒鐘以上的時間才能完成，使用者可能會認為機器人沒有回應。為了解決這個問題，您可以將機器人設定為在履行 Lambda 函數執行時將更新傳送給使用者，以便使用者知道機器人仍在處理其請求。

當您將出貨更新新增至意圖時，機器人會在履行開始時回應，並在履行進行期間定期回應。當您設定開始回應時，您可以指定機器人傳送回應之前的延遲時間。這樣，您就可以支持履行未相對快速完成的案例。設定更新回應時，請指定要傳送更新的頻率。您也可以設定逾時，以限制履行函數必須執行的時間。

您也可以將履行後回應新增至機器人。這可以讓機器人根據履行成功、失敗或逾時向您傳送不同的回應。

只有在使用StartConversation作業與機器人互動時，才會使用履行更新。您可以在使用StartConversation、RecognizeText和RecognizeUtterance作業與機器人互動時使用履行後更新

履行更新

當您的 Lambda 函數完成意圖時，系統會傳送履行更新。當您開啟出貨更新時，您會提供在出貨作業開始時傳送的開始回應，以及在出貨作業進行期間定期傳送的更新回應。

當您指定更新回應時，也會指定逾時，決定履行函數可執行的時間長度。您可以指定最多 15 分鐘 (900 秒) 的逾時長度。

如果您在主控台中設定`active`為 false 或使用或作業來關閉出貨UpdateIntent作業更新，則不會使用為出貨作業更新指定的逾時，而是使用預設逾時 30 秒。CreateIntent

如果履行功能逾時，Amazon Lex V2 會執行以下三項作業之一：

- 履行後回應已設定且處於作用中狀態 — 傳回逾時回應。

- 已設定「履行後回應」且非作用中 — 傳回例外狀況。
- 未設定履行後回應 — 傳回例外狀況。

開始回應

在串流交談期間呼叫 Lambda 履行函數時，Amazon Lex V2 會傳回開始回應。它通常告訴用戶，實現意圖需要一些時間，他們應該等待。當您使用`RecognizeText`或`RecognizeUtterance`作業時，不會傳回開始回應。

您可以指定最多五個回應訊息。Amazon Lex V2 會從中選擇一條訊息向使用者播放。

您可以設定呼叫 Lambda 函數到傳回開始回應之間的延遲。如果 Lambda 函數在延遲完成之前完成其工作，則不會傳回開始回應。

您可以使用控制台或FulfillmentUpdatesSpecification結構中的`active`切換開關來打開和關閉開始響應。如果`active`是 false，則不會播放開始回應。

更新回應

Amazon Lex 會在 Lambda 履行函數執行期間，在串流交談期間定期傳回更新回應。當您使用`RecognizeText`或`RecognizeUtterance`作業時，不會播放更新回應。您可以設定更新回應播放的頻率。例如，您可以在履行功能執行期間每 30 秒播放一次更新回應，讓使用者知道程序正在執行，以及他們應該繼續等待。

您可以指定最多五個更新訊息。Amazon Lex V2 選擇要播放給用戶的消息。使用多個消息可以防止更新重複。

如果使用者在履行 Lambda 函數執行時透過語音、DTMF 或文字提供輸入，Amazon Lex V2 會將更新回應傳回給使用者。

如果 Lambda 函數在第一個更新期間結束之前完成其工作，則不會傳回更新回應。

您可以使用主控台或FulfillmentUpdatesSpecification結構中的`active`切換開關來開啟和關閉更新回應。如果`active`是 false，則不會傳回更新回應。

履行後回應

當履行功能結束時，Amazon Lex V2 會傳回履行後的回應。完成任何意圖時，可以使用履行後的響應，而不僅僅是在流式傳輸對話時。履行後回應可讓使用者知道函數已完成及結果。

您可以使用主控台或PostFulfillmentStatusSpecification結構中的active切換開關，開啟或關閉履行後回應。如果active為假，則不播放響應。

履行後的回應有三種類型：

- 成功 — 在履行 Lambda 函數成功完成工作時傳回。如果履行後回應未啟用。Amazon Lex V2 採取下一個設定的動作。
- 逾時 — 如果 Lambda 函數未在設定的逾時期間結束之前完成其工作，則傳回。如果履行後回應未處於作用中狀態，Amazon Lex V2 會傳回例外狀況。
- 失敗 — 當 Lambda 函數傳回回應Failed中的狀態，或 Amazon Lex V2 在完成意圖時遇到錯誤時傳回。如果履行後回應未處於作用中狀態，Amazon Lex V2 會傳回例外狀況。

您可以為每個類型指定最多五個訊息。Amazon Lex V2 會從中選擇一條訊息向使用者播放。

與履行開始和履行更新回應不同，履行後的回應會同時針對串流和非串流對話播放。

您也可以選擇將 Lambda 函數設定為傳回履行後訊息來覆寫這些訊息。

> **Note**
>
> 如果意圖具有結束回應，則會在履行後回應之後傳回。

履行後範例

為了更好地了解履行後的回應，我們以一個*BookTrip*機器人為例，建立以協助計劃行程而建立的機器人，並以履行 Lambda 函數進行配置，該功能可為客戶預留航空公司的航班。*BookFlight*取*BookFlight*得的插槽之後，Amazon Lex V2 就會叫用履行 Lambda 函數。在此履行過程中，可能會發生下列三種結果之一：

- 成功 — 航班已成功預訂。
- 逾時 — 預訂程序所花費的時間超過設定的履行 Lambda 執行時間 (例如，如果無法在規定的時間內聯絡航空公司)。
- 失敗 — 由於其他原因，預訂失敗。

您可以利用履行後的回應，在每種情況下為客戶提供更有意義的回應。各種情況的範例如下：

- 成功回應 —「我們能夠成功預訂您的機票，並向您發送確認電子郵件。如果您有任何疑問，請隨時使用該電子郵件中提供的聯繫信息與我們聯繫。」
- 逾時回應 —「由於我們的系統流量繁忙，預訂機票的時間比預期更長。我們已經將您的請求放在我們的隊列中，並已向您發送一封電子郵件，其中包含與此請求對應的參考號 預訂機票後，我們將向您發送預訂確認信。如果您有任何疑問，請隨時使用該電子郵件中提供的聯繫信息與我們聯繫。」

> **Note**
>
> 如果您未設定逾時訊息，Lex 會擲回對應於使用案例的 4XX 錯誤。

- 失敗回應 —「很抱歉，我們無法預訂您的機票。我們已經發送了一封電子郵件，其中包含我們在預訂時遇到的問題的詳細信息。」

設定擷取使用者輸入的逾時

Amazon Lex V2 串流 API 可讓機器人自動偵測使用者輸入中的語音。當您建立意圖或插槽時，您可以設定語音的各個層面，例如語音的持續時間上限、等待使用者輸入時的逾時，或 DTMF 輸入的結束字元。您可以針對您的使用案例自訂機器人的行為。例如，您可以將信用卡號碼的位數限制為 16。

您也可以在與機器人開始交談時透過工作階段屬性設定逾時，並在必要時在 Lambda 函數中覆寫逾時。

屬性的組態索引鍵語法如預期運作：

```
x-amz-lex:<InputType>:<BehaviorName>:<IntentName>:<SlotName>
```

InputType 可以是 **audio**、**dtmf** 或 **text**。

您可以透過指定意圖或位置名稱來設定*機器人中所有意圖或位置的預設設定。任何意圖或槽特定的設定都優先於預設設定。

Amazon Lex V2 提供預先定義的工作階段屬性，用於管理StartConversation操作以文字、語音或 DTMF 輸入到機器人的方式。預先定義的屬性全都位於 x-amz-lex 命名空間中。

您可以透過指定意圖或位置名稱來設定機器人中所有意圖、位置或子插槽的預設設定*。任何意圖或槽特定的設定都優先於預設設定。在下面的所有超時中使用這些模式。

對於複合插槽的子插槽，您可以分開 .。例如：

```
<slotName>.<subSlotName>
```

```
x-amz-lex:allow-interrupt:<intentName>:<slotName>.<subSlotName>
```

表達式	案例
意圖：插槽。 SubSlot	僅適用於名為 'SlotSubSlot' 的複合插槽內名為 " 的子插槽
意圖：插槽。 *	適用於名為 'Slot' 的複合插槽內的任何子插槽
意圖：*。 SubSlot	僅適用於任何複合插槽內名為SubSlot " 的子插槽
意圖：* 。 *	適用於任何複合槽內的任何子槽

中斷行為

您可以設定機器人的中斷行為。該屬性是 Amazon Lex V2 定義。

允許中斷

```
x-amz-lex:allow-interrupt:<intentName>:<slotName>
```

定義使用者是否可以中斷 Amazon Lex V2 機器人播放的提示。您可以選擇性地將其關閉。

預設值：真

語音輸入逾時

您可以使用工作階段屬性來設定與機器人的語音互動的逾時值。這些屬性是 Amazon Lex V2 定義。這些屬性可讓您指定 Amazon Lex V2 在收集輸入語音之前等待客戶完成講話的時間長度。

所有這些屬性都在`x-amz-lex:audio`命名空間中。

最大語音長度

```
x-amz-lex:audio:max-length-ms:<intentName>:<slotName>
```

定義 Amazon Lex V2 在截斷語音輸入並將語音傳回至應用程式之前等待的時間長度。如果您預期回應較長，或是想讓客戶有更多時間提供資訊，則可以增加輸入的長度。

預設值：13,000 毫秒 (13 秒)。最大值為 15,000 毫秒（ 15 秒 ）

如果您將`max-length-ms`屬性設定為超過 15,000 毫秒，該值將預設為 15,000 毫秒。

語音逾時

```
x-amz-lex:audio:start-timeout-ms:<intentName>:<slotName>
```

在假設客戶不會說話之前，機器人會等待多長時間。在客戶可能需要更多時間來尋找或回收說話之前的資訊，您可以增加時間。例如，您可能希望讓客戶有時間取出信用卡，以便他們可以輸入號碼。

預設值：4,000 毫秒 (4 秒)

靜音逾時

```
x-amz-lex:audio:end-timeout-ms:<intentName>:<slotName>
```

客戶停止說話以假設語音完成後，機器人會等待多長時間。您可以增加預期的靜音時間，同時提供輸入的情況下的時間。

預設值：600 毫秒 (0.6 秒)

允許音訊輸入

```
x-amz-lex:allow-audio-input:<intentName>:<slotName>
```

您可以啟用此屬性，以便機器人僅通過音頻模式接受用戶輸入。如果此標誌設置為 false，機器人將不接受音頻輸入。根據預設，值是設為 true。

預設值：真

文字輸入逾時

使用下列工作階段屬性來指定您的機器人在文字交談模式下的行為方式。

此屬性位於`x-amz-lex:text`命名空間中。

啟動逾時閾值

```
x-amz-lex:text:start-timeout-ms:<intentName>:<slotName>
```

在重新提示客戶輸入文字之前，機器人會等待多久。如果您想讓客戶有更多時間尋找或召回資訊，然後再提供文字輸入，您可以增加時間。例如，您可能想要在客戶的訂單詳細資訊。或者，您也可以降低臨界值，以提早提示客戶。

預設值：30,000 毫秒 (30 秒)

DTMF 輸入的配置

使用下列工作階段屬性來指定 Amazon Lex V2 機器人在使用音訊交談時如何回應 DTMF 輸入。

所有這些屬性都在`x-amz-lex:dtmf`命名空間中。

刪除字元

```
x-amz-lex:dtmf:deletion-character:<intentName>:<slotName>
```

會清除累積的 DTMF 數字並立即結束輸入的 DTMF 字元。

預設值：*

結束字元

```
x-amz-lex:dtmf:end-character:<intentName>:<slotName>
```

立即結束輸入的 DTMF 字元。如果使用者未按下此字元，則輸入會在結束逾時後結束。

預設值：#

結束逾時

```
x-amz-lex:dtmf:end-timeout-ms:<intentName>:<slotName>
```

在假設輸入已結束之前，機器人應該從最後一個 DTMF 字元輸入中等待多長時間。

預設值：5000 毫秒 (5 秒)

每個話語的 DTMF 位數上限

```
x-amz-lex:dtmf:max-length:<intentName>:<slotName>
```

話語中允許的 DTMF 位數上限。例如，您可以將此值設定為 16，以限制可輸入信用卡號碼的字元數。這個值無法增加。

預設值：1024 個字元

允許 DTMF 輸入

您可以使用工作階段屬性來設定機器人可以接受的輸入類型。這些屬性是 Amazon Lex V2 定義。

```
x-amz-lex:allow-dtmf-input:<intentName>:<slotName>
```

您可以啟用此屬性，讓機器人透過 DTMF 模式接受使用者輸入。如果此旗標設定為 false，機器人將不會接受 DTMF 輸入。根據預設，值是設為 true。

預設值：真

匯入和匯出

您可以匯出機器人定義、機器人地區設定或自訂字彙，然後將其匯入回來建立新資源或覆寫AWS帳號中的現有資源。例如，您可以從測試帳戶匯出機器人，然後在生產帳戶中建立機器人的副本。您也可以將機器人從一個AWS區域複製到另一個區域。

您可以在匯入之前變更匯出資源的資源。例如，您可以導出機器人，然後編輯插槽的 JSON 文件，以添加或刪除插槽值從特定插槽引出語音。編輯完定義後，您可以匯入修改過的檔案。

主題

匯出中

您可以使用主控台或CreatExport作業匯出機器人、機器人地區設定或自訂詞彙。您可以指定要匯出的資源，並且可以提供選擇性密碼，以協助在開始匯出時保護 .zip 檔案。下載 .zip 檔案之後，您必須先使用密碼來存取檔案，才能使用該檔案。如需詳細資訊，請參閱匯入或匯出時使用密碼。

匯出是一種非同步作業。開始匯出之後，您可以使用主控台或DescribeExport作業監視匯出進度。匯出完成後，主控台或DescribeExport作業會顯示狀態COMPLETED，主控台會將匯出的 .zip 檔案下載至您的瀏覽器。如果您使用此DescribeExport操作，Amazon Lex V2 會提供預先簽署的 Amazon S3 URL，您可以在其中下載匯出結果。下載 URL 只能使用五分鐘，但您可以通過再次調用DescribeExport操作來獲得新的 URL。

您可以使用控制台或ListExports操作查看資源的導出歷史記錄。結果會顯示匯出及其目前狀態。在歷史記錄中可以使用匯出 7 天。

當您匯出機器人的Draft版本或機器人地區設定時，JSON 檔案中的定義可能處於不一致的狀態，因為可以在匯出過程中變更機器人或機器人地區設定的Draft版本。如果Draft版本在匯出時發生變更，則匯出檔案中可能不會包含變更。

匯出機器人地區設定時，Amazon Lex 會匯出定義地區設定的所有資訊，包括地區設定、自訂詞彙、意圖、位置類型和位置。

匯出機器人時，Amazon Lex 會匯出為機器人定義的所有語言環境，包括意圖、插槽類型和位置。以下項目不會與機器人一起匯出：

- 機器人別名
- 與機器人相關聯的角色 ARN
- 與機器人和機器人別名關聯的標籤
- 與機器人別名關聯的 Lambda 程式碼掛接

當您匯入機器人時，角色 ARN 和標籤會作為請求參數輸入。如有必要，您需要在匯入後建立機器人別名並指派 Lambda 程式碼掛接。

您可以使用主控台或`DeleteExport`作業移除匯出和相關聯的 .zip 檔案。

如需使用主控台匯出機器人的範例，請參閱匯出機器人 (主控台)。

匯出所需的 IAM 許可

若要匯出機器人、機器人語言環境和自訂字彙，執行匯出的使用者必須具有下列 IAM 許可。

API	• 必要的 IAM 動作	資源
CreateExport	• CreateExport	機器人
UpdateExport	• UpdateExport	機器人
DescribeExport	• DescribeExport • DescribeBot • DescribeCustomVocabulary • DescribeLocale • DescribeIntent • DescribeSlot • DescribeSlotType • ListLocale • ListIntent • ListSlot • ListSlotType	機器人

API	• 必要的 IAM 動作	資源
DescribeExport用於自訂字彙	• DescribeExport • DescribeCustomVocabulary	機器人
DeleteExport	• DeleteExport	機器人
ListExports	• ListExports	*

如需 IAM 政策範例，請參閱 允許使用者匯出機器人和機器人語言環境 。

匯出機器人 (主控台)

您可以從機器人清單、版本清單或版本詳細資料頁面匯出機器人。當您選擇版本時，Amazon Lex V2 會匯出該版本。以下說明假設您開始從機器人列表中導出機器人，但是當您從版本開始時，步驟是相同的。

使用主控台匯出機器人

1. 登入AWS Management Console並開啟亞馬遜萊克斯 V2 主控台，網址為 https://console.aws.amazon.com/lexv2/home。
2. 從機器人列表中，選擇要導出的機器人。
3. 從「動作」中選擇「匯出」。
4. 選擇機器人版本、平台和匯出格式。
5. (選擇性) 輸入 .zip 檔案的密碼。提供密碼有助於保護輸出歸檔。
6. 選擇 Export (匯出)。

開始匯出後，您會返回機器人清單。若要監視匯出的進度，請使用匯入/匯出記錄清單。匯出狀態為「完成」時，主控台會自動將 .zip 檔案下載到您的電腦。

若要再次下載匯出，請在匯入/匯出清單中選擇匯出，然後選擇 [下載]。您可以為下載的 .zip 檔案提供密碼。

若要匯出機器人語言

1. 登入AWS Management Console並開啟亞馬遜萊克斯 V2 主控台，網址為 https://console.aws.amazon.com/lexv2/home。

2. 從機器人列表中，選擇要導出其語言的機器人。
3. 從新增語言中，選擇檢視語言。
4. 從 [所有語言] 清單中，選擇要匯出的語言。
5. 從「動作」中選擇「匯出」。
6. 選擇機器人版本、平台和格式。
7. (選擇性) 輸入 .zip 檔案的密碼。提供密碼有助於保護輸出歸檔。
8. 選擇 Export (匯出)。

開始匯出之後，您會返回語言清單。若要監視匯出的進度，請使用匯入/匯出記錄清單。匯出狀態為「完成」時，主控台會自動將 .zip 檔案下載到您的電腦。

若要再次下載匯出，請在匯入/匯出清單中選擇匯出，然後選擇 [下載]。您可以為下載的 .zip 檔案提供密碼。

匯入中

若要使用主控台匯入先前匯出的機器人、機器人地區設定或自訂字彙，請提供本機電腦上的檔案位置，以及解除鎖定檔案的選用密碼。如需範例，請參閱 導入機器人（控制台）。

使用 API 時，匯入資源需要三個步驟：

1. 使用`CreateUploadUrl`作業建立上傳 URL。您不需要在使用主控台時建立上傳 URL。
2. 上傳包含資源定義的 .zip 檔案。
3. 使用`StartImport`操作開始導入。

上傳網址是具有寫入權限的預先簽署 Amazon S3 URL。URL 在產生後可使用五分鐘。如果您使用密碼保護 .zip 檔案，則必須在開始匯入時提供密碼。如需詳細資訊，請參閱匯入或匯出時使用密碼。

匯入是一個非同步處理程序。您可以使用主控台或作業監視匯入的進`DescribeImport`度。

匯入機器人或機器人地區設定時，匯入檔案中的資源名稱與 Amazon Lex V2 中的現有資源名稱之間可能會發生衝突。亞馬遜萊克斯 V2 可以通過三種方式處理衝突：

- 發生衝突時失敗 — 匯入會停止，且不會從匯入 .zip 檔案匯入任何資源。
- 覆寫 — Amazon Lex V2 會從匯入的 .zip 檔案匯入所有資源，並以匯入檔案中的定義取代任何現有資源。

- 附加 — Amazon Lex V2 會從匯入的 .zip 檔案匯入所有資源，並使用匯入檔案中的定義新增至任何現有資源。這僅適用於機器人地區設定。

您可以使用控制台或`ListImports`操作查看導入到資源的列表。匯入會在清單中保留七天。您可以使用控制台或`DescribeImport`操作來查看有關特定導入的詳細信息。

您也可以使用主控台或`DeleteImport`作業移除匯入和相關聯的 .zip 檔案。

如需使用主控台匯入機器人的範例，請參閱導入機器人（控制台）。

匯入所需的 IAM 許可

若要匯入機器人、機器人語言環境和自訂字彙，執行匯入的使用者必須具有下列 IAM 許可。

API	必要的 IAM 動作	資源
CreateUploadUrl	• CreateUploadUrl	*
StartImport機器人和機器人地區設定	• StartImport • IAM：PassRole • CreateBot • CreateCustomVocabulary • CreateLocale • CreateIntent • CreateSlot • CreateSlotType • UpdateBot • UpdateCustomVocabulary • UpdateLocale • UpdateIntent • UpdateSlot • UpdateSlotType • DeleteBot • DeleteCustomVocabulary • DeleteLocale	1. 要導入新的機器人：機器人，機器人別名。 2. 若要覆寫現有的機器人：機器人。 3. 若要匯入新的語言環境：機器人。

API	必要的 IAM 動作	資源
	• DeleteIntent • DeleteSlot • DeleteSlotType	
StartImport用於自訂字彙	• StartImport • CreateCustomVocabulary • DeleteCustomVocabulary • UpdateCustomVocabulary	機器人
DescribeImport	• DescribeImport	機器人
DeleteImport	• DeleteImport	機器人
ListImports	• ListImports	*

如需 IAM 政策範例，請參閱 允許使用者匯入機器人和機器人語言環境 。

導入機器人（控制台）

使用主控台匯入機器人

1. 登入AWS Management Console並開啟亞馬遜萊克斯 V2 主控台，網址為 https://console.aws.amazon.com/lexv2/home。
2. 從「動作」中選擇「匯入」。
3. 在輸入檔案中，為機器人命名，然後選擇包含定義機器人之 JSON 檔案的 .zip 檔案。
4. 如果 .zip 檔案受密碼保護，請輸入 .zip 檔案的密碼。密碼保護歸檔是可選的，但它有助於保護內容。
5. 建立或輸入為機器人定義許可的 IAM 角色。
6. 指出您的機器人是否受《兒童在線隱私保護法》（COPPA）的約束。
7. 為您的機器人提供閒置逾時設定。如果您未提供值，則會使用 zip 檔案中的值。如果 .zip 檔案不包含逾時設定，Amazon Lex V2 會使用預設值 300 秒 (5 分鐘)。
8. （可選）為您的機器人添加標籤。

9. 選擇是否要警告使用相同名稱覆寫現有機器人。如果您啟用警告，如果您要匯入的機器人會覆寫現有的機器人，您會收到警告，且不會匯入機器人。如果停用警告，匯入的機器人會以相同的名稱取代現有的機器人。
10. 選擇 Import (匯入)。

開始匯入之後，您會返回機器人清單。若要監視匯入的進度，請使用匯入/匯出記錄清單。當匯入狀態為「完成」時，您可以從機器人清單中選擇要修改或建置機器人的機器人。

若要匯入機器人語言

1. 登入AWS Management Console並開啟亞馬遜萊克斯 V2 主控台，網址為 https://console.aws.amazon.com/lexv2/home。
2. 從機器人列表中，選擇要導入語言的機器人。
3. 從新增語言中，選擇檢視語言。
4. 從動作中，選擇匯入。
5. 在「輸入檔案」中，選擇包含要匯入語言的檔案。如果您保護了 .zip 檔案，請在「密碼」中提供密碼。
6. 在語言中，選擇要匯入為的語言。語言不一定要與匯入檔案中的語言相符。您可以將意圖從一種語言複製到另一種語言。
7. 在「語音」中，選擇要用於語音互動的 Amazon Polly 語音，或針對純文字機器人選擇「無」。
8. 在可信度分數閾值中，輸入 Amazon Lex V2 在傳回替代意圖時插入`AMAZON.KendraSearchIntent`、或兩者的閾值。`AMAZON.FallbackIntent`
9. 選擇是否要警告覆寫現有語言。如果您啟用警告，如果您要匯入的語言會覆寫現有語言，您會收到警告，且不會匯入語言。如果停用警告，匯入的語言會取代現有的語言。
10. 選擇「匯入」以開始匯入語言。

開始匯入之後，您會返回語言清單。若要監視匯入的進度，請使用匯入/匯出記錄清單。當匯入狀態為「完成」時，您可以從機器人清單中選擇要修改或建置機器人的語言。

匯入或匯出時使用密碼

Amazon Lex V2 可以用密碼保護您的匯出存檔，或使用標準 .zip 檔案壓縮來讀取受保護的匯入存檔。您應始終使用密碼保護導入和導出存檔。

Amazon Lex V2 會將您的匯出存檔傳送到 S3 儲存貯體，您可以使用預先簽署的 S3 URL 存取該存檔。該網址只能使用五分鐘。任何具有下載 URL 存取權的人都可以使用歸檔。若要協助保護封存中的資料，請在匯出資源時提供密碼。如果您需要在 URL 過期後獲取歸檔，則可以使用控制台或`DescribeExport`操作來獲取新的 URL。

如果您遺失匯出歸檔的密碼，您可以從匯入/匯出歷史記錄表格中選擇「下載」，或使用`UpdateExport`作業來為現有檔案建立新密碼。如果您在匯出的歷史記錄表格中選擇 [下載]，但未提供密碼，Amazon Lex V2 會下載未受保護的 zip 檔案。

用於匯入和匯出的 JSON 格式

您可以使用包含描述資源各部分之 JSON 結構的 .zip 檔案，從 Amazon Lex V2 匯入和匯出機器人、機器人語言環境或自訂字彙。當您匯出資源時，Amazon Lex V2 會建立 .zip 檔案，並使用 Amazon S3 預先簽署的 URL 將其提供給您。匯入資源時，您必須建立包含 JSON 結構的 .zip 檔案，並將其上傳到 S3 預先簽署的 URL。

當您匯出機器人時，Amazon Lex 會在 .zip 檔案中建立下列目錄結構。匯出機器人地區設定時，只會匯出地區設定下的結構。當您匯出自訂字彙時，只會匯出自訂字彙下的結構。

```
BotName_BotVersion_ExportID_LexJson.zip
            -or-
BotName_BotVersion_LocaleId_ExportId_LEX_JSON.zip
        --> manifest.json
        --> BotName
        ----> Bot.json
        ----> BotLocales
        ------> Locale_A
        --------> BotLocale.json
        --------> Intents
        ----------> Intent_A
        ------------> Intent.json
        ------------> Slots
        --------------> Slot_A
        ----------------> Slot.json
        --------------> Slot_B
        ----------------> Slot.json
        ----------> Intent_B
                        ...
        --------> SlotTypes
        ----------> SlotType_A
        ------------> SlotType.json
```

```
        ----------> SlotType_B
                        ...
        --------> CustomVocabulary
        ------------> CustomVocabulary.json

        ------> Locale_B
                        ...
```

清單檔案結構

資訊清單檔案包含匯出檔案的中繼資料。

```
{
    "metadata": {
        "schemaVersion": "1.0",
        "fileFormat": "LexJson",
        "resourceType": "Bot | BotLocale | CustomVocabulary"
    }
}
```

機器人檔案結構

機器人檔案包含機器人的組態資訊。

```
{
    "name": "BotName",
    "identifier": "identifier",
    "version": "number",
    "description": "description",
    "dataPrivacy": {
        "childDirected": true | false
    },
    "idleSessionTTLInSeconds": seconds
}
```

機器人地區設定檔案

機器人地區設定檔案包含機器人地區設定或語言的描述。當您匯出機器人時，.zip 檔案中可能會有多個機器人地區設定檔案。當您匯出機器人地區設定時，zip 檔案中只有一個地區設定。

```
{
```

```
    "name": "locale name",
    "identifier": "locale ID",
    "version": "number",
    "description": "description",
    "voiceSettings": {
        "voiceId": "voice",
        "engine": "standard | neural
    },
    "nluConfidenceThreshold": number
}
```

意圖文件結構

意圖檔案包含意圖的組態資訊。在特定地區設定中，.zip 檔案中的每個意圖都有一個意圖檔案。

以下是範例BookTrip機器人中BookCar意圖的 JSON 結構範例。如需意圖之 JSON 結構的完整範例，請參閱CreateIntent作業。

```
{
    "name": "BookCar",
    "identifier": "891RWHHICO",
    "description": "Intent to book a car.",
    "parentIntentSignature": null,
    "sampleUtterances": [
        {
            "utterance": "Book a car"
        },
        {
            "utterance": "Reserve a car"
        },
        {
            "utterance": "Make a car reservation"
        }
    ],
    "intentConfirmationSetting": {
        "confirmationPrompt": {
            "messageGroupList": [
                {
                    "message": {
                        "plainTextMessage": {
                            "value": "OK, I have you down for a {CarType} hire in
 {PickUpCity} from {PickUpDate} to {ReturnDate}.  Should I book the reservation?"
                        },
```

```
                        "ssmlMessage": null,
                        "customPayload": null,
                        "imageResponseCard": null
                    },
                    "variations": null
                }
            ],
            "maxRetries": 2
        },
        "declinationResponse": {
            "messageGroupList": [
                {
                    "message": {
                        "plainTextMessage": {
                            "value": "OK, I have cancelled your reservation in
progress."
                        },
                        "ssmlMessage": null,
                        "customPayload": null,
                        "imageResponseCard": null
                    },
                    "variations": null
                }
            ]
        }
    },
    "intentClosingSetting": null,
    "inputContexts": null,
    "outputContexts": null,
    "kendraConfiguration": null,
    "dialogCodeHook": null,
    "fulfillmentCodeHook": null,
    "slotPriorities": [
        {
            "slotName": "DriverAge",
            "priority": 4
        },
        {
            "slotName": "PickUpDate",
            "priority": 2
        },
        {
            "slotName": "ReturnDate",
            "priority": 3
```

```
        },
        {
            "slotName": "PickUpCity",
            "priority": 1
        },
        {
            "slotName": "CarType",
            "priority": 5
        }
    ]
}
```

插槽文件結構

槽檔案包含意圖中狹槽的組態資訊。在 .zip 檔案中，針對特定語言環境中的意圖而定義的每個插槽，都有一個插槽檔案。

下列範例是插槽的 JSON 結構，可讓客戶在範BookTrip例機器人中選擇想要租用的BookCar汽車類型。如需插槽 JSON 結構的完整範例，請參閱CreateSlot作業。

```
{
    "name": "CarType",
    "identifier": "KDHJWNGZGC",
    "description": "Type of car being reserved.",
    "multipleValuesSetting": {
        "allowMutlipleValues": false
    },
    "slotTypeName": "CarTypeValues",
    "obfuscationSetting": null,
    "slotConstraint": "Required",
    "defaultValueSpec": null,
    "slotValueElicitationSetting": {
        "promptSpecification": {
            "messageGroupList": [
                {
                    "message": {
                        "plainTextMessage": {
                            "value": "What type of car would you like to rent?  Our
 most popular options are economy, midsize, and luxury"
                        },
                        "ssmlMessage": null,
                        "customPayload": null,
                        "imageResponseCard": null
```

```
                    },
                    "variations": null
                }
            ],
            "maxRetries": 2
        },
        "sampleValueElicitingUtterances": null,
        "waitAndContinueSpecification": null,
    }
}
```

下列範例顯示複合插槽的 JSON 結構。

```
{
  "name": "CarType",
  "identifier": "KDHJWNGZGC",
  "description": "Type of car being reserved.",
  "multipleValuesSetting": {
      "allowMutlipleValues": false
  },
  "slotTypeName": "CarTypeValues",
  "obfuscationSetting": null,
  "slotConstraint": "Required",
  "defaultValueSpec": null,
  "slotValueElicitationSetting": {
      "promptSpecification": {
          "messageGroupList": [
              {
                  "message": {
                      "plainTextMessage": {
                          "value": "What type of car would you like to rent?  Our most
 popular options are economy, midsize, and luxury"
                      },
                      "ssmlMessage": null,
                      "customPayload": null,
                      "imageResponseCard": null
                  },
                  "variations": null
              }
          ],
          "maxRetries": 2
      },
      "sampleValueElicitingUtterances": null,
```

```
      "waitAndContinueSpecification": null,
  },
  "subSlotSetting": {
    "slotSpecifications": {
      "firstname": {
        "valueElicitationSetting": {
          "promptSpecification": {
            "allowInterrupt": false,
            "messageGroupsList": [
              {
                "message": {
                  "imageResponseCard": null,
                  "ssmlMessage": null,
                  "customPayload": null,
                  "plainTextMessage": {
                    "value": "please provide firstname"
                  }
                },
                "variations": null
              }
            ],
            "maxRetries": 2,
            "messageSelectionStrategy": "Random"
          },
          "defaultValueSpecification": null,
          "sampleUtterances": [
            {
              "utterance": "my name is {firstName}"
            }
          ],
          "waitAndContinueSpecification": null
        },
        "slotTypeId": "AMAZON.FirstName"
      },
      "eyeColor": {
        "valueElicitationSetting": {
          "promptSpecification": {
            "allowInterrupt": false,
            "messageGroupsList": [
              {
                "message": {
                  "imageResponseCard": null,
                  "ssmlMessage": null,
                  "customPayload": null,
```

```
                "plainTextMessage": {
                  "value": "please provide eye color"
                }
              },
              "variations": null
            }
          ],
          "maxRetries": 2,
          "messageSelectionStrategy": "Random"
        },
        "defaultValueSpecification": null,
        "sampleUtterances": [
          {
            "utterance": "eye color is {eyeColor}"
          },
          {
            "utterance": "I have eyeColor eyes"
          }
        ],
        "waitAndContinueSpecification": null
      },
      "slotTypeId": "7FEVCB2PQE"
    }
  },
  "expression": "(firstname OR eyeColor)"
 }
}
```

插槽類型檔案結構

插槽類型檔案包含語言或地區設定中使用的自訂插槽類型的組態資訊。在特定地區設定中，.zip 檔案中的每個自訂插槽類型都有一個插槽類型檔案。

以下是插槽類型的 JSON 結構，其中列出了BookTrip示例機器人中可用的汽車類型。如需插槽類型之 JSON 結構的完整範例，請參閱CreateSlotType作業。

```
{
    "name": "CarTypeValues",
    "identifier": "T1YUHGD9ZR",
    "description": "Enumeration representing possible types of cars available for
 hire",
    "slotTypeValues": [{
        "synonyms": null,
```

```
        "sampleValue": {
            "value": "economy"
        }
    }, {
        "synonyms": null,
        "sampleValue": {
            "value": "standard"
        }
    }, {
        "synonyms": null,
        "sampleValue": {
            "value": "midsize"
        }
    }, {
        "synonyms": null,
        "sampleValue": {
            "value": "full size"
        }
    }, {
        "synonyms": null,
        "sampleValue": {
            "value": "luxury"
        }
    }, {
        "synonyms": null,
        "sampleValue": {
            "value": "minivan"
        }
    }],
    "parentSlotTypeSignature": null,
    "valueSelectionSetting": {
        "resolutionStrategy": "TOP_RESOLUTION",
        "advancedRecognitionSetting": {
            "audioRecognitionStrategy": "UseSlotValuesAsCustomVocabulary"
        },
        "regexFilter": null
    }
}
```

下列範例顯示複合插槽類型的 JSON 結構。

```
{
  "name": "CarCompositeType",
```

```
  "identifier": "TPA3CC9V",
  "description": null,
  "slotTypeValues": null,
  "parentSlotTypeSignature": null,
  "valueSelectionSetting": {
    "regexFilter": null,
    "resolutionStrategy": "CONCATENATION"
  },
  "compositeSlotTypeSetting": {
    "subSlots": [
      {
        "name": "model",
        "slotTypeId": "MODELTYPEID" # custom slot type Id for model
      },
      {
        "name": "city",
        "slotTypeId": "AMAZON.City"
      },
      {
        "name": "country",
        "slotTypeId": "AMAZON.Country"
      },
      {
        "name": "make",
        "slotTypeId": "MAKETYPEID" # custom slot type Id for make
      }
    ]
  }
}
```

以下是一種插槽類型，它使用自定義語法來了解客戶的話語。如需詳細資訊，請參閱語法插槽類型。

```
{
  "name": "custom_grammar",
  "identifier": "7KEAQIQKPX",
  "description": "Slot type using a custom grammar",
  "slotTypeValues": null,
  "parentSlotTypeSignature": null,
  "valueSelectionSetting": null,
  "externalSourceSetting": {
    "grammarSlotTypeSetting": {
      "source": {
        "kmsKeyArn": "arn:aws:kms:Region:123456789012:alias/customer-grxml-key",
```

```
        "s3BucketName": "grxml-test",
        "s3ObjectKey": "grxml_files/grammar.grxml"
      }
    }
  }
}
```

自定義詞彙文件結構。

自訂字彙檔案包含單一語言或地區設定的自訂字彙中的項目。每個具有自訂字彙的語言環境的 .zip 檔案中都有一個自訂字彙檔案。

以下是接收餐廳訂單的機器人的自訂詞彙檔案。機器人中的每個語言環境都有一個文件。

```
{
    "customVocabularyItems": [
        {
            "weight": 3,
            "phrase": "wafers"
        },
        {
            "weight": null,
            "phrase": "extra large"
        },
        {
            "weight": null,
            "phrase": "cremini mushroom soup"
        },
        {
            "weight": null,
            "phrase": "ramen"
        },
        {
            "weight": null,
            "phrase": "orzo"
        }
    ]
}
```

標記 資源

為協助您管理 Amazon Lex V2 機器人和機器人別名，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。每個標籤皆包含鍵與值。

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

- 識別和組織您的 AWS 資源。許多AWS資源支援標記，因此您可以將同一標籤分配給不同服務的資源，指出資源是相同的標籤。例如，您可以標記機器人和使用相同的標籤的 Lambda 函數。
- 配置成本。您可以在 AWS Billing and Cost Management 儀表板上啟用標籤。AWS 會使用標籤分類您的成本，並交付每月成本配置報告給您。對於 Amazon Lex V2，您可以使用別名特定的標籤為每個別名分配成本。如需詳細資訊，請參閱《AWS Billing and Cost Management 使用者指南》中的使用成本配置標籤。
- 控制對 資源的存取。您可以搭配 Amazon Lex V2 使用標籤來建立政策以控制對 Amazon Lex V2 資源的存取。這些政策可附加至 IAM 角色或使用者，以啟用標籤型存取控制。

您可以使用AWS Management Console、或 Amazon Lex V2 API 來使用標籤。AWS Command Line Interface

標記您的資源

如果您使用 Amazon Lex V2 主控台，您可以在建立資源時標記資源，或稍後新增標籤。您也可以使用主控台來更新或移除現有的標籤。

如果您使用的是AWS CLI或 Amazon Lex V2 API，您可以使用下列操作來管理資源的標籤：

- CreateBot和 CreateBotAlias— 在建立機器人或機器人別名時套用標籤。
- ListTagsForResource— 檢視與資源相關聯的標籤。
- TagResource— 在現有資源上新增和修改標籤。
- UntagResource— 從資源移除標籤。

Amazon Lex V2 中的下列資源支援標記：

- 機器人 — 使用 Amazon Resource Name (ARN)，如下所示：
  - `arn:aws:lex:${Region}:${account}:bot/${bot-id}`

- 機器人別名 — 使用如下所示的 ARN：
  - arn:aws:lex:*${Region}*:*${account}*:bot-alias/*${bot-id}*/*${bot-alias-id}*

bot-id和bot-alias-id值為大寫字母數字字串，長度為 10 個字元。

標籤限制

下列基本限制適用於 Amazon Lex V2 資源上的標籤：

- 金鑰數目上限 — 50 個使用主控台，200 個使用 API
- 最大金鑰長度 - 128 個字元
- 最大值長度 - 256 個字元
- 金鑰與值的有效字元 — a-z、A-Z、0-9、空格和下列字元：_.:/= +-及 @
- 金鑰和值會區分大小寫
- 請不要使用aws:做為金鑰的字首，它已保留供AWS使用

標記資源 (主控台)

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

在建立機器人時新增標籤

1. 請登入，AWS Management Console並在 https://console.aws.amazon.com/lex/ 開啟 Amazon Lex 主控台。
2. 選擇建立機器人。
3. 在 [設定機器人設定] 的 [進階設定] 區段中，選擇 [新增標籤]。您可以將標籤新增至 Bot 和TestBotAlias別名。
4. 選擇「下一步」繼續建立您的機器人。

在建立機器人別名時新增標籤

1. 請登入，AWS Management Console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. 從左側選單中選擇別名，然後選擇建立別名。
4. 在「一般資訊」中，選擇「從標籤新增標籤」。
5. 選擇 建立 。

新增、移除或修改現有機器人上的標籤

1. 請登入，AWS Management Console並在 https://console.aws.amazon.com/lex/ 開啟 Amazon Lex 主控台。
2. 選擇您想要修改的機器人。
3. 從左側選單中選擇 [設定]，然後選擇 [編輯]。
4. 在「標籤」中進行變更。
5. 選擇 [儲存] 以儲存您對機器人的變更。

若要新增、移除或修改現有別名上的標籤

1. 請登入，AWS Management Console並在 https://console.aws.amazon.com/lex/ 開啟 Amazon Lex 主控台。
2. 選擇您想要修改的機器人。
3. 從左側功能表中選擇別名，然後從別名清單中選擇要修改的別名。
4. 從別名詳細資料的標籤中，選擇修改標籤。
5. 在「管理標籤」中進行變更。
6. 選擇「儲存」以儲存對別名所做的變更。

Amazon Lex V2 的安全

雲安全 AWS 是最高的優先級。身為 AWS 客戶，您可以從資料中心和網路架構中獲益，這些架構是為了滿足對安全性最敏感的組織的需求而建置的。

安全是 AWS 與您之間共同承擔的責任。共同責任模型將其描述為雲端的安全性和雲端中的安全性：

- 雲端的安全性 — AWS 負責保護在 AWS 雲端中執行 AWS 服務的基礎架構。 AWS 還為您提供可以安全使用的服務。若要了解適用於 Amazon Lex V2 的合規計劃，請參閱合規計劃 AWS 服務範圍內的合規計。
- 雲端中的安全性 — 您的責任取決於您使用的 AWS 服務。您也必須對其他因素負責，包括資料的機密性、您公司的要求和適用法律和法規。

本文件可協助您了解如何在使用 Amazon Lex V2 時套用共同的責任模型。下列主題說明如何設定 Amazon Lex V2 以符合安全和合規目標。您也會學到如何使用其他可協助您監控和保護 Amazon Lex V2 資源的 AWS 服務。

主題

- Amazon Lex V2 中的數據保護
- Amazon Lex V2 的身分和存取管理
- 在 Amazon Lex V2 中記錄和監控
- Amazon Lex V2 的合規驗證
- Amazon Lex V2 的彈性
- Amazon Lex V2 的基礎設施
- Amazon Lex V2 和接口虛擬私人雲端端點 ()AWS PrivateLink

Amazon Lex V2 中的數據保護

Amazon Lex V2 符合共同責任模型 AWS 共同責任模，其中包括資料保護的法規和準則。 AWS 負責保護運行所有 AWS 服務的全球基礎設施。 AWS 保持對此基礎架構上託管的數據的控制，包括用於處理客戶內容和個人數據的安全配置控制。 AWS 作為資料控制者或資料處理者的客戶和 APN 合作夥伴負責放置在 AWS 雲端的任何個人資料。

基於資料保護目的，我們建議您使用 AWS Identity and Access Management (IAM) 保護帳戶登入資料並設定個別使用者帳 AWS 戶，以便每位使用者僅獲得履行其工作職責所需的權限。我們也建議您採用下列方式保護資料：

- 每個帳戶均要使用多重要素驗證 (MFA)。
- 使用 SSL/TLS 與 AWS 資源進行通訊。
- 使用設定 API 和使用者活動記錄 AWS CloudTrail。
- 使用 AWS 加密解決方案，以及 AWS 服務中的所有預設安全性控制。
- 使用進階的受管安全服務 (例如 Amazon Macie)，協助探索和保護儲存在 Simple Storage Service (Amazon Simple Storage Service (Amazon S3)) 的個人資料。

我們強烈建議您絕對不要將客戶帳戶號碼等敏感的識別資訊，放在自由格式的欄位中，例如Name (名稱) 欄位。這包括當您使用主控台、API 或 AWS 開發套件使用 Amazon Lex V2 或其他 AWS 服務時。AWS CLI您輸入 Amazon Lex V2 或其他服務的任何資料都可能會被拾取以包含在診斷日誌中。當您提供外部伺服器的 URL 時，請勿在驗證您對該伺服器請求的 URL 中包含憑證資訊。

如需關於資料保護的詳細資訊，請參閱 AWS 安全部落格上的 AWS 共同責任模型和歐盟《一般資料保護規範》(GDPR) 部落格文章。

靜態加密

Amazon Lex V2 會加密其存放的使用者話語和其他資訊。

主題

- 語音樣本
- 工作階段屬性
- 請求屬性

語音樣本

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。此資訊會在靜態時加密，並且僅用於建立機器人和建立客戶體驗。

工作階段屬性

工作階段屬性包含 Amazon Lex V2 和用戶端應用程式之間傳遞的應用程式特定資訊。Amazon Lex V2 會將工作階段屬性傳遞給針對機器人設定的所有 AWS Lambda 功能。如果 Lambda 函數新增或更新工作階段屬性，Amazon Lex V2 會將新資訊傳回給用戶端應用程式。

會話屬性會在工作階段期間保持加密存放。在最後一個使用者表達用語後，您可以將工作階段設定為保持作用至少 1 分鐘，與最多達 24 小時。預設工作階段持續時間為 5 分鐘。

請求屬性

請求屬性包含請求特定的資訊並且僅適用於目前的請求。用戶端應用程式會使用請求屬性，在執行階段將資訊傳送至 Amazon Lex V2。

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 V2 使用 HTTPS 通訊協定與您的用戶端應用程式進行通訊。它使用 HTTPS 和 AWS 簽名與其他服務 (例如 Amazon Polly) 通訊，並代表您的應用程式進 AWS Lambda 行通訊。

Amazon Lex V2 的身分和存取管理

AWS Identity and Access Management (IAM) 可協助管理員安全地控制 AWS 資源存取權。 AWS 服務 IAM 管理員控制哪些人可以通過身份驗證 (登入) 和授權 (具有許可) 來使用 Amazon Lex V2 資源。IAM 是您可以使用的 AWS 服務 ，無需額外付費。

主題

- AWS Amazon Lex V2 的受管政策
- 使用 Amazon Lex V2 的服務連結角色
- 疑難排解 Amazon Lex V2 身分識別

物件

您使用 AWS Identity and Access Management (IAM) 的方式會因您在 Amazon Lex V2 中執行的工作而有所不同。

服務使用者 — 如果您使用 Amazon Lex V2 服務執行工作，則管理員會為您提供所需的登入資料和許可。當您使用更多 Amazon Lex V2 功能完成工作時，您可能需要額外的許可。了解存取許可的管理方式可協助您向管理員請求正確的許可。如果您無法存取 Amazon Lex V2 中的某個功能，請參閱疑難排解 Amazon Lex V2 身分識別。

服務管理員 — 如果您負責公司的 Amazon Lex V2 資源，您可能擁有完整的 Amazon Lex V2 存取權。判斷服務使用者應存取哪些 Amazon Lex V2 功能和資源是您的工作。接著，您必須將請求提交給您的 IAM 管理員，來變更您服務使用者的許可。檢閱此頁面上的資訊，了解 IAM 的基本概念。若要進一步了解貴公司如何將 IAM 與 Amazon Lex V2 搭配使用，請參閱亞馬遜萊克斯 V2 如何使用 IAM。

IAM 管理員 — 如果您是 IAM 管理員，您可能想要了解如何撰寫政策以管理 Amazon Lex V2 存取權的詳細資訊。若要檢視可在 IAM 中使用的 Amazon Lex V2 身分型政策範例，請參閱。Amazon Lex V2 的基於身份的政策示例

使用身分驗證

驗證是您 AWS 使用身分認證登入的方式。您必須以 IAM 使用者身分或假設 IAM 角色進行驗證 (登入 AWS)。 AWS 帳戶根使用者

您可以使用透過 AWS 身分識別來源提供的認證，以聯合身分識別身分登入。 AWS IAM Identity Center (IAM 身分中心) 使用者、貴公司的單一登入身分驗證，以及您的 Google 或 Facebook 登入資料都是聯合身分識別的範例。您以聯合身分登入時，您的管理員先前已設定使用 IAM 角色的聯合身分。當您使 AWS 用同盟存取時，您會間接擔任角色。

根據您的使用者類型，您可以登入 AWS Management Console 或 AWS 存取入口網站。如需登入的詳細資訊 AWS，請參閱AWS 登入 使用者指南中的如何登入您 AWS 帳戶的。

如果您 AWS 以程式設計方式存取，請 AWS 提供軟體開發套件 (SDK) 和命令列介面 (CLI)，以使用您的認證以加密方式簽署您的要求。如果您不使用 AWS 工具，則必須自行簽署要求。如需使用建議的方法自行簽署請求的詳細資訊，請參閱 IAM 使用者指南中的簽署 AWS API 請求。

無論您使用何種身分驗證方法，您可能都需要提供額外的安全性資訊。例如， AWS 建議您使用多重要素驗證 (MFA) 來增加帳戶的安全性。如需更多資訊，請參閱 AWS IAM Identity Center 使用者指南中的多重要素驗證和 IAM 使用者指南中的在 AWS中使用多重要素驗證 (MFA)。

AWS 帳戶 根使用者

當您建立時 AWS 帳戶，您會從一個登入身分開始，該身分可完整存取該帳戶中的所有資源 AWS 服務和資源。此身分稱為 AWS 帳戶 root 使用者，可透過使用您用來建立帳戶的電子郵件地址和密碼登入來存取。強烈建議您不要以根使用者處理日常任務。保護您的根使用者憑證，並將其用來執行只能由根使用者執行的任務。如需這些任務的完整清單，了解需以根使用者登入的任務，請參閱 IAM 使用者指南中的需要根使用者憑證的任務。

聯合身分

最佳作法是要求人類使用者 (包括需要系統管理員存取權的使用者) 使用與身分識別提供者的同盟，才能使用臨時認證 AWS 服務 來存取。

聯合身分識別是來自企業使用者目錄的使用者、Web 身分識別提供者、Identi ty Center 目錄，或使用透過身分識別來源提供的認證進行存取 AWS 服務 的任何使用者。 AWS Directory Service同盟身分存取時 AWS 帳戶，他們會假設角色，而角色則提供臨時認證。

對於集中式存取權管理，我們建議您使用 AWS IAM Identity Center。您可以在 IAM Identity Center 中建立使用者和群組，也可以連線並同步到自己身分識別來源中的一組使用者和群組，以便在所有應用程式 AWS 帳戶 和應用程式中使用。如需 IAM Identity Center 的詳細資訊，請參閱 AWS IAM Identity Center 使用者指南中的什麼是 IAM Identity Center？。

IAM 使用者和群組

IAM 使用者是您內部的身分，具 AWS 帳戶 有單一人員或應用程式的特定許可。建議您盡可能依賴暫時憑證，而不是擁有建立長期憑證 (例如密碼和存取金鑰) 的 IAM 使用者。但是如果特定使用案例需要擁有長期憑證的 IAM 使用者，建議您輪換存取金鑰。如需更多資訊，請參閱 IAM 使用者指南中的為需要長期憑證的使用案例定期輪換存取金鑰。

IAM 群組是一種指定 IAM 使用者集合的身分。您無法以群組身分簽署。您可以使用群組來一次為多名使用者指定許可。群組可讓管理大量使用者許可的程序變得更為容易。例如，您可以擁有一個名為 IAMAdmins 的群組，並給予該群組管理 IAM 資源的許可。

使用者與角色不同。使用者只會與單一人員或應用程式建立關聯，但角色的目的是在由任何需要它的人員取得。使用者擁有永久的長期憑證，但角色僅提供暫時憑證。如需進一步了解，請參閱IAM 使用者指南中的建立 IAM 使用者 (而非角色) 的時機。

IAM 角色

IAM 角色是您 AWS 帳戶 內部具有特定許可的身分。它類似 IAM 使用者，但不與特定的人員相關聯。您可以切換角色，在中暫時擔任 IAM 角色。 AWS Management Console 您可以透過呼叫 AWS CLI 或 AWS API 作業或使用自訂 URL 來擔任角色。如需使用角色的方法詳細資訊，請參閱 IAM 使用者指南中的使用 IAM 角色。

使用暫時憑證的 IAM 角色在下列情況中非常有用：

- 聯合身分使用者存取 — 如需向聯合身分指派許可，請建立角色，並為角色定義許可。當聯合身分進行身分驗證時，該身分會與角色建立關聯，並獲授予由角色定義的許可。如需有關聯合角色的相關資訊，請參閱 IAM 使用者指南中的為第三方身分提供者建立角色。如果您使用 IAM Identity Center，則需要設定許可集。為控制身分驗證後可以存取的內容，IAM Identity Center 將許可集與 IAM 中的角色相關聯。如需有關許可集的資訊，請參閱 AWS IAM Identity Center 使用者指南中的許可集。
- 暫時 IAM 使用者許可 – IAM 使用者或角色可以擔任 IAM 角色來暫時針對特定任務採用不同的許可。
- 跨帳戶存取權：您可以使用 IAM 角色，允許不同帳戶中的某人 (信任的主體) 存取您帳戶的資源。角色是授予跨帳戶存取權的主要方式。但是，對於某些策略 AWS 服務，您可以將策略直接附加到資源（而不是使用角色作為代理）。若要了解跨帳戶存取角色和以資源為基礎的政策之間的差異，請參閱 IAM 使用者指南中的 IAM 中的跨帳戶資源存取。
- 跨服務訪問 — 有些 AWS 服務 使用其他 AWS 服務功能。例如，當您在服務中進行呼叫時，該服務通常會在 Amazon EC2 中執行應用程式或將物件儲存在 Amazon Simple Storage Service (Amazon S3) 中。服務可能會使用呼叫主體的許可、使用服務角色或使用服務連結角色來執行此作業。
  - 轉寄存取工作階段 (FAS) — 當您使用 IAM 使用者或角色在中執行動作時 AWS，您會被視為主體。使用某些服務時，您可能會執行某個動作，進而在不同服務中啟動另一個動作。FAS 會使用主體呼叫的權限 AWS 服務，並結合要求 AWS 服務 向下游服務發出要求。只有當服務收到需要與其 AWS 服務 他資源互動才能完成的請求時，才會發出 FAS 請求。在此情況下，您必須具有執行這兩個動作的許可。如需提出 FAS 請求時的政策詳細資訊，請參閱《轉發存取工作階段》。
  - 服務角色 – 服務角色是服務擔任的 IAM 角色，可代表您執行動作。IAM 管理員可以從 IAM 內建立、修改和刪除服務角色。如需詳細資訊，請參閱 IAM 使用者指南中的建立角色以委派許可給 AWS 服務服務。
  - 服務連結角色 — 服務連結角色是連結至. AWS 服務服務可以擔任代表您執行動作的角色。服務連結角色會顯示在您的中， AWS 帳戶 且屬於服務所有。IAM 管理員可以檢視，但不能編輯服務連結角色的許可。
- 在 Amazon EC2 上執行的應用程式 — 您可以使用 IAM 角色來管理在 EC2 執行個體上執行的應用程式以及發出 AWS CLI 或 AWS API 請求的臨時登入資料。這是在 EC2 執行個體內儲存存取金鑰的較好方式。若要將 AWS 角色指派給 EC2 執行個體並提供給其所有應用程式，請建立連接至執行個體

的執行個體設定檔。執行個體設定檔包含該角色，並且可讓 EC2 執行個體上執行的程式取得暫時憑證。如需詳細資訊，請參閱 IAM 使用者指南中的利用 IAM 角色來授予許可給 Amazon EC2 執行個體上執行的應用程式。

如需了解是否要使用 IAM 角色或 IAM 使用者，請參閱 IAM 使用者指南中的建立 IAM 角色 (而非使用者) 的時機。

使用政策管理存取權

您可以透 AWS 過建立原則並將其附加至 AWS 身分識別或資源來控制中的存取。原則是一個物件 AWS ，當與身分識別或資源相關聯時，會定義其權限。 AWS 當主參與者 (使用者、root 使用者或角色工作階段) 提出要求時，評估這些原則。政策中的許可決定是否允許或拒絕請求。大多數原則會 AWS 以 JSON 文件的形式儲存在中。如需 JSON 政策文件結構和內容的詳細資訊，請參閱 IAM 使用者指南中的 JSON 政策概觀。

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

預設情況下，使用者和角色沒有許可。若要授予使用者對其所需資源執行動作的許可，IAM 管理員可以建立 IAM 政策。然後，管理員可以將 IAM 政策新增至角色，使用者便能擔任這些角色。

IAM 政策定義該動作的許可，無論您使用何種方法來執行操作。例如，假設您有一個允許 `iam:GetRole` 動作的政策。具有該原則的使用者可以從 AWS Management Console AWS CLI、或 AWS API 取得角色資訊。

身分型政策

身分型政策是可以附加到身分 (例如 IAM 使用者、使用者群組或角色) 的 JSON 許可政策文件。這些政策可控制身分在何種條件下能對哪些資源執行哪些動作。若要了解如何建立身分類型政策，請參閱 IAM 使用者指南中的建立 IAM 政策。

身分型政策可進一步分類成內嵌政策或受管政策。內嵌政策會直接內嵌到單一使用者、群組或角色。受管理的策略是獨立策略，您可以將其附加到您的 AWS 帳戶. 受管政策包括 AWS 受管政策和客戶管理的策略。如需了解如何在受管政策及內嵌政策間選擇，請參閱 IAM 使用者指南中的在受管政策和內嵌政策間選擇。

資源型政策

資源型政策是連接到資源的 JSON 政策文件。資源型政策的最常見範例是 IAM 角色信任政策和 Amazon S3 儲存貯體政策。在支援資源型政策的服務中，服務管理員可以使用它們來控制對特定資源

的存取權限。對於附加政策的資源，政策會定義指定的主體可以對該資源執行的動作以及在何種條件下執行的動作。您必須在資源型政策中指定主體。主參與者可以包括帳戶、使用者、角色、同盟使用者或。 AWS 服務

資源型政策是位於該服務中的內嵌政策。您無法在以資源為基礎的政策中使用 IAM 的 AWS 受管政策。

存取控制清單 (ACL)

存取控制清單 (ACL) 可控制哪些主體 (帳戶成員、使用者或角色) 擁有存取某資源的許可。ACL 類似於資源型政策，但它們不使用 JSON 政策文件格式。

Amazon S3 和 Amazon VPC 是支援 ACL 的服務範例。 AWS WAF如需進一步了解 ACL，請參閱 Amazon Simple Storage Service 開發人員指南中的存取控制清單 (ACL) 概觀。

其他政策類型

AWS 支援其他較不常見的原則類型。這些政策類型可設定較常見政策類型授予您的最大許可。

- 許可界限 – 許可範圍是一種進階功能，可供您設定身分型政策能授予 IAM 實體 (IAM 使用者或角色) 的最大許可。您可以為實體設定許可界限。所產生的許可會是實體的身分型政策和其許可界限的交集。會在 `Principal` 欄位中指定使用者或角色的資源型政策則不會受到許可界限限制。所有這類政策中的明確拒絕都會覆寫該允許。如需許可界限的詳細資訊，請參閱 IAM 使用者指南中的 IAM 實體許可界限。
- 服務控制策略（SCP）— SCP 是 JSON 策略，用於指定中組織或組織單位（OU）的最大權限。AWS Organizations AWS Organizations 是一種用於分組和集中管理您企業擁 AWS 帳戶 有的多個服務。若您啟用組織中的所有功能，您可以將服務控制政策 (SCP) 套用到任何或所有帳戶。SCP 限制成員帳戶中實體的權限，包括每個 AWS 帳戶根使用者帳戶。如需 Organizations 和 SCP 的詳細資訊，請參閱 AWS Organizations 使用者指南中的 SCP 運作方式。
- 工作階段政策 – 工作階段政策是一種進階政策，您可以在透過編寫程式的方式建立角色或聯合使用者的暫時工作階段時，作為參數傳遞。所產生工作階段的許可會是使用者或角色的身分型政策和工作階段政策的交集。許可也可以來自資源型政策。所有這類政策中的明確拒絕都會覆寫該允許。如需詳細資訊，請參閱 IAM 使用者指南中的工作階段政策。

多種政策類型

將多種政策類型套用到請求時，其結果形成的許可會更為複雜、更加難以理解。要了解如何在涉及多個政策類型時 AWS 確定是否允許請求，請參閱《IAM 使用者指南》中的政策評估邏輯。

亞馬遜萊克斯 V2 如何使用 IAM

在您使用 IAM 管理 Amazon Lex V2 的存取權限之前，請先了解哪些 IAM 功能可與 Amazon Lex V2 搭配使用。

您可以搭配 Amazon Lex V2 使用的 IAM 功能

IAM 功能	Amazon Lex V2 支持
身分型政策	是
資源型政策	是
政策動作	是
政策資源	是
政策條件索引鍵	否
ACL	否
ABAC (政策中的標籤)	是
臨時憑證	否
主體許可	是
服務角色	是
服務連結角色	部分

若要深入瞭解 Amazon Lex V2 和其他 AWS 服務如何與大多數 IAM 功能搭配使用，請參閱 IAM 使用者指南中的可與 IAM 搭配使用的AWS 服務。

Amazon Lex V2 的基於身份的政策

支援身分型政策	是

身分型政策是可以連接到身分 (例如 IAM 使用者、使用者群組或角色) 的 JSON 許可政策文件。這些政策可控制身分在何種條件下能對哪些資源執行哪些動作。若要了解如何建立身分類型政策，請參閱《IAM 使用者指南》中的建立 IAM 政策。

使用 IAM 身分型政策，您可以指定允許或拒絕的動作和資源，以及在何種條件下允許或拒絕動作。您無法在身分型政策中指定主體，因為這會套用至連接的使用者或角色。如要了解您在 JSON 政策中使用的所有元素，請參閱《IAM 使用者指南》中的 IAM JSON 政策元素參考。

Amazon Lex V2 的基於身份的政策示例

若要檢視 Amazon Lex V2 以身分識別為基礎的政策範例，請參閱。Amazon Lex V2 的基於身份的政策示例

Amazon Lex V2 內的資源型政策

支援以資源基礎的政策	是

資源型政策是附加到資源的 JSON 政策文件。資源型政策的最常見範例是 IAM 角色信任政策和 Amazon S3 儲存貯體政策。在支援資源型政策的服務中，服務管理員可以使用它們來控制對特定資源的存取權限。對於附加政策的資源，政策會定義指定的主體可以對該資源執行的動作以及在何種條件下執行的動作。您必須在資源型政策中指定主體。主體可以包括使用者、角色、聯合身分使用者或 AWS 服務。

您無法將跨帳戶或跨區域政策與 Amazon Lex 搭配使用。如果您為具有跨帳戶或跨區域 ARN 的資源建立政策，Amazon Lex 會傳回錯誤訊息。

Amazon Lex 服務支援資源型政策 (稱為機器人政策) 和機器人別名政策 (附加至機器人或機器人別名)。這些原則定義哪些主參與者可以對機器人或機器人別名執行動作。

動作只能在特定資源上使用。例如，`UpdateBot`動作只能用於機器人資源，`UpdateBotAlias`動作只能用於機器人別名資源。如果您在政策中指定的動作無法用於政策中指定的資源，Amazon Lex 會傳回錯誤訊息。如需動作清單及其可搭配使用的資源，請參閱下表。

動作	支援資源型政策	資源
BuildBot地區	支援	BotId
CreateBot	否	

動作	支援資源型政策	資源
CreateBot別名	否	
CreateBotChannel [僅限權限]	支援	BotId
CreateBot地區	支援	BotId
CreateBot版本	支援	BotId
CreateExport	支援	BotId
CreateIntent	支援	BotId
CreateResource政策	支援	BotId, BotAliasId
CreateSlot	支援	BotId
CreateSlot类型	支援	BotId
CreateUpload网址	否	
DeleteBot	支援	BotId, BotAliasId
DeleteBot別名	支援	BotAlias識別碼
DeleteBotChannel [僅限權限]	支援	BotId
DeleteBot地區	支援	BotId
DeleteBot版本	支援	BotId
DeleteExport	支援	BotId
DeleteImport	支援	BotId
DeleteIntent	支援	BotId
DeleteResource政策	支援	BotId, BotAliasId
DeleteSession	支援	BotAlias識別碼

動作	支援資源型政策	資源
DeleteSlot	支援	BotId
DeleteSlot类型	支援	BotId
DescribeBot	支援	BotId
DescribeBot別名	支援	BotAlias識別碼
DescribeBotChannel [僅限權限]	支援	BotId
DescribeBot地區	支援	BotId
DescribeBot版本	支援	BotId
DescribeExport	支援	BotId
DescribeImport	支援	BotId
DescribeIntent	支援	BotId
DescribeResource政策	支援	BotId, BotAliasId
DescribeSlot	支援	BotId
DescribeSlot类型	支援	BotId
GetSession	支援	BotAlias識別碼
ListBot別名	支援	BotId
ListBotChannels [僅限權限]	支援	BotId
ListBot語言環境	支援	BotId
ListBots	否	
ListBot版本	支援	BotId
ListBuiltInIntents	否	

動作	支援資源型政策	資源
ListBuiltIntSlot类型	否	
ListExports	否	
ListImports	否	
ListIntents	支援	BotId
ListSlots	支援	BotId
ListSlot类型	支援	BotId
PutSession	支援	BotAlias識別碼
RecognizeText	支援	BotAlias識別碼
RecognizeUtterance	支援	BotAlias識別碼
StartConversation	支援	BotAlias識別碼
StartImport	支援	BotId, BotAliasId
TagResource	否	
UpdateBot	支援	BotId
UpdateBot別名	支援	BotAlias識別碼
UpdateBot地區	支援	BotId
UpdateBot版本	支援	BotId
UpdateExport	支援	BotId
UpdateIntent	支援	BotId
UpdateResource政策	支援	BotId, BotAliasId
UpdateSlot	支援	BotId

動作	支援資源型政策	資源
UpdateSlot類型	支援	BotId
UntagResource	否	

要了解如何將以資源為基礎的策略附加到機器人或機器人別名，請參閱Amazon Lex V2 的資源型政策範例。

Amazon Lex V2 內的資源型政策範例

若要檢視以資源為基礎的 Amazon Lex V2 政策範例，請參閱Amazon Lex V2 的資源型政策範例。

Amazon Lex V2 的政策操作

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哪些內容。也就是說，哪個主體在什麼条件下可以對什麼資源執行哪些動作。

JSON 政策的 `Action` 元素描述您可以用來允許或拒絕政策中存取的動作。原則動作通常與關聯的 AWS API 作業具有相同的名稱。有一些例外狀況，例如沒有相符的 API 操作的僅限許可動作。也有一些作業需要政策中的多個動作。這些額外的動作稱為相依動作。

政策會使用動作來授予執行相關聯動作的許可。

若要查看 Amazon Lex V2 動作清單，請參閱服務授權參考資料中由 Amazon Lex V2 定義的動作。

Amazon Lex V2 中的政策動作會在動作前使用下列前置詞：

```
lex
```

若要在單一陳述式中指定多個動作，請用逗號分隔。

```
"Action": [
      "lex:action1",
      "lex:action2"
```

```
        ]
```

若要檢視 Amazon Lex V2 以身分識別為基礎的政策範例，請參閱。Amazon Lex V2 的基於身份的政策示例

Amazon Lex V2 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哪些內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

`Resource` JSON 政策元素可指定要套用動作的物件。陳述式必須包含 `Resource` 或 `NotResource` 元素。最佳實務是使用其 Amazon Resource Name (ARN) 來指定資源。您可以針對支援特定資源類型的動作 (稱為資源層級許可) 來這麼做。

對於不支援資源層級許可的動作 (例如列出操作)，請使用萬用字元 (*) 來表示陳述式適用於所有資源。

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

若要查看 Amazon Lex V2 資源類型及其 ARN 的清單，請參閱服務授權參考資料中由 Amazon Lex V2 定義的資源。若要了解可以使用哪些動作指定每個資源的 ARN，請參閱 Amazon Lex V2 定義的動作。

若要檢視 Amazon Lex V2 以身分識別為基礎的政策範例，請參閱。Amazon Lex V2 的基於身份的政策示例

Amazon Lex V2 的政策條件密鑰

支援服務特定政策條件金鑰	否

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

`Condition` 元素 (或 `Condition` 區塊) 可讓您指定使陳述式生效的條件。`Condition` 元素是選用項目。您可以建立使用條件運算子的條件運算式 (例如等於或小於)，來比對政策中的條件和請求中的值。

若您在陳述式中指定多個 Condition 元素，或是在單一 Condition 元素中指定多個索引鍵， AWS 會使用邏輯 AND 操作評估他們。如果您為單一條件索引鍵指定多個值，請使用邏輯OR運算來 AWS 評估條件。必須符合所有條件，才會授與陳述式的許可。

您也可以在指定條件時使用預留位置變數。例如，您可以只在使用者使用其 IAM 使用者名稱標記時，將存取資源的許可授予該 IAM 使用者。如需更多資訊，請參閱 IAM 使用者指南中的 IAM 政策元素：變數和標籤。

AWS 支援全域條件金鑰和服務特定條件金鑰。若要查看所有 AWS 全域條件金鑰，請參閱《IAM 使用者指南》中的AWS 全域條件內容金鑰。

若要查看 Amazon Lex V2 條件金鑰清單，請參閱服務授權參考資料中適用於 Amazon Lex V2 的條件金鑰。若要了解可以使用條件金鑰的動作和資源，請參閱 Amazon Lex V2 定義的動作。

若要檢視 Amazon Lex V2 以身分識別為基礎的政策範例，請參閱。Amazon Lex V2 的基於身份的政策示例

Amazon Lex V2 中的訪問控制列表（ ACL ）

支援 ACL	否

存取控制清單 (ACL) 可控制哪些主體 (帳戶成員、使用者或角色) 擁有存取某資源的許可。ACL 類似於資源型政策，但它們不使用 JSON 政策文件格式。

以屬性為基礎的存取控制 (ABAC) 搭配 Amazon Lex V2

支援 ABAC (政策中的標籤)	是

屬性型存取控制 (ABAC) 是一種授權策略，可根據屬性來定義許可。在中 AWS，這些屬性稱為標籤。您可以將標籤附加到 IAM 實體 (使用者或角色) 和許多 AWS 資源。為實體和資源加上標籤是 ABAC 的第一步。您接著要設計 ABAC 政策，允許在主體的標籤與其嘗試存取的資源標籤相符時操作。

ABAC 在成長快速的環境中相當有幫助，並能在政策管理變得繁瑣時提供協助。

如需根據標籤控制存取，請使用 aws:ResourceTag/*key-name*、aws:RequestTag/*key-name* 或 aws:TagKeys 條件索引鍵，在政策的條件元素中，提供標籤資訊。

如果服務支援每個資源類型的全部三個條件金鑰，則對該服務而言，值為 Yes。如果服務僅支援某些資源類型的全部三個條件金鑰，則值為 Partial。

如需 ABAC 的詳細資訊，請參閱 IAM 使用者指南中的什麼是 ABAC?。如要查看含有設定 ABAC 步驟的教學課程，請參閱 IAM 使用者指南中的使用屬性型存取控制 (ABAC)。

使用臨時登入資料搭配 Amazon Lex

支援臨時憑證	否

當您使用臨時憑據登錄時，某些 AWS 服務 不起作用。如需其他資訊，包括哪些 AWS 服務 與臨時登入資料搭配AWS 服務 使用，請參閱 IAM 使用者指南中的 IAM。

如果您使用除了使用者名稱和密碼以外的任何方法登入，則您正在 AWS Management Console 使用臨時認證。例如，當您 AWS 使用公司的單一登入 (SSO) 連結存取時，該程序會自動建立暫時認證。當您以使用者身分登入主控台，然後切換角色時，也會自動建立臨時憑證。如需切換角色的詳細資訊，請參閱 IAM 使用者指南中的切換至角色 (主控台)。

您可以使用 AWS CLI 或 AWS API 手動建立臨時登入資料。然後，您可以使用這些臨時登入資料來存取 AWS。 AWS 建議您動態產生臨時登入資料，而不是使用長期存取金鑰。如需詳細資訊，請參閱 IAM 中的暫時性安全憑證。

Amazon Lex V2 的跨服務主體許可

支援轉寄存取工作階段 (FAS)	是

當您使用 IAM 使用者或角色在中執行動作時 AWS，您會被視為主體。使用某些服務時，您可能會執行某個動作，進而在不同服務中啟動另一個動作。FAS 會使用主體呼叫的權限 AWS 服務，並結合要求 AWS 服務 向下游服務發出要求。只有當服務收到需要與其 AWS 服務 他資源互動才能完成的請求時，才會發出 FAS 請求。在此情況下，您必須具有執行這兩個動作的許可。如需提出 FAS 請求時的政策詳細資訊，請參閱《轉發存取工作階段》。

Amazon Lex V2 的服務角色

支援服務角色	是

服務角色是服務擔任的 IAM 角色，可代您執行動作。IAM 管理員可以從 IAM 內建立、修改和刪除服務角色。如需詳細資訊，請參閱 IAM 使用者指南中的建立角色以委派許可給 AWS 服務服務。

> **Warning**
>
> 變更服務角色的許可可可可可能會中斷 Amazon Lex V2 的功能。只有在 Amazon Lex V2 提供指引時才編輯服務角色。

Amazon Lex V2 的服務連結角色

支援服務連結角色	部分

服務連結角色是一種連結至. AWS 服務服務可以擔任代表您執行動作的角色。服務連結角色會顯示在您的中， AWS 帳戶 且屬於服務所有。IAM 管理員可以檢視，但不能編輯服務連結角色的許可。

如需建立或管理服務連結角色的詳細資訊，請參閱可搭配 IAM 運作的AWS 服務。在表格中尋找服務，其中包含服務連結角色欄中的 Yes。選擇是連結，以檢視該服務的服務連結角色文件。

Amazon Lex V2 的基於身份的政策示例

依預設，使用者和角色沒有建立或修改 Amazon Lex V2 資源的權限。他們也無法使用 AWS Management Console、 AWS Command Line Interface (AWS CLI) 或 AWS API 來執行工作。若要授予使用者對其所需資源執行動作的許可，IAM 管理員可以建立 IAM 政策。然後，管理員可以將 IAM 政策新增至角色，使用者便能擔任這些角色。

若要了解如何使用這些範例 JSON 政策文件建立 IAM 身分型政策，請參閱《IAM 使用者指南》中的建立 IAM 政策。

如需 Amazon Lex V2 定義的動作和資源類型的詳細資訊，包括每個資源類型的 ARN 格式，請參閱服務授權參考中的 Amazon Lex V2 的動作、資源和條件金鑰。

主題

- 政策最佳實務
- 使用 Amazon Lex V2 控制台
- 允許使用者將函數新增至機器人

- 允許使用者將頻道新增至機器人
- 允許使用者建立和更新機器人
- 允許使用者使用自動 Chatbot 設計工具
- 允許使用者使用 AWS KMS 金鑰來加密和解密檔案
- 允許使用者刪除機器人
- 允許使用者與機器人進行對話
- 允許特定使用者管理以資源為基礎的政策
- 允許使用者匯出機器人和機器人語言環境
- 允許使用者匯出自訂字彙
- 允許使用者匯入機器人和機器人語言環境
- 允許使用者匯入自訂字彙
- 允許使用者將機器人從 Amazon Lex 遷移到 Amazon Lex V2
- 允許使用者檢視他們自己的許可
- 允許使用者在 Amazon Lex V2 中使用視覺交談建立器繪製對話流程
- 允許使用者建立和檢視機器人複本，但不能刪除它們

政策最佳實務

以身分識別為基礎的政策可決定某人是否可以在您的帳戶中建立、存取或刪除 Amazon Lex V2 資源。這些動作可能會讓您的 AWS 帳戶產生費用。當您建立或編輯身分型政策時，請遵循下列準則及建議事項：

- 開始使用 AWS 受管原則並邁向最低權限權限 — 若要開始授與使用者和工作負載的權限，請使用可授與許多常見使用案例權限的AWS 受管理原則。它們在您的 AWS 帳戶. 建議您透過定義特定於您使用案例的 AWS 客戶管理政策，進一步降低使用權限。如需更多資訊，請參閱 IAM 使用者指南中的 AWS 受管政策或任務職能的AWS 受管政策。
- 套用最低權限許可 – 設定 IAM 政策的許可時，請僅授予執行任務所需的許可。為實現此目的，您可以定義在特定條件下可以對特定資源採取的動作，這也稱為最低權限許可。如需使用 IAM 套用許可的更多相關資訊，請參閱 IAM 使用者指南中的 IAM 中的政策和許可。
- 使用 IAM 政策中的條件進一步限制存取權 – 您可以將條件新增至政策，以限制動作和資源的存取。例如，您可以撰寫政策條件，指定必須使用 SSL 傳送所有請求。您也可以使用條件來授與對服務動作的存取權 (如透過特定) 使用這些動作 AWS 服務，例如 AWS CloudFormation。如需詳細資訊，請參閱 IAM 使用者指南中的 IAM JSON 政策元素：條件。

- 使用 IAM Access Analyzer 驗證 IAM 政策，確保許可安全且可正常運作 – IAM Access Analyzer 驗證新政策和現有政策，確保這些政策遵從 IAM 政策語言 (JSON) 和 IAM 最佳實務。IAM Access Analyzer 提供 100 多項政策檢查及切實可行的建議，可協助您編寫安全且實用的政策。如需更多資訊，請參閱 IAM 使用者指南中的 IAM Access Analyzer 政策驗證。
- 需要多因素身份驗證 (MFA) — 如果您的案例需要 IAM 使用者或根使用者 AWS 帳戶，請開啟 MFA 以獲得額外的安全性。如需在呼叫 API 操作時請求 MFA，請將 MFA 條件新增至您的政策。如需更多資訊，請參閱 IAM 使用者指南中的設定 MFA 保護的 API 存取。

如需 IAM 中最佳實務的相關資訊，請參閱 IAM 使用者指南中的 IAM 安全最佳實務。

使用 Amazon Lex V2 控制台

若要存取 Amazon Lex V2 主控台，您必須擁有一組最低限度的許可。這些許可必須允許您列出和檢視有關 AWS 帳戶. 如果您建立比最基本必要許可更嚴格的身分型政策，則對於具有該政策的實體 (使用者或角色) 而言，主控台就無法如預期運作。

您不需要為僅對 AWS CLI 或 AWS API 進行呼叫的使用者允許最低主控台權限。反之，只需允許存取符合他們嘗試執行之 API 操作的動作就可以了。

為確保使用者和角色仍然可以使用 Amazon Lex V2 主控台，使用者必須擁有主控台存取權。如需建立具有主控台存取權限的使用者的詳細資訊，請參閱 IAM 使用者指南中的在您的 AWS 帳戶中建立 IAM 使用者。

允許使用者將函數新增至機器人

此範例顯示允許 IAM 使用者將 Amazon Comprehend、情緒分析和 Amazon Kendra 查詢許可新增至 Amazon Lex V2 機器人的政策。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Id1",
            "Effect": "Allow",
            "Action": "iam:PutRolePolicy",
            "Resource": "arn:aws:iam::*:role/aws-service-role/lexv2.amazonaws.com/
AWSServiceRoleForLexV2Bots*"
        },
        {
```

```
            "Sid": "Id2",
            "Effect": "Allow",
            "Action": "iam:GetRolePolicy",
            "Resource": "arn:aws:iam::*:role/aws-service-role/lexv2.amazonaws.com/
AWSServiceRoleForLexV2Bots*"
        }
    ]
}
```

允許使用者將頻道新增至機器人

此範例是允許 IAM 使用者將訊息通道新增至機器人的政策。使用者必須先制定此原則，才能在訊息平台上部署 Bot。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Id1",
            "Effect": "Allow",
            "Action": "iam:PutRolePolicy",
            "Resource": "arn:aws:iam::*:role/aws-service-role/
channels.lexv2.amazonaws.com/AWSServiceRoleForLexV2Channels*"
        },
        {
            "Sid": "Id2",
            "Effect": "Allow",
            "Action": "iam:GetRolePolicy",
            "Resource": "arn:aws:iam::*:role/aws-service-role/
channels.lexv2.amazonaws.com/AWSServiceRoleForLexV2Channels*"
        }
    ]
}
```

允許使用者建立和更新機器人

此範例顯示允許 IAM 使用者建立和更新任何機器人的範例政策。此原則包含在主控台或使用 AWS CLI 或 AWS API 完成此動作的權限。

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

```
    {
      "Action": [
        "lex:CreateBot",
        "lex:UpdateBot".
        "iam:PassRole"
      ],
      "Effect": "Allow",
      "Resource": ["arn:aws:lex:Region:123412341234:bot/*]
    }
  ]
}
```

允許使用者使用自動 Chatbot 設計工具

此範例顯示允許 IAM 使用者執行自動 Chatbot 設計工具的範例政策。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::<customer-bucket>/<bucketName>",
                # Resource should point to the bucket or an explicit folder.
                # Provide this to read the entire bucket
                "arn:aws:s3:::<customer-bucket>/<bucketName>/*",
                # Provide this to read a specifc folder
                "arn:aws:s3:::<customer-bucket>/<bucketName>/<pathFormat>/*"
            ]
        },
        {
            # Use this if your S3 bucket is encrypted with a KMS key.
            "Effect": "Allow",
            "Action": [
                "kms:Decrypt"
            ],
            "Resource": [
                "arn:aws:kms:<Region>:<customerAccountId>:key/<kmsKeyId>"
            ]
```

```
    ]
}
```

允許使用者使用 AWS KMS 金鑰來加密和解密檔案

此範例顯示範例政策，該政策允許 IAM 使用者使用 AWS KMS 客戶受管金鑰來加密和解密資料。

```
{
    "Version": "2012-10-17",
    "Id": "sample-policy",
    "Statement": [
        {
            "Sid": "Allow Lex access",
            "Effect": "Allow",
            "Principal": {
                "Service": "lexv2.amazonaws.com"
            },
            "Action": [
                # If the key is for encryption
                "kms:Encrypt",
                "kms:GenerateDataKey"
                # If the key is for decryption
                "kms:Decrypt"
            ],
            "Resource": "*"
        }
   ]
}
```

允許使用者刪除機器人

此範例顯示允許 IAM 使用者刪除任何機器人的範例政策。此原則包含在主控台或使用 AWS CLI 或 AWS API 完成此動作的權限。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
          "lex:DeleteBot",
          "lex:DeleteBotLocale",
          "lex:DeleteBotAlias",
```

```
            "lex:DeleteIntent",
            "lex:DeleteSlot",
            "lex:DeleteSlottype"
        ],
        "Effect": "Allow",
        "Resource": ["arn:aws:lex:Region:123412341234:bot/*",
                     "arn:aws:lex:Region:123412341234:bot-alias/*"]
    }
  ]
}
```

允許使用者與機器人進行對話

此範例顯示範例政策，允許 IAM 使用者與任何機器人進行對話。此原則包含在主控台或使用 AWS CLI 或 AWS API 完成此動作的權限。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
          "lex:StartConversation",
          "lex:RecognizeText",
          "lex:RecognizeUtterance",
          "lex:GetSession",
          "lex:PutSession",
          "lex:DeleteSession"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:lex:Region:123412341234:bot-alias/*"
    }
  ]
}
```

允許特定使用者管理以資源為基礎的政策

下列範例授與特定使用者管理以資源為基礎之策略的權限。它允許控制台和 API 訪問與機器人和機器人別名相關聯的策略。

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

```
        {
            "Sid": "ResourcePolicyEditor",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:role/ResourcePolicyEditor"
            },
            "Action": [
                "lex:CreateResourcePolicy",
                "lex:UpdateResourcePolicy",
                "lex:DeleteResourcePolicy",
                "lex:DescribeResourcePolicy"
            ]
        }
    ]
}
```

允許使用者匯出機器人和機器人語言環境

下列 IAM 權限政策可讓使用者建立、更新和取得機器人或機器人地區設定的匯出。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "lex:CreateExport",
                "lex:UpdateExport",
                "lex:DescribeExport",
                "lex:DescribeBot",
                "lex:DescribeBotLocale",
                "lex:ListBotLocales",
                "lex:DescribeIntent",
                "lex:ListIntents",
                "lex:DescribeSlotType",
                "lex:ListSlotTypes",
                "lex:DescribeSlot",
                "lex:ListSlots",
                "lex:DescribeCustomVocabulary"
            ],
            "Effect": "Allow",
            "Resource": ["arn:aws:lex:Region:123456789012:bot/*"]
        }
    ]
```

```
}
```

允許使用者匯出自訂字彙

下列 IAM 權限政策允許使用者從機器人地區設定匯出自訂詞彙。

```
{"Version": "2012-10-17",
    "Statement": [
        {"Action": [
                "lex:CreateExport",
                "lex:UpdateExport",
                "lex:DescribeExport",
                "lex:DescribeCustomVocabulary"
            ],
            "Effect": "Allow",
            "Resource": ["arn:aws:lex:Region:123456789012:bot/*"]
        }
    ]
}
```

允許使用者匯入機器人和機器人語言環境

下列 IAM 權限政策可讓使用者匯入機器人或機器人地區設定，並檢查匯入狀態。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "lex:CreateUploadUrl",
                "lex:StartImport",
                "lex:DescribeImport",
                "lex:CreateBot",
                "lex:UpdateBot",
                "lex:DeleteBot",
                "lex:CreateBotLocale",
                "lex:UpdateBotLocale",
                "lex:DeleteBotLocale",
                "lex:CreateIntent",
                "lex:UpdateIntent",
                "lex:DeleteIntent",
                "lex:CreateSlotType",
```

```
                "lex:UpdateSlotType",
                "lex:DeleteSlotType",
                "lex:CreateSlot",
                "lex:UpdateSlot",
                "lex:DeleteSlot",
                "lex:CreateCustomVocabulary",
                "lex:UpdateCustomVocabulary",
                "lex:DeleteCustomVocabulary",
                "iam:PassRole",
            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:lex:Region:123456789012:bot/*",
                "arn:aws:lex:Region:123456789012:bot-alias/*"
            ]
        }
    ]
}
```

允許使用者匯入自訂字彙

下列 IAM 權限政策允許使用者將自訂詞彙匯入機器人地區設定。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "lex:CreateUploadUrl",
                "lex:StartImport",
                "lex:DescribeImport",
                "lex:CreateCustomVocabulary",
                "lex:UpdateCustomVocabulary",
                "lex:DeleteCustomVocabulary"

            ],
            "Effect": "Allow",
            "Resource": [
                "arn:aws:lex:Region:123456789012:bot/*"
            ]
        }
    ]
}
```

允許使用者將機器人從 Amazon Lex 遷移到 Amazon Lex V2

下列 IAM 權限政策可讓使用者開始將機器人從 Amazon Lex 遷移到 Amazon Lex V2。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "startMigration",
            "Effect": "Allow",
            "Action": "lex:StartMigration",
            "Resource": "arn:aws:lex:>Region<:>123456789012<:bot:*"
        },
        {
            "Sid": "passRole",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::>123456789012<:role/>v2 bot role<"
        },
        {
            "Sid": "allowOperations",
            "Effect": "Allow",
            "Action": [
                "lex:CreateBot",
                "lex:CreateIntent",
                "lex:UpdateSlot",
                "lex:DescribeBotLocale",
                "lex:UpdateBotAlias",
                "lex:CreateSlotType",
                "lex:DeleteBotLocale",
                "lex:DescribeBot",
                "lex:UpdateBotLocale",
                "lex:CreateSlot",
                "lex:DeleteSlot",
                "lex:UpdateBot",
                "lex:DeleteSlotType",
                "lex:DescribeBotAlias",
                "lex:CreateBotLocale",
                "lex:DeleteIntent",
                "lex:StartImport",
                "lex:UpdateSlotType",
                "lex:UpdateIntent",
                "lex:DescribeImport",
                "lex:CreateCustomVocabulary",
```

```
                "lex:UpdateCustomVocabulary",
                "lex:DeleteCustomvocabulary",
                "lex:DescribeCustomVocabulary",
                "lex:DescribeCustomVocabularyMetadata"
            ],
            "Resource": [
                "arn:aws:lex:>Region<:>123456789012<:bot/*",
                "arn:aws:lex:>Region<:>123456789012<:bot-alias/*/*"
            ]
        },
        {
            "Sid": "showBots",
            "Effect": "Allow",
            "Action": [
                "lex:CreateUploadUrl",
                "lex:ListBots"
            ],
            "Resource": "*"
        }
    ]
}
```

允許使用者檢視他們自己的許可

此範例會示範如何建立政策，允許 IAM 使用者檢視附加到他們使用者身分的內嵌及受管政策。此原則包含在主控台上或以程式設計方式使用 AWS CLI 或 AWS API 完成此動作的權限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
```

```
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

允許使用者在 Amazon Lex V2 中使用視覺交談建立器繪製對話流程

下列 IAM 權限政策允許使用者在 Amazon Lex V2 中使用視覺化交談產生器繪製交談流程。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {"Action": [
                "lex:UpdateIntent ",
                "lex:DescribeIntent "
            ],
            "Effect": "Allow",
            "Resource": ["arn:aws:lex:Region:123456789012:bot/*"]
        }
    ]
}
```

允許使用者建立和檢視機器人複本，但不能刪除它們

您可以將以下許可附加到 IAM 角色，以僅允許其建立和檢視機器人複本。藉由省略`lex:DeleteBotReplica`，您可以防止角色刪除機器人複本。如需詳細資訊，請參閱 [複製機器人和管理機器人複本的權限]。

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

```
    "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "lex:CreateBotReplica",
            "lex:DescribeBotReplica",
            "lex:ListBotReplica",
            "lex:ListBotVersionReplicas",
            "lex:ListBotAliasReplicas",
         ],
         "Resource": [
             "arn:aws:lex:*:*:bot/*",
             "arn:aws:lex:*:*:bot-alias/*"
         ]
     },
      {
         "Effect": "Allow",
         "Action": [
             "iam:GetRole"
         ],
         "Resource": [
             "arn:aws:iam::*:role/aws-service-role/replication.lexv2.amazonaws.com/
AWSServiceRoleForLexV2Replication*"
         ]
      },
      {
         "Effect": "Allow",
         "Action": [
             "iam:CreateServiceLinkedRole",
         ],
         "Resource": [
             "arn:aws:iam::*:role/aws-service-role/replication.lexv2.amazonaws.com/
AWSServiceRoleForLexV2Replication*"
         ],
         "Condition": {
             "StringEquals": {
                 "iam:AWSServiceName": "lexv2.amazonaws.com"
             }
         }
      }
    ]
}
```

Amazon Lex V2 的資源型政策範例

以資源為基礎的政策會附加至資源，例如機器人或機器人別名。使用以資源為基礎的策略，您可以指定誰可以存取資源，以及他們可以對其執行的動作。例如，您可以新增以資源為基礎的政策，讓使用者能夠修改特定機器人，或允許使用者對特定機器人別名使用執行階段作業。

當您使用以資源為基礎的策略時，您可以允許其他 AWS 服務存取您帳戶中的資源。例如，您可以允許 Amazon Connect 訪問 Amazon Lex 機器人。

若要瞭解如何建立機器人或機器人別名，請參閱建築機器人。

主題

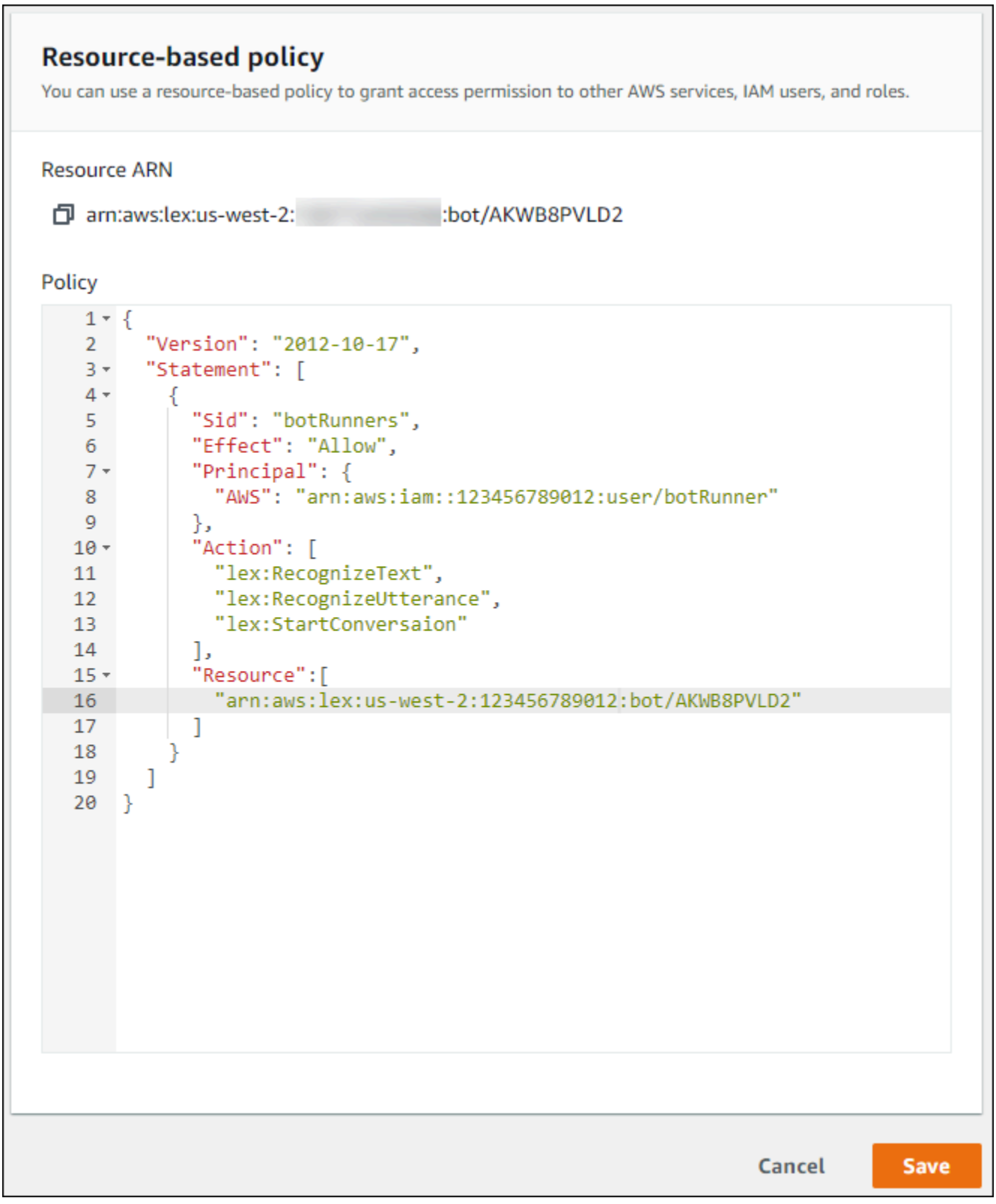
- 使用主控台指定以資源為基礎的策略
- 使用 API 指定以資源為基礎的政策
- 允許 IAM 角色更新機器人並列出機器人別名
- 允許使用者與機器人進行對話
- 允許 AWS 服務使用特定的 Amazon Lex V2 機器人

使用主控台指定以資源為基礎的策略

您可以使用 Amazon Lex 主控台管理機器人和機器人別名的資源型政策。您可以輸入策略的 JSON 結構，控制台將其與資源相關聯。如果已經存在與資源相關聯的策略，您可以使用主控台來檢視和修改策略。

當您使用原則編輯器儲存原則時，主控台會檢查原則的語法。如果原則包含錯誤 (例如不存在的使用者或資源不支援的動作)，則會傳回錯誤且不儲存原則。

以下顯示主控台中機器人的資源型政策編輯器。機器人別名的原則編輯器類似。


Resource-based policy
You can use a resource-based policy to grant access permission to other AWS services, IAM users, and roles.
Resource ARN
arn:aws:lex:us-west-2: :bot/AKWB8PVLD2
Policy
{
"Version": "2012-10-17",
"Statement": [
{
"Sid": "botRunners",
"Effect": "Allow",
"Principal": {
"AWS": "arn:aws:iam::123456789012:user/botRunner"
},
"Action": [
"lex:RecognizeText",
"lex:RecognizeUtterance",
"lex:StartConversaion"
],
"Resource":[
"arn:aws:lex:us-west-2:123456789012:bot/AKWB8PVLD2"
]
}
]
}
Cancel
Save

若要開啟機器人的政策編輯器

1. 登錄到 AWS Management Console 並打開 Amazon Lex 控制台 https://console.aws.amazon.com/lex/.
2. 從「機器人」清單中，選擇您要編輯其政策的機器人。
3. 在以資源為基礎的政策區段中，選擇編輯。

若要開啟機器人別名的原則編輯器

1. 登錄到 AWS Management Console 並打開 Amazon Lex 控制台 https://console.aws.amazon.com/lex/.
2. 從「機器人」清單中，選擇包含您要編輯之別名的機器人。
3. 從左側選單中選擇別名，然後選擇要編輯的別名。
4. 在以資源為基礎的政策區段中，選擇編輯。

使用 API 指定以資源為基礎的政策

您可以使用 API 操作來管理機器人和機器人別名的資源型政策。有建立、更新和刪除策略的作業。

CreateResource政策

將具有指定政策陳述式的新資源策略新增至機器人或機器人別名。

CreateResourcePolicyStatement

將新的資源政策陳述式新增至機器人或機器人別名。

DeleteResource政策

從機器人或機器人別名中移除資源策略。

DeleteResourcePolicyStatement

從機器人或機器人別名移除資源政策陳述式。

DescribeResource政策

取得資源策略和策略修訂版本。

UpdateResource政策

將機器人或機器人別名的現有資源策略取代為新的別名。

範例

Java

下列範例顯示如何使用以資源為基礎的原則作業來管理以資源為基礎的原則。

```
        /*
         * Create a new policy for the specified bot alias
         * that allows a role to invoke lex:UpdateBotAlias on it.
         * The created policy will have revision id 1.
         */

        CreateResourcePolicyRequest createPolicyRequest =
                CreateResourcePolicyRequest.builder()
                        .resourceArn("arn:aws:lex:Region:123456789012:bot-
alias/MYBOTALIAS/TSTALIASID")
                        .policy("{\"Version\": \"2012-10-17\",\"Statement
\": [{\"Sid\": \"BotAliasEditor\",\"Effect\": \"Allow\",\"Principal\":
 {\"AWS\": \"arn:aws:iam::123456789012:role/BotAliasEditor\"},\"Action\":
 [\"lex:UpdateBotAlias\"],\"Resource\":[\"arn:aws:lex:Region:123456789012:bot-
alias/MYBOTALIAS/TSTALIASID\"]]}")

        lexmodelsv2Client.createResourcePolicy(createPolicyRequest);

        /*
         * Overwrite the policy for the specified bot alias with a new policy.
         * Since no expectedRevisionId is provided, this request overwrites the
 current revision.
         * After this update, the revision id for the policy is 2.
         */
        UpdateResourcePolicyRequest updatePolicyRequest =
        UpdateResourcePolicyRequest.builder()
                        .resourceArn("arn:aws:lex:Region:123456789012:bot-
alias/MYBOTALIAS/TSTALIASID")
                        .policy("{\"Version\": \"2012-10-17\",\"Statement
\": [{\"Sid\": \"BotAliasEditor\",\"Effect\": \"Deny\",\"Principal\":
 {\"AWS\": \"arn:aws:iam::123456789012:role/BotAliasEditor\"},\"Action\":
 [\"lex:UpdateBotAlias\"],\"Resource\":[\"arn:aws:lex:Region:123456789012:bot-
alias/MYBOTALIAS/TSTALIASID\"]]}")

        lexmodelsv2Client.updateResourcePolicy(updatePolicyRequest);

        /*
         * Creates a statement in an existing policy for the specified bot alias
```

```
         * that allows a role to invoke lex:RecognizeText on it.
         * This request expects to update revision 2 of the policy. The request will
 fail
         * if the current revision of the policy is no longer revision 2.
         * After this request, the revision id for this policy will be 3.
         */

        CreateResourcePolicyStatementRequest createStatementRequest =
                CreateResourcePolicyStatementRequest.builder()
                        .resourceArn("arn:aws:lex:Region:123456789012:bot-
alias/MYBOTALIAS/TSTALIASID")
                        .effect("Allow")

 .principal(Principal.builder().arn("arn:aws:iam::123456789012:role/
BotRunner").build())
                        .action("lex:RecognizeText")
                        .statementId("BotRunnerStatement")
                        .expectedRevisionId(2)
                        .build();

        lexmodelsv2Client.createResourcePolicyStatement(createStatementRequest);

        /*
         * Deletes a statement from an existing policy for the specified bot alias
 by statementId.
         * Since no expectedRevisionId is supplied, the request will remove the
 statement from
         * the current revision of the policy for the bot alias.
         * After this request, the revision id for this policy will be 4.
         */
        DeleteResourcePolicyRequest deleteStatementRequest =
                DeleteResourcePolicyRequest.builder()
                        .resourceArn("arn:aws:lex:Region:123456789012:bot-
alias/MYBOTALIAS/TSTALIASID")
                        .statementId("BotRunnerStatement")
                        .build();

        lexmodelsv2Client.deleteResourcePolicy(deleteStatementRequest);

        /*
         * Describe the current policy for the specified bot alias
         * It always returns the current revision.
         */
        DescribeResourcePolicyRequest describePolicyRequest =
```

```
                DescribeResourcePolicyRequest.builder()
                        .resourceArn("arn:aws:lex:Region:123456789012:bot-
alias/MYBOTALIAS/TSTALIASID")
                        .build();

        lexmodelsv2Client.describeResourcePolicy(describePolicyRequest);


        /*
         * Delete the current policy for the specified bot alias
         * This request expects to delete revision 3 of the policy. Since the
 revision id for
         * this policy is already at 4, this request will fail.
         */
        DeleteResourcePolicyRequest deletePolicyRequest =
                DeleteResourcePolicyRequest.builder()
                        .resourceArn("arn:aws:lex:Region:123456789012:bot-
alias/MYBOTALIAS/TSTALIASID")
                        .expectedRevisionId(3);
                        .build();

        lexmodelsv2Client.deleteResourcePolicy(deletePolicyRequest);
```

允許 IAM 角色更新機器人並列出機器人別名

下列範例授予特定 IAM 角色的許可，以呼叫 Amazon Lex V2 模型建置 API 操作以修改現有機器人。使用者可以列出機器人的別名並更新機器人，但無法刪除機器人或機器人別名。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "botBuilders",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:role/BotBuilder"
            },
            "Action": [
                "lex:ListBotAliases",
                "lex:UpdateBot"
            ],
            "Resource": [
```

```
                "arn:aws:lex:Region:123456789012:bot/MYBOT"
            ]
        }
    ]
}
```

允許使用者與機器人進行對話

下列範例授予特定使用者在機器人的單一別名上呼叫 Amazon Lex V2 執行階段 API 操作的權限。

特別拒絕使用者更新或刪除機器人別名的權限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "botRunners",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:user/botRunner"
            },
            "Action": [
                "lex:RecognizeText",
                "lex:RecognizeUtterance",
                "lex:StartConversation",
                "lex:DeleteSession",
                "lex:GetSession",
                "lex:PutSession"
            ],
            "Resource": [
                "arn:aws:lex:Region:123456789012:bot-alias/MYBOT/MYBOTALIAS"
            ]
        },
        {
            "Sid": "botRunners",
            "Effect": "Deny",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:user/botRunner"
            },
            "Action": [
                "lex:UpdateBotAlias",
                "lex:DeleteBotAlias"
            ],
            "Resource": [
```

```
                "arn:aws:lex:Region:123456789012:bot-alias/MYBOT/MYBOTALIAS"
            ]
        }
    ]
}
```

允許 AWS 服務使用特定的 Amazon Lex V2 機器人

下列範例授予 AWS Lambda 和 Amazon Connect 的權限，以呼叫 Amazon Lex V2 執行階段 API 作業。

服務主體需要條件區塊，且必須使用全域前後關聯索引鍵AWS:SourceAccount和AWS:SourceArn。

這AWS:SourceAccount是呼叫 Amazon Lex V2 機器人的帳戶識別碼。

這AWS:SourceArn是 Amazon Connect 服務執行個體或 Lambda 函數的資源 ARN，呼叫 Amazon Lex V2 機器人別名的來源。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "connect-bot-alias",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "connect.amazonaws.com"
                ]
            },
            "Action": [
                "lex:RecognizeText",
                "lex:StartConversation"
            ],
            "Resource": [
                "arn:aws:lex:Region:123456789012:bot-alias/MYBOT/MYBOTALIAS"
            ],
            "Condition": {
                "StringEquals": {
                    "AWS:SourceAccount": "123456789012"
                },
                "ArnEquals": {
                    "AWS:SourceArn":
 "arn:aws:connect:Region:123456789012:instance/instance-id"
```

```
                }
            }
        },
        {
            "Sid": "lambda-function",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "lambda.amazonaws.com"
                ]
            },
            "Action": [
                "lex:RecognizeText",
                "lex:StartConversation"
            ],
            "Resource": [
                "arn:aws:lex:Region:123456789012:bot-alias/MYBOT/MYBOTALIAS"
            ],
            "Condition": {
                "StringEquals": {
                    "AWS:SourceAccount": "123456789012"
                },
                "ArnEquals": {
                    "AWS:SourceArn":
 "arn:aws:lambda:Region:123456789012:function/function-name"
                }
            }
        }
    ]
}
```

AWS Amazon Lex V2 的受管政策

受 AWS 管理的策略是由建立和管理的獨立策略 AWS。 AWS 受管理的策略旨在為許多常見使用案例提供權限，以便您可以開始將權限指派給使用者、群組和角色。

請記住， AWS 受管理的政策可能不會為您的特定使用案例授與最低權限權限，因為這些權限可供所有 AWS 客戶使用。我們建議您定義使用案例專屬的客戶管理政策，以便進一步減少許可。

您無法變更受 AWS 管理策略中定義的權限。如果 AWS 更新 AWS 受管理原則中定義的權限，則此更新會影響附加原則的所有主體識別 (使用者、群組和角色)。 AWS 當新的啟動或新 AWS 服務 的 API 操作可用於現有服務時，最有可能更新 AWS 受管理策略。

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 V2 和 Amazon Lex 模型建置服務中的所有動作。

許可詳細資訊

此政策包含以下許可：

- lex— 模型建置服務中的 Amazon Lex V2 和 Amazon Lex 資源的唯讀存取權。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AmazonLexReadOnlyStatement1",
            "Effect": "Allow",
            "Action": [
                "lex:GetBot",
                "lex:GetBotAlias",
                "lex:GetBotAliases",
                "lex:GetBots",
                "lex:GetBotChannelAssociation",
                "lex:GetBotChannelAssociations",
                "lex:GetBotVersions",
                "lex:GetBuiltinIntent",
                "lex:GetBuiltinIntents",
                "lex:GetBuiltinSlotTypes",
                "lex:GetIntent",
                "lex:GetIntents",
                "lex:GetIntentVersions",
                "lex:GetSlotType",
                "lex:GetSlotTypes",
```

```
                "lex:GetSlotTypeVersions",
                "lex:GetUtterancesView",
                "lex:DescribeBot",
                "lex:DescribeBotAlias",
                "lex:DescribeBotChannel",
                "lex:DescribeBotLocale",
                "lex:DescribeBotRecommendation",
                "lex:DescribeBotReplica",
                "lex:DescribeBotVersion",
                "lex:DescribeExport",
                "lex:DescribeImport",
                "lex:DescribeIntent",
                "lex:DescribeResourcePolicy",
                "lex:DescribeSlot",
                "lex:DescribeSlotType",
                "lex:ListBots",
                "lex:ListBotLocales",
                "lex:ListBotAliases",
                "lex:ListBotAliasReplicas",
                "lex:ListBotChannels",
                "lex:ListBotRecommendations",
                "lex:ListBotReplicas",
                "lex:ListBotVersions",
                "lex:ListBotVersionReplicas",
                "lex:ListBuiltInIntents",
                "lex:ListBuiltInSlotTypes",
                "lex:ListExports",
                "lex:ListImports",
                "lex:ListIntents",
                "lex:ListRecommendedIntents",
                "lex:ListSlots",
                "lex:ListSlotTypes",
                "lex:ListTagsForResource",
                "lex:SearchAssociatedTranscripts",
                "lex:ListCustomVocabularyItems"
            ],
            "Resource": "*"
        }
    ]
}
```

AWS 受管政策： AmazonLexRunBotsOnly

您可將 AmazonLexRunBotsOnly 政策連接到 IAM 身分。

此政策授予唯讀許可，允許存取執行 Amazon Lex V2 和 Amazon Lex 交談機器人。

許可詳細資訊

此政策包含以下許可：

- lex— 只讀訪問亞馬遜 Lex V2 和亞馬遜 LAmazon Lex 運行時中的所有操作。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "lex:PostContent",
                "lex:PostText",
                "lex:PutSession",
                "lex:GetSession",
                "lex:DeleteSession",
                "lex:RecognizeText",
                "lex:RecognizeUtterance",
                "lex:StartConversation"
            ],
            "Resource": "*"
        }
    ]
}
```

AWS 受管政策： AmazonLexFullAccess

您可將 AmazonLexFullAccess 政策連接到 IAM 身分。

此政策授予管理許可，允許使用者建立、讀取、更新和刪除 Amazon Lex V2 和 Amazon Lex 資源，以及執行 Amazon Lex V2 和 Amazon Lex 交談機器人。

許可詳細資訊

此政策包含以下許可：

- lex— 允許主體對 Amazon Lex V2 和 Amazon Lex 模型建立和執行時期服務中的所有動作進行讀取和寫入存取。
- cloudwatch— 允許校長檢視 Amazon CloudWatch 指標和警示。
- iam— 可讓主參與者建立和刪除服務連結角色、傳遞角色，以及將原則附加至角色和中斷連結。權限被限制為「亞馬遜」亞馬遜萊克斯操作和「Lexv2.Amazonaws.com」Amazon Lex V2 操作.
- kendra— 允許校長列出 Amazon Kendra 索引。
- kms— 允許主參與者描述 AWS KMS 索引鍵和別名。
- lambda— 允許主體列出 AWS Lambda 函數並管理連接到任何 Lambda 函數的權限。
- polly— 允許校長描述 Amazon Polly 聲音並合成語音。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AmazonLexFullAccessStatement1",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:DescribeAlarms",
                "cloudwatch:DescribeAlarmsForMetric",
                "kms:DescribeKey",
                "kms:ListAliases",
                "lambda:GetPolicy",
                "lambda:ListFunctions",
                "lambda:ListAliases",
                "lambda:ListVersionsByFunction"
                "lex:*",
                "polly:DescribeVoices",
                "polly:SynthesizeSpeech",
                "kendra:ListIndices",
                "iam:ListRoles",
                "s3:ListAllMyBuckets",
                "logs:DescribeLogGroups",
                "s3:GetBucketLocation"
            ],
            "Resource": [
                "*"
            ]
        },
```

```
        {
            "Sid": "AmazonLexFullAccessStatement2",
            "Effect": "Allow",
            "Action": [
                "bedrock:ListFoundationModels"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "bedrock:InvokeModel"
            ],
            "Resource": "arn:aws:bedrock:*::foundation-model/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "lambda:AddPermission",
                "lambda:RemovePermission"
            ],
            "Resource": "arn:aws:lambda:*:*:function:AmazonLex*",
            "Condition": {
                "StringEquals": {
                    "lambda:Principal": "lex.amazonaws.com"
                }
            }
        },
        {
            "Sid": "AmazonLexFullAccessStatement3",
            "Effect": "Allow",
            "Action": [
                "iam:GetRole",
                "iam:GetRolePolicy"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/lex.amazonaws.com/
AWSServiceRoleForLexBots",
                "arn:aws:iam::*:role/aws-service-role/channels.lex.amazonaws.com/
AWSServiceRoleForLexChannels",
                "arn:aws:iam::*:role/aws-service-role/lexv2.amazonaws.com/
AWSServiceRoleForLexV2Bots*",
                "arn:aws:iam::*:role/aws-service-role/channels.lexv2.amazonaws.com/
AWSServiceRoleForLexV2Channels*",
```

```
                "arn:aws:iam::*:role/aws-service-role/replication.lexv2.amazonaws.com/
AWSServiceRoleForLexV2Replication*"
            ]
        },
        {
            "Sid": "AmazonLexFullAccessStatement4",
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/lex.amazonaws.com/
AWSServiceRoleForLexBots"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "lex.amazonaws.com"
                }
            }
        },
        {
            "Sid": "AmazonLexFullAccessStatement5",
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/channels.lex.amazonaws.com/
AWSServiceRoleForLexChannels"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "channels.lex.amazonaws.com"
                }
            }
        },
        {
            "Sid": "AmazonLexFullAccessStatement6",
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": [
```

```
                "arn:aws:iam::*:role/aws-service-role/lexv2.amazonaws.com/
AWSServiceRoleForLexV2Bots*"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "lexv2.amazonaws.com"
                }
            }
        },
        {
            "Sid": "AmazonLexFullAccessStatement7",
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/channels.lexv2.amazonaws.com/
AWSServiceRoleForLexV2Channels*"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "channels.lexv2.amazonaws.com"
                }
            }
        },
        {
            "Sid": "AmazonLexFullAccessStatement8",
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/replication.lexv2.amazonaws.com/
AWSServiceRoleForLexV2Replication*"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:AWSServiceName": "replication.lexv2.amazonaws.com"
                }
            }
        },
        {
            "Sid": "AmazonLexFullAccessStatement9",
            "Effect": "Allow",
```

```
            "Action": [
                "iam:DeleteServiceLinkedRole",
                "iam:GetServiceLinkedRoleDeletionStatus"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/lex.amazonaws.com/
AWSServiceRoleForLexBots",
                "arn:aws:iam::*:role/aws-service-role/channels.lex.amazonaws.com/
AWSServiceRoleForLexChannels",
                "arn:aws:iam::*:role/aws-service-role/lexv2.amazonaws.com/
AWSServiceRoleForLexV2Bots*",
                "arn:aws:iam::*:role/aws-service-role/channels.lexv2.amazonaws.com/
AWSServiceRoleForLexV2Channels*",
                "arn:aws:iam::*:role/aws-service-role/replication.lexv2.amazonaws.com/
AWSServiceRoleForLexV2Replication*"
            ]
        },
        {
            "Sid": "AmazonLexFullAccessStatement10",
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/lex.amazonaws.com/
AWSServiceRoleForLexBots"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "lex.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Sid": "AmazonLexFullAccessStatement11",
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/lexv2.amazonaws.com/
AWSServiceRoleForLexV2Bots*"
```

```
            ],
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "lexv2.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Sid": "AmazonLexFullAccessStatement12",
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/channels.lexv2.amazonaws.com/
AWSServiceRoleForLexV2Channels*"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "channels.lexv2.amazonaws.com"
                    ]
                }
            }
        },
        {
            "Sid": "AmazonLexFullAccessStatement13",
            "Effect": "Allow",
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::*:role/aws-service-role/replication.lexv2.amazonaws.com/
AWSServiceRoleForLexV2Replication*"
            ],
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": [
                        "lexv2.amazonaws.com"
                    ]
                }
            }
```

```
        }
    ]
}
```

AWS 受管政策： AmazonLexReplicationPolicy

您不得將 AmazonLexReplicationPolicy 連接到 IAM 實體。此政策附加至服務連結角色，可讓 Amazon Lex V2 代表您執行動作。如需詳細資訊，請參閱 使用 Amazon Lex V2 的服務連結角色。

此政策授予管理許可，讓 Amazon Lex V2 代表您跨區域複寫 AWS 資源。您可以附加此原則，以允許角色輕鬆複寫資源，包括機器人、地區設定、版本、別名、對應方式、位置類型、位置和自訂字彙。

許可詳細資訊

此政策包含以下許可。

- lex— 允許主參與者複製其他區域中的資源。
- iam— 允許主體從 IAM 傳遞角色。這是必要的，因此 Amazon Lex V2 具有在其他區域複寫資源的許可。

```
{
    "Version": "2012-10-17",
    "Statement": [
 {
  "Sid": "ReplicationPolicyStatement1",
  "Effect": "Allow",
  "Action": [
   "lex:BuildBotLocale",
   "lex:ListBotLocales",
   "lex:CreateBotAlias",
   "lex:UpdateBotAlias",
   "lex:DeleteBotAlias",
   "lex:DescribeBotAlias",
   "lex:CreateBotVersion",
   "lex:DeleteBotVersion",
   "lex:DescribeBotVersion",
   "lex:CreateExport",
   "lex:DescribeBot",
```

```
    "lex:UpdateExport",
    "lex:DescribeExport",
    "lex:DescribeBotLocale",
    "lex:DescribeIntent",
    "lex:ListIntents",
    "lex:DescribeSlotType",
    "lex:ListSlotTypes",
    "lex:DescribeSlot",
    "lex:ListSlots",
    "lex:DescribeCustomVocabulary",
    "lex:StartImport",
    "lex:DescribeImport",
    "lex:CreateBot",
    "lex:UpdateBot",
    "lex:DeleteBot",
    "lex:CreateBotLocale",
    "lex:UpdateBotLocale",
    "lex:DeleteBotLocale",
    "lex:CreateIntent",
    "lex:UpdateIntent",
    "lex:DeleteIntent",
    "lex:CreateSlotType",
    "lex:UpdateSlotType",
    "lex:DeleteSlotType",
    "lex:CreateSlot",
    "lex:UpdateSlot",
    "lex:DeleteSlot",
    "lex:CreateCustomVocabulary",
    "lex:UpdateCustomVocabulary",
    "lex:DeleteCustomVocabulary",
    "lex:DeleteBotChannel",
    "lex:DeleteResourcePolicy"
   ],
   "Resource": [
    "arn:aws:lex:*:*:bot/*",
    "arn:aws:lex:*:*:bot-alias/*"
   ]
  },
  {
   "Sid": "ReplicationPolicyStatement2",
   "Effect": "Allow",
   "Action": [
    "lex:CreateUploadUrl",
    "lex:ListBots"
```

```
  ],
  "Resource": "*"
 },
 {
  "Sid": "ReplicationPolicyStatement3",
  "Effect": "Allow",
  "Action": [
   "iam:PassRole"
  ],
  "Resource": "*",
  "Condition": {
   "StringEquals": {
    "iam:PassedToService": "lexv2.amazonaws.com"
   }
  }
 }
    ]
}
```

Amazon Lex V2 更新受 AWS 管政策

檢視 Amazon Lex V2 AWS 受管政策更新的詳細資料，因為此服務開始追蹤這些變更。如需有關此頁面變更的自動警示，請訂閱 Amazon Lex V2 Amazon Lex V2 的文檔歷史 頁面上的 RSS 摘要。

變更	描述	日期
AmazonLexReadOnly – 更新現有政策	Amazon Lex V2 新增了新的許可，以允許機器人資源的唯讀存取複本。	2024年5月10日
AmazonLexFullAccess – 更新現有政策	Amazon Lex V2 新增了新的許可，允許將機器人資源複寫到其他區域。	2024年4月16日
AmazonLexFullAccess – 更新現有政策	Amazon Lex V2 新增了新的許可，允許將機器人資源複寫到其他區域。	2024 年 1 月 31 日

變更	描述	日期
AmazonLexReplicationPolicy – 新政策	Amazon Lex V2 新增了一項新政策，允許將機器人資源複寫到其他區域。	2024 年 1 月 31 日
AmazonLexReadOnly – 更新現有政策	Amazon Lex V2 新增了新的許可，允許列出自訂詞彙項目的唯讀存取權。	2022 年 11 月 29 日
AmazonLexFullAccess – 更新現有政策	Amazon Lex V2 新增了新的許可，允許對 Amazon Lex V2 模型建置服務作業進行唯讀存取。	2021 年 8 月 18 日
AmazonLexReadOnly – 更新現有政策	Amazon Lex V2 新增了新的許可，允許對 Amazon Lex V2 自動 Chatbot 設計器操作進行唯讀存取。	2021 年 12 月 1 日
AmazonLexFullAccess – 更新現有政策	Amazon Lex V2 新增了新的許可，允許對 Amazon Lex V2 模型建置服務作業進行唯讀存取。	2021 年 8 月 18 日
AmazonLexReadOnly – 更新現有政策	Amazon Lex V2 新增了新的許可，允許對 Amazon Lex V2 模型建置服務作業進行唯讀存取。	2021 年 8 月 18 日
AmazonLexRunBots僅限 — 更新至現有原則	Amazon Lex V2 新增了新的許可，允許對 Amazon Lex V2 執行階段服務作業進行唯讀存取。	2021 年 8 月 18 日
Amazon Lex V2 開始跟踪更改	Amazon Lex V2 開始追蹤其 AWS 受管政策的變更。	2021 年 8 月 18 日

使用 Amazon Lex V2 的服務連結角色

Amazon Lex V2 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 V2 的唯一 IAM 角色類型。Amazon Lex V2 預先定義服務連結角色，並包含服務代表您呼叫其他服 AWS 務所需的所有許可。

服務連結角色可讓您輕鬆設定 Amazon Lex V2，因為您不必手動新增必要的許可。Amazon Lex V2 會定義其服務連結角色的許可，除非另有定義，否則只有 Amazon Lex V2 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。

如需支援服務連結角色其他服務的資訊，請參閱可搭配 IAM 運作的 AWS 服務，並尋找 Service-Linked Role (服務連結角色) 欄顯示 Yes (是) 的服務。選擇具有連結的 Yes (是)，以檢視該服務的服務連結角色文件。

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。

您只能在第一次刪除相關資源後刪除服務連結角色。這樣可以保護您的 Amazon Lex V2 資源，因為您無法意外移除存取資源的許可。

主題

- 為 Amazon Lex V2 建立服務連結角色
- 編輯 Amazon Lex V2 的服務連結角色
- 刪除 Amazon Lex V2 的服務連結角色
- Amazon Lex V2 的服務連結角色許可
- 支援 Amazon Lex V2 服務連結角色的區域

為 Amazon Lex V2 建立服務連結角色

您不需要手動建立服務連結角色，因為 Amazon Lex V2 在、或 AWS API 中執行相關動作 (如需詳細資訊，請參閱以取得更多資訊) AWS Management Console， AWS CLI所以 Amazon Lex V2 會Amazon Lex V2 的服務連結角色許可為您建立服務連結角色。

如果您刪除此服務連結角色，然後需要再次建立角色，則可以使用相同的程序在帳戶中建立新角色。

編輯 Amazon Lex V2 的服務連結角色

Amazon Lex V2 不允許您編輯服務連結角色。因為可能有各種實體會參考服務連結角色，所以您無法在建立角色之後變更其名稱。不過，您可以使用 IAM 編輯角色的說明。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 V2 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 V2 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。

若要查看在 Amazon Lex V2 中刪除特定服務連結角色資源的步驟，請參閱中Amazon Lex V2 的服務連結角色許可的角色特定章節。

使用 IAM 手動刪除服務連結角色

刪除與服務連結角色相關的資源後，請使用 IAM 主控台 AWS CLI、或 AWS API 刪除角色。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 V2 的服務連結角色許可

Amazon Lex V2 使用具有下列前置詞的服務連結角色。

主題

- AWSServiceRoleForLexV2Bots_
- AWSServiceRoleForLexV2Channels_
- AWSServiceRoleForLexV2Replication

AWSServiceRoleForLexV2Bots_

AWSServiceRoleForLexV2Bots_ 角色授予將您的機器人連接到其他必要服務的權限。此角色包含信任原則，可讓 lexv2.amazonaws.com 服務擔任該角色，並包含執行下列動作的權限。

- 使用 Amazon Polly 在動作支援的所有 Amazon Lex V2 資源上合成語音。
- 如果機器人設定為使用 Amazon Comprehend 情緒分析，請在該動作支援的所有 Amazon Lex V2 資源上偵測情緒。
- 如果機器人設定為將音訊日誌存放在 S3 儲存貯體中，請將物件放在指定的儲存貯體中。
- 如果機器人設定為儲存音訊和文字記錄，請建立記錄串流並將記錄放入指定的記錄群組中。
- 如果機器人設定為使用 AWS KMS 金鑰來加密資料，請產生特定的資料金鑰。
- 如果機器人設定為使用`KendraSearchIntent`意圖，請查詢指定 Amazon Kendra 索引的存取權。

若要建立角色

每次建立機器人時，Amazon Lex V2 都會在您的帳戶中建立一個新的 AWSServiceRoleForLexV2Bots_ 角色，並在您的帳戶中加上隨機尾碼。當您將其他功能新增至機器人時，Amazon Lex V2 會修改該角色。例如，如果您將 Amazon Comprehend 情緒分析新增至機器人，Amazon Lex V2 會將該`lex:DetectSentiment`動作的許可新增至服務角色。

刪除 角色

1. 登錄到 AWS Management Console 並打開 Amazon Lex 控制台 https://console.aws.amazon.com/lex/.
2. 在左側導覽窗格中，選取 [機器人]，然後選擇您要刪除其服務連結角色的機器人。
3. 選擇機器人的任何版本。
4. IAM 許可執行階段角色位於版本詳細資料中。
5. 返回機器人頁面，然後選擇要刪除的機器人旁邊的選項按鈕。
6. 選取動作，然後選擇刪除。
7. 請按照刪除服務連結角色中的步驟刪除 IAM 角色。

AWSServiceRoleForLexV2Channels_

AWSServiceRoleForLexV2Channels_ 角色授予列出帳戶中的機器人以及為機器人呼叫對話 API 的權限。這個角色包含一個信任原則，以允許通道 .lexv2.amazonaws.com 服務擔任該角色。如果機器人設定為使用通道與簡訊服務通訊， AWSServiceRoleForLexV2Channels_ 角色許可政策允許 Amazon Lex V2 完成下列動作。

- 列出帳戶中所有機器人的權限。
- 識別文本，獲取會話並將會話權限放在指定的機器人別名上。

若要建立角色

當您建立通道整合以在簡訊平台上部署機器人時，Amazon Lex V2 會在您的帳戶中為每個通道建立新的服務連結角色，並加上隨機尾碼。

刪除 角色

1. 登錄到 AWS Management Console 並打開 Amazon Lex 控制台 https://console.aws.amazon.com/lex/.
2. 在左側導覽窗格中，選取「機器人」。
3. 選擇一個機器人。
4. 在左側導覽窗格中，選擇「部署」下的「通道整合」。
5. 選取您要刪除其服務連結角色的頻道。
6. IAM 許可執行時期角色位於「一般」設定中
7. 選擇「刪除」，然後再次選擇「刪除」來刪除色版。
8. 請按照刪除服務連結角色中的步驟刪除 IAM 角色。

AWSServiceRoleForLexV2Replication

該 AWSServiceRoleForLexV2Replication 角色授予在第二個區域複製機器人的權限。此角色包含信任原則，可允許複寫 .lexv2.amazonaws.com 服務擔任該角色，同時也包含AmazonLexReplicationPolicy AWS 受管理原則，允許執行下列動作的權限。

- 將機器人 IAM 角色傳遞給複本機器人，以重新複製複本機器人的適當許可。
- 在其他地區建立和管理機器人和機器人資源 (版本、別名、意圖、位置、自訂詞彙等)。

若要建立角色

當您為機器人啟用全域彈性時，Amazon Lex V2 會在您的帳 AWSServiceRoleForLexV2Replication 戶中建立服務連結角色。確保您擁有正確的許可，可授與 Amazon Lex V2 服務許可以建立服務連結角色。

若要刪除使用的 Amazon Lex V2 資源，以 AWSServiceRoleForLexV2Replication 便您可以刪除角色

1. 登錄到 AWS Management Console 並打開 Amazon Lex 控制台 https://console.aws.amazon.com/lex/.

2. 選擇啟用全域復原能力的機器人。
3. 選取「部署」下的「全域復原」。
4. 選取停用全域復原。
5. 對已啟用全域恢復能力的所有機器人重複此程序。
6. 請按照刪除服務連結角色中的步驟刪除 IAM 角色。

支援 Amazon Lex V2 服務連結角色的區域

Amazon Lex V2 支援在提供服務的所有區域使用服務連結角色。如需詳細資訊，請參閱 AWS 區域與端點。

疑難排解 Amazon Lex V2 身分識別

使用下列資訊可協助您診斷和修正使用 Amazon Lex V2 和 IAM 時可能遇到的常見問題。

主題

- 我沒有授權在 Amazon Lex V2 中執行操作
- 我沒有授權執行 iam：PassRole
- 我是管理員，並希望允許其他人訪問 Amazon Lex V2
- 將程式設計存取權授予使用者
- 我想允許我的 AWS 帳戶以外的人訪問我的 Amazon Lex V2 資源

我沒有授權在 Amazon Lex V2 中執行操作

如果 AWS Management Console 告訴您您沒有執行動作的授權，則您必須聯絡您的管理員以尋求協助。您的管理員是為您提供簽署憑證的人員。

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，Mateo 會請求管理員更新他的政策，允許他使用 *my-example-widget* 動作存取 lex:*GetWidget* 資源。

我沒有授權執行 iam：PassRole

如果您收到未獲授權執行iam:PassRole動作的錯誤訊息，則必須更新您的政策以允許您將角色傳遞給 Amazon Lex V2。

有些 AWS 服務 允許您將現有角色傳遞給該服務，而不是建立新的服務角色或服務連結角色。如需執行此作業，您必須擁有將角色傳遞至該服務的許可。

當名為的 IAM 使用者marymajor嘗試使用主控台在 Amazon Lex V2 中執行動作時，會發生下列範例錯誤。但是，動作請求服務具備服務角色授予的許可。Mary 沒有將角色傳遞至該服務的許可。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

在這種情況下，Mary 的政策必須更新，允許她執行 iam:PassRole 動作。

如果您需要協助，請聯絡您的 AWS 系統管理員。您的管理員提供您的簽署憑證。

我是管理員，並希望允許其他人訪問 Amazon Lex V2

若要允許其他人存取 Amazon Lex V2，您必須為需要存取的人員或應用程式建立 IAM 實體 (使用者或角色)。他們將使用該實體的憑證來存取 AWS。然後，您必須將政策附加到實體，以便在 Amazon Lex V2 中授予他們正確許可。

若要立即開始使用，請參閱《IAM 使用者指南》中的建立您的第一個 IAM 委派使用者及群組。

將程式設計存取權授予使用者

如果使用者想要與 AWS 之外的 AWS Management Console. 授與程式設計存 AWS取權的方式取決於正在存取的使用者類型。

若要授與使用者程式設計存取權，請選擇下列其中一個選項。

哪個使用者需要程式設計存取權？	到	By
人力身分 (IAM Identity Center 中管理的使用者)	使用臨時登入資料來簽署對 AWS CLI、 AWS SDK 或 AWS API 的程式設計要求。	請依照您要使用的介面所提供的指示操作。 • 如需詳細資訊 AWS CLI，請參閱《使 AWS CLI 用AWS Command Line Interface者

哪個使用者需要程式設計存取權？	到	By
		指南》 AWS IAM Identity Center中的〈配置使用〉。 • 如需 AWS SDK、工具和 AWS API，請參閱 AWS SDK 和工具參考指南中的 IAM 身分中心身分驗證。
IAM	使用臨時登入資料來簽署對 AWS CLI、 AWS SDK 或 AWS API 的程式設計要求。	遵循《IAM 使用者指南》中的〈將臨時登入資料搭配 AWS 資源使用〉中的指示
IAM	(不建議使用) 使用長期認證簽署對 AWS CLI、 AWS SDK 或 AWS API 的程式設計要求。	請依照您要使用的介面所提供的指示操作。 • 如需相關資訊 AWS CLI，請參閱使用指南中的使用 IAM 使用者登入資料進行驗證。AWS Command Line Interface • 對於 AWS SDK 和工具，請參閱 AWS SDK 和工具參考指南中的使用長期憑據進行身份驗證。 • 如需 AWS API，請參閱 IAM 使用者指南中的管理 IAM 使用者的存取金鑰。

我想允許我的 AWS 帳戶以外的人訪問我的 Amazon Lex V2 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 V2 是否支援這些功能，請參閱亞馬遜萊克斯 V2 如何使用 IAM。
- 若要了解如何提供對您所擁有資源 AWS 帳戶 的存取權，請參閱《IAM 使用者指南》中您擁有的另一 AWS 帳戶 個 IAM 使用者提供存取權限。
- 若要了解如何將資源存取權提供給第三方 AWS 帳戶，請參閱 IAM 使用者指南中的提供第三方 AWS 帳戶 擁有的存取權。
- 如需了解如何透過聯合身分提供存取權，請參閱 IAM 使用者指南中的將存取權提供給在外部進行身分驗證的使用者 (聯合身分)。
- 若要了解跨帳戶存取使用角色和以資源為基礎的政策之間的差異，請參閱 IAM 使用者指南中的 IAM 中的跨帳戶資源存取。

在 Amazon Lex V2 中記錄和監控

監控是維護 Amazon Lex V2 和其他 AWS 解決方案的可靠性、可用性和效能的重要組成部分。AWS 提供下列監控工具來觀看 Amazon Lex V2、在發生錯誤時報告，並在適當時採取自動動作：

- Amazon 會即時 CloudWatch監控您的 AWS 資源和執行 AWS 的應用程式。您可以收集和追蹤指標、建立自訂儀板表，以及設定警示，在特定指標達到您指定的閾值時通知您或採取動作。例如，您可以 CloudWatch 追蹤 Amazon EC2 執行個體的 CPU 使用率或其他指標，並在需要時自動啟動新執行個體。如需詳細資訊，請參閱 Amazon CloudWatch 使用者指南。
- AWS CloudTrail擷取您帳戶或代表您 AWS 帳戶發出的 API 呼叫和相關事件，並將日誌檔傳送到您指定的 Amazon S3 儲存貯體。您可以識別呼叫的使用者和帳戶 AWS、進行呼叫的來源 IP 位址，以及呼叫發生的時間。如需詳細資訊，請參閱 AWS CloudTrail 使用者指南。

Amazon Lex V2 的合規驗證

第三方稽核員會評估 Amazon Lex V2 的安全性和合規性，做為多個 AWS 合規計劃的一部分。Amazon Lex V2 是符合 HIPAA 資格的服務。它符合 PCI、SOC 和 ISO 標準。

若要瞭解 AWS 服務 是否屬於特定規範遵循方案的範圍內，請參閱AWS 服務 遵循規範計劃方案中的，並選擇您感興趣的合規方案。如需一般資訊，請參閱AWS 規範計劃AWS。

您可以使用下載第三方稽核報告 AWS Artifact。如需詳細資訊，請參閱下載中的報告中的 AWS Artifact。

您在使用時的合規責任取決 AWS 服務 於資料的敏感性、公司的合規目標以及適用的法律和法規。AWS 提供下列資源以協助遵循法規：

- 安全性與合規性快速入門指南 — 這些部署指南討論架構考量，並提供部署以安全性和合規性 AWS 為重點的基準環境的步驟。
- 在 Amazon Web Services 上架構 HIPAA 安全性與合規性 — 本白皮書說明公司如何使用建立符合 HIPAA 資格的應 AWS 用程式。

> **Note**
>
> 並非所有人 AWS 服務 都符合 HIPAA 資格。如需詳細資訊，請參閱 HIPAA 資格服務參照。

- AWS 合規資源AWS — 此工作簿和指南集合可能適用於您的產業和所在地。
- AWS 客戶合規指南 — 透過合規的角度瞭解共同的責任模式。這份指南總結了在多個架構 (包括美國國家標準技術研究所 (NIST)、支付卡產業安全標準委員會 (PCI) 和國際標準化組織 (ISO)) 中，保 AWS 服務 護指引並對應至安全控制的最佳實務。
- 使用AWS Config 開發人員指南中的規則評估資源 — 此 AWS Config 服務會評估您的資源組態符合內部實務、產業準則和法規的程度。
- AWS Security Hub— 這 AWS 服務 提供了內部安全狀態的全面視圖 AWS。Security Hub 使用安全控制，可評估您的 AWS 資源並檢查您的法規遵循是否符合安全業界標準和最佳實務。如需支援的服務和控制清單，請參閱 Security Hub controls reference。
- Amazon GuardDuty — 透過監控環境中的 AWS 帳戶可疑和惡意活動，藉此 AWS 服務 偵測您的工作負載、容器和資料的潛在威脅。 GuardDuty 可協助您滿足特定合規性架構所要求的入侵偵測需求，如 PCI DSS 等各種合規性需求。
- AWS Audit Manager— 這 AWS 服務 有助於您持續稽核您的 AWS 使用情況，以簡化您管理風險的方式，以及遵守法規和業界標準的方式。

Amazon Lex V2 的彈性

AWS 全球基礎架構是圍繞區 AWS 域和可用區域建立的。 AWS 區域提供多個實體分離和隔離的可用區域，這些區域透過低延遲、高輸送量和高度備援的網路連線。透過可用區域，您可以設計與操作的應用程式和資料庫，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、容錯能力和擴展能力，均較單一或多個資料中心的傳統基礎設施還高。

如需區域和可用區域的相關 AWS 資訊，請參閱AWS 全域基礎結構。

除了 AWS 全球基礎設施之外，Amazon Lex V2 還提供多種功能來協助支援您的資料彈性和備份需求。

> **Note**
>
> 如需 Amazon Lex V2 中的全域復原能力的詳細資訊，可讓您在預先確定的配對中在第二個區域中建立複寫的機器人，請參閱全域彈性。

Amazon Lex V2 的基礎設施

作為受管服務，Amazon Lex V2 受到 Amazon Web Services：安 AWS 全程序概觀白皮書中所述的全球網路安全程序的保護。

您可以使用 AWS 已發佈的 API 呼叫，透過網路存取 Amazon Lex V2。用戶端必須支援 Transport Layer Security (TLS) 1.0 或更新版本。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。

Amazon Lex V2 和接口虛擬私人雲端端點 ()AWS PrivateLink

您可以透過建立介面虛擬私人雲端端點，在虛擬私人雲端和 Amazon Lex VPC 2 之間建立私有連線。界面端點採用這種技術 AWS PrivateLink，可讓您在沒有網際網路閘道、NAT 裝置、VPN 連線或 AWS 直 Connect 連線的情況下私有存取 Amazon Lex V2 API。VPC 中的執行個體不需要公有 IP 地址即可與 Amazon Lex V2 API 進行通訊。您的 VPC 和 Amazon Lex V2 之間的流量不會離開 Amazon 網路。

每個介面端點都是由您子網路中的一或多個彈性網路介面表示。

如需詳細資訊，請參閱 Amazon VPC 使用者指南中的介面虛擬私人雲端端點 (AWS PrivateLink)。

Amazon Lex V2 VPC 端點的注意事項

在為 Amazon Lex V2 設定介面虛擬私人雲端端點之前，請務必先查看 Amazon VPC 使用者指南中的界面端點屬性和限制。

Amazon Lex V2 支援從您的 VPC 呼叫其所有 API 動作。

為 Amazon Lex V2 創建一個接口 VPC 端點

您可以使用 Amazon VPC 主控台或 AWS Command Line Interface (AWS CLI) 為 Amazon Lex V2 服務建立 VPC 端點。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的建立介面端點。

使用下列服務名稱為 Amazon Lex V2 建立虛擬私人雲端端點：

- com.amazonaws.*region*.models-v2-lex
- com.amazonaws.*region*.runtime-v2-lex

如果您為端點啟用私有 DNS，則可以使用該區域的預設 DNS 名稱向 Amazon Lex V2 發出 API 請求，例如`runtime-v2-lex.us-east-1.amazonaws.com`。

如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的透過介面端點存取服務。

為 Amazon Lex V2 建立 VPC 端點政策

您可以將端點政策附加到虛擬私人雲端端點，以控制對 Amazon Lex V2 的存取。此政策會指定下列資訊：

- 可執行動作的主體。
- 可執行的動作。
- 可供執行動作的資源。

如需詳細資訊，請參閱 Amazon VPC 使用者指南中的使用 VPC 端點控制對服務的存取。

範例：適用於 Amazon Lex V2 動作的 VPC 端點政策

以下是適用於 Amazon Lex V2 的端點政策範例。連接到端點時，此政策會授予對所有資源上所有主體列出之 Amazon Lex V2 動作的存取權。

```
{
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
            "lex:RecognizeText",
            "lex:RecognizeUtterance",
```

```
            "lex:StartConversation",
            "lex:DeleteSession",
            "lex:GetSession",
            "lex:DeleteSession"
         ],
         "Resource":"*"
      }
   ]
}
```

準則和最佳實務

請參閱下列準則和最佳做法，以最佳化機器人的行為以及與客戶的互動。

簽署請求

API 參考中的所有 Amazon Lex V2 模型建置和執行階段請求都使用簽章 V4 來驗證請求。如需有關驗證要求的詳細資訊，請參閱. AWS 一般參考

保護機密資訊

運行時 API 操作，RecognizeText並RecognizeUtterance採取會話 ID 作為必要參數。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傳送任何機密資訊，例如使用者登入資訊、電子郵件或社會安全號碼。此 ID 主要用於唯一識別與機器人的對話。

從用戶話語中捕獲插槽值

Amazon Lex V2 會使用您在插槽類型定義中提供的列舉值來訓練其機器學習模型。假設您定義了使用下列範例語調`GetPredictionIntent`用的意圖：

```
"Tell me the prediction for {sign}"
```

其中 {sign} 是具有 12 個枚舉值的自定義類型`ZodiacSign`的插槽：`Aries`通過`Pisces`。現在假設用戶說「告訴我對地球的預測」：

- 如果您執行下列其中一個動作，Amazon Lex V2 會推斷出「地球」是一個ZodiacSign值：
  - 將`valueSelectionStrategy`欄位設定為`ORIGINAL_VALUE`使用CreateSlotType操作
  - 在主控台中選取 [擴充值]
- 如果您透過執行下列其中一個動作，將辨識限制為您針對插槽類型定義的值，Amazon Lex V2 無法辨識「地球」值：
  - 將`valueSelectionStrategy`欄位設定為`TOP_RESOLUTION`使用`CreateSlotType`操作
  - 在主控台中選取 [限制為插槽值和同義字]

當您定義槽值的同義字時，它們會被識別為與槽值相同。不過，會傳回槽值，而不是同義字。

由於 Amazon Lex V2 會將此值傳遞給您的用戶端應用程式或 Lambda 函數，因此您應該先檢查插槽值是否為有效值，然後再將其用於履行活動。

當 Amazon Lex V2 呼叫 Lambda 函數或傳回與用戶端之間的語音互動結果時，無法保證插槽值的情況。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。

插槽值中的首字母縮寫

定義包含縮寫的槽值時，請使用下列模式：

- 以句點分隔的大寫字母 (V.D.)
- 以空格分隔的大寫字母 (D V D)

用於日期和時間的內置插槽

亞馬遜。日期和亞馬遜時間內置插槽類型捕獲日期和時間（絕對和相對）。相對日期和時間會在 Amazon Lex V2 接收請求的時間和日期以及處理請求的區域中解決。

對於`AMAZON.Time`內置插槽類型，如果用戶未指定時間在中午之前或之後，則時間不明確。在這種情況下，亞馬遜 Lex V2 將再次提示用戶。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。

避免機器人訓練資料中的模糊

在機器人中提供混淆的訓練資料，可降低 Amazon Lex V2 瞭解使用者輸入的能力。假設您的機器人中有兩個意圖（`OrderPizza`和`OrderDrink`），並且包含「我想訂購」作為示例語句。當您建置機器人時，Amazon Lex V2 無法將此語音對應到特定意圖。因此，當使用者在執行階段輸入此語音時，Amazon Lex V2 無法挑選具有高度信心的意圖。

如果您有兩個意圖具有相同的範例語調，請使用輸入內容協助 Amazon Lex V2 在執行階段區分兩個意圖。如需詳細資訊，請參閱設定意圖前後關聯。

使用標準別名

- 您的機器人的 TSTALIASID 別名指向草稿版本，只能用於手動測試。Amazon Lex 會限制您可以對機器人的 TSTALIASID 別名發出的執行時間請求數量。
- 當您更新機器人的草稿版本時，Amazon Lex 會使用機器人的 TSTALIASID 別名關閉任何用戶端應用程式的任何進行中交談。一般而言，您不應在生產環境中使用機器人的 TSTALIASID 別名，因為草稿版本可以更新。您應該發布一個版本和一個別名，並使用它們來代替。
- 當您更新別名時，Amazon Lex 需要幾分鐘的時間來取得變更。當您修改機器人的草稿版本時，TSTALIASID 別名會立即挑選變更。

配額

服務配額 (也稱為限制) 是您 AWS 帳戶允許的服務資源數目上限。如需詳細資訊，請參閱AWS 一般參考中的 AWS 服務配額。

部分服務配額可以調整或增加。請參閱下表中的「可調整」資料欄，瞭解是否可以調整配額，並參閱「自助服務」資料欄，查看是否可以透過「服務配額」主控台要求配額調整。聯繫 AWS Support 以增加可調節的配額，但不能通過自助服務。增加服務配額可能需要幾天的時間。如果您要在較大專案中增加配額，請務必將這段時間加入您的計劃中。

> **Note**
>
> 字元限制是以 Unicode 程式碼單位的數目來計算。在大多數情況下，一個 Unicode 字符相當於一個 Unicode 代碼單元。某些特殊字元可能大於一個單位，而且不同編碼的計數可能會有所不同。如需計算字串長度的詳細資訊，請參閱本文件。

建置時間配額

當您建立機器人時，系統會強制執行下列最大配額。

描述	預設	可調整	自助式服務
每 AWS 帳戶的機器人	100	是	是
每個 AWS 帳戶的 Bot 頻道關聯	5,000	否	N/A
機器人網路的機器人	5	否	N/A
每個機器人的機器人	25	否	N/A
每個機器人版本	100	否	N/A
每個機器人中每個語言環境的意圖	• 1,000 在恩澳大利亞, 英 GB, 和在美 • 250 在所有其他語言環境	是	否

描述	預設	可調整	自助式服務
每個機器人中每個地區的插槽	• 4,000 在恩澳大利亞，恩-GB 和在美 • 2,000 個在所有其他地區設定	否	N/A
每個機器人語言環境的自定義	• 250 在恩澳大利亞, 英 GB, 和在美 • 100 在所有其他地區設定	否	N/A
每個機器人中每個地區的自定義插槽類型值和同義詞	50,000	否	N/A
每個機器人中每個地區設定的範例語彙中的字元總數	• 200 萬在恩澳大利亞, 恩 GB, 和恩美 • 所有其他語言環境中的 200,000 個	否	N/A
每個機器人別名的通道關	10	否	N/A
每個意圖的插槽	100	否	N/A
每個意圖的語音樣本	1,500	是	是
每個樣本語言的字元數	500	否	N/A
文字回應長度	4,000	否	N/A
每個插槽的語音範例	10	是	是
每個範例插槽語音的字元	500	否	N/A
每個插槽的提示	30	否	N/A

描述	預設	可調整	自助式服務
每個自訂插槽類型的值和同義字	10,000	否	N/A
每個自訂插槽類型值的字元數	500	否	N/A
頻道關聯名稱中的字元	100	否	N/A
您帳戶中每個區域中所有機器人的同時自動 Chatbot 設計器分析作業數	10	否	N/A
自定義語法插槽類型 XML 文件的大小	100 KB	否	N/A

執行期配額

下列最大配額會在執行階段強制執行。

描述	預設	可調整	自助式服務
輸入RecognizeText和的文字大小RecognizeUtterance	1024 個字元	否	N/A
用於RecognizeUtterance 操作的語音輸入長度	15 秒	是	否
RecognizeUtterance 標頭大小	16 KB	否	N/A

描述	預設	可調整	自助式服務
的組合要求和工作階段標頭的大小 RecognizeUtterance	12 KB	否	N/A
RecognizeText 、RecognizeUtterance 或StartConversation 的並行文字模式交談數目上限 TestBotAlias	2	否	N/A
RecognizeText 、RecognizeUtterance 或其他別名的並行文字模式交談數StartConversation 目上限	50	是	否
同時語音模式對話的最大數量 RecognizeUtterance TestBotAlias	2	否	N/A
其他別名的同時語音模式對RecognizeUtterance 話數目上限	125	是	否
同時語音模式對話的最大數量 StartConversation TestBotAlias	2	否	N/A

描述	預設	可調整	自助式服務
其他別名的同時語音模式對StartConversation 話數目上限	200	是	否
使用時PutSession ，並行階段作業管理作業數目上限 (GetSession 、或DeleteSession) TestBotAlias	2	否	N/A
使用其他別名時PutSession ，同時執行階段作業管理作業數目上限 (GetSession 、或DeleteSession)	50	是	否
Lambda 函數的最大輸入大小	12 KB	否	N/A
Lambda 函數的最大輸出大小	50 KB	否	N/A
Lambda 函數輸出中工作階段屬性的最大大小 (在 base-64 編碼之後)	12 KB	否	N/A
Lambda 函數的逾時上限	30 秒	是	否

Amazon Lex V2

Amazon Lex V2 主控台和 API 可讓您更輕鬆地建置和管理機器人。使用本指南了解在遷移機器人時，Amazon Lex V2 API 的改進功能。

您可以使用 Amazon Lex 主控台或 API 移轉機器人。如需詳細資訊，請參閱 Amazon Lex 開發人員指南中的移轉機器人。

Amazon Lex V2

多種語言可以添加到機器人，以便您可以將其作為單個資源進行管理。簡化的資訊架構可讓您有效率地管理機器人版本。諸如「對話流程」、部分儲存機器人組態以及大量上傳語音等功能，可為您提供更大的彈性。

機器人中的多種語言

您可以使用 Amazon Lex V2 API 新增多種語言。您可以獨立新增、修改和建置每種語言。插槽類型等資源的範圍在語言層級上。您可以在不同的語言之間快速移動以比較和優化對話。您可以在主控台中使用一個儀表板來檢閱所有語言的話語，以加快分析和反覆運算的速度。機器人操作員可以透過單一機器人設定來管理所有語言的權限和記錄作業。您必須提供語言做為執行階段參數，才能與 Amazon Lex V2 機器人進行交談。如需詳細資訊，請參閱亞馬遜萊克斯 V2 支援的語言和語言環境。

簡化資訊架構

Amazon Lex V2 API 遵循簡化的資訊架構 (IA)，其意圖和插槽類型範圍為某種語言。您可以在機器人級別進行版本化，以便諸如意圖和插槽類型之類的資源不會單獨版本化。默認情況下，使用可變的草稿版本創建一個機器人，用於測試更改。您可以從草稿版本建立編號的快照。您可以選擇要包含在版本中的語言。機器人內的所有資源（語言、意圖、位置類型）都會封存為建立機器人版本的一部分。如需詳細資訊，請參閱版本。

提高了建置器

您擁有額外的建置器生產力工具和功能，可讓您更靈活地控制自己的機器人設計流程。

儲存部分組態

Amazon Lex V2 API 可讓您在開發期間儲存部分變更。例如，您可以儲存參照已刪除槽類型的狹槽。這種靈活性使您可以保存工作並在以後返回。您可以在構建機器人之前解決這些更改。在 Amazon Lex V2 中，部分儲存可套用至插槽、版本和別名。

重命名資源

使用 Amazon Lex V2，您可以在資源建立後重新命名資源。使用資源名稱將易記的中繼資料與每個資源建立關聯。Amazon Lex V2 API 會為每個資源指派一個唯一的 10 個字元資源識別碼。所有資源都有資源名稱。您可以重新命名以下資源：

- 機器人
- 意圖
- 槽類型
- Slot
- 別名

您可以使用資源。若您透過AWS Command Line Interface或 Amazon Lex V2 來使用 Amazon Lex V2，特定命令將需要資源 ID。

簡化 Lambda 函數的管理

在 Amazon Lex V2 API 中，您可以為每種語言定義一個 Lambda 函數，而不是針對每個意圖定義一個函數。Lambda 函數會在語言的別名中設定，並用於對話方塊和履行程式碼掛接。您仍然可以選擇為每個意圖個別啟用或停用對話方塊和履行代碼掛接。如需詳細資訊，請參閱使用AWS Lambda函數啟用自訂邏輯。

精細設定

Amazon Lex V2 API 會將語音和意圖分類可信度分數閾值從機器人移至語言範圍。情緒分析旗標會從機器人範圍移至別名範圍。機器人範圍的工作階段逾時和隱私權設定，以及別名範圍的交談記錄保持不變。

預設後援意圖

當您建立語言時，Amazon Lex V2 API 會新增預設的後援意圖。使用它來配置機器人的錯誤處理，而不是特定的錯誤處理提示。

優化會話變量更新

使用 Amazon Lex V2 API，您可以直接使用RecognizeText和RecognizeUtterance操作更新工作階段狀態，而不需依賴工作階段 API。

使用建立 Amazon Lex V2AWS CloudFormation

Amazon Lex V2AWS CloudFormationAWS 您可以建立一個範本，描述所有所需的AWS資源 (例如 Amazon Lex V2AWS CloudFormation 定義和設定這些資源。

使用時AWS CloudFormation，您可以重複使用您的範本，重複使用您的範本，重複使用您的範本，重複使用您的範本，只需描述一次您的資源，即可在多個 AWS 帳戶 帳戶與區域內重複佈建相同資源。

Amazon Lex V2AWS CloudFormation

若要為 Amazon Lex V2 定資源和相關服務的資源，則必須了解AWS CloudFormation範本。範本是以 JSON 或 YAML 格式化的文本檔案。而您亦可以透過這些範本的說明，了解欲在 AWS CloudFormation 堆疊中佈建的資源。如果您不熟悉 JSON 或 YAML，您可以使用 AWS CloudFormation Designer 協助您開始使用 AWS CloudFormation 範本。如需詳細資訊，請參閱 AWS CloudFormation 使用者指南中的什麼是 AWS CloudFormation Designer？。

Amazon Lex V2 支援在中建立下列資源AWS CloudFormation：

- AWS::Lex::Bot
- AWS::Lex::BotAlias
- AWS::Lex::BotVersion
- AWS::Lex::ResourcePolicy

如需詳細資訊 (包括這些資源的 JSON 和 YAML 範本範例)，請參閱AWS CloudFormation使用者指南中的 Amazon Lex V2 資源類型參考資料。

進一步了解 AWS CloudFormation

若要進一步了解 AWS CloudFormation，請參閱下列資源：

- AWS CloudFormation
- AWS CloudFormation使用者指南
- AWS CloudFormation API 參考
- AWS CloudFormation命令列介面使用者指南

Amazon Lex V2 的文檔歷史

- 最新文件更新：2024 年 5 月 10 日

下表說明每個版本的 Amazon Lex V2 中的重要變更。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
更新到 AWS 受管理的策略	Amazon Lex V2 為AmazonLexReadOnly受管政策新增了新的許可，以允許讀取已在其他區域複寫的機器人資源。	2024年5月10日
更新到 AWS 受管理的策略	Amazon Lex V2 為AmazonLexFullAccess受管政策新增了新的許可，以允許將複寫的機器人資源更新到其他區域。	2024年4月15日
區域擴展	Amazon Lex V2 現在可在 AWS GovCloud (美國西部) (us-gov-west-1) 中使用。	2024年3月22日
更新到 AWS 受管理的策略	Amazon Lex V2 為AmazonLexReplicationPolicy受管政策新增了新的許可，以允許將複寫的機器人資源更新到其他區域。	2024年3月7日
新功能	您可以使用 Fn.length（）函數來確定在 Amazon Lex V2 中的字符串值的值的長度。如需詳細資訊，請參閱條件式分支-函數。	2024年3月4日
更新至特徵	Amazon Lex V2 中提供生成式人工智慧功能的 QnA 內建插槽，現已成為 GA。如需詳細資	2024年2月28日

	訊，請參閱使用生成式 AI 最佳化機器人建立與效能。	
更新到 AWS 受管理的策略	Amazon Lex V2 為AmazonLexReplicationPolicy受管政策新增了新的許可，以允許將複寫的機器人資源更新到其他區域。	2024年2月28日
更新到 AWS 受管理的策略	Amazon Lex V2 為AmazonLexFullAccess受管政策新增了新的許可，以允許將機器人資源複寫到其他區域。	2024年2月8日
新的受管理策略	Amazon Lex V2 新增受管政策，提供在其他區域複寫機器人資源的許可。如需詳細資訊，請參閱AmazonLexReplicationPolicy。	2024年2月8日
新功能	您可以使用全球備援，在 Amazon Lex V2 的第二個 AWS 區域中複寫機器人。如需詳細資訊，請參閱全域復原。	2024年2月8日
新功能	您現在可以利用 Amazon Lex V2 中的生成式人工智慧功能。如需詳細資訊，請參閱使用生成式 AI 最佳化機器人建立與效能。	2023 年 11 月 29 日
新功能	Amazon Lex V2 現在可以使用選擇性記錄功能，在意圖或插槽層級擷取文字和/或音訊。如需詳細資訊，請參閱選擇性記錄。	2023 年 11 月 8 日

新功能	Amazon Lex V2 現在可以使用內建插槽來判斷是/否/可能/不知道使用的回應。AMAZON.Confirmation 如需詳細資訊，請參閱內建插槽類型。	2023 年 8 月 17 日
新功能	除了使用分析儀表板的其他對話指標外，您還可以查看意圖和時段的績效指標。如需詳細資訊，請參閱分析。	2023 年 7 月 18 日
新功能	您可以使用「測試工作台」提高機器人的準確性和履行成功率。如需詳細資訊，請參閱測試工作台。	2023 年 6 月 6 日
新功能	現在，您可以從某些熱門商業垂直行業的模板創建機器人。如需詳細資訊，請參閱機器人範本。	2023 年 2 月 23 日
新功能	您現在可以將多個機器人合併到一個機器人網路中，以建立整合的客戶體驗。如需詳細資訊，請參閱機器人網路。	2023 年 2 月 9 日
新功能	Amazon Lex V2 現在支援阿拉伯灣 (阿拉伯聯合大公國)、廣東話 (香港)、芬蘭文 (芬蘭)、挪威文 (挪威)、波蘭文 (波蘭) 和瑞典文 (瑞典) 語言環境。如需詳細資訊，請參閱 Amazon Lex V2 支援的語言和語言環境。	2022 年 12 月 6 日

已更新為 AWS 受管理策略	Amazon Lex V2 為AmazonLexReadOnly受管政策新增了新的許可，以允許顯示自訂詞彙項目。	2022 年 11 月 29 日
新功能	Amazon Lex V2 可以使用主控台或 API 自訂語音到文字輸出，以顯示片語或單字的替代表示法。如需詳細資訊，請參閱建立語音辨識的自訂字彙。	2022 年 11 月 7 日
新功能	Amazon Lex V2 可以將權重屬性新增至項目元素，以代表語音辨識期間詞組提升的程度。如需詳細資訊，請參閱文法權重。	2022 年 10 月 28 日
新功能	Amazon Lex V2 可用來擷取由使用者使用的文字或字元組成的自由表單輸入AMAZON.FreeFormInput 。如需詳細資訊，請參閱內建插槽類型。	2022 年 10 月 19 日
新功能	Amazon Lex V2 可以在最終語音轉換為文字輸出中顯示片語或單字的替代表示法。如需詳細資訊，請參閱建立語音辨識的自訂字彙。	2022 年 10 月 19 日
新功能	Amazon Lex V2 現在支援印地文 (印度) 和荷蘭文 (荷蘭) 地區設定。如需詳細資訊，請參閱 Amazon Lex V2 支援的語言和語言環境。	2022 年 10 月 14 日

新功能	Amazon Lex V2 管理使用者輸入的方式有更新。現在，您可以選取 Amazon Lex V2 是否在交談流程中的任何時間點接受文字、音訊或 DTMF 輸入。如需詳細資訊，請參閱可配置屬性。	2022 年 9 月 22 日
新功能	Amazon Lex V2 管理交談流程的方式有更新。可視化對話構建器是一種拖放對話構建器，可輕鬆設計和可視化對話路徑。如需詳細資訊，請參閱視覺化交談產生器。	2022 年 9 月 14 日
新功能	Amazon Lex V2 構建複雜插槽的方式有一個更新。現在，您可以在插槽中創建複雜的子插槽，以便在複雜的對話設計中管理意圖。若要取得更多資訊，請參閱建立複合槽。	2022 年 9 月 9 日
新功能	Amazon Lex V2 管理與使用者交談路徑流程的方式有更新。您現在可以透過排序對話中的下一個步驟來建立複雜的對話路徑。如需詳細資訊，請參閱建立交談路徑。	2022 年 8 月 17 日
新功能	Amazon Lex V2 管理與使用者交談流程的方式有更新。您現在可以透過排序提示來建立複雜的對話。如需詳細資訊，請參閱設定提示。	2022 年 7 月 5 日

新功能	Amazon Lex V2 管理與使用者交談流程的方式有更新。您現在可以使用條件建立複雜的對話。如需詳細資訊，請參閱瞭解新的交談流程。	2022 年 5 月 3 日
新功能	為內置語法插槽類型添加了行業語法示例。如需詳細資訊，請參閱產業文法。	2022 年 3 月 22 日
新功能	已新增有關將 Amazon Lex V2 與亞馬 Amazon Chime 開發套件整合的文件。如需詳細資訊，請參閱 Amazon Chime 開發套件。	2022 年 3 月 18 日
新功能	Amazon Lex V2 現在可為語音轉錄提供可信度分數。使用分數來幫助確定用戶的正確響應。如需詳細資訊，請參閱使用語音轉錄可信度分數。	2022 年 1 月 27 日
新功能	現在，您可以在插槽中添加上下文和動態提示，以提高機器人的準確性。如需詳細資訊，請參閱使用提示來提升準確性。	2022 年 1 月 13 日
新功能	Amazon Lex V2 新增了對自訂字彙的支援，以改善音訊輸入的語音辨識。如需詳細資訊，請參閱建立自訂字彙以改善語音辨識。	2022 年 1 月 12 日
新功能	Amazon Lex V2 現在支持 AWS PrivateLink。如需詳細資訊，請參閱 VPC 端點 (AWS PrivateLink)。	2022 年 1 月 7 日

新功能	Amazon Lex V2 現在支援加泰羅尼亞語 (西班牙) 語言環境。如需詳細資訊，請參閱 Amazon Lex V2 支援的語言和語言環境。	2022 年 1 月 3 日
新功能	現在，您可以使用自己的自定義語法創建插槽類型。如需詳細資訊，請參閱使用自訂文法插槽類型。	2021 年 12 月 20 日
新功能	AWS CloudFormation 現在支持 Amazon Lex V2。如需詳細資訊，請參閱AWS CloudFormation 資源。	2021 年 12 月 20 日
新功能	Amazon Lex V2 現在支援葡萄牙文 (巴西)、葡萄牙文 (葡萄牙) 和普通話 (中國) 地區設定。如需詳細資訊，請參閱 Amazon Lex V2 支援的語言和語言環境。	2021 年 12 月 16 日
新功能	Amazon Lex V2 現在提供自動 Chatbot 設計器的預覽，協助您開始從客服中心文字稿建立聊天機器人。如需詳細資訊，請參閱使用自動 Chatbot 設計工具 (預覽)。	2021 年 12 月 1 日
新功能	您現在可以使用 spell-by-letter 和 spell-by-word 樣式來輸入 Amazon Lex V2 難以理解的插槽值。如需詳細資訊，請參閱使用拼字樣式擷取位置值。	2021 年 11 月 19 日

新功能	您現在可以使用 Amazon Polly 神經文字轉語音 (NTTS) 語音與使用者進行音訊對話。如需詳細資訊，請參閱 Amazon Polly 中的聲音。	2021 年 11 月 19 日
新功能	Amazon Lex V2 現在支援英文 (南非) 地區設定。如需詳細資訊，請參閱 Amazon Lex V2 支援的語言和語言環境。	2021 年 11 月 9 日
新功能	Amazon Lex V2 現在支援德文 (奧地利) 地區設定。如需詳細資訊，請參閱 Amazon Lex V2 支援的語言和語言環境。	2021 年 11 月 5 日
新功能	您現在可以為使用者提供更新訊息，這些訊息會在履行功能開始時以及在函數執行時定期播放。您也可以建立訊息，在功能完成時通知使用者履行狀態。如需詳細資訊，請參閱設定出貨進度更新。	2021 年 10 月 7 日
區域擴展	Amazon Lex V2 現已在非洲 (開普敦) (af-south-1) 和亞太區域 (首爾) (ap-northeast-2) 推出。	2021 年 9 月 22 日
新功能	您現在可以檢視使用者傳送至機器人的話語的統計資料。如需詳細資訊，請參閱檢視話語統計資料。	2021 年 9 月 22 日
新功能	Amazon Lex V2 現在支援韓文 (韓國) 地區設定。如需詳細資訊，請參閱 Amazon Lex V2 支援的語言和語言環境。	2021 年 9 月 9 日

新功能	Amazon Lex V2 現在提供適用於英國郵遞區號的內建插槽類型。如需詳細資訊，請參閱亞馬遜PostalCode。	2021 年 7 月 27 日
新功能	Amazon Lex V2 現在支援英文 (印度) 語言環境。如需詳細資訊，請參閱 Amazon Lex V2 支援的語言和語言環境。	2021 年 7 月 15 日
新功能	Amazon Lex V2 現在提供了一個工具，可將機器人從 Amazon Lex V1 遷移到 Amazon Lex V2 API。如需詳細資訊，請參閱 Amazon Lex 開發人員指南中的遷移機器人。	2021 年 7 月 13 日
新功能	Amazon Lex V2 現在可讓您以英文 (美國) 語言接受單一插槽的多個值。如需詳細資訊，請參閱在槽中使用多個值。	2021 年 6 月 15 日
新功能	您現在可以為英語構建更大的機器人。如需詳細資訊，請參閱 配額。	2021 年 6 月 11 日
新功能	使用 Amazon Lex V2 資源型政策來管理對機器人和機器人別名的存取。如需詳細資訊，請參閱 Amazon Lex V2 中以資源為基礎的政策。	2021 年 5 月 20 日

新功能	Amazon Lex V2 現在可讓您匯入和匯出機器人和機器人語言環境。您可以使用此功能在帳戶和 AWS 區域之間複製機器人和機器人語言環境。如需詳細資訊，請參閱匯入和匯出。	2021 年 5 月 18 日
區域擴展	Amazon Lex V2 現已在加拿大 (中部) (ca-central-1) 推出。	2021 年 5 月 17 日
新功能	Amazon Lex V2 現在支援日文 (日本) 地區設定。如需詳細資訊，請參閱 Amazon Lex V2 支援的語言和語言環境。	2021 年 4 月 1 日
新功能	Amazon Lex V2 現在支援三種新的內建插槽類型：AMAZON.City AMAZON.Country 、和AMAZON.State 。	2021 年 3 月 12 日
新的指南	這是 Amazon Lex V2 使用者指南的第一個版本。	2021 年 1 月 21 日

API 參考

API 參考現在是一個單獨的文檔。

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。

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。