aws

V2 开发人员指南

Amazon Lex

Amazon Lex: V2 开发人员指南

Table of Contents

什么是 Amazon Lex V2？

Amazon Lex V2 是一项 AWS 服务，可用于通过语音和文本构建应用程序的对话界面。Amazon Lex V2 具备自然语言理解 (NLU) 和自动语音识别 (ASR) 的深度功能性和灵活性，使您能够通过生动的对话互动构建高度参与的用户体验并创建新的产品类别。

任何开发人员都能够通过 Amazon Lex V2 快速构建对话机器人。无需深度学习专业知识，您只需在 Amazon Lex V2 控制台中指定基本对话流程即可创建机器人。Amazon Lex V2 管理对话并在对话中动态调整响应。借助此控制台，您可构建、测试和发布您的文本或语音聊天自动程序。随后，您可将对话接口添加到移动设备、Web 应用程序和聊天平台 (例如，Facebook Messenger) 上的自动程序。

Amazon Lex V2 提供 AWS Lambda 集成功能，用于集成 AWS 平台上的许多其他服务，包括 Amazon Connect、Amazon Comprehend 和 Amazon Kendra。通过与 Lambda 集成，机器人能够访问预构建的无服务器企业连接器，从而链接到 SaaS 应用程序（如 Salesforce）中的数据。

对于 2022 年 8 月 17 日之后创建的机器人，您可以通过条件分支来控制与机器人的对话流程。通过条件分支，您无需编写 Lambda 代码即可创建复杂的对话。

Amazon Lex V2 具有以下优势：

- 简易性：Amazon Lex V2 指导您通过控制台在几分钟内创建您自己的机器人。您只需提供几个示例短语，Amazon Lex V2 即可构建完整的自然语言模型，机器人可通过此模型使用语音和文本进行交互来提问、回答和完成复杂的任务。

- 大众化的深度学习技术：Amazon Lex V2 提供 ASR 和 NLU 技术来创建口语语言理解 (SLU) 系统。借助 SLU，Amazon Lex V2 采用自然语言语音和文本输入，理解输入背后的意图，并通过调用相应的业务功能来实现用户意图。

  语音识别和自然语言理解在计算机科学领域内要解决的部分最具挑战性的难题，这需要基于大量数据和基础设施来展开尖端的深度学习算法培训。Amazon Lex V2 让所有开发者都能获得深度学习技术。Amazon Lex V2 机器人将传入语音转换为文本并理解用户意图以生成智能响应，以便您能够集中精力为您的客户构建增值的机器人，从而定义可通过对话界面生成的全新的产品类别。

- 无缝部署和扩展：借助 Amazon Lex V2，您可以直接从 Amazon Lex V2 控制台构建、测试和部署机器人。通过 Amazon Lex V2，您能够发布要在移动设备、Web 应用程序和聊天服务（如 Facebook

Messenger）上使用的语音或文本机器人。Amazon Lex V2 会自动扩展。您无需担心预置硬件和管理基础设施来支持您的机器人体验。

- 与 AWS 平台的内置集成：Amazon Lex V2 可与其他 AWS 服务（例如 AWS Lambda 和 Amazon CloudWatch）进行原生运行。您可借助 AWS 平台来实施安全性、监控、用户身份验证、业务逻辑、存储和移动应用程序开发。

- 经济高效：通过使用 Amazon Lex V2，无前期成本或最低费用。您只需为发出的文本或语音请求付费。基于请求的即付即用定价和低成本使这项服务成为构建对话接口的经济高效的方式。通过 Amazon Lex V2 免费套餐，您可轻松试用 Amazon Lex V2，无需任何初期投资。

为 Amazon Lex 支付费用

Amazon Lex V2 仅针对您提出的文本或语音请求向您收费。此模型为您提供了一种可变成本服务，它可以随着您的业务增长而增长，同时为您提供 AWS 基础架构的成本优势。有关更多信息，请参阅 Amazon Lex 定价。

在注册 AWS 时，系统将在 AWS 中为您的 AWS 账户自动注册所有服务，包括 Amazon Lex。不过，您只需为使用的服务付费。如果您是 Amazon Lex 的新客户，则可以免费使用 Amazon Lex。有关更多信息，请参阅 AWS Free Tier。

若要查看您的账单，请转到 AWS Billing and Cost Management 控制台中的账单和成本管理控制面板。要了解关于AWS 账户 账单的更多信息，请参阅 AWS Billing 用户指南。如果您有关于 AWS 账单和 AWS 账户的问题，请联系 AWS 支持。

您是否是首次接触 Amazon Lex V2 的用户？

如果您是首次接触 Amazon Lex V2 的用户，我们建议您按顺序阅读以下内容：

1. 工作原理：介绍 Amazon Lex V2 以及可用来创建聊天机器人的功能。
2. Amazon Lex V2 入门：设置账户并测试 Amazon Lex V2。
3. API 参考：介绍有关 API 操作的详细信息。

最新功能

浏览以下 Amazon Lex V2 的最新功能：

主题

区域支持 AWS GovCloud（美国西部）

Amazon Lex V2 现已在 AWS GovCloud（美国西部）上市。

- Amazon Lex 终端节点和配额

Amazon Lex V2 的生成式人工智能功能

Amazon Lex V2 现在可让您的机器人利用 Amazon Bedrock 的生成式人工智能功能。

- 描述性机器人生成器

  - 新文章
  - 文档
- 辅助槽位解析
  - 新文章
  - 文档
- 言语生成
  - 新文章
  - 文档
- AMAZON.QnAIntent（对话常见问题）
  - 新文章
  - 文档
- AWS Machine Learning 博文

AMAZON.Confirmation 内置槽位，用于显示是/否/可能/不知道，以消除歧义。

Amazon Lex V2 现在提供 AMAZON.Confimation 内置槽位，以提高槽位确认和“是/否/可能/不知道”响应的准确性。

- 文档

使用 Analytics 衡量业务绩效

Amazon Lex V2 允许用户在 Analytics 控制面板上查看意图和槽位的性能。

- 新文章
- 文档

使用 Test Workbench 评估机器人性能

Amazon Lex V2 允许用户创建和运行测试集，以衡量机器人性能并改进机器人指标。

- 新文章
- 文档
- AWS Machine Learning 博文

垂直领域特定机器人模板

Amazon Lex V2 现在为用户提供了预先构建的机器人模板，包括语音和聊天模式的 ready-to-use 对话流程以及训练数据和对话提示。

- 新文章
- 文档

机器人网络

Amazon Lex V2 允许用户将多个机器人合并到一个网络中，并能够根据用户输入将请求路由到相应的机器人。

- 新文章
- 文档

可视化对话生成器

Amazon Lex V2 提供拖放式对话生成器，可在丰富的视觉环境中通过意图轻松设计和可视化对话路径。

- 新文章
- 文档
- AWS Machine Learning 博文

复合槽位类型

Amazon Lex V2 允许用户通过逻辑表达式将多个槽位组合成一个复合槽位。

- 新文章

- 文档

条件分支

Amazon Lex V2 现在允许用户编写条件，以更好地控制客户与您的机器人对话的路径。

- 新文章
- 文档

自动聊天机器人设计器

Amazon Lex V2 为用户提供了根据对话记录自动设计聊天机器人的选项。请阅读以下页面以获取使用示例。

- 新文章
- 文档
- AWS Machine Learning 博文
- Amazon Lex 自动聊天机器人设计器页面

运行时系统提示

Amazon Lex V2 为用户提供配置运行时系统提示的选项，以改善对短语的识别，从而改善槽位值的引发。

- 新文章
- 文档

自定义词汇表

Amazon Lex V2 为用户提供创建自定义词汇表的选项，该词汇表可以包含专有名词或特定领域的单词，供 Amazon Lex V2在音频输入中识别。

- 新文章
- 文档

- AWS Machine Learning 博文

语法槽位类型

Amazon Lex V2 允许用户按照语音识别语法规范 (SRGS) 创作 XML 格式的语法，以便在对话中收集信息。

- 新文章
- 文档
- AWS 机器学习博客文章

工作原理

借助 Amazon Lex V2，您能够通过文本或语音界面构建用于和用户对话的应用程序。以下是使用 Amazon Lex V2 的典型步骤：

1. 创建机器人并添加一种或多种语言。配置机器人，使其了解用户的目标，与用户进行对话以引发信息，并实现用户的意图。
2. 测试机器人。您可以使用由 Amazon Lex V2 控制台提供的测试窗口客户端。
3. 发布版本和创建别名
4. 部署机器人。您可以在自己的应用程序或消息收发平台（如 Facebook Messenger 或 Slack）上部署机器人。

开始之前，请熟悉以下 Amazon Lex V2 核心概念和术语：

- 机器人：机器人执行自动化任务，如订购披萨、预定酒店、订花等。Amazon Lex V2 机器人由自动语音识别 (ASR) 和自然语言理解 (NLU) 功能提供支持。

  Amazon Lex V2 机器人可理解通过文本或语音提供的用户输入并支持自然语言交流。
- 语言：Amazon Lex V2 机器人可以用一种或多种语言进行交流。每种语言都独立于其他语言，您可以将 Amazon Lex V2 配置为使用本地单词和短语与用户交流。有关更多信息，请参阅Amazon Lex V2 支持的语言和区域设置。
- 意图：意图表示用户要执行的操作。您创建机器人以支持一个或多个相关意图。例如，您可以创建一个披萨和饮料订购意图。对于每个目的，您需要提供以下必要信息：
  - 意图名称：意图的描述性名称。例如，**`OrderPizza`**。
  - 示例言语：用户表达意图的可能方式。例如，用户可能会说“我能订购披萨吗”和“我想订购披萨”。
  - 如何履行意图：在用户提供必要的信息后，您希望如何履行意图。建议您创建一个 Lambda 函数来履行意图。

    您可以选择对意图进行配置，使 Amazon Lex V2 将履行意图所需的必要信息返回给客户端应用程序。

  此外，Amazon Lex V2 还提供内置意图来快速设置您的机器人。有关更多信息，请参阅内置意图。

  Amazon Lex 始终包含每个机器人的回退意图。每当 Amazon Lex 无法推断出用户的意图时，就会使用回退意图。有关更多信息，请参阅AMAZON.FallbackIntent。

- 插槽：一个意图可能需要零个或零个以上的插槽或参数。您可以添加槽，作为意图配置的一部分。在运行时，Amazon Lex V2 提示用户提供特定的插槽值。用户必须为所有必需插槽提供值，然后 Amazon Lex V2 才能履行意图。

  例如，`OrderPizza` 意图需要诸如尺寸、饼皮类型和披萨数量等插槽。对于每个插槽，您需要提供插槽类型和一个或多个提示，以便 Amazon Lex V2 发送到客户端来从用户那里引发值。用户可以回复包含额外词的插槽值，如“请来一张大号披萨”或“我还是吃小号的吧”。Amazon Lex V2 仍然可以理解该插槽值。
- 插槽类型：每个插槽都具有一种类型。您可创建您自己的插槽类型或使用内置插槽类型。例如，您可针对 `OrderPizza` 目的创建并使用以下槽类型：
  - 大小：使用枚举值 `Small`、`Medium` 和 `Large`。
  - 馅饼皮：使用枚举值 `Thick` 和 `Thin`。

  Amazon Lex V2 还提供了内置插槽类型。例如，`AMAZON.Number` 是可用于订购披萨数量的内置槽类型。有关更多信息，请参阅内置意图。
- 版本：版本是您工作的带编号快照，您可以发布版本以用于您的工作流的不同阶段，如开发、测试部署和生产。创建版本后，您可以使用创建版本时存在的机器人。创建版本之后，在您继续使用应用程序时它将保持不变。
- 别名：别名是指向机器人特定版本的指针。通过别名，您可以更新您的客户端应用程序正在使用的版本。例如，您可以将别名指向您机器人的版本 1。当您准备更新机器人时，您可以发布版本 2，然后更改别名以指向新版本。由于您的应用程序使用的是别名而不是特定版本，因此您的所有客户端无需进行更新即可获得新功能。

有关 Amazon Lex V2 可用 AWS 区域的列表，请参阅 Amazon Web Services 一般参考指南中的 Amazon Lex V2 端点和限额。

Amazon Lex V2 支持的语言和区域设置

Amazon Lex V2 支持多种语言和区域设置。本主题提供了支持的语言、支持这些语言的功能以及用于提高机器人性能的特定语言指南。

支持的语言和区域设置

Amazon Lex V2 支持以下语言和区域设置。

代码	语言和区域设置
ar_AE	海湾阿拉伯语（阿拉伯联合酋长国）
ca_ES	加泰罗尼亚语（西班牙）
de_AT	德语（奥地利）
de_DE	德语（德国）
en_AU	英语（澳大利亚）
en_GB	英语（英国）
en_IN	英语（印度）
en_US	英语（美国）
en_ZA	英语（南非）
es_419	西班牙语（拉丁美洲）
es_ES	西班牙语（西班牙）
es_US	西班牙语（美国）
fi_FI	芬兰语（芬兰）
fr_CA	法语（加拿大）
fr_FR	法语（法国）
hi_IN	印地语（印度）
it_IT	意大利语（意大利）
ja_JP	日语（日本）
ko_KR	韩语（韩国）
nl_NL	荷兰语（荷兰）

代码	语言和区域设置
no_NO	挪威语（挪威）
pl_PL	波兰语（波兰）
pt_BR	葡萄牙语（巴西）
pt_PT	葡萄牙语（葡萄牙）
sv_SE	瑞典语（瑞典）
zh_CN	普通话（中国）
zh_HK	广东话（香港）

Amazon Lex V2 功能支持的语言和区域设置

下表列出了仅限于某些语言和区域设置的 Amazon Lex V2 功能。其他 Amazon Lex V2 功能在所有语言和区域设置中均支持。

功能	支持的语言和区域设置
亚马逊。 AlphaNumeric	除韩语 (ko_KR) 之外的所有语言和区域设置
AMAZON.KendraSearchIntent	英语（美国）(en_US)
使用自定义词汇改善语音识别	英语（英国）(en_GB) 英语（美国）(en_US)
自动聊天机器人设计器	英语（美国）(en_US)
区域可用性	以下语言和区域设置在亚太地区（新加坡）(ap-southeast-1) 和非洲（开普敦）(ap-south-1) 区域不可用： • 海湾阿拉伯语（阿拉伯联合酋长国）(ar_AE) • 加泰罗尼亚语（西班牙）(ca_ES)

功能	支持的语言和区域设置
	• 芬兰语（芬兰）(fi_FI) • 印地语（印度）(hi_IN) • 荷兰语（荷兰）(nl_NL) • 挪威语（挪威）(no_NO) • 波兰语 (pl_PL) • 葡萄牙语（巴西）(pt_BR) • 葡萄牙语（葡萄牙）(pt_PT) • 瑞典语 (sv_SE) • 普通话（中国）(zh_CN) • 广东话（香港）(zh_HK)
设置意图上下文	英语（美国）(en_US)
语法插槽类型	英语（澳大利亚）(en_AU) 英语（英国）(en_GB) 英语（美国）(en_US)
使用一个插槽中的多个值	英语（美国）(en_US)
使用运行时提示改善对插槽值的识别	英语（英国）(en_GB) 英语（美国）(en_US)
使用拼写样式捕获槽位值	英语（澳大利亚）(en_AU) 英语（英国）(en_GB) 英语（美国）(en_US)
使用置信度分数	英语（英国）(en_GB) 英语（美国）(en_US)

Amazon Lex V2 的语言指南

为了提高您的机器人性能，在以下语言中，您应该遵循以下准则：

阿拉伯语

Amazon Lex V2 所用的阿拉伯语种类是海湾阿拉伯语。当您为机器人提供示例言语时，请记住这一点。请注意，阿拉伯语文字是从右向左书写的。

印地语

Amazon Lex V2 能够为在印地语和英语之间自由切换的印地语终端用户提供服务。如果您计划构建支持这种语言切换的机器人，建议以下最佳实践：

- 在机器人定义中，用拉丁字母书写英语单词。
- 至少 50% 的示例言语应代表同一句子中的语言切换。在这些言语中，对印地语单词使用天城文，对英语单词使用拉丁文（例如，“मैं ticket book करना चाहता हूं।”）。
- 如果您希望用户使用拉丁文中的印地语单词或天城文中的英语单词与机器人通信，那么应该在示例言语的拉丁文中加入印地语单词的示例（例如，“mujhe ek ticket book karni hai”）以及在天城文中加入英语单词示例（例如，“मेझ टकिट की बुकंगि में मदद चाहएि”）。
- 如果您希望用户完全使用印地语或英语语句与机器人交流，则应包括完全使用一种语言的示例言语（例如，“我想订票”）。

区域

有关提供了 Amazon Lex V2 的 AWS 区域的列表，请参阅 AWS 一般参考 中的 AWS 区域和端点。

Amazon Lex V2 入门

Amazon Lex V2 提供 API 操作，您可以将这些操作与现有应用程序集成。有关支持操作的列表，请参阅 API 参考。您可以使用以下任意选项：

- AWS 软件开发工具包：使用软件开发工具包时，会使用您提供的凭证对 Amazon Lex V2 请求自动进行签名和身份验证。建议您使用软件开发工具包构建应用程序。
- AWS CLI — 您无需编写任何代码 AWS CLI 即可使用任何 Amazon Lex V2 功能。
- AWS 控制台：该控制台是开始测试和使用 Amazon Lex V2 的最简单方法。

如果您是首次使用 Amazon Lex V2，建议先阅读 工作原理。

主题

步骤 1：设置 AWS 账户并创建管理员用户

首次使用 Amazon Lex V2 前，请完成以下任务：

1. 报名参加 AWS
2. 创建 IAM 用户

报名参加 AWS

如果您已经有一个 AWS 帐户，请跳过此任务。

当您注册 Amazon Web Services (AWS) 时，您的 AWS 账户将自动注册使用中的所有服务 AWS，包括 Amazon Lex V2。您只需为使用的服务付费。

使用 Amazon Lex V2 时，您仅需为实际使用的资源付费。如果您是新 AWS 客户，则可以免费开始使用 Amazon Lex V2。有关更多信息，请参阅 AWS 免费使用套餐。

如果您已经有一个 AWS 帐户，请跳到下一个任务。如果您没有 AWS 帐户，请按照以下步骤创建一个。

创建 AWS 账户

1. 打开 https://portal.aws.amazon.com/billing/signup。
2. 按照屏幕上的说明进行操作。

   在注册时，将接到一通电话，要求使用电话键盘输入一个验证码。

   当您注册时 AWS 账户，就会创建AWS 账户根用户一个。根用户有权访问该账户中的所有 AWS 服务 和资源。作为安全最佳实践，请为用户分配管理访问权限，并且只使用根用户来执行需要根用户访问权限的任务。

记下您的 AWS 账户 ID，因为下个任务需要用到它。

创建 IAM 用户

中的 AWS服务（例如 Amazon Lex V2）要求您在访问时提供凭证，以便服务可以确定您是否有权访问该服务所拥有的资源。

创建一个 IAM 用户账户以访问您的 Amazon Lex V2 账户：

- 使用 AWS Identity and Access Management (IAM) 创建 IAM 用户
- 将用户添加到具有管理权限的 IAM 组中。
- 向您创建的 IAM 用户授予管理权限。

然后，您可以使用特殊的 URL 和 IAM 用户的证书 AWS 进行访问。

本指南中的入门练习假定您拥有具有管理权限的用户 (adminuser)。请按照以下过程在您的账户中创建 adminuser。

创建管理员用户和登录控制台

1. adminuser在您的 AWS 账户中创建一个名为的管理员用户。有关说明，请转到 IAM 用户指南中的创建您的第一个 IAM 用户和管理员组。
2. 作为用户，您可以使用特殊的 URL 登录。 AWS Management Console 有关更多信息，请参阅《IAM 用户指南》https://docs.aws.amazon.com/IAM/latest/UserGuide/getting-started_how-users-sign-in.html中的用户如何登录您的账户。

有关 IAM 的更多信息，请参阅以下文档：

- AWS Identity and Access Management (IAM)
- 入门
- IAM 用户指南

授予编程式访问权限

如果用户想在 AWS 外部进行交互，则需要编程访问权限 AWS Management Console。授予编程访问权限的方式取决于正在访问的用户类型 AWS。

要向用户授予编程式访问权限，请选择以下选项之一。

哪个用户需要编程式访问权限？	目的	方式
人力身份 （在 IAM Identity Center 中管理的用户）	使用临时证书签署向 AWS CLI、 AWS 软件开发工具包或 AWS API 发出的编程请求。	按照您希望使用的界面的说明进行操作。 • 有关的 AWS CLI，请参阅《AWS Command Line Interface 用户指南》 AWS IAM Identity Center中的 “配置 AWS CLI 要使用”。 • 有关 AWS 软件开发工具包、工具和 AWS API，请参阅《软件开发工具包和AWS 工具参考指南》中的 IAM 身份中心身份验证。
IAM	使用临时证书签署向 AWS CLI、 AWS 软件开发工具包或 AWS API 发出的编程请求。	按照 IAM 用户指南中的将临时证书与 AWS 资源配合使用中的说明进行操作。
IAM	（不推荐使用） 使用长期凭证签署向 AWS CLI、 AWS 软件开发工具包或 AWS API 发出的编程请求。	按照您希望使用的界面的说明进行操作。 • 有关信息 AWS CLI，请参阅用户指南中的使用 IAM 用

哪个用户需要编程式访问权限？	目的	方式
		户证书进行身份验证。AWS Command Line Interface • 有关 AWS SDK 和工具，请参阅 S AWS DK 和工具参考指南中的使用长期凭证进行身份验证。 • 有关 AWS API，请参阅 IAM 用户指南中的管理 IAM 用户的访问密钥。

后续步骤

步骤 2：入门（控制台）

步骤 2：入门（控制台）

学习如何使用 Amazon Lex V2 的最简单方法是使用控制台。为了便于您入门，我们创建了以下练习，所有练习都使用控制台进行：

- 练习 1：使用蓝图创建 Amazon Lex V2 机器人，蓝图是预定义的机器人，提供了所有必要的机器人配置。您只需做最少的工作即可测试 end-to-end 设置。
- 练习 2：查看您的客户端应用程序和 Amazon Lex V2 机器人之间发送的 JSON 结构。

主题

- 练习 1：根据示例创建机器人
- 练习 2：查看对话流程

练习 1：根据示例创建机器人

在本练习中，您将创建自己的第一个 Amazon Lex V2 机器人并在 Amazon Lex V2 控制台中对其进行测试。在此练习中，使用 OrderFlowers 示例。

示例概览

您可以通过 OrderFlowers 示例来创建 Amazon Lex V2 机器人。有关机器人结构的更多信息，请参阅工作原理。

- 意图：OrderFlowers
- 槽类型：一个称为 `FlowerTypes` 的自定义槽类型，具有枚举值：`roses`、`lilies` 和 `tulips`。
- 槽：在机器人实现此意图之前，意图需要以下信息（即槽）。
  - `PickupTime`（AMAZON.TIME 内置类型）
  - `FlowerType`（FlowerTypes 自定义类型）
  - `PickupDate`（AMAZON.DATE 内置类型）
- 表达：以下示例表达表示用户的意图：
  - “我想要取花。”
  - “我想要订些花。”
- 提示：在机器人确定此意图后，它会使用以下提示来填充槽：
  - 用于 `FlowerType` 槽的提示：“您想要订哪种类型的花？”
  - `PickupDate` 槽的提示：“您想在哪天来取 {FlowerType}？”
  - `PickupTime` 槽的提示：“您想在什么时间来取 {FlowerType}？”
  - 确认语句：“好的，您可于 {PickupDate} {PickupTime} 来取您的 {FlowerType}。这样可以吗？”

创建 Amazon Lex V2 机器人（控制台）

1. 登录 AWS Management Console，并通过以下网址打开 Amazon Lex 控制台：https://console.aws.amazon.com/lex/。
2. 选择创建机器人。
3. 对于创建方法，请选择从示例开始。
4. 在示例机器人部分，从列表中选择 OrderFlowers。
5. 在机器人配置部分，为机器人指定名称和（可选）描述。该名称在您的账户中必须是唯一的。
6. 在权限部分，选择使用基本 Amazon Lex 权限创建新角色。这将创建一个 AWS Identity and Access Management (IAM) 角色，该角色具有 Amazon Lex V2 运行您的机器人所需的权限。
7. 在儿童在线隐私保护法 (COPPA) 部分，做出相应的选择。
8. 在会话超时和高级设置部分，保留默认值。

9. 选择下一步。Amazon Lex V2 为您创建了机器人。

在创建机器人之后，您必须添加机器人支持的一种或多种语言。每种语言包含机器人用来与用户对话的意图、插槽类型和插槽。

为机器人添加语言

1. 在语言部分，选择支持的语言并添加描述。
2. 保留语音交互和意图分类置信度分数阈值字段的默认值。
3. 选择完成，为向机器人添加语言。

选择完成后，控制台将打开意图编辑器。您可以通过意图编辑器来检查机器人使用的意图。完成对机器人的检查后，您可以对其进行测试。

测试 OrderFlowers 机器人

1. 在页面顶部，选择构建。等待机器人构建。
2. 构建完成后，选择测试以打开测试窗口。
3. 测试机器人。从其中一句言语样本开始对话，例如“我想去摘花。”

后续步骤

您已经通过模板创建了第一个机器人。您可以通过控制台创建自己的机器人。有关创建自定义机器人的说明以及创建机器人的更多信息，请参阅 构建机器人。

练习 2：查看对话流程

在本练习中，您可以查看在您的客户端应用程序和您在 练习 1：根据示例创建机器人 中创建的 Amazon Lex V2 机器人之间发送的 JSON 结构。该对话通过 RecognizeText 操作生成 JSON 结构。RecognizeUtterance 返回的信息与响应中的 HTTP 标头相同。

JSON 结构按对话的每个回合划分。一个回合是指来自客户端应用程序的请求和来自机器人的响应。

第 1 回合

在对话的第 1 回合，客户端应用程序会启动与您的机器人的对话。请求 URI 和正文均提供请求相关信息。

```
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 标识正在与客户端应用程序通信的机器人。它还包括由客户端应用程序生成的会话标识符，用于标识用户和机器人之间的特定对话。
- 请求的正文包含用户在客户端应用程序中键入的文本。在这种情况下，只发送文本，但是您的应用程序可以发送其他信息，例如请求属性或会话状态。有关更多信息，请参阅 RecognizeText 操作。

根据 `text`，Amazon Lex V2 可以检测到用户订购鲜花的意图。Amazon Lex V2 选择一个意图的插槽 (`FlowerType`) 和该插槽的一个提示，然后向客户端应用程序发送以下响应。客户端向用户显示响应。

```
{
    "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 回合

在第 2 回合中，用户响应第 1 回合中 Amazon Lex V2 机器人的提示，为 FlowerType 插槽填充一个值。

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

第 2 回合的响应显示 FlowerType 插槽已填满，并提供提示以引发下一个插槽值。

```
{
    "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 回合

在第 3 回合中，用户响应第 2 回合中 Amazon Lex V2 机器人的提示，为 PickupDate 插槽填充一个值。

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

第 3 回合的响应显示 FlowerType 和 PickupDate 插槽都已填满，并提供提示以引发最后一个插槽值。

```
{
    "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 回合

在第 4 回合中，用户提供意图的最终插槽值，即提取鲜花的时间。

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

在响应中，Amazon Lex V2 向用户发送确认提示以确认订单是否正确。设置 `dialogAction` 为 `ConfirmIntent`，`confirmationState` 为 `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 回合

在最后一回合中，用户响应确认提示。

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

在该响应中，Amazon Lex V2 通过将 `confirmationState` 设置为 `Confirmed` 并将 `dialogAction` 设置为 close 来表示该意图已完成。所有插槽值均可供客户端应用程序使用。

```
{
    "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"
    }
}
```

构建机器人

您可以创建 Amazon Lex V2 机器人来与用户交互，从而引发完成任务所需的信息。例如，您可以创建一个机器人来收集订购一束鲜花或预订酒店房间所需的信息。

要构建机器人，您需要以下信息：

1. 机器人用来与客户交互的语言。您可以选择一种或多种语言，每种语言均包含独立的意图、槽位和槽位类型。
2. 机器人帮助用户履行的意图或目标。机器人可以包含一个或多个意图，例如订购鲜花，或者预订酒店和租车。您需要决定用户发起意图时所用的语句或言语。
3. 为履行意图而需要从用户收集的信息或槽位。例如，您可能需要用户提供鲜花的类型或酒店预订的开始日期。您需要定义 Amazon Lex V2 用来从用户引发槽位值的一个或多个提示。
4. 您需要用户提供的槽位类型。您可能需要创建自定义槽位类型，例如用户可能订购的鲜花清单，也可以使用内置的槽位类型，例如针对预订开始日期使用 AMAZON.Date 槽位类型。
5. 意图内和意图之间的用户交互流程。您可以配置对话流程以定义用户与机器人之间在调用意图后的交互。您可以创建 Lambda 函数来验证和履行意图。

主题

> **Note**
>
> Amazon Lex V2 于 2022 年 8 月 17 日发布了对用户对话管理方式的更改。借助此更改，您可以更好地控制用户的对话路径。有关更多信息，请参阅了解对话流管理。2022 年 8 月 17 日之前创建的机器人不支持对话框代码挂钩消息、设置值、配置后续步骤和添加条件。

了解对话流管理

Amazon Lex V2 于 2022 年 8 月 17 日发布了对用户对话管理方式的更改。借助此更改，您可以更好地控制用户的对话路径。

在变更之前，Amazon Lex V2 基于意图中各个插槽的优先级来引发相应的插槽，从而管理对话。您可以使用 Lambda 函数中的 DialogAction 根据用户输入动态地修改此行为并更改对话路径。为此，可以跟踪对话的当前状态，并且根据会话状态以编程方式决定下一步的操作。

借助此更改，您可以使用 Amazon Lex V2 控制台或 API 创建对话路径和条件分支，而无需使用 Lambda 函数。Amazon Lex V2 跟踪对话状态，并根据创建机器人时定义的条件来控制下一步的操作。借此，您可以在设计机器人时轻松创建复杂的对话。

这些更改可帮助您完全控制与客户的对话。但是，您无需定义路径。如果您未指定对话路径，Amazon Lex V2 将根据意图中的插槽优先级来创建默认路径。您可以继续使用 Lambda 函数来动态定义对话路径。在这种情况下，将根据在 Lambda 函数中配置的会话状态来恢复对话。

本次更新提供：

- 用于创建具有复杂对话流程的机器人的全新控制台体验。
- 对用于创建机器人以支持新对话流程的现有 API 做出更新。
- 意图调用时发送消息的初始响应。
- 对插槽引发的新响应、Lambda 调用作为对话框代码挂钩和确认。
- 在对话的每个回合指定后续步骤的功能。
- 用于设计多个对话路径的条件评估。
- 在对话期间随时设置插槽值和会话属性。

对于旧版的机器人，请注意以下几点：

- 2022 年 8 月 17 日之前创建的机器人将继续使用旧机制来管理对话流程。在该日期之后创建的机器人使用新的对话流程管理方式。
- 2022 年 8 月 17 日之后通过导入创建的新机器人将使用新的对话流程管理方式。现有机器人上的导入继续使用旧的对话管理方式。
- 要为 2022 年 8 月 17 日之前创建的机器人启用新的对话流程管理，请导出该机器人，然后使用新的机器人名称导入该机器人。通过导入创建的机器人将使用新的对话流程管理。

对于在 2022 年 8 月 17 日之后创建的新机器人，请注意以下几点：

- Amazon Lex V2 完全遵循定义的对话流程，旨在提供所需的体验。您应该配置所有流程分支，以避免在运行时出现默认对话路径。
- 代码挂钩之后的对话步骤需要完整地配置，因为不完整的步骤可能会导致机器人失败。我们建议您验证在 2022 年 8 月 17 日之前创建的机器人，因为对于这些机器人，系统不会自动验证代码挂钩之后的对话步骤。

创建自动程序

您可以通过以下方式使用 Amazon Lex V2 创建机器人：

1. 使用 Amazon Lex V2 控制台通过网站界面创建机器人。有关更多信息，请参阅使用 Amazon Lex V2 控制台创建机器人。
2. 使用描述性机器人生成器利用 Amazon Bedrock 的生成式人工智能功能创建机器人。有关更多信息，请参阅使用描述性机器人生成器。
3. 使用机器人模板创建与常见业务用例相匹配的预配置机器人。有关更多信息，请参阅从机器人模板生成预定义的机器人。
4. 使用 AWS SDK 通过 API 操作创建机器人。
5. 使用自动聊天机器人设计器，通过代理与客户之间的现有聊天转录来创建机器人。有关更多信息，请参阅使用自动聊天机器人设计器。
6. 导入现有的机器人定义。有关更多信息，请参阅导入。
7. 使用 AWS CloudFormation 创建机器人。有关更多信息，请参阅使用 AWS CloudFormation 创建 Amazon Lex V2 资源。

主题

- 使用 Amazon Lex V2 控制台创建机器人
- 从机器人模板生成预定义的机器人
- 使用自动聊天机器人设计器

使用 Amazon Lex V2 控制台创建机器人

通过定义名称、描述和一些基本信息开始创建您的机器人。

要创建机器人，请执行以下操作：

1. 登录到 AWS Management Console，并通过以下网址打开 Amazon Lex 控制台：https://console.aws.amazon.com/lex/。
2. 选择创建机器人。
3. 在创建方法部分中，选择创建。
4. 在机器人配置部分中，指定机器人的名称和描述（可选）。
5. 在 IAM 权限部分中，选择 AWS Identity and Access Management (IAM) 角色，该角色为 Amazon Lex V2 提供访问其他 AWS 服务（例如 Amazon CloudWatch）的权限。您可以通过 Amazon Lex V2 创建角色，也可以选择具有 CloudWatch 权限的现有角色。
6. 在儿童在线隐私保护法 (COPPA) 部分中，选择相应的回应。
7. 在空闲会话超时部分中，选择 Amazon Lex V2 保持用户会话打开的持续时间。Amazon Lex V2 会在会话期间保留会话变量，以便您的机器人可以使用相同的变量恢复对话。
8. 在高级设置部分中，添加标签以便帮助识别机器人并且用于控制访问和监控资源。
9. 选择下一步创建机器人，然后进入添加语言的过程。

从机器人模板生成预定义的机器人

Amazon Lex V2 提供预构建的解决方案以便实现大规模的用户体验，并鼓励用户参与数字互动。这些预构建的机器人模板可实现客户体验的自动化和标准化。机器人模板中提供了直接可用的对话流程，以及适用于语音和聊天模式的训练数据和对话框提示。您可以在优化资源的同时加快机器人解决方案的交付，从而专注于客户关系的开发。

您可以根据自己的业务用例创建预构建的机器人。您可以使用 AWS CloudFormation 控制台选择相关服务的预构建选项，例如Amazon S3、Amazon Connect 和 DynamoDB。

目前，Amazon Lex V2 支持以下垂直行业：

- 金融服务
- 零售订购
- 汽车保险
- 电信
- 航空服务
- 更多...

您可以使用所提供的业务解决方案模板来构建机器人，并根据您的业务需求来对机器人进行自定义。

> **Note**
>
> 这些模板通过 AWS CloudFormation 堆栈创建 Amazon Lex V2 之外的资源。可能需要在其他控制台（例如 Lambda 和 DynamoDB）中修改该堆栈。

构建和部署机器人模板所需的先决条件：

- AWS 账户
- 拥有对以下 AWS 服务的访问权限：
  - Amazon Lex V2，用于创建机器人
  - Lambda，用于企业登录功能
  - DynamoDB，用于创建表格
  - IAM 访问权限，用于创建策略和角色
  - AWS CloudFormation，用于运行堆栈
- IAM 访问权限和私有密钥凭证
- Amazon Connect 实例（可选）

> **Note**
>
> 使用不同的 AWS 服务会产生对应于每项服务的相应使用成本。

要使用 Amazon Lex V2 模板构建机器人，请执行以下操作：

1. 登录到 AWS Management Console，并通过以下网址打开 Amazon Lex 控制台：https://console.aws.amazon.com/lex/。
2. 点击橙色按钮从模板创建机器人。
3. 选择要用于机器人模板的垂直行业。注意：目前有 5 个机器人模板可用。即将推出更多可用模板。
4. 对您要使用的模板选择创建。AWS CloudFormation 中将打开一个选项卡，以便您在其中编辑 AWS CloudFormation 堆栈的参数。您所选模板的所有选项都已完成。您还可以通过选择了解详情来详细了解机器人模板的工作原理。

5. 在 AWS CloudFormation 控制台中，AWS CloudFormation 将为您所选模板的每个值创建默认配置。您还可以选择自己的堆栈名称、AWS CloudFormation 参数、Amazon DynamoDB 表和 Amazon Connect 参数（可选）。
6. 在窗口底部，选择创建堆栈。
7. AWS CloudFormation 在后台处理请求以配置您的新机器人。该过程需要几分钟完成。注意：该过程会自动为 DynamoDB 表、Amazon Connect 联系流程和 Amazon Connect 实例创建资源。您可以在 AWS CloudFormation 控制台中跟踪进度，然后在 CloudFormation 堆栈创建完成后返回到 Amazon Lex V2 控制台。
8. 如果成功构建，则会显示一条消息。您可以选择前往机器人列表以进入机器人页面，从而在该页面中找到可以测试和使用的新机器人。

配置机器人模板

Lambda 函数：机器人模板会自动为您的部署创建所需的 Lambda 函数。如果模板解决方案中包含多个机器人，则 AWS CloudFormation 参数中会列出多个 Lambda 函数。如果您要与机器人一起部署现有 Lambda 函数，则可以输入自定义 Lambda 函数的名称。

Amazon DynamoDB：机器人模板会自动创建加载示例策略数据所需的 DynamoDB 表。您也可以输入您的自定义 DynamoDB 表的名称。您的自定义 DynamoDB 表的格式应与机器人模板部署创建的默认表相同。

Amazon Connect：您可以通过输入 ConnectInstanceARN 和唯一的 ContactFlowName，将自己的 Amazon Connect 实例配置为使用新的机器人模板。在使用 Amazon Connect 的情况下，您可以使用端到端的 IVR 系统测试您的机器人。

对机器人模板进行故障排除

- 检查您是否拥有创建所选模板的对应权限。用户需要 CloudFormation:CreateStack 权限以及对模板中列出的 AWS 资源的权限。需要用户权限的资源列表位于创建模板页面的底部。
- 如果您的机器人模板创建失败，Amazon Lex V2 控制台中的红色横幅会提供指向负责创建该模板的 AWS CloudFormation 堆栈的链接。在 AWS CloudFormation 控制台中，您可以查看事件选项卡，以查看导致模板失败的具体错误。查看 AWS CloudFormation 错误后，请参阅 CloudFormation 故障排除以了解更多信息。
- 机器人模板仅适用于示例数据。您必须将数据填入 DynamoDB 表中，以便模板使用您的自定义数据。

使用自动聊天机器人设计器

> **Note**
>
> 您只能使用英语（美国）的转录。

自动聊天机器人设计器可帮助您根据现有对话转录来设计机器人。该自动聊天机器人设计器对转录进行分析，并提出包含意图和插槽类型的初始设计。您可以迭代机器人设计，添加提示，以及构建、测试和部署机器人。

使用 Amazon Lex V2 控制台或 API 创建新机器人或向机器人添加语言后，您可以上传双方对话的转录。自动聊天机器人设计器将对转录进行分析，并确定机器人的意图和插槽类型。该设计器还会对影响特定意图或插槽类型的创建的对话进行标记以供您审阅。

您可以使用 Amazon Lex V2 控制台或 API 来分析对话转录，并就机器人的意图和插槽类型提出相关建议。

聊天机器人设计器完成分析后，您可以查看建议的意图和插槽类型。添加建议的意图或插槽类型后，您可以使用控制台或 API 对其进行修改或将其从机器人设计中删除。

自动聊天机器人设计器支持使用 Contact Lens for Amazon Connect 架构的对话转录文件。如果您使用的是其他联络中心应用程序，则必须将对话转录转换为聊天机器人设计器使用的格式。有关信息，请参阅输入转录格式。

要使用自动聊天机器人设计器，您必须向运行该设计器的 IAM 角色授予相关访问权限。有关具体的 IAM 策略，请参阅允许用户使用自动聊天机器人设计器。要使 Amazon Lex V2 能够使用可选 AWS KMS 密钥加密输出数据，您需要使用 允许用户使用密 AWS KMS 钥加密和解密文件 中所示的策略更新密钥。

> **Note**
>
> 如果您使用 KMS key，则无论所用的 IAM 角色，都必须提供 KMS key 策略。

主题

- 导入对话转录
- 创建意图和插槽类型

- 输入转录格式
- 输出转录格式

导入对话转录

导入对话转录的过程包括三个步骤：

1. 将转录转换为正确的格式，为导入做好准备。如果您使用的是 Contact Lens for Amazon Connect，则转录已经是正确的格式。
2. 将转录上传到 Amazon S3 存储桶。如果您使用的是 Contact Lens，则您的转录已在 S3 存储桶中。
3. 使用 Amazon Lex V2 控制台或 API 操作对转录进行分析。完成训练所需要的时间取决于转录的数量以及对话的复杂性。通常，每分钟分析 500 行转录。

其中的各个步骤将在后文中予以说明。

从 Contact Lens for Amazon Connect 导入转录

Amazon Lex V2 自动聊天机器人设计器与 Contact Lens 转录文件兼容。要使用 Contact Lens 转录文件，必须打开 Contact Lens 并记下其输出文件的位置。

从 Contact Lens 导出转录

1. 在您的 Amazon Connect 实例中打开 Contact Lens。有关说明，请参阅《Amazon Connect 管理员指南》中的启用 Contact Lens for Amazon Connect。
2. 记下 Amazon Connect 用于您的实例的 S3 存储桶的位置。要查看位置，请在 Amazon Connect 控制台中打开数据存储页面。有关说明，请参阅《Amazon Connect 管理员指南》中的更新实例设置。

打开 Contact Lens 并记下转录文件位置后，请访问使用 Amazon Lex V2 控制台分析转录以查看转录导入和分析的说明。

准备转录

创建转录文件，以准备好您的转录。

- 为每个对话创建一个转录文件，以便列出各方之间的交互。对话中的每次交互可能有多行内容。您可以提供对话的已编辑版本和未编辑版本。
- 文件必须为输入转录格式中指定的 JSON 格式。

- 您必须提供至少 1,000 个对话回合。为了更好地发现您的意图和插槽类型，您应该提供约 10,000 个或更多个对话回合。自动聊天机器人设计器仅处理前 700,000 个回合。
- 您可以上传的转录文件没有数量限制，也没有文件大小限制。

如果您计划按日期筛选已导入的转录，则这些文件必须位于以下目录结构中：

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

转录文件的文件名中必须包含“yyyy-mm-dd”格式的日期。

从其他联络中心应用程序导出转录

1. 使用联络中心应用程序的工具导出对话。对话必须至少包含输入转录格式中指定的信息。
2. 将您的联络中心应用程序生成的转录转换为输入转录格式中所述的格式。该转换应由您负责执行。

我们提供三个用于准备转录的脚本。它们是：

- 将 Contact Lens 转录与 Amazon Lex V2 对话日志结合起来的脚本。Contact Lens 转录不包括与 Amazon Lex V2 机器人交互的 Amazon Connect 对话部分。该脚本要求为 Amazon Lex V2 开启对话日志，同时需要查询对话日志 CloudWatch Logs 和 Contact Lens S3 存储桶的相应权限。
- 用于将 Amazon Transcribe 通话分析功能转换为 Amazon Lex V2 输入格式的脚本。
- 用于将 Amazon Connect 聊天转录转换为 Amazon Lex V2 输入格式的脚本。

您可以访问以下 GitHub 存储库下载脚本：https://github.com/aws-samples/amazon-lex-bot-recommendation-integration。

将转录上传到 S3 存储桶

如果您使用的是 Contact Lens，则您的转录文件已包含在 S3 存储桶中。有关转录文件的位置和文件名，请参阅《Amazon Connect 管理员指南》中的示例 Contact Lens 输出文件。

如果您正在使用其他联络中心应用程序，但尚未为转录文件设置 S3 存储桶，请按照以下步骤操作。否则，如果您已有 S3 存储桶，则在登录 Amazon S3 控制台后，请从此过程的第 5 步开始执行。

将文件上传到 S3 存储桶

1. 登录到 AWS Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 请选择 Create bucket（创建桶）。
3. 指定存储桶的名称，然后选择“区域”。该区域必须与您在 Amazon Lex V2 中使用的区域相同。根据需要针对您的用例设置其他选项。
4. 请选择 Create bucket（创建桶）。
5. 在存储桶列表中，选择现有存储桶或者您刚创建的存储桶。
6. 请选择 Upload（上传）。
7. 添加您要上传的转录文件。
8. 请选择 Upload（上传）。

使用 Amazon Lex V2 控制台分析转录

您只能使用空语言的自动机器人设计。可以将新语言添加到现有机器人，也可以创建新机器人。

要在新机器人中创建新语言，请执行以下操作：

1. 登录到 AWS Management Console，并通过以下网址打开 Amazon Lex 控制台：https://console.aws.amazon.com/lex/。
2. 选择创建机器人。
3. 选择从自动聊天机器人设计器开始。填写信息以创建机器人。
4. 选择 Next（下一步）。
5. 在为机器人添加语言中，填写语言信息。
6. 在 S3 上的转录文件位置部分中，选择包含您的转录文件的 S3 存储桶，以及文件的本地路径（如有必要）。
7. 可选的选项如下：
   - 在处理过程中对转录数据进行加密的 AWS KMS 密钥。如果不选择密钥，则使用服务 AWS KMS 密钥。
   - 按照特定的日期范围对转录进行筛选。如果您选择对转录进行筛选，则转录文件必须位于正确的文件夹结构中。有关更多信息，请参阅准备转录。
8. 选择完成。

等待 Amazon Lex V2 处理转录。分析完成后，系统将显示一条完成消息。

如何停止分析转录

如果您需要停止对已上传转录的分析，则可以停止 BotRecommendationStatus 状态为“正在处理”的正在运行的 BotRecommendation 任务。从控制台提交任务后，或者使用适用于 StopBotRecommendation API 的 CLI SDK，您可以点击横幅上显示的停止处理按钮。有关更多信息，请参阅 StopBotRecommendation

调用 StopBotRecommendation 后，内部 BotRecommendationStatus 被设置为 Stopping，您无需支付任何费用。要确保任务已停止，您可以调用 DescribeBotRecommendation API 并验证 BotRecommendationStatus 是否为 Stopped。此过程通常需要 3 到 4 分钟。

调用 StopBotRecommendation API 后，您无需支付处理费用。

创建意图和插槽类型

聊天机器人设计器创建意图和插槽类型后，您可以选择要添加到机器人中的意图和插槽类型。您可以查看每种意图和插槽类型的详细信息，以帮助您确定哪些建议与您的用例最相关。

您可以点击推荐意图的名称以查看聊天机器人设计器建议的示例言语和插槽。如果您选择显示相关转录，则还可以滚动浏览您提供的对话。这些转录会影响聊天机器人设计器对这一意图的推荐。如果点击某个示例言语，则可以查看影响该特定言语的主对话和相关的对话框回合。

您可以点击特定插槽类型的名称来查看推荐的插槽值。如果选择显示相关转录，则可以查看影响此插槽类型的对话，突出显示对该插槽类型引发的代理提示。如果点击某个特定的插槽类型值，则可以查看影响该值的主对话和相关的对话框回合。

要查看和添加意图和插槽类型，请执行以下操作：

1. 登录到 AWS Management Console，并通过以下网址打开 Amazon Lex 控制台：https://console.aws.amazon.com/lex/。
2. 从机器人列表中，选择您要使用的机器人。
3. 选择查看语言。
4. 从语言列表中，选择要使用的语言。
5. 在对话结构中，选择审核。
6. 在意图和插槽类型列表中，选择要添加到机器人的意图和插槽类型。您可以选择意图或插槽类型来查看详细信息和相关转录。

意图按照 Amazon Lex V2 对意图与已处理转录关联的置信度进行排序。

输入转录格式

以下列出了为机器人生成意图和插槽类型的输入文件格式。输入文件必须包含以下字段。其他字段将被忽略。

输入格式与 Contact Lens for Amazon Connect 的输出格式兼容。如果您正在使用 Contact Lens，则无需修改转录文件。有关更多信息，请参阅示例 Contact Lens 输出文件。如果您使用的是其他联络中心应用程序，则必须将您的转录文件转换为此格式。

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

输入文件中必须存在以下字段：

- Participants：标识对话中的参与者以及这些参与者的角色。
- Version：输入文件格式的版本。该字段的值始终为“1.1.0”。

- ContentMetadata：指示您是否从转录中删除了敏感信息。如果转录包含敏感信息，请将该 `Output` 字段设置为“Raw”。
- CustomerMetadata：对话的唯一标识符。
- Transcript：对话中各方之间的对话文本。对话的每一个回合都有一个唯一的标识符。

输出转录格式

输出转录格式与输入转录格式几乎完全相同。但是，该格式中还包括客户元数据和字段，以列出影响意图和插槽类型的建议的片段。您可以从控制台的审核页面或使用 Amazon Lex V2 API 下载输出转录。有关更多信息，请参阅输入转录格式。

```
{
    "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：该 `CustomerMetadata` 字段中添加了两个字段，即包含对话的输入文件的名称以及输入格式（始终为“Lex”）。
- InfluencingSegments：标识影响意图或插槽类型的建议的对话片段。意图或插槽类型的 ID 用于标识受对话影响的特定意图或插槽类型。

添加语言

您可以向机器人添加一种或多种语言和区域设置，以便机器人能够以用户的语言与用户交流。您可以为每种语言分别定义意图、插槽和插槽类型，以便言语、提示和插槽值均是特定于该语言的。

您的机器人必须包含至少一种语言。

要为机器人添加语言，请执行以下操作：

1. 在新语言部分中，选择要使用的语言。您可以添加描述来帮助标识列表中的语言。
2. 如果您的机器人支持语音交互，请在语音交互部分中选择 Amazon Lex V2 用于与用户通信的 Amazon Polly 语音。如果您的机器人不支持语音，请选择无。
3. 对于意图分类置信度分数阈值，设置 Amazon Lex V2 用来确定意图是否为正确意图的值。您可以在测试机器人后再调整该值。
4. 选择 Add（添加）。

添加意图

意图是您的用户想要实现的目标，例如订购鲜花或预订酒店。您的机器人必须有至少一个意图。

默认情况下，所有机器人均包含一个内置意图，即回退意图。当 Amazon Lex V2 无法识别任何其他意图时，就会使用此意图。例如，如果用户对酒店预订意图说“我想订购鲜花”，则会触发回退意图。

要添加意图，请执行以下操作：

1. 登录到 AWS Management Console，并通过以下网址打开 Amazon Lex 控制台：https://console.aws.amazon.com/lex/。
2. 从机器人列表中，选择要添加意图的机器人，然后在添加语言中选择查看语言。
3. 选择要添加意图的语言，然后选择意图。
4. 选择添加意图，指定意图的名称，然后选择添加。
5. 在意图编辑器中，添加意图的详细信息。

    - 对话流程：使用对话流程图查看与您的机器人的对话。您可以选择对话的不同部分以在意图编辑器中跳转到该部分。
    - 意图详情：指定意图的名称和描述，以帮助确定意图的目的。您还可以查看 Amazon Lex V2 指定给该意图的唯一标识符。
    - 上下文：设置意图的输入和输出上下文。上下文是与意图相关联的状态变量。输出上下文在履行意图时设置。只有当上下文处于活动状态时，才能识别带有输入上下文的意图。始终可以识别没有输入上下文的意图。
    - 示例言语：您应提供预期将被用户用于发起某个意图的 10 个或更多个短语。Amazon Lex V2 对这些短语进行归纳，以识别出用户想要发起该意图。

- 初始响应：调用意图后发送给用户的初始消息。您可以提供响应、初始化值，并在意图开始时定义 Amazon Lex V2 为响应用户而采取的下一步操作。
- 插槽：定义履行该意图所需的插槽或参数。每个插槽均设有相应的类型，该类型用于定义可以在该插槽中输入的值。您可以从自定义插槽类型中进行选择，也可以选择内置插槽类型。
- 确认：这些提示和响应用于确认或拒绝意图的履行。该确认提示会提示用户对插槽值进行检查。例如，“我已经预订了周五的酒店。是否确认？” 当用户拒绝确认时，会将拒绝响应发送给用户。您可以提供响应、设置值并定义 Amazon Lex V2 在接收到来自用户的确认或拒绝响应后采取的下一步操作。
- 履行：在履行过程中向用户发送的响应。您可以设置在履行过程开始时的履行进度更新消息，以及在履行过程中的履行进度更新消息。例如，“正在更改您的密码。该过程可能需要几分钟”以及“您的请求仍在处理中”。履行更新消息仅适用于流式传输对话。您还可以设置履行后成功消息、失败消息和超时消息。可以为流式传输和常规对话发送履行后消息。例如，如果履行成功，您可以发送“您的密码已更改”。如果履行失败，您可以发送包含更多信息的回复，例如“您的密码无法更改，请联系帮助中心寻求帮助”。如果履行时间超过所配置的超时时间，您可以向用户发送消息通知，例如“服务器繁忙。请稍后重试。” 您可以提供响应、设置值，并定义 Amazon Lex V2 为响应用户而采取的下一步操作。
- 关闭响应：在意图履行并播放所有其他消息后向用户发送的响应。例如，感谢您预订酒店房间。或者可以提示用户发起不同的意图，例如，“感谢您预订酒店，是否需要预订租车？” 您可以提供响应并配置在履行意图并以关闭响应做出响应后的后续操作。
- 代码挂钩：指明您是否使用 AWS Lambda 函数来初始化意图并验证用户输入。您可以在用于运行机器人的别名中指定 Lambda 函数。

6. 选择保存意图以保存该意图。

> **Note**
>
> Amazon Lex V2 于 2022 年 8 月 17 日发布了对用户对话管理方式的更改。借助此更改，您可以更好地控制用户的对话路径。有关更多信息，请参阅了解对话流管理。2022 年 8 月 17 日之前创建的机器人不支持对话框代码挂钩消息、设置值、配置后续步骤和添加条件。

按特定顺序配置提示

选中按顺序播放消息复选框，即可将机器人配置为按预定义顺序播放消息。否则，机器人会按随机顺序播放消息和变体。

按顺序执行的提示可在多次重试的过程中按顺序播放消息组中的消息和变体。当用户对提示的响应无效时，或者用于确认意图时，您可以使用该消息的不同措辞。每个插槽中最多可以设置两种原始消息的变体。您可以选择是按顺序播放消息还是随机播放消息。

按顺序执行的提示支持所有四种类型的消息，即文本消息、自定义负载响应、SSML 和卡组。响应将按顺序排列在同一个消息组中。不同的消息组是各自独立的。

主题

示例言语

您可以创建示例言语，这些言语是您预期用户用于发起意图的短语的变体。例如，对于 **BookFlight** 意图，您可以包括以下言语：

1. 我想预订航班
2. 帮我预定航班
3. 我需要预定航班机票
4. 从 {*DepartureCity*} 到 {*DestinationCity*} 的航班

您应提供 10 个或更多示例言语。提供相应的示例以表示用户可能说出的各种句子结构和单词。也可以考虑不完整的句子，例如以上的示例 3 和示例 4。您也可以使用您在示例言语中为意图定义的槽位，方法是在槽位名称前后添加花括号，如示例 4 中的 {*DepartureCity*} 所示。如果您在示例言语中包含槽位名称，Amazon Lex V2 会使用用户在言语中提供的值来填充意图的槽位。

各种示例言语可帮助 Amazon Lex V2 进行归纳，从而有效地识别出用户想要发起意图。

您可以在意图编辑器、可视化对话生成器中或通过 CreateIntent 或 UpdateIntent API 操作来添加示例言语。您还可以利用 Amazon Bedrock 的生成式人工智能功能自动生成示例言语。有关更多信息，请参阅言语生成。

使用意图编辑器或可视化对话生成器

1. 在意图编辑器中，导航到示例言语部分。在可视化对话生成器中，在开始块中找到示例言语部分。
2. 在带有透明文本 **I want to book a flight** 的框中，键入示例言语。选择添加话语以添加言语。
3. 查看您在预览或纯文本模式下添加的示例言语。在纯文本中，每行都是单独的言语。在预览模式下，将鼠标悬停在言语上可显示以下选项：
   - 选择文本框以编辑该言语。
   - 选择文本框右侧的 x 按钮可删除该言语。
   - 拖动文本框左侧的按钮可更改示例言语的顺序。
4. 使用顶部的搜索栏搜索您的示例言语，使用旁边的下拉菜单按您添加言语的顺序或者按字母顺序进行排序。

使用 API 操作

1. 使用 CreateIntent 操作创建新意图，或者使用 UpdateIntent 操作更新现有意图。
2. API 请求包含一个 `sampleUtterances` 字段，该字段映射到 SampleUtterance 对象数组。
3. 对于每个要添加的示例言语，请向该数组添加 `SampleUtterance` 对象。将示例言语添加为 `utterance` 字段的值。
4. 要编辑和删除示例言语，请发送 `UpdateIntent` 请求。您在 `sampleUtterances` 字段中提供的言语列表将替换现有的言语。

> **Important**
>
> 您在 `UpdateIntent` 请求中留空的任何字段都将导致意图中的现有配置被删除。使用 DescribeIntent 操作返回机器人配置，并将您不想删除的所有配置复制到 `UpdateIntent` 请求中。

目的结构

以下主题描述了机器人为履行意图而采取的不同步骤以及如何配置这些步骤中的每一个步骤：

主题

- 初始回应

- Slots
- 确认
- 执行
- 关闭响应

初始回应

Amazon Lex V2 在确定意图之后并且在开始引发插槽值之前，会向用户发送初始响应。您可以使用此响应告知用户已识别的意图，并提示用户您需要收集相关信息，以便履行该意图。

例如，如果意图是预约汽车保养服务，则最初响应可能是：

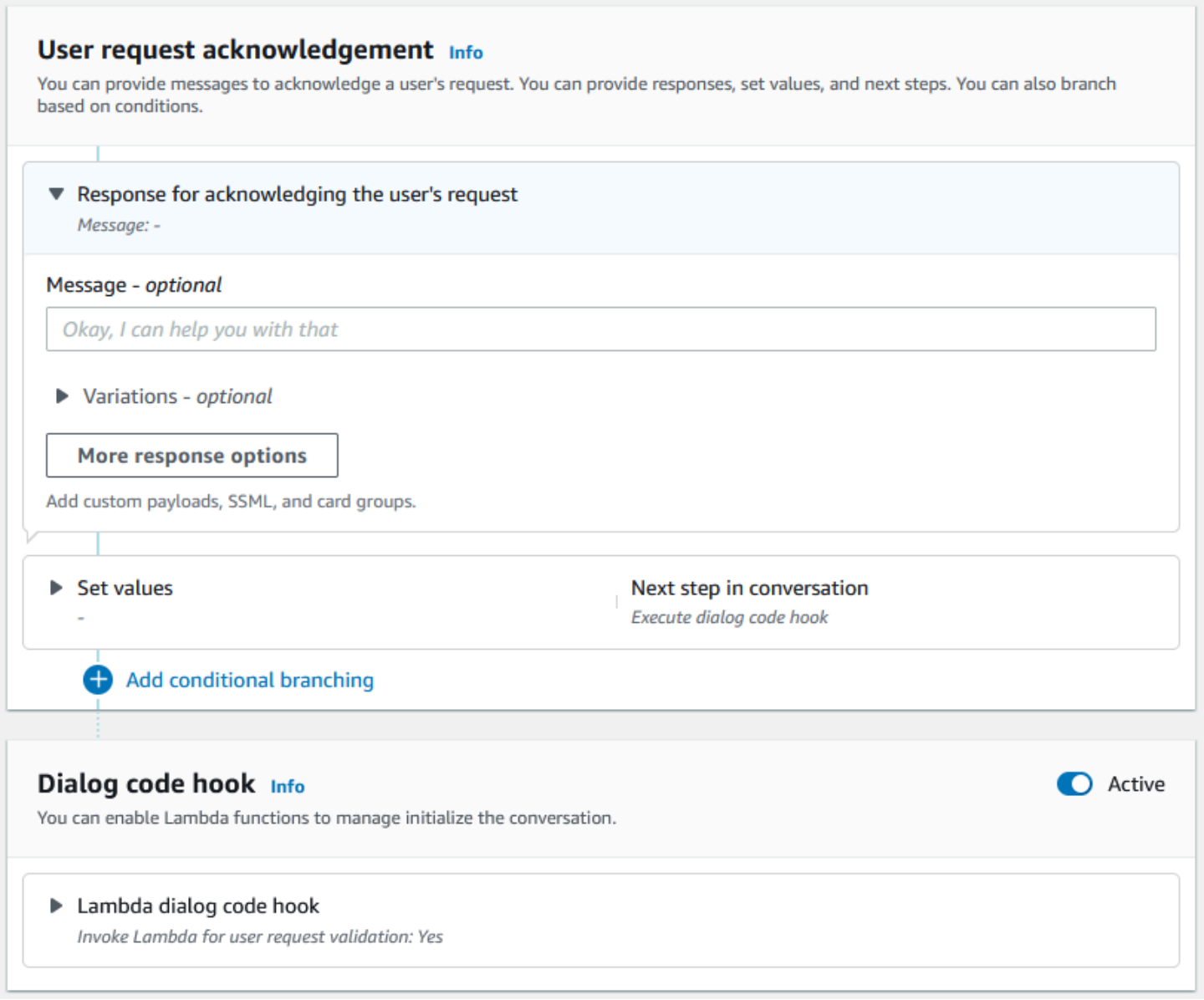
> 我可以帮您安排预约。烦请您提供您爱车的品牌、型号和购车年份。

初始响应消息不是必需的。如果您不提供回复，Amazon Lex V2 将继续按照初始响应的下一步进行操作。

您可以在初始响应中配置以下选项：

- 配置下一步：您可以提供对话的下一步，例如跳转到特定的对话框操作、引发特定的插槽或跳转到不同的意图。有关更多信息，请参阅配置对话中的后续步骤。
- 设定值：您可以设置插槽和会话属性的值。有关更多信息，请参阅在对话期间设定值。
- 添加条件分支：您可以在播放初始响应后应用条件。当条件计算为 true 时，将执行您定义的操作。有关更多信息，请参阅添加条件以构建对话的分支。
- 执行对话框代码挂钩：您可以定义 Lambda 代码挂钩来初始化数据并执行业务逻辑。有关更多信息，请参阅调用对话框代码挂钩。如果为意图启用了执行 Lambda 函数的选项，则默认情况下会执行对话框代码挂钩。您可以通过切换活动按钮来禁用对话框代码挂钩。

如果未设置条件或明确的下一步操作，Amazon Lex V2 将按优先级继续执行下一个插槽。



> **Note**
>
> Amazon Lex V2 于 2022 年 8 月 17 日发布了对用户对话管理方式的更改。借助此更改，您可以更好地控制用户的对话路径。有关更多信息，请参阅了解对话流管理。2022 年 8 月 17 日之前创建的机器人不支持对话框代码挂钩消息、设置值、配置后续步骤和添加条件。

Slots

插槽是用户为履行意图而提供的值。有两种类型的插槽：

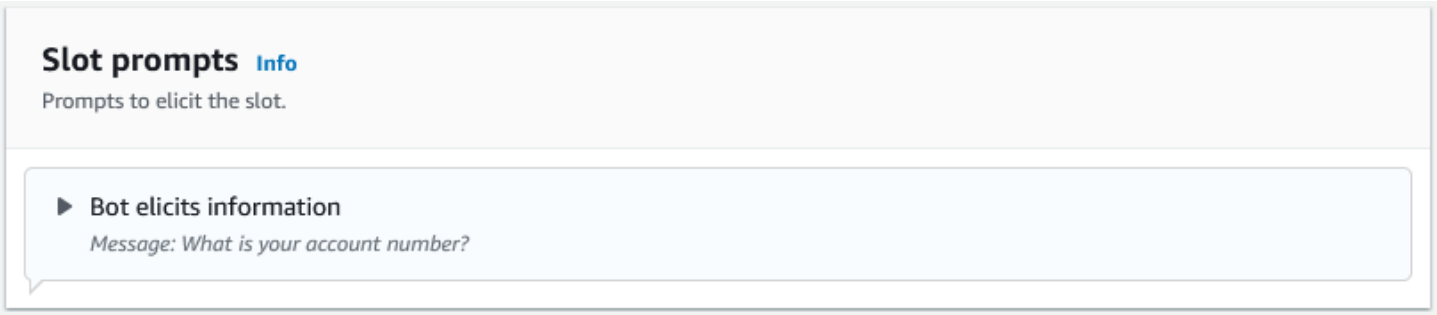
- 内置插槽类型：您可以使用内置的插槽类型来捕获标准值，例如数字、名称和城市。有关系统支持的内置插槽类型的列表，请参阅内置槽位类型。
- 自定义插槽类型：您可以使用自定义插槽类型来捕获特定于意图的自定义值。例如，您可以使用自定义插槽类型来捕获账户类型，例如“信用账户”或“储蓄账户”。有关更多信息，请参阅自定义槽位类型。

要在意图中定义插槽，您必须配置以下内容：

- 插槽信息：该字段包含插槽的名称和描述（可选）。例如，您可以将插槽名指定为“AccountNumber”以获取账号。如果插槽作为履行意图的对话流程的一部分是必需的，则必须将其标记为必填项。
- 插槽类型：定义可以接受的插槽值列表的插槽类型。您可以创建自定义插槽类型或使用预定义的插槽类型。
- 插槽提示：向用户提出的收集信息的问题。您可以配置用于收集信息的重试次数以及每次重试时使用的提示的变体。您还可以在每次重试后启用 Lambda 函数调用，以处理所捕获的输入并尝试解析为有效的输入。
- 等待并继续（可选）：通过启用此行为，用户可以说出“稍等片刻”等短语，机器人即等待用户查找并提供信息。仅对流式传输对话启用此功能。有关更多信息，请参阅允许机器人等待用户提供更多信息。
- 插槽捕获响应：您可以根据从用户输入中捕获插槽值的结果来配置成功响应和失败响应。
- 条件分支：您可以在播放初始响应后应用条件。当条件计算为 true 时，将执行您定义的操作。有关更多信息，请参阅添加条件以构建对话的分支。
- 对话框代码挂钩：您还可以使用 Lambda 代码挂钩来验证插槽值并执行业务逻辑。有关更多信息，请参阅调用对话框代码挂钩。
- 用户输入类型：您可以配置输入类型，以便机器人可以接受特定的模式。默认情况下，音频和 DTMF 模式均是支持的。您可以选择将其设置为仅音频或仅限 DTMF。
- 音频输入超时和长度：您可以配置音频超时，包括语音超时和静默超时。此外，您还可以设置最大音频长度。
- DTMF 输入超时、字符和长度：您可以设置 DTMF 超时以及删除字符和结尾字符。此外，您还可以设置最大 DTMF 长度。
- 文本长度：您可以设置文本模式的最大长度。

播放插槽提示后，用户提供插槽值作为输入。如果 Amazon Lex V2 无法理解用户提供的插槽值，则将重新尝试引发该插槽，直到理解某个值或超过您为该插槽配置的最大重试次数。使用高级重试设

置，您可以配置超时、限制输入类型，以及启用或禁用初始提示和重试的中断。每次尝试捕获输入后，Amazon Lex V2 都可以利用为重试提供的调用标签来调用为机器人配置的 Lambda 函数。例如，您可以使用 Lambda 函数来应用您的业务逻辑来尝试将其解析为有效值。可以在插槽提示的高级选项中启用此 Lambda 函数。



您可以定义在输入插槽值或超过最大重试次数后，机器人应向用户发送的响应。例如，对于用于安排汽车服务的机器人，您可以在输入车辆识别码 (VIN) 时向用户发送消息：

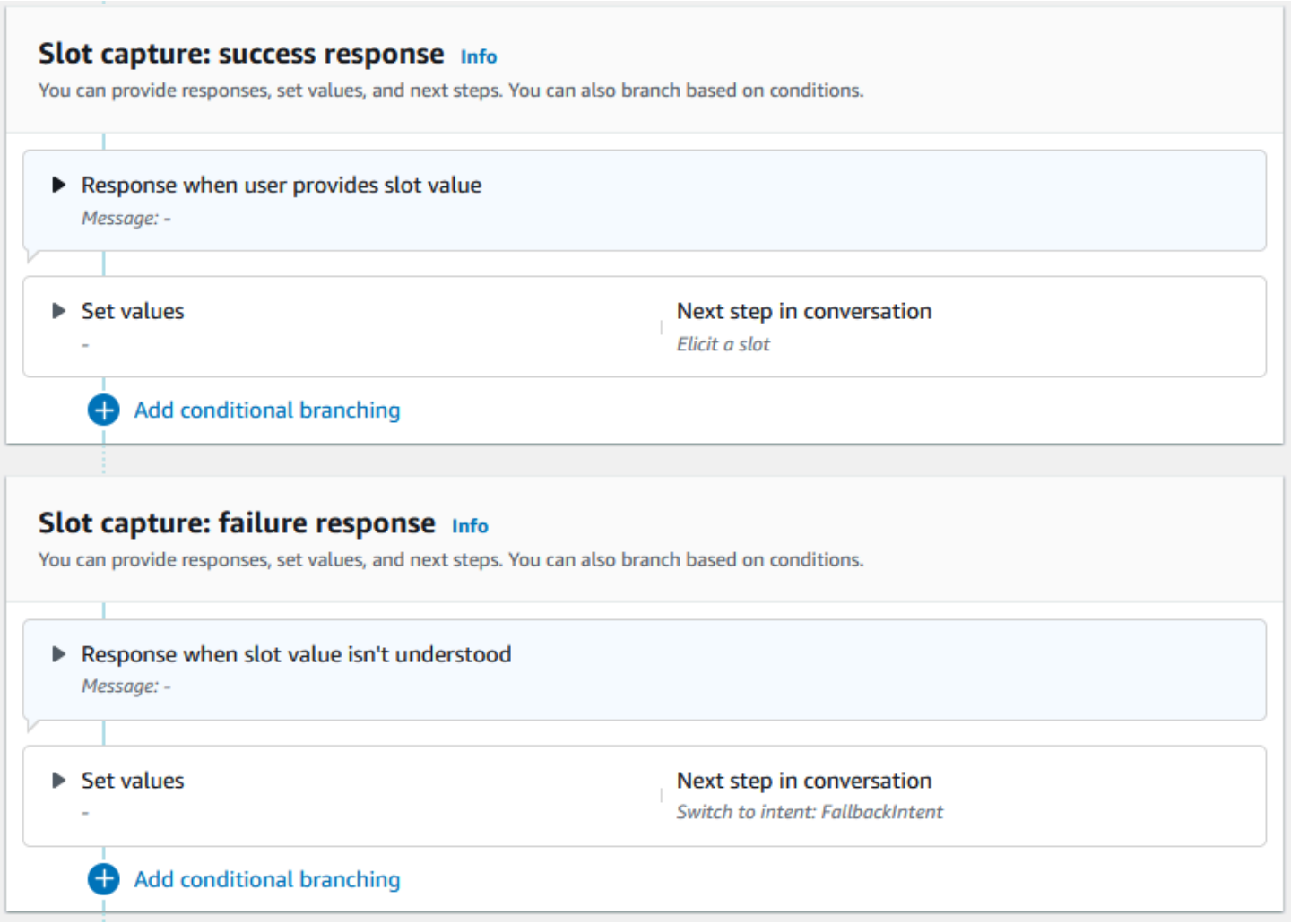
感谢您提供车辆的 VIN 号码。现在将开始安排预约。

您可以创建两个响应：

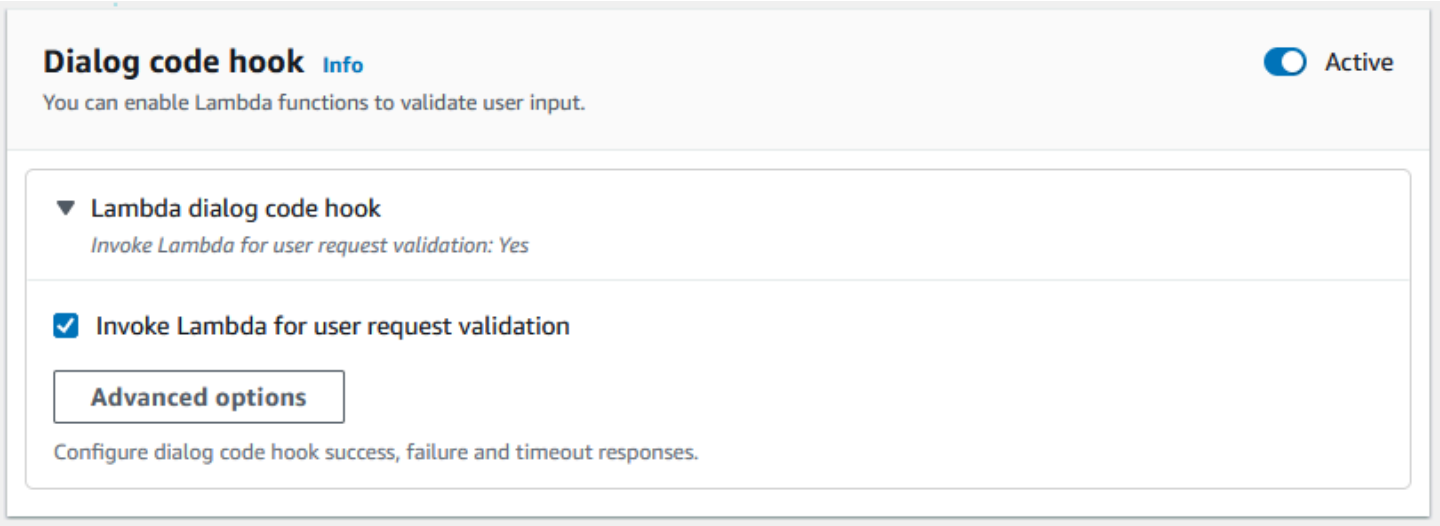
- 成功响应：当 Amazon Lex V2 了解插槽值时发送。
- 失败响应：当 Amazon Lex V2 在达到最大重试次数后仍无法理解用户的插槽值时发送。

您可以设置值、配置后续步骤，并应用与每个响应相对应的条件以设计对话流程。

如果未设置条件或明确的下一步操作，Amazon Lex V2 将按优先级继续执行下一个插槽。



您可以使用 Lambda 函数来验证用户输入的插槽值并确定下一步应执行的操作。例如，您可以使用验证功能来确保输入的值位于正确的范围内，或者格式正确。要激活 Lambda 函数，请在对话框代码挂钩部分中选择调用 Lambda 函数复选框和活动按钮。您可以为对话框代码挂钩指定调用标签。此调用标签可以在 Lambda 函数中用于编写与插槽引发相对应的业务逻辑。



该意图不需要的插槽不是主对话流程的一部分。但是，如果用户言语包含您的机器人识别为对应于可选插槽的值，则可以用该值填充该插槽。例如，如果您将业务智能机器人配置为具有可选 `City` 插槽和用户言语 **`What is the sales for April in San Diego?`**，则该机器人将该可选插槽填充为 **`San Diego`**。您可以将业务逻辑配置为使用可选的插槽值（如果存在）。

使用后续步骤无法引发该意图所不需要的插槽。这些步骤只能在意图引发期间填充（如上一个示例所示），也可以通过在 Lambda 函数中设置对话状态来被引发。如果该插槽是使用 Lambda 函数引发的，则必须使用 Lambda 函数在插槽引发完成后决定对话的下一步操作。要在构建机器人时启用对下一步的支持，您必须将该插槽标记为意图所必需的。

> **Note**
>
> Amazon Lex V2 于 2022 年 8 月 17 日发布了对用户对话管理方式的更改。借助此更改，您可以更好地控制用户的对话路径。有关更多信息，请参阅了解对话流管理。2022 年 8 月 17 日之前创建的机器人不支持对话框代码挂钩消息、设置值、配置后续步骤和添加条件。

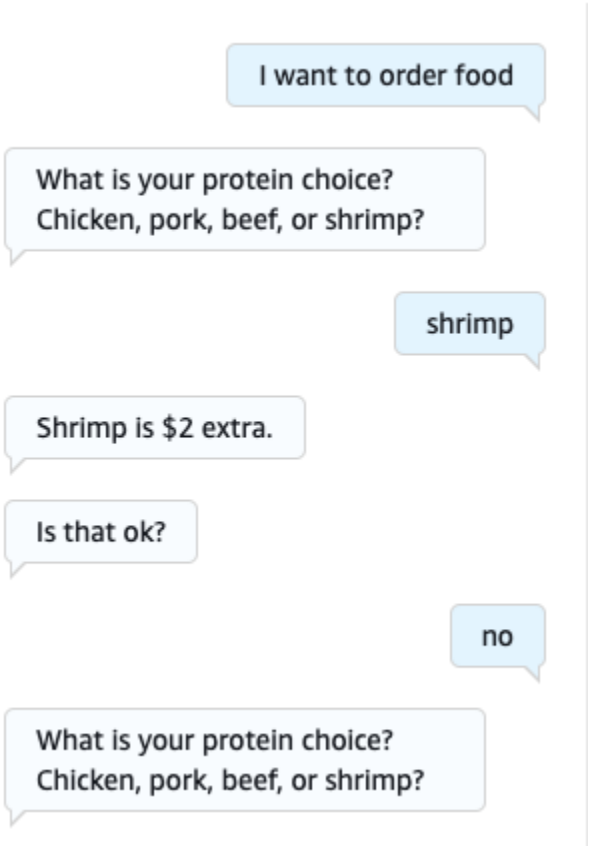
以下主题介绍如何配置机器人以重新引发已填充的插槽值，以及如何创建由多个值组成的插槽：

主题

- 重新引发机器人
- 使用一个插槽中的多个值

重新引发机器人

您可以将机器人配置以重新引发一个已填充的插槽，方法是将该插槽值设置为 **null**，然后将对话的下一步设置为循环返回该插槽。例如，在客户拒绝确认基于额外信息的插槽引发后，您可能需要重新引发该插槽，如以下对话所示：



您可以使用意图编辑器或使用可视化对话生成器配置从确认响应返回重新引发该插槽的循环。

> **Note**
>
> 只要您事先将某个插槽值设置为 **null**，即可在对话中的任何时候循环回到重新引发该插槽。

使用意图编辑器重现以上示例

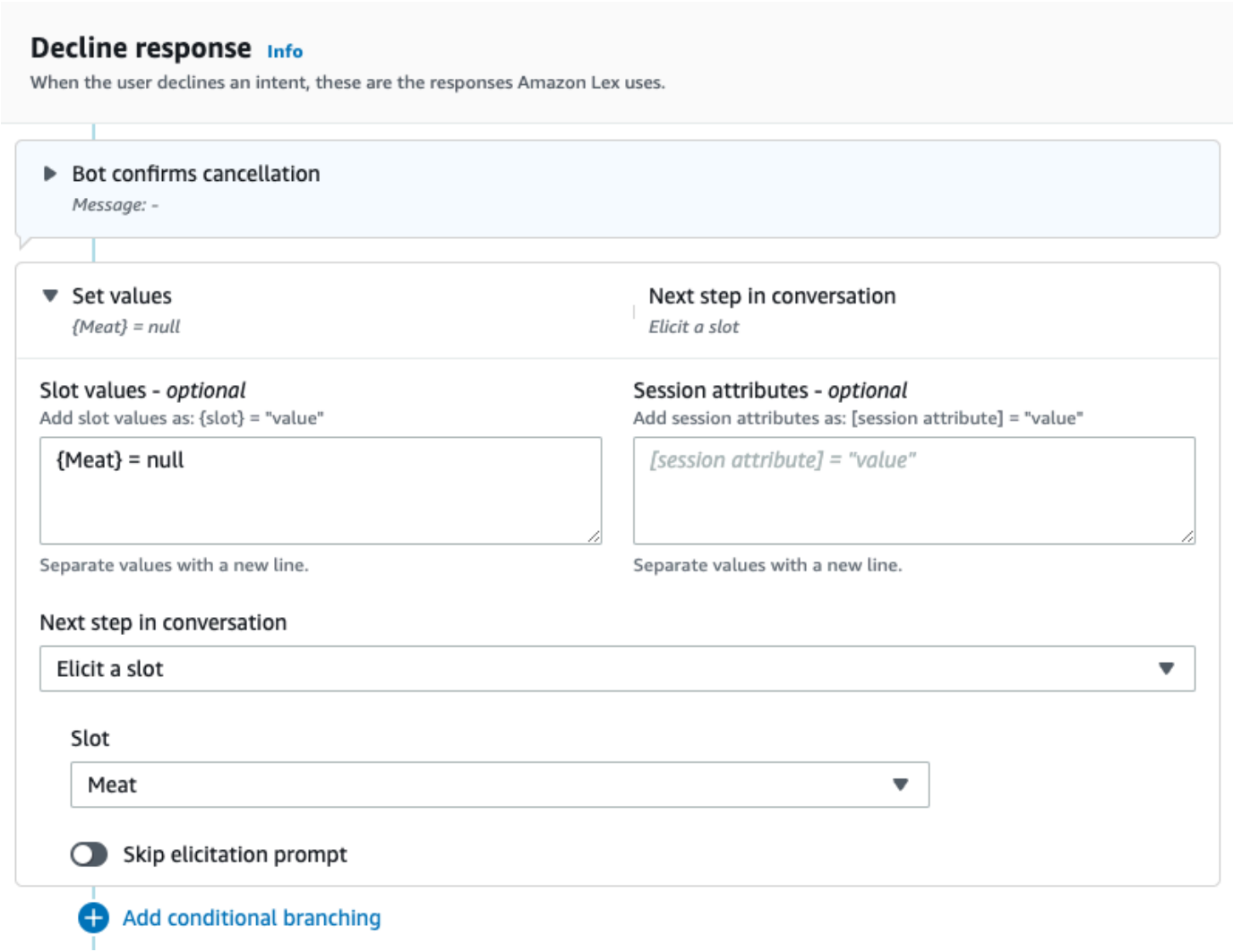
1. 在意图编辑器的确认部分，选择确认意图提示旁边的右箭头以展开该部分。
2. 选择底部的高级选项。
3. 在拒绝响应部分中，选择设置值旁边的右箭头以展开该部分。按照以下步骤填写此部分，如下图所示：

a. 将您要重新引发的插槽值设置为 **null**。在本示例中，需要重新引发插槽 Meat，因此在插槽值部分中输入 **{Meat} = null**。

b. 在对话的下一步下的下拉菜单中，选择引发插槽

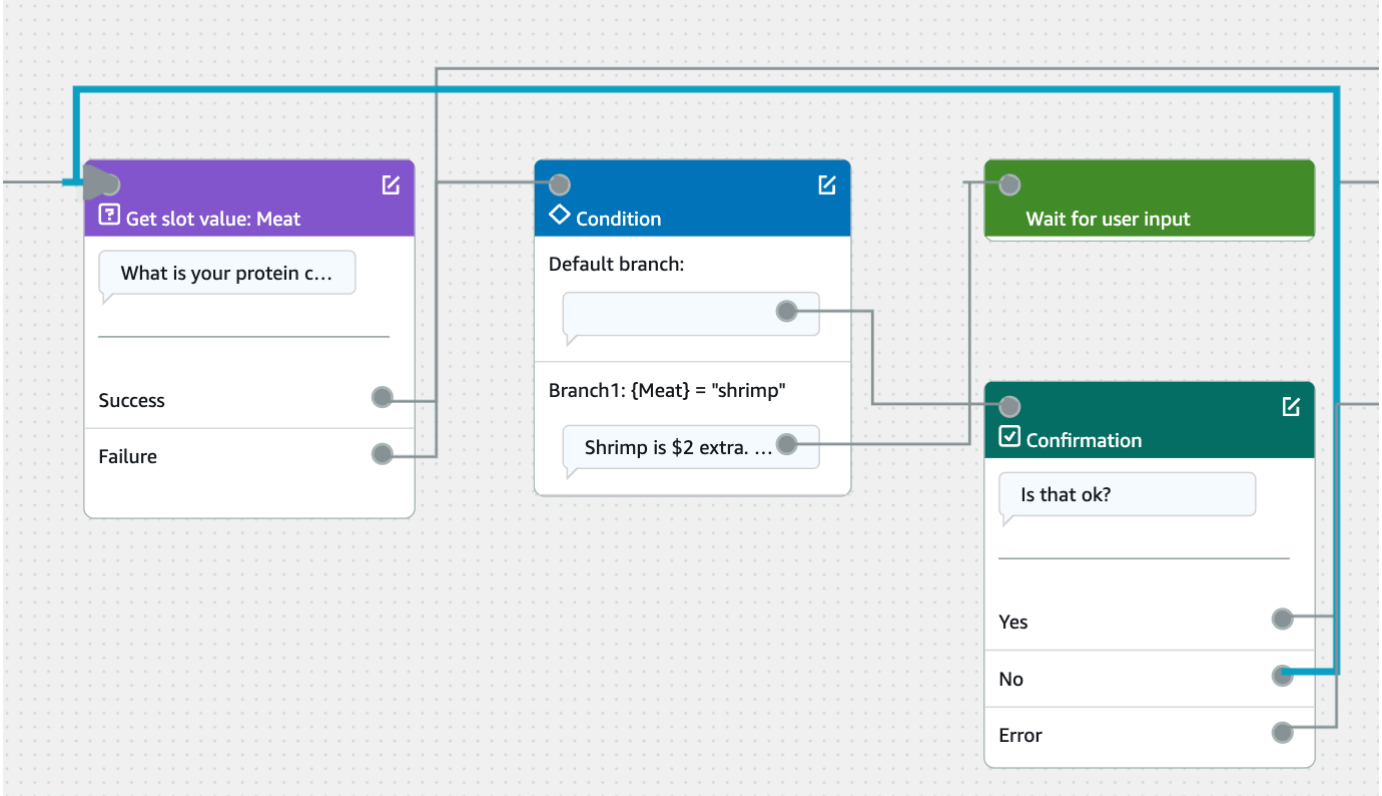
c. 将出现插槽部分。在其下方的下拉菜单中，选择您要重新引发的插槽。

d. 选择更新选项以确认您的更改。

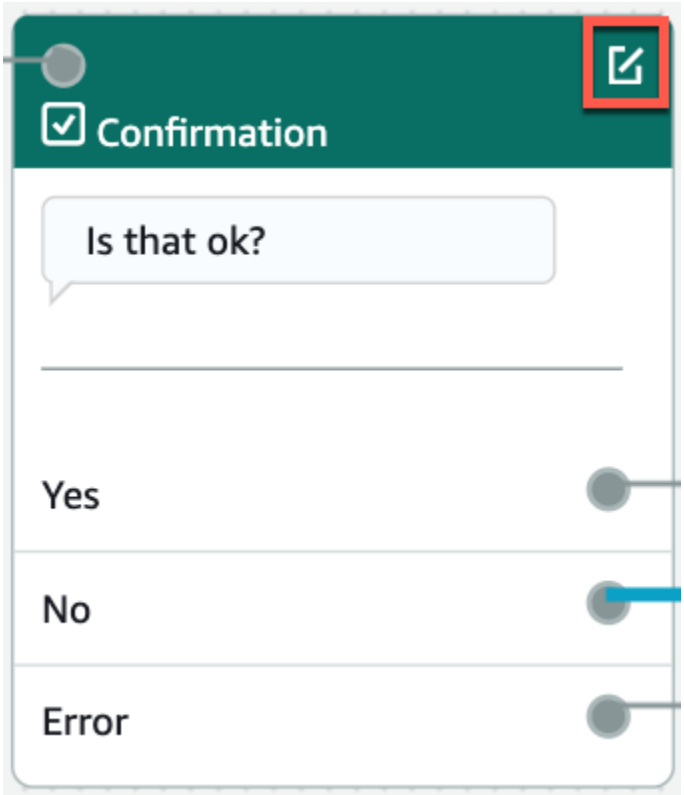


使用可视化对话生成器重现上述示例

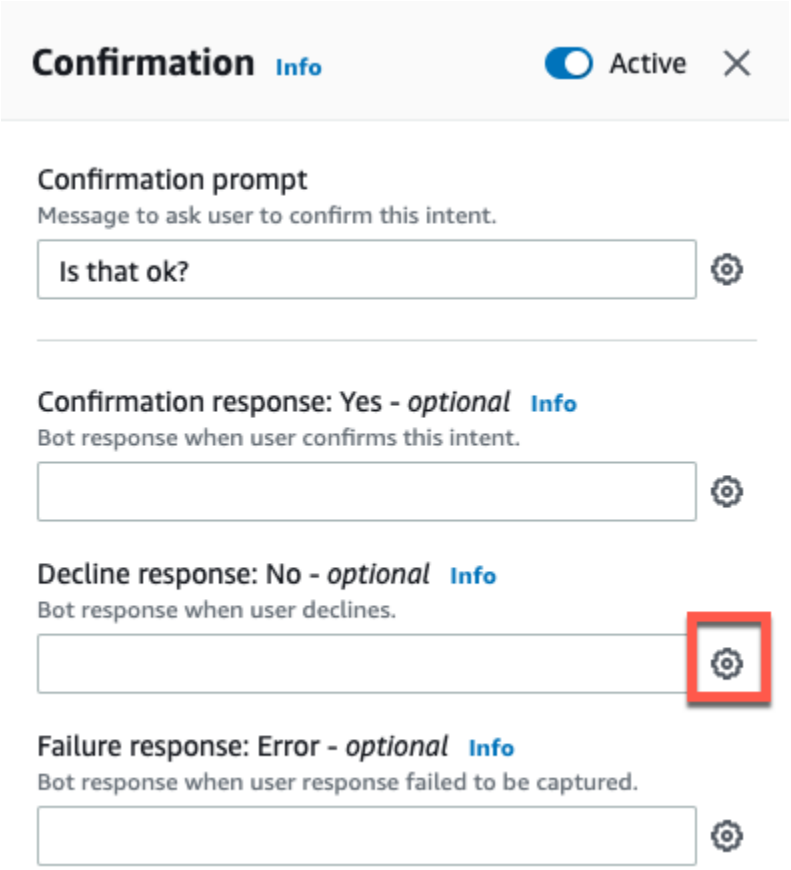
1. 创建从确认块的否端口到获取插槽值：Meat 块的传入端口的连接。



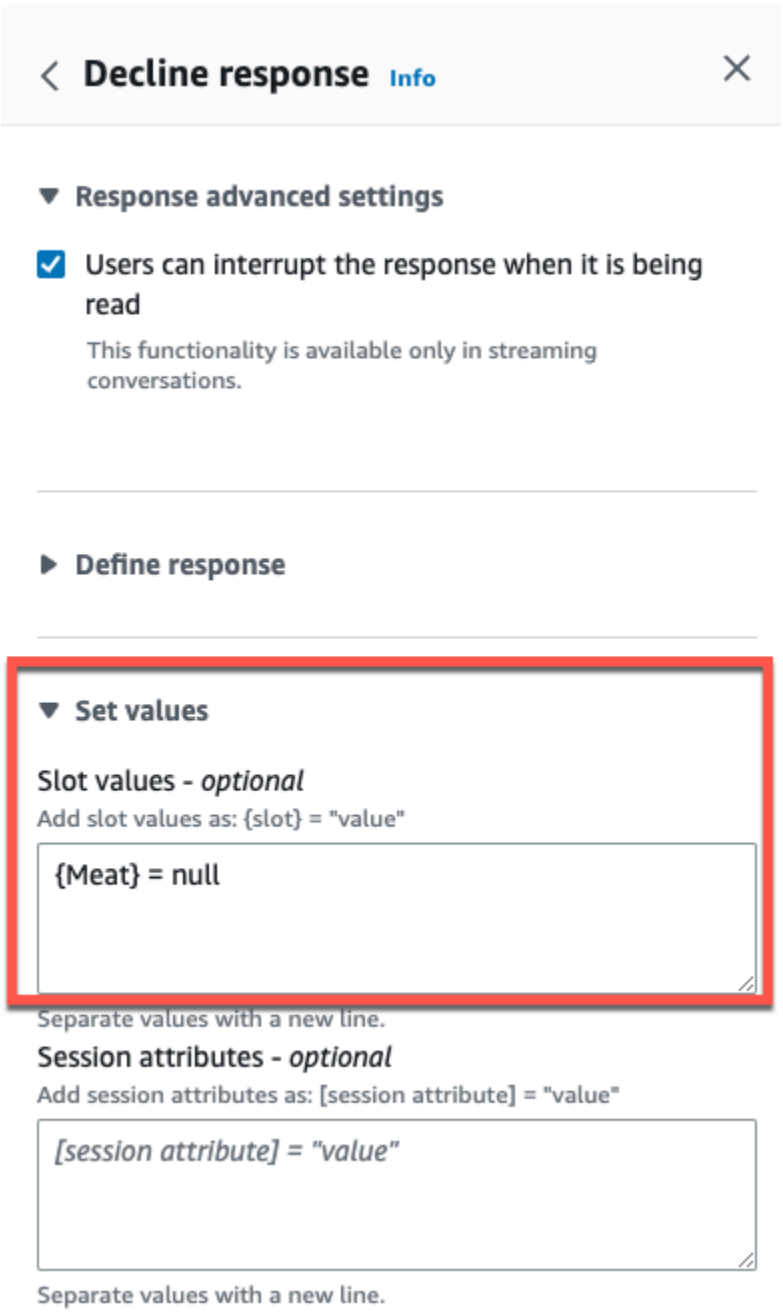
2. 选择确认块右上角的编辑图标。



3. 在拒绝响应部分中，选择机器人响应旁边的齿轮图标。



4. 在设置值部分，在插槽值框中添加“{Meat} = null”。



5. 选择保存意图。

使用一个插槽中的多个值

> **Note**
>
> 多值插槽仅适用于英语（美国）。

对于某些意图，您可能需要为单个插槽捕获多个值。例如，披萨订购机器人可能有以下言语的意图：

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

该意图预期该 {toppings} 插槽包含客户想要加在所订购披萨上的一系列配料，例如“pepperoni and pineapple”（意大利辣香肠和菠萝）。

要将插槽配置为捕获多个值，请将插槽上的 allowMultipleValues 字段设置为 true。您可以使用控制台或通过 CreateSlot 或 UpdateSlot 操作来设置该字段。

您只能将具有自定义插槽类型的插槽标记为多值插槽。

对于多值插槽，Amazon Lex V2 在响应 RecognizeText 或 RecognizeUtterance 操作时会返回插槽值列表。以下是 OrderPizza 机器人为言语“I want a pizza with pepperoni and pineapple”（我想要一个意大利辣香肠和菠萝的披萨）返回的插槽信息。

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

多值插槽始终返回一系列的值。当言语只包含一个值时，返回的值列表仅包含一个响应。

Amazon Lex V2 可识别由空格、逗号 (,) 和连词 “and” 分隔的多个值。多值插槽支持使用文本和语音输入。

您可以在提示中使用多个值的插槽。例如，您可以将意图的确认提示设置为

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

当 Amazon Lex V2 将向用户发送提示时，将发送“Would you like me to order your pepperoni and pineapple pizza?”（是否需要为您预定意大利辣香肠和菠萝披萨？）

多值插槽支持单个默认值。如果提供了多个默认值，Amazon Lex V2 将仅使用第一个可用值来填充插槽。有关更多信息，请参阅使用默认槽值。

您可以使用插槽模糊处理来掩盖对话日志中多值插槽的值。当您对插槽值进行模糊处理时，每个插槽值将被替换为插槽的名称。有关更多信息，请参阅掩盖对话日志中的槽位值。

确认

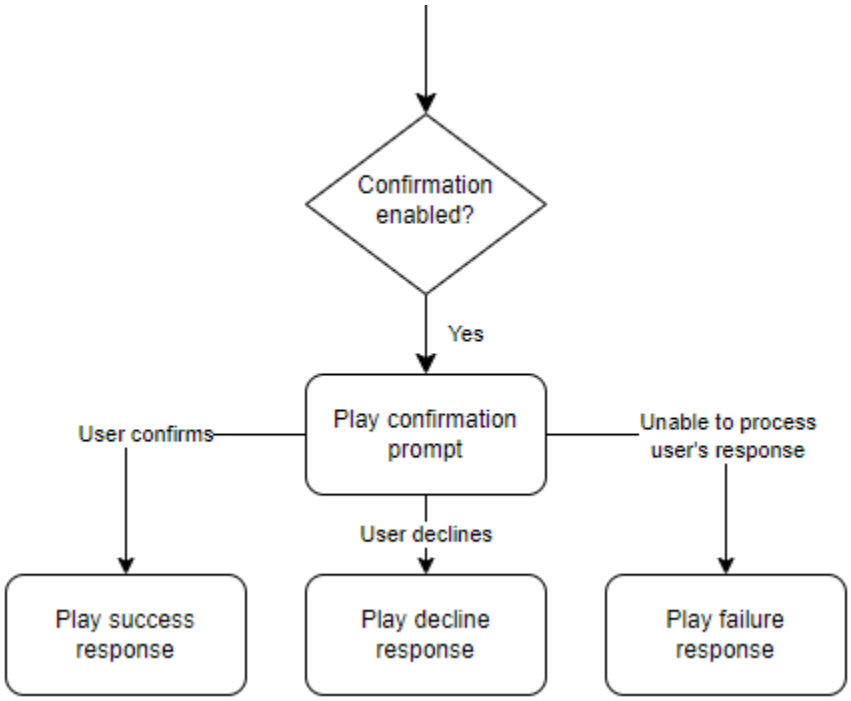
在与用户的对话完成并填充意图的插槽值后，您可以配置确认提示以询问用户插槽值是否正确。例如，安排车辆保养预约的机器人可能会提示用户以下内容：

```
已帮您为 2017 年款本田思域车辆预约 3 月 25 日下午 3:00 的保养服务。请确认是否正确？
```

您可以定义 3 种类型的确认提示响应：

- 确认响应：当用户确认意图时，将向用户发送此响应。例如，在用户对提示“是否确认下单？”回答“是”之后。

- 拒绝响应：当用户拒绝该意图时，将向用户发送此响应。例如，在用户对提示“是否确认下单？”回答“否”之后。
- 失败响应：当无法处理确认提示时，将向用户发送此响应。例如，如果用户的回复无法理解或无法解析为“是”或“否”。

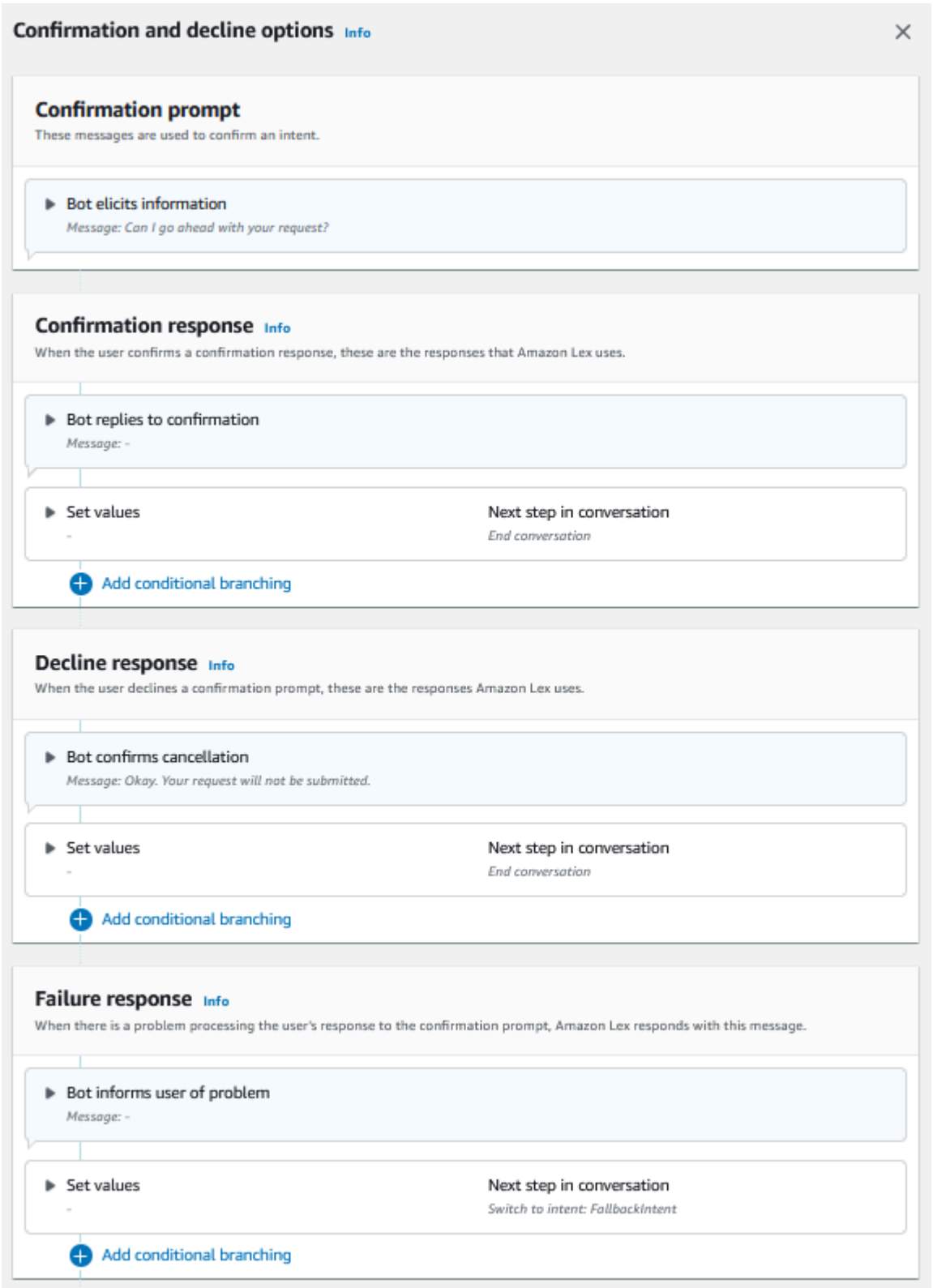


如果您未指定确认提示，Amazon Lex V2 将进入履行步骤或结束响应。

您可以设置值、配置后续步骤，并应用与每个响应相对应的条件以设计对话流程。如果未设置条件或明确的下一步操作，Amazon Lex V2 将继续执行履行步骤。

您也可以启用对话框代码挂钩，以便在发送要履行的意图之前验证意图中捕获的信息。要使用代码挂钩，请在确认提示高级选项中启用对话框代码挂钩。此外，配置上一个状态的下一步以执行对话框代码挂钩。有关更多信息，请参阅调用对话框代码挂钩。

> **Note**
>
> 如果您使用代码挂钩在运行时触发确认步骤，则必须在构建时将确认步骤标记为活动。


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**
>
> Amazon Lex V2 于 2022 年 8 月 17 日发布了对用户对话管理方式的更改。借助此更改，您可以更好地控制用户的对话路径。有关更多信息，请参阅了解对话流管理。2022 年 8 月 17 日之前创建的机器人不支持对话框代码挂钩消息、设置值、配置后续步骤和添加条件。

使用 Lambda 函数来验证意图。

您可以定义 Lambda 代码挂钩以验证意图，然后再发送该意图以进行履行。要使用代码挂钩，请在确认提示高级选项中启用对话框代码挂钩。

使用代码挂钩时，您可以定义 Amazon Lex V2 在代码挂钩运行后执行的操作。您可以创建三种类型的响应：

- 成功响应：当代码挂钩成功完成时发送给用户。
- 失败响应：当代码挂钩未成功运行或代码挂钩在响应中返回 `Failure` 时，发送给用户。
- 超时响应：当代码挂钩未在其配置的超时时间内完成时发送给用户。

执行

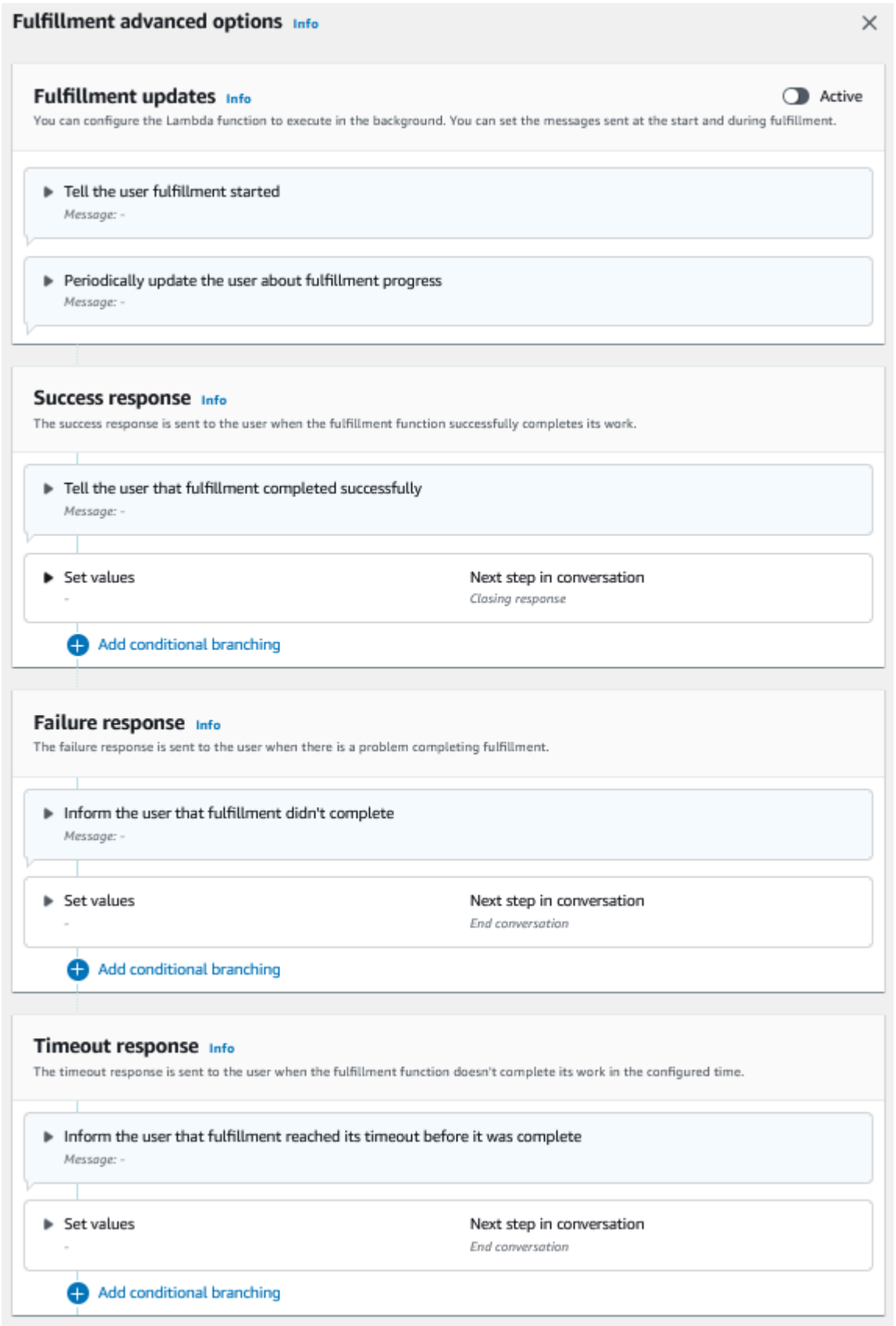
在用户为意图提供所有插槽值后，Amazon Lex V2 会履行用户的请求。您可以配置以下履行选项：

- 履行代码挂钩：您可以使用此选项来控制履行 Lambda 调用。如果禁用该选项，将在不调用 Lambda 函数的情况下成功完成该履行。
- 履行更新：您可以为需要超过几秒钟才能完成的 Lambda 函数启用履行更新，以便用户知道流程正在进行中。有关更多信息，请参阅配置履行进度更新。此功能仅适用于流式传输对话。
- 履行响应：您可以配置成功响应、失败响应和超时响应。系统会根据履行 Lambda 调用的状态向用户返回相应的响应。

可能的履行响应有以下三种：

- 成功响应：履行 Lambda 成功完成时发送的消息。
- 失败响应：如果履行失败或 Lambda 由于某种原因无法完成，则发送的消息。
- 超时响应：如果履行 Lambda 函数未在配置的超时时间内完成，则发送的消息。

您可以设置值、配置后续步骤，并应用与每个响应相对应的条件以设计对话流程。如果未设置条件或明确的下一步操作，Amazon Lex V2 将执行结束响应。


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**
>
> Amazon Lex V2 于 2022 年 8 月 17 日发布了对用户对话管理方式的更改。借助此更改，您可以更好地控制用户的对话路径。有关更多信息，请参阅了解对话流管理。2022 年 8 月 17 日之前创建的机器人不支持对话框代码挂钩消息、设置值、配置后续步骤和添加条件。

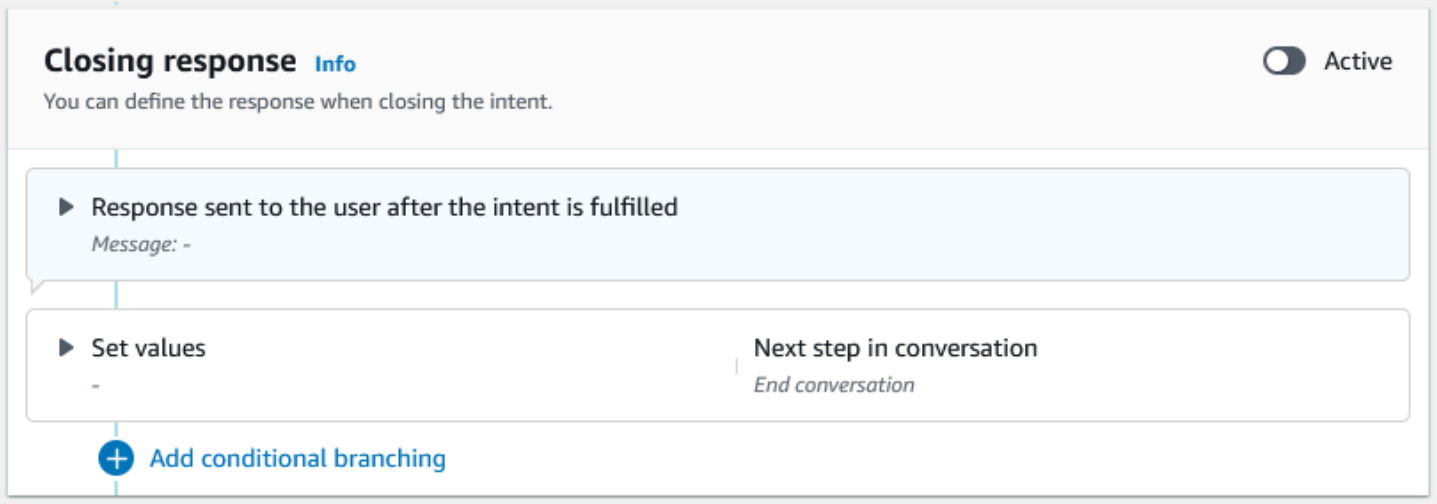
关闭响应

该关闭响应将在您用户的意图履行后发送给用户。您可以使用关闭响应来结束对话，也可以使用关闭响应来提示用户继续发起其他意图。例如，在旅行预订机器人中，您可以将预订酒店房间的关闭响应设置为：

```
现已成功预订酒店房间。请问您是否需要其他帮助？
```

您可以设置值、配置后续步骤，并在关闭响应之后应用条件以设计对话流程。如果未设置条件或明确的下一步操作，Amazon Lex V2 将结束对话。

如果您没有提供关闭响应，或者如果没有一个条件计算为 true，Amazon Lex V2 将结束与您的机器人的对话。

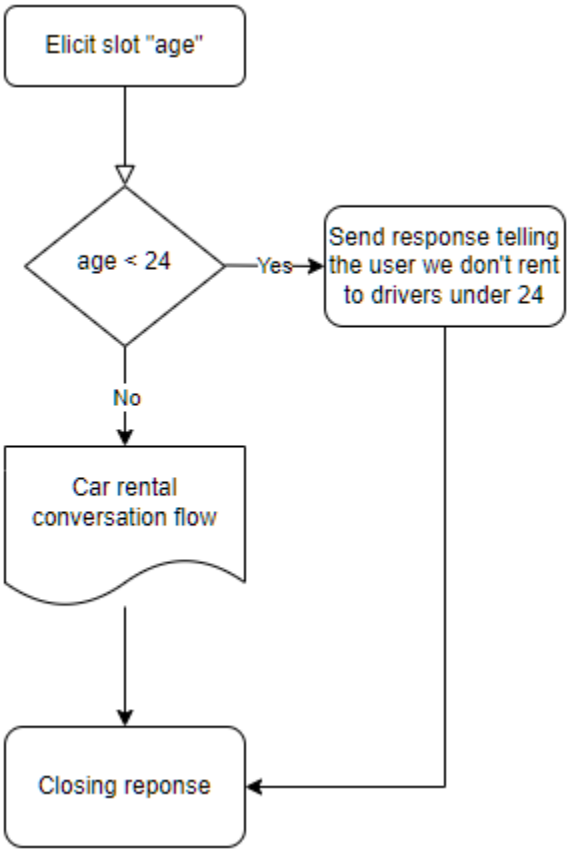


> **Note**
>
> Amazon Lex V2 于 2022 年 8 月 17 日发布了对用户对话管理方式的更改。借助此更改，您可以更好地控制用户的对话路径。有关更多信息，请参阅了解对话流管理。2022 年 8 月 17 日之前创建的机器人不支持对话框代码挂钩消息、设置值、配置后续步骤和添加条件。

创建对话路径

通常，Amazon Lex V2 会管理与您用户的对话流程。对于简单的机器人，利用默认流程即足以产生良好的用户体验。但是，对于更复杂的机器人，您可能需要控制对话并在对话流程中实现更复杂对话路径的引导。

例如，在预订租车的机器人中，您可能需要不将车辆租给年轻驾驶人。在这种情况下，您可以创建一个条件来检查驾驶人是否低于指定年龄，如果是，则跳转到结束响应。



要设计这样的交互，您可以配置对话中每个时刻的下一步，评估条件，设置值并调用代码挂钩。

条件分支可帮助您创建用户进行复杂交互的对话路径。您可以在任何时间节点使用条件分支，将对话的控制权移交给机器人。例如，您可以在机器人引发第一个槽位值之前创建一个条件，您可以在引发每个

槽位值之间创建一个条件，或者您可以在机器人关闭对话之前创建一个条件。有关可以添加条件的节点列表，请参阅添加意图。

当您创建机器人时，Amazon Lex V2 会根据各个槽位的优先级来创建默认的对话路径。要自定义对话路径，可以修改对话中任何节点的下一步。有关更多信息，请参阅 配置对话中的后续步骤。

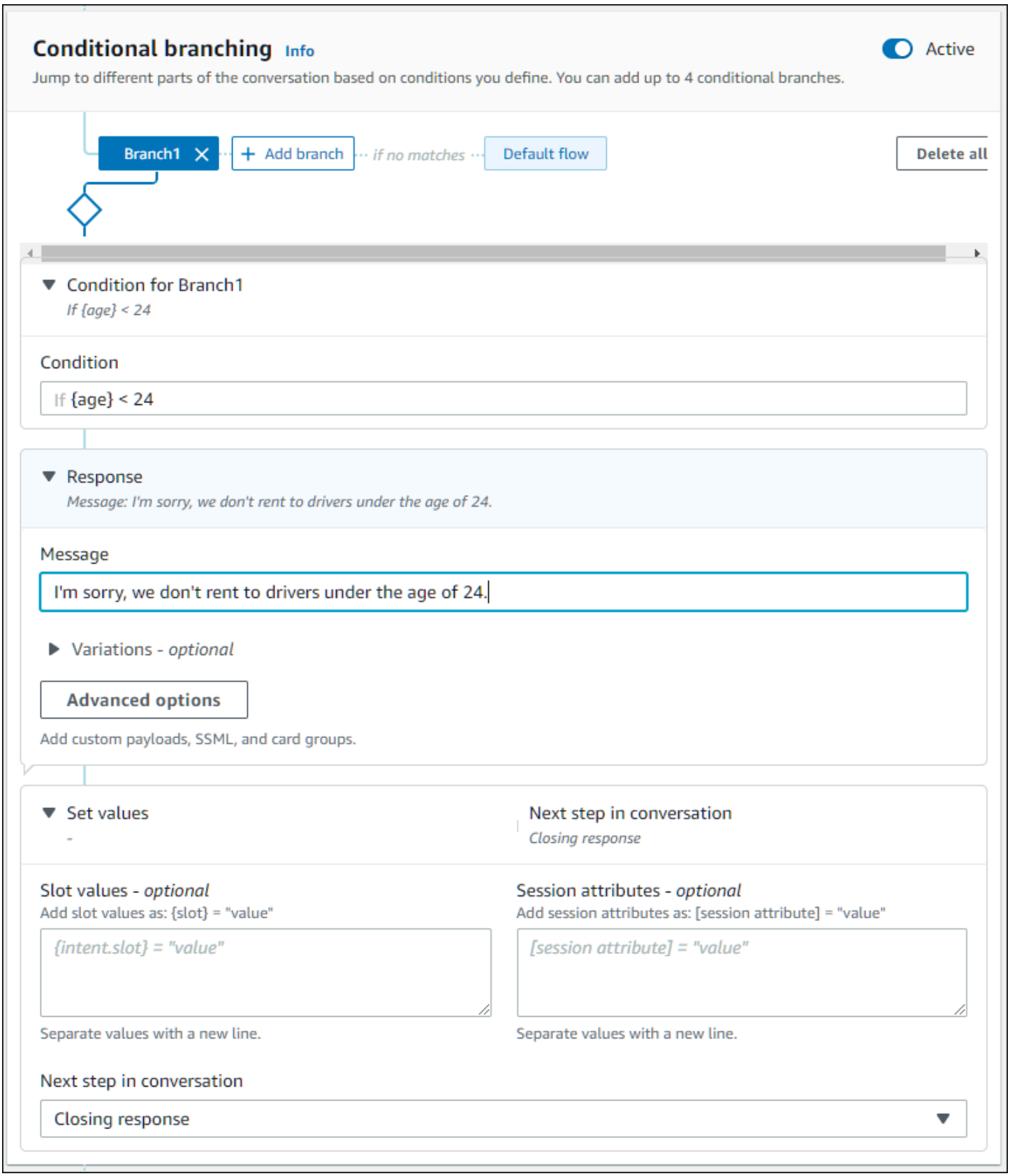
要创建基于条件的替代路径，可以在对话中的任何节点使用条件分支。例如，您可以在机器人引发第一个槽位值之前创建一个条件。您可以在引发每个槽位值之间创建一个条件，也可以在机器人关闭对话之前创建一个条件。有关允许您添加条件的节点列表，请参阅添加条件以构建对话的分支。

您可以根据槽位值、会话属性、输入模式和输入转录或来自 Amazon Kendra 的响应来设置条件。

您可以在对话中的每个节点处设置槽位和会话属性值。有关更多信息，请参阅 在对话期间设定值。

您也可以将下一个操作设置为对话框代码挂钩以运行 Lambda 函数。有关更多信息，请参阅 调用对话框代码挂钩。

下图是在控制台中为某个槽位创建路径的过程。在此示例中，Amazon Lex V2 将引发槽位“age”。如果槽位的值小于 24，Amazon Lex V2 会跳转到结束响应，否则 Amazon Lex 将遵循默认路径。


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**
>
> Amazon Lex V2 于 2022 年 8 月 17 日发布了对用户对话管理方式的更改。借助此更改，您可以更好地控制用户的对话路径。有关更多信息，请参阅 了解对话流管理。2022 年 8 月 17 日之前创建的机器人不支持对话框代码挂钩消息、设置值、配置后续步骤和添加条件。

配置对话中的后续步骤

您可以配置对话中每个阶段的下一步以设计对话。通常，Amazon Lex V2 会按照以下顺序自动为对话的每个阶段配置默认的后续步骤。

初始回应 → 槽位引发 → 确认（如果处于活动状态）→ 履行（如果处于活动状态）→ 结束响应（如果处于活动状态）→ 结束对话

您可以修改默认的后续步骤，并根据预期的用户体验来设计对话。可以在对话的每个阶段配置以下后续步骤：

跳转到

- 初始响应：从意图开始时重新开始对话。在配置下一步时，您可以选择跳过初始响应。
- 引发槽位：您可以引发意图中的任何槽位。
- 评估条件：您可以评估对话中任何步骤处的条件并且构建对话的分支。
- 调用对话框代码挂钩：您可以在任何步骤调用业务逻辑。
- 确认意图：将提示用户确认意图。
- 履行意图：将在下一步开始履行意图。
- 结束响应：该结束响应将返回给用户。

切换到

- 意图：您可以过渡到不同的意图并继续就此意图进行对话。您可以选择在执行过渡的同时跳过该意图的初始响应。
- 意图：特定槽位：如果您已经在当前意图中捕获了一些槽位值，则可以直接在不同的意图中引发特定的槽位。

等待用户输入：机器人等待用户提供输入以识别任何新意图。你可以配置相关提示，例如“还有什么我可以帮你的吗？”，然后再设置下一步。机器人将处于 `ElicitIntent` 对话框状态。

结束对话：结束与机器人的对话。

> **Note**
>
> Amazon Lex V2 于 2022 年 8 月 17 日发布了对用户对话管理方式的更改。借助此更改，您可以更好地控制用户的对话路径。有关更多信息，请参阅 了解对话流管理。2022 年 8 月 17 日之前创建的机器人不支持对话框代码挂钩消息、设置值、配置后续步骤和添加条件。

在对话期间设定值

Amazon Lex V2 提供了在对话的每个步骤中设置槽位值和会话属性值的功能。然后，您可以在对话期间使用这些值来评估条件，或者在履行意图期间使用这些值。

您可以为当前意图设置槽位值。如果对话的下一步是调用另一个意图，则可以设置新意图的槽位值。

如果被分配的槽位未填，或者无法解析 JSON 路径，则该属性将设置为 null。

使用槽位值和会话属性时，请使用以下句法：

- 槽位值：用大括号（“{}”）将槽位名称括起来。对于当前意图中的槽位值，您只需要使用槽位名称即可。例如，{slot}。如果要在下一个意图中设置值，则必须同时使用意图名称和槽位名称来标识该槽位。例如，{intent.slot}。

  示例：

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

  槽位的值可能为以下任一值：

  - 常量字符串
  - 引用 Amazon Lex 响应中的转录块的 JSON 路径（适用于 en-US 和 en-GB）
  - 会话属性

  示例：

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

- {username_slot_value} = [username]

> **Note**
>
> 槽位值也可以设置为 null。如果您需要重新引发某个已填槽位值，则必须先将该值设置为 null，然后再提示客户输入槽位值。如果被分配的槽位未填，或者无法解析 JSON 路径，则该属性将设置为 null。

- 会话属性：用方括号（“[]”）将属性名称括起来。例如，[sessionAttribute]。

  示例：

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

  会话属性的值可能为以下任一值：

  - 常量字符串
  - 引用 Amazon Lex 响应中的转录块的 JSON 路径（适用于 en-US 和 en-GB）
  - 槽位值参考

> **Note**
>
> 如果被分配的槽位未填，或者无法解析 JSON 路径，则该属性将设置为 null。

> **Note**
>
> Amazon Lex V2 于 2022 年 8 月 17 日发布了对用户对话管理方式的更改。借助此更改，您可以更好地控制用户的对话路径。有关更多信息，请参阅 了解对话流管理。2022 年 8 月 17 日之前创建的机器人不支持对话框代码挂钩消息、设置值、配置后续步骤和添加条件。

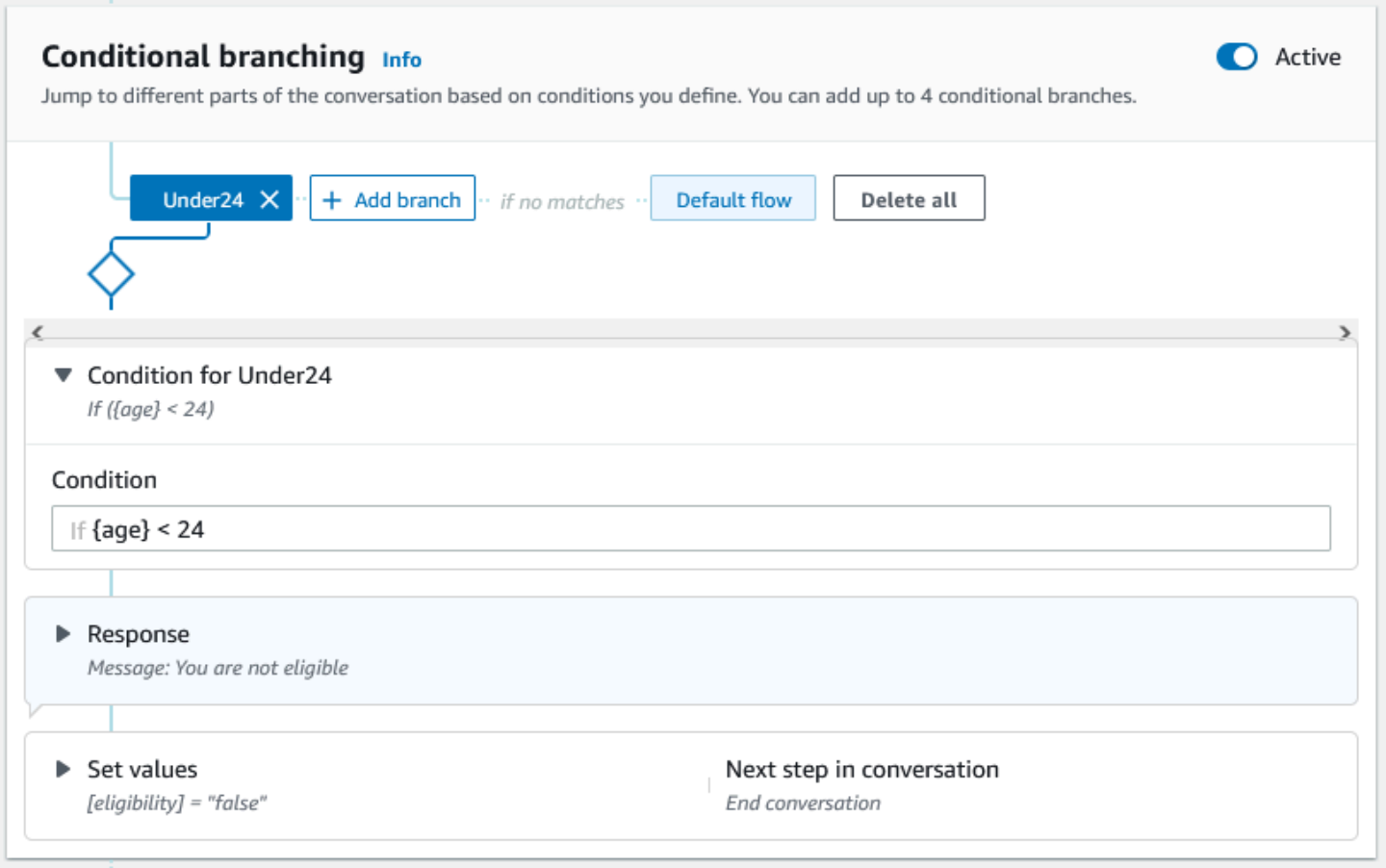
添加条件以构建对话的分支

您可以使用条件分支来控制客户与机器人的对话路径。您可以根据槽位值、会话属性、输入模式的内容和输入转录字段或来自 Amazon Kendra 的响应来构建对话分支。

您可以定义最多 4 个分支。每个分支均设有条件，并且必须在满足该条件的情况下，Amazon Lex V2 才能采用该分支。如果所有分支的条件均不满足，则将采用默认分支。

定义分支时，您可以定义 Amazon Lex V2 在与该分支对应的条件评估为 true 时应执行的操作。您可以定义以下任一操作：

- 发送给用户的回应。
- 要应用于槽位的槽位值。
- 当前会话的会话属性值。
- 对话的下一步。有关更多信息，请参阅 创建对话路径。



每个条件分支均设有布尔表达式，并且必须在满足该布尔表达式的情况下，Amazon Lex V2 才能采用该分支。您可以使用比较运算符、布尔运算符、函数和量词运算符来处理您的条件。例如，如果 {age} 槽位小于 24，则以下条件返回 true。

```
{age} < 24
```

如果 {toppings} 多值槽位包含“pineapple”一词，则以下条件返回 true。

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

对于更复杂的条件，您可以将多个比较运算符与布尔运算符组合使用。例如，如果 {make} 槽位值为“Honda” 且 {model} 槽位值为“Civic”，则以下条件返回 true。使用圆括号设置评估顺序。

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

以下主题提供了关于条件分支运算符和函数的详细信息。

> **Note**
>
> Amazon Lex V2 于 2022 年 8 月 17 日发布了对用户对话管理方式的更改。借助此更改，您可以更好地控制用户的对话路径。有关更多信息，请参阅 了解对话流管理。2022 年 8 月 17 日之前创建的机器人不支持对话框代码挂钩消息、设置值、配置后续步骤和添加条件。

主题

比较运算符

Amazon Lex V2 支持以下条件比较运算符：

- 等于 (=)
- 不等于 (!=)
- 小于 (<)
- 小于等于 (<=)
- 大于 (>)

- 大于等于 (>=)

使用比较运算符时，将遵循以下规则。

- 左侧必须是引用值。例如，要引用槽位值，请使用 {slotName}。要引用会话属性值，请使用 [attribute]。对于输入模式和输入转录，请使用 $.inputMode 和 $.inputTranscript。
- 右侧必须为常数，且类型必须与左侧相同。
- 系统将任何引用尚未设置的属性的表达式视为无效，并且不会对其求值。
- 比较多值槽位时，使用的值是所有解释值的逗号分隔列表。

该等比较是基于引用值的槽位类型进行的，按照以下方式进行解析：

- 字符串：根据字符串的 ASCII 表示进行比较。比较不区分大小写。
- 数字：将基于数字的槽位从字符串表示形式转换为数字，然后进行比较。
- 日期/时间：根据时间序列对基于时间的槽位进行比较。较早的日期或时间即视作较小。对于持续时间，较短的时间段即视为较小。

布尔运算符

Amazon Lex V2 支持使用布尔运算符来组合比较运算符。利用布尔运算符可创建类似以下内容的语句：

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

您还可以使用以下布尔运算符：

- AND (&&)
- OR (||)
- NOT (!)

量词运算符

量词运算符评估序列中的元素并确定一个或多个元素是否满足条件。

- CONTAINS：确定指定值是否被包含在多值槽位中，如果包含在多值槽位中，则返回 true。例如，如果用户要在订购的披萨上添加菠萝，则 {toppings} CONTAINS "pineapple" 将返回 true。

函数

函数必须以字符串 `fn.` 为前缀。该函数的参数是对槽位、会话属性或请求属性的引用。Amazon Lex V2 提供了用于从槽位值中获取信息的两个函数，即 sessionAttribute, 或 requestAttribute。

- fn.COUNT()：计算多值槽位中的值数量。

  例如，如果槽位 `{toppings}` 包含值“pepperoni, pineapple”：

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

- fn.IS_SET()：如果在当前会话中设置了槽位、会话属性或请求属性，则该值为 true。

  基于上一示例：

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

- fn.length () — 值是在当前会话中设置的会话属性、插槽值或插槽属性的值的长度。此功能不支持多值插槽或复合槽。

  例如：

  如果插槽中`{credit-card-number}`包含的值为 “123456781234”：

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

示例条件表达式

以下是一些示例条件表达式。注意：`$.` 表示 Amazon Lex JSON 响应的入口点。`$.` 之后的值将在检索该值的 Amazon Lex 响应中解析。只有支持 ASR 转录分数的相同区域才支持在 Amazon Lex 响应中使用对转录块的 JSON 路径引用的条件表达式。

值类型	应用场景	条件表达式
自定义槽位	`pizzaSize` 槽位值等于“large”	`{pizzaSize} = "large"`
自定义槽位	`pizzaSize` 等于“large”或“medium”	`{pizzaSize} = "large"` OR `{pizzaSize} = "medium"`

值类型	应用场景	条件表达式
自定义槽位	带有( ) 和 AND/OR 的表达式	{pizzaType} = "pepperoni" OR {pizzaSize} = "medium" OR {pizzaSize} = "small"
自定义槽位（多值槽位）	检查其中的一种配料是否是“Onion”（洋葱）	{toppings} CONTAINS "Onion"
自定义槽位（多值槽位）	配料的数量超过 3	fn.COUNT({topping}) > 2
AMAZON.AlphaNumeric	bookingID 是“ABC123”	{bookingID} = "ABC123"
AMAZON.Number	年龄槽位值大于 30	{age} > 30
AMAZON.Number	年龄槽位值等于 10	{age} = 10
AMAZON.Date	dateOfBirth 槽位值在 1990 年之前	{dateOfBirth} < "1990-10-01"
AMAZON.State	destinationState 槽位值等于“Washington”	{destinationState} = "washington"
AMAZON.Country	destinationCountry 槽位值不是“United States”	{destinationCountry} != "united states"
AMAZON.FirstName	firstName 槽位值是“John”	{firstName} = "John"
AMAZON.PhoneNumber	phoneNumber 槽位值是“716767891932”	{phoneNumer} = 716767891932
AMAZON.Percentage	检查百分比槽位值是否大于等于 78	{percentage} >= 78
AMAZON.EmailAddress	emailAddress 槽位值为“userA@hmail.com”	{emailAddress} = "userA@hmail.com"

值类型	应用场景	条件表达式
AMAZON.LastName	lastName 槽位值是“Doe”	{lastName} = "Doe"
AMAZON.City	“City”槽位值等于“Seattle”	{city} = "Seattle"
AMAZON.Time	“Time”是晚上 8 点之后	{time} > "20:00"
AMAZON.StreetName	streetName 槽位值是“Boren Avenue”	{streetName} = "boren avenue"
AMAZON.Duration	travelDuration 槽位值小于 2 小时	{travelDuration} < P2H
输入模式	输入模式为语音	$.inputMode = "Speech"
输入转录	输入转录等于“I want a large pizza”	$.inputTranscript = "I want a large pizza"
会话属性	检查 customer_subscription_type 属性	[customer_subcription_type] = "yearly"
请求属性	检查 retry_enabled 标志	((retry_enabled)) = "TRUE"
Kendra 响应	Kendra 响应包含常见问题	fn.IS_SET(((x-amz-lex:kendra-search-response-question_answer-question-1))
带转录的条件表达式	使用转录 JSON 路径的条件表达式	$.transcriptions[0].transcriptionConfidence < 0.8 AND $.transcriptions[1].transcriptionConfidence > 0.5

值类型	应用场景	条件表达式
设置会话属性	使用转录 JSON 路径和槽位值设置会话属性	`[sessionAttribute] = "$.transcriptions...." AND [sessionAttribute] = "{<slotName>}"`
设置槽位值	使用会话属性和转录 JSON 路径设置槽位值	`{slotName} = [<sessionAttribute>] AND {slotName} = "$.transcriptions...."`

> **Note**
>
> `slotName` 指的是 Amazon Lex 机器人中的槽位的名称。如果该槽位未解析 (null)，或者该槽位不存在，则运行时会忽略这些指定。`sessionAttribute` 指客户在构建时设置的会话属性的名称。

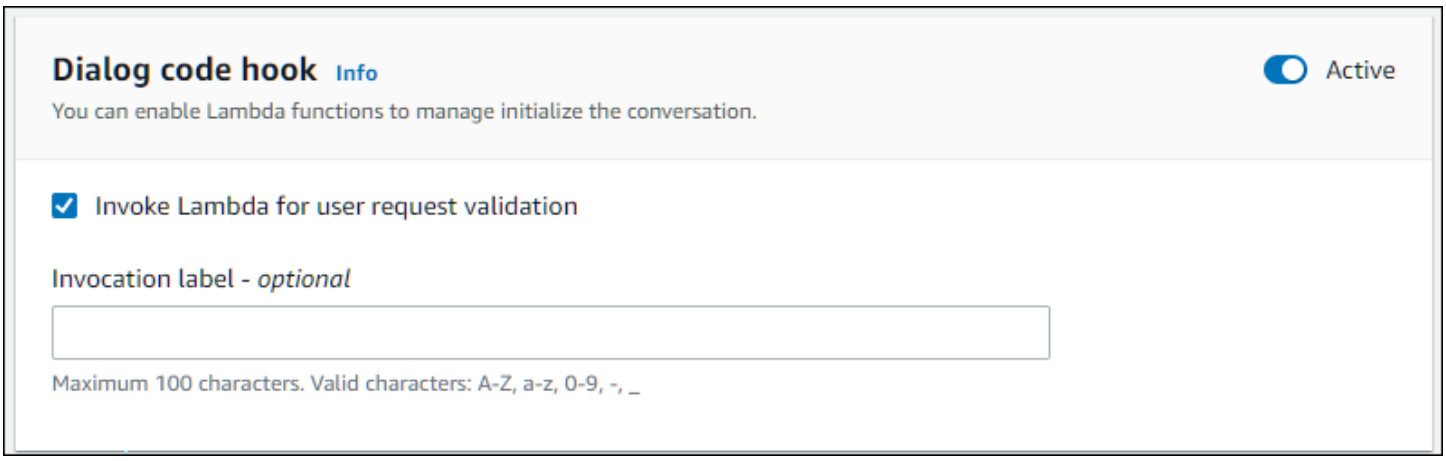
调用对话框代码挂钩

在对话的每一步中，当 Amazon Lex 向用户发送消息时，您均可以使用 Lambda 函数作为对话的下一步。您可以使用该函数，根据对话的当前状态来实现业务逻辑。

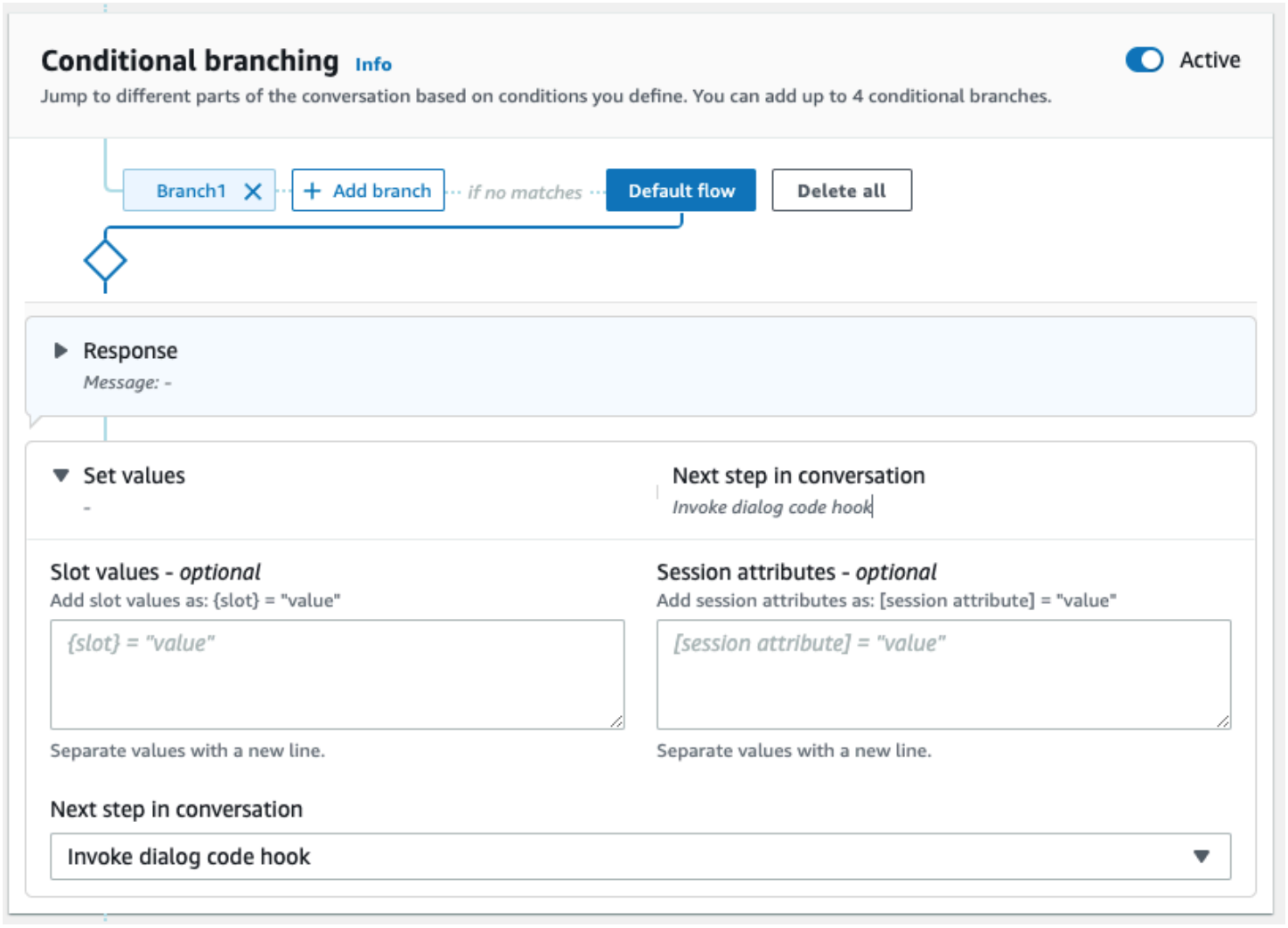
运行的 Lambda 函数与您正在使用的机器人别名相关联。要在意图中的所有对话框代码挂钩中调用 Lambda 函数，您必须针对该意图选择使用 Lambda 函数来初始化和验证。有关选择 Lambda 函数的更多信息，请参阅创建 Lambda 函数并将其附加到机器人别名。

使用 Lambda 函数需要两个步骤。首先，您需要在对话中的任何节点激活对话框代码挂钩。接下来，您需要将对话的下一步设置为使用对话框代码挂钩。

下图显示了已激活的对话框代码挂钩。

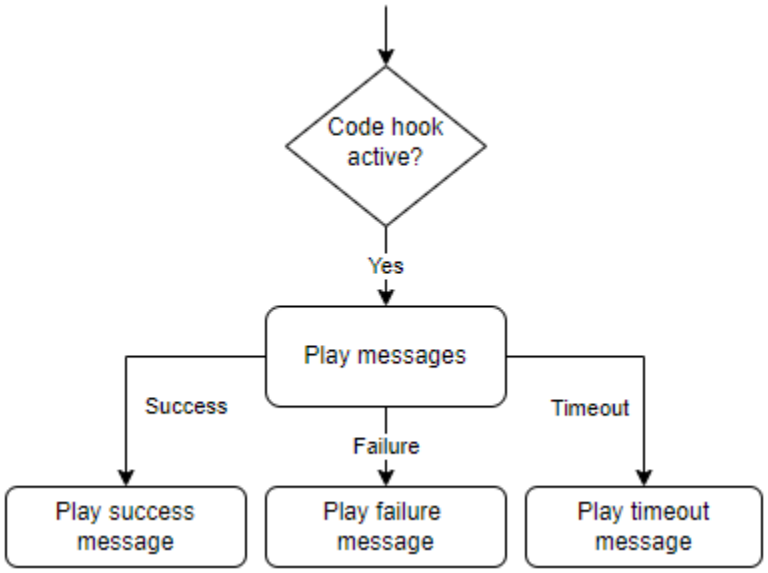


接下来，将代码挂钩设置为对话步骤的下一个操作。为此，您可以将对话的下一步配置为“调用对话框代码挂钩”。请参见下图中所示的条件分支，在该条件分支中，调用对话框代码挂钩是对话默认路径的下一步。



当代码挂钩处于活动状态时，您可以设置返回给用户的以下三个响应：

- 成功：在 Lambda 函数成功完成时发送该响应。
- 失败：如果运行 Lambda 函数时出现问题，或者 Lambda 函数返回的 `intent.state` 值为 `Failed`，则发送该响应。
- 超时：如果 Lambda 函数未在其配置的超时时间内完成，则发送该响应。



选择 Lambda 对话框代码挂钩，然后选择高级选项以查看与 Lambda 函数调用相对应的三个响应选项。您可以设置值、配置后续步骤，并应用与每个响应相对应的条件以设计对话流程。如果没有条件或明确的下一步操作，Amazon Lex V2 将根据对话的当前状态决定下一步操作。

您还可以在高级选项页面上选择启用或禁用 Lambda 函数调用。启用该函数后，将通过 Lambda 调用来调用对话框代码挂钩，然后根据 Lambda 调用结果显示成功、失败或超时消息。禁用该功能后，Amazon Lex V2 不运行 Lambda 函数，而是将对话代码挂钩视作成功并继续运行。

您还可以设置调用标签，该标签将在被该消息调用时发送到 Lambda 函数。您可以使用该调用标签来帮助识别要运行的 Lambda 函数的部分。

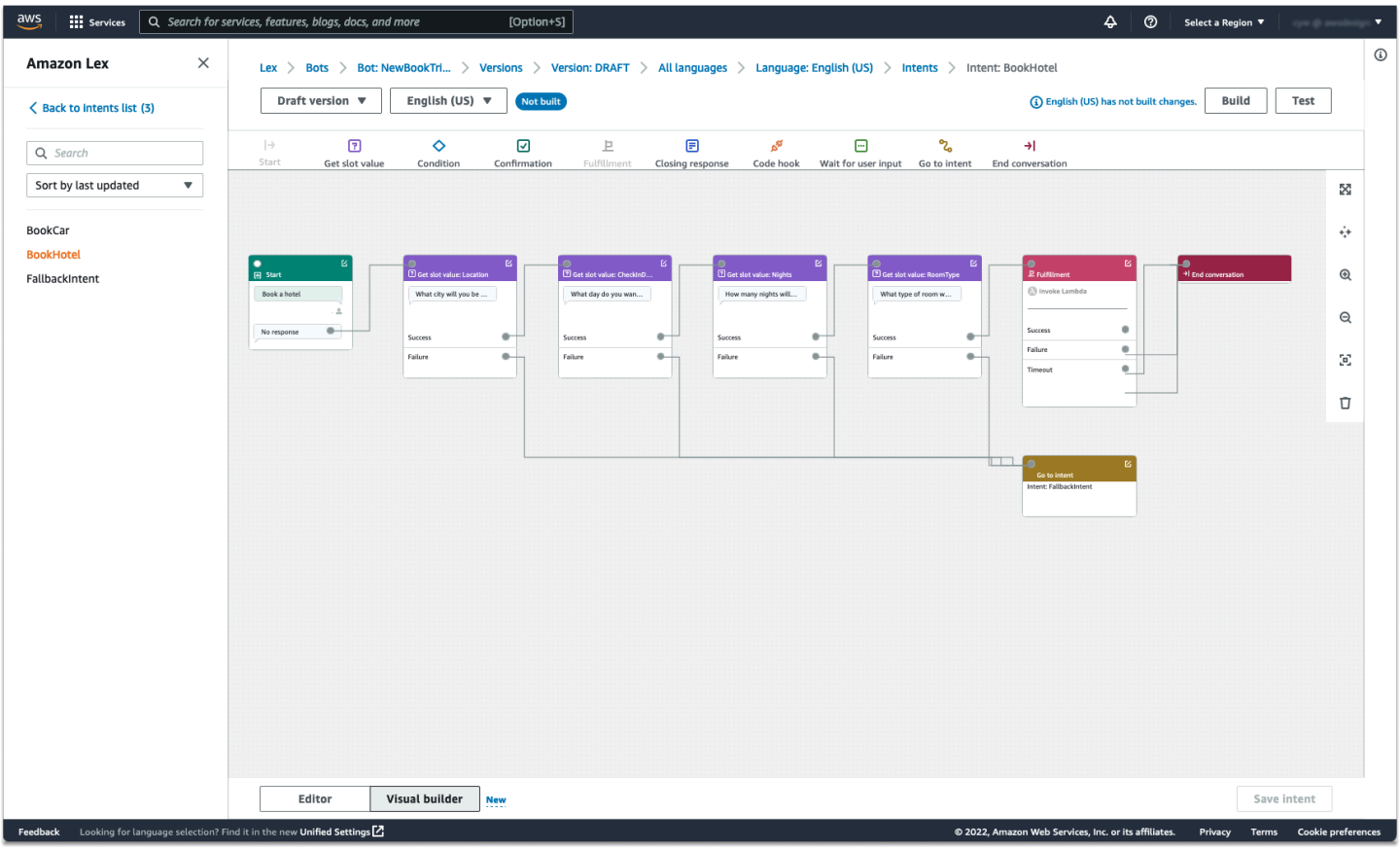
> **Note**
>
> Amazon Lex V2 于 2022 年 8 月 17 日发布了对用户对话管理方式的更改。借助此更改，您可以更好地控制用户的对话路径。有关更多信息，请参阅 了解对话流管理。2022 年 8 月 17 日之前创建的机器人不支持对话框代码挂钩消息、设置值、配置后续步骤和添加条件。

使用可视化对话生成器

可视化对话生成器是一个拖放式对话生成器，可在丰富的视觉环境中利用意图来轻松设计并可视化对话路径。

要访问可视化对话生成器，请执行以下操作：

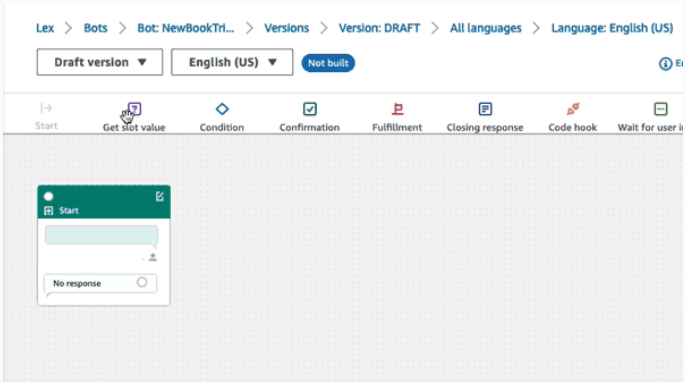
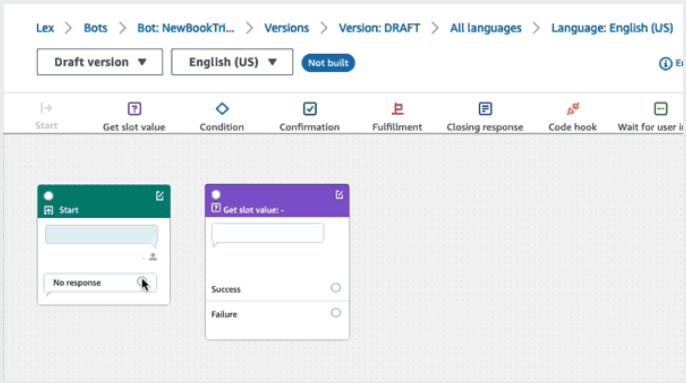
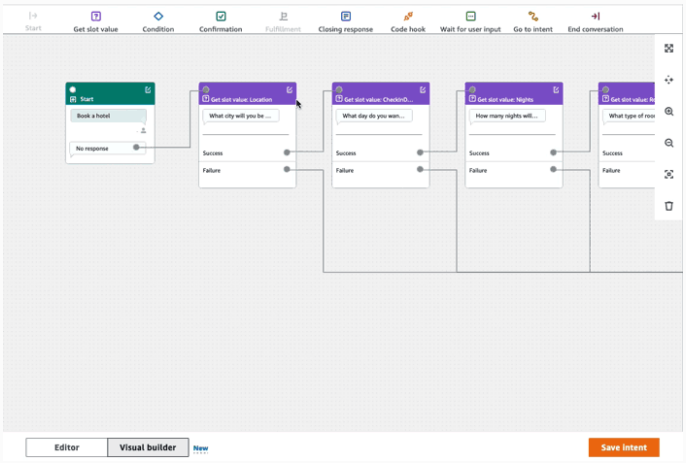
1. 在 Amazon Lex V2 控制台的左侧导航窗格中，选择机器人，然后选择意图。
2. 通过以下方式中的任一种打开意图编辑器：
   - 选择意图部分右上角的添加意图，然后选择添加空意图或内置意图。
   - 意图部分中选择意图的名称。
3. 在意图编辑器中，在屏幕底部的窗格中选择可视化生成器以打开可视化对话生成器。
4. 要返回菜单意图编辑器界面，请选择编辑器。

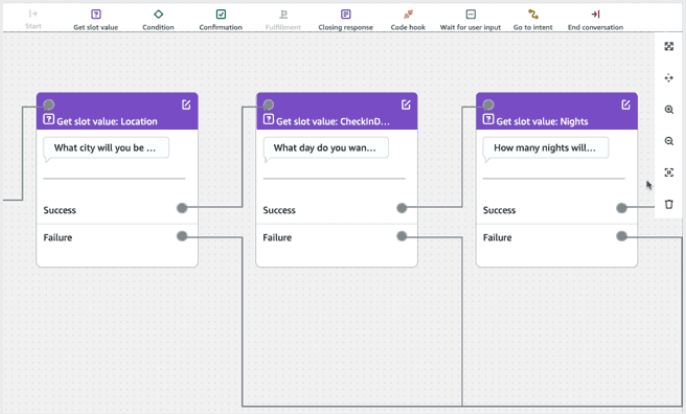
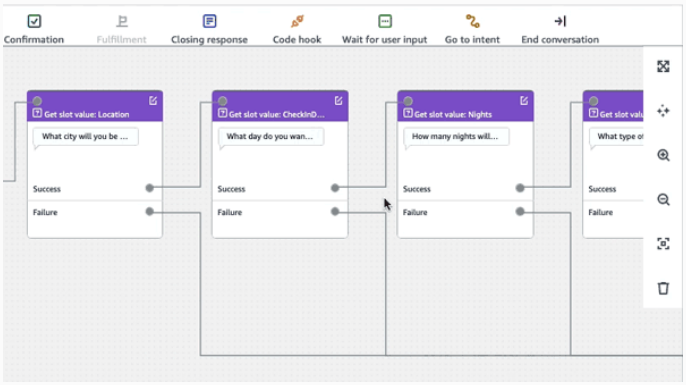
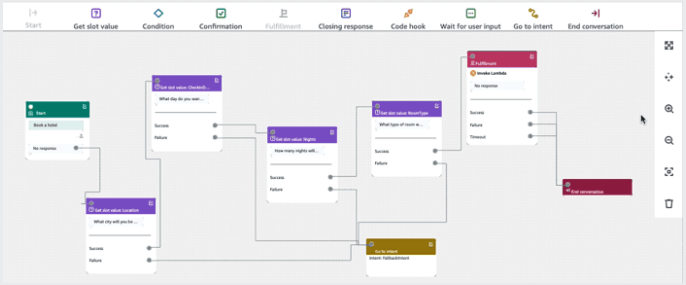


可视化对话生成器采用更直观的用户界面，能够帮助您直观地查看和修改对话流程。您可以拖放对应的数据块，以便扩展现有流程或重新排列对话步骤的顺序。您无需编写任何 Lambda 代码即可开发具有复杂分支的对话流程。

此更改有助于将对话流程设计与 Lambda 中的其他业务逻辑分开。可视化对话生成器可以与现有的意图编辑器配合使用，并可用于构建对话流程。但是，对于更复杂的对话流程，建议使用可视化编辑器视图。

当您保存意图时，Amazon Lex V2 可以在确定连接断开、Amazon Lex V2 建议连接时自动连接意图，或者您可以针对该块选择自己的连接。

操作	示例
向工作区添加数据块	
建立数据块之间的连接	
在块上打开配置面板	

操作	示例
缩放以适合	
从对话流程中删除数据块	
自动清理工作区	

术语：

数据块：对话流的基本构成单元。每个数据块都有特定的功能，可以处理对话的不同用例。

端口：每个数据块均包含可用于将一个数据块连接到另一个数据块的端口。数据块可以包含输入端口和输出端口。每个输出端口代表对应数据块的特定功能变体（例如错误、超时或成功）。

边缘：边缘是指一个数据块的输出端口与另一个数据块的输入端口之间的连接。它是对话流程中分支的一部分。

对话流程：一组由边缘连接的数据块，用于描述与客户的意图层面的交互。

数据块

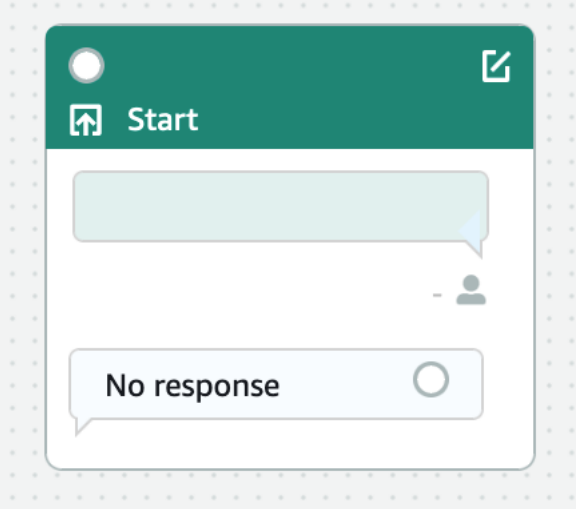
数据块是对话流程设计的构成元素。这些数据块代表意图中的不同状态，从意图开始到用户输入，再到结束。

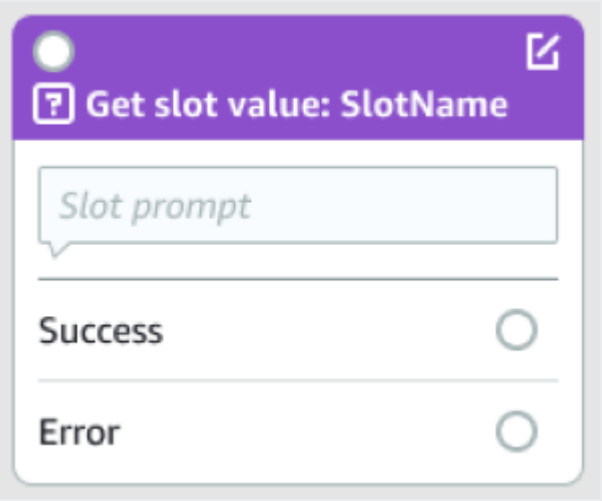
根据数据块的类型，每个数据块均有一个入口点以及一个或多个出口点。当对话通过出口点时，可以为每个出口点配置相应的消息。对于具有多个出口点的数据块，出口点与对应于该节点的状态相关。对于条件节点，出口点代表不同的条件。

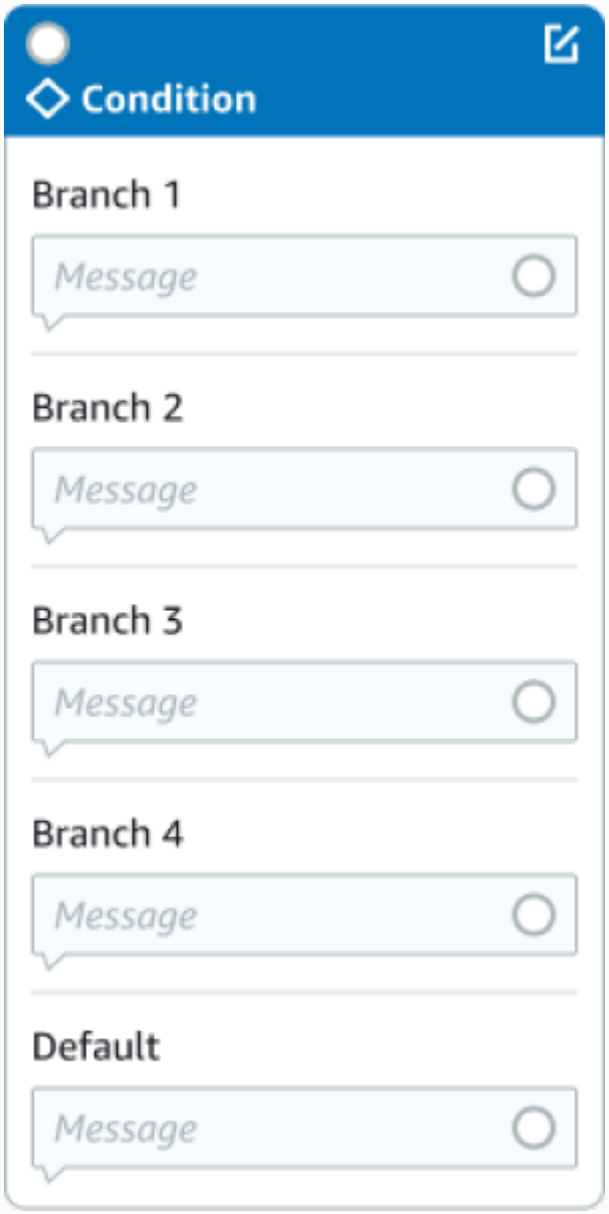
每个数据块都设有配置面板，点击数据块右上角的编辑图标即可打开该面板。配置面板包含可以配置为与每个数据块相对应的详细字段。

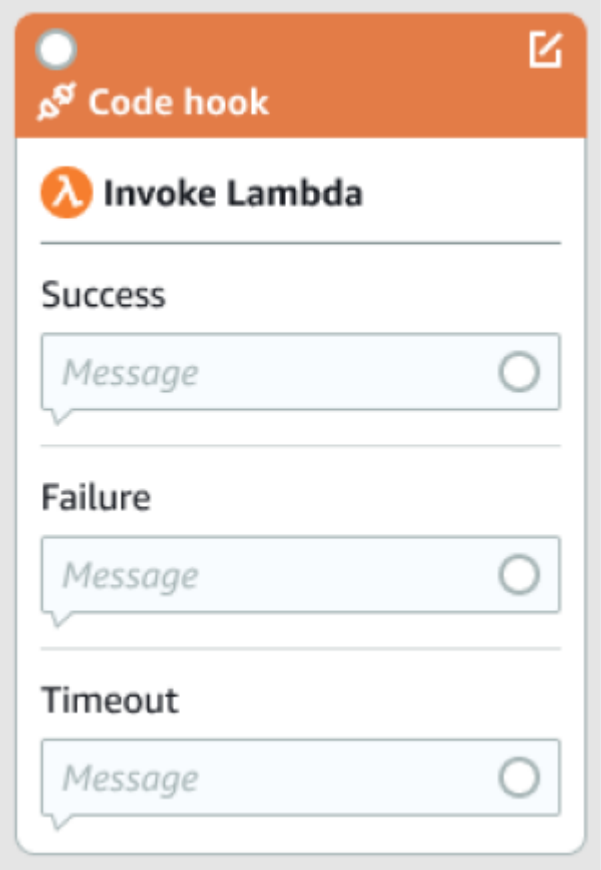
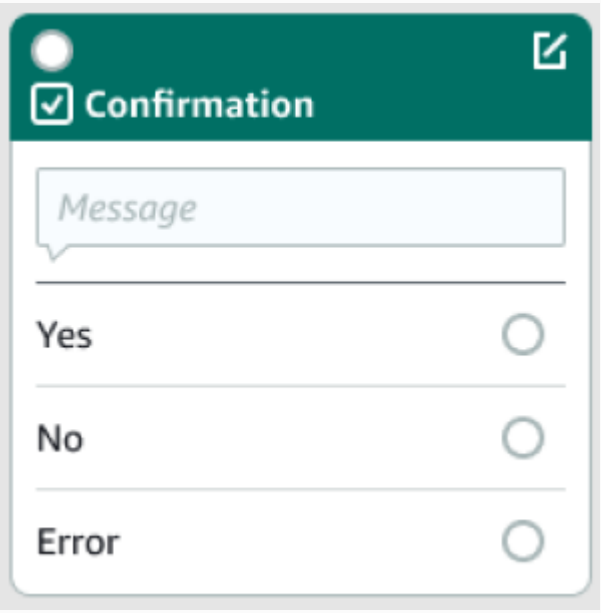
可以通过拖动新的数据块直接在该节点上配置机器人提示和消息，也可以在右侧面板中修改该等机器人提示和消息以及数据块的其他属性。

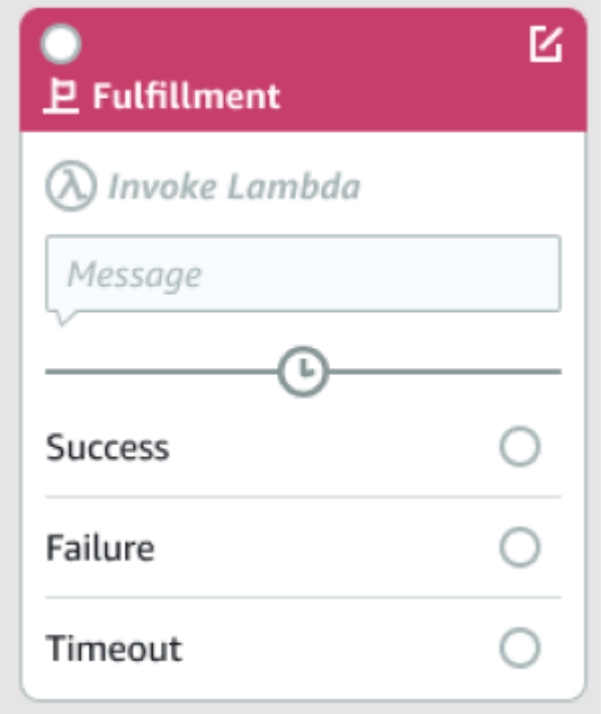
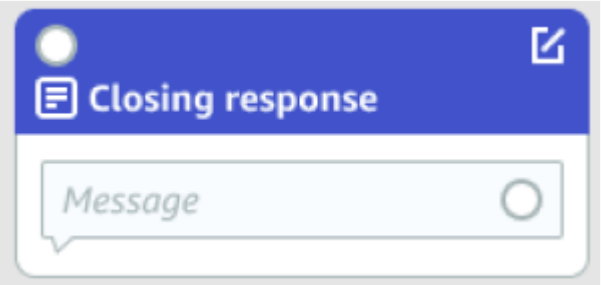
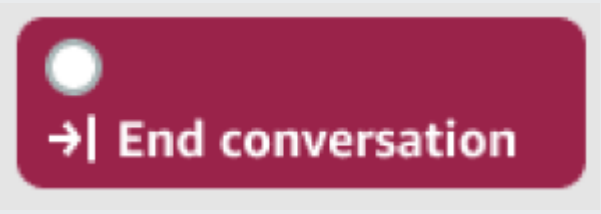
数据块类型：以下是可视化对话生成器中可以使用的数据块类型。

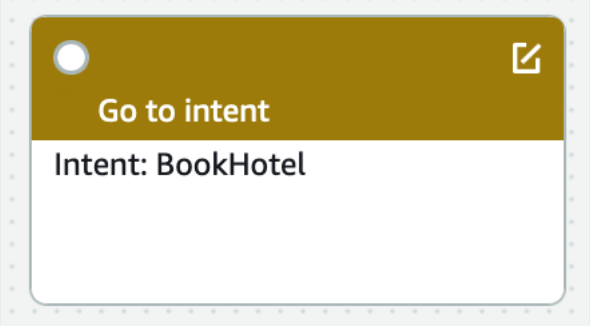
数据块类型	Block（阻止）
开始：对话流中的根数据块或第一个数据块。也可以对此数据块进行配置，以便机器人可以发送初始响应（意图已被识别的消息）。有关更多信息，请参阅初始回应。	

数据块类型	Block（阻止）
获取插槽值：此数据块尝试引发单个插槽的值。此数据块的设置是等待客户对插槽引发提示做出响应。有关更多信息，请参阅Slots。	

数据块类型	Block（阻止）
条件：此数据块包含条件。其中最多包含 4 个自定义分支（带条件）以及一个默认分支。有关更多信息，请参阅添加条件以构建对话的分支。	

数据块类型	Block（阻止）
对话框代码挂钩：此数据块处理对话框 Lambda 函数的调用。此数据块包含基于对话框 Lambda 函数成功、失败或超时的机器人响应。有关更多信息，请参阅调用对话框代码挂钩。	
确认：此数据块会在意图履行之前向客户进行查询。其中包含基于客户对确认提示做出“是”或“否”回复的机器人响应。有关更多信息，请参阅确认。	

数据块类型	Block（阻止）
履行：此数据块处理意图的履行，通常在插槽引发之后。可以将其配置为在履行成功或失败时调用 Lambda 函数并使用消息进行响应。有关更多信息，请参阅执行。	
关闭响应：依据此数据块，机器人将回复一条消息，然后再结束对话。有关更多信息，请参阅关闭响应。	
结束对话：此数据块指示对话流程的结束。	
等待用户输入：此数据块可用于捕获客户的输入并根据言语切换到其他意图。	

数据块类型	Block（阻止）
转到意图：该数据块可以用于转到一个新的意图，或者直接引发该意图的特定插槽。	

端口类型

所有数据块都包含一个输入端口，用于连接其父数据块。对话只能从其父数据块的输出端口流向特定数据块的输入端口。但是，数据块可能包含零个、一个或多个输出端口。如果数据块没有任何输出端口，则表示当前意图中对话流程的结束（GoToIntent、EndConversation、WaitForUserInput）。

意图设计规则：

- 意图中的所有流程均始于起始数据块。
- 与每个出口点对应的消息是可选的。
- 您可以将数据块配置为在配置面板中设置与每个出口点相对应的值。
- 一个意图中的单个流程中只能存在一个开始、确认、履行和关闭数据块。可能存在多个条件、对话框代码挂钩、获取插槽值、结束对话、转移和等待用户输入数据块。
- 条件数据块无法与条件数据块直接连接。对话框代码挂钩也是如此。
- 循环流允许使用三个数据块，但不允许使用指向“开始意图”的传入连接器。
- 可选插槽没有传入连接器或传出连接，主要用于捕获意图引发期间存在的任何数据。对话路径中的所有其他插槽都必须是必填的插槽。

数据块：

- 起始数据块必须具有传出边缘。
- 如果插槽是必需的，则每个获取插槽值数据块都必须具有从成功端口输出的传出边缘。
- 如果处于活动状态，则每个条件数据块都必须具有从每个分支输出的传出边缘。
- 一个条件数据块无法具有多个父项。
- 活动的条件数据块必须具有传入边缘。

- 每个活动代码挂钩数据块必须具有从成功、失败和超时中的每个端口输出的传出边缘。
- 活动的代码挂钩数据块必须具有传入边缘。
- 活动的确认数据块必须具有传入边缘。
- 活动的履行数据块必须具有传入边缘。
- 活动的关闭数据块必须具有传入边缘。
- 一个条件数据块必须至少具有一个非默认分支。
- 跳转意图数据块必须指定意图。

边缘：

- 条件数据块无法与另一个条件数据块直接连接。
- 代码挂钩数据块无法与另一个代码挂钩数据块直接连接。
- 条件数据块只能连接到零个或一个代码挂钩数据块。
- 连接（代码挂钩 -> 条件 -> 代码挂钩）无效。
- 履行数据块无法具有代码挂钩数据块子项。
- 作为履行数据块的子项的条件数据块无法具有代码挂钩数据块子项。
- 关闭数据块无法具有代码挂钩数据块子项。
- 作为关闭数据块的子项的条件数据块无法具有代码挂钩数据块子项。
- 一个包含开始、确认或获取插槽值数据块的关联链中只能有一个代码挂钩数据块。

> **Note**
>
> Amazon Lex V2 于 2022 年 8 月 17 日发布了对用户对话管理方式的更改。借助此更改，您可以更好地控制用户的对话路径。有关更多信息，请参阅了解对话流管理。2022 年 8 月 17 日之前创建的机器人不支持对话框代码挂钩消息、设置值、配置后续步骤和添加条件。

内置意图

对于常见操作，您可以使用标准内置意图库。要基于内置意图创建意图，请在控制台中选择一个内置意图，为其指定新名称。新意图将具有基础意图的配置，例如示例言语。

在当前实现中，不能执行以下操作：

- 向基础意图添加示例言语，或从基础意图中删除示例言语
- 为内置意图配置槽

向机器人添加内置意图

1. 登录 AWS Management Console 并打开 Amazon Lex 控制台，网址为 https://console.aws.amazon.com/lex/。
2. 选择要添加内置意图的机器人。
3. 在左侧菜单中，选择语言，然后选择意图。
4. 选择添加意图，然后选择使用内置意图。
5. 在内置意图中，选择要使用的意图。
6. 指定意图的名称，然后选择添加。
7. 使用意图编辑器，根据需要为您的机器人配置意图。

主题

AMAZON.CancelIntent

对表示用户要取消当前交互的单词和短语的响应。在结束与用户的交互之前，您的应用程序可以使用此意图来删除插槽类型值和其他属性。

常见言语：

- 取消
- 没关系
- 算了

AMAZON.FallbackIntent

当意图的用户输入与机器人的预期不符时，您可以配置 Amazon Lex V2 以调用回退意图。例如，如果用户输入“我想要订购糖果”，与 `OrderFlowers` 机器人中的意图不匹配，Amazon Lex V2 会调用回退意图来处理该响应。

当您使用控制台创建机器人或使用CreateBotLocale操作向机器人添加区域时，内置`AMAZON.FallbackIntent`意图类型会自动添加到您的机器人中。

调用回退意图分两步。在第一步中，基于用户的输入来匹配回退意图。匹配回退意图时，自动程序的行为方式取决于为提示配置的重试次数。

在以下情况下，Amazon Lex V2 与回退意图匹配：

- 用户输入到意图的内容不符合自动程序的预期
- 音频输入为噪声，或文本输入未被识别为单词。
- 用户的输入不明确且 Amazon Lex V2 无法确定要调用的意图。

在以下情况下调用回退意图：

- 经过配置的尝试次数后，意图无法将用户输入识别为槽位值。
- 经过配置的尝试次数后，意图无法将用户输入识别为对确认提示的响应。

您不能将以下内容添加到回退意图：

- 言语
- 槽值
- 确认提示

将 Lambda 函数用于回退意图

调用回退目的时，响应取决于 CreateIntent 操作的 `fulfillmentCodeHook` 参数设置。自动程序执行下列操作之一：

- 将意图信息返回给客户端应用程序。
- 调用别名的验证和履行 Lambda 函数。它通过为会话设置的会话变量调用该函数。

有关设置在调用回退目的时的响应的更多信息，请参见 CreateIntent 操作的 `fulfillmentCodeHook` 参数。

如果将 Lambda 函数与回退意图配合使用，则可以使用此函数来调用另一个意图或与用户进行某种形式的通信，例如收集回呼号码或开启与客户服务代表的会话。

回退意图可以在同一会话中多次调用。例如，假设您的 Lambda 函数使用 `ElicitIntent` 对话框操作来提示用户输入一个不同的意图。如果 Amazon Lex V2 在配置的尝试次数后无法推断用户的意图，将再次调用回退意图。当用户在配置的尝试次数后未使用有效的槽位值进行响应时，它也会调用回退意图。

您可以配置 Lambda 函数以跟踪使用会话变量调用回退意图的次数。如果该意图的调用次数超过了 Lambda 函数中设置的阈值，则该函数可以采取不同的操作。有关会话变量的更多信息，请参阅设置会话属性。

AMAZON.HelpIntent

对表示用户在与机器人交互时需要帮助的词语或短语的响应。调用该意图时，您可以对您的 Lambda 函数或应用程序进行配置，以便提供有关机器人功能的信息、询问有关需要帮助的方面的后续问题，或者将交互移交给人工客服。

常见言语：

- help
- 我需要帮助
- 可以帮帮我吗

AMAZON.KendraSearchIntent

要搜索已使用 Amazon Kendra 编制索引的文档，请使用 `AMAZON.KendraSearchIntent` 意图。在与用户进行的对话中，如果 Amazon Lex V2 无法确定下一个操作，则会触发搜索意图。

`AMAZON.KendraSearchIntent` 仅适用于英语（美国）区域以及美国东部（弗吉尼亚州北部）、美国西部（俄勒冈州）和欧洲地区（爱尔兰）。

Amazon Kendra 是一项 machine-learning-based 搜索服务，用于索引 PDF 文档或微软 Word 文件等自然语言文档。它可以搜索已编制索引的文档并为问题返回以下类型的响应：

- 答案
- 可能是问题答案的常见问题解答条目
- 与问题相关的文档

有关使用 AMAZON.KendraSearchIntent 的示例，请参阅示例：为 Amazon Kendra 索引创建常见问题机器人。

如果您为机器人配置了 AMAZON.KendraSearchIntent 意图，则 Amazon Lex V2 在无法确定某个意图的用户言语时，会调用该意图。如果没有来自 Amazon Kendra 的响应，则对话将按照机器人中的配置继续进行。

> **Note**
>
> 目前在槽位引发期间，Amazon Lex V2 不支持 AMAZON.KendraSearchIntent。如果 Amazon Lex V2 无法确定某个槽位的用户言语，即调用 AMAZON.FallbackIntent。

当您在同一机器人中将 AMAZON.KendraSearchIntent 与 AMAZON.FallbackIntent 配合使用时，Amazon Lex V2 将按照以下方式使用意图：

1. Amazon Lex V2 调用 AMAZON.KendraSearchIntent。该意图调用 Amazon Kendra Query 操作。
2. 如果 Amazon Kendra 返回响应，则 Amazon Lex V2 向用户显示该结果。
3. 如果没有来自 Amazon Kendra 的响应，则 Amazon Lex V2 将重新提示用户。下一个操作取决于用户的响应。
   - 如果用户的响应包含 Amazon Lex V2 可识别的言语（例如填充槽位值或确认意图），则与用户的对话将按照机器人的配置继续进行。
   - 如果用户的响应未包含 Amazon Lex V2 可识别的言语，则 Amazon Lex V2 将再次调用 Query 操作。
4. 如果在配置的重试次数之后没有响应，Amazon Lex V2 将调用 AMAZON.FallbackIntent 并结束与用户的对话。

有三种方法可以使用 AMAZON.KendraSearchIntent 向 Amazon Kendra 发出请求：

- 让搜索意图为您提出请求。Amazon Lex V2 以用户的言语作为搜索字符串调用 Amazon Kendra。在创建意图时，您可以定义限制 Amazon Kendra 返回的响应数的查询筛选条件字符串。Amazon Lex V2 使用查询请求中的筛选条件。
- 使用您的 Lambda 函数向请求添加其他查询参数以缩小搜索结果范围。您将包含 Amazon Kendra 查询参数的 `kendraQueryFilterString` 字段添加到 `delegate` 对话框操作。使用 Lambda 函数向请求添加查询参数时，这些查询参数将优先于您在创建意图时定义的查询筛选条件。
- 使用 Lambda 函数创建新查询。您可以创建 Amazon Lex V2 发送的完整 Amazon Kendra 查询请求。您可以在 `delegate` 对话操作的 `kendraQueryRequestPayload` 字段中指定查询。`kendraQueryRequestPayload` 字段优先于 `kendraQueryFilterString` 字段。

要在创建机器人时指定 `queryFilterString` 参数，或者在对话 Lambda 函数中调用 `delegate` 操作时指定 `kendraQueryFilterString` 字段，请指定用作 Amazon Kendra 查询的属性筛选条件的字符串。如果字符串不是有效的属性筛选器，您将在运行时收到 `InvalidBotConfigException` 异常。有关属性筛选条件的更多信息，请参阅《Amazon Kendra 开发人员指南》中的使用文档属性筛选查询。

要控制 Amazon Lex V2 发送到 Amazon Kendra 的查询，您可以在 Lambda 函数的 `kendraQueryRequestPayload` 字段中指定查询。如果查询无效，则 Amazon Lex V2 返回 `InvalidLambdaResponseException` 异常。有关更多信息，请参阅《Amazon Kendra 开发人员指南》中的查询操作。

有关如何使用 `AMAZON.KendraSearchIntent` 的示例，请参阅示例：为 Amazon Kendra 索引创建常见问题机器人。

Amazon Kendra 搜索的 IAM 策略

要使用该`AMAZON.KendraSearchIntent`意图，您必须使用能够提供 AWS Identity and Access Management (IAM) 策略的角色，这些策略允许 Amazon Lex V2 担任有权调用 Amazon `Query` Kendra 意图的运行时角色。您使用的 IAM 设置取决于您是使用 Amazon Lex V2 控制台创建，还是使用 AWS 开发工具包或 AWS Command Line Interface (AWS CLI) 创建。`AMAZON.KendraSearchIntent`使用控制台时，您可以选择向 Amazon Lex V2 服务相关角色添加调用 Amazon Kendra 的权限，或使用专门用于调用 Amazon Kendra `Query` 操作的角色。使用 AWS CLI 或 SDK 创建 Intent 时，必须使用专门用于调用`Query`操作的角色。

附加权限

您可以使用控制台将访问 Amazon Kendra Query 操作的权限附加到默认 Amazon Lex V2 服务相关角色。当您将权限附加到服务相关角色时，无需专门创建和管理运行时角色即可连接到 Amazon Kendra 索引。

用于访问 Amazon Lex V2 控制台的用户、角色或组必须有权限管理角色策略。将以下 IAM 策略附加到控制台访问角色。当您授予这些权限时，角色将有权更改现有服务相关角色策略。

```
{
"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

您可以使用控制台 AWS CLI、或 API 来指定在调用 Amazon Kendra Query 操作时要使用的运行时角色。

用于指定运行时角色的用户、角色或组必须具有 iam:PassRole 权限。以下策略定义权限。您可以使用 iam:AssociatedResourceArn 和 iam:PassedToService 条件上下文键进一步限制权限的范围。有关更多信息，请参阅AWS Identity and Access Management 用户指南中的 IAM 和 AWS STS 条件上下文密钥。

```
{
```

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

Amazon Lex V2 调用 Amazon Kendra 时所需使用的运行时角色必须具有 `kendra:Query` 权限。当您使用现有 IAM 角色获取调用 Amazon Kendra `Query` 操作的权限时，该角色必须附加以下策略。

您可以使用 IAM 控制台、IAM API 或 AWS CLI 创建策略并将其附加到角色。这些说明使用 AWS CLI 创建角色和策略。

> **Note**
>
> 以下代码针对 Linux 和 macOS 编排了格式。对于 Windows，将 Linux 行继续符 (\) 替换为脱字号 (^)。

向角色添加 Query 操作权限

1. 在当前目录中创建一个名为 **`KendraQueryPolicy.json`** 的文档，向其中添加以下代码并保存

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

2. 在中 AWS CLI，运行以下命令来创建用于运行 Amazon Kendra `Query` 操作的 IAM 策略。

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

3. 将该策略附加到用于调用 Query 操作的 IAM 角色。

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

您可以选择更新 Amazon Lex V2 服务相关角色或使用您在为机器人创建 AMAZON.KendraSearchIntent 时所创建的角色。以下过程演示如何选择要使用的 IAM 角色。

为 AMAZON.KendraSearchIntent 指定运行时角色

1. 登录 AWS Management Console 并打开 Amazon Lex 控制台，网址为 https://console.aws.amazon.com/lex/。
2. 选择要向其添加 AMAZON.KendraSearchIntent 的自动程序。
3. 选择意图旁边的加号（+）。
4. 在添加意图中，选择搜索现有意图。
5. 在搜索意图中，输入 **AMAZON.KendraSearchIntent**，然后选择添加。
6. 在复制内置意图中，输入意图的名称，如 **KendraSearchIntent**，然后选择添加。
7. 打开 Amazon Kendra 查询部分。
8. 对于 IAM 角色，选择下列选项之一：
   - 要更新 Amazon Lex V2 服务相关角色以便机器人能够查询 Amazon Kendra 索引，请选择添加 Amazon Kendra 权限。
   - 要使用有权调用 Amazon Kendra Query 操作的角色，请选择使用现有角色。

使用请求和会话属性作为筛选器

要从来自 Amazon Kendra 的响应中筛选出与当前对话相关的项目，请在创建机器人时添加 queryFilterString 参数以使用会话和请求属性作为筛选条件。您可以在创建意图时指定属性的占位符，以便 Amazon Lex V2 在调用 Amazon Kendra 之前将其替换为值。有关请求属性的更多信息，请参阅设置请求属性。有关会话属性的更多信息，请参阅 设置会话属性。

以下是使用字符串来筛选 Amazon Kendra 查询的 queryFilterString 参数示例。

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

以下是使用名为 "SourceURI" 的会话属性来筛选 Amazon Kendra 查询的 queryFilterString 参数示例。

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

以下是使用名为 "DepartmentName" 的请求属性来筛选 Amazon Kendra 查询的 queryFilterString 参数示例。

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

AMAZON.KendraSearchInteng 筛选条件使用的格式与 Amazon Kendra 搜索筛选条件的格式相同。有关更多信息，请参阅《Amazon Kendra 开发人员指南》中的使用文档属性筛选搜索结果。

与 AMAZON.KendraSearchIntent 一起使用的查询筛选条件字符串必须确保每个筛选条件的首字母为小写字母。例如，以下是 AMAZON.KendraSearchIntent 的有效查询筛选条件。

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

使用搜索响应

Amazon Kendra 在意图 `IntentClosingSetting` 语句的响应中返回对搜索的响应。除非 Lambda 函数生成结束响应消息，否则意图必须具有 `closingResponse` 语句。

Amazon Kendra 有五种类型的响应。

- 以下两个响应需要您为 Amazon Kendra 索引设置常见问题。有关更多详细信息，请参阅直接向索引添加问题和答案。
  - `x-amz-lex:kendra-search-response-question_answer-question-<N>`：与搜索匹配的常见问题中的问题。
  - `x-amz-lex:kendra-search-response-question_answer-answer-<N>`：与搜索匹配的常见问题中的答案。
- 以下三个响应需要您为 Amazon Kendra 索引设置数据来源。有关更多详细信息，请参阅创建数据来源。
  - `x-amz-lex:kendra-search-response-document-<N>`：索引中与言语文本相关的文档摘录。
  - `x-amz-lex:kendra-search-response-document-link-<N>` — 索引中与言语文本相关的文档的 URL。
  - `x-amz-lex:kendra-search-response-answer-<N>` — 索引中能作为问题答案的文档摘录。

在 `request` 属性中返回响应。每个属性最多可以具有五个响应，编号为 1 到 5。有关响应的更多信息，请参阅《Amazon Kendra 开发人员指南》中的响应类型。

`closingResponse` 语句必须具有一个或多个消息组。每个消息组都包含一条或多条消息。每条消息均可以包含一个或多个占位符变量，这些变量替换为来自 Amazon Kendra 的响应中的请求属性。消息组中必须至少有一条消息的所有变量都由运行时响应中的请求属性值替换，或者组中必须有一条没有占位符变量的消息。请求属性使用双括号 (“((“”))”) 进行设置。以下消息组消息与来自 Amazon Kendra 的任何响应匹配：

- “我为你找到了一个常见问题解答问题：((x-amz-lex: kendra-search-response-question _answer-question-1))，答案是 ((: _answer-answer-answer-1))” x-amz-lex kendra-search-response-question
- “我找到了一份有用的文档的摘录：((x-amz-lex: kendra-search-response-document -1))”
- “我想你的问题的答案是 ((x-amz-lex: kendra-search-response-answer -1))”

使用 Lambda 函数管理请求和响应

AMAZON.KendraSearchIntent 意图可以使用您的对话代码挂钩和履行代码挂钩来管理对 Amazon Kendra 的请求和响应。当您想要修改发送到 Amazon Kendra 的查询时，请使用对话代码挂钩 Lambda 函数；当您想要修改响应时，请使用履行代码挂钩 Lambda 函数。

使用对话代码挂钩创建查询

您可以使用对话代码挂钩创建要发送到 Amazon Kendra 的查询。使用对话代码挂钩是可选的。如果您未指定对话框代码挂钩，则 Amazon Lex V2 会根据用户言语来构造查询并使用您在配置意图时提供的 queryFilterString（如有）。

您可以在对话框代码挂钩响应中使用两个字段来修改对 Amazon Kendra 的请求：

- kendraQueryFilterString — 使用此字符串为 Amazon Kendra 请求指定属性筛选条件。您可以使用索引中定义的任何索引字段筛选查询。有关筛选字符串的结构，请参阅《Amazon Kendra 开发人员指南》中的使用文档属性筛选查询。如果指定的筛选器字符串无效，则会出现 InvalidLambdaResponseException 异常。kendraQueryFilterString 字符串将覆盖为意图配置的 queryFilterString 中指定的任何查询字符串。
- kendraQueryRequestPayload — 使用此字符串指定 Amazon Kendra 查询。您的查询可以使用 Amazon Kendra 的任何功能。如果您没有指定有效的查询，则会出现 InvalidLambdaResponseException 异常。有关更多信息，请参阅《Amazon Kendra 开发人员指南》中的查询。

创建筛选条件或查询字符串后，将响应发送到 Amazon Lex V2，并将响应的 dialogAction 字段设置为 delegate。Amazon Lex V2 将查询发送到 Amazon Kendra，然后将查询响应返回给履行代码挂钩。

对响应使用实现代码挂钩

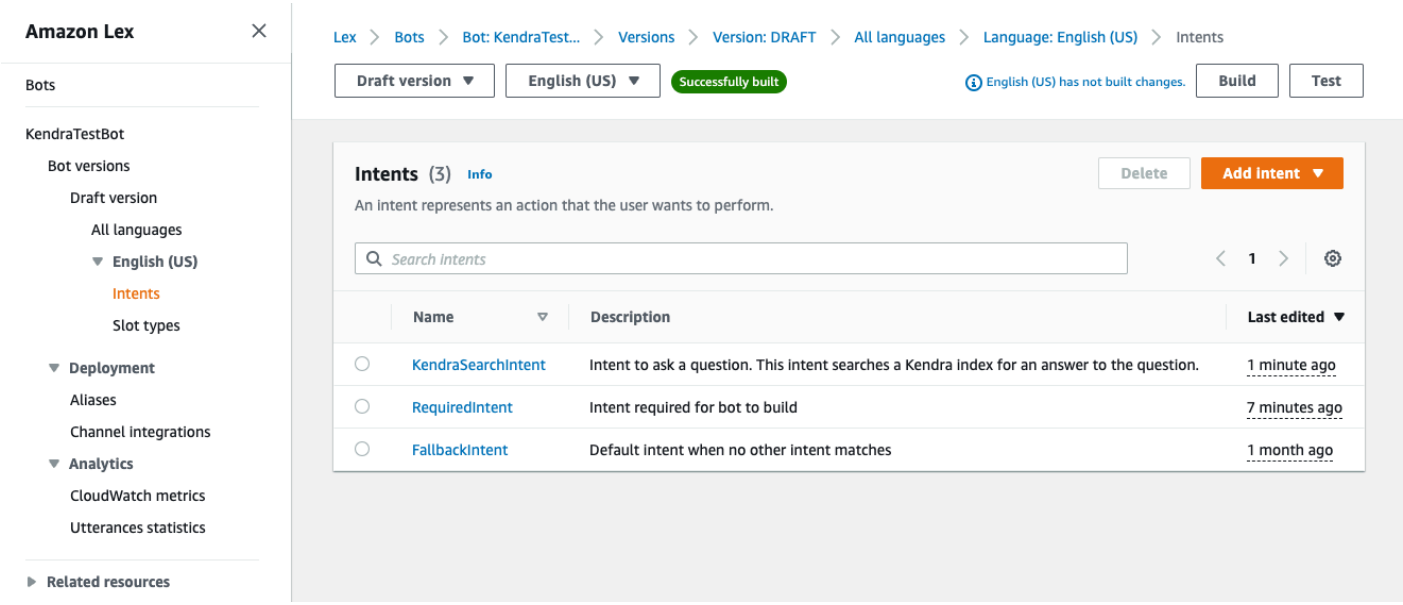
在 Amazon Lex V2 将查询发送到 Amazon Kendra 后，查询响应将返回到 AMAZON.KendraSearchIntent 履行 Lambda 函数。代码挂钩的输入事件包含来自 Amazon Kendra 的完整响应。查询数据与 Amazon Kendra Query 操作返回的结构相同。有关更多信息，请参阅《Amazon Kendra 开发人员指南》中的查询响应语法。

实现代码挂钩是可选的。如果代码挂钩不存在，或者代码挂钩未在响应中返回消息，则 Amazon Lex V2 将对响应使用 closingResponse 语句。

示例：为 Amazon Kendra 索引创建常见问题机器人

此示例创建一个使用 Amazon Kendra 索引为用户的问题提供答案的 Amazon Lex V2 机器人。常见问题解答自动程序为用户管理对话。它使用 AMAZON.KendraSearchIntent 意图查询索引并向用户提供响应。以下将简要地介绍使用 Amazon Kendra 索引创建常见问题机器人的方法：

1. 创建一个自动程序，您的客户将与其交互以从其获取答案。
2. 创建自定义意图。由于 AMAZON.KendraSearchIntent 和 AMAZON.FallbackIntent 是替代意图，因此您的机器人需要至少一个其他意图，该至少一个其他意图中必须包含至少一个言语。此意图使您能够构建自动程序，但不用于其他方面。因此，您的常见问题机器人将包含至少三个意图，如下图所示：



3. 将 AMAZON.KendraSearchIntent 意图添加到机器人中，并将其配置为与 Amazon Kendra 索引配合使用。
4. 测试该机器人，即执行查询并验证您的 Amazon Kendra 索引的结果是否为回答查询的文档。

先决条件

在使用此示例之前，您需要创建 Amazon Kendra 索引。有关更多信息，请参阅《Amazon Kendra 开发人员指南》中的 Amazon Kendra 控制台入门。在本示例中，选择示例数据集（AWS 示例文档）作为您的数据来源。

要创建常见问题机器人，请执行以下操作：

1. 登录 AWS Management Console 并打开 Amazon Lex 控制台，网址为 https://console.aws.amazon.com/lex/。
2. 在导航窗格中，选择自动程序。
3. 选择创建机器人。
   a. 对于创建方法，请选择创建空白机器人。
   b. 在机器人配置部分中，指定指示机器人用途的机器人名称（例如 **KendraTestBot**），并且指定描述（可选）。该名称在您的账户中必须是唯一的。
   c. 在 IAM 权限部分中，选择使用基本 Amazon Lex 权限创建角色。此操作将创建一个 AWS Identity and Access Management (IAM) 角色，该角色具有 Amazon Lex V2 运行您的机器人所需的权限。
   d. 在儿童在线隐私保护法 (COPPA) 部分中，选择否。
   e. 在空闲会话超时和高级设置部分中，保留默认值并且选择下一步。
   f. 此时，系统将转至为机器人添加语言部分。在语音交互下的菜单中，选择无。这只是一个基于文本的应用程序。对于其余字段保留默认设置。
   g. 选择完成。Amazon Lex V2 会创建您的机器人和一个名为的默认意图 NewIntent，然后将您带到配置此意图的页面

要成功构建机器人，您必须创建独立于 `AMAZON.FallbackIntent` 和 `AMAZON.KendraSearchIntent` 的至少一个意图。该意图是构建 Amazon Lex V2 机器人所必需的，但不用于常见问题响应。该意图必须包含至少一个示例言语，并且该言语不得适用于客户提出的任何问题。

要创建所需的意图，请执行以下操作：

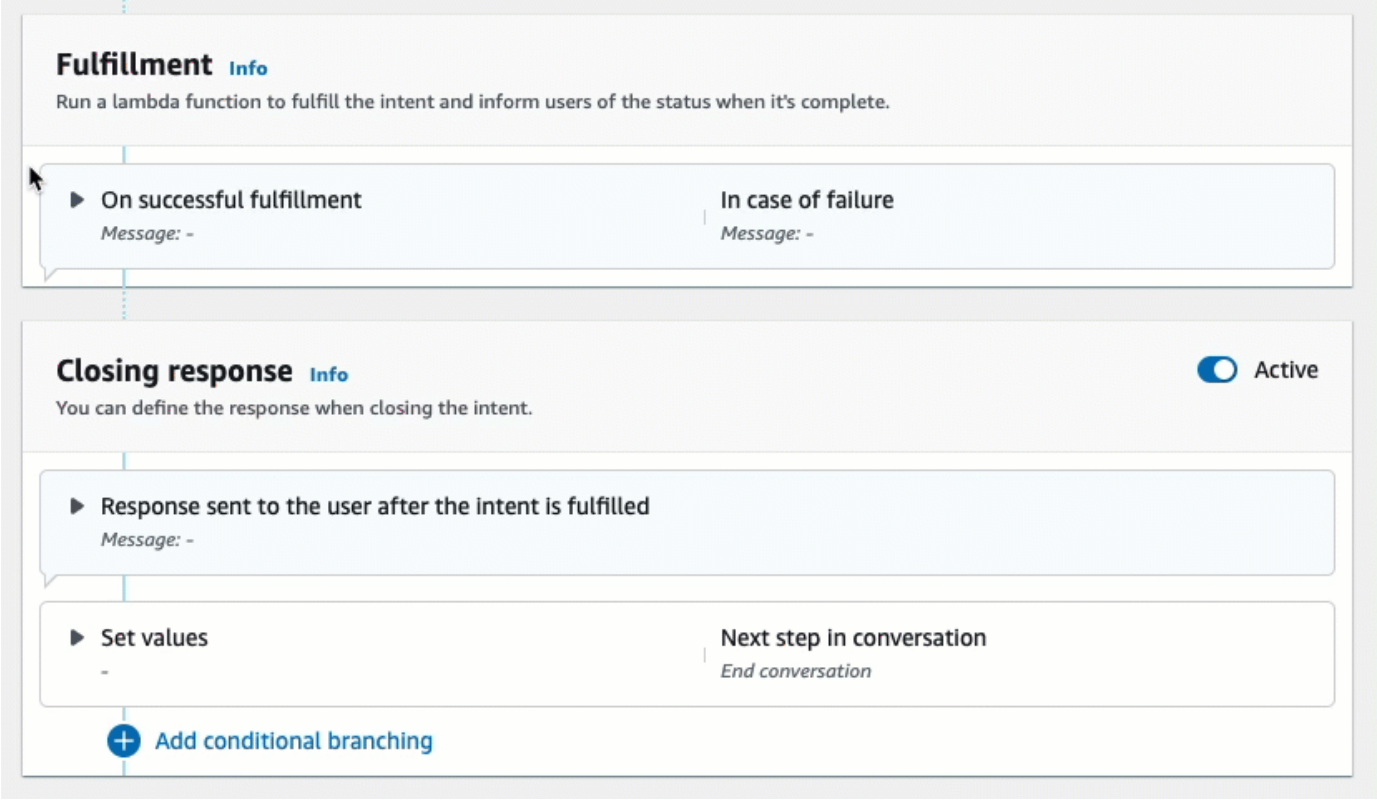
1. 在意图详细信息部分中，指定意图的名称，例如 **RequiredIntent**。
2. 在示例言语部分，在添加言语旁边的框中键入言语，例如 **Required utterance**。然后选择添加言语。
3. 选择保存意图。

创建搜索 Amazon Kendra 索引的意图以及该意图应返回的响应消息。

创建亚马逊。 KendraSearchIntent 意图和响应消息：

1. 在导航窗格中选择返回意图列表，以返回到机器人的意图页面。选择添加意图，然后从下拉菜单中选择使用内置意图。
2. 在弹出的框中，选择内置意图下的菜单。在搜索栏中输入 **AMAZON.KendraSearchIntent**，然后在列表中将其选中。
3. 指定该意图的名称，例如 **KendraSearchIntent**。
4. 从 Amazon Kendra 索引下拉菜单中，选择您需要通过该意图搜索的索引。您在先决条件部分中创建的索引应为可用状态。
5. 选择添加。
6. 在意图编辑器中，向下滚动至履行部分，选择向右箭头展开该部分，然后在成功履行时下的框中添加以下消息：

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



有关 Amazon Kendra 搜索响应的更多信息，请参阅使用搜索响应。

7. 选择保存意图，然后选择构建以构建自动程序。机器人准备就绪后，屏幕顶部的横幅会变为绿色并显示成功消息。

最后，使用控制台测试窗口来测试来自自动程序的响应。

要测试常见问题机器人，请执行以下操作：

1. 成功构建机器人后，选择测试。
2. 在控制台测试窗口中输入 **`What is Amazon Kendra?`**。验证机器人是否使用链接进行响应。
3. 有关配置的更多信息`AMAZON.KendraSearchIntent`，请参阅AMAZON.KendraSearchIntent和KendraConfiguration。

AMAZON.PauseIntent

对允许用户暂停与机器人的交互以便稍后返回到该交互的单词和短语的响应。您的 Lambda 函数或应用程序需要将意图数据保存在会话变量中，或者在恢复当前意图时，您需要使用 GetSession 操作来检索意图数据。

常见言语：

- 暂停
- 暂停一下

AMAZON.QnAIntent

> **Note**
>
> 在利用生成式人工智能功能之前，您必须满足以下先决条件
>
> 1. 导航到 Amazon Bedrock 控制台并注册您打算使用的 Anthropic Claude 模型的访问权限（有关更多信息，请参阅模型访问权限）。有关使用 Amazon Bedrock 的定价信息，请参阅 Amazon Bedrock 定价。
> 2. 为机器人区域设置开启生成式人工智能功能。为此，请按照利用生成式人工智能优化机器人的创建和性能中的步骤进行操作。

通过使用 Amazon Bedrock FM 搜索和汇总常见问题回复来回答客户问题。当某个言语未被归类为机器人中存在的任何其他意图时，就会激活该意图。请注意，引发槽位值时，不会因为错过的言语而激活该意图。AMAZON.QnAIntent 被识别后，将使用指定的 Amazon Bedrock 模型搜索已配置的知识库并回答客户的问题。

如果 FM 的响应不令人满意，或者调用 FM 失败，Amazon Lex V2 就会调用 AMAZON.FallbackIntent。

> **Warning**
>
> 不能在同一个机器人区域设置中使用 AMAZON.QnAIntent 和 AMAZON.KendraSearchIntent。

可使用以下知识库选项。您必须已创建该知识库并为其中的文档编制了索引。

- OpenSearch 服务域-包含已编入索引的文档。要创建域名，请按照创建和管理亚马逊 OpenSearch 服务域中的步骤进行操作。
- Amazon Kendra 索引 – 包含已编入索引的常见问题文档。要创建 Amazon Kendra 索引，请按照创建索引中的步骤进行操作。
- Amazon Bedrock 知识库 – 包含已编入索引的数据源。要设置知识库，请按照构建知识库中的步骤进行操作。

如果选择该意图，则需要配置以下字段，然后选择添加以添加该意图。

- Bedrock 模型 – 选择要用于该意图的提供商和基础模型。目前支持 Anthropic Claude V2 和 Antropic Claude Instant。
- 知识库 – 选择您希望模型从中提取信息以回答客户问题的来源。以下是可用的来源。
  - OpenSearch— 配置以下字段。
    - 域端点 – 提供您为域创建的域端点或在创建域之后提供给您的域端点。
    - 索引名称 – 提供要搜索的索引。有关更多信息，请参阅在 Amazon OpenSearch 服务中为数据编制索引。
    - 选择向客户返回响应的方式。
      - 确切响应 – 启用此选项后，“答案”字段中的值将按原样用于机器人响应。配置的 Amazon Bedrock 基础模型用于按原样选择确切的答案内容，无需进行任何内容合成或总结。指定在 OpenSearch 数据库中配置的问答字段的名称。

    - 包含字段：返回模型使用您指定的字段生成的答案。最多指定在 OpenSearch 数据库中配置的五个字段的名称。使用分号 (;) 分隔字段。
- Amazon Kendra：配置以下字段。
    - Amazon Kendra 索引：选择您希望机器人搜索的 Amazon Kendra 索引。
    - Amazon Kendra 筛选条件：要创建筛选条件，请选中此复选框。有关 Amazon Kendra 搜索筛选条件 JSON 格式的更多信息，请参阅 Using document attributes to filter search results。
    - 确切响应：要让您的机器人返回 Amazon Kendra 返回的确切响应，请选中此复选框。否则，您选择的 Amazon Bedrock 模型会根据结果生成响应。

      > **Note**
      >
      > 要使用该功能，必须先按照 Adding frequently asked questions (FAQs) to an index 中的步骤向索引添加常见问题。
- Amazon Bedrock 知识库：如果选择此选项，请指定知识库的 ID。您可以通过在控制台中查看知识库的详细信息页面或发送GetKnowledgeBase请求来找到 ID。

来自 QnAIntent 的响应将存储到请求属性中，如下所示：

- `x-amz-lex:qnA-search-response`：QnAIntent 对问题或言语的响应。
- `x-amz-lex:qnA-search-response-source`：指向用于生成响应的文档或文档列表。

AMAZON.RepeatIntent

对允许用户重复上一条消息的单词和短语的响应。您的应用程序需要使用 Lambda 函数将之前的意图信息保存在会话变量中，或者您需要使用 GetSession 操作来获取之前的意图信息。

常见言语：

- 重复
- 再说一遍
- 重复一遍

AMAZON.ResumeIntent

对允许用户恢复上一个已暂停意图的词语和短语的响应。您的 Lambda 函数或应用程序必须管理恢复上一个意图所需的信息。

常见言语：

- 继续
- 继续
- 继续下去

AMAZON.StartOverIntent

对允许用户停止处理当前意图并从头开始的词语和短语的响应。您可以使用 Lambda 函数或 `PutSession` 操作再次引发第一个插槽。

常见言语：

- 从头开始
- 重新开始
- 再次开始

AMAZON.StopIntent

对表示用户想要停止处理当前意图并结束与机器人的交互的词语和短语的响应。您的 Lambda 函数或应用程序应清除所有现有属性和插槽类型值，然后结束该交互。

常见言语：

- stop
- off
- 闭嘴

添加槽类型

槽位类型定义了用户可以为您的意图变量提供的值。您可以为每种语言定义槽位类型，以便这些值是特定于该语言的。例如，对于列出油漆颜色的槽位类型，可以包括英语值“red”、法语值“rouge”和西班牙语值“rojo”。

本主题介绍如何创建自定义槽位类型，用于为您的意图槽位提供值。您也可以为标准值使用内置槽位类型。例如，您可以使用内置槽位类型 AMAZON.Country 来获取全球各国家/地区列表。

要创建槽位类型，请执行以下操作：

1. 登录到 AWS Management Console，并通过以下网址打开 Amazon Lex 控制台：https://console.aws.amazon.com/lex/。
2. 从机器人列表中选择要添加语言的机器人，然后选择对话结构，然后选择所有语言。
3. 选择要添加槽位类型的语言，然后选择槽位类型。
4. 选择添加槽位类型，指定槽位类型的名称，然后选择添加。
5. 在槽位类型编辑器中，添加您的槽位类型的详细信息。
   - 槽位值解析：确定槽位值的解析方式。如果您选择扩展值，Amazon Lex V2 将使用这些值作为代表值来进行训练。如果您使用限制为槽位值，则该槽位的允许值仅限于您提供的值。
   - 槽位类型值：该槽位的值。如果选择限制为槽位值，则可以为该值添加同义词。例如，您可以为值“football”添加同义词“soccer”。如果用户在与您的机器人的对话中输入“soccer”，则该槽位的实际值为“football”。
   - 使用槽位值作为自定义词汇：启用此选项可帮助提高对音频对话中槽位值和同义词的识别。当槽位值是常用术语，例如“是”、“否”、“一”、“二”、“三”等时，请勿启用此选项。
6. 选择保存槽位类型。

Amazon Lex V2 提供以下槽位类型：

主题

内置槽位类型

Amazon Lex 支持内置插槽类型，用于定义如何识别和处理插槽中的数据。您可以在您的意图中创建这些类型的槽。因此，无需为常用槽数据 (如日期、时间和位置) 创建枚举值。内置槽类型没有版本。

槽类型	简短描述	支持的区域设置
亚马逊。 AlphaNumeric	识别由字母和数字组成的单词。	除韩语 (ko-KR) 之外的所有区域设置
AMAZON.City	识别代表城市的词语。	所有区域设置
AMAZON.Confirmation	识别表示“是”、“否”、“也许”和“不知道”的单词，并将其转换为标准（是/否/可能/不知道）格式。	英语（en-US、en-GB、en-AU、en-IN、en-ZA）
AMAZON.Country	识别代表国家/地区的单词。	所有区域设置
AMAZON.Date	识别代表日期的词语并将其转换为标准格式。	所有区域设置
AMAZON.Duration	识别代表持续时间的词语并将其转换为标准格式。	所有区域设置
亚马逊。 EmailAddress	识别表示电子邮件地址的词语并将其转换为标准电子邮件地址。	所有区域设置
亚马逊。 FirstName	识别代表名字的词语。	所有区域设置
亚马逊。 LastName	识别代表姓氏的词语。	所有区域设置
AMAZON.Number	识别数字词语并将其转换为数字。	所有区域设置

槽类型	简短描述	支持的区域设置
AMAZON.Percentage	识别表示百分比的词语并将其转换为一个数字和一个百分比符号 (%)。	所有区域设置
亚马逊。 PhoneNumber	识别表示电话号码的词语并将其转换为数字字符串。	所有区域设置
AMAZON.State	识别代表省/市/自治区的单词。	所有区域设置
亚马逊。 StreetName	识别代表街道名称的单词。	所有区域设置
AMAZON.Time	识别指示时间的单词并将其转换为时间格式。	所有区域设置
AMAZON.UKPostalCode	识别代表英国邮政编码的单词并将其转换为标准格式。	仅限英语（英国）(en-GB)
亚马逊。 FreeFormInput	识别由任何单词或字符组成的字符串。	所有区域设置

亚马逊。 AlphaNumeric

识别由字母和数字组成的字符串，例如 **APQ123**。

此插槽类型在韩语 (ko-KR) 区域设置中不可用。

您可以对包含以下内容的字符串使用 `AMAZON.AlphaNumeric` 槽类型：

- 字母字符，例如 **ABC**
- 数字字符，例如 **123**
- 字母数字字符的组合，例如 **ABC123**

AMAZON.AlphaNumeric 槽位类型支持使用拼写样式进行输入。您可以使用 spell-by-letter 和 spell-by-word 样式来帮助客户输入字母。有关更多信息，请参阅使用拼写样式捕获槽位值。

您可以向 AMAZON.AlphaNumeric 槽类型添加正则表达式以验证为槽输入的值。例如，您可以使用正则表达式来验证：

- 加拿大邮政编码
- 驾照编号
- 车辆识别号码

使用标准正则表达式。Amazon Lex V2 支持在正则表达式中使用以下字符：

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

Amazon Lex V2 还支持正则表达式中的 Unicode 字符。格式为 \u*Unicode*。使用四位数表示 Unicode 字符。例如，[\u0041-\u005A] 等同于 [A-Z]。

不支持以下正则表达式运算符：

- 无限重复符：*、+ 或 {x,}，无上限。
- 通配符（.）

正则表达式的最大长度为 300 个字符。存储在使用正则表达式的 AMAZON.AlphaNumeric 槽位类型中的字符串的最大长度为 30 个字符。

以下是一些示例正则表达式。

- 字母数字字符串，例如 **APQ123** 或 **APQ1**：[A-Z]{3}[0-9]{1,3} 或更受约束的 [A-DP-T]{3}[1-5]{1,3}
- 美国邮政服务优先邮件国际格式，例如 **CP123456789US**：CP[0-9]{9}US
- 银行汇款路径号码，例如 **123456789**：[0-9]{9}

要为槽类型设置正则表达式，请使用控制台或 CreateSlotType 操作。保存槽类型时验证正则表达式。如果表达式无效，Amazon Lex V2 将返回错误消息。

在槽位类型中使用正则表达式时，Amazon Lex V2 会根据正则表达式检查该类型的槽位的输入。如果输入与表达式匹配，则接受槽的值。如果输入不匹配，Amazon Lex V2 提示用户重复输入。

AMAZON.City

提供本地和世界城市列表。槽位类型可以识别城市名称的常见变体。Amazon Lex V2 不将变体转换为官方名称。

示例：

- New York
- 雷克雅未克
- 东京
- Versailles

AMAZON.Confirmation

此槽位类型可识别与 Amazon Lex V2 的“是”、“否”、“也许”和“不知道”短语和单词相对应的输入短语，并将其转换为四个值之一。该槽位类型可用于捕获用户的确认或认可。根据最终解析的值，您可以创建条件来设计多个对话路径。

例如：

如果 {confirmation} = "Yes"，则履行该意图

否则，引发另一个槽位

示例：

- 是：是的、是、好吧、当然、我明白了、我可以同意...
- 不：不、否、不行、算了吧、我拒绝、不可能...
- 也许：有可能、也许、有时、我可能会、这可能是对的.....
- 不知道：不了解、未知、不知道、不确定、谁知道.....

自 2023 年 8 月 17 日起，如果现有名为“确认”的自定义槽位类型，则必须更改名称以避免与内置槽位确认发生冲突。在 Lex 控制台的左侧导航栏中，转到槽位类型（适用于名为“确认”的现有自定义槽位类型），然后更新槽位类型名称。新的槽位类型名称不得为“确认”，这是为内置确认槽位类型预留的关键字。

AMAZON.Country

世界各个国家/地区的名字。示例：

- 澳大利亚
- 德国
- 日本
- 美国
- 乌拉圭

AMAZON.Date

将表示日期的词语转换为日期格式。

日期以 ISO-8601 日期格式提供给您的意图。您的意图在插槽中收到的日期可能会有所不同，具体取决于用户说出的特定短语。

- 映射到特定日期的言语（例如“今天”、“现在”或“11 月 25 日”）会转换为完整的日期：2020-11-25。默认为当前日期或之后的日期。
- 映射到未来特定星期的言语（例如“下周”）会转换为本周最后一天的日期。在 ISO-8601 格式中，一周的第一天是星期一，最后一天是星期日。例如，如果今天是 2020 年 11 月 25 日，则“下周”会转换为 2020-11-29。映射到本周或上一周的日期会转换为该周的第一天。例如，如果今天是 2020 年 11 月 25 日，则“上周”会转换为 2020-11-16。
- 映射到未来某个月但不是特定日期（例如“下个月”）的言语会转换为该月的最后一天。例如，如果今天是 2020 年 11 月 25 日，则“下个月”会转换为 2020-12-31。映射到本月或上个月的日期会转换为该月的第一天。例如，如果今天是 2020 年 11 月 25 日，则“本月”映射到 2020-11-01。
- 映射到未来某一年但不是特定月份或日期（例如“下一年”）的言语会转换为该下一年的最后一天。例如，如果今天是 2020 年 11 月 25 日，则“下一年”会转换为 2021-12-31。映射到今年或去年的日期会转换为今年的第一天。例如，如果今天是 2020 年 11 月 25 日，则“上一年”会转换为 2019-01-01。

AMAZON.Duration

将表示持续时间的词语转换为数字持续时间。

持续时间被解析为基于 ISO-8601 持续时间格式的格式 PnYnMnWnDTnHnMnS。P 表示这是持续时间，n 是数值，n 后面的大写字母是特定的日期或时间元素。例如，P3D 表示 3 天。T 用于表示其余值代表时间元素而不是日期元素。

示例：

- “十分钟”：PT10M
- “五个小时”：PT5H
- “三天”：P3D
- “四十五秒”：PT45S
- “八周”：P8W
- “七年”：P7Y
- “五小时十分钟”：PT5H10M
- “两年三小时十分钟”：P2YT3H10M

亚马逊。 EmailAddress

识别表示以 username@domain 形式提供的电子邮件地址的单词。地址中的用户名可以包括下列特殊字符：下划线 (_)、连字符 (-)、句点 (.) 和加号 (+)。

AMAZON.EmailAddress 槽位类型支持使用拼写样式进行输入。您可以使用 spell-by-letter 和 spell-by-word 样式来帮助客户输入电子邮件地址。有关更多信息，请参阅使用拼写样式捕获槽位值。

亚马逊。 FirstName

常用的名字。该槽位类型可以识别正式姓名、非正式昵称以及由多个单词组成的姓名。发送给您的意图的名称即为用户发送的值。Amazon Lex V2 不会将昵称转换为正式姓名。

对于听起来相似但拼写不同的名字，Amazon Lex V2 会向您的意图发送一份通用表格。

AMAZON.FirstName 槽位类型支持使用拼写样式进行输入。您可以使用 spell-by-letter 和 spell-by-word 样式来帮助客户输入姓名。有关更多信息，请参阅使用拼写样式捕获槽位值。

示例：

- Emily
- John

- Sophie
- Anil Kumar

亚马逊。 FirstName 还会根据原始值返回一个紧密相关的名称列表。您可以使用已解析值列表来恢复拼写错误、向用户确认姓名或在用户目录中执行数据库查找有效名称。

例如，输入“John”可能会返回其他相关姓名，例如“John J”和“John-Paul”。

以下显示了 AMAZON.FirstName 内置槽位类型的响应格式：

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

亚马逊。 LastName

常用的姓氏。对于听起来相似但拼写不同的姓名，Amazon Lex V2 会向您的意图发送一份通用表格。

AMAZON.LastName 槽位类型支持使用拼写样式进行输入。您可以使用 spell-by-letter 和 spell-by-word 样式来帮助客户输入姓名。有关更多信息，请参阅使用拼写样式捕获槽位值。

示例：

- 布罗斯基
- 达舍
- 埃弗斯
- 帕雷斯
- Welt

亚马逊。 LastName 还会根据原始值返回一个紧密相关的名称列表。您可以使用已解析值列表来恢复拼写错误、向用户确认姓名或在用户目录中执行数据库查找有效名称。

例如，输入“Smith”可能会返回其他相关姓名，例如“Smyth”和“Smithe”。

以下显示了 AMAZON.LastName 内置槽位类型的响应格式：

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

AMAZON.Number

将表示数值的词语或数字转换为数字，包括十进制数字。下表介绍 AMAZON.Number 槽类型如何捕获数字词汇。

输入	响应
一百二十三点四五	123.45
一二三点四五	123.45
零点四二	0.42
点四二	0.42
232.998	232.998
50	50
-15	-15
负 15	-15

AMAZON.Percentage

将表示百分比的词汇和符号转换为一个带百分比符号 (%) 的数字值。

如果用户输入不带百分号或“百分之”这个词的数字，则槽值将设置为该数字。下表介绍 AMAZON.Percentage 槽类型如何捕获百分比。

输入	响应
百分之 50	50%
百分之 0.4	0.4%
23.5%	23.5%
百分之二十五	25%

亚马逊。 PhoneNumber

如下将表示电话号码的数字或词转换为不含标点符号的字符串格式。

Type	描述	输入	结果
前面带加号 (+) 的国际号码	前面带加号 (+) 的 11 位数字。	+61 7 4445 1061 +1 (509) 555-1212	+61744431061 +15095551212
前面不带加号 (+) 的国际号码	前面不带加号 (+) 的 11 位数字	1 (509) 555-1212 61 7 4445 1061	15095551212 61744451061
国家/地区代码	不带国际代码的 10 位数字	(03) 5115 4444 (509) 555-1212	0351154444 5095551212
本地号码	不带国际代码或区号的电话号码	555-1212	5551212

AMAZON.State

国家/地区内部的地理和政治区域的名称。

示例：

- 巴伐利亚
- 福岛县
- 太平洋西北地区
- 昆士兰
- 威尔士

亚马逊。 StreetName

典型街道地址内的街道名称。这只包括街道名称，不包括门牌号码。

示例：

- 堪培拉大道
- 前街
- 市场路

AMAZON.Time

将表示时间的单词转换为时间值。AMAZON.Time 可以解析确切的时间、不明确的值和时间范围。该槽位值可以解析为以下时间范围：

- AM
- PM
- MO（上午）
- AF（下午）
- EV（傍晚）
- NI（晚上）

当用户输入不明确的时间时，Amazon Lex V2 使用 Lambda 事件的 slots 属性将不确定的时间的解析传递给 Lambda 函数。例如，如果自动程序提示用户输入交付时间，用户可能以“10 点钟”作为响应。这样的时间是不明确的。它可能指上午 10:00 或晚上 10:00。这种情况下，interpretedValue 字段中的值为 null，而 resolvedValues 字段包含对该时间的两种可能的解析。Amazon Lex V2 将以下内容输入 Lambda 函数：

```
"slots": {
```

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

当用户以明确的时间作为响应时，Amazon Lex V2 在 Lambda 事件的 `slots` 属性的 `interpretedValue` 字段中将该时间发送到 Lambda 函数。例如，如果用户在收到输入交付时间的提示时，以“早上 10:00”作为响应，则 Amazon Lex V2 将以下内容输入 Lambda 函数：

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

例如，如果用户在收到输入交付时间的提示时，以“上午”作为响应，则 Amazon Lex V2 将以下内容输入 Lambda 函数：

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

有关 Amazon Lex V2 发送到 Lambda 函数的数据的更多信息，请参阅[解释输入事件格式](#)。

AMAZON.UK PostalCode

将代表英国邮政编码的单词转换为英国邮政编码的标准格式。AMAZON.UKPostalCode 槽位类型会验证邮政编码并将其解析为一组标准化格式，但不会检查邮政编码是否有效。您的应用程序必须验证邮政编码。

AMAZON.UKPostalCode 槽位类型仅在英语（英国）（en-GB）区域设置中可用。

AMAZON.UKPostalCode 槽位类型支持使用拼写样式进行输入。您可以使用 spell-by-letter 和 spell-by-word 样式来帮助客户输入字母。有关更多信息，请参阅使用拼写样式捕获槽位值。

插槽类型只能识别下面列出的英国使用的有效邮政编码格式。有效的格式为（“A” 代表字母，“9” 代表数字）：

- AA9A 9AA
- A9A 9AA
- A9 9AA
- A99 9AA
- AA9 9AA
- AA99 9AA

对于文本输入，用户可以输入大小写字母的任意组合。用户可以在邮政编码中使用或省略空格。解析后的值将始终包括位于邮政编码中正确位置处的空格。

对于口述输入，用户可以说出单个字符，也可以使用双字母发音，例如“两个 A”或“两个 9”。用户还可以使用两位数的发音，例如“九十九”代表“99”。

> **Note**
>
> 并非所有英国邮政编码都能被识别。仅支持上面列出的格式。

亚马逊。 FreeFormInput

AMAZON.FreeFormInput 可用于捕获最终用户的自由格式输入。识别由单词或字符组成的字符串。解析的值是整个输入言语。

例如：

机器人：请根据您的通话体验提供反馈。

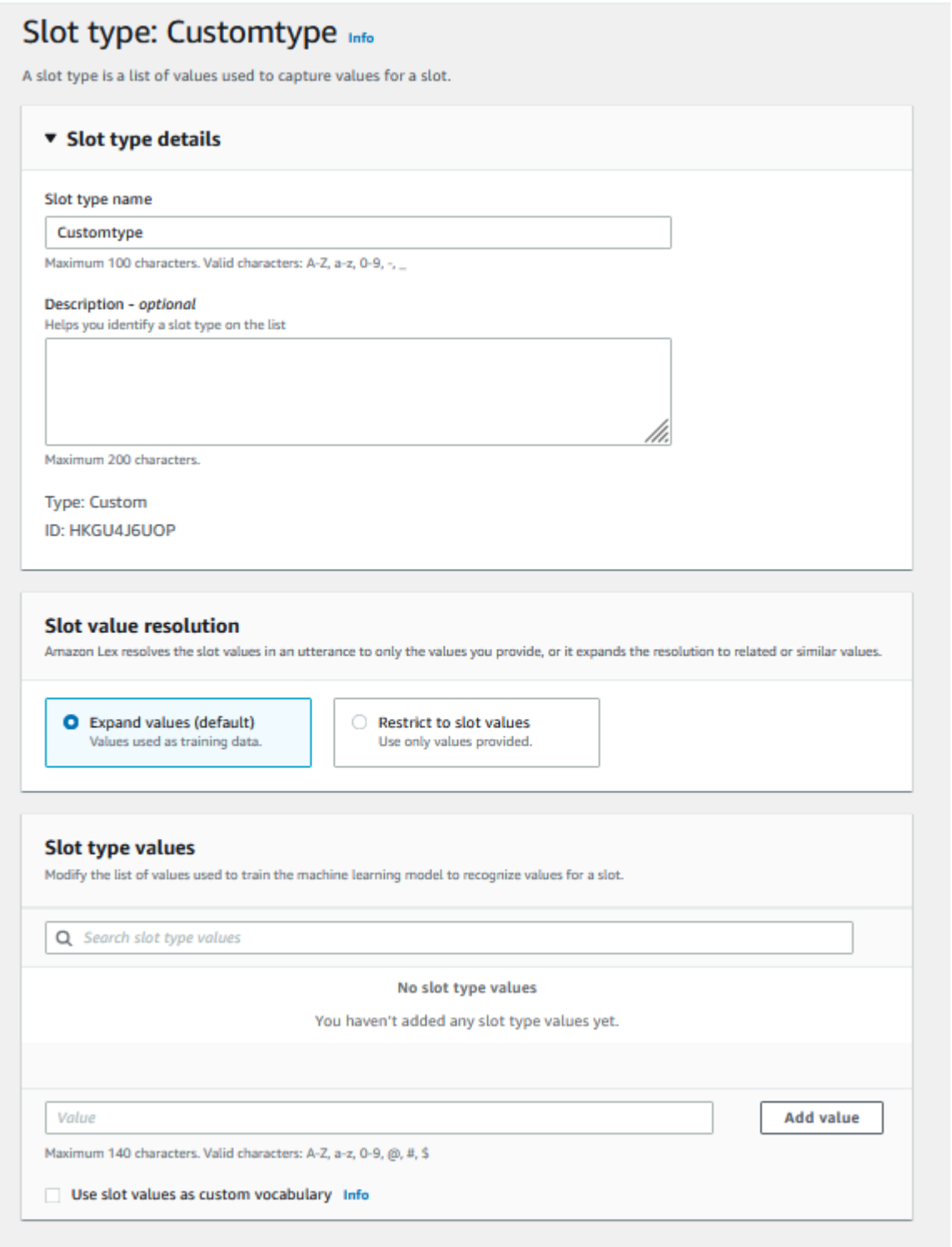
用户：我的所有问题都得到了解答，并且交易已完成。

注意：

- AMAZON.FreeFormInput 可用于捕获最终用户的原样自由格式输入。
- AMAZON.FreeFormInput 无法用于示例意图言语中。
- AMAZON.FreeFormInput 无法具有示例槽位言语。
- AMAZON.FreeFormInput 只有在被引发时才会被识别。
- AMAZON.FreeFormInput 不支持等待并继续。
- AMAZON.FreeFormInput 在 Amazon Connect 聊天频道中目前不支持。
- 当 AMAZON.FreeFormInput 槽位被引发时，FallbackIntent 不会被触发。
- 当 AMAZON.FreeFormInput 槽位被引发时，将不会执行意图切换。

自定义槽位类型

对于每个目的，您都可以指定参数来指示目的要完成用户请求所需的信息。这些参数，或者说槽，都有一个类型。槽位类型是指 Amazon Lex V2 用于训练机器学习模型以识别槽位值的值列表。例如，您可以定义一个名为 Genres 的槽位类型，其值为“comedy”（喜剧）、“adventure”（探险剧）、“documentary”（纪录片）等。您可以定义槽位类型值的同义词。例如，可以为“comedy”值定义同义词“funny”和“humorous”。



或者，您可以通过配置槽位类型来扩展槽位值。槽位值将用作训练数据，并且当用户提供的值与槽位值以及这些值的同义词相似时，该模型将槽位解析为用户提供的值。这是默认行为。Amazon Lex V2 为槽位维护一个可能解析值的列表。列表中的每个条目都提供一个解析值，Amazon Lex V2 将其识别为槽位的更多可能值。解析值是与槽位值的尽可能匹配。该列表最多包含五个值。

或者，您可以通过配置槽位类型来限制对槽位值的解析。在这种情况下，只有当用户输入的槽位值与该槽位值相同或是该槽位值的同义词时，该模型才会将该槽位值解析为现有的槽位值。例如，如果用户输入“funny”，会解析为槽值“comedy”。

当用户输入的值是槽位类型值的同义词时，模型将返回该槽位类型值作为 `resolvedValues` 列表中的第一个条目。例如，如果用户输入“funny”（滑稽），则模型会在 `originalValue` 字段中填充值“funny”，在 resolvedValues 字段的第一个条目中填充“comedy”（喜剧）。您可以在使用 CreateSlotType 操作创建或更新槽类型时配置 `valueSelectionStrategy`，以便使用解析列表中的第一个值填充槽值。

自定义槽位类型支持使用拼写样式进行输入。您可以使用 spell-by-letter 和 spell-by-word 样式来帮助客户输入字母。有关更多信息，请参阅使用拼写样式捕获槽位值。

如果您使用 Lambda 函数，该函数的输入事件中会包含一个名为 `resolvedValues` 的解析列表。以下示例显示了 Lambda 函数输入的槽位部分：

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

对于每个槽类型，最多可定义 10000 个值和同义词。每个自动程序的槽类型值和同义词的总数最多为 50000。例如，您可以拥有 5 种插槽类型，每种类型包含 5,000 个值和 5,000 个同义词，或者您可以拥有 10 种插槽类型，每种类型包含 2,500 个值和 2,500 个同义词。

自定义槽位类型不应与内置槽位类型同名。例如，不应使用预留关键字“日期”、“数字”或“确认”来命名自定义槽位类型。这些关键字是为内置槽位类型预留的。有关所有内置槽位类型的列表，请参阅内置槽位类型。

语法插槽类型

使用语法插槽类型，您可以按照 SRGS 规范以 XML 格式编写自己的语法，以便在对话中收集信息。Amazon Lex V2 可以识别符合语法中指定规则的言语。您还可以在语法文件中使用 ECMAScript

标签提供语义解释规则。然后，当出现匹配时，Amazon Lex 会将标签中设置的属性作为已解析值返回。

您只能在英语（澳大利亚）、英语（英国）和英语（美国）区域设置中创建语法插槽类型。

语法插槽类型分为两个部分。首先是使用 SRGS 规范格式编写的语法本身。该语法解释用户的言语。如果该言语被语法所接收，该言语是匹配的，否则就会被拒绝。如果匹配了某个言语，则该言语将传递给脚本（如有）。

第二个是语法插槽类型的一部分，是用 ECMAScript 编写的可选脚本，将输入转换为插槽类型返回的解析值。例如，您可以使用脚本将口述数字转换为数字值。ECMAScript 语句的前后以 <tag> 元素标记。

以下示例依据 SRGS 规范采用 XML 格式，其中显示了 Amazon Lex V2 接受的有效语法。该语法定义了一种接受卡号，并确定这些卡号是用于普通账户还是高级账户的语法插槽类型。有关可接受句法的更多信息，请参阅以下主题：语法定义和脚本格式。

```
<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>
```

上述语法仅接受两种类型的卡号：7241 或 7001。这两者都可以选择以“卡号”为前缀。它还包含可用于语义解释的 ECMAScript 标签。通过语义解释，言语“卡号七二四一”将返回以下对象：

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

此对象在 RecognizeText、RecognizeUtterance 和 StartConversation 操作返回的 `resolvedValues` 对象中作为 JSON 序列化字符串返回。

添加语法插槽类型

要添加语法插槽类型，请执行以下操作：

1. 将插槽类型的 XML 定义上传到 S3 存储桶。记下存储桶名称和文件路径。

   > **Note**
   >
   > 最大文件大小为 100 KB。

2. 登录到 AWS Management Console，并通过以下网址打开 Amazon Lex 控制台：https://console.aws.amazon.com/lex/。
3. 从左侧菜单中选择机器人，然后选择要添加语法插槽类型的机器人。
4. 选择查看语言，然后选择要添加语法插槽类型的语言。
5. 选择查看插槽类型。
6. 选择添加插槽类型，然后选择添加语法插槽类型。
7. 指定插槽类型的名称，然后选择添加。
8. 选择包含定义文件的 S3 存储桶，然后输入文件路径。选择保存插槽类型。

语法定义

本主题显示了 Amazon Lex V2 支持的 SRGS 规范的各个部分。所有规则均在 SRGS 规范中定义。有关更多信息，请参阅《W3C 正式推荐标准》中的1.0 版语音识别语法规范。

主题

- 标头声明

- 支持的 XML 元素
- 令牌
- 规则引用
- 序列和封装
- 重复
- 语言
- 标签
- Weight

本文档包括复制和衍生自 W3C 的 1.0 版语音识别语法规范（参见 https://www.w3.org/TR/speech-grammar/）的材料。引文信息如下：

SRGS 规范文档，即 W3C 正式推荐标准，可通过以下许可从 W3C 获得。

许可证文本

许可证

使用和/或复制本文档或与本声明相链接的 W3C 文档，即表示您（被许可人）认可您已阅读、理解并将遵守以下条款和条件：

特此允许出于任何目的以任何方式复制和分发本文档或与本声明相链接的 W3C 文档的内容，且无需支付任何费用或特许权使用费，前提是您在使用的文档的所有副本或其任何部分中包含以下内容：

- 指向原始 W3C 文档的链接或网址。
- 原作者的先前版权声明，或者如果该等先前版权声明不存在，则为以下形式的声明（首选超文本，但允许使用文本表示）：“版权所有 © [$文件日期] 万维网联合会（MIT、ERCIM、Keio、Beihang）。http://www.w3.org/Consortium/Legal/2015/doc-license”
- （如果存在）W3C 文档的状态。

如果篇幅允许，应提供本声明的全文。我们要求您在实施本文档或其任何部分内容时创建的任何软件、文档或其他项目或产品中提供作者身份归属。

本许可不授予对 W3C 文档进行修改或衍生的权利，但以下情况除外：为便于实施本文档中规定的技术规范，任何人均可在软件、软件随附的支持材料以及软件文档中准备和分发衍生作品以及本文档的部分内容，前提是所有此类作品都包含以下声明。但是，明确禁止发布本文档的衍生作品以用作技术规范。

此外，“代码组件”（明确标记为 Web IDL 的部分中的 Web IDL、W3C 定义的标记（HTML、CSS 等）以及明确标记为代码示例的计算机编程语言代码）均根据 W3C 软件许可证获得许可。

该声明是：

“版权所有 © 2015 W3C®（MIT、ERCIM、Keio、Beihang）。本软件或文档包括从 [W3C 文档的标题和 URI] 复制或衍生的材料。”

免责声明

本文档“按原样”提供，并且版权所有者不作任何明示或暗示的陈述或保证，包括但不限于对适销性、适用于特定用途、不侵权或侵犯所有权的保证；本文档的内容适用于任何目的；也不保证该等内容的实施不会侵犯任何第三方的专利、版权、商标或其他权利。

版权所有者对因使用本文档或执行或履行其内容而造成的任何直接、间接、特殊或后果性损害不承担任何责任。

未经事先明确书面许可，版权所有者的名称和商标不得用于与本文档或其内容相关的广告或宣传。本文档中的版权所有权将始终归版权所有者所有。

标头声明

下表显示了语法插槽类型支持的标头声明。有关更多信息，请参阅《W3C 正式推荐标准》1.0 版语音识别语法规范中的语法标头声明。

声明	规范要求	XML 表单	Amazon Lex 支持	规范
语法版本	必填	4.3：`grammar` 元素上的 `version` 属性	必填	SRGS
XML 命名空间	必填项（仅限 XML）	4.3：`grammar` 元素上的 `xmlns` 属性	必填	SRGS

声明	规范要求	XML 表单	Amazon Lex 支持	规范
文档类型	必填项（仅限 XML）	4.3：XML DOCTYPE	推荐	SRGS
字符编码	推荐	4.4：XML 声明中的 encoding 属性	推荐	SRGS
语言	在语音模式下是必需的 在 DTMF 模式下被忽略	4.5：grammar 元素上的 xml:lang 属性	在语音模式下是必需的 在 DTMF 模式下被忽略	SRGS
Mode	可选	4.6：grammar 元素上的 mode 属性	可选	SRGS
根规则	可选	4.7：grammar 元素上的 root 属性	必填	SRGS
标记格式	可选	4.8：grammar 元素上的 tag-format 属性	支持字符串文本和 ECMAScript	SRGS、SISR
基本 URI	可选	4.9：grammar 元素上的 xml:base 属性	可选	SRGS
发音词典	可选、允许多个	4.10：lexicon 元素	不支持	SRGS、PLS
元数据	可选、允许多个	4.11.1：meta 元素	必填	SRGS

声明	规范要求	XML 表单	Amazon Lex 支持	规范
XML 元数据	可选、仅限 XML	4.11.2：metadata 元素	可选	SRGS
标签	可选、允许多个	4.12：tag 元素	不支持全局标记	SRGS

示例

```
<?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">
```

支持的 XML 元素

Amazon Lex V2 支持以下 XML 元素作为自定义语法：

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

令牌

下表显示了语法插槽类型支持的令牌规范。有关更多信息，请参阅《W3C 正式推荐标准》1.0 版语音识别语法规范中的令牌。

令牌类型	示例	支持?
单个无引号令牌	hello	是
单个无引号令牌：非字母	2	是
单个引号的令牌、无空格	"hello"	支持。如果双引号只包含单个令牌，则将其删除
两个由空格分隔的令牌	bon voyage	是
四个由空格分隔的令牌	this is a test	是
单个引号的令牌，有空格	"San Francisco	否
前后有 <token> 标记的单个 XML 令牌	<token>San Francisco</token>	不支持（与单个引号且有空格的令牌相同）

备注

- 单个引号的令牌，有空格：该规范要求将用双引号括起来的单词视为单个令牌。Amazon Lex V2 将其视为以空格分隔的令牌。
- <token> 中的单个 XML 令牌：该规范需要以 <token> 分隔的单词来表示一个令牌。Amazon Lex V2 将其视为以空格分隔的令牌。
- 当您的语法中发现任何一种用法时，Amazon Lex V2 都会引发验证错误。

示例

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

```
</rule>
```

规则引用

下表汇总了语法文档中可能存在的各种形式的规则引用。有关更多信息，请参阅《W3C 正式推荐标准》1.0 版语音识别语法规范中的规则引用。

引用类型	XML 表单	支持
2.2.1 对本地规则的显式引用	`<ruleref uri="#rulename"/>`	是
2.2.2 对由 URI 标识的语法的命名规则的显式引用	`<ruleref uri="grammarURI#rulename"/>`	否
2.2.2 对由 URI 标识的语法的根规则的隐式引用	`<ruleref uri="grammarURI"/>`	否
2.2.2 对由具有媒体类型的 URI 标识的语法的命名规则的显式引用	`<ruleref uri="grammarURI#rulename" type="media-type"/>`	否
2.2.2 对由具有媒体类型的 URI 标识的语法的根规则的隐式引用	`<ruleref uri="grammarURI" type="media-type"/>`	否

2.2.3 特殊规则定义	`<ruleref special="NULL"/>`
	`<ruleref special="VOID"/>`
	`<ruleref special="GARBAGE"/>`
		否

备注

1. 语法 URI 是一个外部 URI。例如，`http://grammar.example.com/world-cities.grxml`。
2. 媒体类型可以是：

- application/srgs+xml
- text/plain

示例

```
<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>
```

序列和封装

以下示例显示了支持的序列。有关更多信息，请参阅《W3C 正式推荐标准》1.0 版语音识别语法规范中的序列和封装。

示例

```
<!-- 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>
```

重复

下表显示了支持的规则重复扩展。有关更多信息，请参阅《W3C 正式推荐标准》1.0 版语音识别语法规范中的重复。

XML 表单 示例	行为	支持?
repeat="n" repeat="6"	所包含的表达式恰好重复“n”次。“n”必须为“0”或正整数。	是
repeat="m-n" repeat="4-6"	所包含的扩展在“m”与“n”次（含）之间重复。“m”和“n”必须都是“0”或正整数，并且“m”必须小于或等于“n”。	是
repeat="m-" repeat="3-"	所包含的扩展会重复“m”次或更多次（含）。“m”必须是“0”或正整数。例如，“3-”声明所包含的扩展可以出现三次、四次、五次或更多次。	是
repeat="0-1"	所包含的扩展是可选的。	是
<item repeat="2-4" repeat-prob="0.8">		否

语言

以下讨论适用于语法的语言标识符。有关更多信息，请参阅《W3C 正式推荐标准》1.0 版语音识别语法规范中的语言。

默认情况下，语法是单一语言文档，语法标头的语言声明中提供了语言标识符。除非另有声明，否则该语法中的所有令牌均将根据该语法的语言进行处理。不支持语法级别的语言声明。

在以下示例中：

1. Amazon Lex V2 支持语言“en-US”的语法标头声明。
2. 不支持项目级语言附件（以##突出显示）。如果语言附件与标头声明不同，Amazon Lex V2 会引发验证错误。

```
<?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>
```

标签

以下讨论适用于为语法定义的标签。有关更多信息，请参阅《W3C 正式推荐标准》1.0 版语音识别语法规范中的标记。

根据 SRGS 规范，可以通过以下方式定义标记：

1. 作为标头声明的一部分，如标头声明中所述。
2. 作为 <rule> 定义的一部分。

支持以下标记格式：

- `semantics/1.0`（SISR、ECMAScript）
- `semantics/1.0-literals`（SISR 字符串文本）

不支持以下标记格式：

- `swi-semantics/1.0`（Nuance 专有）

示例

```
<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>
```

Weight

您可以为元素添加权重属性。权重是一个正的浮点值，表示语音识别期间项目中短语的增强程度。有关更多信息，请参阅《W3C 正式推荐标准》1.0 版语音识别语法规范中的权重。

权重必须大于 0 且小于等于 10，且只能有一个小数位。如果权重大于 0 且小于 1，则该短语会被负增强。如果权重大于 1 且小于等于 10，则该短语会被正增强。权重 1 等同于无权重，而且该短语不做增强处理。

为项目指定适当的权重以提高语音识别性能是一项艰巨的任务。以下是一些在指定权重时可以遵循的提示：

- 开始时使用没有指定项目权重的语法。
- 确定语音中经常被误认的模式。
- 对权重应用不同的值，直到您注意到语音识别性能有所改善并且没有回归为止。

示例 1

例如，如果有适用于机场的语法，并且您发现 New York 经常被误认为是 Newark，则可以通过将“New York”的权重指定为 5 来对该值进行正增强。

```
<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>
```

示例 2

例如，您有适用于机票预订代码的语法，该机票预定代码以英文字母开头，后跟三位数字。该预定代码最有可能以 B 或 D 开头，但您会发现 B 经常被误认为是 P，D 经常被误认为是 T。此时，您可以对 B 和 D 进行正增强。

```
<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>
```

脚本格式

Amazon Lex V2 支持使用以下 ECMAScript 功能来定义语法。

在语法中指定标记时，Amazon Lex V2 支持以下 ECMAScript 功能。`tag-format` 必须在语法中使用 ECMAScript 标记时发送到 `semantics/1.0`。有关更多信息，请参阅 ECMA-262 ECMAScript 2021 语言规范。

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

主题

- 变量语句
- 表达式
- If 语句
- 切换语句
- 函数声明
- 迭代语句
- 块语句
- 注释
- 不支持的语句

本文档包含来自 ECMAScript 标准的材料（参见 https://www.ecma-international.org/publications-and-standards/standards/ecma-262/）。ECMAScript 语言规范文档可根据以下许可证从 Ecma International 获得。

许可证文本

(iv) 通过实施（例如全部或部分复制和粘贴）其中的功能，在符合标准的产品中使用本规范。

但是，不得以任何方式修改本文档本身的内容，包括删除版权声明或对 Ecma International 的引用，除非需要将其翻译成英语以外的语言或其他格式。

Ecma International 文件的正式版本是 Ecma International 网站上的英文版本。如果翻译版本和正式版本之间存在差异，则以正式版本为准。

上文所授予的有限权限是永久性的，不会被 Ecma International 或其继任者或受让人撤销。本文档以及本文中所包含的信息是“按原样”提供的，Ecma International 不作任何明示或暗示的担保，包括但不限于对使用本文中的信息不会侵犯任何所有权的任何担保，或者对适销性或适用于特定目的的任何默示担保。”

变量语句

变量语句定义了一个或多个变量。

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

表达式

表达式类型	语法	示例	支持?
正则表达式文本	包含有效正则表达式特殊字符的字符串文本	`"^\d\.$"`	否
函数	`function functionName(parameters) { functionBody}`	`var x = function calc() { return 10; }`	否
删除	`delete expression`	`delete obj.property;`	否
Void	`void expression`	`void (2 == '2');`	否

表达式类型	语法	示例	支持?
Typeof	`typeof expression`	`typeof 42;`	否
成员索引	`expression [ expressions ]`	`var fruits = ["apple"]; fruits[0];`	是
成员点	`expression . identifier`	`out.value`	是
Arguments	`expression (arguments)`	`new Date('1994-10-11')`	是
后增量	`expression++`	`var x=10; x++;`	是
后减量	`expression--`	`var x=10; x--;`	是
预增量	`++expression`	`var x=10; ++x;`	是
预减量	`--expression`	`var x=10; --x;`	是
一元加号/一元减号	`+expression / -expression`	`+x / -x;`	是
按位非	`~ expression`	`const a = 5; consol e.log( ~a );`	是
逻辑非	`! expression`	`!(a > 0 || b > 0)`	是

表达式类型	语法	示例	支持?
倍增	`expression ('*' | '/' | '%') expression`	`(x + y) * (a / b)`	是
加	`expression ('+' | '-') expression`	`(a + b) - (a - (a + b))`	是
位移位	`expression ('<<' | '>>' | '>>>') expression`	`(a >> b) >>> c`	是
相对	`expression ('<' | '>' | '<=' | '>=') expression`	`if (a > b) { ... }`	是
In	`expression in expression`	`fruits[0] in otherFruits;`	是
等于	`expression ('==' | '!=' | '===' | '!==') expression`	`if (a == b) { ... }`	是
按位与/按位异或/按位或	`expression ('&' | '^' | '|') expression`	`a & b / a ^ b / a | b`	是
逻辑和/逻辑或	`expression ('&&' | '||') expression`	`if (a && (b ||c)) { ...}`	是

表达式类型	语法	示例	支持?
三元	`expression ? expression : expression`	`a > b ? obj.prop : 0`	是
赋值	`expression = expression`	`out.value = "string";`	是
赋值运算符	`expression ('*=' | '/=' | '+=' | '-=' | '%=') expression`	`a *= 10;`	是
赋值按位运算符	`expression ('<<=' | '>>=' | '>>>=' | '&=' | '^=' | '|=') expression`	`a <<= 10;`	是
标识符	`identifierSequence` 其中 identifierSequence 是一系列有效字符	`fruits=[10, 20, 30];`	是
Null 文本	`null`	`x = null;`	是
布尔文本	`true | false`	`x = true;`	是
字符串文本	`'string' / "string"`	`a = 'hello', b = "world";`	是
十进制文本	`integer [.] digits [exponent]`	`111.11 e+12`	是

表达式类型	语法	示例	支持?
十六进制文本	`0 (x | X)[0-9a-fA-F]`	`0x123ABC`	是
八进制文本	`0 [0-7]`	`"051"`	是
数组文本	`[ expression, ... ]`	`v = [a, b, c];`	是
对象文本	`{property: value, ...}`	`out = {value: 1, flag: false};`	是
带圆括号	`( expressions )`	`x + (x + y)`	是

If 语句

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

注意：在上一个示例中，expressions 和 statements 必须是本文档中支持的选项之一。

切换语句

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

注意：在上一个示例中，expressions 和 statements 必须是本文档中支持的选项之一。

函数声明

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

迭代语句

迭代语句可以是以下任意一种：

```
// 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
}
```

块语句

```
{
   statements
}

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

```
}
```

注意：在上一个示例中，块中提供的 `statements` 必须是本文档中支持的选项之一。

注释

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

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

不支持的语句

Amazon Lex V2 不支持以下 ECMAScript 功能。

主题

- 空语句
- Continue 语句
- Break 语句
- Return 语句
- Throw 语句
- Try 语句
- Debugger 语句
- 带标签的语句
- Class 声明

空语句

空语句用于不提供任何语句。以下是空语句的句法：

```
;
```

Continue 语句

支持将不带标签的 continue 语句与迭代语句配合使用。不支持带标签的 continue 语句。

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

Break 语句

支持将不带标签的 break 语句与迭代语句配合使用。不支持带标签的 break 语句。

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

Return 语句

```
return expression;
```

Throw 语句

Throw 语句用于引发用户定义的异常。

```
throw expression;
```

Try 语句

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

Debugger 语句

Debugger 语句用于调用环境提供的调试功能。

```
debugger;
```

带标签的语句

带标签的语句可以与 break 或 continue 语句配合使用。

```
label:
   statements


// Example
let str = '';

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

console.log(str);
```

Class 声明

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

行业语法

行业语法是一组用于语法插槽类型的 XML 文件。在将交互式语音响应工作流程迁移到 Amazon Lex V2 时，您可以使用这些语法快速提供一致的最终用户体验。您可以从三个域的一系列预先构建的语法中进行选择：金融服务、保险和电信。还有一组通用的语法可以用作您语法的基础。

这些语法包含收集信息的规则以及用于语义解释的 ECMAScript 标记。

金融服务语法（下载）

金融服务支持以下语法：账户和路由号码、信用卡和贷款号码、信用评分、账户开立和注销日期以及社保号码。

账号

```
<?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>
```

路由号码

```
<?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>
```

信用卡号

```
<?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>
```

贷款 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 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>
```

信用评分

```
<?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>
```

开户日期

```
<?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>
```

自动发薪日期

```
<?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="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>
```

对账单日期

```
<?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>
```

转账金额

```
<?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>
```

社保号码

```
<?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>
```

保险语法（下载）

保险域支持以下语法：理赔和保单编号、驾照和车牌号、到期日期、保险日期和续保日期、理赔和保单金额。

理赔 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="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>
```

策略 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>
```

驾照号码

```
<?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>
```

车牌号

```
<?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="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>
```

保单到期日期（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">

         <!-- 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>
```

保单续保日期（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: 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>
```

保单开始日期

```
<?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>
```

理赔金额

```
<?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>
```

保费金额

```
<?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>
```

保单数量

```
<?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>
```

电信语法（下载）

电信支持以下语法：电话号码、序列号、SIM 卡号、美国邮政编码、信用卡到期日期、套餐开始日期、续订日期和到期日期、服务开始日期、设备数量和账单金额。

电话号码

```
<?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>
```

序列号

```
<?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 卡号

```
<?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>
```

美国邮政编码

```
<?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="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>
```

套餐到期日期（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">

         <!-- 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>
```

套餐续订日期（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 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>
```

套餐开始日期（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>
```

服务开始日期（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>
```

设备数量

```
<?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>
```

账单金额

```
<?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>
```

通用语法（下载）

我们提供以下通用语法：字母数字、货币、日期 (mm/dd/yy)、数字、问候语、犹豫和代理。

字母数字

```
<?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="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>
```

Currency

```
<?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/mm）

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

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

```
                    <item><ruleref uri="#above_twenty"/><tag>out += rules.above_twenty+
 " ";</tag></item>
                </one-of>
                <item><ruleref uri="#months"/><tag>out = out + rules.months;</tag></
item>
            </item>
        </rule>

        <rule id="months">
           <one-of>
             <item>january<tag>out="january";</tag></item>
             <item>february<tag>out="february";</tag></item>
             <item>march<tag>out="march";</tag></item>
             <item>april<tag>out="april";</tag></item>
             <item>may<tag>out="may";</tag></item>
             <item>june<tag>out="june";</tag></item>
             <item>july<tag>out="july";</tag></item>
             <item>august<tag>out="august";</tag></item>
             <item>september<tag>out="september";</tag></item>
             <item>october<tag>out="october";</tag></item>
             <item>november<tag>out="november";</tag></item>
             <item>december<tag>out="december";</tag></item>
             <item>jan<tag>out="january";</tag></item>
             <item>feb<tag>out="february";</tag></item>
             <item>aug<tag>out="august";</tag></item>
             <item>sept<tag>out="september";</tag></item>
             <item>oct<tag>out="october";</tag></item>
             <item>nov<tag>out="november";</tag></item>
             <item>dec<tag>out="december";</tag></item>
           </one-of>
       </rule>

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

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

    <rule id="teens">
        <one-of>
            <item>ten<tag>out=10;</tag></item>
            <item>tenth<tag>out=10;</tag></item>
            <item>eleven<tag>out=11;</tag></item>
            <item>twelve<tag>out=12;</tag></item>
            <item>thirteen<tag>out=13;</tag></item>
            <item>fourteen<tag>out=14;</tag></item>
            <item>fifteen<tag>out=15;</tag></item>
            <item>sixteen<tag>out=16;</tag></item>
            <item>seventeen<tag>out=17;</tag></item>
            <item>eighteen<tag>out=18;</tag></item>
            <item>nineteen<tag>out=19;</tag></item>
            <item>tenth<tag>out=10;</tag></item>
            <item>eleventh<tag>out=11;</tag></item>
            <item>twelveth<tag>out=12;</tag></item>
            <item>thirteenth<tag>out=13;</tag></item>
            <item>fourteenth<tag>out=14;</tag></item>
            <item>fifteenth<tag>out=15;</tag></item>
            <item>sixteenth<tag>out=16;</tag></item>
            <item>seventeenth<tag>out=17;</tag></item>
            <item>eighteenth<tag>out=18;</tag></item>
            <item>nineteenth<tag>out=19;</tag></item>
```

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

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

日期（mm/yy）

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

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

```
        <rule id="months">
           <tag>out.mon=""</tag>
           <one-of>
               <item>january<tag>out.mon+="january";</tag></item>
               <item>february<tag>out.mon+="february";</tag></item>
               <item>march<tag>out.mon+="march";</tag></item>
               <item>april<tag>out.mon+="april";</tag></item>
               <item>may<tag>out.mon+="may";</tag></item>
               <item>june<tag>out.mon+="june";</tag></item>
               <item>july<tag>out.mon+="july";</tag></item>
               <item>august<tag>out.mon+="august";</tag></item>
               <item>september<tag>out.mon+="september";</tag></item>
               <item>october<tag>out.mon+="october";</tag></item>
               <item>november<tag>out.mon+="november";</tag></item>
               <item>december<tag>out.mon+="december";</tag></item>
               <item>jan<tag>out.mon+="january";</tag></item>
               <item>feb<tag>out.mon+="february";</tag></item>
               <item>aug<tag>out.mon+="august";</tag></item>
               <item>sept<tag>out.mon+="september";</tag></item>
               <item>oct<tag>out.mon+="october";</tag></item>
               <item>nov<tag>out.mon+="november";</tag></item>
               <item>dec<tag>out.mon+="december";</tag></item>
           </one-of>
       </rule>

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

        <rule id="teens">
            <one-of>
                <item>ten<tag>out=10;</tag></item>
                <item>eleven<tag>out=11;</tag></item>
```

```
                <item>twelve<tag>out=12;</tag></item>
                <item>thirteen<tag>out=13;</tag></item>
                <item>fourteen<tag>out=14;</tag></item>
                <item>fifteen<tag>out=15;</tag></item>
                <item>sixteen<tag>out=16;</tag></item>
                <item>seventeen<tag>out=17;</tag></item>
                <item>eighteen<tag>out=18;</tag></item>
                <item>nineteen<tag>out=19;</tag></item>
            </one-of>
        </rule>

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

        <rule id="thousands">
            <!-- <item>
                <ruleref uri="#digits"/>
                <tag>out = (1000 * rules.digits);</tag>
                thousand
            </item> -->
            <item>two thousand<tag>out=2000;</tag></item>
            <item repeat="0-1">and</item>
            <item repeat="0-1"><ruleref uri="#digits"/><tag>out += rules.digits;</
tag></item>
            <item repeat="0-1"><ruleref uri="#teens"/><tag>out += rules.teens;</tag></
item>
            <item repeat="0-1"><ruleref uri="#above_twenty"/><tag>out +=
 rules.above_twenty;</tag></item>
        </rule>

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

```
        </rule>

</grammar>
```

日期（dd/mm/yyyy）

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

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

         <rule id="months">
```

```
        <tag>out.mon=""</tag>
        <one-of>
            <item>january<tag>out.mon+="january";</tag></item>
            <item>february<tag>out.mon+="february";</tag></item>
            <item>march<tag>out.mon+="march";</tag></item>
            <item>april<tag>out.mon+="april";</tag></item>
            <item>may<tag>out.mon+="may";</tag></item>
            <item>june<tag>out.mon+="june";</tag></item>
            <item>july<tag>out.mon+="july";</tag></item>
            <item>august<tag>out.mon+="august";</tag></item>
            <item>september<tag>out.mon+="september";</tag></item>
            <item>october<tag>out.mon+="october";</tag></item>
            <item>november<tag>out.mon+="november";</tag></item>
            <item>december<tag>out.mon+="december";</tag></item>
            <item>jan<tag>out.mon+="january";</tag></item>
            <item>feb<tag>out.mon+="february";</tag></item>
            <item>aug<tag>out.mon+="august";</tag></item>
            <item>sept<tag>out.mon+="september";</tag></item>
            <item>oct<tag>out.mon+="october";</tag></item>
            <item>nov<tag>out.mon+="november";</tag></item>
            <item>dec<tag>out.mon+="december";</tag></item>
        </one-of>
    </rule>

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

    <rule id="teens">
        <one-of>
            <item>ten<tag>out=10;</tag></item>
            <item>eleven<tag>out=11;</tag></item>
            <item>twelve<tag>out=12;</tag></item>
```

```
                <item>thirteen<tag>out=13;</tag></item>
                <item>fourteen<tag>out=14;</tag></item>
                <item>fifteen<tag>out=15;</tag></item>
                <item>sixteen<tag>out=16;</tag></item>
                <item>seventeen<tag>out=17;</tag></item>
                <item>eighteen<tag>out=18;</tag></item>
                <item>nineteen<tag>out=19;</tag></item>
            </one-of>
        </rule>

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

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

 </grammar>
```

数字（数字）

```
<?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>
```

数字（序数）

```
<?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>
```

代理

```
<?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>
```

问候语

```
<?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>
```

犹豫

```
<?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>
```

复合插槽类型

复合插槽是两个或多个插槽的组合，用于在单个用户输入中捕获多条信息。例如，您可以将机器人配置为通过请求“城市和州/省/自治区或邮政编码”来引发位置。相反，如果对话配置为使用单独的插槽类型，则会导致对话体验僵硬（询问“哪个城市？”，然后紧接着询问“邮政编码是什么？”）。使用复合插槽，您可以通过单个插槽来捕获所有信息。复合插槽是称为子插槽的组合，例如城市、州/省/自治区和邮政编码。

您可以组合使用可用的 Amazon Lex 插槽类型（内置）和您自己的插槽（自定义插槽）。您可以设计逻辑表达式来捕获所需子插槽中的信息。例如：城市和州/省/自治区或邮政编码。

复合插槽类型仅适用于 en-US。

创建复合插槽类型

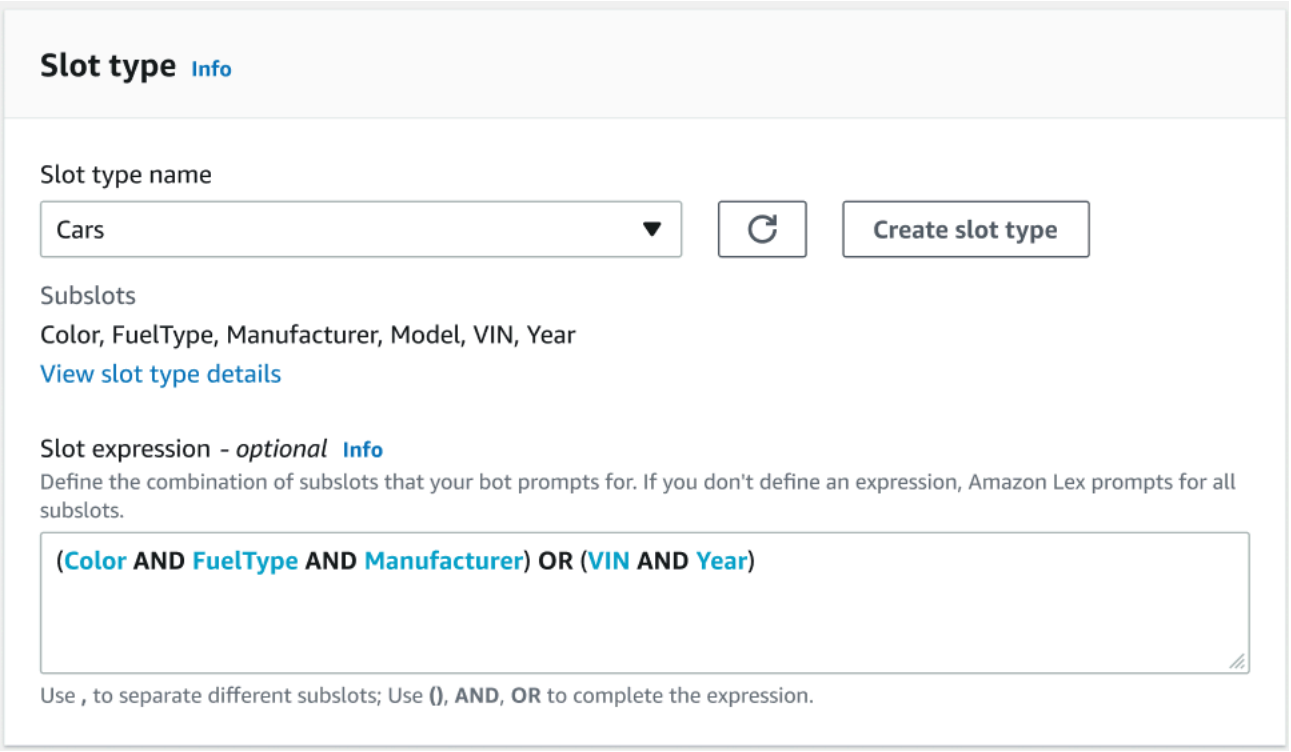
要在复合插槽中使用子插槽，必须先配置复合插槽类型。为此，请使用添加插槽类型控制台步骤或 API 操作。为复合插槽类型选择名称和描述后，必须提供子插槽的信息。有关添加插槽类型的更多信息，请参阅添加槽类型

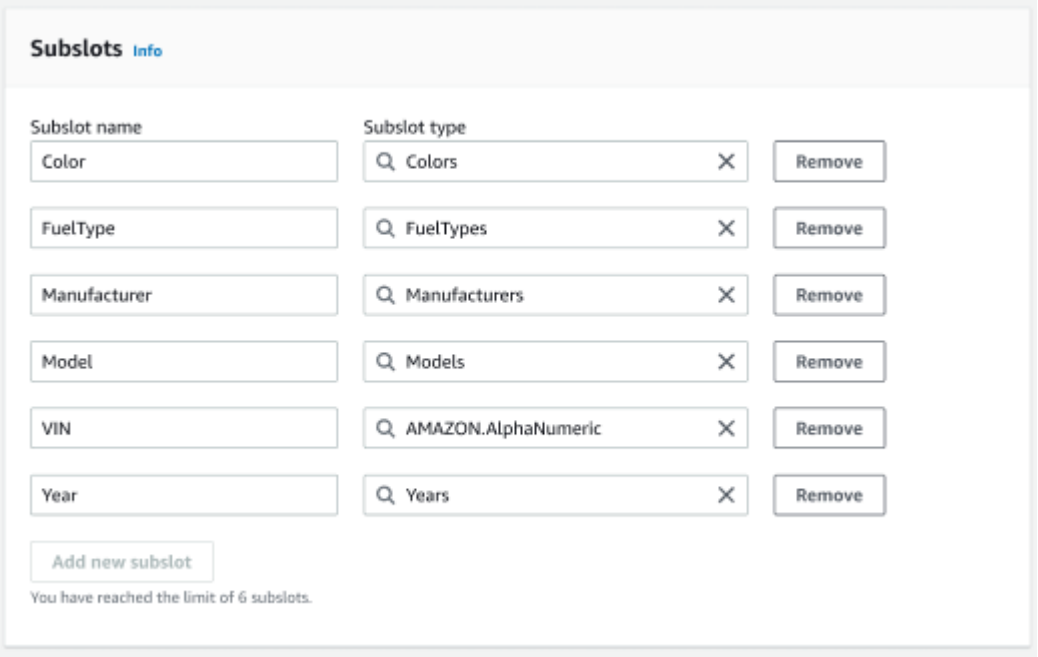
子插槽

复合插槽类型需要配置底层插槽，称为子插槽。如果要在一次请求中从客户处引发多条信息，请配置子插槽的组合。例如：城市、州/省/自治区以及邮政编码。您可以为一个复合插槽添加最多 6 个子插槽。

单一插槽类型的插槽可用于向复合插槽类型添加子插槽。但是，您无法使用复合插槽类型作为子插槽的插槽类型。

下图以复合插槽“汽车”为例，其中包括子插槽“颜色”、“燃油类型”、“制造商”、“型号”、“VIN”和“年份”。





表达式生成器

为便于复合插槽的履行，您可以选择使用表达式生成器。借助表达式生成器，您可以设计逻辑插槽表达式，以按所需顺序捕获所需的子插槽值。作为布尔表达式的一部分，您可以使用诸如 AND 和 OR 等运算符。根据设计的表达式，当履行所需的子插槽时，即视为履行该复合插槽。

使用复合插槽类型

对于某些意图，您可能需要将不同的插槽作为单个插槽的一部分来捕获。例如，车辆保养预约安排机器人可能有以下言语的意图：

```
My car is a {car}
```

该意图预期 {car} 复合插槽包含一系列插槽，其中包含车辆的详细信息。例如，“2021 White Toyota Camry”（2021 款白色丰田凯美瑞）。

复合插槽不同于多值插槽。复合插槽由多个插槽组成，每个插槽都有自己的值。而多值插槽是可以包含一系列值的单一插槽。有关多值插槽的更多信息，请参阅使用一个插槽中的多个值

对于复合插槽，Amazon Lex 会在响应 RecognizeText 或 RecognizeUtterance 操作时为每个子插槽返回一个值。以下是 CarService 机器人为言语“I want to schedule a service for my “2021 White Toyota Camry"”（我想为我的“2021 款白色丰田凯美瑞”预订保养服务）返回的插槽信息。

```
"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"
            }
        }
    },
    ...
}
```

可以在对话的第一回合或第 n 回合中引发复合插槽。基于所提供的输入值，复合插槽可以引发其余所需的子插槽。

复合插槽总是为每个子插槽返回一个值。当言语中不包含给定子插槽的可识别值时，该特定子插槽不会返回任何响应。

复合插槽适用于文本和语音输入。

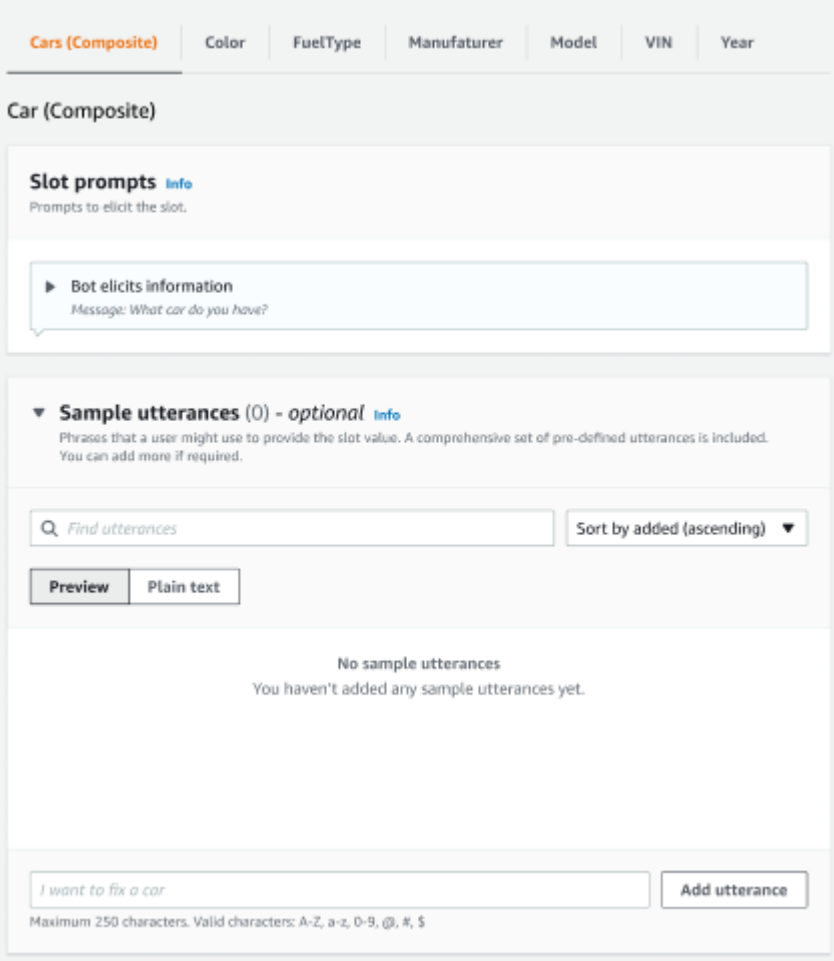
向意图中添加插槽时，复合插槽只能作为自定义插槽类型使用。

您可以在提示中使用复合插槽。例如，您可以为意图设置确认提示。

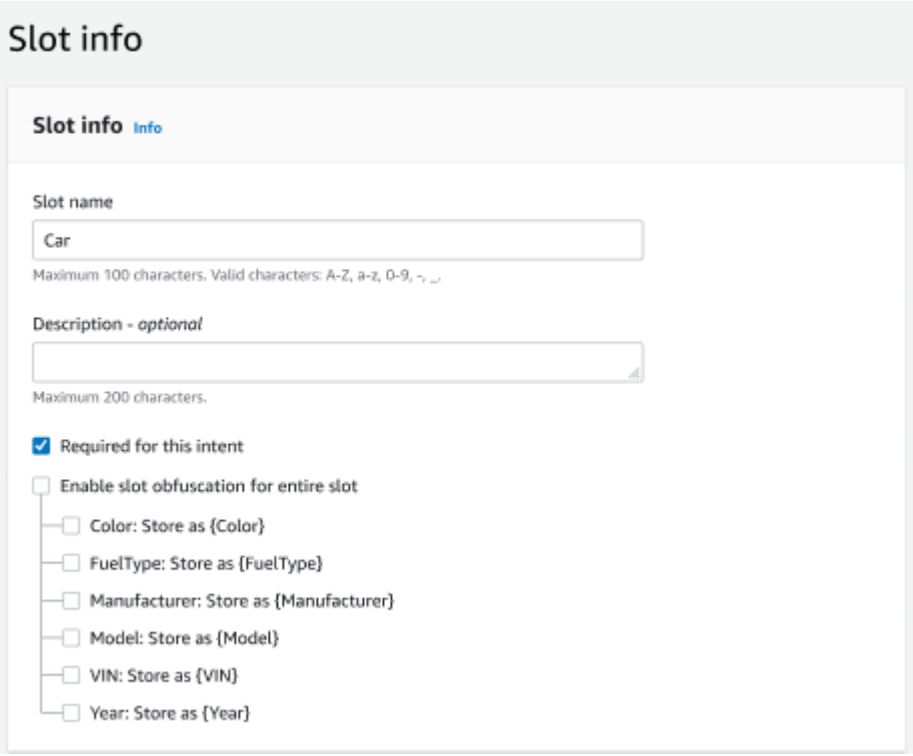
```
Would you like me to schedule service for your 2021 White Toyota Camry?
```

当向用户发送提示时，Amazon Lex 会发送“Would you like me to schedule service for your 2021 White Toyota Camry?”（是否需要为您的 2021 款白色丰田凯美瑞预约保养服务？）。

每个子插槽都配置为一个插槽。您可以添加插槽提示来引发子插槽和示例言语。您可以为子槽启用“等待并继续”以及默认值。有关更多信息，请参阅使用默认槽值。



您可以使用插槽模糊处理来掩盖对话日志中的整个复合插槽。请注意，插槽模糊处理应用于复合插槽级别，启用后，属于复合插槽的子插槽的值将被模糊处理。当您对插槽值进行模糊处理时，每个插槽值将被替换为插槽的名称。有关更多信息，请参阅掩盖对话日志中的槽位值。



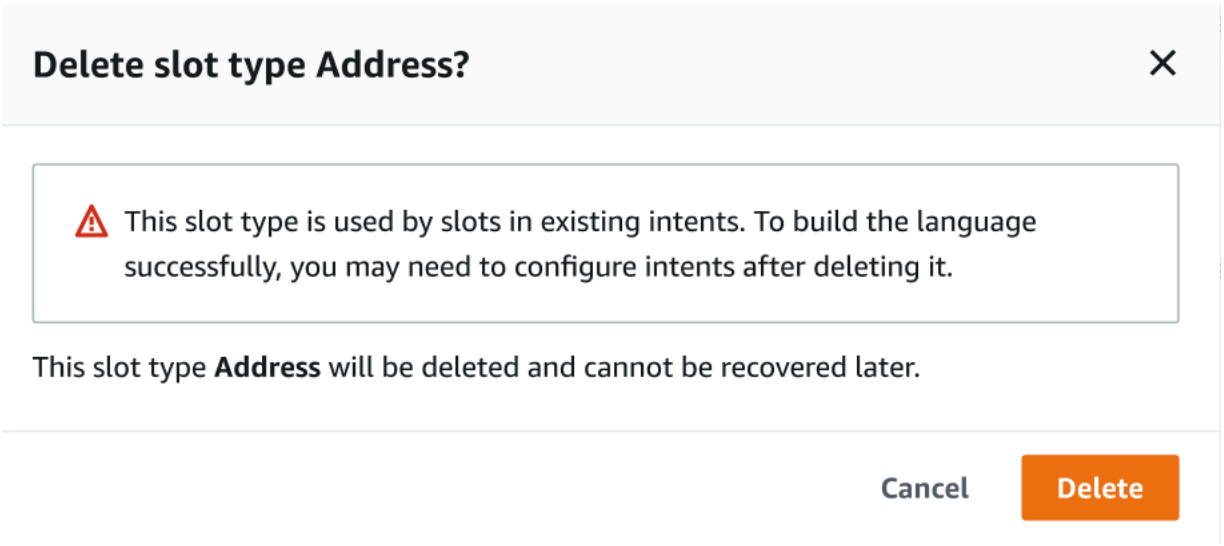
编辑复合插槽类型

您可以从复合插槽配置中编辑子插槽，以修改子插槽名称和插槽类型。但是，当意图正在使用复合插槽时，您需要首先编辑意图，然后才能修改子插槽。

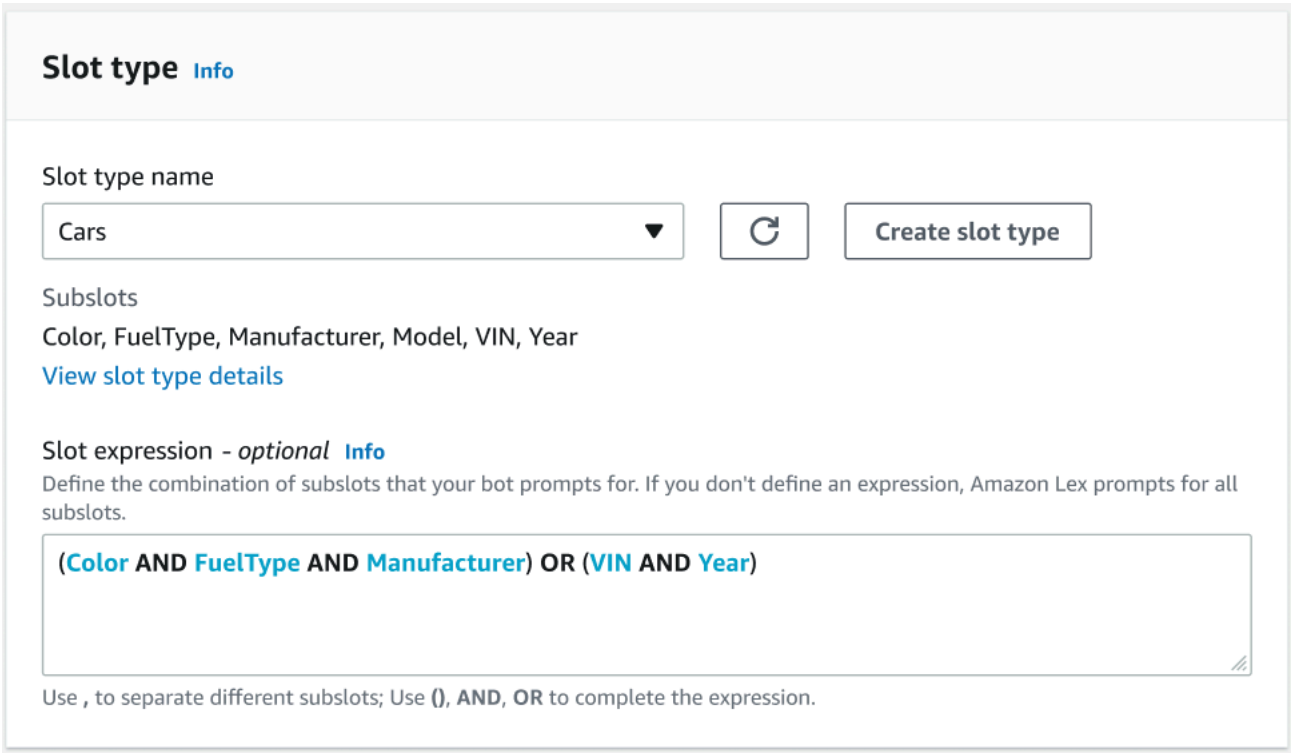
> Existing intents use this slot type. To build the language successfully, you may need to configure those intents after editing sub slots.

删除复合插槽类型

您可以从复合插槽配置中删除子插槽。请注意，当意图正在使用子插槽时，子插槽仍会从该意图中删除。



表达式生成器中的插槽表达式提供关于子插槽已被删除的警报。



使用控制台测试机器人

Amazon Lex V2 控制台包含一个测试窗口，可用于测试与机器人的交互。您可以使用该测试窗口与您的机器人进行测试对话，以查看您的应用程序从机器人接收到的响应。

您可以使用机器人执行两种类型的测试。第一种是快速测试，用于使用创建机器人时使用的确切短语来测试您的机器人。例如，如果您在意图中添加了言语“我想摘花”，则可以使用该确切的短语来测试机器人。

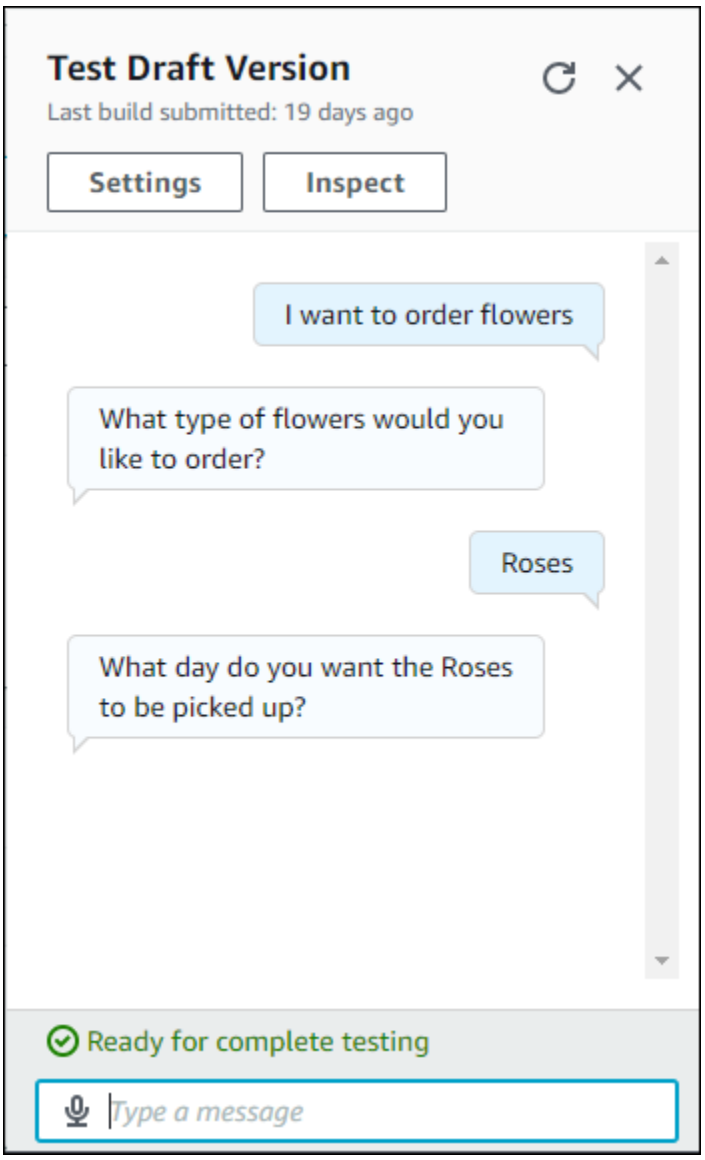
第二种是完整测试，用于与您配置的言语相关的短语来测试您的机器人。例如，您可以使用短语“可以帮我预定鲜花吗”来开始与您机器人的对话。

您可以使用特定的别名和语言测试机器人。如果您正在测试机器人的开发版本，则使用 `TestBotAlias` 别名进行测试。

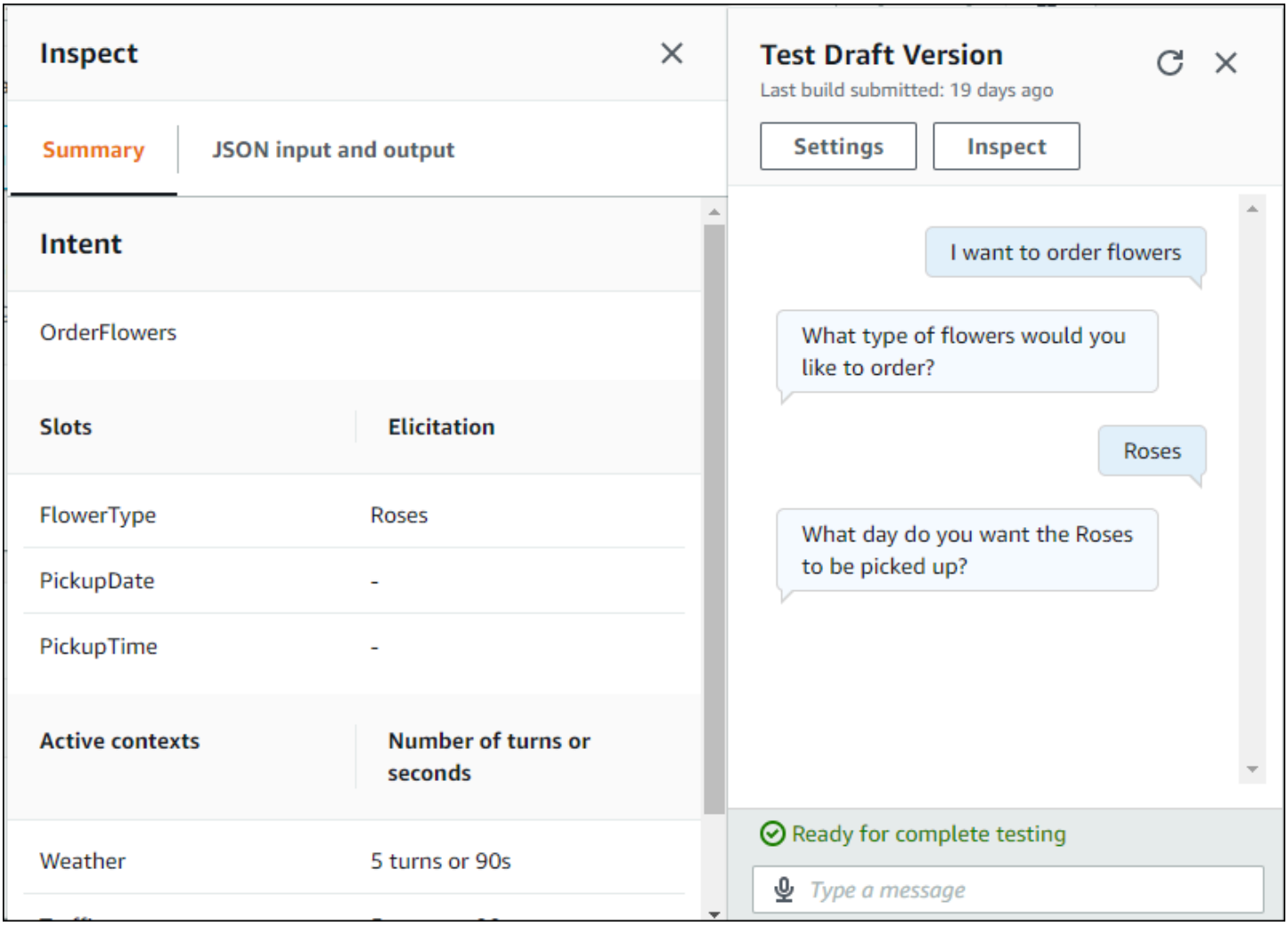
要打开测试窗口，请执行以下操作：

1. 登录到 AWS Management Console，并通过以下网址打开 Amazon Lex 控制台：https://console.aws.amazon.com/lex/。
2. 从机器人列表中选择要测试的机器人。
3. 在左侧菜单中选择别名。
4. 从别名列表中选择要测试的别名。
5. 从语言中选择要测试语言的单选按钮，然后选择测试。

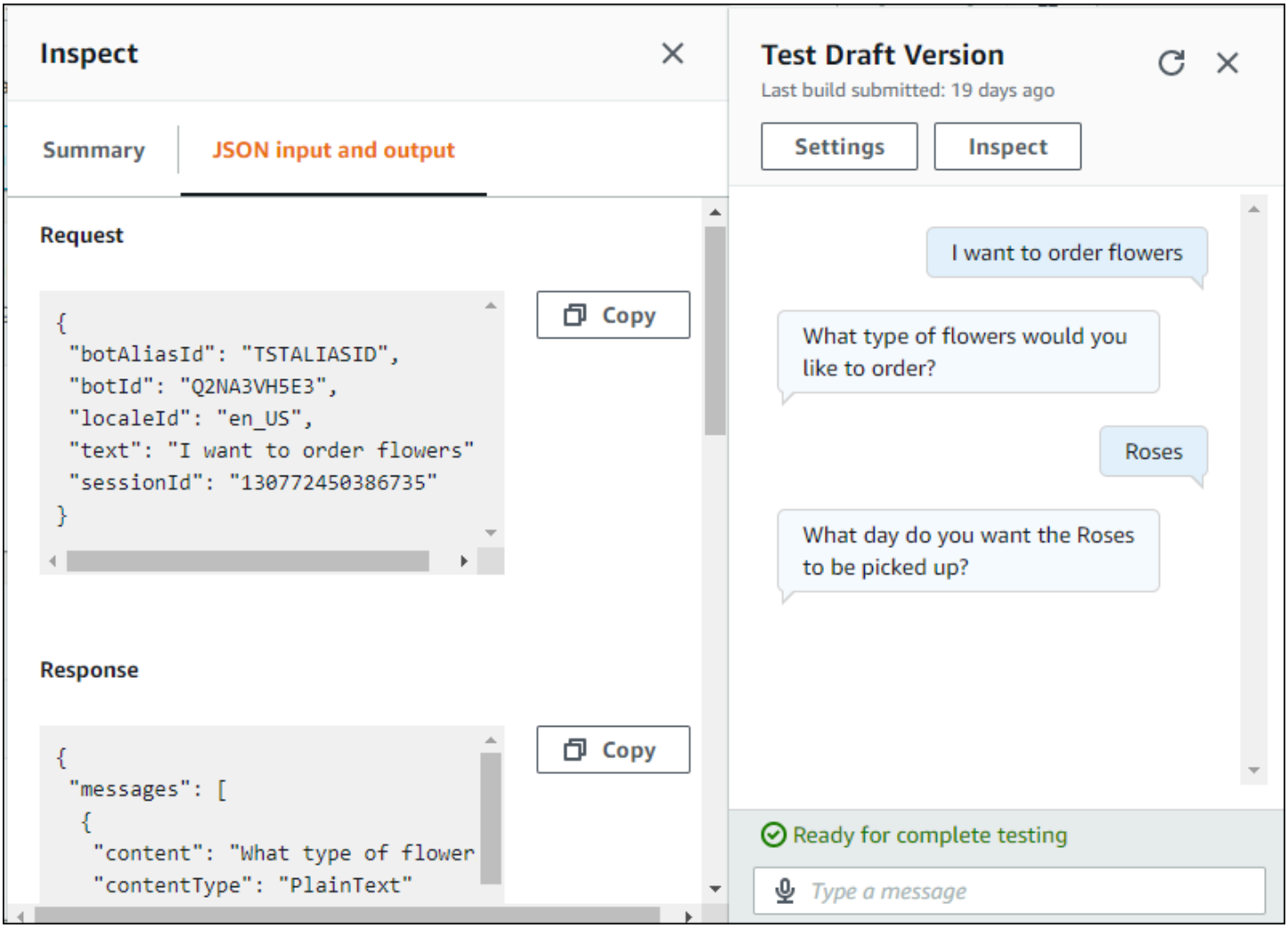
选择测试后，将在控制台中打开测试窗口。您可以使用测试窗口与机器人进行交互，如下图中所示。



除了对话外，您还可以在测试窗口中选择检查以查看机器人返回的响应。第一个视图显示了从您的机器人返回到测试窗口的信息的摘要。



您还可以使用测试检查窗口来查看在机器人和测试窗口之间发送的 JSON 结构。您可以看到来自测试窗口的请求以及来自 Amazon Lex V2 的响应。


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

利用生成式人工智能优化机器人的创建和性能

> **Note**
>
> 这些功能使用生成式人工智能。使用该服务时，请记住，它可能会给出不准确或不恰当的响应。有关更多信息，请参阅 AWS 负责任的 AI 策略。
> 由 Amazon Bedrock 提供支持： AWS 实现自动滥用检测。由于 Amazon Lex V2 生成式人工智能功能建立在 Amazon Bedrock 之上，因此用户继承了 Amazon Bedrock 中实施的控制措施，以确保安全、负责任地使用人工智能。

利用 Amazon Bedrock 的生成式人工智能功能，自动执行和加快 Amazon Lex V2 机器人构建流程。您可以借助 Amazon Bedrock 执行以下流程。

- 创建新的机器人，并使用自然语言描述，有效地为它们填充相关的意图和槽位类型。
- 根据机器人的意图自动生成示例言语。
- 提高机器人的槽位解析性能。
- 创建意图以帮助回答客户的问题。

您可以通过控制台或 API 为 Amazon Lex V2 激活生成式人工智能功能。

> **Note**
>
> 在利用生成式人工智能功能之前，您必须满足以下先决条件
>
> 1. 导航到 Amazon Bedrock 控制台并注册您打算使用的 Anthropic Claude 模型的访问权限（有关更多信息，请参阅模型访问权限）。有关使用 Amazon Bedrock 的定价信息，请参阅 Amazon Bedrock 定价。
> 2. 为机器人区域设置开启生成式人工智能功能。为此，请按照利用生成式人工智能优化机器人的创建和性能中的步骤进行操作。

Using the console

1. 登录 AWS Management Console 并打开 Amazon Lex V2 主机，网址为 https://console.aws.amazon.com/lexv2/home。

2. 选择要开启生成式人工智能功能的机器人和机器人中的区域设置。
3. 在生成式人工智能配置部分中，选择配置。
4. 切换要激活的每项功能的已启用按钮。选择要用于该功能的模型和版本。启用一项功能可能会产生额外费用。有关使用 Amazon Bedrock 的定价信息，请参阅 Amazon Bedrock 定价。要了解有关某项功能的更多信息，请从以下列表中选择相应的主题。开启要激活的功能后，请选择保存。此时会显示绿色的成功横幅，以确认这些功能已开启。

Using the API

1. 要为新机器人启用生成式 AI 功能，请使用该CreateBot操作创建新机器人。
2. 发送CreateBotLocale请求，必要时修改generativeAISettings对象。如果您要为现有机器人启用这些功能，UpdateBotLocale请改为发送请求。

   a. 要启用描述性机器人生成器，请修改 `descriptiveBotBuilder` 对象。指定要在 `modelArn` 字段中使用的基础模型，并将 `enabled` 值设置为 `True`。

   b. 要改进槽位解析，请修改 `slotResolutionImprovement` 对象。指定要在 `modelArn` 字段中使用的基础模型，并将 `enabled` 值设置为 `True`。

   c. 要启用示例言语生成，请修改 `sampleUtteranceGeneration` 对象。指定要在 `modelArn` 字段中使用的基础模型，并将 `enabled` 值设置为 `True`。

主题

使用描述性机器人生成器

> **Note**
>
> 在利用生成式人工智能功能之前，您必须满足以下先决条件
>
> 1. 导航到 Amazon Bedrock 控制台并注册您打算使用的 Anthropic Claude 模型的访问权限（有关更多信息，请参阅模型访问权限）。有关使用 Amazon Bedrock 的定价信息，请参阅 Amazon Bedrock 定价。

2. 为机器人区域设置开启生成式人工智能功能。为此，请按照利用生成式人工智能优化机器人的创建和性能中的步骤进行操作。

描述性机器人生成器允许您利用 Amazon Bedrock 对大型语言模型的访问权限来提高机器人创建过程的效率。您可以使用自然语言进行提示，包括机器人的用途和应执行的操作。Amazon Lex V2 利用 Amazon Bedrock 的功能，根据您的描述为机器人生成相关的意图和槽位类型。选择要保留的意图和槽位类型后，您就可以迭代机器人，以根据您的特定使用案例对其进行修改。有了描述性机器人生成器，您无需为机器人手动创建意图和槽位类型，从而节省时间。

描述性机器人生成器在英语区域设置中可用（请参阅Amazon Lex V2 支持的语言和区域设置的表中以 en_ 开头的区域设置）。

创建机器人之前，请执行以下操作。

1. 请查看使用自然语言描述创建机器人时所需的权限中的步骤，检查您的角色是否具有正确的权限。
2. 决定要使用的描述。您可以参阅机器人描述示例中的机器人描述示例。

通过使用自然语言来描述机器人应能做什么，创建一个机器人。Amazon Lex V2 调用 Amazon Bedrock 模型来生成适合机器人使用案例的意图和槽位类型。您可使用控制台或 API 创建机器人。

Console

使用描述性机器人生成器创建机器人

1. 登录 AWS Management Console，并通过以下网址打开 Amazon Lex V2 控制台：https://console.aws.amazon.com/lexv2/home。
2. 在机器人页面中，选择创建机器人。
3. 对于创建方法，请选择描述性机器人生成器 - GenAI。
4. 为机器人提供名称和可选描述，配置 IAM 权限，并选择机器人是否受 COPPA 约束。然后选择下一步。
5. 选择创建机器人时所用的语言、机器人的语音以及意图分类的置信度阈值（有关更多信息，请参阅使用意图置信度分数）。
6. 在描述性机器人生成器 - GenAI 下，为要创建的机器人提供描述。您的描述应既详细 又准确，以帮助为机器人生成适当和充分的意图。包括一份改进意图创建过程的操作列表。
7. 在选择模型下选择模型提供商和模型。

8. 要在其他区域设置中创建机器人，请选择添加其他语言。添加完语言后，请选择完成。Amazon Lex V2 会创建您的机器人，而描述性机器人生成器会为其生成意图和槽位。生成区域设置后，横幅将从蓝色变为绿色。选择查看以查看生成的意图和槽位类型。

> **Note**
>
> 描述性机器人生成器目前仅在英语区域设置中可用。但是，您可以在创建机器人后将其复制到非英语区域设置。

查看生成的意图和槽位类型，并将其添加到机器人

1. 如果有足够多的意图和槽位类型适用于机器人的使用案例，您就可以查看生成的意图。
   a. 查看生成的意图。
      i. 选中意图旁边的复选框，将该意图从要添加到机器人的意图列表中删除。
      ii. 选择意图名称以查看为该意图生成的示例言语和槽位。
      iii. 默认情况下，所有言语和槽位都处于选中状态。选中一个复选框可将该项目从意图中删除。选择添加至选择可将选中的项目保留在意图中。
   b. 查看生成的槽位类型。
      i. 选中槽位类型旁边的复选框，将该槽位类型从要添加到机器人的意图列表中删除。
      ii. 将槽位类型添加到机器人后，可以为其添加值
2. 如果您对自己的意图和槽位类型感到满意，请选择页面顶部的添加意图和槽位类型，将意图和槽位类型添加到机器人。
3. 资源添加完毕后，会显示绿色的成功横幅。转到意图和槽位类型以编辑生成的意图和槽位类型，并添加更多值。
4. 如果生成的意图和生成的槽位类型大多不适用于要创建的机器人，请执行以下步骤。
   a. 在描述性机器人生成器详细信息部分中选择新生成。
   b. 重写提示并选择重新生成以生成新的意图和槽位类型。如果您使用不同的模型，结果会有所不同。

> **Important**
>
> 不能保证会生成相同的意图和槽位。每次重新生成意图和槽位类型时都要收费。

API

使用自然语言描述创建机器人

当通过 API 使用描述性机器人生成器时，它会在 Amazon S3 存储桶中的 .zip 文件中创建机器人定义。您下载此文件并将机器人定义导入 Amazon Lex V2 以创建机器人。

1. 发送 CreateBot 请求以创建新机器人。然后发送 CreateBotLocale 请求，为机器人创建区域设置。
2. 发送 StartBotResourceGeneration 请求，指定机器人的 ID、版本和区域设置。您可以使用 DRAFT 作为机器人版本。在 generationInputPrompt 字段中提供提示。您的描述应既详细又准确，以帮助为机器人生成适当和充分的意图。包括一份改进意图创建过程的操作列表。
3. 记下响应中的 generationId。
4. 使用您在 StartBotResourceGeneration 响应中收到的 generationId 发送 DescribeBotResourceGeneration 请求。包括机器人 ID、版本和区域设置。
5. 如果 DescribeBotResourceGeneration 响应中的 generationStatus 是 Complete，则也将填充 generatedBotLocaleUrl 字段。按照下载对象中的步骤，使用此 Amazon S3 URI 下载机器人定义。

检查生成的机器人定义并将其导入

1. 按照下载对象中的步骤，使用 DescribeBotResourceGeneration 响应的 generationStatus 中的 Amazon S3 URI 下载机器人定义。
2. 您可以通过编辑文件，根据机器人的特定使用案例直接修改生成的内容。您也可以发送另一个 StartBotResourceGeneration 请求来重新生成意图和槽位。

   > **Important**
   >
   > 不能保证会生成相同的意图和槽位。每次重新生成意图和槽位类型时都要收费。

3. 要导入机器人定义，请按照导入中的步骤操作。
4. 导入后，您可以使用 UpdateIntent、UpdateSlot 和 UpdateSlotType 操作修改生成的意图和槽位。

要列出有关某个机器人区域设置的所有已生成项目的元数据，请使用 ListBotResourceGenerations 操作。使用 DescribeBotResourcGeneration 请求中任何返回的 generationId 值来检索已生成机器人定义的 Amazon S3 URI。

主题

机器人描述示例

行业	提示示例
金融服务	我们提供金融卡服务，帮助用户在收到新卡时执行一些任务，例如激活卡、通过电子邮件或邮寄方式发送 PIN 码、验证新卡（使用邮政编码）。我们还帮助他们完成与现有信用卡相关的任务，例如查询信用卡优惠、挂失、申请新卡、重置卡 PIN 码或支付账单。
餐饮服务	我想要一个机器人来帮助顾客订餐（使用商品 ID、数量、尺寸）、查看订单状态和取消订单。使用订单 ID 编制订单索引。
航空公司	我们是一个航空公司域，可帮助用户预订机票、查看预订详情、获取已预订航班的收据、查询航班状态、重新安排预订的航班、获取航班详情以及取消预订的航班。如果其他意图有助于支持域描述中的函数，也可以生成这些意图。
保险	目标：我们是一家销售汽车、房屋和年金保单的保险公司。我想要一个可以查看索赔状态、提出索赔、支付保单和取消保单的机器人。我们使用 policy_id 和 SSN 的后 4 位进行账户识别和验证。我希望机器人至少具有以下意图和槽位：身份验证 - policy_id、SSN 的后 4 位；保单类型：汽车、房屋、年金；保单状态：查询余额、检查到期日、检查承保范围；付款：一次性付款、分期付款、金额
车辆管理	我们正在构建一个“拖车查询”机器人，帮助城市中汽车被拖走的驾驶员查找汽车的位置。该机器

行业	提示示例
	人应询问汽车被拖出的地址或位置，以及有关车辆的详细信息，如车牌、汽车品牌、车型和制造年份。该机器人应回复拖车停车场的位置和营业时间。
旅行	我是一家旅行社，我想要机器人帮我的顾客预订迪士尼之旅。迪士尼在世界各地有多个园区可供选择，还有酒店、餐饮和特殊娱乐设施可供预订。该机器人的用户应能够修改或取消预订。预订必须至少包括园区、日期和酒店。餐饮或娱乐项目为可选项，可稍后添加或更改。

使用自然语言描述创建机器人时所需的权限

- 要在 Amazon Lex V2 控制台上访问该功能，请确保您的控制台角色具有 `bedrock:ListFoundationModels` 权限。
- 与机器人关联的 IAM 角色应具有 `bedrock:InvokeModel` 权限。在 Amazon Lex 控制台中启用该功能时，只要您的机器人使用的是由 Amazon Lex 生成的服务相关角色，该策略就会自动添加到机器人角色中。

```
{
  "Version": "2012-10-17",
  "Statement": [
      {
          "Effect": "Allow",
          "Action": [
              "bedrock:InvokeModel"
          ],
          "Resource": [
              "arn:aws:bedrock:region::foundation-model/model-id"
          ]
      }
  ]
}
```

言语生成

> **Note**
>
> 在利用生成式人工智能功能之前，您必须满足以下先决条件
>
> 1. 导航到 Amazon Bedrock 控制台并注册您打算使用的 Anthropic Claude 模型的访问权限（有关更多信息，请参阅模型访问权限）。有关使用 Amazon Bedrock 的定价信息，请参阅 Amazon Bedrock 定价。
> 2. 为机器人区域设置开启生成式人工智能功能。为此，请按照利用生成式人工智能优化机器人的创建和性能中的步骤进行操作。

使用言语生成功能自动创建符合您意图的示例言语。Amazon Lex V2 可根据意图名称、描述和现有的示例言语为您生成示例言语，而无需手动输入示例言语，这样您就可以减少在发现和编写自己的示例言语上花费的时间和精力。当 Amazon Lex V2 生成言语后，您可以编辑和删除这些言语。使用此工具可以加快意图识别过程的示例言语的创建。

要允许生成言语，请按照利用生成式人工智能优化机器人的创建和性能中的步骤激活生成式人工智能功能。

您可以使用控制台或 API 生成言语。

Console

1. 导航到机器人中任何意图的示例言语部分（在可视化对话生成器中，该部分位于开始块中）。
2. 选择生成言语按钮以生成 5 个示例言语。如果您的意图包含超过 25 个示例言语，则生成言语按钮将处于禁用状态。
3. 生成的言语以绿色横幅显示，以将生成的言语与现有言语区分开来。
4. 将鼠标悬停在言语上方可显示对生成的言语进行编辑、删除和排序的选项。

API

1. 发送 GenerateBotElement 请求，填写要生成示例言语的意图和机器人 ID、版本和区域设置。
2. 响应会返回一个 SampleUtterance 对象列表，每个对象都包含一个生成的言语。
3. 要将言语添加到意图中，请发送 UpdateIntent 请求并将这些言语添加到 `sampleUtterances` 字段中。

主题

言语生成权限

要在 Amazon Lex V2 控制台上访问该功能，请确保您的控制台角色具有 `bedrock:ListFoundationModels` 和 `bedrock:InvokeModel` 权限。

使用辅助槽位解析

> **Note**
>
> 在利用生成式人工智能功能之前，您必须满足以下先决条件
>
> 1. 导航到 Amazon Bedrock 控制台并注册您打算使用的 Anthropic Claude 模型的访问权限（有关更多信息，请参阅模型访问权限）。有关使用 Amazon Bedrock 的定价信息，请参阅 Amazon Bedrock 定价。
> 2. 为机器人区域设置开启生成式人工智能功能。为此，请按照利用生成式人工智能优化机器人的创建和性能中的步骤进行操作。

您可以使用辅助槽位解析来提高机器人对话流中某些内置槽位的准确性。辅助槽位解析使用 Amazon Bedrock 大型语言模型 (LLM) 来改善对某些内置槽位的识别，从而改善对槽位引发期间客户响应的解释。对于无法正常解析的言语，Amazon Lex 将尝试使用 Amazon Bedrock 进行第二次解析。

辅助槽位解析允许您利用 Amazon Bedrock 基础模型的强大功能来提高以下内置槽位的准确性：

- 不支持正则表达式的 `AMAZON.Alphanumeric`
- `AMAZON.City`
- `AMAZON.Country`
- `AMAZON.Date`
- `AMAZON.Number`
- `AMAZON.PhoneNumber`
- `AMAZON.Confirmation`

您可以为使用上面所列内置槽位的任何意图启用辅助槽位解析。辅助槽位解析不适用于上面未列出的自定义槽位或 Amazon 内置槽位。

在 Amazon Lex 机器人中启用辅助槽位解析后，您可以使用对话日志和指标收集有关提高准确性的数据。

- 对话日志 - 如果使用 Amazon Bedrock 来解析槽位，则解释将 `Bedrock` 作为 `interpretationSource`。
- CloudWatch 指标 - 指标将在 CloudWatch 指标中列出的维度下发布。要了解更多信息，请参阅使用 Amazon CloudWatch 监控 Amazon Lex。

要使用描述性机器人生成器，请按照辅助槽位解析的权限中的步骤确保您的 IAM 角色具有适当权限。

主题

辅助槽位解析示例

下面给出了一些示例，其中辅助槽位解析能够智能地将用户的言语解析为一个值。

AMAZON.Number

Vertical	slotType	slotName	slotPrompt	言语	解析值
旅行	AMAZON.Number	numberOfNightsStayed	您在这趟旅行中住了几晚？	整整一周，7 个晚上。	7
银行业	AMAZON.Number	numberOfPeopleOnTheAccount	该账户上有多少人？	我和我的妻子。	2
旅行	AMAZON.Number	numberOfStops	有多少站？	日本一站。洛杉矶一站。	2

AMAZON.AlphaNumeric

Vertical	slotType	slotName	slotPrompt	言语	解析值
汽车租赁	AMAZON.Alphanumeric	transactionId	您的事务 ID 是什么？	可能是 AWE8349RJ。	AWE8349RJ
旅行	AMAZON.Alphanumeric	confirmationCode	您的预订确认编号是多少？	确认编号是 BLT2UE。	BLT2UE

AMAZON.Date

Vertical	slotType	slotName	slotPrompt	言语	解析值	currentDate
汽车租赁	AMAZON.Date	dueDate	租赁协议何时到期？	租约将于下个月 1 日到期。	2023-12-01	2023-11-09
旅行	AMAZON.Date	returnDate	您什么时候回来？	今天晚些时候，大约 7 点。	2023-11-09	2023-11-09

AMAZON.PhoneNumber

Vertical	slotType	slotName	slotPrompt	言语	解析值
保险	AMAZON.PhoneNumber	policyHolder	保单持有人的电话号码是多少？	保单持有人的电话号码是 123-456-7890。	1234567890
零售	AMAZON.PhoneNumber	phoneLookup	能否告知您的电话号码，这	我想应该用的是 413-570-9	4135709617

Vertical	slotType	slotName	slotPrompt	言语	解析值
			样我才能找到您的账户？	617，让我再确认一下。	

AMAZON.Country

Vertical	slotType	slotName	slotPrompt	言语	解析值
旅行	AMAZON.Country	nativeCountry	您的原籍国是哪里？	我是印度人。	印度
银行业	AMAZON.Country	countryItinerary	您将使用借记卡前往哪些国家/地区？	我将前往新德里。	印度

AMAZON.City

Vertical	槽位类型	意图	问题	响应	解析值
保险	AMAZON.City	policyHolderCity	保单持有人居住在哪个城市？	我住在斯普林菲尔德。	斯普林菲尔德
旅行	AMAZON.City	destinationCity	您要前往哪个城市？	我要去东京。	东京

AMAZON.Confirmation

Vertical	slotType	slotName	slotPrompt	言语	解析值
保险	AMAZON.Confirmation	policyExpired	保单是否过期？	是，很遗憾，已经过期了。	是
银行业	AMAZON.Confirmation	hasInvestments	您是否进行了任何投资？	我尚未投资任何东西。	否

在生成式人工智能配置屏幕中启用辅助槽位解析

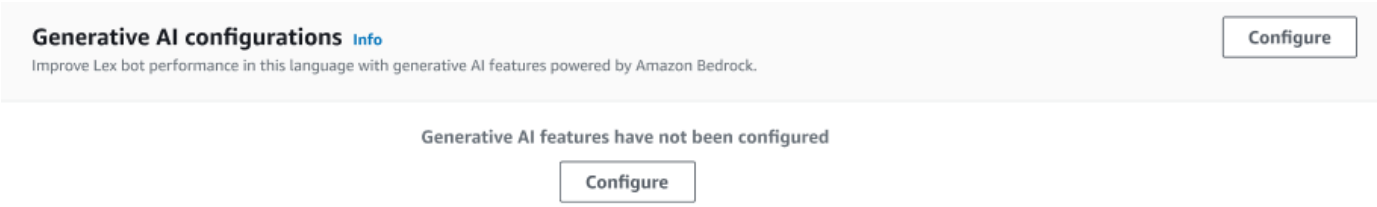
您可以导航到生成式人工智能屏幕，为支持的内置槽位启用辅助槽位解析。

如果槽位是支持的内置槽位，则可以选择在槽位级别激活辅助槽位解析。

1. 登录 AWS Management Console，并通过以下网址打开 Amazon Lex V2 控制台：https://console.aws.amazon.com/lexv2/home。
2. 在机器人下的导航窗格中，选择要用于辅助槽位解析的机器人。
3. 为要启用的机器人选择语言英语(US)。
4. 转到屏幕上的生成式人工智能配置部分。
5. 如果该功能尚未启用，请选择转到 Amazon Bedrock 注册并启用该功能。

> **Note**
>
> 如果还没有对 Amazon Bedrock 基础模型的访问权限，您应该看到转到 Amazon Bedrock。单击转到 Amazon Bedrock 进入 Amazon Bedrock 页面，您可以在其中注册对基础模型的访问权限。辅助槽位解析目前支持 Claude V2 和 Claude Instant V1。为获得出色效果，建议使用 Claude V2。

6. 如果您已经具有对 Bedrock 基础模型的访问权限，则应该会看到配置按钮。单击该按钮可进入生成式人工智能配置页面，以激活 Lex 中的生成式人工智能功能。



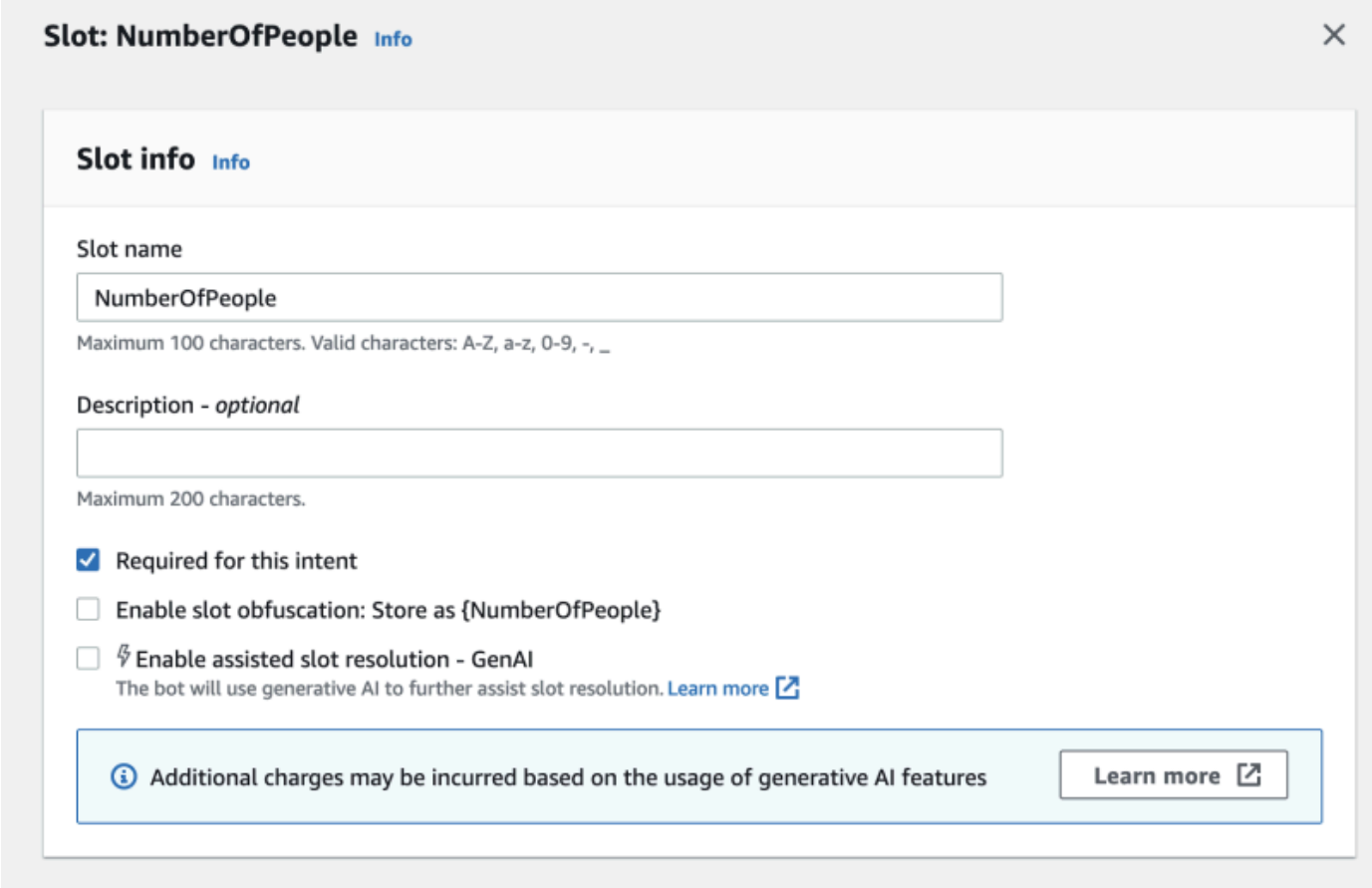
7. 在框的右上角，向右移动滑块以选择已启用设置。
8. 选择启用按钮以激活所选槽位的辅助槽位解析。
9. 您可以通过从列表中选择槽位并选择禁用按钮来禁用辅助槽位解析。

在槽位设置中启用辅助槽位解析

您可以导航到每个有槽位的意图的槽位级别，从而为支持的内置槽位启用辅助槽位解析。槽位必须是上述支持的内置槽位之一，才能选择激活辅助槽位解析。如果槽位没有激活辅助槽位解析的选项，则该选项将显示为灰色。

> **Note**
>
> 您必须先在生成式人工智能面板上激活辅助槽位解析功能，才能为各槽位激活该功能。

1. 登录 AWS 管理控制台，并通过以下网址打开 Amazon Lex V2 控制台：https://console.aws.amazon.com/lexv2/home。
2. 在机器人下的导航窗格中，选择要用于辅助槽位解析的机器人。
3. 在“所有语言”下，选择英语(US) 以展开列表。
4. 在左侧面板中，选择意图以查看所选机器人中的意图列表。
5. 在意图屏幕中，选择包含要修改的槽位的意图。
6. 选择意图名称以查看该意图的槽位。
7. 选择槽位部分中的高级选项按钮。
8. 选中启用辅助槽位解析复选框以启用该功能。



9. 选择屏幕右下角的更新槽位按钮。这将为您选择的槽位激活辅助槽位解析。

您可以通过调用 API 来为支持的内置槽位启用辅助槽位解析。

- 按照利用生成式人工智能优化机器人的创建和性能中的步骤为您的机器人区域设置启用辅助槽位解析。
- 发送 UpdateSlot 请求，指定要启用辅助槽位解析的槽位。在 `slotResolutionSetting` 字段中，将 `slotResolutionStrategy` 值设置为 `EnhancedFallback`。要创建启用辅助槽位解析的新槽位，请改为发送 CreateSlot 请求。

辅助槽位解析的权限

- 要在 Amazon Lex V2 控制台上访问该功能，请确保您的控制台角色具有 `bedrock:ListFoundationModels` 权限。
- 与机器人关联的 IAM 角色应具有 `bedrock:InvokeModel` 权限。在 Amazon Lex 控制台中启用该功能时，只要您的机器人使用的是由 Amazon Lex 生成的服务相关角色，该策略就会自动添加到机器人角色中。

```
{
  "Version": "2012-10-17",
  "Statement": [
      {
          "Effect": "Allow",
          "Action": [
              "bedrock:InvokeModel"
          ],
          "Resource": [
              "arn:aws:bedrock:Region::foundation-model/modelId"
          ]
      }
  ]
}
```

AMAZON.QnAIntent

> **Note**
>
> 在利用生成式人工智能功能之前，您必须满足以下先决条件

1. 导航到 Amazon Bedrock 控制台并注册您打算使用的 Anthropic Claude 模型的访问权限（有关更多信息，请参阅模型访问权限）。有关使用 Amazon Bedrock 的定价信息，请参阅 Amazon Bedrock 定价。
2. 为机器人区域设置开启生成式人工智能功能。为此，请按照利用生成式人工智能优化机器人的创建和性能中的步骤进行操作。

您可以利用 Amazon Bedrock FM 在机器人对话中帮助回答客户的问题。Amazon Lex V2 提供了一个内置 `AMAZON.QnAIntent`，您可以将其添加到机器人中。此意图通过识别客户问题并从以下知识库（例如 **`Can you provide me details on the baggage limits for my international flight?`**）中搜索答案，来利用 Amazon Bedrock 的生成式人工智能功能。该功能减少了在 Amazon Lex V2 意图中使用面向任务的对话配置问题和答案的需求。此意图还会根据对话历史记录识别后续问题（例如 **`What about domestic flight?`**），并相应地提供答案。

按照AMAZON.QnAIntent 的权限中的步骤确保您的 IAM 角色具有访问 `AMAZON.QnAIntent` 的适当权限。

要利用 `AMAZON.QnAIntent`，您必须设置以下知识库之一。

- 亚马逊 OpenSearch 服务数据库 — 有关更多信息，请参阅创建和管理亚马逊 OpenSearch 服务域。
- Amazon Kendra 索引 – 有关更多信息，请参阅创建索引。
- Amazon Bedrock 知识库 – 有关更多信息，请参阅构建知识库。

您可以通过以下两种方式之一设置 `AMAZON.QnAIntent`：

使用生成式人工智能配置进行设置

1. 在 Amazon Lex V2 控制台中，从左侧导航窗格中选择机器人，然后从机器人部分选择要添加意图的机器人。
2. 从左侧导航窗格中，选择要添加意图的语言。
3. 在生成式人工智能配置部分中，选择配置。
4. 在 QnA 配置部分中，选择创建 QnA 意图。

通过向机器人添加内置意图进行设置

1. 在 Amazon Lex V2 控制台中，从左侧导航窗格中选择机器人，然后从机器人部分选择要添加意图的机器人。
2. 从左侧导航窗格中，选择要添加意图的语言下的意图。
3. 选择添加意图，然后从下拉菜单中选择使用内置意图。
4. 有关 AMAZON.QnAIntent 配置的更多详细信息，请参阅AMAZON.QnAIntent。

> **Note**
>
> 当某个言语未被归类为机器人中存在的任何其他意图时，就会激活 AMAZON.QnAIntent。当某个言语未被归类为机器人中存在的任何其他意图时，就会激活该意图。请注意，引发槽位值时，不会因为错过的言语而激活该意图。AMAZON.QnAIntent 被识别后，将使用指定的 Amazon Bedrock 模型搜索已配置的知识库并回答客户的问题。

主题

AMAZON.QnAIntent 的权限

要在 Amazon Lex V2 控制台上访问该功能，请确保您的控制台角色具有 bedrock:ListFoundationModels 权限。

与机器人关联的 IAM 角色应具有 AMAZON.QnAIntent 所需的以下权限。机器人角色应具有调用 bedrock:InvokeModel 权限。您还应为在机器人的 AMAZON.QnAIntent 中指定的每个数据存储附加一条语句（请参阅以下策略中的 Permissions to access Amazon Kendra index、Permissions to access OpenSearch Service index 和 Permissions to access knowledge base in Amazon Bedrock 语句）。在 Amazon Lex 控制台中启用该功能时，只要您的机器人使用的是由 Amazon Lex 生成的服务相关角色，这些策略就会自动添加到机器人角色中。

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

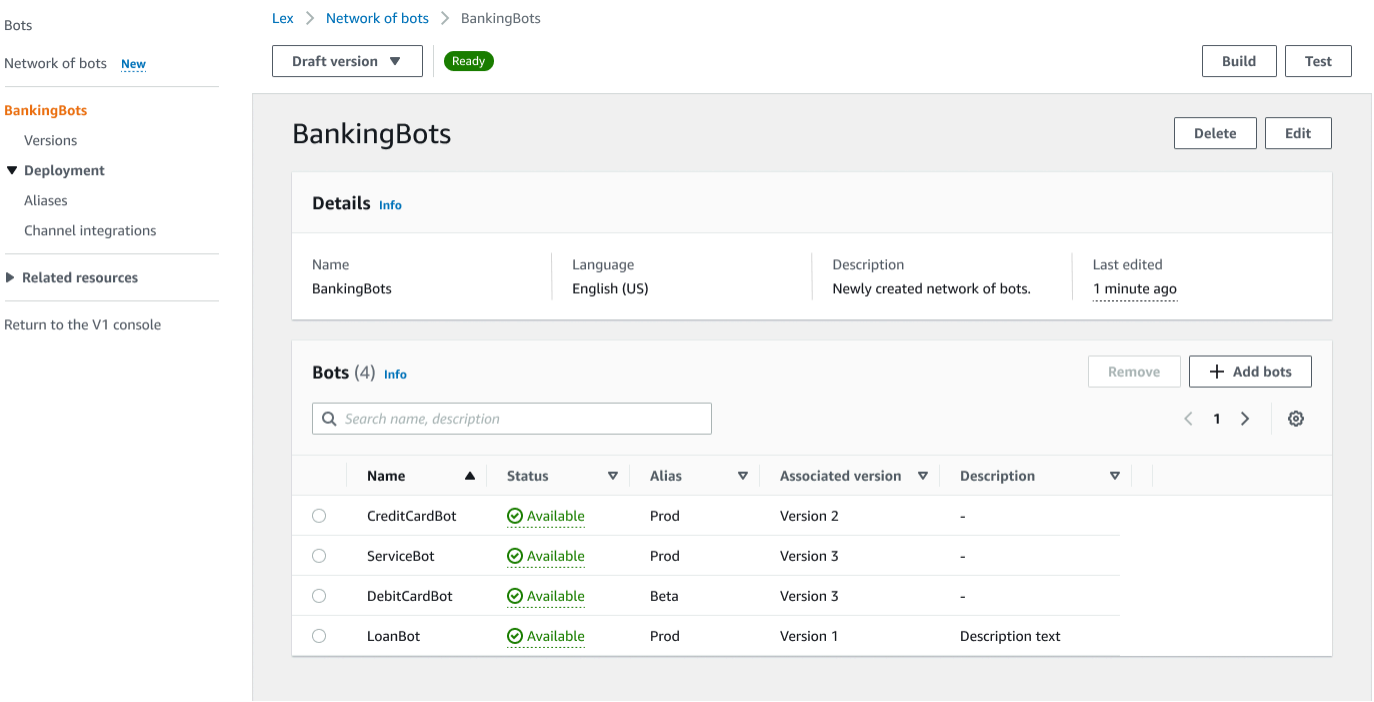
创建机器人网络

通过机器人网络，企业能够跨多个机器人提供统一的用户体验。借助机器人网络，企业可以在单个网络中添加多个机器人，从而实现灵活而独立的机器人生命周期管理。该网络向终端用户开放一个统一的接口，并根据用户输入将请求路由到相应的机器人。

当改进的机器人部署到生产环境时，团队可以通过维护机器人并将其添加到网络中来协作创建机器人网络，以满足各种业务需求。开发人员可以通过将多个机器人整合到单个网络中来简化并加快部署和改进。

机器人网络目前仅以美式英语提供。

> **Note**
>
> 目前，机器人网络仅限于一个账户。您无法从其他账户添加机器人。



创建机器人网络

登录 AWS Management Console，并通过以下网址打开 Amazon Lex V2 控制台：https://console.aws.amazon.com/lexv2/home。从侧边菜单中选择机器人网络。要创建机器人网络，必须构建至少一个机器人。

步骤 1：配置机器人网络

1. 在详细信息部分，输入您的网络名称并对其进行可选描述。
2. 在 IAM 权限部分，选择一个 AWS Identity and Access Management (IAM) 角色，该角色为 Amazon Lex V2 提供访问其他 AWS 服务（例如 Amazon CloudWatch）的权限。您可以通过 Amazon Lex V2 创建角色，也可以选择具有 CloudWatch 权限的现有角色。有关更多信息，请参阅 Amazon Lex V2 的身份和访问权限管理。
3. 在儿童在线隐私保护法 (COPPA) 部分，选择相应的回应。有关更多信息，请参阅数据隐私。
4. 在空闲会话超时部分，选择 Amazon Lex V2 保持用户会话打开的持续时间。Amazon Lex V2 会在会话期间保留会话变量，以便您的机器人可以使用相同的变量恢复对话。有关更多信息，请参阅设置会话超时。
5. 在添加语言设置部分，选择一种语音供您的机器人与用户互动。您可以在语音样本中键入短语，然后选择播放来收听语音。
6. 在高级设置部分，可以选择性添加有助于识别机器人的标签。这些标签可用于控制访问和监控资源。有关更多信息，请参阅标记资源。
7. 选择下一步创建机器人网络，然后进入添加机器人。

步骤 2：添加机器人

1. 在机器人部分，选择 + 添加机器人。
2. 此时会弹出添加机器人模式。从机器人下拉菜单中选择要添加的机器人，然后从别名下拉菜单中选择要使用的机器人的别名。

   该别名必须指向机器人的带编号版本，而不是草稿版本。您最多可添加 5 个机器人。一个机器人可以添加到最多 25 个不同的网络中。
3. 选择 + 添加机器人可向您的网络添加更多机器人。要删除机器人，请选择目标机器人旁边的删除。添加完机器人后，选择保存关闭模式。
4. 选择保存以完成网络的创建。

管理你的机器人网络

创建机器人网络后，系统显示网络管理和构建页面。或者，您可以在侧边菜单中选择机器人网络，然后选择要管理的网络名称来访问此页面。

1. 要编辑您的网络信息，请选择详细信息上方的编辑。要删除网络，请选择详细信息上方的删除。

2. 在机器人部分，您可以选择 + 添加机器人来添加更多机器人。您也可以导航到 Amazon Lex V2 控制台侧边菜单的机器人页面，添加机器人。切换目标机器人旁边的单选按钮，然后从操作下拉菜单中选择添加到机器人网络。

   从弹出的模式中的机器人网络下拉菜单中，选择目标网络添加机器人。然后从机器人别名下拉菜单中选择要使用的机器人别名。选择添加，将机器人添加到您选择的网络。
3. 您可以通过切换机器人旁边的单选按钮并选择移除，将机器人从网络中移除。
4. 完成网络配置后，选择右上角的构建来构建您的网络。构建网络可能需要几分钟时间。如果构建成功，页面顶部会显示绿色成功横幅。
5. 建立网络后，您可以选择右上角的测试，聊天窗口将出现在右下角。您可以通过该聊天窗口与网络中的机器人对话，并确保对话流程和转换配置正确。

> **Note**
>
> 如果您在网络中添加、移除或更新机器人，则必须重建网络。

版本

您可以创建不同版本的机器人网络。要管理版本，请从 Amazon Lex V2 控制台的侧边菜单中选择您的网络，然后选择版本。

1. 选择创建版本，为您的机器人网络创建新版本。您还可以选择添加说明。选择创建来创建版本。
2. 当您切换机器人网络版本旁边的单选按钮时，您可以选择将别名与版本关联，将别名指向该版本。
3. 要管理网络的某个版本，请在版本部分选择该版本的名称。在下一页面，您可以编辑版本的详细信息并管理版本中的机器人及其关联别名。

别名

您可以通过别名来部署网络。要管理别名，请从 Amazon Lex V2 控制台的侧边菜单中选择您的网络，然后选择别名。

1. 选择创建别名以创建新别名。
2. 在别名详细信息部分，为别名指定名称和（可选）描述。您可以选择一个版本将别名关联到与版本关联部分，然后在标签部分添加标签。选择创建以创建别名。

3. 要管理网络的别名，请在别名部分选择别名的名称。在下一页面，您可以编辑别名的详细信息并管理其标签、频道集成和基于资源的策略。您还可以查看其与网络版本关联的历史记录。

频道集成

要将您的机器人网络与消息收发平台集成，请从 Amazon Lex V2 控制台的侧边菜单中选择您的机器人网络。然后选择频道集成。

1. 选择添加频道，将您的网络与新频道集成。
2. 在平台部分，从选择平台中选择要部署机器人的平台。将会创建一个 IAM 角色。从 KMS 密钥的下拉菜单中选择一个密钥来保护您的信息。
3. 在集成配置频道，输入名称和（可选）描述。从下拉菜单中选择别名。
4. 从平台获取您的账户 SID 和身份验证令牌，然后填写账户 SID 和身份验证令牌字段。有关更多信息，请参阅集成机器人。
5. 选择创建以完成频道集成。

> **Note**
>
> 目前，Amazon Connect 语音或聊天不支持机器人网络。

部署机器人

创建并测试完成后，可以部署机器人与您的客户进行交互。在本节中，学习如何在更新后创建机器人版本。当准备部署机器人时，使用别名表示不同版本的机器人。了解如何将您的机器人与消息收发平台、移动应用程序和网站集成。

主题

版本控制和别名

Amazon Lex V2 支持创建机器人和机器人网络的版本和别名，以便您控制客户端应用程序使用的实现方案。版本相当于您的工作的带编号快照。您可以为您希望向客户提供的机器人版本指定别名。创建新版本之前，您可以继续更新机器人的 Draft 版本，而不会影响用户体验。

版本

Amazon Lex V2 支持创建机器人的版本，以便您控制客户端应用程序使用的实现方案。版本是您工作的带编号快照，您可以创建版本以用于您的工作流的不同阶段，如开发、测试部署和生产。

草稿版本

当您创建 Amazon Lex V2 机器人时，只有一个版本，即 Draft 版本。

Draft 是您机器人的工作副本。您只能更新 Draft 版本，直到您创建第一个版本，Draft 是您所拥有的唯一机器人版本。

您机器人的 Draft 版本与 TestBotAlias 相关联。TestBotAlias 只应用于手动测试。Amazon Lex V2 限制可对机器人的 TestBotAlias 别名进行的运行时请求的数量。

创建版本

当您对 Amazon Lex V2 机器人进行版本控制时，您创建该机器人的带编号快照，从而能够以该机器人在创建该版本时的状态来使用该机器人。创建数字版本后，该版本在您继续处理应用程序的草稿版本时将保持不变。

创建版本时，您可以选择包含在该版本中的区域设置。您无需选择机器人中的所有区域设置。此外，在创建版本时，可以从先前版本中选择区域设置。例如，如果您有三个版本的机器人，则在创建第四版本时，可以从 `Draft` 版本中选择一个区域设置，并且从第二版中选择一个区域设置。

如果从 `Draft` 版本中删除区域设置，则该区域设置不会从带编号版本中删除。

如果机器人版本在六个月内未使用，Amazon Lex V2 会将该版本标记为非活动状态。当版本处于非活动状态时，则无法对机器人使用运行时操作。要使机器人处于活动状态，请重新构建与该版本关联的所有语言。

更新 Amazon Lex V2 机器人

您只能更新 Amazon Lex V2 机器人的 `Draft` 版本。各版本无法更改。您在控制台中或者使用 CreateBotVersion 操作更新资源后，随时可以创建新的版本。

删除 Amazon Lex V2 机器人或版本

Amazon Lex V2 支持使用控制台或以下 API 操作之一删除机器人或版本：

- DeleteBot
- DeleteBotVersion

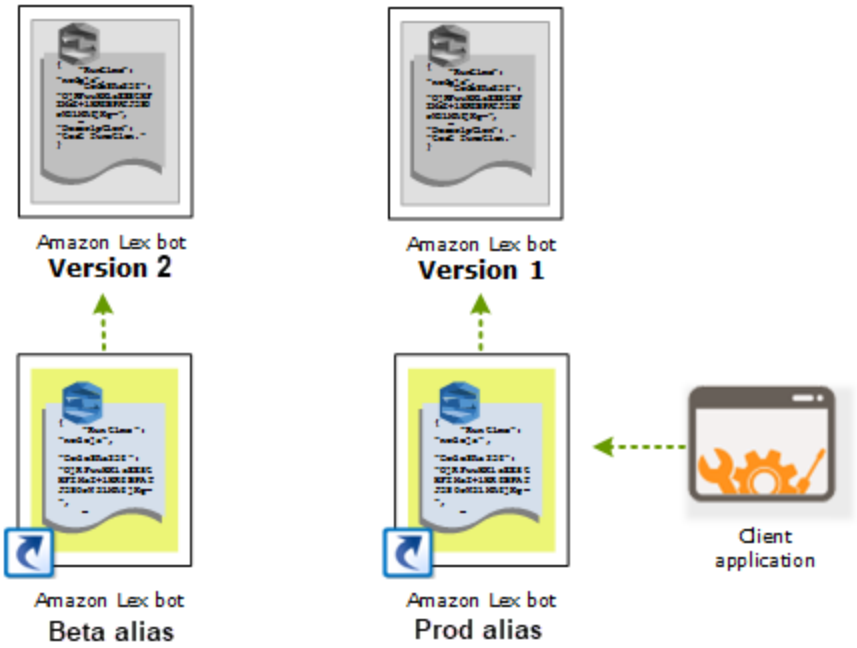
Aliases

Amazon Lex V2 机器人支持别名。别名 是指向自动程序特定版本的指针。利用别名，您可以轻松更新您的客户端应用程序正在使用的版本。例如，您可以将别名指向您自动程序的版本 1。当您准备好更新机器人时，可以创建版本 2，然后更改别名以指向新版本。由于您的应用程序使用的是别名而不是特定版本，因此您的所有客户端无需进行更新即可获得新功能。

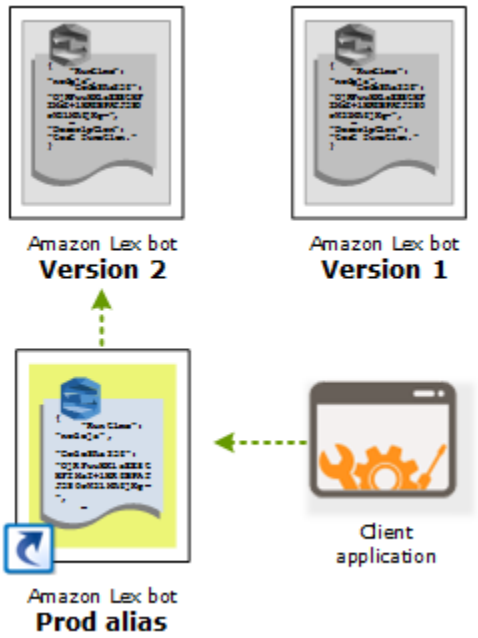
别名是指向 Amazon Lex V2 机器人特定版本的指针。使用别名可允许客户端应用程序使用特定版本的自动程序，而应用程序无需跟踪它是哪个版本。

创建机器人时，Amazon Lex V2 会创建一个名为 `TestBotAlias` 的别名，以便您用于测试机器人。`TestBotAlias` 别名始终与您机器人的 `Draft` 版本相关联。您只能使用 `TestBotAlias` 别名进行测试，Amazon Lex V2 限制可对该别名进行的运行时请求的数量。

以下示例显示 Amazon Lex V2 机器人的两个版本，即版本 1 和版本 2。这两个自动程序版本各有一个关联的别名，分别为 BETA 和 PROD。客户端应用程序使用 PROD 别名来访问自动程序。



当您创建机器人的第二版本时，您可以使用控制台或 UpdateBotAlias 操作更新别名以指向机器人的新版本。当您更改别名时，您的所有客户端应用程序都将使用新版本。如果新版本出了问题，您只需通过更改别名以指向之前的版本，即可回滚到该版本。



当您将客户端应用程序设置为调用 Amazon Lex Runtime V2 API 以允许客户端与您的机器人交互时，请使用指向您希望客户端使用的版本的别名。

> **Note**
>
> 尽管您可以在控制台中测试机器人的 Draft 版本，但建议在您将机器人与您的客户端应用程序集成时，首先创建一个版本，然后创建一个指向该版本的别名。本部分中说明了在您的客户端应用程序中使用别名的原因。当您更新别名时，Amazon Lex V2 将在所有正在进行的会话中使用当前版本。新会话将使用新版本。

使用 Java 应用程序与 Amazon Lex V2 机器人进行交互

AWS SDK for Java 2.0 提供了一个接口，可用于在 Java 应用程序中与机器人进行交互。使用 SDK for Java 来为用户构建客户端应用程序。

以下应用程序与您在练习 1：根据示例创建机器人中创建的 OrderFlowers 机器人进行交互。使用 SDK for Java 的 LexRuntimeV2Client 来调用 RecognizeText 操作执行与机器人的对话。

该对话的输出如下所示：

```
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.
```

有关在客户端应用程序与 Amazon Lex V2 机器人之间发送的 JSON 结构，请参阅练习 2：查看对话流程。

要运行应用程序，您需要提供以下信息：

- “botId”：您在创建机器人时为该机器人指定的标识符。您可以在 Amazon Lex V2 控制台的机器人设置页面上查看该机器人 ID。

- “botAliasId”：您在创建机器人别名时为该机器人别名指定的标识符。您可以在 Amazon Lex V2 控制台的别名页面上查看该机器人别名 ID。如果列表中未显示别名 ID，请选择右上角的齿轮图标并打开别名 ID。
- “localeId”：您用于机器人的区域设置的标识符。有关区域设置列表，请参阅Amazon Lex V2 支持的语言和区域设置。
- “accessKey”和“secretKey”：您账户的身份验证密钥。如果没有密钥集，请使用 AWS Identity and Access Management 控制台创建密钥。
- “sessionId”：与 Amazon Lex V2 机器人会话的标识符。在这种情况下，代码使用随机的 UUID。
- “region”：如果您的机器人不在美国东部（弗吉尼亚州北部）区域，请务必更改“区域”设置。

应用程序使用名为 `getRecognizeTextRequest` 的函数创建对机器人的单独请求。该函数使用必要参数生成一个请求以发送到 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;
    }
}
```

全球弹性

> **Note**
>
> 此功能仅适用于在美国东部（弗吉尼亚北部）和美国西部（俄勒冈）地区创建的 Amazon Connect 和 Amazon Lex V2 实例。
> 要获得对此功能的访问权限，请联系您的 Amazon Connect 解决方案架构师或技术客户经理。

全球弹性允许您在辅助区域复制机器人。通过在两个区域中自动复制用户的机器人，可以使辅助区域处于活动状态。如果发生区域性中断，您将有一个备用区域。全球弹性激活后，创建的新机器人将在第二个 AWS 区域复制。

激活此功能后，您可以近乎实时地在配对 AWS 区域中自动复制 Amazon Lex V2 机器人及其资源、版本和别名。使用此功能，您可以监控原始和副本机器人的版本号，以确保机器人副本与原始机器人保持同步。启用复制后，您可以激活要在其中复制机器人的预先确定的 AWS 区域（区域基于预先确定的配对）。源区域中对源机器人进行的任何更新都会自动更新到第二个区域中复制的机器人。

> **Note**
>
> 激活全球弹性后，只有在激活该功能后创建的机器人、版本和别名才会在复制的区域中复制。之前创建的机器人、版本和别名将不会出现在复制的区域中。标识的第二个区域是只读的，并且是预先确定的成对区域。机器人更新仅限于最初创建机器人的区域。

有关使用全球弹性的其他信息：

- 全球弹性目前仅适用于预先确定的区域对。

us-east-1	us-west-2
eu-west-2	eu-central-1

- 您可以创建任何 Amazon Lex V2 机器人的副本。启用 “全球弹性” 后，您必须为机器人创建新版本和新别名。
- 在全球弹性中启用的别名只能与支持全球弹性的版本相关联。

限制:

- Global Selisiency 不会复制使用使用 LLM 的老虎机创建的机器人，例如 CFAQ 和 Utterance Generation。
- Global Selisiency 不会复制机器人网络，但任何属于机器人网络的机器人仍然可以单独复制。

主题

复制机器人和管理机器人副本的权限

如果 IAM 角色附加了AmazonLexFullAccess策略，则它可以创建和管理机器人副本。

如果您希望创建具有最低全球弹性权限的角色，请使用以下策略，其中包含以下语句。

- 访问用于机器人复制的 Amazon Lex V2 服务相关角色的权限。
- 允许 Amazon Lex V2 代表您创建用于机器人复制的服务相关角色的权限。
- 调用机器人复制 API 的权限。

```
{
    "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/*"
            ]
        }
    ]
}
```

您可以通过修改权限来进一步限制权限，如下所示。

- 将 * 替换为特定的机器人或机器人别名 ID，将权限限制为特定的机器人或机器人别名。
- 使用lex BotReplica操作的子集将角色限制为特定操作。

有关示例，请参阅允许用户创建和查看机器人副本，但不允许将其删除。

部署全球弹性

全球弹性信息面板

您可以在 “全球弹性” 面板中访问以下信息：

- 来源详情-有关您的机器人的源区域、副本类型、启用复制的日期和上次创建版本的信息。使用这些信息来跟踪你的机器人的迭代次数。
- 复制详细信息-创建机器人副本后，您可以跟踪复制的区域、副本类型、上次版本同步日期和上次复制的版本。使用此信息来跟踪您的机器人副本的同步。
- 来源区域-启用全球弹性的区域。您可以在源区域进行更改，以便在两个区域中复制机器人。
- 副本类型-指明机器人是只读还是能够根据区域进行读写。
- 副本区域 — 用于复制源机器人以实现全球弹性的辅助区域。全球弹性目前仅适用于IAD/PDX和LDN/FRA区域组合。
- 启用复制的日期-启用机器人副本的日期和时间。
- 上次创建的版本-源区域中与副本关联的最后一个机器人版本。

实现全球弹性

> **Note**
>
> 此功能仅适用于在美国东部（弗吉尼亚北部）和美国西部（俄勒冈）地区创建的 Amazon Connect 和 Amazon Lex V2 实例。
> 要获得对此功能的访问权限，请联系您的 Amazon Connect 解决方案架构师或技术客户经理。

在 Amazon Lex V2 控制台中激活全球弹性之前，必须确保启用机器人复制的用户有权创建服务关联角色 (SLR)。调用时 CreateReplica ，全球弹性将使用这些 FAS 凭证在启用的账户中创建 SLR。有关在 Amazon Lex V2 中设置 SLR 以实现全球弹性的更多信息，请参阅 AW S 托管策略：。AmazonLexFullAccess

激活全球弹性并为第二个区域设置机器人复制：

1. 登录 AWS 管理控制台并打开 Amazon Lex 控制台，网址为 https://console.aws.amazon.com/lex/。
2. 从左侧导航面板的机器人导航栏中选择要复制的机器人。
3. 选择部署 > 全球弹性。
4. 选择窗口右上角的 “创建副本” 按钮，创建机器人的草稿版本。

> **Note**
>
> 检查以确保在辅助区域中没有任何与您要复制的机器人同名的机器人。（您的机器人必须具有唯一的名称）。

5. 转至 “全球弹性”，单击 “创建副本”-此操作将创建机器人的草稿版本。（除了查看状态或查看 future 版本的详细信息外，您无需返回 “全球弹性” 选项卡）。

> **Note**
>
> 您还可以通过转到别名并选择为启用全球弹性的机器人创建新别名来创建用于在 Global Serimency 中复制的别名机器人。只有在启用复制后创建的别名才会被复制。

6. 前往别名-为启用全球弹性的机器人创建新别名。只有在启用复制后创建的别名才能复制。
7. 转至版本-为启用全球弹性的机器人创建新版本。只有在启用复制后创建的版本才能复制。

> **Note**
>
> 客户仍然可以完全控制其基于资源的策略和复制机器人的标签。需要在两个区域部署具有相同标识符的 Lambda 函数和 CloudWatch 日志组。用户无需在副本区域中再次关联 lambda 函数。

禁用全球弹性

您可以随时通过选择 “禁用全球弹性” 按钮来禁用全球弹性。此操作可阻止您的源机器人以及与之关联的任何别名和版本在其他地区复制。

使用具有全球弹性的 API

您可以使用以下 API 在 “全球弹性” 中进行 API 调用。有关全球弹性 API 和 Amazon Lex V2 的更多信息，请参阅 Amazon Lex V2 API 指南。

- CreateBotReplica

  启用全球弹性并创建复制的机器人。需要 `replicaRegion`。

  有关更多信息，请参阅 Lex API 指南CreateBotReplica中的。

- DeleteBotReplica

  禁用全局弹性并删除复制的机器人。需要使用 `replicaRegion` 和 `botId`。

  有关更多信息，请参阅 Lex API 指南DeleteBotReplica中的。
- ListBotReplicas

  列出辅助区域中复制的机器人。需要 `botId`。

  有关更多信息，请参阅 Lex API 指南ListBotReplicas中的。
- DescribeBotReplica

  复制的机器人的信息摘要。需要使用 `replicaRegion` 和 `botId`。

  有关更多信息，请参阅 Lex API 指南DescribeBotReplica中的。

将 Amazon Lex V2 机器人与消息收发平台集成

本节介绍如何将 Amazon Lex V2 机器人和 Facebook、Slack 和 Twilio 消息收发平台集成。如果您还没有 Amazon Lex V2 机器人，请创建一个。在本主题中，我们假定您使用的是 练习 1：根据示例创建机器人 中创建的机器人。当然，您也可以使用其他机器人。

> **Note**
>
> 在存储你的 Facebook、Slack 或 Twilio 配置时，Amazon Lex V2 使用 AWS KMS key 来加密信息。首次创建通往其中一个消息传递平台的频道时，Amazon Lex V2 会在您的 AWS 账户中创建默认的客户托管密钥 (`aws/lex`)，或者您可以选择自己的客户托管密钥。Amazon Lex V2 仅支持对称密钥。有关更多信息，请参阅 AWS Key Management Service 开发人员指南。

当消息收发平台向 Amazon Lex V2 发送请求时，它将包含平台特定信息作为您的 Lambda 函数的请求属性。使用该属性可自定义您的机器人的行为方式。有关更多信息，请参阅设置请求属性。

通用请求属性

属性	描述
x-amz-lex:channels: 平台	下列值之一： • `Facebook`

属性	描述
	• Slack • Twilio

将 Amazon Lex V2 机器人与 Facebook Messenger 集成

您可以将 Amazon Lex V2 机器人托管在 Facebook Messenger 中。在此情况下，Facebook 用户可以与您的机器人交互以履行意图。

在开始之前，您需要访问 https://developers.facebook.com 注册 Facebook 开发人员账户。

请执行下列步骤：

主题

第 1 步：创建 Facebook 应用程序

在 Facebook 开发人员门户上，创建一个 Facebook 应用程序和一个 Facebook 页面。

要创建 Facebook 应用程序，请执行以下操作：

1. 打开 https://developers.facebook.com/apps
2. 选择 Create App (创建应用程序)。
3. 在创建应用页面中，选择 Business，然后选择继续。
4. 在添加应用名称、应用联系邮箱和业务帐户字段中，选择适用于您的应用程序的值。选择创建应用以继续。
5. 在为应用添加产品部分中，选择 Messenger 磁贴中的设置。
6. 在访问口令部分，选择添加或移除主页。
7. 选择要与您的应用程序配合使用的页面，然后选择继续。
8. 在可管理的业务功能中，保留默认值，然后选择完成。

9. 在确认页面上，请选择确认。
10. 在访问口令部分，选择生成口令，然后复制该令牌。您可以在 Amazon Lex V2 控制台中输入此令牌。
11. 在左侧菜单中，选择设置 > 基本。
12. 对于应用密钥，选择显示，然后复制密钥。您可以在 Amazon Lex V2 控制台中输入此令牌。

下一步

第 2 步：将 Facebook Messenger 与 Amazon Lex V2 机器人集成

第 2 步：将 Facebook Messenger 与 Amazon Lex V2 机器人集成

在此步骤中，将您的 Amazon Lex V2 机器人与 Facebook 关联起来。

1. 登录到 AWS Management Console，并通过以下网址打开 Amazon Lex 控制台：https://console.aws.amazon.com/lex/。
2. 从机器人列表中选择您创建的 Amazon Lex V2 机器人。
3. 在左侧菜单中，选择频道集成，然后选择添加频道。
4. 在创建频道中，执行以下操作：
   a. 对于平台，请选择 Facebook。
   b. 对于标识策略，请选择 AWS KMS 密钥以保护频道信息。默认密钥是 Amazon Lex V2 提供的密钥。
   c. 对于集成配置，指定频道的名称和描述（可选）。选择指向要使用的机器人版本的别名，然后选择该频道支持的语言。
   d. 对于其他配置，输入以下内容：
      - 别名：标识正在调用 Amazon Lex V2 的应用程序的字符串。您可以使用任何字符串。记下该字符串，然后访问 Facebook 开发人员控制台并输入该字符串。
      - 页面访问令牌：您从 Facebook 开发人员控制台复制的页面访问令牌。
      - 应用程序密钥：您从 Facebook 开发人员控制台复制的密钥。
   e. 选择 Create（创建）。
   f. Amazon Lex V2 会显示您的机器人的频道列表。从列表中选择您刚刚创建的频道。
   g. 记下回调 URL中显示的回调 URL。记下该 URL，然后访问 Facebook 开发人员控制台并输入该 URL。

下一步

[第 3 步：完成 Facebook 集成]

第 3 步：完成 Facebook 集成

在此步骤中，使用 Facebook 开发人员控制台完成与 Amazon Lex V2 的集成。

要完成 Facebook Messenger 集成，请执行以下操作：

1. 打开 https://developers.facebook.com/apps
2. 从应用程序列表中，选择要与 Facebook Messenger 集成的应用程序。
3. 在左侧菜单中，选择 Messenger，然后选择设置。
4. 在 Webhooks 部分中：
    a. 选择添加回调网址。
    b. 在编辑回调网址中，输入以下内容：
        - 回调网址：输入您从 Amazon Lex V2 控制台记录下的回调 URL。
        - 验证口令：输入您在 Amazon Lex V2 控制台中输入的别名。
    c. 选择 Verify and Save。
    d. 在页面旁边的 Webhooks 下选择添加订阅。
    e. 在弹出的窗口中，选择 `messages`，然后点击保存。

下一步

[第 4 步：测试集成]

第 4 步：测试集成

您现在即可在 Facebook Messenger 中发起与 Amazon Lex V2 机器人的对话。

要测试 Facebook Messenger 与 Amazon Lex V2 机器人的集成，请执行以下操作：

1. 打开您在第 1 步中与机器人关联的 Facebook 页面。
2. 在 Messenger 窗口中，使用 练习 1：根据示例创建机器人 中提供的测试言语。

将 Amazon Lex V2 机器人与 Slack 集成

本主题提供有关将 Amazon Lex V2 机器人与 Slack 消息收发应用程序集成的指南。请执行下列步骤：

主题

第 1 步：注册 Slack 并创建 Slack 团队

注册 Slack 账户并创建 Slack 团队。有关说明，请参阅使用 Slack。在下一部分中，您将创建可供所有 Slack 团队安装的 Slack 应用程序。

下一步

第 2 步：创建 Slack 应用程序

第 2 步：创建 Slack 应用程序

请在此部分中执行以下操作：

1. 在 Slack API 控制台中创建 Slack 应用程序。
2. 配置该应用程序，以将交互式消息收发功能添加到机器人中。

有关应用程序凭证（客户端 ID、客户端密钥和验证令牌），请参见本节结尾。在下一步中，您将使用此信息将机器人集成到 Amazon Lex V2 控制台中。

要创建 Slack 应用程序，请执行以下操作：

1. 通过以下网址登录 Slack API 控制台：https://api.slack.com。
2. 创建一个应用程序。

   在您成功创建应用程序以后，Slack 会显示应用程序的 Basic Information 页面。
3. 按如下所示配置应用程序的功能：

- 在左侧菜单中，选择互动性和快捷方式。
  - 选择打开交互式组件的开关。
  - 在 Request URL 框中，指定任何有效的 URL。例如，您可以使用 **`https://slack.com`**。

    > **Note**
    >
    > 目前，请输入任何有效的 URL，以便获得在下一步中需要的验证令牌。您在 Amazon Lex 控制台中添加机器人频道关联后，将需要更新此 URL。

  - 选择 Save Changes。

4. 在左侧菜单的 Settings 中，选择 Basic Information。记录以下应用程序凭证：
   - 客户端 ID
   - 客户端密钥
   - 验证令牌

下一步

[第 3 步：将 Slack 应用程序与 Amazon Lex V2 机器人集成]

第 3 步：将 Slack 应用程序与 Amazon Lex V2 机器人集成

在本节中，将您创建的 Slack 应用程序与您使用频道集成创建的 Amazon Lex V2 机器人集成。

1. 登录到 AWS Management Console，并通过以下网址打开 Amazon Lex 控制台：https://console.aws.amazon.com/lex/。
2. 从机器人列表中选择您创建的 Amazon Lex V2 机器人。
3. 在左侧菜单中，选择频道集成，然后选择添加频道。
4. 在创建频道中，执行以下操作：
   a. 对于平台，请选择 Slack。
   b. 对于标识策略，请选择 AWS KMS 密钥以保护频道信息。默认密钥是 Amazon Lex V2 提供的密钥。
   c. 对于集成配置，指定频道的名称和描述（可选）。选择指向要使用的机器人版本的别名，然后选择该频道支持的语言。

> **Note**
>
> 如果您的机器人支持多种语言，则必须为每种语言创建不同的频道以及不同的应用程序。

d. 对于其他配置，输入以下内容：

- 客户端 ID：输入来自 Slack 的客户端 ID。
- 客户端密钥：输入 Slack 的客户端密钥。
- 验证令牌：输入来自 Slack 的验证令牌。
- 成功页面 URL：用户通过身份验证后 Slack 应打开的页面的 URL。通常，此项可留空。

5. 选择创建以创建频道。
6. Amazon Lex V2 会显示您的机器人的频道列表。从列表中选择您刚刚创建的频道。
7. 在回调 URL中，记录端点和 OAuth 端点。

下一步

[第 4 步：完成 Slack 集成]

第 4 步：完成 Slack 集成

在本部分中，使用 Slack API 控制台完成与 Slack 应用程序的集成。

1. 通过以下网址登录 Slack API 控制台：https://api.slack.com。选择您在第 2 步：创建 Slack 应用程序中创建的应用程序。
2. 按如下所述更新 OAuth & Permissions 功能：

a. 在左侧菜单中，选择 OAuth & Permissions (OAuth 和权限)。

b. 在重定向 URL 部分中，添加 Amazon Lex 上一步中提供的 OAuth 端点。选择 Add，然后选择 Save URLs。

c. 在机器人令牌作用域部分中，使用添加 OAuth 作用域按钮添加两个权限。用以下文本筛选列表：

- **`chat:write`**
- **`team:read`**

3. 通过将请求 URL 值更新为 Amazon Lex 在上一步中提供的端点来更新互动性和快捷方式功能。输入您在第 3 步中保存的端点，然后选择保存更改。
4. 按如下所述订阅 Event Subscriptions 功能：
   - 通过选择 On 选项启用事件。
   - 将请求 URL 值设置为 Amazon Lex 在上一步中提供的端点。
   - 在订阅机器人事件部分中，选择添加机器人用户事件，然后添加 **`message.im`** 机器人事件，以便启用最终用户与 Slack 机器人之间的直接消息收发。
   - 保存更改。
5. 从消息选项卡中启用消息的发送，如下所示：
   - 从左侧菜单中，选择应用程序主页。
   - 在显示选项卡部分中，选择允许用户从消息选项卡中发送 Slash 命令和消息。
6. 在设置下选择管理分发。选择 Add to Slack 以安装应用程序。如果您已通过多个工作区的身份验证，请先从右上角的下拉列表中选择正确的工作区。接下来，选择允许以授权机器人回复消息。

> **Note**
>
> 如果您后期对 Slack 应用程序设置进行任何更改，则必须重新执行此子步骤。

下一步

步骤 5：测试集成

步骤 5：测试集成

现在，使用浏览器窗口来测试 Slack 与 Amazon Lex V2 机器人的集成。

要测试您的 Slack 应用程序，请执行以下操作：

1. 打开 Slack。在左侧菜单的直接消息部分中，选择您的机器人。如果看不到您的自动程序，请选择 Direct Messages 旁边的加号图标 (+) 进行搜索。
2. 使用您的 Slack 应用程序进行聊天。您的机器人会对消息做出响应。

   如果您是使用 练习 1：根据示例创建机器人 创建的机器人，可以使用该练习中的示例对话。

将 Amazon Lex V2 机器人与 Twilio SMS 集成

本主题提供将 Amazon Lex V2 机器人与 Twilio Simple Messaging Service (SMS) 集成的说明。请执行下列步骤：

主题

第 1 步：创建 Twilio SMS 账户

注册 Twilio 账户，并记录以下账户信息：

- ACCOUNT SID
- AUTH TOKEN

有关注册指南，请参阅 https://www.twilio.com/console。

下一步

第 2 步：将 Twilio 消息收发服务端点与 Amazon Lex V2 机器人集成

第 2 步：将 Twilio 消息收发服务端点与 Amazon Lex V2 机器人集成

1. 登录到 AWS Management Console，并通过以下网址打开 Amazon Lex 控制台：https://console.aws.amazon.com/lex/。
2. 从机器人列表中选择您创建的 Amazon Lex V2 机器人。
3. 在左侧菜单中，选择频道集成，然后选择添加频道。
4. 在创建频道中，执行以下操作：

    a. 对于平台，选择 Twilio。

    b. 对于标识策略，请选择 AWS KMS 密钥以保护频道信息。默认密钥是 Amazon Lex V2 提供的密钥。

    c. 对于集成配置，指定频道的名称和描述（可选）。选择指向要使用的机器人版本的别名，然后选择该频道支持的语言。

d. 要进行其他配置，请在 Twilio 控制面板中输入账户 SID 和身份验证令牌。

5. 选择 Create（创建）。
6. 从频道列表中选择您刚刚创建的频道。
7. 复制回调 URL。

下一步

[第 3 步：完成 Twilio 集成]

第 3 步：完成 Twilio 集成

使用 Twilio 控制台完成 Amazon Lex V2 机器人与 Twilio SMS 的集成。

1. 访问 https://www.twilio.com/console 以打开 Twilio 控制台。
2. 从左侧菜单中选择 All Products & Services，然后选择 Phone Number。
3. 如果您有电话号码，请选择该电话号码。如果您没有电话号码，请选择 Buy a Number 以获取电话号码。
4. 在 Messaging 部分的 A MESSAGE COMES IN 中，输入来自 Amazon Lex V2 控制台的回调 URL。
5. 选择 Save（保存）。

下一步

[第 4 步：测试集成]

第 4 步：测试集成

使用您的手机来测试 Twilio SMS 与您的自动程序之间的集成。使用您的手机向 Twilio 号码发送消息。

如果您是使用 练习 1：根据示例创建机器人 创建的机器人，可以使用该练习中的示例对话。

将 Amazon Lex V2 机器人与联络中心集成

您可以将 Amazon Lex V2 机器人与联络中心集成，从而通过 Amazon Lex V2 流式处理 API 启用自助服务用例。将这些机器人用作电话上的交互式语音应答 (IVR) 客服或集成到联络中心的基于文本的聊天机器人。有关流式处理 API 的更多信息，请参阅 流式传输到 Amazon Lex V2 机器人。

通过流式处理 API，您可以启用以下功能：

- 中断 ("barge-in")：呼叫者可以在提示完成之前打断机器人并回答问题。有关更多信息，请参阅允许用户打断机器人。
- 等待并继续：如果呼叫者在通话期间需要时间检索其他信息（例如信用卡号或预订 ID），则可以指示机器人等待。有关更多信息，请参阅允许机器人等待用户提供更多信息。
- DTMF 支持：呼叫者可以通过语音或 DTMF 交替提供信息。
- SSML 支持：您可以使用 SSML 标签配置 Amazon Lex V2 机器人提示，以便更好地控制从文本生成语音。有关更多信息，请参阅 Amazon Polly 开发者指南中的由 SSML 文档生成语音。
- 可配置的超时时间：您可以配置等待客户结束讲话的时间，然后 Amazon Lex V2 会收集他们的语音输入，例如回答是或否的问题，或提供日期或信用卡号码等信息。有关更多信息，请参阅配置捕获用户输入的超时。
- 履行进度更新：您可以配置机器人在业务逻辑执行期间，根据履行状态以多条消息进行回复，从而达成意图履行。您可以设置机器人在履行开始和完成时发送响应消息，并为长时间运行的 Lambda 函数提供定期更新。有关更多信息，请参阅配置履行进度更新。

Amazon Chime SDK

使用 Amazon Chime SDK 将实时音频、视频、屏幕共享和消息收发功能添加到您的 Web 或移动应用程序中。Amazon Chime SDK 提供公共交换电话网 (PSTN) 音频服务，以便您使用 AWS Lambda 函数来构建自定义电话应用程序。

Amazon Chime PSTN 音频已与 Amazon Lex V2 集成。借助此集成，您在联络中心访问作为交互式语音响应 (IVR) 系统的 Amazon Lex V2 机器人，以进行音频交互。在以下场景中，以此来使用 PSTN 音频服务集成 Amazon Lex V2。

联络中心集成：您可以使用 Amazon Chime Voice Connector 和 Amazon Chime SDK PSTN 音频服务来访问 Amazon Lex V2 机器人。在任何使用会话初始协议 (SIP) 进行语音通信的联络中心应用程序中使用该等集成。借助此集成，您现有的本地或基于云的联络中心可实现自然语言语音对话体验，并支持 SIP。有关支持的联络中心平台列表，请参阅 Amazon Chime Voice Connector 资源。

下图显示了使用 SIP 的联络中心与 Amazon Lex V2 之间的集成。

直接电话支持：您可以构建自定义 IVR 解决方案，以便使用 Amazon Chime SDK 中配置的电话号码直接访问 Amazon Lex V2 机器人。

有关更多信息，请参阅 Amazon Chime SDK 指南 中的以下主题：

- 使用 Amazon Chime Voice Connector 进行 SIP 集成
- 使用 Amazon Chime SDK PSTN 音频服务
- Amazon Chime PSTN 音频与 Amazon Lex V2 的集成

当 Amazon Chime SDK 向 Amazon Lex V2 发送请求时，会将平台特定信息包含在您的 Lambda 函数和对话日志中。通过此信息来确定正在向您的机器人发送流量的联络中心应用程序。

通用请求属性	Value
x-amz-lex:channels:platform	`Amazon Chime SDK PSTN Audio`

Amazon Connect

Amazon Connect 是一个全渠道云联系中心。只需执行几个步骤，便能设置联系中心，添加任意位置的客服，并开始与客户接洽。有关更多信息，请参阅 Amazon Connect 管理员指南中的开始使用 Amazon Connect。

您可以使用全渠道通信为客户创建个性化体验。例如，您可以根据客户偏好和预计等待时间提供聊天和语音联系方式供选择。同时，客服可以在一个界面处理所有客户。例如，他们可以与客户聊天，创建任务，并响应传送给他们的任务。

您可以使用 Amazon Connect 与客户进行音频互动，或使用 Amazon Connect Chat 进行纯文本互动。

有关更多信息，请参阅 Amazon Connect 管理员指南中的以下主题：

- 什么是 Amazon Connect
- 添加 Amazon Lex V2 机器人
- Amazon Connect 获取客户输入联系数据块

当联络中心向 Amazon Lex V2 发送请求时，会将平台特定信息作为请求属性添加到您的 Lambda 函数和对话日志中。通过此信息来确定哪个联络中心应用程序正在向您的机器人发送流量。

通用请求属性

属性	值
x-amz-lex:channels: 平台	下列值之一： • `Connect` • `Connect Chat`

Genesys Cloud

Genesys Cloud 是一套用于企业通信、协作和联络中心管理的云服务。Genesys Cloud 建立在分布式云环境之上， AWS 并使用分布式云环境，为工作中的组织提供安全访问。

有关更多信息，请参阅 Genesys Cloud 网站上的以下页面：

- 关于 Genesys Cloud 联络中心
- 关于 Amazon Lex V2 集成

当联络中心向 Amazon Lex V2 发送请求时，会将平台特定信息作为请求属性添加到您的 Lambda 函数和对话日志中。通过此信息来确定哪个联络中心应用程序正在向您的机器人发送流量。

通用请求属性

属性	值
x-amz-lex:channels: 平台	• `Genesys Cloud`

了解更多

- 通过 Amazon Lex 和 Genesys Cloud 为您的联络中心提供支持

管理对话

构建机器人后，您可以将您的客户端应用程序与 Amazon Lex V2 运行时操作集成，以便与您的机器人进行对话。

用户启动与您的机器人的对话时，Amazon Lex V2 会创建一个会话。会话封装了您的应用程序和机器人之间交换的信息。有关更多信息，请参阅使用 Amazon Lex V2 API 管理会话。

典型的对话涉及用户和机器人之间的来回流程。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.
```

当您使用 RecognizeText 或 RecognizeUtterance 操作时，必须在客户端应用程序中管理对话。当您使用 StartConversation 操作时，Amazon Lex V2 会为您管理对话。

要管理对话，您必须向机器人发送用户言语，直到对话达到合乎逻辑的结局。当前对话是在会话状态下捕获的。每次用户言语后，会话状态都会更新。会话状态包含对话的当前状态，并由机器人在对每个用户言语的响应中返回。

对话可以处于以下任意状态：

- ElicitIntent：表示机器人尚未确定用户的意图。
- ElicitSlot：表示机器人已检测到用户的意图并正在收集履行意图所需的信息。
- ConfirmIntent：表示机器人正在等待用户确认收集的信息是否正确。
- Closed：表示用户的意图已完成，并且与机器人的对话已达到合乎逻辑的结局。

用户可以在第一个意图完成后指定新意图。有关更多信息，请参阅管理对话上下文。

意图可以具有以下状态：

- 进行中：表示机器人正在收集完成意图所需的信息。这与 `ElicitSlot` 对话状态有关。
- 等待中：表示当机器人要求提供特定插槽的信息时，用户请求机器人等待。
- 已履行：表示与意图关联的 Lambda 函数中的业务逻辑成功运行。
- 履行就绪：表示机器人收集了履行意图所需的所有信息，并且客户端应用程序可以运行履行业务逻辑。
- 失败：表示意图达成失败。

请参阅以下主题，了解如何通过 Amazon Lex V2 API 来管理您的机器人与用户之间的对话上下文和会话。

主题

管理对话上下文

对话上下文是用户、您的客户端应用程序或 Lambda 函数向 Amazon Lex 机器人提供的用于履行意图的信息。对话上下文包括用户提供的槽数据、客户端应用程序设置的请求属性以及客户端应用程序和 Lambda 函数创建的会话属性。

主题

设置意图上下文

您可以设置 Amazon Lex 根据上下文触发意图。上下文是一个在定义机器人时与意图相关联的状态变量。通过控制台或 CreateIntent 操作创建意图时，您可以为该意图配置上下文。目前，只支持在英语（美国）(en-US) 语言环境中使用上下文。

上下文有两种类型的关系，即输出上下文和输入上下文。当关联的意图履行时，输出上下文就会变为活动状态。在 RecognizeText 或 RecognizeUtterance 操作的响应中，会将输出上下文返回到您的应用程序，并为当前会话设置上下文。上下文被激活后，会保持活动状态，直到定义上下文时配置的回合数或时间限制。

输入上下文指定了可以识别意图的条件。只有当对话的所有输入上下文都处于活动状态时，才能识别出意图。没有输入上下文的意图始终可以被识别。

Amazon Lex 自动管理通过使用输出上下文履行意图而激活的上下文的生命周期。您还可以在调用 `RecognizeText` 或 `RecognizeUtterance` 操作时设置活动上下文。

您也可以通过 Lambda 函数为意图设置对话上下文。将 Amazon Lex 的输出上下文发送到 Lambda 函数输入事件。Lambda 函数可以在其响应中发送上下文。有关更多信息，请参阅使用 AWS Lambda 函数启用自定义逻辑。

例如，假设您打算预订一辆配置返回名为 “book_car_fulfilled”的输出上下文的租车。履行意图后，Amazon Lex 会将输出上下文变量设置为 “book_car_fulfilled”。由于“book_car_fulfilled”上下文处于活动状态，因此，只要用户的话语被识别为试图引发该意图，就可以考虑将“book_car_fulfilled”上下文设置为输入上下文的意图进行识别。您可以将其用于只有在预订车辆后才有意义的意图，例如，通过电子邮件发送收据或修改预订。

输出上下文

履行意图时，Amazon Lex 会激活意图的输出上下文。您可以通过输出上下文来控制哪些意图符合当前意图条件。

每个上下文都有一个在会话中维护的参数列表。这些参数是已实现意图的槽值。您可以使用这些参数为其他意图预填充槽值。有关更多信息，请参阅使用默认槽值。

当您通过控制台或 CreateIntent 操作创建意图时，您可以配置输出上下文。您可以为一个意图配置多个输出上下文。履行意图时，所有输出上下文都将被激活，并在 RecognizeText 或 RecognizeUtterance 响应中返回。

在定义输出上下文时，还要定义其生存时间以及该上下文包含在 Amazon Lex 的响应中的时长或回合数。一个回合是指从您的应用程序向 Amazon Lex 发出的一个请求。一旦回合数或时间到期，上下文将不再处于活动状态。

您的应用程序可以根据需要使用输出上下文。例如，您的应用程序可以通过输出上下文来：

- 根据上下文更改应用程序的行为。例如，旅行应用程序对上下文“book_car_fulfilled”的操作可能与对“rental_hotel_fulfilled”的操作不同。

- 将输出上下文作为下一句话的输入上下文返回给 Amazon Lex。如果 Amazon Lex 将话语识别为企图引发意图，则会通过上下文将可以返回的意图限制为具有指定上下文的意图。

输入上下文

您可以设置输入上下文来限制对话中识别意图的点。没有输入上下文的意图始终可以被识别。

您可以通过控制台或 `CreateIntent` 操作设置意图响应的输入上下文。一个意图可以包含多个输入上下文。

对于具有多个输入上下文的意图，所有上下文都必须处于活动状态才能触发该意图。当您调用 RecognizeText、RecognizeUtterance 或 PutSession 操作时，可以设置输入上下文。

您可以将意图中的槽配置为采用当前活动上下文中的默认值。当 Amazon Lex 识别出新的意图但未收到槽值时，使用默认值。定义槽时，可以在表单 `#context-name.parameter-name` 中指定上下文名称和槽名称。有关更多信息，请参阅使用默认槽值。

使用默认槽值

使用默认值时，您可以为新意图指定一个源，以填充用户输入中未提供的槽值。此来源可以是之前的对话、请求或会话属性，也可以是您在构建时设置的固定值。

您可以将以下内容作为默认值来源。

- 之前的对话（上下文）：#context-name.parameter-name
- 会话属性：[attribute-name]
- 请求属性：<attribute-name>
- 固定值：任何与先前值不匹配的值

通过 CreateIntent 操作向意图添加槽时，可以添加默认值列表。将按这些值列出的顺序对其进行排列。例如，假设您有一个带槽的意图，其槽定义如下：

```
"slots": [
    {
        "botId": "string",
        "defaultValueSpec": {
            "defaultValueList": [
                {
                    "defaultValue": "#book-car-fulfilled.startDate"
                },
```

```
                {
                    "defaultValue": "[reservationStartDate]"
                }
            ]
        },
        Other slot configuration settings
    }
]
```

识别出意图后，名为“reservation-start-date”的槽可以设置为如下值：

1. 如果“book-car-fulfilled""上下文处于活动状态，则将“startDate”参数的值用作默认值。
2. 如果“book-car-fulfilled”上下文未处于活动状态，或者未设置“startDate”参数，则将“reservationStartDate”会话属性的值用作默认值。
3. 如果前两个默认值均未使用，则该槽没有默认值，Amazon Lex 将照常得出一个值。

如果使用槽的默认值，则即使需要该槽，也不会引发该槽。

设置会话属性

会话属性包含特定于应用程序的信息，于会话期间在机器人与客户端应用程序之间传递。Amazon Lex 向为机器人配置的所有 Lambda 函数传递会话属性。如果 Lambda 函数添加或更新会话属性，Amazon Lex 会将新信息返回给客户端应用程序。

在 Lambda 函数中使用会话属性来初始化机器人并自定义提示和响应卡。例如：

- 初始化：在订购披萨机器人中，客户端应用程序在第一次调用 RecognizeText 或 RecognizeUtterance 操作时传递用户位置作为会话属性，例如，`"Location": "111 Maple Street"`。Lambda 函数根据该信息查找最近的披萨店下订单。
- 个性化提示：配置提示和响应卡，以引用会话属性。例如，“[FirstName] 您好，您想要什么配料？”如果您传递该用户的名字作为会话属性 (`{"FirstName": "Vivian"}`)，Amazon Lex 将用该名字替换占位符。然后，它会向该用户发送个性化提示，“Vivian 您好，您想要什么配料？”

会话属性在会话存续期内一直存在。Amazon Lex 将其存储在加密数据存储中，直到会话结束。客户端可以通过调用 RecognizeText 或 RecognizeUtterance 操作同时设置 `sessionAttributes` 字段来创建会话属性。Lambda 函数可以在响应中创建会话属性。在客户端或 Lambda 函数创建会话属性后，每当客户端应用程序在发给 Amazon Lex 的请求中不包括 `sessionAttribute` 字段时，都会使用存储的属性值。

例如，假设您有两个会话属性 {"x": "1", "y": "2"}。如果客户端调用 RecognizeText 或 RecognizeUtterance 操作而未指定 sessionAttributes 字段，Amazon Lex 通过存储的会话属性 ({"x": 1, "y": 2}) 调用 Lambda 函数。如果 Lambda 函数未返回会话属性，Amazon Lex 会将存储的会话属性返回给客户端应用程序。

如果客户端应用程序或 Lambda 函数传递会话属性，Amazon Lex 将更新存储的会话属性。传递现有值 (例如 {"x": 2}) 将更新存储的值。如果您传递一组新的会话属性（如 {"z": 3}），将删除现有值而只保留新值。当传递空映射 {} 时，将擦除存储的值。

要向 Amazon Lex 发送会话属性，您要创建属性的字符串到字符串映射。下面显示了如何映射会话属性：

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

对于 RecognizeText 操作，您可以使用 sessionState 结构的 sessionAttributes 字段将映射插入请求正文中，如下所示：

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

对于 RecognizeUtterance 操作，您对映射进行 base64 编码，然后将其作为 x-amz-lex-session-state 标头的一部分进行发送。

如果要在会话属性中发送二进制或结构化数据，则必须先将该数据转换为简单字符串。有关更多信息，请参阅设置复杂属性。

设置请求属性

请求属性包含请求特定的信息，并仅应用于当前请求。客户端应用程序会将此信息发送给 Amazon Lex。可以使用请求属性传递不需要在整个会话中保留的信息。您可以创建自己的请求属性，也可以使用预定义属性。要发送请求属性，请使用 RecognizeUtterance 的 x-amz-lex-request-attributes 标头或 RecognizeText 请求中的 requestAttributes 字段。由于请求属性不像会话属性那样在不同请求间保留，因此不会在 RecognizeUtterance 或 RecognizeText 响应中返回。

> **Note**
>
> 要发送在请求间保留的信息，请使用会话属性。

设置用户定义的请求属性

用户定义的请求属性是您在每个请求中发送给自动程序的数据。可在 RecognizeUtterance 请求的 amz-lex-request-attributes 标头或 RecognizeText 请求的 requestAttributes 字段中发送信息。

要向 Amazon Lex 发送请求属性，请创建属性的字符串到字符串映射。下面显示了如何映射请求属性：

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

对于 PostText 操作，您可以使用 requestAttributes 字段将映射插入请求正文中，如下所示：

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

对于 PostContent 操作，您对映射进行 base64 编码，然后将其作为 x-amz-lex-request-attributes 标头发送。

如果您要在请求属性中发送二进制或结构化数据，必须先将该数据转换为简单字符串。有关更多信息，请参阅设置复杂属性。

设置超时会话

Amazon Lex 保留上下文信息（槽数据和会话属性），直到对话会话结束。要控制会话在自动程序中的持续时间，请设置会话超时。默认情况下，会话持续时间为 5 分钟，但您可以指定介于 0 到 1440 分钟 (24 小时) 之间的任何持续时间。

例如，假设您创建一个支持 OrderShoes 和 GetOrderStatus 等意图的 ShoeOrdering 机器人。当 Amazon Lex 检测到用户的意图是订购鞋子时，它会要求提供槽数据。例如，要求提供鞋子尺码、

颜色、品牌等。如果用户提供了一些槽数据，但未完成鞋子的购买，Amazon Lex 将在整个会话中记住所有槽数据和会话属性。如果用户在会话到期之前返回到会话，该用户可以提供其余槽数据并完成购买。

在 Amazon Lex 控制台中，您应在创建机器人时设置会话超时。使用 AWS 命令行界面 (AWS CLI) 或 API 时，您应在使用 CreateBot 操作创建机器人时通过设置 idleSessionTTLInSeconds 字段来设置超时。

在意图之间共享信息

Amazon Lex 支持在意图之间共享信息。要在意图之间共享信息，请使用输出上下文或会话属性。

要使用输出上下文，请在创建或更新意图时定义输出上下文。履行意图时，来自 Amazon Lex V2 的响应将意图中的上下文和槽值作为上下文参数。您可以将这些参数用作后续意图、应用程序代码或 Lambda 函数的默认值。

要使用会话属性，您需要在 Lambda 或应用程序代码中设置这些属性。例如，ShoeOrdering 自动程序的用户从订购鞋子开始。该自动程序将与用户进行对话，收集槽数据，如鞋子尺寸、颜色和品牌。当用户下单时，履行订单的 Lambda 函数将设置 orderNumber 会话属性，其中包含订单号。要获取订单状态，用户可使用 GetOrderStatus 目的。自动程序可向用户询问槽数据，如订单号和订购日期。自动程序在获得所需的信息以后将返回订单状态。

如果您认为您的用户可能会在同一会话期间改换目的，则可将自动程序设计为返回最新订单的状态。您不必再向用户询问订单信息，而是使用 orderNumber 会话属性在不同目的间共享信息并实现 GetOrderStatus 目的。自动程序通过返回用户最后所下订单的状态完成此操作。

设置复杂属性

会话和请求属性是属性和值的字符串到字符串映射。在许多情况下，您可以使用字符串映射在客户端应用程序与自动程序之间传输属性值。但在某些情况下，您可能需要传输无法轻易转换为字符串映射的二进制数据或复杂结构。例如，以下 JSON 对象表示由美国人口最多的三个城市组成的数组：

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

该数据数组无法正常转换为字符串到字符串映射。在这种情况下，您可以将对象转换为一个简单字符串，以便通过 RecognizeText 和 RecognizeUtterance 操作将其发送给机器人。

例如，如果您使用的是 JavaScript，则可以使用 `JSON.stringify` 操作将对象转换为 JSON，使用 `JSON.parse` 操作将 JSON 文本转换为 JavaScript 对象：

```
// 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);
```

要使用 `RecognizeUtterance` 操作发送属性，您必须先对属性进行 base64 编码，然后再将其添加到请求标头，如以下 JavaScript 代码所示：

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

您可以通过先将二进制数据转换为 base64 编码字符串、然后将该字符串作为会话属性中的值发送，来向 `RecognizeText` 和 `RecognizeUtterance` 操作发送二进制数据：

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

```
}
```

使用 Amazon Lex V2 API 管理会话

用户启动与您的机器人的对话时，Amazon Lex V2 会创建一个会话。您的应用程序与 Amazon Lex V2 之间交换的信息组成了对话的会话状态。当您发出请求时，会话会通过您指定的标识符进行标识。有关会话标识符的更多信息，请参阅 RecognizeText 或 RecognizeUtterance 操作中的 sessionId 字段。

您可以修改在应用程序和自动程序之间发送的会话状态。例如，您可以创建和修改其中包含会话自定义信息的会话属性，也可以通过设置对话上下文来更改对话流，以解释下一个表达。

您可以通过三种方式来更新会话状态。

- 将会话信息作为调用 RecognizeText 或 RecognizeUtterance 操作的一部分，以内联方式传递。
- 使用 Lambda 函数和在每轮对话之后调用的 RecognizeText 或 RecognizeUtterance 操作。有关更多信息，请参阅使用 AWS Lambda 函数启用自定义逻辑。另一种方式是在应用程序中使用 Amazon Lex V2 运行时 API 来更改会话状态。
- 通过操作可以帮助您管理与机器人的对话中的会话信息。这些操作包括 PutSession、GetSession 和 DeleteSession。使用这些操作，您可以获取自动程序与用户会话的状态信息，还可以对状态执行更精细的控制。

如果您希望获取会话的当前状态，请使用 GetSession 操作。此操作会返回会话的当前状态，包括与用户进行的对话的状态、已经设置的任意会话属性以及当前意图和 Amazon Lex V2 识别为与用户言语匹配的任何其他意图的插槽值。

PutSession 操作可让您直接操作当前会话状态。您可以设置会话，包括机器人接下来要执行的对话操作类型以及 Amazon Lex V2 向用户发送的消息。这让您可以控制与自动程序的对话流。将对话操作 type 字段设置为 Delegate，可以让 Amazon Lex V2 确定机器人的下一个操作。

您可以使用 PutSession 操作创建与自动程序的新会话，并设置自动程序在开始时应使用的目的。您还可以使用 PutSession 操作将一个目的更改为另一个。创建会话或更改目的时，您还可以设置会话状态，例如槽值和会话属性。新目的完成后，您可以选择重启之前的目的。

来自 PutSession 操作的响应包含与 RecognizeUtterance 操作相同的信息。您可以使用此信息向用户提示信息的下一部分，就像您对待 RecognizeUtterance 操作的响应一样。

使用 DeleteSession 操作以删除现有会话并使用新会话重新开始。例如，在您测试自动程序时，可以使用 DeleteSession 操作从自动程序删除测试会话。

会话操作可用于履行 Lambda 函数。例如，如果您的 Lambda 函数返回 Failed 作为履行状态，您可以通过 PutSession 操作将对话操作类型设置为 close，将 fulfillmentState 设置为 ReadyForFulfillment 以重试履行步骤。

您可以对会话操作采取下列措施：

- 让自动程序启动对话而不是等待用户启动。
- 在对话期间切换目的。
- 返回到以前的目的。
- 在交互过程中启动或重新启动对话。
- 验证槽值并让自动程序重新提示无效的值。

下文分别详细介绍了这些操作。

启动新会话

如果您希望自动程序开始与用户的对话，可以使用 PutSession 操作。

- 创建没有槽的欢迎目的，并创建一条总结性消息向用户提示阐明目的。例如，“您希望订购什么？ 您可以说‘订一杯饮料’或‘订一个披萨’。”
- 调用 PutSession 操作。将目的名称设置为欢迎目的的名称，并将对话操作设置为 Delegate。
- Amazon Lex 将使用您的欢迎意图中的提示做出响应，以启动与用户的对话。

切换意图

您可以使用 PutSession 操作将一个目的切换为另一个。您还可以使用它切换回以前的目的。您可以使用 PutSession 操作为新目的设置会话属性值或槽值。

- 调用 PutSession 操作。将目的名称设置为新目的的名称，并将对话操作设置为 Delegate。您还可以为新目的设置所需的任意槽值和会话属性。
- Amazon Lex 使用新意图启动与用户的对话。

恢复以前的意图

要恢复以前的意图，可以通过 GetSession 操作获取意图状态，执行所需的交互，然后通过 PutSession 操作将意图设置回其之前的对话状态。

- 调用 GetSession 操作。存储意图状态。
- 进行另一次互动，例如履行不同的意图。
- 根据为上一个意图保存的信息调用该 PutSession 操作。这会将用户返回到对话中相同位置的上一个目的。

在某些情况下，可能需要恢复用户与自动程序的对话。例如，假设您创建了一个客户服务自动程序。您的应用程序确定用户需要与客户服务代表交谈。在与用户交谈之后，客户服务代表可以使用所收集的信息将对话引导回自动程序。

要恢复会话，请使用类似于下文的步骤：

- 您的应用程序确定用户需要与客户服务代表交谈。
- 使用 GetSession 操作获取目的的当前对话状态。
- 客户服务代表与用户交谈并解决问题。
- 使用 PutSession 操作设置目的的对话状态。这可能包括设置槽值、设置会话属性或者更改目的。
- 自动程序恢复与用户的对话。

验证插槽值

您可以使用客户端应用程序验证对自动程序的响应。如果响应无效，您可以使用 PutSession 操作获取用户的新响应。例如，假设您的鲜花订购自动程序只能卖郁金香、玫瑰和百合。如果用户订购康乃馨，您的应用程序可以执行以下操作：

- 检查 PostText 或 PostContent 响应返回的槽值。
- 如果槽值无效，则调用 PutSession 操作。您的应用程序应清除槽值，设置 slotToElicit 字段，并将 dialogAction.type 值设置为 elicitSlot。（可选）如果您希望更改 Amazon Lex 用于引发插槽值的消息，可以设置 message 和 messageFormat 字段。

使用 AWS Lambda 函数启用自定义逻辑

借助 AWS Lambda 函数，您可以通过自己定义的自定义函数更好地控制 Amazon Lex V2 机器人的行为。

Amazon Lex V2 为每种语言的每个机器人别名使用一个 Lambda 函数，而不是为每种意图使用一个 Lambda 函数。

要将 Lambda 函数与您的 Amazon Lex V2 机器人集成，请执行以下步骤：

1. 确定您要从输入事件中的哪些字段中提取信息，以便用于 Lambda 函数。
2. 确定您要操作响应中的哪些字段，然后从 Lambda 函数返回这些字段。
3. 通过您选择的编程语言在 AWS Lambda 中创建函数并编写脚本。
4. 确保该函数返回的结构与响应格式相匹配。
5. 部署 Lambda 函数。
6. 通过控制台或 API 操作，将 Lambda 函数与 Amazon Lex V2 机器人别名相关联。
7. 选择要通过控制台或 API 操作调用 Lambda 函数的对话阶段。
8. 构建您的 Amazon Lex V2 机器人并测试 Lambda 函数是否按预期运行。借助 Amazon CloudWatch 调试您的函数。

主题

解释输入事件格式

将 Lambda 函数集成到您的 Amazon Lex V2 机器人的第一步是了解 Amazon Lex V2 事件中的字段，并从这些字段中确定在编写脚本时需要使用的信息。以下 JSON 对象显示了传递给 Lambda 函数的 Amazon Lex V2 事件的一般格式：

> **Note**
>
> 输入格式可以更改，但不必对 `messageVersion` 做出相应更改。您的代码不应在有新字段时引发错误。

```
{
    "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
                },
                ...
            }
        },
        ...
    ]
}
```

输入事件中的每个字段如下所述：

messageVersion

消息的版本，用于标识进入 Lambda 函数的事件数据的格式以及来自 Lambda 函数的响应的预期格式。

> **Note**
>
> 您在定义意图时可以配置此值。在当前实现中，Amazon Lex V2 仅支持消息版本 1.0。因此，控制台假定 1.0 的默认值，并且不显示消息版本。

invocationSource

调用 Lambda 函数的代码挂钩。以下是可能的值：

`DialogCodeHook`：Amazon Lex V2 在用户输入内容后调用了 Lambda 函数。

`FulfillmentCodeHook`：Amazon Lex V2 在填充了所有必需的槽位后调用了 Lambda 函数并且意图已准备好履行。

inputMode

用户言语模式。可能的值如下所示：

`DTMF`：用户通过按键键盘（双音多频）输入言语。

`Speech`：用户说出了言语。

`Text`：用户键入了言语。

responseContentType

机器人对用户的响应模式。`text/plain; charset=utf-8` 表示写入了最后一个言语，而以 `audio` 开头的值表示说出了最后一个言语。

sessionId

用于对话的字母数字会话标识符。

inputTranscript

用户输入的转录。

- 对于文本输入，表示用户键入的文本。对于 DTMF 输入，表示用户输入的密钥。
- 对于语音输入，这是 Amazon Lex V2 将用户言语转换成的文本，以便调用意图或填补槽位。

invocationLabel

该取值表示调用 Lambda 函数的响应。您可以为初始响应、槽位和确认响应设置调用标签。

自动程序

有关处理请求的机器人的信息，包括以下字段：

- id：创建机器人时分配给机器人的标识符。您可以在 Amazon Lex V2 控制台的机器人设置页面上看到该机器人 ID。
- name：创建机器人时赋予机器人的名称。
- localeId：用于机器人的区域设置的标识符。有关区域设置列表，请参阅 Amazon Lex V2 支持的语言和区域设置。
- version：用于处理请求的机器人的版本。
- aliasId：创建机器人别名时，为其分配的标识符。您可以在 Amazon Lex V2 控制台的别名页面上看到机器人别名 ID。如果您在列表中看不到别名 ID，请选择右上角的齿轮图标并打开别名 ID。
- aliasName：为机器人提供的别名。

interpretations

Amazon Lex V2 认为可能与用户言语相匹配的意图相关信息的列表。每一项都是一个结构，提供有关言语与意图的匹配的信息，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
        }
    }
```

```
}
```

结构中的字段如下：

- intent：包含意图相关信息的结构。有关结构的详细信息，请参阅意图。
- nluConfidence：该分数表明 Amazon Lex V2 对意图与用户意图相匹配的信心程度。
- sentimentResponse：对响应情绪的分析，包含以下字段：
  - sentiment：表示言语的情绪，取值为 POSITIVE、NEGATIVE、NEUTRAL 或 MIXED。
  - sentimentScore：将每种情绪映射到一个数字的结构，该数字表明 Amazon Lex V2 对言语传达这种情绪的信心程度。
- interpretationSource – 表示槽位是由 Amazon Lex 还是由 Amazon Bedrock 解析。

proposedNextState

如果 Lambda 函数将 sessionState 的 dialogAction 设置为 Delegate，则会出现此字段，并显示 Amazon Lex V2 对下一步对话的建议。否则，下一个状态取决于您在 Lambda 函数响应中返回的设置。只有满足以下两个语句时，此结构才存在：

1. invocationSource 的值为 DialogCodeHook。
2. dialogAction 的预测 type 是 ElicitSlot。

您可以通过此信息在对话的正确位置添加 runtimeHints。有关更多信息，请参阅 使用运行时提示改善对插槽值的识别。proposedNextState 是包含以下字段的结构：

proposedNextState 的结构如下所示：

```
"proposedNextState": {
    "dialogAction": {
        "slotToElicit": string,
        "type": "Close | ConfirmIntent | Delegate | ElicitIntent | ElicitSlot"
    },
    "intent": {
        // see ## for details about the structure
    },
    "prompt": {
        "attempt": string
    }
}
```

- dialogAction：包含有关 Amazon Lex V2 提出的下一步行动的信息。结构中的字段如下：
  - slotToElicit：Amazon Lex V2 提出的下一个要引发的槽位。仅当 type 的值为 ElicitSlot 时才显示此字段。
  - type：Amazon Lex V2 提出的下一步对话。以下是可能的值：

    Delegate：Amazon Lex V2 决定下一步操作。

    ElicitIntent：下一步操作是引发用户的意图。

    ElicitSlot：下一步操作是从用户引发槽位值。

    Close：结束意图履行流程并表示用户不会做出回应。

    ConfirmIntent：下一步操作是询问用户槽位是否正确以及意图是否已准备履行。
- intent：机器人已确定用户正在尝试履行的意图。有关结构的详细信息，请参阅意图。
- prompt：包含字段 attempt 的结构，该字段映射到一个值，该值指定 Amazon Lex V2 提示用户进入下一槽位的次数。可能的值为第一次尝试的 Initial，以及后续尝试的 Retry1、Retry2、Retry3、Retry4 和 Retry5。

requestAttributes

一种结构，其中包含客户端在请求中发送的请求特定的属性。可以使用请求属性传递不需要在整个会话中保留的信息。如果没有请求属性，该值将为空。有关更多信息，请参阅设置请求属性。

sessionState

用户与您的 Amazon Lex V2 机器人之间对话的当前状态。有关结构的详细信息，请参阅会话状态。

transcriptions

Amazon Lex V2 认为可能与用户言语匹配的转录的列表。有关更多信息，请参阅使用语音转录置信度分数。每一项都是一个对象，具有以下格式，包含有关一个可能转录的信息：

```
{
    "transcription": string,
    "transcriptionConfidence": number,
    "resolvedContext": {
        "intent": string
    },
```

```
    "resolvedSlots": {
        slot name: {
            // see ## for details about the structure
        },
        ...
    }
}
```

这些字段如下所述：

- transcription：Amazon Lex V2 认为可能与用户的音频话言语匹配的转录。
- transcriptionConfidence：该分数表明 Amazon Lex V2 对意图与用户意图相匹配的信心程度。
- resolvedContext：一种包含字段 `intent` 的结构，该字段映射到言语相关意图。
- resolvedSlots：一种结构，其键是通过言语解析的每个槽位的名称。每个槽位名称都映射到一个包含该槽位相关信息的结构。有关结构的详细信息，请参阅槽值。

准备响应格式

将 Lambda 函数集成到您的 Amazon Lex V2 机器人中的第二步是了解 Lambda 函数响应中的字段，并确定您要操作哪些参数。以下 JSON 对象显示了返回到 Amazon Lex V2 的 Lambda 响应的一般格式：

```
{
    "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,
        ...
    }
}
```

响应中的每个字段如下所述：

sessionState

您想要返回的用户与您的 Amazon Lex V2 机器人之间的对话状态。有关结构的详细信息，请参阅会话状态。此字段为必填。

messages

Amazon Lex V2 在下一轮对话中返回给客户的消息的列表。如果您提供的 `contentType` 取值是 `PlainText`、`CustomPayload` 或 `SSML`，请在 `content` 字段中写下您要返回给客户的消息。如果您提供的 `contentType` 取值是 `ImageResponseCard`，请在 `imageResponseCard` 字段中提供卡的详细信息。如果您不提供消息，Amazon Lex V2 将使用创建机器人时定义的相应消息。

如果 `dialogAction.type` 的取值是 `ElicitIntent` 或 `ConfirmIntent`，则 `messages` 字段为必填字段。

列表中的每一项都是以下格式的结构，其中包含要返回给用户的消息的相关信息。示例如下：

```
{
    "contentType": "CustomPayload | ImageResponseCard | PlainText | SSML",
    "content": string,
    "imageResponseCard": {
        "title": string,
        "subtitle": string,
        "imageUrl": string,
        "buttons": [
            {
                "text": string,
                "value": string
            },
```

```
            ...
        ]
    }
}
```

每个字段的描述如下：

- contentType：要使用的消息类型。

  CustomPayload：一个响应字符串，您可以自定义该字符串以包含应用程序的数据或元数据。

  ImageResponseCard：带有按钮的图像，客户可以选择。有关更多信息，请参阅 ImageResponseCard。

  PlainText：纯文本字符串。

  SSML：包含语音合成标记语言的字符串，用于自定义音频响应。
- content：要发送给用户的消息。如果消息类型为 PlainText、CustomPayload 或 SSML，则使用此字段。
- imageResponseCard：包含要向用户显示的响应卡的定义。如果消息类型为 ImageResponseCard，则使用此字段。映射到一个包含以下字段的结构：
  - title：响应卡的标题。
  - subtitle：提示用户选择按钮。
  - imageUrl：指向卡的图像的链接。
  - buttons：包含按钮相关信息的结构的列表。如果客户选择该按钮，每个结构都包含一个包含要显示文本的 text 字段，以及一个包含要发送到 Amazon Lex V2 的值的 value 字段。最多可以包含三个按钮。

requestAttributes

一种结构，其中包含针对客户的响应的请求特定的属性。请参阅设置请求属性了解更多信息。该字段是可选的。

响应中的必填字段

Lambda 响应必须包含至少一个 sessionState 对象。该响应提供一个 dialogAction 对象并指定 type 字段。根据您提供的 dialogAction 的 type，Lambda 响应可能还包含其他必填字段。这些要求如下所述，并附带最少的工作示例：

Delegate

Delegate 让 Amazon Lex V2 决定下一步操作。没有其他必填字段。

```
{
    "sessionState": {
        "dialogAction": {
            "type": "Delegate"
    }
}
```

ElicitIntent

ElicitIntent 提示客户表达意图。messages 字段中必须包含至少一条消息才能提示引发意图。

```
{
    "sessionState": {
        "dialogAction": {
            "type": "ElicitIntent"
    },
    "messages": [
        {
            "contentType": PlainText,
            "content": "How can I help you?"
        }
    ]
}
```

ElicitSlot

ElicitSlot 提示客户提供槽位值。dialogAction 对象的 slotToElicit 字段中必须包含槽位的名称。还必须包含 sessionState 对象中 intent 的 name。

```
{`
    "sessionState": {
        "dialogAction": {
            "slotToElicit": "OriginCity",
            "type": "ElicitSlot"
        },
        "intent": {
            "name": "BookFlight"
        }
```

```
    }
}
```

ConfirmIntent

ConfirmIntent 确认客户的槽位值以及意图是否准备履行。必须包含 sessionState 对象中 intent 的 name，以及待确认的 slots。messages 字段中还必须包含至少一条消息，要求用户确认槽位值。您的消息应提示响应“是”或“否”。如果用户响应“是”，则 Amazon Lex V2 会将意图的 confirmationState 设置为 Confirmed。如果用户响应“否”，则 Amazon Lex V2 会将意图的 confirmationState 设置为 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

Close 结束意图的履行过程，并表示预计用户不会给出进一步的响应。您必须在 sessionState 对象中包含 intent 的 name 和 state。兼容的意图状态包括 Failed、Fulfilled 和 InProgress。

```
"sessionState": {
    "dialogAction": {
        "type": "Close"
    },
    "intent": {
        "name": "BookFlight",
        "state": "Failed | Fulfilled | InProgress"
    }
}
```

Lambda 事件和响应中的常见结构

在 Lambda 响应中，有许多结构会反复出现。本节提供了有关这些常见结构的详细信息。

意图

```
"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
       }
}
```

intent 字段映射到包含以下字段的对象：

confirmationState

表示用户是否已确认意图的槽位，以及该意图是否已准备履行。以下是可能的值：

Confirmed：用户确认槽位值正确。

Denied：用户表示槽位值不正确。

None：用户尚未进入确认阶段。

名称

意图的名称。

slots

有关履行意图所需的槽位的信息。有关结构的详细信息，请参阅槽值。

state

表示意图的履行状态。以下是可能的值：

Failed：机器人未能履行意图。

Fulfilled：机器人已完成意图的履行。

FulfillmentInProgress：机器人正在履行意图。

InProgress：机器人正在引发履行意图所需的槽位值。

ReadyForFulfillment：机器人已引发意图的所有槽位值，并已准备好履行意图。

Waiting：机器人正在等待用户响应（仅限流式对话）。

kendraResponse

包含有关 Kendra 搜索查询结果的信息。仅当意图为 KendraSearchIntent 时才显示此字段。有关更多信息，请参阅 Kendra 的查询 API 调用中的响应语法。

槽值

slots 字段存在于 intent 结构中，并映射到一个结构，该结构的键是用于该意图的槽位的名称。如果该槽位不是多值槽位（有关更多详细信息，请参阅 使用一个插槽中的多个值），则会将其映射到具有以下格式的结构。请注意，shape 取值为 Scalar。

```
{
    slot name: {
        "shape": "Scalar",
        "value": {
            "originalValue": string,
            "interpretedValue": string,
            "resolvedValues": [
                string,
                ...
            ]
        }
    }
}
```

如果该槽位是多值槽位，则其映射到的对象中包含另一个名为 values 的字段，该字段映射到一个结构列表，其中每个结构都包含有关构成多值槽位的槽位信息。列表中每个对象的格式与常规槽位所映射到的对象的格式相匹配。请注意，shape 的取值为 List，但 values 下面组件槽位的 shape 的取值为 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,
                    ...
                ]
            }
        },
        ...
    ]
}
```

槽位对象中的字段如下所述：

shape

槽位的形状。如果槽位中有多个值（有关更多详细信息，请参阅 使用一个插槽中的多个值），则该值为 `List`，否则该值为 `Scalar`。

value

一个对象，其中包含有关用户为槽位提供的值以及 Amazon Lex 的解释信息，格式如下：

```
{
    "originalValue": string,
    "interpretedValue": string,
    "resolvedValues": [
        string,
```

```
        ...
    ]
}
```

这些字段如下所述：

- originalValue：Amazon Lex 确定用户对引发槽位的响应中与槽位值相关的部分。
- interpretedValue：Amazon Lex 根据用户输入为该槽位确定的值。
- resolvedValues：Amazon Lex 确定的值列表，是对用户输入的可能解析。

values

包含有关构成多值槽位的槽位信息的对象的列表。每个对象的格式与普通槽位的格式相匹配，shape 和 value 字段如上所述。仅当槽位由多个值组成时，values 才会出现（有关更多详细信息，请参阅使用一个插槽中的多个值）。以下 JSON 对象显示了两个组件槽位：

```
"values": [
    {
        "shape": "Scalar",
        "value": {
            "originalValue": string,
            "interpretedValue": string,
            "resolvedValues": [
                string,
                ...
            ]
        }
    },
    {
        "shape": "Scalar",
        "value": {
            "originalValue": string,
            "interpretedValue": string,
            "resolvedValues": [
                string,
                ...
            ]
        }
    },
    ...
]
```

会话状态

sessionState 字段映射到一个对象，该对象包含与用户对话的状态相关的信息。该对象中实际显示的字段取决于对话操作的类型。有关 Lambda 响应中的必填字段，请参阅 响应中的必填字段。sessionState 对象的格式如下所示：

```
"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
}
```

这些字段如下所述：

activeContexts

包含用户在会话中使用的上下文相关信息的对象的列表。通过上下文来促进和控制意图识别。有关上下文的更多信息，请参阅 设置意图上下文。每个对象的格式如下：

```
{
    "name": string,
    "contextAttributes": {
        string: string
    },
    "timeToLive": {
        "timeToLiveInSeconds": number,
        "turnsToLive": number
    }
}
```

这些字段如下所述：

- name：上下文的名称。
- contextAttributes：一个对象，其中包含上下文属性名称及其映射到的值。
- timeToLive：一个对象，用于指定上下文保持活动状态的时长。此对象可以包含以下一个或两个字段：
  - timeToLiveInSeconds：上下文保持活动状态的秒数。
  - turnsToLive：上下文保持活动状态的回合数。

sessionAttributes

表示会话特定上下文信息的键值对的映射。有关更多信息，请参阅设置会话属性。对象的格式如下：

```
{
```

```
    string: string,
    ...
}
```

runtimeHints

为客户可能在槽位中使用的短语提供提示，以提高音频识别能力。与发音相似的单词相比，您在提示中提供的值可以增强对这些值的音频识别。runtimeHints 对象的格式如下所示：

```
{
    "slotHints": {
        intent name: {
            slot name: {
                "runtimeHintValues": [
                    {
                        "phrase": string
                    },
                    ...
                ]
            },
            ...
        },
        ...
    }
}
```

slotHints 字段映射到一个对象，该对象中的字段是机器人中意图的名称。每个意图名称都映射到一个对象，该对象的字段是对应意图的槽位名称。每个槽位名称都映射到一个带有单个字段 runtimeHintValues 的结构，即对象列表。每个对象都包含一个映射到提示的 phrase 字段。

dialogAction

确定 Amazon Lex V2 要采取的下一步操作。对象的格式如下所示：

```
{
    "slotElicitationStyle": "Default | SpellByLetter | SpellByWord",
    "slotToElicit": string,
    "type": "Close | ConfirmIntent | Delegate | ElicitIntent | ElicitSlot"
}
```

这些字段如下所述：

- slotElicitationStyle：当 `dialogAction` 的 `type` 的取值为 `ElicitSlot` 时，确定 Amazon Lex V2 如何解释来自用户的音频输入。有关更多信息，请参阅使用拼写样式捕获槽位值。以下是可能的值：

  `Default`：Amazon Lex V2 以默认方式解释音频输入以履行槽位。

  `SpellByLetter`：Amazon Lex V2 监听用户对槽位值的拼写。

  `SpellByWord`：Amazon Lex V2 根据与每个字母相关的单词（例如，“a as in apple”）来监听用户对槽位值的拼写。
- slotToElicit：定义 `dialogAction` 的 `type` 的取值为 `ElicitSlot` 时要从用户那里引发的槽位。
- type：定义机器人应执行的操作。以下是可能的值：

  `Delegate`：让 Amazon Lex V2 决定下一步操作。

  `ElicitIntent`：提示客户表达意图。

  `ConfirmIntent`：确认客户的槽位值以及意图是否准备履行。

  `ElicitSlot`：提示客户为意图提供槽位值。

  `Close`：结束意图履行流程。

意图

有关 `intent` 字段的结构，请参阅 意图。

originatingRequestId

请求的唯一标识符。该字段对于 Lambda 响应是可选字段。

创建 Lambda 函数并将其附加到机器人别名

创建 Lambda 函数

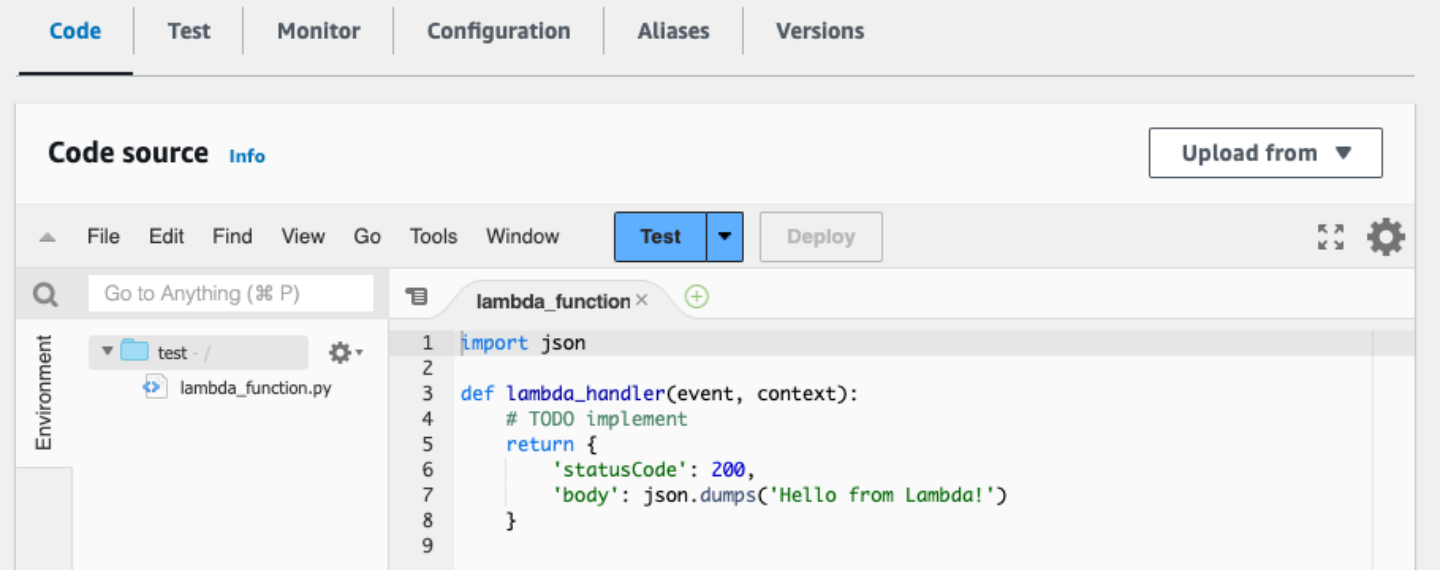
要为您的 Amazon Lex V2 机器人创建 Lambda 函数，请从您的 AWS Management Console 访问 AWS Lambda 并创建一个新函数。了解有关 AWS Lambda 的更多详细信息，请参阅 AWS Lambda 开发者指南。

1. 访问 https://console.aws.amazon.com/lambda/，登录 AWS Management Console 并打开 AWS Lambda 控制台。

2. 在左侧边栏中选择函数。
3. 选择创建函数。
4. 您可以选择从头开始创作以最少的代码开始，选择使用蓝图以从列表中选择常见用例的示例代码，或者选择容器镜像为您的函数选择要部署的容器镜像。如果您选择从头开始创作，请继续执行以下步骤：

   a. 给您的函数起一个有意义的函数名称来描述它的作用。
   b. 从运行时下拉菜单中选择一种语言来编写函数。
   c. 为您的函数选择一个指令集架构。
   d. 默认情况下，Lambda 创建具有基本权限的新角色。要使用现有角色或通过 AWS 策略模板创建角色，请展开更改默认执行角色菜单并选择一个选项。
   e. 展开高级设置菜单以配置更多选项。
5. 选择创建函数。

从头开始创建新函数时所看到的内容如下图所示。



Lambda 处理程序函数因您使用的语言而异。它至少需要一个 `event` JSON 对象作为参数。您可以进入 解释输入事件格式 查看 Amazon Lex V2 提供的 `event` 中包含的字段。修改处理程序函数，最终返回与 准备响应格式 中所述格式相匹配的 `response` JSON 对象。

编写完函数后，选择部署以允许使用该函数。

请记住，您最多可以将每个机器人别名与一个 Lambda 函数相关联。但是，您可以在 Lambda 代码中为机器人定义任意数量的函数，然后在 Lambda 处理程序函数中调用这些函数。例如，虽然同一个机

器人别名中的所有意图都必须调用相同的 Lambda 函数，但您可以创建一个路由器函数来为每个意图激活单独的函数。可以为应用程序使用或修改的路由器函数示例如下所示：

```
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
```

添加和调用 Lambda 函数

要在您的 Amazon Lex V2 机器人中调用 Lambda 函数，必须先将该函数附加到机器人别名，然后在对话中设置机器人调用该函数的时间。您可以通过控制台或 API 操作执行这些步骤。

在与用户的对话中，您可以在以下时间使用 Lambda 函数：

- 在识别意图后的初始响应中，例如，在用户说要订购披萨之后。
- 从用户引发槽位值后，例如，在用户告诉机器人他们要订购的披萨大小之后。
- 在每次重试之间引发一个槽位时，例如，客户没有使用公认的披萨尺寸时。
- 确认意图时，例如，在确认披萨订单时。
- 要履行意图时，例如，要订购披萨时。
- 在履行意图之后以及机器人结束对话之前。例如，切换到点饮料的意图时。

主题

- 使用 控制台
- 使用 API 操作

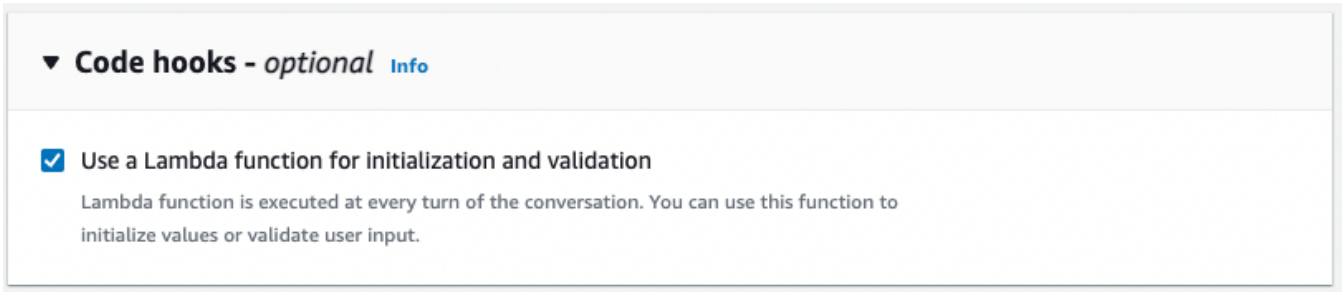
使用 控制台

将 Lambda 函数附加到机器人别名

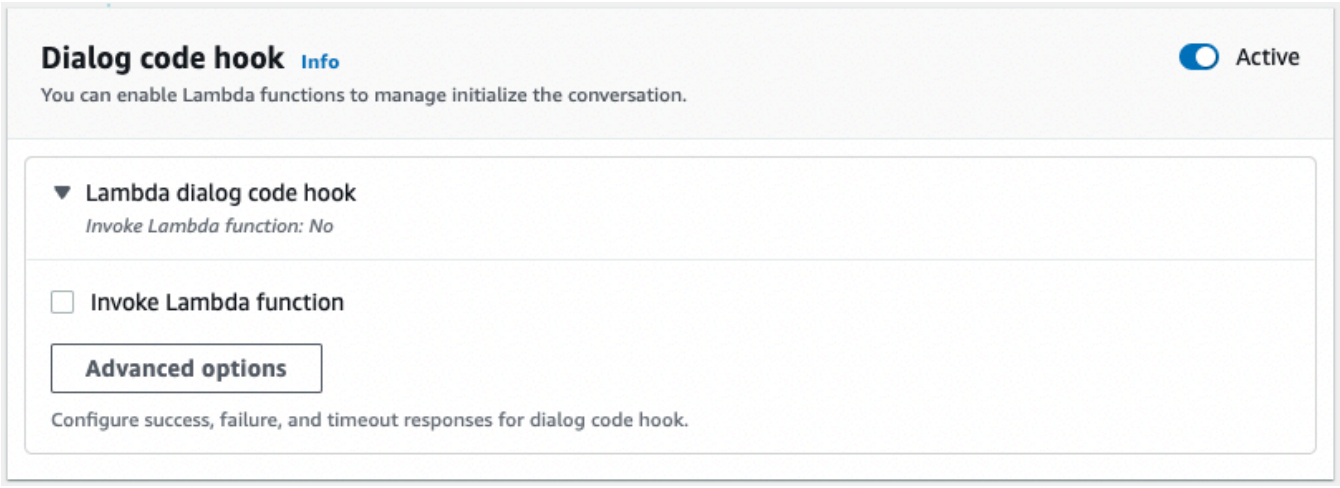
1. 登录 AWS Management Console，并通过以下网址打开 Amazon Lex 控制台：https://console.aws.amazon.com/lex/。
2. 从左侧面板中选择机器人，然后从机器人列表中选择要附加 Lambda 函数的机器人的名称。
3. 在左侧面板中，从部署菜单下选择别名。
4. 从别名列表中，选择要向其附加 Lambda 函数的别名。
5. 在语言面板中，选择您希望 Lambda 函数使用的语言。如果面板中不存在某种语言，请选择管理别名中的语言以添加该语言。
6. 在源下拉菜单中，选择要附加的 Lambda 函数的名称。
7. 在 Lambda 函数版本或别名下拉菜单中，选择要使用的 Lambda 函数的版本或别名。然后选择保存。在机器人支持的语言中，所有意图都使用相同的 Lambda 函数。

设置调用 Lambda 函数的意图

1. 选择机器人后，在要调用 Lambda 函数的机器人的语言下方的左侧菜单中选择意图。
2. 选择要在其中调用 Lambda 函数的意图以打开意图编辑器。
3. 设置 Lambda 代码挂钩有两个选项：
    1. 要在对话的每一步之后调用 Lambda 函数，请滚动到意图编辑器底部的代码挂钩部分，然后选中使用 Lambda 函数进行初始化和验证复选框，如下图所示：



2. 或者，在对话阶段使用对话框代码挂钩部分来调用 Lambda 函数。对话框代码挂钩部分如下所示：



可通过两种方法控制 Amazon Lex V2 如何调用代码挂钩以获取响应：

- 切换活动按钮以将其标记为活动或非活动状态。当代码挂钩处于活动状态时，Amazon Lex V2 将调用该代码挂钩。当代码挂钩处于非活动状态时，Amazon Lex V2 不会运行代码挂钩。
- 展开 Lambda 对话框代码挂钩部分，然后选中调用 Lambda 函数复选框将其标记为已启用或已禁用。只有在代码挂钩标记为活动状态时，才能启用或禁用该挂钩。当标记为已启用时，代码挂钩正常运行。当标记为已禁用时，不会调用代码挂钩，Amazon Lex V2 则表现为代码挂钩已成功返回。要配置对话框代码挂钩成功、失败或超时后的响应，请选择高级选项。

可以在以下对话阶段调用 Lambda 代码挂钩：

- 要调用该函数作为初始响应，请滚动到初始响应部分，点击响应以确认用户的请求旁边的箭头展开，然后选择高级选项。在弹出的菜单底部找到对话框代码挂钩。
- 要在槽位引发后调用该函数，请滚动到槽位部分，单击相关槽位提示旁边的箭头展开，然后选择高级选项。在弹出的菜单底部附近找到对话框代码挂钩，位于默认值正上方。

  您也可以在每次引发后调用该函数。为此，请展开槽位提示部分的机器人引发信息，选择更多提示选项，然后选择每次引发后调用 Lambda 代码挂钩对应复选框。
- 要调用意图确认函数，请滚动至确认部分，点击确认意图提示旁边的箭头展开，然后选择高级选项。在弹出的菜单底部找到对话框代码挂钩。

- 要调用意图履行函数，请滚动至履行部分。切换活动按钮以将代码挂钩设置为活动状态。点击成功履行时旁边的箭头展开，然后选择高级选项。选中履行 Lambda 代码挂钩部分的使用 Lambda 函数进行履行旁边的复选框，将代码挂钩设置为已启用。

4. 设置调用 Lambda 函数的对话阶段后，请再次构建机器人以测试该函数。

使用 API 操作

将 Lambda 函数附加到机器人别名

如果您要创建新的机器人别名，请执行 CreateBotAlias 操作来附加 Lambda 函数。要将 Lambda 函数附加到现有机器人别名，请执行 UpdateBotAlias 操作。修改 botAliasLocaleSettings 字段以包含正确的设置：

```
{
    "botAliasLocaleSettings" : {
        locale: {
            "codeHookSpecification": {
                "lambdaCodeHook": {
                    "codeHookInterfaceVersion": "1.0",
                    "lambdaARN": "arn:aws:lambda:region:account-id:function:function-
name"
                }
            },
            "enabled": true
        },
        ...
    }
}
```

1. botAliasLocaleSettings 字段映射到一个对象，该对象的键是要在其中附加 Lambda 函数的区域设置。有关支持的区域设置和有效密钥代码的列表，请参阅 支持的语言和区域设置。

2. 要查找 Lambda 函数的 lambdaARN，请访问 https://console.aws.amazon.com/lambda/home，打开 AWS Lambda 控制台，在左侧栏中选择函数，然后选择要与机器人别名关联的函数。在函数概述的右侧，找到函数 ARN 下的 lambdaARN。其中应包含区域、账户 ID 和函数名称。

3. 要允许 Amazon Lex V2 为别名调用 Lambda 函数，请将 enabled 字段设置为 true。

设置调用 Lambda 函数的意图

要在意图期间设置 Lambda 函数调用，请在创建新意图时执行 CreateIntent 操作，或者在现有意图中调用该函数时执行 UpdateIntent 操作。在意图操作中控制 Lambda 函数调用的字段是 dialogCodeHook、initialResponseSetting、intentConfirmationSetting 和 fulfillmentCodeHook。

如果您在引发槽位期间调用该函数，则在创建新槽位时执行 CreateSlot 操作，或者执行 UpdateSlot 操作在现有槽位中调用该函数。在槽位操作中控制 Lambda 函数调用的字段是 valueElicitationSetting 对象的 slotCaptureSetting 字段。

1. 要将 Lambda 对话框代码挂钩设置为在对话的每回合之后运行，请将 dialogCodeHook 字段中 DialogCodeHookSettings 对象的 enabled 字段设置为 true。

```
"dialogCodeHook": {
    "enabled": boolean
}
```

2. 另外，您可以通过修改与您要调用函数的对话阶段对应的结构中的 codeHook 和/或 elicitationCodeHook 字段，将 Lambda 对话框代码挂钩设置为仅在对话的特定点运行。要使用 Lambda 对话框代码挂钩履行意图，请使用 CreateIntent 或 UpdateIntent 操作中的 fulfillmentCodeHook 字段。这三种类型的代码挂钩的结构和用途如下：

codeHook

codeHook 字段定义了代码挂钩在对话的给定阶段运行的设置。它是一个 DialogCodeHookInvocationSetting 对象，结构如下：

```
"codeHook": {
    "active": boolean,
    "enableCodeHookInvocation": boolean,
    "invocationLabel": string,
    "postCodeHookSpecification": PostDialogCodeHookInvocationSpecification object,
}
```

- 将 Amazon Lex V2 的 active 字段更改为 true，从而在对话中该取值对应的时刻调用代码挂钩。
- 将 Amazon Lex V2 的 enableCodeHookInvocation 字段更改为 true，从而允许代码挂钩正常运行。如果将其标记为 false，Amazon Lex V2 则表现为代码挂钩已成功返回。
- invocationLabel 表示调用代码挂钩的对话步骤。

- 通过 postCodeHookSpecification 字段指定代码挂钩成功、失败或超时后的操作和消息。

elicitationCodeHook

elicitationCodeHook 字段定义了在需要重新引发一个或多个槽位时运行的代码挂钩的设置。如果槽位引发失败或意图确认被拒绝，则可能会出现这种情况。elicitationCodeHook 字段是具有以下结构的 ElicitationCodeHookInvocationSetting 对象：

```
"elicitationCodeHook": {
    "enableCodeHookInvocation": boolean,
    "invocationLabel": string
}
```

- 将 Amazon Lex V2 的 enableCodeHookInvocation 字段更改为 true，从而允许代码挂钩正常运行。如果将其标记为 false，Amazon Lex V2 则表现为代码挂钩已成功返回。
- invocationLabel 表示调用代码挂钩的对话步骤。

fulfillmentCodeHook

fulfillmentCodeHook 字段定义了为履行意图而运行的代码挂钩的设置。该字段映射到以下 FulfillmentCodeHookSettings 对象：

```
"fulfillmentCodeHook": {
    "active": boolean,
    "enabled": boolean,
    "fulfillmentUpdatesSpecification": FulfillmentUpdatesSpecification object,
    "postFulfillmentStatusSpecification": PostFulfillmentStatusSpecification object
}
```

- 将 Amazon Lex V2 的 active 字段更改为 true，从而在对话中该取值对应的时刻调用代码挂钩。
- 将 Amazon Lex V2 的 enabled 字段更改为 true，从而允许代码挂钩正常运行。如果将其标记为 false，Amazon Lex V2 则表现为代码挂钩已成功返回。
- 通过 fulfillmentUpdatesSpecification 字段指定在履行意图期间出现用户更新的消息以及与之相关的时间。
- 通过 postFulfillmentStatusSpecification 字段指定代码挂钩成功、失败或超时后的消息和操作。

您可以通过将 active 和 enableCodeHookInvocation/enabled 字段设置为 true，在对话的以下时间点调用 Lambda 代码挂钩：

初始响应中

要在识别意图后在初始响应中调用 Lambda 函数，请使用 CreateIntent 或 UpdateIntent 操作的 initialResponse 字段中的 codeHook 结构。initialResponse 字段映射到以下 InitialResponseSetting 对象：

```
"initialResponse": {
    "codeHook": {
        "active": boolean,
        "enableCodeHookInvocation": boolean,
        "invocationLabel": string,
        "postCodeHookSpecification": PostDialogCodeHookInvocationSpecification object,
    },
    "initialResponse": FulfillmentUpdatesSpecification object,
    "nextStep": PostFulfillmentStatusSpecification object,
    "conditional": ConditionalSpecification object
}
```

引发槽位之后或重新引发槽位期间

要在引发槽值后调用 Lambda 函数，请使用 CreateSlot 或 UpdateSlot 操作的 valueElicitation 字段中的 slotCaptureSetting。slotCaptureSetting 字段映射到以下 SlotCaptureSetting 对象：

```
"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
}
```

- 要在成功引发槽位后调用 Lambda 函数，请使用 codeHook 字段。
- 要在槽位引发失败且 Amazon Lex V2 尝试重试槽位引发后调用 Lambda 函数，请使用 elicitationCodeHook 字段。

确认或拒绝意图后

要在确认意图时调用 Lambda 函数，请使用 CreateIntent 或 UpdateIntent 操作的 intentConfirmationSetting 字段。intentConfirmation 字段映射到以下 IntentConfirmationSetting 对象：

```
"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
}
```

- 要在用户确认意图及其槽位后调用 Lambda 函数，请使用 codeHook 字段。
- 要在用户拒绝意图确认且 Amazon Lex V2 尝试重试槽位引发后调用 Lambda 函数，请使用 elicitationCodeHook 字段。

履行意图期间

要调用 Lambda 函数来履行意图，请使用 CreateIntent 或 UpdateIntent 操作的 `fulfillmentCodeHook` 字段。`fulfillmentCodeHook` 字段映射到以下 FulfillmentCodeHookSettings 对象：

```
{
    "active": boolean,
    "enabled": boolean,
    "fulfillmentUpdatesSpecification": FulfillmentUpdatesSpecification object,
    "postFulfillmentStatusSpecification": PostFulfillmentStatusSpecification object
}
```

3. 设置调用 Lambda 函数的对话阶段后，请执行 `BuildBotLocale` 操作重建机器人以测试该函数。

调试 Lambda 函数

Amazon CloudWatch Logs 是一款用于跟踪 API 调用和指标的工具，可帮助您调试 Lambda 函数。当您通过控制台或 API 调用测试机器人时，CloudWatch 会记录对话的每个步骤。如果您在 Lambda 代码中使用打印函数，CloudWatch 也会显示该函数。

查看 Lambda 函数的 CloudWatch 日志

1. 访问 https://console.aws.amazon.com/cloudwatch/，登录 AWS Management Console 并进入 CloudWatch 控制台。
2. 在左侧栏的日志菜单下，选择日志组。
3. 选择您的 Lambda 函数日志组，其格式应为 `/aws/lambda/`*`function-name`*。
4. 日志流列表中包含与机器人进行的每个会话的日志。选择要查看的日志流。
5. 在日志事件列表中，点击时间戳旁边的右箭头以展开该事件的详细信息。您从 Lambda 代码中打印的任何内容都将显示为日志事件。根据该信息来调试您的代码。
6. 调试代码后，请记得部署 Lambda 函数。如果您选择通过控制台部署，请在重新测试机器人行为之前重新加载测试窗口。

自定义机器人交互

了解以下功能，这些功能可用于通过扩展和调整用户的默认行为来自定义机器人与用户的交互：

主题

分析用户言语的情绪

您可以使用情绪分析来确定用户语句中表达的情绪。通过情绪信息，您可以管理对话流或执行呼叫后分析。例如，如果用户情绪是消极的，您可以创建一个流，将对话交给人工代理。

Amazon Lex 与 Amazon Comprehend 集成以检测用户情绪。来自 Amazon Comprehend 的响应可表示文本的整体情绪是积极、中性、消极还是混杂。响应包含用户语句最可能传达的情绪以及每个情绪类别的分数。分数表示正确检测到情绪的可能性。

您可以使用控制台或使用 Amazon Lex API 为机器人启用情绪分析。您可以对机器人的别名启用情绪分析。在 Amazon Lex 控制台上：

1. 选择别名。
2. 在详细信息部分中，选择编辑。
3. 选择启用情绪分析以开启或关闭情绪分析。
4. 选择 Confirm (确认) 以保存所做的更改。

如果您使用的是 API，请将 `detectSentiment` 字段设置为 `true`，然后调用 CreateBotAlias 操作。

启用情绪分析后，来自 RecognizeText 和 RecognizeUtterance 操作的响应将在 `interpretations` 结构中返回 `sentimentResponse` 字段和其他元数据。`sentimentResponse` 字段具有 `sentiment` 和 `sentimentScore` 两个字段，包含情绪分析的结果。如果您使用的是 Lambda 函数，则 `sentimentResponse` 字段将包含在发送到函数的事件数据中。

以下是 `sentimentResponse` 字段作为 `RecognizeText` 或 `RecognizeUtterance` 响应的一部分返回的示例。

```
sentimentResponse {
    "sentimentScore": {
        "mixed": 0.030585512690246105,
        "positive": 0.94992071056365967,
        "neutral": 0.0141543131828308,
        "negative": 0.00893945890665054
    },
    "sentiment": "POSITIVE"
}
```

Amazon Lex 代表您调用 Amazon Comprehend，以确定机器人处理的每个言语中的情绪。启用情绪分析即表示您同意 Amazon Comprehend 的服务条款和协议。有关 Amazon Comprehend 定价的更多信息，请参阅 Amazon Comprehend 定价。

有关 Amazon Comprehend 情绪分析工作原理的更多信息，请参阅《Amazon Comprehend 开发人员指南》中的确定情绪。

使用置信度分数

Amazon Lex V2 使用两个步骤来确定用户所说的内容。第一步是利用自动语音识别 (ASR) 来创建用户音频言语的转录。第二步是利用自然语言理解 (NLU) 来确定用户言语的含义，以识别用户的意图或插槽的值。

默认情况下，Amazon Lex V2 会返回 ASR 和 NLU 中最有可能的结果。有些情况下，Amazon Lex V2 可能难以确定最有可能的结果。在这种情况下，会返回几个可能的结果以及表明相应结果的正确可能性的置信度分数。置信度分数是 Amazon Lex V2 提供的评分，显示了其对结果的相对置信度。置信度分数范围是 0.0 到 1.0。

您可以将自己的专有领域知识与置信度分数相结合，以帮助确定对 ASR 或 NLU 结果的正确解释。

ASR（或转录）置信度分数是 Amazon Lex V2 对特定转录正确性的置信度的评分。NLU（或意图）置信度分数是 Amazon Lex V2 对最优转录中指定的意图正确性的置信度的评分。使用最适合您的应用程序的置信度分数。

主题

使用意图置信度分数

当用户说话时，Amazon Lex V2 会使用自然语言理解 (NLU) 来理解用户的请求并返回正确的意图。默认情况下，Amazon Lex V2 会返回您的机器人定义的最可能的意图。

在某些情况下，Amazon Lex V2 可能难以确定最可能的意图。例如，用户可能会说出模棱两可的言语，或者可能有两个相似的意图。为了帮助确定正确的意图，您可以将自己的专有领域知识与解释列表中的 NLU 置信度分数相结合。置信度分数是 Amazon Lex V2 提供的评级，表明其对意图是正确意图的置信度。

要确定某个解释内两个意图之间的差异，您可以比较其置信度分数。例如，如果一个意图的置信度分数为 0.95，而另一个意图的置信度分数为 0.65，则第一个意图可能是正确的。但是，如果一个意图的分数为 0.75，而另一个意图的分数为 0.72，则这两个意图之间存在歧义，您可以在应用程序中使用领域知识来区分。

您还可以使用置信度分数来创建测试应用程序，以确定对意图言语的更改是否会影响机器人的行为。例如，您可以使用一组言语来获取机器人意图的置信度分数，然后用新的言语更新意图。然后，您可以查看置信度分数，以确定是否有所改善。

Amazon Lex V2 返回的置信度分数是比较值。请避免将其视作绝对分数进行信任。根据对 Amazon Lex V2 的改进，这些值可能会发生变化。

Amazon Lex V2 会在每个响应中以 `interpretations` 结构返回最有可能的意图以及最多 4 个替代意图及其相关分数。以下 JSON 代码显示了 RecognizeText 操作的响应中的 `interpretations` 结构：

```
"interpretations": [
   {
      "intent": {
         "confirmationState": "string",
         "name": "string",
         "slots": {
            "string" : {
               "value": {
                  "interpretedValue": "string",
                  "originalValue": "string",
                  "resolvedValues": [ "string" ]
               }
            }
         },
         "state": "string"
      },
      "nluConfidence": number
```

```
    }
  ]
```

AMAZON.FallbackIntent

在以下两种情况下，Amazon Lex V2 将返回 `AMAZON.FallbackIntent` 作为首要意图：

1. 所有可能意图的置信度分数均小于置信度阈值。您可以使用默认阈值，也可以设置您自己的阈值。如果您已配置 `AMAZON.KendraSearchIntent`，Amazon Lex V2 在这种情况下也会返回该意图。
2. `AMAZON.FallbackIntent` 的解释置信度高于所有其他意图的解释置信度。

请注意，Amazon Lex V2 不会显示 `AMAZON.FallbackIntent` 的置信度分数。

设置和更改置信度阈值

置信度阈值必须是介于 0.00 与 1.00 之间的数字。您可以通过以下方式设置您的机器人中每种语言的阈值：

使用 Amazon Lex V2 控制台

- 要在使用添加语言向机器人添加语言时设置阈值，可以在可信度分数阈值面板中插入所需的值。
- 要更新阈值，您可以打开机器人的某个语言的语言详细信息面板，然后在该面板中选择编辑。然后在可信度分数阈值面板中插入所需的值。

使用 API 操作

- 要设置阈值，请设置 CreateBotLocale 操作的 `nluIntentConfidenceThreshold` 参数。
- 要更新置信度阈值，请设置 UpdateBotLocale 操作的 `nluIntentConfidenceThreshold` 参数。

会话管理

要更改 Amazon Lex V2 在与用户对话中使用的意图，您可以使用对话框代码挂钩 Lambda 函数中的响应，也可以在自定义应用程序中使用会话管理 API。

使用 Lambda 函数

当您使用 Lambda 函数时，Amazon Lex V2 会使用包含该函数输入的 JSON 结构对其进行调用。JSON 结构包含一个名为 `currentIntent` 的字段，该字段包含 Amazon Lex V2 已确定为最有可能的用户言语意图的意图。JSON 结构还包括一个 `alternativeIntents` 字段，该字段包含最多四

个可能满足用户意图的额外意图。每个意图都包含一个名为 nluIntentConfidenceScore 的字段，其中包含 Amazon Lex V2 指定给该意图的置信度分数。

要使用替代意图，您可以在 Lambda 函数的 ConfirmIntent 或 ElicitSlot 对话框操作中指定该意图。

有关更多信息，请参阅使用 AWS Lambda 函数启用自定义逻辑。

使用会话管理 API

要使用与当前意图不同的意图，请使用 PutSession 操作。例如，如果您认为第一个替代意图比 Amazon Lex V2 选择的意图更可取，则可以使用 PutSession 操作来更改意图，以便用户与之交互的下一个意图就是您选择的意图。

有关更多信息，请参阅使用 Amazon Lex V2 API 管理会话。

使用语音转录置信度分数

当用户说出语音言语时，Amazon Lex V2 会使用自动语音识别 (ASR) 对用户请求进行转录，然后再解释该用户请求。默认情况下，Amazon Lex V2 使用最有可能的音频转录进行解释。

在某些情况下，可能有多种可能的音频转录。例如，用户说出的言语可能模棱两可，例如用户说“My name is John”（我叫约翰），听起来可能像是“My name is Juan”（我叫胡安）。在这种情况下，您可以使用消除歧义的技术或将您的专有领域知识与转录置信度分数相结合，以帮助确定转录列表中正确的转录。

Amazon Lex V2 在对 Lambda 代码挂钩函数的请求中包括用户输入的最优转录和最多两个备用转录。每个转录都包含一个评估该转录的正确性的可信度分数。每个转录还包括从用户输入中推断出的任何槽位值。

您可以比较两个转录的置信度分数，以确定这两者之间是否存在歧义。例如，如果一个转录的置信度分数为 0.95，而另一个转录的置信度分数为 0.65，则第一个转录可能是正确的，并且这两者之间的歧义较低。如果两个转录的置信度分数分别为 0.75 和 0.72，则这两者之间的歧义性较高。您可以使用自己的专有领域知识加以区分。

例如，如果置信度分数分别为 0.75 和 0.72 的两个转录中推断的槽位值是“John”和“Juan”，则可以在数据库中执行用户查询以确定这些名称是否存在并删除其中一个转录。如果“John”不是数据库中的用户，而“Juan”是数据库中的用户，则可以使用对话框代码挂钩将名字的推断槽位值更改为“Juan”。

Amazon Lex V2 返回的置信度分数是比较值。请勿将其视作绝对分数进行信任。根据对 Amazon Lex V2 的改进，这些值可能会发生变化。

音频转录置信度分数仅适用于英语（英国）(en_GB) 和英语（美国）(en_US)。仅支持 8 kHz 音频输入的置信度分数。Amazon Lex V2 控制台上没有为来自测试窗口的音频输入提供转录置信度分数，因为 Amazon Lex V2 使用 16 kHz 的音频输入。

> **Note**
>
> 在对现有机器人使用音频转录置信度分数之前，必须先重新构建该机器人。现有版本的机器人不支持转录置信度分数。要使用转录置信度分数，必须创建新版本的机器人。

您可以将置信度分数用于多种对话设计模式：

- 如果由于环境嘈杂或信号质量差而导致最高置信度分数低于阈值，则可以向用户发送同样问题的提示以便捕获质量更好的音频。
- 如果多个转录对槽位值的置信度分数相似，例如“John”和“Juan”，则可以将这些值与预先存在的数据库进行比较以消除输入，也可以提示用户从这两个值中选择一个。例如，“say 1 for John or say 2 for Juan.”（如果是约翰，请说 1，如果是胡安，请说 2。）
- 如果您的业务逻辑要求在置信度分数接近最优转录的情况下，根据备用转录中的特定关键字进行意图切换，则可以使用对话框代码挂钩 Lambda 函数或使用会话管理操作来更改意图。有关更多信息，请参阅会话管理。

Amazon Lex V2 向 Lambda 代码挂钩函数发送以下 JSON 结构，其中包含针对用户输入的最多三个转录：

```
"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 结构包含转录文本、针对该言语解析的意图以及在该言语中检测到的任何槽位的值。对于文本用户输入，转录包含一个置信度分数为 1.0 的单个转录。

转录的内容取决于对话的回合以及所识别的意图。

就第一回合意图引发而言，Amazon Lex V2 确定前三个转录。对于最优转录，Amazon Lex V2 返回转录中的意图以及任何推断的槽位值。

对于随后的回合，即槽位引发回合，结果取决于每个转录的推断意图，如下所示。

- 如果最优转录的推断意图与上一回合相同，并且所有其他转录的意图相同，则
  - 所有转录都包含推断出的槽位值。

- 如果最优转录的推断意图与上一回合不同，并且所有其他转录具有上一个意图，则
  - 该最优转录包含新意图的推断槽位值。
  - 其他转录具有上一个意图以及该上一个意图的推断槽位值。

- 如果最优转录的推断意图与上一回合不同，一个转录与上一个意图相同，并且一个转录具有不同的意图，则
  - 该最优转录包含新的推断意图以及言语中的任何推断槽位值。
  - 具有上一个推断意图的转录包含该意图的推断槽位值。
  - 具有不同意图的转录不包含推断的意图名称，也不包含推断的槽位值。

- 如果最优转录的推断意图与上一回合不同，并且所有其他转录具有不同的意图，则
  - 该最优转录包含新的推断意图以及言语中的任何推断槽位值。
  - 其他转录不包含推断的意图，也不包含推断的槽位值。

- 如果前两个转录的推断意图与上一回合相同和不同，并且第三个转录具有不同的意图，则
  - 前两个转录包含新的推断意图以及言语中的任何推断槽位值。
  - 第三个转录不包含意图名称，也不包含已解析的槽位值。

会话管理

要更改 Amazon Lex V2 在与用户对话中使用的意图，请使用对话框代码挂钩 Lambda 函数中的响应。或者，您可以在自定义应用程序中使用会话管理 API。

使用 Lambda 函数

当您使用 Lambda 函数时，Amazon Lex V2 会使用包含该函数输入的 JSON 结构对其进行调用。JSON 结构包含一个名为 `transcriptions` 的字段，其中包含 Amazon Lex V2 为该言语确定的可能转录。该 `transcriptions` 字段包含一到三个可能的转录，每个转录都有对应的置信度分数。

要使用备用转录中的意图，请在 Lambda 函数的 `ConfirmIntent` 或 `ElicitSlot` 对话操作中指定该意图。要使用备用转录中的槽位值，请在 Lambda 函数响应的 `intent` 字段中设置该值。有关更多信息，请参阅使用 AWS Lambda 函数启用自定义逻辑。

示例代码

以下代码示例是一个 Python Lambda 函数，使用音频转录来改善用户的对话体验。

要使用示例代码，您必须满足以下条件：

- 拥有单语言的机器人，语言可以是英语（英国）(en_GB)或英语（美国）(en_US)。
- 拥有意图 OrderBirthStone。确保在意图定义的代码挂钩部分中选中使用 Lambda 函数进行初始化和验证。
- 该意图应有两个槽位“BirthMonth”和“Name”，这两个槽位应均为 `AMAZON.AlphaNumeric` 类型。
- 拥有定义了 Lambda 函数的别名。有关更多信息，请参阅创建 Lambda 函数并将其附加到机器人别名。

```
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)
```

使用会话管理 API

要使用与当前意图不同的意图，请使用 PutSession 操作。例如，如果您认为第一个替代意图比 Amazon Lex V2 选择的意图更可取，则可以使用 PutSession 操作来更改意图。这样，用户与之交互的下一个意图即为您选择的意图。

您也可以使用 PutSession 操作来更改 intent 结构中的槽位值，以使用替代转录中的值。

有关更多信息，请参阅使用 Amazon Lex V2 API 管理会话。

自定义语音转录

机器人的默认行为有时可能会导致语音转录不准确。以下功能可帮助您的机器人识别较不常见或容易混淆的单词或名称。

主题

- 使用自定义词汇改善语音识别
- 使用运行时提示改善对插槽值的识别
- 使用拼写样式捕获槽位值

使用自定义词汇改善语音识别

您可以创建特定语言的自定义词汇表，以便向 Amazon Lex V2 提供有关如何处理与机器人的音频对话的更多信息。自定义词汇表是一个列表，其中包含您希望 Amazon Lex V2 在音频输入中识别的特殊词。这些通常是 Amazon Lex V2 无法识别的专有名词或特定领域的专有词汇。

例如，假设您有一个技术支持机器人。您可以在自定义词汇表中添加“backup”（备份），以帮助机器人将音频正确地转录为“backup”，即使音频听起来像“pack up”（收拾行装）。自定义词汇还可以帮助识别音频中的生僻词汇，例如金融服务领域的“solvency”（偿付能力）或者诸如“Cognito”或“Monitron”等专有名词。

自定义词汇表基础知识

- 自定义词汇表用于将音频输入转录到机器人。您必须提供示例言语以便识别意图或插槽值。

- 自定义词汇表是特定语言所独有的。必须为每种语言单独配置自定义词汇表。只有英语（英国）和英语（美国）语言支持自定义词汇表。
- 自定义词汇表可配合 Amazon Lex V2 支持的联络中心集成使用。Amazon Lex V2 控制台中的测试窗口支持适用于 2022 年 7 月 31 日当天或之后构建的所有 Amazon Lex V2 机器人的自定义词汇表。如果您在测试窗口中遇到自定义词汇表的问题，请重新构建机器人并重试。

Amazon Lex V2 使用自定义词汇表来引发意图和插槽。相同的自定义词汇表文件也用于意图和插槽。添加插槽类型时，您可以选择关闭机器人的自定义词汇表功能。

引发意图：您可以创建用于引发意图的自定义词汇表。这些短语将在机器人确定用户的意图时用于转录。例如，如果您在自定义词汇表中配置了“backup”（备份）一词，Amazon Lex V2 会将用户输入的内容转录为"can you please backup my photos?"（“能备份我的照片吗？”），即使音频听起来像"can you please pack up my photos."（“能收拾我的照片吗？”）您可以通过配置权重 0、1、2 或 3 来指定每个短语的增强程度。您还可以通过添加 `displayAs` 字段来为最终语音到文本输出中的短语指定替代表示形式。

用于在引发意图期间改善转录的自定义词汇表短语不会影响引发插槽时的转录。有关创建用于引发意图的自定义词汇表的更多信息，请参阅创建自定义词汇表以引发意图和插槽。

引发自定义插槽：您可以使用自定义词汇表来改善音频对话的插槽识别。要提高 Amazon Lex V2 机器人识别插槽值的能力，请创建自定义插槽并将插槽值添加到自定义插槽，然后选择使用插槽值作为自定义词汇。插槽值的示例包括产品名称、目录或专有名词。自定义词汇表中应避免出现常用单词或短语，例如“是”和“否”。

添加插槽值后，这些值用于在机器人预期自定义插槽的输入时，改善插槽识别。这些值不用于在引发意图时的转录。有关更多信息，请参阅添加槽类型。

创建自定义词汇表的最佳实践

引发意图

- 自定义词汇表最适合用于目标特定单词和短语。仅将 Amazon Lex V2 不容易识别的单词添加到自定义词汇表中。
- 基于转录中单词不被识别的频率以及输入中出现该单词的罕见程度来决定赋予单词的权重。难以发音的单词需要更高的权重。
- 使用具有代表性的测试集来确定权重是否合适。您可以通过在对话日志中打开音频日志记录来收集音频测试集。

- 避免在自定义词汇表中使用诸如 “on”、“it”、“to”、“yes”、“no” 等短词。

引发自定义插槽

- 将值添加到您希望被识别的自定义插槽类型中。添加自定义插槽类型的所有可能的插槽值，无论该插槽值的常见或罕见程度。
- 仅当自定义插槽类型包含一系列诸如产品名称或共同基金等目录值或者实体，才启用该选项。
- 如果插槽类型用于捕获诸如“是”、“否”、“我不知道”、“也许”等通用短语或者诸如“一”、“二”、“三”等通用词，则禁用该选项。
- 为了获得最佳性能，请将插槽值和同义词的数量限制为 500 或以下。

输入首字母缩略词，或者其字母应以用圆点或空格隔开的单个字母的形式单独发音的其他单词。请勿使用单个字母，除非这些字母是短语的一部分，例如 “J.P.Morgan” 或 “A.W.S.”。您可以使用大写或小写字母来定义首字母缩略词。

创建自定义词汇表以引发意图和插槽

您可以使用 Amazon Lex V2 控制台来创建和管理自定义词汇表，也可以使用 Amazon Lex V2 API 操作。可通过以下两种方法，使用控制台来创建自定义词汇表：

控制台

在控制台中导入自定义词汇：

1. 访问 https://console.aws.amazon.com/lexv2/home，打开 Amazon Lex V2 控制台。
2. 从机器人列表中，选择要添加自定义词汇表的机器人。
3. 在机器人详情页面上，在添加语言部分中选择查看语言。
4. 从语言列表中，选择要添加自定义词汇表的语言。

直接通过控制台创建新的自定义词汇表：

1. 在语言详细信息页面的自定义词汇部分中点击创建。系统将打开编辑窗口，该窗口中当前不存在自定义词汇。
2. 根据需要输入 “phrase”、“DisplayAs” 和 “weight” 的值。您可以更新已添加项目的字段或将其从列表中删除，从而进一步对添加的项目进行内联编辑。
3. 点击保存。注意：只有在您点击保存后，新的自定义词汇才会保存在您的机器人中。

4. 您可以继续在此页面中进行内联编辑，完成后点击保存。
5. 此外，您还可以通过此页面右上角的下拉菜单来导入、导出和删除自定义词汇表文件。

API

使用 **ListCustomVocabularyItems** API 查看自定义词汇表条目：

1. 使用 ListCustomVocabularyItems 操作查看自定义词汇表条目。请求正文如下：

```
{
   "maxResults": number,
   "nextToken": "string"
}
```

2. 请注意，maxResults 和 nextToken 是请求正文的可选字段。
3. ListCustomVocabularyItems 操作的响应如下：

```
{
   "botId": "string",
   "botVersion": "string",
   "localeId": "string",
   "customVocabularyItems": [
        {
            "itemId": "string",
            "phrase": "string",
            "weight": number,
            "displayAs": "string"
        }
   ]
}
```

使用 **BatchCreateCustomVocabularyItem** API 创建新的自定义词汇表条目：

1. 如果尚未对您机器人的区域设置创建自定义词汇表，请按照使用 StartImport 创建自定义词汇表的步骤操作。
2. 创建自定义词汇表后，使用 BatchCreateCustomVocabularyItem 操作创建新的自定义词汇表条目。请求正文如下：

```
{
```

```
   "customVocabularyItemList": [
        {
            "phrase": "string",
            "weight": number,
            "displayAs": "string"
        }
   ]
}
```

3. 请注意，weight 和 displayAs 是请求正文的可选字段。
4. BatchCreateCustomVocabularyItem 的响应如下：

```
{
    "botId": "string",
    "botVersion": "string",
    "localeId": "string",
    "errors": [
        {
            "itemId": "string",
            "errorMessage": "string",
            "errorCode": "string"
        }
    ],
    "resources": [
        {
            "itemId": "string",
            "phrase": "string",
            "weight": number,
            "displayAs": "string"
        }
    ]
}
```

5. 由于这是批量操作，因此如果其中一个项目创建失败，请求不会失败。错误列表将包含有关该特定条目的操作失败的原因的信息。资源列表将包含所有成功创建的条目。
6. 对于 BatchCreateCustomVocabularyItem，可能会出现以下类型的错误：

   - RESOURCE_DOES_NOT_EXIST：自定义词汇表不存在。调用此操作之前，请按照创建自定义词汇表的步骤操作。
   - DUPLICATE_INPUT：输入列表中包含重复的短语。
   - RESOURCE_ALREADY_EXISTS：您的自定义词汇表中已存在该条目的给定短语。

- INTERNAL_SERVER_FAILURE：在处理您的请求时，后端出现错误。这可能表示服务中断或其他问题。

使用 **BatchDeleteCustomVocabularyItem** API 删除现有的自定义词汇表条目：

1. 如果尚未对您机器人的区域设置创建自定义词汇表，请按照使用 StartImport 创建自定义词汇表的步骤操作，以便创建自定义词汇表。
2. 创建自定义词汇表后，使用 BatchDeleteCustomVocabularyItem 操作删除现有的自定义词汇表条目。请求正文如下：

```
{
   "customVocabularyItemList": [
        {
            "itemId": "string"
        }
   ]
}
```

3. BatchDeleteCustomVocabularyItem 的响应如下：

```
{
    "botId": "string",
    "botVersion": "string",
    "localeId": "string",
    "errors": [
        {
            "itemId": "string",
            "errorMessage": "string",
            "errorCode": "string"
        }
    ],
    "resources": [
        {
            "itemId": "string",
            "phrase": "string",
            "weight": number,
            "displayAs": "string"
        }
    ]
}
```

4. 由于这是批量操作，因此如果其中一个项目删除失败，请求不会失败。错误列表将包含有关该特定条目的操作失败的原因的信息。资源列表将包含所有成功删除的条目。
5. 对于 BatchDeleteCustomVocabularyItem，可能会出现以下类型的错误：
   - RESOURCE_DOES_NOT_EXIST：您尝试删除的自定义词汇表条目不存在。
   - INTERNAL_SERVER_FAILURE：在处理您的请求时，后端出现错误。这可能表示服务中断或其他问题。

使用 **BatchUpdateCustomVocabularyItem** API 更新现有的自定义词汇表条目：

1. 如果尚未对您机器人的区域设置创建自定义词汇表，请按照使用 StartImport 创建自定义词汇表的步骤操作，以便创建自定义词汇表。
2. 创建自定义词汇表后，使用 BatchUpdateCustomVocabularyItem 操作更新现有的自定义词汇表条目。请求正文如下：

```
{
   "customVocabularyItemList": [
        {
            "itemId": "string",
            "phrase": "string",
            "weight": number,
            "displayAs": "string"
        }
   ]
}
```

3. 请注意，weight 和 displayAs 是请求正文的可选字段。
4. BatchUpdateCustomVocabularyItem 的响应如下：

```
{
    "botId": "string",
    "botVersion": "string",
    "localeId": "string",
    "errors": [
        {
            "itemId": "string",
            "errorMessage": "string",
            "errorCode": "string"
        }
    ],
```

```
    "resources": [
        {
            "itemId": "string",
            "phrase": "string",
            "weight": number,
            "displayAs": "string"
        }
    ]
}
```

5. 由于这是批量操作，因此如果其中一个项目删除失败，请求不会失败。错误列表将包含有关该特定条目的操作失败的原因的信息。资源列表将包含所有成功更新的条目。
6. 对于 BatchUpdateCustomVocabularyItem，可能会出现以下类型的错误：

   - RESOURCE_DOES_NOT_EXIST：您尝试更新的自定义词汇表条目不存在。
   - DUPLICATE_INPUT：输入列表中包含重复的 itemId。
   - RESOURCE_ALREADY_EXISTS：您的自定义词汇表中已存在该条目的给定短语。
   - INTERNAL_SERVER_FAILURE：在处理您的请求时，后端出现错误。这可能表示服务中断或其他问题。

创建自定义词汇表文件

自定义词汇表文件是一个以制表符分隔的值列表，其中包含要识别的短语、要增强的权重以及将替换语音转录中短语的 displayAs 字段。增强值较高的短语当在音频输入中出现时，更有可能被使用。

自定义词汇表文件必须命名为 **CustomVocabulary.tsv**，并且必须将其压缩为 ZIP 文件后才能导入。ZIP 文件必须小于 300 MB。自定义词汇表中的最大短语数量为 500。

- phrase：应识别的 1 到 4 个单词。短语中的单词以空格隔开。文件中不得有重复的短语。“短语”字段为必填项。
- weight：指定短语识别的增强程度。该值是整数 0、1、2 或 3。如果未指定权重，则默认值为 1。基于转录中单词不被识别的频率以及输入中出现该单词的罕见程度来决定权重。权重 0 表示不会应用任何增强，并且该条目将仅用于使用 displayAs 字段执行替换。
- displayAs：依据您的需求定义转录输出中对该短语的显示。这是自定义词汇表中的可选字段。

自定义词汇表文件必须包含标题为 “phrase”、“weight” 和 “displayAs” 的标题行。这些标题可以按任意顺序排列，但必须遵循上述命名法。

以下是一个自定义词汇表文件的示例。分隔 “phrase”、“weight” 和 “displayAs” 所需的制表符用文本 “[TAB]” 表示。如果您使用此示例，请将文本替换为制表符。

```
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
```

使用运行时提示改善对插槽值的识别

借助运行时提示，您可以根据上下文为 Amazon Lex V2 提供一组插槽值，以便改善音频对话中的识别效果并改善插槽的解析。您可以使用运行时提示在运行时提供一系列短语列表，作为插槽值解析的候选短语。

例如，如果与航班预订机器人交互的用户经常飞往旧金山、雅加达、首尔和莫斯科，则可以在引发目的地时使用这四个城市的列表配置运行时提示，以提高对常飞城市的识别能力。

运行时提示仅适用于英语（美国）和英语（英国）语言。运行时提示可以与以下插槽类型配合使用：

- 自定义插槽类型
- AMAZON.City
- AMAZON.Country
- AMAZON.FirstName
- AMAZON.LastName
- AMAZON.State
- AMAZON.StreetName

运行时提示基础知识

- 仅在从用户引发插槽值时才使用运行时提示。
- 使用运行时提示时，提示的值优先于类似的值。例如，对于点餐机器人，您可以在自定义插槽中将一系列菜单餐品设置为在引发具体餐点时使用的运行时提示，以便将“fillet”（菲力）优先于发音相近的“fella”（小伙子）。
- 如果用户输入与运行时提示中提供的值不同，则将原始用户输入用于插槽。

- 对于自定义插槽类型，作为运行时提示提供的值将用于解析插槽，即使这些值在机器人创建期间不属于自定义插槽也是如此。
- 仅支持 8 kHz 音频输入的运行时提示。运行时提示可配合 Amazon Lex V2 支持的联络中心集成使用。Amazon Lex V2 控制台上没有为来自测试窗口的音频输入提供运行时提示，因为 Amazon Lex V2 使用 16 kHz 的音频输入。

> **Note**
>
> 为现有机器人使用运行时提示之前，请首先重新构建该机器人。机器人的现有版本不支持运行时提示。要使用运行时提示，必须创建新版本的机器人。

你可以使用 PutSession、RecognizeText、RecognizeUtterance 或 StartConversation 操作向 Amazon Lex V2 发送运行时提示。您也可以使用 Lambda 函数添加运行时提示。

您可以在对话开始时发送运行时提示，为机器人中使用的每个插槽配置提示，也可以在对话期间将提示作为会话状态的一部分发送。`runtimeHints` 属性将一个插槽映射到该插槽的提示。

一旦您向 Amazon Lex V2 发送运行时提示，这些运行时提示将在整个对话的每一回合中持续存在，直到会话结束。如果您发送的是 null `runtimeHints` 结构，则使用现有的提示。您可以通过如下方式修改提示：

- 向机器人发送新 `runtimeHints` 结构。新结构的内容取代了现有结构。
- 向机器人发送空的 `runtimeHints` 结构。这将清除机器人的运行时提示。

在上下文中添加插槽值

当您的应用程序包含有关用户下一个可能的言语的信息时，通过提供预期的插槽值作为运行时提示，为您的机器人添加上下文。向您的机器人添加 Lambda 对话框代码挂钩（有关更多信息，请参阅使用 AWS Lambda 函数启用自定义逻辑），然后使用 解释输入事件格式 中的 proposedNextState 字段来确定为改善与用户的对话而应包含的运行时提示。

例如，在银行应用程序中，您可以为特定用户生成账户昵称列表，然后在引发该用户想要访问的账户时使用该列表。

当您有上下文来帮助机器人解释用户输入时，请在对话开始时发送运行时提示。例如，如果您有用户的电话号码，则可以使用此信息来查找用户，以便在您引发用户名以验证用户凭证时，可以使用 `PutSession` 或 `StartConversation` 操作向机器人传递用户名字和姓氏的提示。

在对话期间，您可能会从一个插槽值中收集有助于处理另一个插槽值的信息。例如，在汽车保养应用程序中，当您有用户的账号时，可以查找客户拥有的车辆，然后将其作为提示传递给另一个插槽。

输入首字母缩略词，或者其字母应以用圆点或空格隔开的单个字母的形式单独发音的其他单词。除非单个字母是短语的一部分，例如“J. P. Morgan”或“A.W.S”，否则请勿使用这些字母。您可以使用大写或小写字母来定义首字母缩略词。

向插槽添加提示

要向插槽添加运行时提示，请使用 `runtimeHints` 结构，该结构是 `sessionState` 结构的一部分。以下是 `runtimeHints` 结构的示例。该结构针对“MakeAppointment”的意图，提供适用于两个插槽“FirstName”和“LastName”的提示。

```
{
    "sessionState": {
        "intent": {},
        "activeContexts": [],
        "dialogAction": {},
        "originatingRequestId": {},
        "sessionAttributes": {},
        "runtimeHints": {
            "slotHints": {
                "MakeAppointment": {
                    "FirstName": {
                        "runtimeHintValues": [
                            {
                                "phrase": "John"
                            },
                            {
                                "phrase": "Mary"
                            }
                        ]
                    },
                    "LastName": {
                        "runtimeHintValues": [
                            {
                                "phrase": "Stiles"
                            },
                            {
                                "phrase": "Major"
                            }
                        ]
                    }
```

```
                }
            }
        }
    }
}
```

您还可以在对话期间使用 Lambda 函数添加运行时提示。要添加运行时提示，您可以将 `runtimeHints` 结构添加到 Lambda 函数发送到 Amazon Lex V2 的响应的会话状态中。有关更多信息，请参阅准备响应格式。

您需要在请求中指定有效的 `intentName` 和 `slotName`，否则 Amazon Lex V2 会返回运行时错误。

使用拼写样式捕获槽位值

Amazon Lex V2 提供内置槽位，用于捕获用户特定的信息，例如名字、姓氏、电子邮件地址或字母数字标识符。例如，您可以使用 `AMAZON.LastName` 槽位捕获诸如“Jackson”或“Garcia”等姓氏。但是，Amazon Lex V2 可能会混淆难以发音或在对应区域设置中不常见的姓氏，例如“Xiulan”。要捕获此类姓氏，您可以要求用户以逐个字母拼写或逐个单词拼写样式提供输入。

Amazon Lex V2 提供了三种槽位引发样式供您使用。您设置槽位引发样式后，会改变 Amazon Lex V2 解释用户输入的方式。

逐个字母拼写：您可以使用此样式，以指示机器人听拼写而不是整个短语。例如，要捕获诸如“Xiulan”等姓氏，您可以指示用户逐个字母地说出自己的姓氏。机器人将捕获该拼写并将这些字母解析为单词。例如，如果用户说“x i u l a n”，则机器人会将姓氏捕获为“xiulan”。

逐个单词拼写：在语音对话中，尤其是在使用电话时，有几个字母会听起来相似，例如“t”、“b”、“p”。当捕获字母数字值或拼写名称导致值不正确时，您可以提示用户输入该字母的同时提供识别词。例如，如果对预订 ID 请求的语音响应为“abp123”，则您的机器人可能会改为识别短语“abb123”。如果该值不正确，您可以指示用户将输入提供为“a as in alpha b as in boy p as in peter one two three”（alpha 的 a，boy 的 b，peter 的 p，然后是 123）。机器人会将输入解析为“abp123”。

使用逐个单词的拼写样式时，您可以使用以下格式：

- “as in”(a as in apple)
- “for”(a for apple)
- “like”(a like apple)

默认：使用单词发音进行槽位捕获的自然样式。例如，可以自然地捕获诸如“John Stiles”等姓名。如果未指定槽位引发样式，则机器人将使用默认样式。对于 AMAZON.AlphaNumeric 和 AMAZON.UKPostal 代码槽位类型，默认样式支持逐个字母拼写的输入。

如果使用字母和单词组合说出名字“Xiulan”，例如“x as in x-ray i u l as in lion a n”，则必须将槽位引发样式设置为逐个单词拼写的样式。使用逐个字母拼写的样式将无法识别该名字。

请创建一个语音界面，用于以自然对话样式来捕获槽位值，以获得更好的体验。对于使用自然样式无法正确捕获的输入，您可以重新提示用户，并将槽位引发样式设置为逐个字母拼写或逐个单词拼写。

对于英语（美国）、英语（英国）和英语（澳大利亚），您可以对以下槽位类型使用逐个单词拼写以及逐个字母拼写样式：

- 亚马逊。 AlphaNumeric
- 亚马逊。 EmailAddress
- 亚马逊。 FirstName
- 亚马逊。 LastName
- AMAZON.UK PostalCode
- 自定义槽位类型

启用拼写

您可以在运行时启用在从用户引发槽位时的逐个字母拼写以及逐个单词拼写。您可以使用 PutSession、RecognizeText、RecognizeUtterance 或 StartConversation 操作来设置拼写样式。您还可以使用 Lambda 函数启用逐个字母拼写以及逐个单词拼写。

在上述任一 API 操作的请求中或配置 Lambda 响应时，您可以使用 sessionState 字段的 dialogAction 字段来设置拼写样式（有关更多信息，请参阅准备响应格式）。只有当对话框操作类型为 ElicitSlot 并且要引发的槽位是支持的槽位类型之一时，您才能设置样式。

以下 JSON 代码显示了设置为使用逐个单词拼写样式的 dialogAction 字段：

```
"dialogAction": {
    "slotElicitationStyle": "SpellByWord",
    "slotToElicit": "BookingId",
    "type": "ElicitSlot"
}
```

slotElicitationStyle 字段可设置为 SpellByLetter、SpellByWord 或 Default。如果您不指定值，则值设置为 Default。

> **Note**
>
> 您无法通过控制台启用逐个字母拼写或逐个单词拼写的引发样式。

示例代码

如果第一次尝试解析槽位值不起作用，则通常会更改拼写样式。以下代码示例是一个 Python Lambda 函数，在第二次尝试解析槽位时使用逐个单词拼写样式。

要使用示例代码，您必须满足以下条件：

- 拥有单语言，即英语（英国）(en_GB) 的机器人。
- 拥有一个意图“CheckAccount”，该意图中具有示例言语：“I would like to check my account”。确保在意图定义的代码挂钩部分中选中使用 Lambda 函数进行初始化和验证。
- 意图应该有 AMAZON.UKPostalCode 内置类型的槽位“PostalCode”。
- 拥有定义了 Lambda 函数的别名。有关更多信息，请参阅创建 Lambda 函数并将其附加到机器人别名。

```
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
```

监控机器人的性能

要维护 Amazon Lex V2 聊天机器人的可靠性、可用性和性能，实施监控非常重要。本主题介绍如何使用对话日志来监控用户与聊天机器人之间的对话，使用话语统计数据来确定机器人检测到和错过的话语，以及如何使用 Amazon Lo CloudWatch gs 和监控 AWS CloudTrail Amazon Lex V2。此外还说明了 Amazon Lex V2 运行时和频道关联指标。

利用这些工具和指标，您可以了解自己需要采取哪些策略和措施来提高机器人的性能。

主题

使用 Analytics 衡量业务绩效

借助 Analytics，您可以评估机器人与客户交互的成功率和失败率相关指标，以此来评估机器人的性能。您还可以直观地查看机器人与客户之间的对话流程模式。Analytics 以图形和图表的形式来汇总这些指标，从而简化使用体验。Analytics 中的相关工具可用于筛选结果，从而帮助您辨别出涉及意图、槽位、言语和对话的问题。您可以使用这些数据对机器人进行迭代和改进，从而创造更好的客户体验。

> **Note**
>
> 要访问 Analytics，用户的 IAM 角色 中需要附有 AWS 托管策略： AmazonLexFullAccess 或者包含分析 API 权限的自定义策略。有关如何使用自定义策略处理用户权限的详细信息，请参阅管理分析的访问权限。如果将 AWS 托管策略： AmazonLexReadOnly 附加到客户的 IAM 角色中，则系统将显示错误消息，以提示您需要添加到用户 IAM 角色中的权限，以便用户能够访问 Analytics 控制面板。

要访问 Analytics，请执行以下操作：

1. 登录 AWS Management Console 并打开 Amazon Lex V2 主机，网址为 https://console.aws.amazon.com/lexv2/home。

2. 在机器人下的导航窗格中，选择要在分析中查看的机器人。
3. 在分析下选择要查看的部分。

主题

关键定义

本主题提供了关键定义，以帮助您更好地理解机器人分析。这些定义涉及机器人的以下性能方面：意图、槽位、对话和言语。以下字段与许多性能指标相关：

- Intent 对象的 `state` 字段。
- `dialogAction`对象内对象的`type`字段。SessionState

意图

Amazon Lex V2 通过以下方式对意图分类：

- 成功：机器人成功履行了意图。满足以下条件之一：
  - 意图 `state` 是 `ReadyForFulfillment`，并且 `dialogAction` 的 `type` 是 `Close`。
  - 意图 `state` 是 `Fulfilled`，并且 `dialogAction` 的 `type` 是 `Close`。
- 失败：机器人未能履行意图。意图状态。满足以下条件之一：
  - 意图 `state` 是 `Failed`，并且 `dialogAction` 的 `type` 是 `Close`（例如，用户拒绝了确认提示）。
  - 机器会在意图完成之前切换到 `AMAZON.FallbackIntent`。
- 已切换：在最初的意图被归类为成功或失败之前，机器人识别出另一个不同意图并且切换到该意图。
- 已删除：在该意图被归类为成功或失败之前，客户没有做出回应。

槽值

Amazon Lex V2 通过以下方式对槽位分类：

- 成功：机器人已填充该槽位并且成功过渡到另一个槽位或确认步骤。
- 失败：即使达到最大重试次数，机器人也无法填充该槽位。
- 已删除：在将槽位归类为成功或失败之前，客户没有做出回应或切换到其他意图。

对话

当客户在对 Amazon Lex V2 进行运行时调用时提供 `sessionId`，然后 Amazon Lex V2 会生成 `originatingRequestId`。如果客户未在您为机器人设置的会话超时 (`idleSessionTTLInSeconds`) 内做出响应，则会话将过期。如果客户使用相同的 `sessionId` 返回会话，Amazon Lex V2 将生成新的 `originatingRequestId`。

对于分析而言，对话是指由 `sessionId` 与 `originatingRequestId` 构成的唯一组合。Amazon Lex V2 通过以下方式将对话进行分类：

- 成功：对话的最终意图被归类为成功。
- 失败：对话的最终意图失败。如果 Amazon Lex V2 默认设置为 AMAZON.FallbackIntent，则对话也被归类为失败。
- 已删除：在该对话被归类为成功或失败之前，客户没有做出回应。

言语

Amazon Lex V2 通过以下方式对言语分类：

- 检测到：Amazon Lex V2 将该言语识别为试图调用为机器人配置的意图。
- 遗漏：Amazon Lex V2 未能识别出该言语。

筛选结果

您可以在各页面的顶部筛选机器人分析的结果。
适用于分析的筛选选项。

您可以使用以下参数作为筛选条件：

- 时间：您可以按相对或绝对时间范围来筛选结果。您选择开始和结束时间之后，Amazon Lex V2 将检索于开始时间之后开始并且在结束时间之前结束的对话。
  - 相对范围：选择 1d 可查看过去一天的结果，选择 1w 可查看过去一周的结果，或者选择 1m 可查看过去一个月的结果。

    如需更多选项，请选择自定义，然后在相对范围菜单中选择一个持续时间。如需设置持续时间的更多参数，请选择自定义范围，在持续时间字段中输入相应的数字，然后从下拉菜单中选择时间单位。
  - 绝对范围：选择自定义，然后选择绝对范围菜单，以筛选指定时间范围内的对话。您可以在日历上选择开始和结束日期，也可以以 YYYY/MM/DD 格式输入。

  > **Note**
  >
  > 您最多可以筛选 30 天内的结果。

- 机器人筛选:要按机器人的区域设置、别名和版本进行筛选，请选择标有所有区域、所有别名和所有版本的下拉菜单。
- 模式：选中齿轮图标，然后在模式下拉菜单中选择显示语音或文本结果。
- 频道：选中齿轮图标，然后在频道下拉菜单中选择要显示结果的频道。有关频道集成的更多信息，请参阅将 Amazon Lex V2 机器人与消息收发平台集成和 Amazon Connect 联络中心

概述：机器人性能总览

概述页面对机器人在对话、言语识别和意图使用方面的性能进行汇总。该页面包含以下部分：

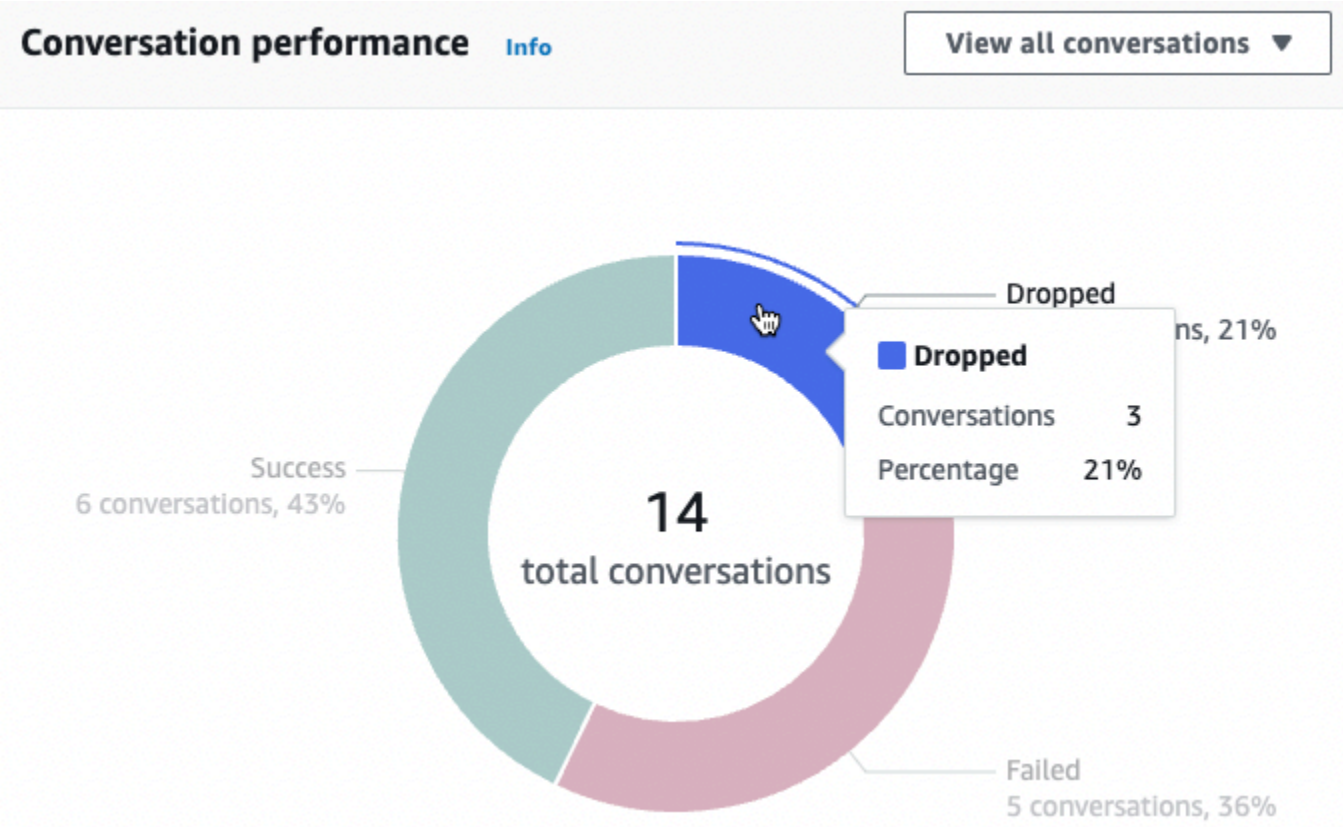
- 对话性能
- 言语识别率
- 对话性能历史数据
- 最常被使用的 5 个意图
- 最常失败的 5 个意图

对话性能

使用此图表来跟踪归类为成功、失败和已删除的对话的数量和百分比。要访问对话列表，请选择查看所有对话以展开下拉菜单。您可以选择查看用户与机器人的所有对话的列表，也可以筛选出特定结果（成

功、失败或已删除）的对话。点击这些链接可跳转到对话控制面板的对话子部分。有关更多信息，请参阅 对话。

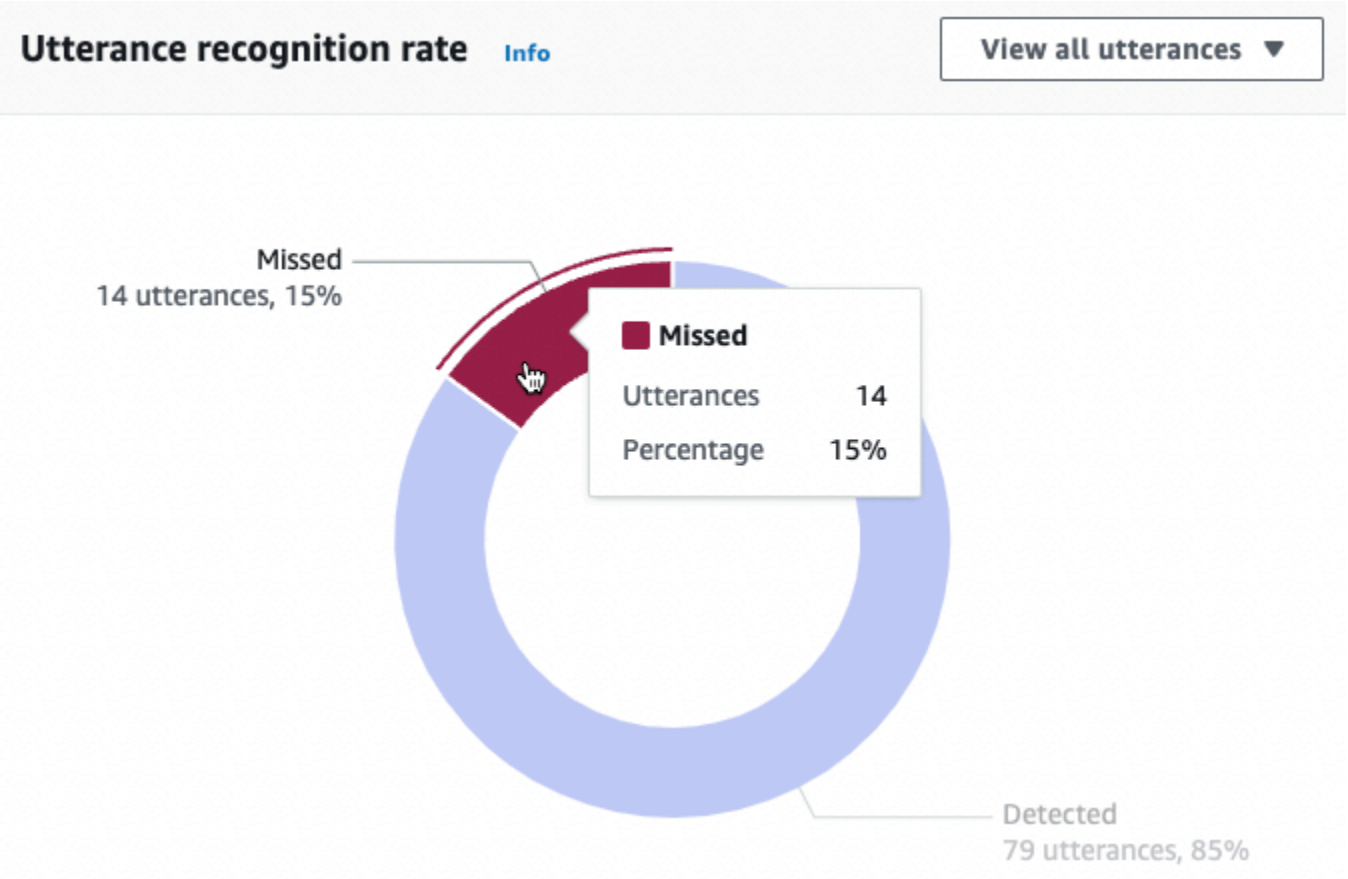
要查看图表中某一部分的对话数量和百分比，请将鼠标悬停在该部分上，如下图所示。



言语识别率

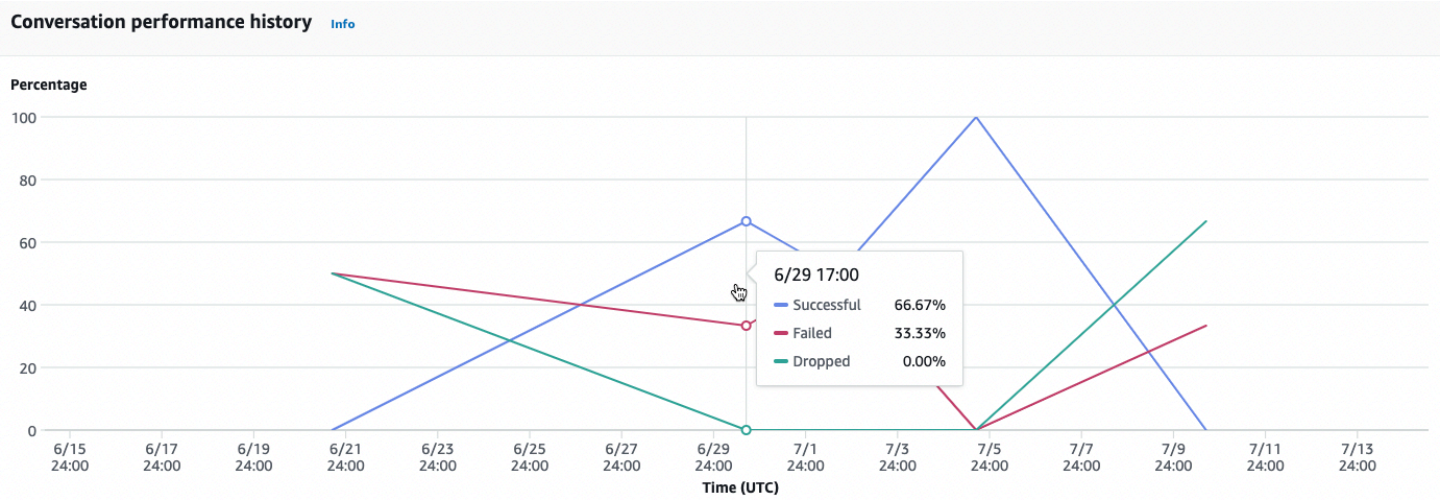
使用此图表来跟踪机器人检测到和遗漏的言语的数量和百分比。要访问言语列表，请选择查看言语以展开下拉菜单。您可以选择查看所有用户言语的列表，也可以筛选出特定结果（遗漏或检测到）的言语。点击这些链接可跳转到性能控制面板的言语识别子部分。有关更多信息，请参阅查看言语以导航至言语识别。

要查看图表中某一部分的言语数量和百分比，请将鼠标悬停在该部分上，如下图所示。



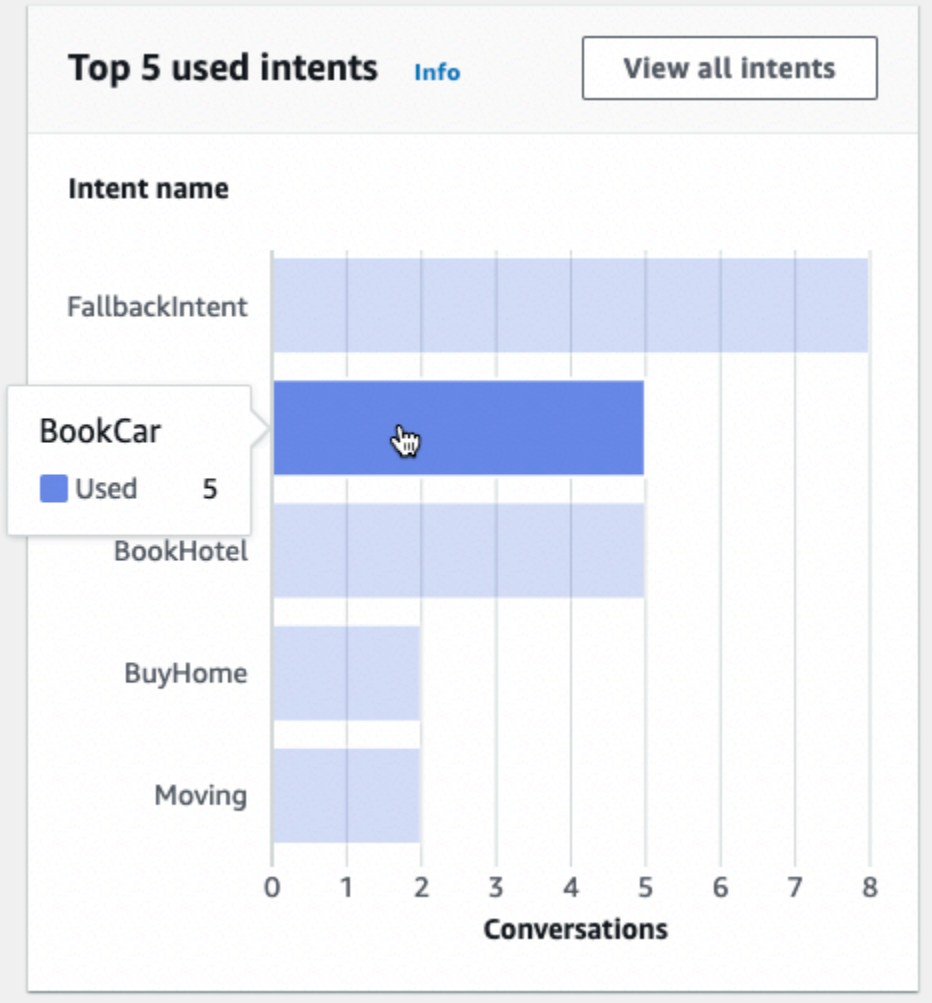
对话性能历史数据

使用此图表可以在筛选条件中设置的时间范围内跟踪被归类为成功、失败和已删除的对话的百分比。要查看某一时间间隔内特定结果的对话的百分比，请将鼠标悬停在该间隔上，如下图所示。



最常被使用的 5 个意图

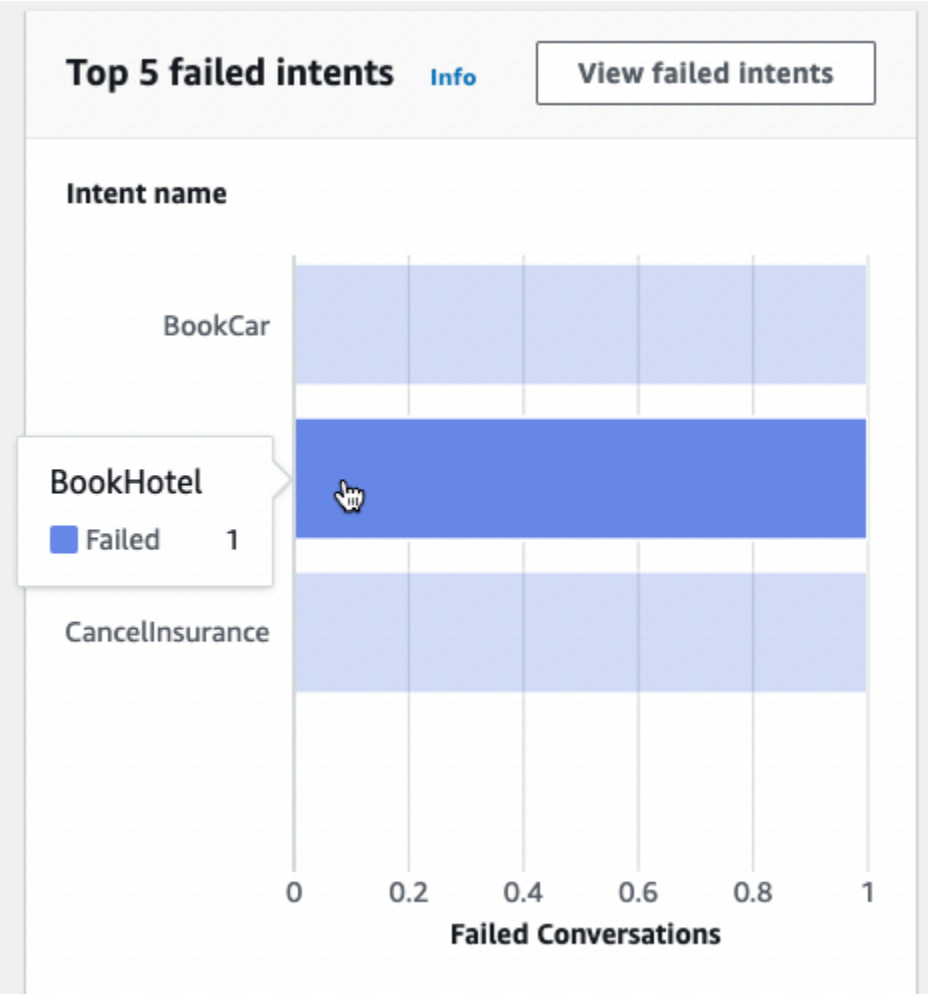
使用此图表来确定机器人中最常被客户使用的五大意图。将鼠标悬停在某个条形上可查看机器人识别出该意图的次数，如下图所示。



选择查看所有意图，导航至性能控制面板的意图性能子部分，可查看机器人在履行意图方面的性能指标。有关更多信息，请参阅 意图性能。

最常失败的 5 个意图

使用此图表来确定机器人未能履行的前五个意图（有关失败意图的定义，请参阅意图）。将鼠标悬停在某个条形上可查看机器人未能履行该意图的次数，如下图所示。



选择查看失败的意图，导航至性能控制面板的意图性能子部分，可查看机器人未能履行的意图的指标。有关更多信息，请参阅 意图性能。

对话控制面板：机器人对话概览

对话控制面板用以直观地展示客户与机器人之间对话的相关指标（有关对话的定义，请参阅对话）。

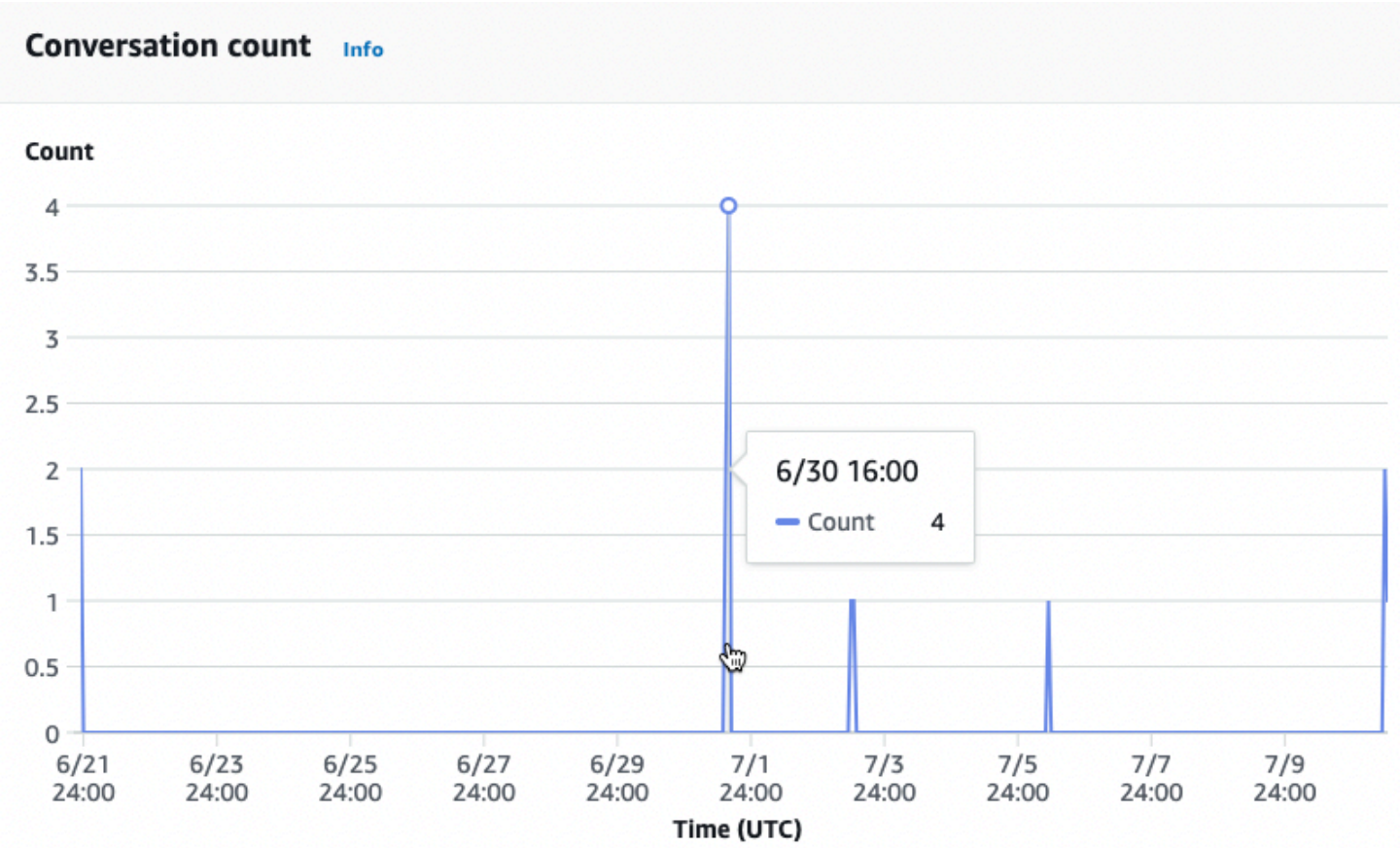
摘要部分展示关于用户与机器人之间对话的以下信息。这些数字是根据筛选设置计算得出的。

- 对话总数：与机器人的对话的总数。
- 对话平均持续时间：用户与机器人之间对话的平均时间（以分钟和秒为单位）。格式为 mm:ss。
- 每次对话的平均回合数：对话的平均回合数。

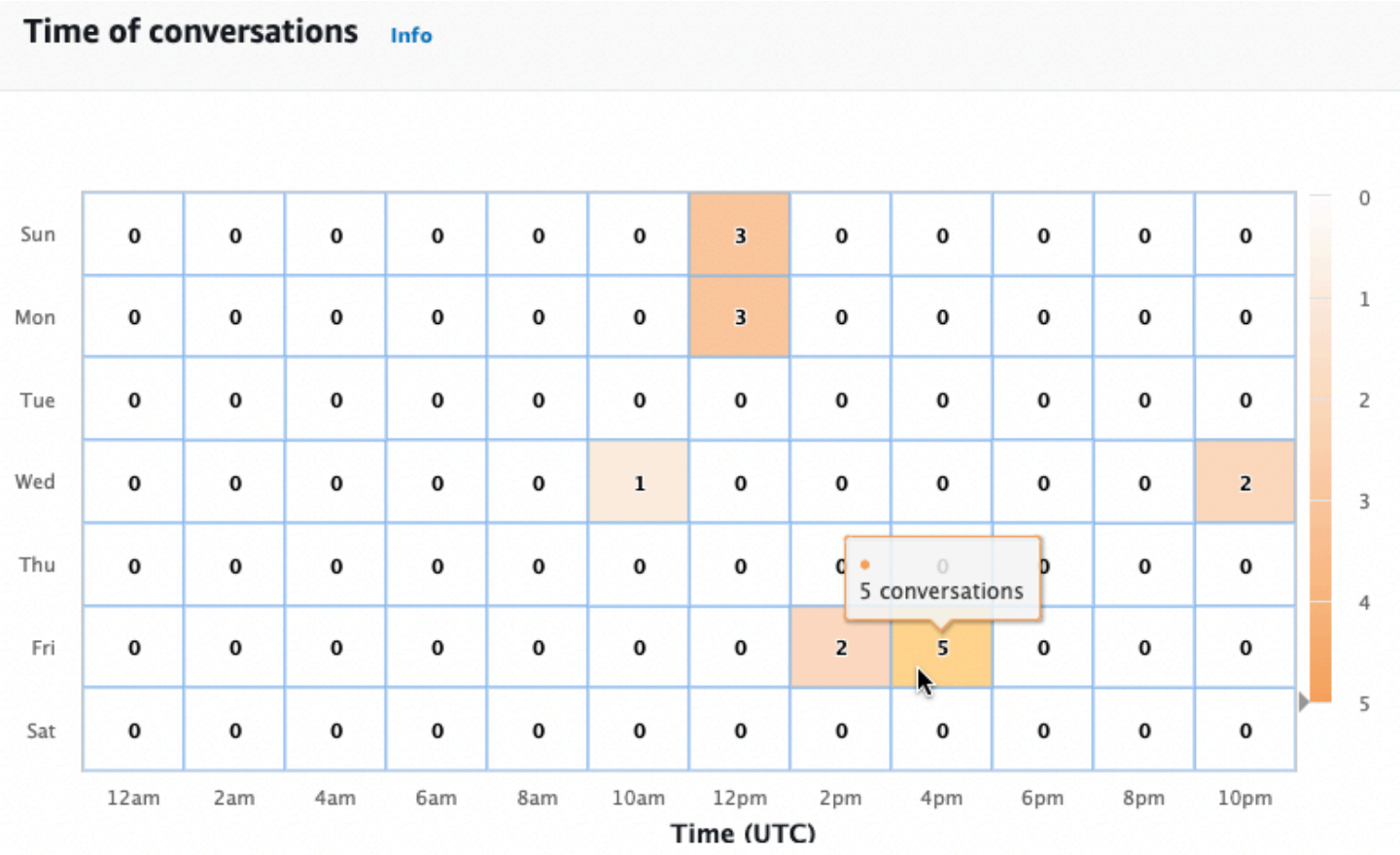
对话计数和消息计数部分中均以图表形式分别显示在筛选条件中指定的时间范围内的对话和消息数量。将鼠标悬停在某个时间段上，即可查看该时间段内的对话或消息数量。时间段的长短取决于您所指定的时间范围：

- 1 周内：显示每小时的计数。
- 1 周或更长时间：显示每天的计数。

关于将鼠标悬停在某个时间段上的显示效果，请参见下图。



对话时间部分基于您在筛选条件中所指定的时间范围，以每两小时的时间间隔展示在一周中的每一天中，机器人与客户进行的对话数量。阴影较深的单元格表示发生较多对话的时间。将鼠标悬停在某个单元格上可查看从该时间段开始 2 小时内的对话数量。例如，下图中的操作显示了下午 4:00 至下午 6:00 (UTC) 之间发生的对话数量。



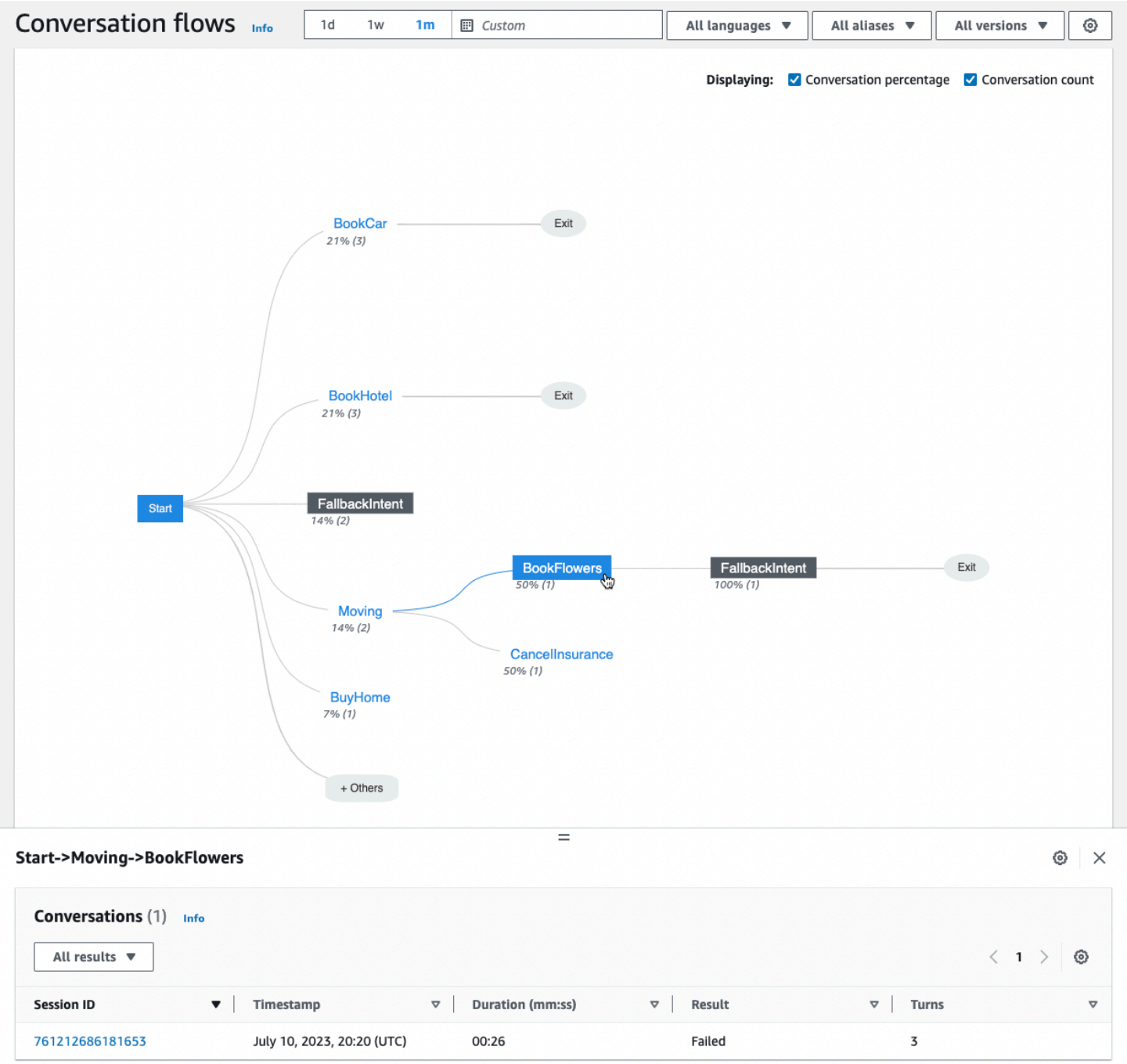
对话控制面板包含两个工具，即对话流程和对话。要访问某个工具，请在左侧导航窗格的对话控制面板下选择该工具。

对话流程

使用对话流程可直观地了解客户在与机器人对话中所采用的意图顺序。每个意图下方将给出在对话当时调用该意图的对话的百分比和数量。在顶部选择对话百分比和对话计数可切换所显示的数据。默认情况下，对话当时最常用的五大意图将按频率降序显示。选择 +其他可显示所有意图。

选择某个意图可展开一个新的分支列，其中汇总了在对话当时的一系列意图，按频率降序排序。

选中对话流程中的某个节点后，系统将在下方展开一个窗口，以显示遵循该意图顺序的对话列表。选择与某个对话对应的会话 ID 可查看有关该对话的详细信息。下图显示了对话流程以及底部展开的对话窗口。



对话

对话工具会显机器人的对话列表。您可以选中一列，以便按该列以升序或降序排序。

要按结果筛选对话，请选择所有结果，然后选择成功、失败或已删除。

要按持续时间筛选对话，请执行以下操作：

1. 选择标有按持续时间筛选对话的搜索栏。
2. 通过以下方式之一定义筛选条件：
   - 使用预定义的选项。
     a. 选择持续时间。
     b. 设置运算符为 =（等于）、>（大于）或 <（小于）。
     c. 选择时间长度。
   - 以 “Duration {operator} {number} sec”的格式输入。例如，要搜索持续时间在 30 秒以上的所有对话，请输入 **`Duration > 30 sec`**。指定时间长度（以秒为单位）。

要查看关于该会话的详细信息（包括元数据、意图使用率和转录），请选择对话的会话 ID。

> **Note**
>
> 由于对话是 `sessionId` 和 `originatingRequestId` 的唯一组合，因此相同的 `sessionId` 可能会在表格中多次出现。

详细信息部分包含以下元数据：

- 时间戳：指定对话的日期和开始时间。时间采用 hh: mm: ss 格式。
- 持续时间：以 mm: ss 格式指定对话的持续时间。该持续时间不包括会话超时持续时间 (`idleSessionTTLInSeconds`)。
- 结果：指定对话是被归类为成功、失败还是已删除。有关这些结果的详细信息，请参阅对话。
- 模式：指定对话是 `Speech`、`Text` 或 `DTMF`（按键键盘输入）。由多种模式组成的对话是 `Multimode`。
- 频道：指定进行对话的频道（如果适用）。请参阅 将 Amazon Lex V2 机器人与消息收发平台集成。
- 语言：指定机器人的语言。

详细信息下方将给出机器人在对话中所引发的意图。选择转到意图，可打开意图编辑器并跳转该意图。选择转到转录可自动滚动转录以跳转到机器人引发该意图的第一个实例。

选择意图名称旁边的右箭头可查看有关针对该意图所引发的槽位的详细信息，包括槽位名称、机器人为每个槽位引发的值以及机器人尝试引发每个槽位的次数。

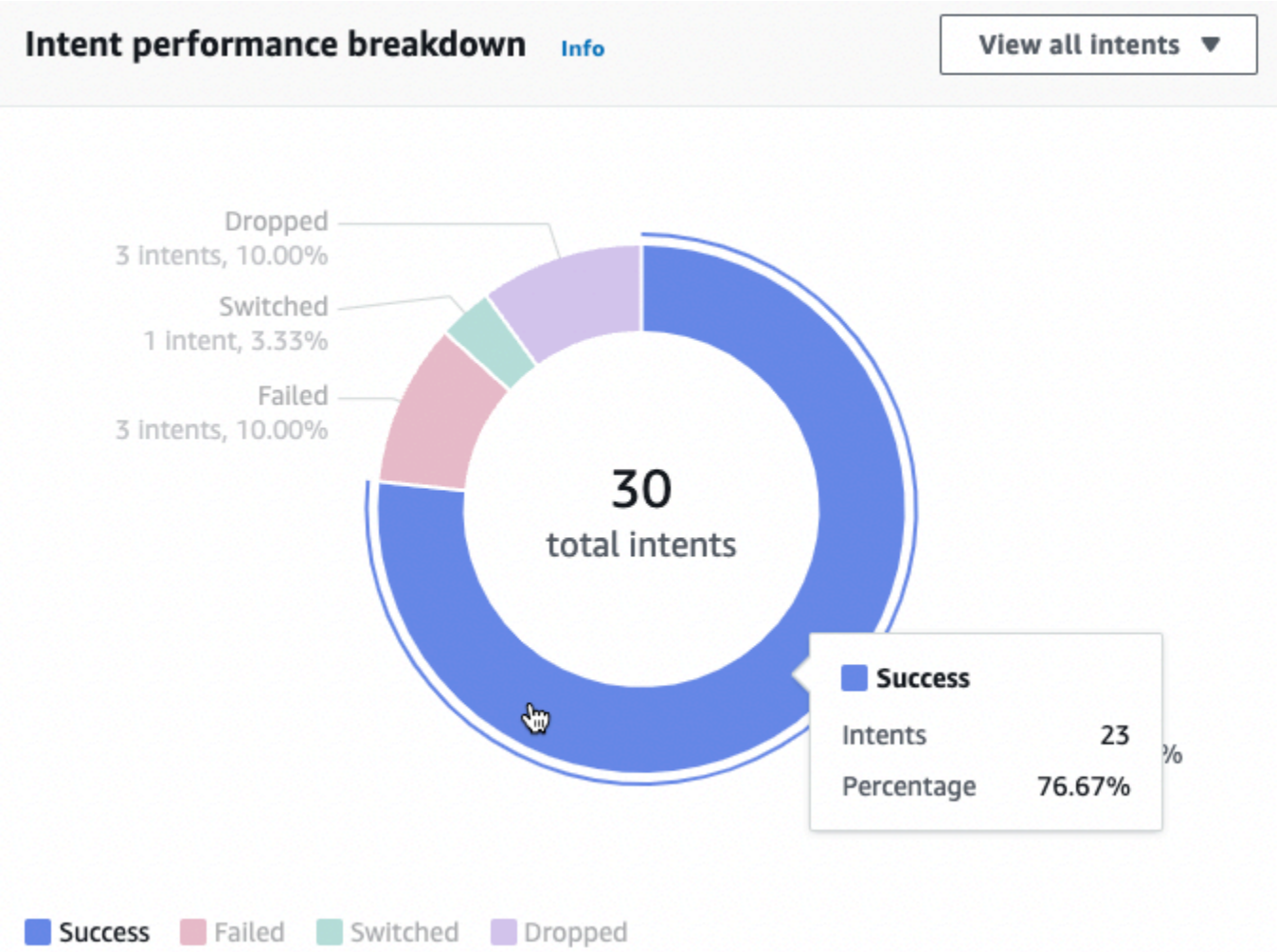
您可在转录部分中查看对话言语以及机器人在引发意图和槽位时的行为。左侧显示的是用户的言语，右侧显示的是机器人的言语。使用标有筛选此会话中的转录的搜索栏可在转录中查找相应文本。在显示：旁边，每个对话回合下方均会显示以下三种信息（可选择是否显示）：

- 时间戳：指定言语的时间。
- 意图状态：指定机器人在言语期间引发的意图以及意图的结果（如果适用）。可能的意图状态如下：
  - 已调用的意图：*intent name*，即机器人已识别出客户正在调用的意图。
  - 已切换意图：*intent name*，即机器人已基于言语切换到不同的意图。
  - *intent name*：成功，即机器人已履行该意图。
- 槽位状态：指定机器人在言语期间引发的槽位（如果适用）以及客户提供的值。

性能控制面板：机器人意图和言语指标概览

您可以在性能控制面板中查看机器人在意图履行和言语识别方面的性能的详细信息。

意图性能细分部分中展示机器人调用意图的总次数，并且按照成功、失败、已删除和已切换的分类来分别展示各个类别的意图的次数和百分比。有关这些定义的解释，请参阅意图。将鼠标悬停在图表中的某一个部分上可展开一个方框，以查看该结果的对话的数量和百分比，如下图所示。



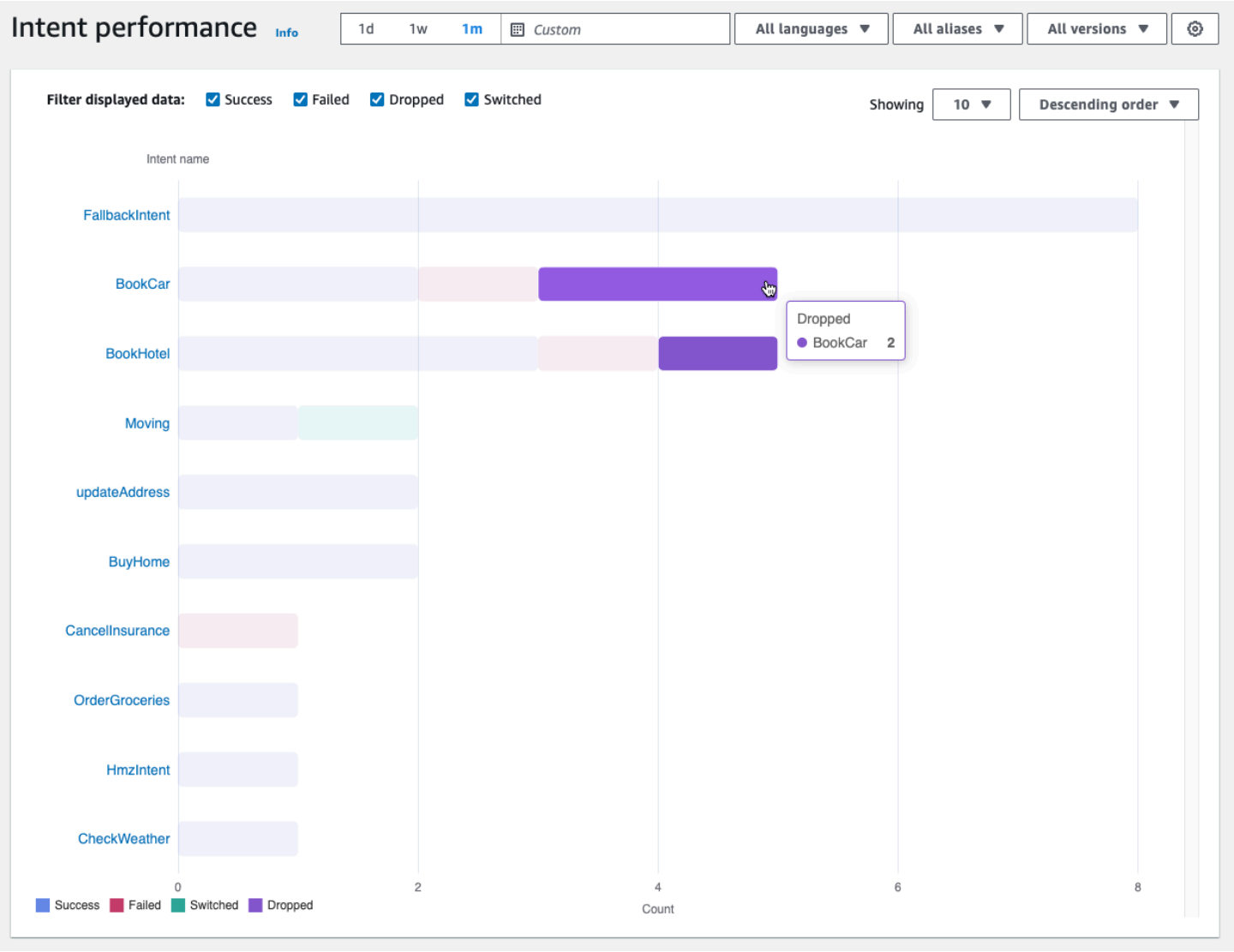
选择查看所有意图可显示一个下拉菜单，以便从中选择查看机器人所引发的各个意图。您也可以选择查看特定结果（成功、失败、已删除或已切换）的意图。点击这些链接可跳转到性能控制面板的意图性能子部分。有关更多信息，请参阅 意图性能。

话语识别部分汇总了遗漏和检测到的言语的数量。选择查看详细信息以导航到该机器人的言语列表。选择遗漏的言语下的数字可查看遗漏言语的列表，并且选择检测到的言语下的数字可查看机器人所检测到的言语的列表。有关更多信息，请参阅 言语识别。

在左侧边栏中选择性能控制面板下的意图性能和言语识别，可查看关于机器人中的意图和言语的详细信息。

意图性能

此控制面板按频率降序汇总了机器人使用的意图的性能。每个意图旁边的条形用以直观地展示该意图被归类为成功、失败、已删除和已切换的次数。有关这些定义的解释，请参阅意图。将鼠标悬停在栏的某个条形上，即可查看使用该意图并获得该结果的对话数量，如下图所示：



> **Note**
>
> 该控制面板中展示一组筛选条件设置的前 1,000 个结果。要获得更有针对性的结果，请配置更精细的筛选条件设置。

您可以在图表顶部选中成功、失败、已删除或已切换复选框，以设置要查看的意图状态。

使用显示右侧的下拉菜单可调整要显示的意图数量以及这些意图的排序，即按频率的升序还是降序显示。

选择某个意图名称可导航至该意图的页面，其中显示三个图表：意图性能明细、槽位性能和意图切换。

意图性能明细部分显示机器人使用该意图的总次数，并且按照成功、失败、已删除和已切换来分类展示各个类别的意图履行次数和百分比。有关这些定义的解释，请参阅意图。将鼠标悬停在图表的某一部分上，即可查看意图履行产生该结果的次数和百分比。

槽位性能部分展示属于当前意图的槽位的指标。要按某一列排序，可选中该列以便按升序排序，再次选中该列则按降序排序。您可以使用搜索栏查找特定的槽位，也可以使用页码按钮来导航至不同的槽位。

> **Note**
>
> 该控制面板中展示一组筛选条件设置的前 1,000 个结果。要获得更有针对性的结果，请配置更精细的筛选条件设置。

意图切换部分列出机器人从当前意图切换到另一个意图的实例，其中包含以下信息：

- 阶段：机器人切换该意图的对话阶段。
- 意图切换到：机器人对当前意图执行切换的目标意图。
- 会话计数：阶段和意图切换到组合的会话数。

> **Note**
>
> 该控制面板中展示一组筛选条件设置的前 1,000 个结果。要获得更有针对性的结果，请配置更精细的筛选条件设置。

言语识别

该页面列出机器人遗漏和检测到的所有言语，同时还提供了相应工具，以便向意图中添加示例言语以帮助训练机器人。有关这些定义的解释，请参阅言语。要切换遗漏的言语与检测到的言语，可点击顶部的选项卡。

> **Note**
>
> 该控制面板中展示一组筛选条件设置的前 1,000 个结果。要获得更有针对性的结果，请配置更精细的筛选条件设置。

要在意图中添加言语，请执行以下操作：

1. 找到要为该意图添加的示例言语，选中该言语旁边的复选框。
2. 选择添加到意图，然后打开意图下的下拉菜单以选择要向其添加言语的意图。
3. 选择添加。

使用 API 进行分析

本节介绍用于检索机器人分析的 API 操作。

> **Note**
>
> 要使用ListUtterance指标和 ListUtteranceAnalyticsData，您的 IAM 角色必须具有执行话语操作的权限，该操作ListAggregated提供对话语相关分析的访问权限。有关详细信息以及适用于 IAM 角色的 IAM 策略，请参阅查看言语统计数据。

- 以下 API 操作可检索机器人的摘要指标：
  - ListSession指标
  - ListIntent指标
  - ListIntentStageMetrics
  - ListUtterance指标
- 以下 API 操作可检索会话和言语的元数据列表：
  - ListSessionAnalyticsData
  - ListUtteranceAnalyticsData
- P ListIntentat hs 操作检索有关客户在与机器人对话时所采用的意图顺序的指标。

筛选结果

您需要针对 Analytics API 请求来指定 `startTime` 和 `endTime`。API 将返回在 `startTime` 之后开始并在 `endTime` 之前结束的会话、意图、意图阶段或言语。

`filters` 是 Analytics API 请求中的可选字段。它映射到AnalyticsSession过滤器、过滤器或AnalyticsIntentAnalyticsUtterance过滤器对象的列表。AnalyticsIntentStageFilter在每个对象中，使用字段创建表达式以作为筛选条件。例如，如果您将以下筛选条件添加到列表中，则机器人将搜索时长超过 30 秒的对话。

```
{
```

```
    "name": "Duration",
    "operator": "GT",
    "value": "30 sec",
}
```

检索机器人的指标

使用 ListSessionMetrics、ListIntentMetrics、ListIntentStageMetrics 和 ListUtteranceMetrics 操作来检索会话、意图、意图阶段和言语的摘要指标。

对于这些操作，请填写以下必填字段：

- 提供 startTime 和 endTime 以定义要检索结果的时间范围。
- 在指标metrics、指标或AnalyticsSession指标对象列表中指定要计算的AnalyticsUtterance指标。AnalyticsIntent AnalyticsIntentStageMetric在每个对象中，使用 name 字段指定要计算的指标，使用 statistic 字段指定计算的是 Sum、Average 或 Max 数，并且使用 order 字段指定结果的排序方式是 Ascending 还是 Descending。

  > **Note**
  >
  > metrics 和 binBy 对象均包含 order 字段。您只能在这两个对象中的一个对象中指定排序 order。

该请求中的其余字段为可选字段。您可以通过以下方式筛选和整理结果：

- 筛选结果：使用 filters 字段来筛选结果。有关更多信息，请参阅筛选结果。
- 按类别对结果进行分组-指定groupBy字段、包含单个 “AnalyticsSession结果”、“AnalyticsIntent结果” 或 “AnalyticsUtterance结果” 对象的列表。AnalyticsIntentStageResult在对象中，指定要以其为基础对结果进行分组的类别的 name 字段。

  如果您在请求中指定groupBy字段，则响应中的results对象将包含AnalyticsSessionGroupBy密钥groupByKeys、密钥或AnalyticsIntentGroupBy密AnalyticsUtteranceGroupBy钥对象的列表，每个对象都name包含您在请求中指定的对象，value字段中包含该类别的成员。AnalyticsIntentStageGroupByKey
- 按时间对结果进行分箱-指定binBy字段，即包含单个AnalyticsBinBySpecification对象的列表。在对象中，将 name 字段指定为 ConversationStartTime 以便按对话开始的时间对结果进行分箱，或者指定为 UtteranceTimestamp 以便按言语发生的时间对结果进行分箱。在 interval

字段中指定要对结果进行分箱的时间间隔，并且在 order 字段中指定排序是 Ascending 还是 Descending。

如果您在请求中指定一个binBy字段，则响应中的results对象将包含binKeys一个AnalyticsBin关键对象列表，每个对象都name包含您在请求中指定的关键对象以及value字段中定义该数据桶的时间间隔。

> **Note**
>
> metrics 和 binBy 对象均包含 order 字段。您只能在这两个对象中的一个对象中指定排序 order。

使用以下字段来处理响应的显示：

- 在 maxResults 字段中指定一个介于 1 到 1,000 之间的数字，以限制单个响应中返回的结果数。
- 如果结果数大于您在 maxResults 字段中指定的数字，则响应中包含 nextToken。再次发出请求，但使用 nextToken 字段中的值以便返回下一批结果。

如果使用 ListUtteranceMetrics，则可以在 attributes 字段中指定要返回的属性。此字段映射到包含单个AnalyticsUtterance属性对象的列表。在 name 字段中指定 LastUsedIntent，以返回 Amazon Lex V2 在言语时使用的意图。

在响应中，该results字段映射到结果对象AnalyticsIntentStageResult、AnalyticsSessionAnalyticsIntent结果对象或AnalyticsUtterance结果对象的列表。每个对象均包含 metrics 字段，以返回您所请求的指标的汇总统计数据以及使用您指定的方法创建的任何数据分箱或组。

在机器人中检索会话和言语的元数据

使用ListSessionAnalyticsData和ListUtteranceAnalyticsData操作检索有关各个会话和话语的元数据。

填写必填的 startTime 和 endTime 字段以定义要检索结果的时间范围。

该请求中的其余字段为可选字段。要对结果进行筛选和排序，请执行以下操作：

- 筛选结果：使用 filters 字段来筛选结果。有关更多信息，请参阅筛选结果。

- 对@@ 结果进行排序-使用包含SessionDataSortBy或UtteranceDataSortBy对象的sortBy字段对结果进行排序。在字段 name 中指定要作为排序依据的值，并且在 order 字段中指定以 Ascending 还是 Descending 进行排序。

使用以下字段来处理响应的显示：

- 在 maxResults 字段中指定一个介于 1 到 1,000 之间的数字，以限制单个响应中返回的结果数。
- 如果结果数大于您在 maxResults 字段中指定的数字，则响应中包含 nextToken。再次发出请求，但使用 nextToken 字段中的值以便返回下一批结果。

在响应中，sessions或utterances字段映射到SessionSpecification或UtteranceSpecification对象的列表。每个对象均包含单个会话或言语的元数据。

在机器人中检索会话和言语的元数据

使用 P ListIntentat hs 操作检索有关客户在与机器人对话时所采用的意图顺序的指标。

对于这些操作，请填写以下必填字段：

- 提供 startTime 和 endTime 以定义要检索结果的时间范围。
- 提供 intentPath 以定义检索指标的意图顺序。路径中的意图用正斜杠隔开。例如，将 intentPath 字段填充为 **/BookCar/BookHotel**，以查看关于用户以该顺序调用 BookCar 和 BookHotel 意图的次数的详细信息。

使用可选的 filters 字段来筛选结果。有关更多详细信息，请参阅筛选结果。

查看言语统计数据

您可以使用言语统计数据来确定您的用户向机器人发送的言语。系统将统计 Amazon Lex V2 成功检测到的言语以及未成功检测到的言语。您可以使用此信息来优化机器人。

例如，如果您发现用户正在发送 Amazon Lex V2 遗漏的言语，则可以将该言语添加到意图中。系统将使用新的言语来更新该意图的 Draft 版本，以便您在将其部署到机器人之前对其进行测试。

当 Amazon Lex V2 将某个言语识别为试图调用为机器人配置的意图时，该言语即被检测到。当 Amazon Lex V2 未能识别某个言语而是调用 AMAZON.FallbackIntent 时，将错过该言语。

可以使用 ListUtteranceMetrics API 和 ListAggregatedUtterance API 来查看言语统计信息。

以下条件下，系统不会使用 ListUtteranceMetrics API 生成言语统计数据：

- 当使用控制台创建机器人时，“儿童在线隐私保护法案”设置被设置为是，或者在使用 CreateBot 操作创建机器人时，childDirected 字段被设置为 true。

该 ListUtteranceMetrics API 还提供其他功能，包括：

- 更多可用信息，例如检测到言语的被映射意图。
- 更多筛选功能（包括频道和模式）。
- 更长的保留日期范围（30 天）。
- 即使您已选择退出数据存储，也可以使用该 API。控制台依赖于 ListUtteranceMetrics API 来实现对遗漏和检测到的言语的功能。

以下条件下，系统不会使用 ListAggregatedUtterance API 生成言语统计数据：

- 当使用控制台创建机器人时，“儿童在线隐私保护法案”设置被设置为是，或者在使用 CreateBot 操作创建机器人时，childDirected 字段被设置为 true。
- 您正在对一个或多个槽位使用槽位模糊处理。
- 您已选择退出 Amazon Lex 改进计划。

该 ListAggregatedUtterance API 所提供的功能包括：

- 可用的详细信息较少（不存在言语的被映射意图）。
- 有限的筛选功能（不包括频道和模式）。
- 较短的保留日期范围（15 天）。

您可以使用言语统计信息来查看是否检测到或遗漏了特定的话语，以及机器人交互中上次使用该言语的时间。

当用户与您的机器人交互时，Amazon Lex V2 会持续不断地存储言语。您可以使用控制台或 ListAggregatedUtterances 操作来查询统计数据。数据保留期为 15 天。如果用户选择退出数据存储，则不可用。您可以使用 DeleteUtterances 操作或选择退出数据存储来删除言语。如果您关闭帐户，则所有话语都将被删除。 AWS 所存储的言语使用服务器管理的密钥进行加密。

如果使用 ListUtteranceMetrics 删除机器人版本，则该版本的言语统计数据最多可用 30 天，而如果使用 ListAggregatedUtterances 删除机器人版本，则该版本的言语统计数据最多可用 15 天。您无法在 Amazon Lex V2 控制台中查看已删除版本的统计数据。要查看已删除版本的统计数据，可以使用 ListAggregatedUtterances 和 ListUtteranceMetrics 操作。

对于 ListAggregatedUtterances 和 ListUtteranceMetrics API，按言语的文本对言语进行汇总。例如，客户发出“我想订购披萨”的话语的所有实例均会在回复中汇总到同一行中。使用该RecognizeUtterance操作时，使用的文本是输入脚本。

要使用 ListAggregatedUtterances 和 ListUtteranceMetrics API，请将以下策略应用于角色。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ListAggregatedUtterancesPolicy",
            "Effect": "Allow",
            "Action": "lex:ListAggregatedUtterances",
            "Resource": "*"
        }
    ]
}
```

管理分析的访问权限

要为用户提供分析访问权限，请将允许其调用 API 操作进行分析的策略附加到该用户的 IAM 角色中。您可以将 AWS 托管策略： AmazonLexFullAccess 附加到 IAM 角色以提供对 Amazon Lex API 操作的完全访问权限，也可以创建仅授予分析权限的自定义策略并将其附加到 IAM 角色中。

要创建包含分析权限的自定义策略，请执行以下操作：

1. 如果您需要首先创建 IAM 角色，请执行创建角色以向 IAM 用户委派权限中所述的步骤。
2. 按照创建 IAM 策略中的步骤来使用以下 JSON 对象创建策略。要为 IAM 角色启用对特定机器人的分析访问权限，请将每个机器人的 ARN 添加到 Resource 字段。将 *region*、*account-id* 和 *BOTID* 替换为与机器人对应的值。您也可以用您选择的名称替换语句标识符。*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. 按照添加和删除 IAM 身份权限中的步骤，将您创建的策略附加到要授予分析权限的角色。
4. 现在，该角色即有权查看您指定的机器人的分析数据。

启用对话日志

使用对话日志来存储用户与机器人的对话。查看这些日志，找出机器人与用户交互中的问题，并利用这些洞察来修改机器人的行为。本节还介绍如何对槽位值进行模糊处理以保护用户的隐私。

主题

- 使用对话日志进行日志记录
- 掩盖对话日志中的槽位值
- 选择性对话日志捕获

使用对话日志进行日志记录

您可以启用对话日志 来存储自动程序的交互。您可以使用这些日志查看自动程序的性能，并解决与对话相关的问题。您可以记录RecognizeText操作的文本。您可以记录RecognizeUtterance操作的文本和音频。您可以启用对话日志，以查看用户与机器人所进行对话的详细信息。

例如，与您的自动程序开展的会话具有会话 ID。您可以使用此 ID 获取对话的记录，包括用户话语和相应的自动程序响应。您还可以获取元数据，例如话语的意图名称和槽位值。

> **Note**
>
> 您不能将对话日志用于受儿童在线隐私保护法案 (COPPA) 约束的自动程序。

对话日志是为别名配置的。每个别名的文本日志和音频日志都可以有不同的设置。您可以为每个别名启用文本日志和/或音频日志。文本日志在 CloudWatch 日志中存储文本输入、音频输入脚本和相关元数据。音频日志将音频输入存储在 Amazon S3 中。您可以使用 AWS KMS 客户托管的 CMK，为文本日志和音频日志启用加密。

要配置日志记录，请使用控制台或CreateBot别名或UpdateBot别名操作。为别名启用对话日志后，对该别名使用RecognizeText或RecognizeUtterance操作将文本或音频语句记录在已配置的 CloudWatch 日志日志组或 S3 存储桶中。

主题

用于对话日志的 IAM 策略

根据您选择的日志类型，Amazon Lex V2 需要权限才能使用亚马逊 CloudWatch 日志和亚马逊简单存储服务 (S3) 存储桶来存储您的日志。您必须创建 AWS Identity and Access Management 角色和权限才能让 Amazon Lex V2 访问这些资源。

为对话日志创建 IAM 角色和策略

要启用对话日志，您必须授予 CloudWatch 日志和 Amazon S3 的写入权限。如果您为 S3 对象启用对象加密，则需要向用于加密对象的 AWS KMS 密钥授予访问权限。

您可以使用 IAM 控制台、IAM API 或 AWS Command Line Interface 来创建角色和策略。这些说明 AWS CLI 使用创建角色和策略。

> **Note**
>
> 以下代码针对 Linux 和 macOS 编排了格式。对于 Windows，将 Linux 行继续符 (\) 替换为脱字号 (^)。

为对话日志创建 IAM 角色

1. 在名为 **LexConversationLogsAssumeRolePolicyDocument.json** 的当前目录中创建一个文档，向其中添加以下代码并保存。此策略文档将 Amazon Lex V2 作为受信任实体添加到角色中。这样，Amazon Lex 可代入将日志传送到为对话日志配置的资源的角色。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "lexv2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

2. 在中 AWS CLI，运行以下命令为对话日志创建 IAM 角色。

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

接下来，创建策略并将其附加到该角色以允许 Amazon Lex V2 写入 CloudWatch 日志。

创建用于将对话文本记录到 Log CloudWatch s 的 IAM 策略

1. 在名为 **LexConversationLogsCloudWatchLogsPolicy.json** 的当前目录中创建一个文档，向其中添加 IAM 策略并保存。

```
{
  "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，创建向 CloudWatch 日志组授予写入权限的 IAM 策略。

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

3. 将该策略附加到您为对话日志创建的 IAM 角色中。

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

如果要将音频日志记录到 S3 存储桶，请创建允许 Amazon Lex V2 写入存储桶的策略。

要创建用于将音频日志记录到 S3 存储桶中的 IAM 策略，请执行以下操作：

1. 在名为 **LexConversationLogsS3Policy.json** 的当前目录中创建一个文档，向其中添加以下策略并保存。

```
{
  "Version": "2012-10-17",
  "Statement": [
      {
          "Effect": "Allow",
          "Action": [
              "s3:PutObject"
```

```
            ],
            "Resource": "arn:aws:s3:::bucket-name/*"
        }
    ]
}
```

2. 在中 AWS CLI，创建授予您的 S3 存储桶写入权限的 IAM 策略。

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

3. 将该策略附加到您为对话日志创建的角色。

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

授予传递 IAM 角色的权限

当您使用控制台 AWS Command Line Interface、或 AWS 软件开发工具包指定用于对话日志的 IAM 角色时，指定对话日志 IAM 角色的用户必须有权将该角色传递给 Amazon Lex V2。要允许用户将角色传递给 Amazon Lex V2，您必须向该用户的 IAM 用户、角色或组授予 `PassRole` 权限。

以下策略定义要授予用户、角色或组的权限。您可以使用 `iam:AssociatedResourceArn` 和 `iam:PassedToService` 条件键来限制权限的范围。有关更多信息，请参阅用户指南中的授予用户将角色传递给 AWS 服务的权限以及 IAM 和 AWS STS 条件上下文密钥。AWS Identity and Access Management

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

配置对话日志

您可以使用控制台或 CreateBotAlias 和 UpdateBotAlias 操作的 conversationLogSettings 字段来启用和禁用对话日志。您可以启用或禁用音频日志和/或文本日志。日志记录将在新自动程序会话上启动。对日志设置的更改不会体现在活动会话中。

要存储文本日志，请在您的 AWS 账户中使用一个 Amazon Log CloudWatch s 日志组。您可以使用任何有效的日志组。日志组必须与 Amazon Lex V2 机器人位于同一区域中。有关创建 CloudWatch 日志组的更多信息，请参阅 Amazon Logs 用户指南中的使用日志组和 CloudWatch 日志流。

要存储音频日志，请在您的 AWS 账户中使用 Amazon S3 存储桶。您可以使用任何有效的 S3 存储桶。该存储桶必须与 Amazon Lex V2 机器人位于同一区域。有关创建 S3 存储桶的更多信息，请参阅《Amazon Simple Storage Service 入门指南》中的创建存储桶。

当您使用控制台管理对话日志时，控制台会更新您的服务角色，以便拥有访问日志组和 S3 存储桶的权限。

如果您不使用控制台，则必须为 IAM 角色提供策略，以便 Amazon Lex V2 拥有写入所配置的日志组或存储桶的权限。如果您使用创建服务相关角色 AWS Command Line Interface，则必须使用custom-suffix选项为该角色添加自定义后缀，如下例所示。有关更多信息，请参阅 为对话日志创建 IAM 角色和策略。

```
aws iam create-service-linked-role \
    --aws-service-name lexv2.amazon.aws.com \
    --custom-suffix suffix
```

您用于启用对话日志的 IAM 角色必须具有 iam:PassRole 权限。应将以下策略附加到角色：

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

```
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::account:role/role"
        }
    ]
}
```

启用对话日志

使用控制台启用日志

1. 访问 https://console.aws.amazon.com/lexv2 以打开 Amazon Lex V2 控制台。
2. 从列表中，选择一个机器人。
3. 在左侧菜单中选择别名。
4. 在别名列表中，选择要为其配置对话日志的别名。
5. 在对话日志部分，选择管理对话日志。
6. 对于文本日志，请选择启用，然后输入 Amazon Log CloudWatch s 日志组名称。
7. 对于音频日志，请选择启用，然后输入 S3 存储桶信息。
8. 可选。要加密音频日志，请选择用于加密的密 AWS KMS 钥。
9. 选择 Save (保存) 以开始记录对话。如有必要，Amazon Lex V2 将更新您的服务角色，使其具有访问 CloudWatch 日志日志组和所选 S3 存储桶的权限。

禁用对话日志

使用控制台禁用日志

1. 访问 https://console.aws.amazon.com/lexv2 以打开 Amazon Lex V2 控制台。
2. 从列表中，选择一个机器人。
3. 在左侧菜单中选择别名。
4. 在别名列表中，选择要为其配置对话日志的别名。
5. 在对话日志部分，选择管理对话日志。
6. 禁用文本日志记录、音频日志记录或这两者，以关闭日志记录。
7. 选择 Save (保存) 以停止记录对话。

在 Amazon 日志中查看文本 CloudWatch 日志

Amazon Lex V2 将您的对话文本日志存储在亚马逊 CloudWatch 日志中。要查看日志，请使用 CloudWatch 日志控制台或 API。有关更多信息，请参阅 Amazon Logs 用户指南中的使用筛选模式搜索CloudWatch 日志数据和 CloudWatch 日志见解查询语法。

要使用 Amazon Lex V2 控制台查看日志，请执行以下操作：

1. 访问 https://console.aws.amazon.com/lexv2 以打开 Amazon Lex V2 控制台。
2. 从列表中，选择一个机器人。
3. 从左侧菜单中选择 “分析”，然后选择 “CloudWatch 指标”。
4. 在指标页面上查看您的机器人的CloudWatch 指标。

您也可以使用 CloudWatch 控制台或 API 来查看日志条目。要查找日志条目，请导航到为该别名配置的日志组。您可以在 Amazon Lex V2 控制台中或使用DescribeBot别名操作找到日志的日志流前缀。

用户言语的日志条目位于多个日志流中。对话中的一个话语在一个日志流中具有一个带指定前缀的条目。日志流中的条目包含以下信息：

message-version

消息架构版本。

自动程序

有关客户正在与之交互的机器人的详细信息。

消息

机器人发回给用户的响应。

utteranceContext

有关处理该言语的信息。

- `runtimeHints`：用于转录和解释用户输入的运行时上下文。有关更多信息，请参阅 使用运行时提示改善对插槽值的识别。
- `slotElicitationStyle`：用于解释用户输入的槽位引发样式。有关更多信息，请参阅 使用拼写样式捕获槽位值。

sessionState

用户与机器人之间对话的当前状态。有关更多信息，请参阅 管理对话。

interpretations

Amazon Lex V2 确定可以满足用户言语的意图列表。使用置信度分数。

interpretationSource

表示槽位是由 Amazon Lex 还是由 Amazon Bedrock 解析。值：Lex | Bedrock

sessionId

正在进行对话的用户会话的标识符。

inputTranscript

用户输入的转录。

- 对于文本输入，表示用户键入的文本。对于 DTMF 输入，表示用户输入的密钥。
- 对于语音输入，这是 Amazon Lex V2 将用户言语转换成的文本，以便调用意图或填补槽位。

未加工的 InputTranscript

应用任何文本处理之前的用户输入原始转录。注意：文本处理仅适用于 en-US 和 en-GB 区域设置。

transcriptions

用户输入的潜在转录列表。有关更多信息，请参阅 使用语音转录置信度分数。

rawTranscription

使用语音转录置信度分数。有关更多信息，请参阅 使用语音转录置信度分数。

missedUtterance

指示 Amazon Lex V2 是否能够识别用户的言语。

requestId

Amazon Lex V2 为用户输入生成的请求 ID。

时间戳

用户输入的时间戳。

developerOverride

指示是否使用对话代码挂钩更新了对话流程。有关使用对话代码挂钩的更多信息，请参阅使用 AWS Lambda 函数启用自定义逻辑。

inputMode

指示输入的类型。可以是音频、DTMF 或文本。

requestAttributes

处理用户输入时使用的请求属性。

audioProperties

如果启用了音频对话日志，并且用户输入是音频格式，则包括音频输入的总时长、语音持续时间以及音频中的静默持续时间。它还包括一个指向音频文件的链接。

bargeIn

指示用户输入是否中断了之前的机器人响应。

responseReason

生成响应的原因。可以是以下值之一：

- `UtteranceResponse`：对用户输入的响应
- `StartTimeout`：当用户未提供输入时，服务器生成的响应
- `StillWaitingResponse`：当用户请求机器人等待时，服务器生成的响应
- `FulfillmentInitiated`：服务器生成的指示即将触发履行的响应
- `FulfillmentStartedResponse`：服务器生成的指示履行已开始的响应
- `FulfillmentUpdateResponse`：在履行过程中，服务器定期生成的响应
- `FulfillmentCompletedResponse`：履行完成后，服务器生成的响应。

operationName

用于与机器人交互的 API。可以为以下值之一：`PutSession`、`RecognizeText`、`RecognizeUtterance` 或 `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"
}
```

日志条目的内容取决于事务的结果以及机器人和请求的配置。

- 如果 missedUtterance 字段为 true，则 intent、slots 和 slotToElicit 字段不会显示在条目中。
- 如果音频日志已禁用或者 inputDialogMode 字段为 Text，则不显示 s3PathForAudio 字段。
- 仅当您为自动程序定义了响应卡时，才会显示 responseCard 字段。
- 仅当您在请求中指定了请求属性时，才会显示 requestAttributes 映射。
- 只有在 AMAZON.KendraSearchIntent 请求搜索 Amazon Kendra 索引时，kendraResponse 字段才会出现。
- 当在机器人的 Lambda 函数中指定了替代意图时，developerOverride 字段的值为 True。
- 仅当在请求中指定了会话属性时，才会显示 sessionAttributes 映射。
- 仅当将自动程序配置为返回情绪值时，才会显示 sentimentResponse 映射。

> **Note**
>
> 输入格式可以更改，但不必对 `messageVersion` 做出相应更改。您的代码不应在有新字段时引发错误。

访问 Amazon S3 中的音频日志

Amazon Lex V2 将对话的音频日志存储在 S3 存储桶中。

您也可以使用 Amazon S3 控制台或 API 来访问音频日志。您可以在 Amazon Lex V2 控制台或 `DescribeBotAlias` 操作响应的 `conversationLogSettings` 字段中查看音频文件的 S3 对象键前缀。

使用 CloudWatch 指标监控对话日志状态

使用 Amazon CloudWatch 监控您的对话日志的交付指标。您可以在指标上设置警报，以便在日志记录发生问题时获取通知。

Amazon Lex V2 在 `AWS/Lex` 命名空间中为对话日志提供了四个指标：

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

成功指标表明 Amazon Lex V2 已成功将音频或文本日志写入其目标。

失败指标表明 Amazon Lex V2 无法将音频或文本日志传输到指定目标。这通常是配置错误。当您的失败指标大于零时，请检查以下内容：

- 确保 Amazon Lex V2 是 IAM 角色的可信实体。
- 对于文本记录，请确保 CloudWatch 日志组存在。对于音频日志记录，请确保 S3 存储桶存在。
- 确保 Amazon Lex V2 用于访问 CloudWatch 日志日志组或 S3 存储桶的 IAM 角色具有日志组或存储桶的写入权限。
- 确保 S3 存储桶与 Amazon Lex V2 机器人位于相同的区域，并且属于您的账户。

掩盖对话日志中的槽位值

Amazon Lex V2 能够模糊处理（即隐藏）槽位的内容，使其内容不可见。为了保护作为槽值捕获的敏感数据，可以启用槽模糊处理，为日志记录掩蔽这些值。

当您选择模糊处理槽位值时，Amazon Lex V2 在对话日志中将槽位的值替换为槽位的名称。对于名为 `full_name` 的槽，该槽的值将被模糊处理，如下所示：

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

如果言语中包含括号字符 ({})，Amazon Lex V2 将用两个反斜杠 (\\) 转义括号字符。例如，文本 `{John Stiles}` 的模糊处理如下所示：

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

对话日志中的槽值会被模糊处理。槽位值在 `RecognizeText` 和 `RecognizeUtterance` 操作的响应中仍然可用，并且槽位值可用于验证和实现 Lambda 函数。如果您在提示或响应中使用槽值，则对话日志中不会对这些槽值进行模糊处理。

在第一轮对话中，如果 Amazon Lex V2 能够识别出言语中的槽位和槽位值，则会对槽位值进行模糊处理。如果没有识别出槽位值，Amazon Lex V2 不对言语进行模糊处理。

在第二轮和接下来的轮次中，Amazon Lex V2 知晓要引发的槽位以及是否需要对槽位值进行模糊处理。如果 Amazon Lex V2 识别到该槽位值，则会对该值进行模糊处理。如果 Amazon Lex V2 未识别出值，则对整个言语进行模糊处理。无法理解的话语中的任何槽值都不会被模糊处理。

Amazon Lex V2 也不会对您存储在请求或会话属性中的槽位值进行模糊处理。如果您将应模糊处理的槽值作为属性存储，则必须加密该值，或者以其他方式对该值进行模糊处理。

Amazon Lex V2 不会对音频中的槽位值进行模糊处理。它会对音频转录中的槽值进行模糊处理。

您可以使用控制台或使用 Amazon Lex V2 API 选择需要对哪些槽位进行模糊处理。在控制台中，在槽的设置中选择 Slot obfuscation (槽模糊处理)。如果您使用的是 API，请在调用CreateSlot或UpdateSlot操作`DEFAULT_OBFUSCATION`时将插槽的`obfuscationSetting`字段设置为。

选择性对话日志捕获

用户可利用选择性对话日志捕获来选择如何使用实时对话中的文本和音频数据捕获对话日志。

要启用和捕获选择性对话日志捕获功能的输出，需要在 Amazon Lex V2 控制台中激活该功能，并在 API 设置中启用必要的会话属性，以捕获日志中的选定输出。

您可以对选择性对话日志捕获选择以下选项：

- 仅文本
- 仅音频
- 文本和音频

您可以捕获对话的特定部分，并选择为对话日志捕获音频、文本，还是这两者均捕获。

> **Note**
>
> 选择性对话日志捕获仅适用于 Amazon Lex V2。

主题

管理选择性对话日志捕获

使用 Lex 控制台，您可以启用选择性对话日志捕获设置，并选择要为哪些槽位启用选择性对话日志捕获功能。

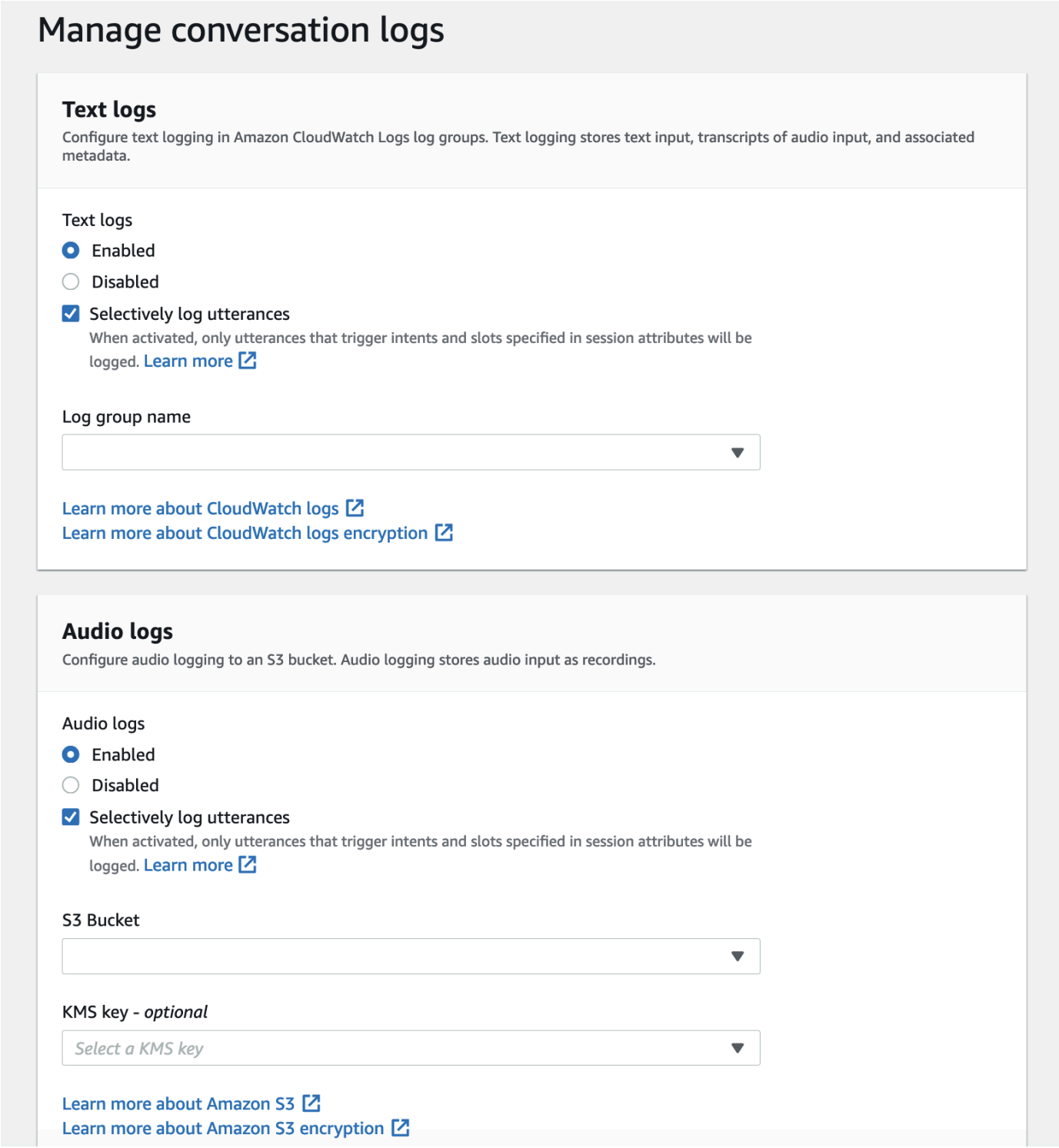
在 Amazon Lex V2 控制台中激活选择性对话日志捕获：

1. 登录 AWS Management Console 并打开 Amazon Lex V2 主机，网址为 https://console.aws.amazon.com/lexv2/home。
2. 从左侧面板中选择机器人，然后选择要启用选择性对话日志捕获的机器人。选择一个现有机器人，或创建一个新的机器人。
3. 在左侧面板的部署部分下为所选机器人选择别名。
4. 选择你的机器人的别名，然后选择管理对话日志。

5. 在管理对话日志面板中，对于文本日志，点击单选按钮以选择是启用还是禁用文本日志。如果选择启用文本日志，则需要输入日志组名称或从下拉菜单中选择现有的日志组名称。如果您选择性地对文本文件进行日志记录，请选中选择性地对言语进行日志记录复选框。

   > **Note**
   >
   > 通过在构建时间设置中的对话日志设置（文本和/或音频）中选中选择性记录话语复选框来启用文本和/或音频日志。BotAlias您必须配置 CloudWatch 日志组和 Amazon S3 存储桶才能选择此选项。

6. 在音频日志部分中，点击单选按钮以选择是启用还是禁用音频日志。如果您选择启用音频日志，则需要指定 Amazon S3 存储桶位置以及（可选）用于加密音频数据的 KMS 密钥。如果您选择性地对音频文件进行日志记录，请选中选择性地对言语进行日志记录复选框。



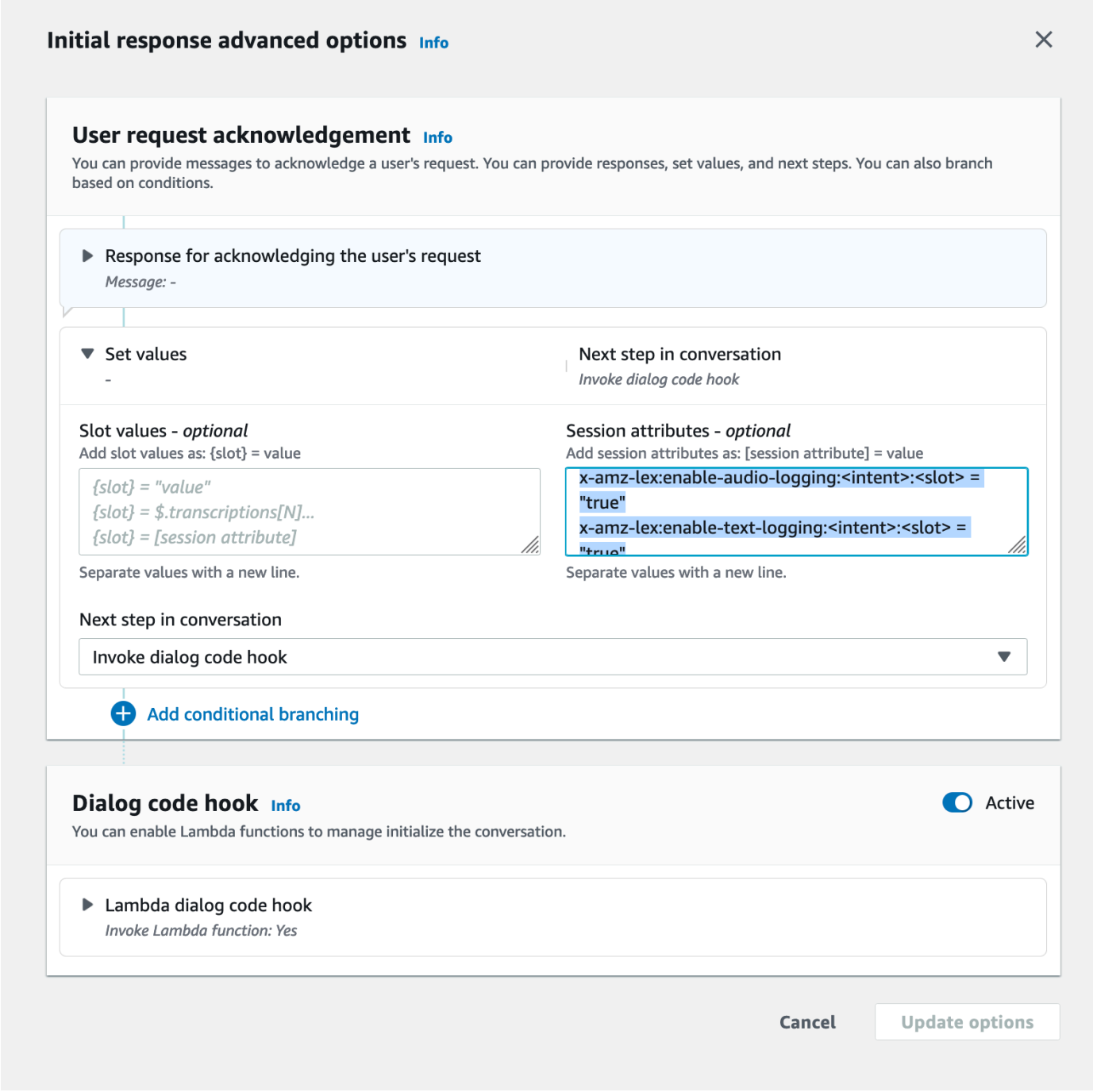
7. 选择面板右下角的保存以保存您的选择性对话日志捕获设置。

在 Lex 控制台中激活选择性对话日志捕获：

1. 转到意图，选择意图名称、初始响应、高级设置、设置值、会话属性。

2. 根据要为其启用选择性对话日志捕获的意图和槽位，完成以下属性设置：

   - `x-amz-lex:enable-audio-logging:intent:slot = "true"`
   - `x-amz-lex:enable-text-logging:intent:slot = "true"`



> **Note**
>
> 设置 `x-amz-lex:enable-audio-logging:intent:slot = "true"` 以捕获对话中仅包含特定槽位的言语。言语的日志记录操作作取决于在与会话属性表达式相比较的同时对言语内 *intent*:*slot* 进行的评估，以及相应的标志值。要对言语进行日志记录，会话属性中必须有至少一个表达式允许该日志记录操作，并且启用日志记录标志需要设置为 `true`。*intent* 和 *slot* 的值也可以是 "*"。如果槽位和/或意图值为 "*"，则表示的任何槽位和/或意图值 "*" 都将与其匹配。与 `x-amz-lex:enable-audio-logging` 类似，名为 `x-amz-lex:enable-text-logging` 的新会话属性将用于控制文本日志。

3. 选择更新选项，然后构建机器人以包含更新的设置。

> **Note**
>
> 您的 IAM 角色必须具有访问权限，才能允许您向 Amazon S3 存储桶写入数据并使用 KMS 密钥加密数据。Lex 将更新您的 IAM 角色，使其具有 Lex 访问 CloudWatch 日志组和选定的 Amazon S3 存储桶的权限。

选择性对话日志捕获的使用指南：

只有已在对话日志设置中启用文本和/或音频日志后，才能为文本和/或音频日志启用选择性对话日志捕获。您可以为文本和/或音频日志启用选择性对话日志捕获，从而禁用对话中所有意图和槽位的日志记录。要为特定意图和槽位生成文本和/或音频日志，必须将这些意图和槽位的文本或/和音频选择性对话日志捕获会话属性设置为 true。

- 如果启用了选择性对话日志捕获，并且不存在带前缀 x-amz-lex:的会话属性，enable-audio-logging 则默认情况下，所有话语的日志记录都将处于禁用状态。这种情况在:启用文本记录方面也是如此。x-amz-lex
- 如果会话属性中至少有一个表达式允许，则将专门存储文本和/或音频对话片段的言语日志。
- 只有已在机器人别名的“对话日志设置”中启用文本和/或音频的选择性对话日志捕获之后，会话属性中定义的文本和/或音频的选择性对话日志捕获配置才会生效；否则，会话属性将被忽略。
- 启用选择性对话日志捕获后，“解释” 和 “转录” 中 SessionState未使用会话属性启用日志记录的任何时隙值都将在生成的文本日志中进行模糊处理。

- 除用户可以提供槽位值以及意图引发的意图引发回合之外，将通过将机器人引发的槽位与选择性对话日志捕获会话属性进行匹配来对生成音频和/或文本日志的决定进行评估。在意图引发回合中，将当前回合中被填充的槽位与选择性对话日志捕获会话属性进行匹配
- 被认为已填充的槽位来自回合结束时的会话状态。因此，Dialog Codehook Lambda 对处于会话状态的槽位所做的任何更改都将影响选择性对话日志捕获的行为。
- 在意图引发回合中，如果用户给出了多个槽位值，则只有当文本/音频会话属性允许对该回合中被填充的所有槽位进行日志记录时，才会生成文本和/或音频日志。
- 推荐的操作方法是在会话开始时设置选择性对话日志捕获会话属性，并且避免在会话期间对其进行修改。
- 如果任何槽位包含敏感数据，则应始终启用槽位模糊处理。

选择性对话日志捕获示例

以下是选择性对话日志捕获的业务使用案例示例。

使用案例：

一家金融科技公司利用 Amazon Lex V2 机器人来为其账单支付系统 IVR 提供支持。为满足合规和审计要求，系统中必须保留用户进行授权同意的录音。但是，启用通用音频日志是不可行的，因为这会使它们不合规，因为无法混淆音频日志中的敏感插槽 CardNumber，例如 CVV 和其他信息。基于此，该公司可以选择为音频日志启用选择性对话日志捕获，并将会话属性设置为仅为包含授权同意的言语生成音频日志。

BotAlias 设置：

- 已启用文本日志：true
- 已启用文本日志选择性记录：false
- 已启用音频日志：true
- 已启用音频日志选择性记录：true

会话属性：

```
x-amz-lex:enable-audio-logging:PayBill:AuthorizationConsent = "true"
```

对话示例：

- 用户（音频输入）：“我想要支付账号 35XU68 的账单。”

- 机器人：“请问您的应付金额是多少美元？”
- 用户（音频输入）：“235。”
- 机器人：“请问您的信用卡号是多少？”
- 用户（音频输入）：“9239829722200348。”
- 机器人：“您即将使用尾号 0348 的信用卡支付 235 美元。请说‘我授权支付 235 美元。’”
- 用户（音频输入）：“我授权支付 235 美元。”
- 机器人：“您的账单已经成功支付。”

对话日志输出：

在这种情况下，将生成所有回合的文本日志。但是，只有在引出PayBill意图中的AuthorizationConsent槽位时，才会录制特定回合的音频日志，而不会为任何其他回合生成音频日志。

监控运营指标

亚马逊 CloudWatch 和两 AWS CloudTrail 项 AWS 服务与 Amazon Lex V2 集成，可帮助您监控用户与机器人的互动。使用这些服务来录制操作、发送近乎实时的数据，并设置在满足条件时的通知和自动操作。

主题

- 使用 Amazon 衡量运营指标 CloudWatch
- 使用查看事件 AWS CloudTrail

使用 Amazon 衡量运营指标 CloudWatch

您可以使用监控 Amazon Lex V2 CloudWatch，它会收集原始数据并将其处理为可读的近乎实时的指标。这些统计数据会保存 15 个月，从而使您能够访问历史信息，并能够更好地了解您的 Web 应用程序或服务的执行情况。此外，可以设置用于监测特定阈值的警报，并在达到相应阈值时发送通知或执行操作。有关更多信息，请参阅 Amazon CloudWatch 用户指南。

Amazon Lex V2 服务在 `AWS/Lex` 命名空间内报告以下指标。

指标	描述
AssistedSlotResolutionModelAccessDeniedErrorCount	Amazon Lex V2 被拒绝访问 Amazon Bedrock 的次数 RecognizeUtterance 和 StartConversation 操作的有效维度： • BotId、 BotAliasId、 LocaleId、操作、 InputMode、 ModelType、模型 • BotId、 BotVersion、 LocaleId、操作、 InputMode、 ModelType、模型 RecognizeText 的有效维度： • BotId、 BotAliasId、 LocaleId、操作、 ModelType、模型 • BotId、 BotVersion、 LocaleId、操作、 ModelType、模型 单位：计数
AssistedSlotResolutionModelInvocationCount	调用 Amazon Bedrock 的次数。 RecognizeUtterance 和 StartConversation 操作的有效维度： • BotId、 BotAliasId、 LocaleId、操作、 InputMode、 ModelType、模型 • BotId、 BotVersion、 LocaleId、操作、 InputMode、 ModelType、模型 RecognizeText 的有效维度： • BotId、 BotAliasId、 LocaleId、操作、 ModelType、模型 • BotId、 BotVersion、 LocaleId、操作、 ModelType、模型 单位：计数
AssistedSlotResolutionModel	调用 Amazon Bedrock 时出现 5xx 的次数。

指标	描述
SystemErrorCount	RecognizeUtterance 和 StartConversation 操作的有效维度： • BotId、 BotAliasId、 LocaleId、操作、 InputMode、 ModelType、模型 • BotId、 BotVersion、 LocaleId、操作、 InputMode、 ModelType、模型 RecognizeText 的有效维度： • BotId、 BotAliasId、 LocaleId、操作、 ModelType、模型 • BotId、 BotVersion、 LocaleId、操作、 ModelType、模型 单位：计数
AssistedSlotResolutionModelThrottlingErrorCount	Amazon Lex 被 Amazon Bedrock 节流的次数。 RecognizeUtterance 和 StartConversation 操作的有效维度： • BotId、 BotAliasId、 LocaleId、操作、 InputMode、 ModelType、模型 • BotId、 BotVersion、 LocaleId、操作、 InputMode、 ModelType、模型 RecognizeText 的有效维度： • BotId、 BotAliasId、 LocaleId、操作、 ModelType、模型 • BotId、 BotVersion、 LocaleId、操作、 ModelType、模型 单位：计数

指标	描述
AssistedSlotResolutionResolvedSlotCount	Amazon Bedrock 返回槽位值的次数。 RecognizeUtterance 和 StartConversation 操作的有效维度： • BotId、 BotAliasId、 LocaleId、操作、 InputMode、 ModelType、模型 • BotId、 BotVersion、 LocaleId、操作、 InputMode、 ModelType、模型 RecognizeText 的有效维度： • BotId、 BotAliasId、 LocaleId、操作、 ModelType、模型 • BotId、 BotVersion、 LocaleId、操作、 ModelType、模型 单位：计数
KendraIndexAccessError	Amazon Lex V2 无法访问您的 Amazon Kendra 索引的次数。 • 操作， BotId， BotAliasId， LocaleId 单位：计数
KendraLatency	Amazon Kendra 对来自 AMAZON.KendraSearchIntent 的请求做出响应所花费的时间。 有效维度： • 操作， BotId， BotVersion， LocaleId • 操作， BotId， BotAliasId， LocaleId 单位：毫秒

指标	描述
KendraSuccess	Amazon Lex V2 无法访问您的 Amazon Kendra 索引的次数。 有效维度： • 操作， BotId， BotVersion， LocaleId • 操作， BotId， BotAliasId， LocaleId 单位：计数
KendraSystemErrors	Amazon Lex V2 无法查询 Amazon Kendra 索引的次数。 有效维度： • 操作、 BotId、 BotAliasId、 InputMode、 LocaleId 单位：计数
KendraThrottledEvents	Amazon Kendra 限制来自 AMAZON.KendraSearchIntent 的请求的次数。 有效维度： • 操作， BotId， BotAliasId， InputMode。 LocaleId 单位：计数

指标	描述
RuntimeConcurrency	指定时间段内的并发连接数。RuntimeConcurrency 将被报告为 StatisticSet 。 RecognizeUtterance 或 StartConversation 操作的有效维度： • 操作、 BotId、 BotVersion、 InputMode、 LocaleId • 操作、 BotId、 BotAliasId、 InputMode、 LocaleId 其他操作的有效维度： • 操作， BotId， BotVersion， LocaleId • 操作， BotId， BotAliasId， LocaleId 单位：计数
RuntimeInvalidLambdaResponses	指定时间段内无效 AWS Lambda 响应的数量。 有效维度： • 操作、 BotId、 BotAliasId、 InputMode、 LocaleId 单位：计数
RuntimeLambdaErrors	指定时间段内的 Lambda 运行时错误数量。 有效维度： • 操作、 BotId、 BotAliasId、 InputMode、 LocaleId 单位：计数

指标	描述
RuntimePollyErrors	在指定时段内的无效 Amazon Polly 响应的数量。 有效维度： • 操作、 BotId、 BotAliasId、 InputMode、 LocaleId 单位：计数
RuntimeRequestCount	指定时间段内的运行时请求数。 RecognizeUtterance 和 StartConversation 操作的有效维度： • 操作、 BotId、 BotVersion、 InputMode、 LocaleId • 操作、 BotId、 BotAliasId、 InputMode、 LocaleId 其他操作的有效维度： • 操作， BotId， BotVersion， LocaleId • 操作， BotId， BotAliasId， LocaleId 单位：计数
RuntimeRequestLength	与 Amazon Lex V2 机器人对话的总时长。仅适用于该StartConversation操作。 有效维度： • BotAlias编号、 BotId、 LocaleId、操作 • BotId、 BotAliasId、 LocaleId、操作 单位：毫秒

指标	描述
RuntimeSuccessfulRequestLatency **Important** 该指标是 RuntimeSuccessfulRequestLatency ，而不是 RuntimeSuccessfulRequestLatency 。	从发送请求到传回响应的时间段内的成功请求的延迟。 RecognizeUtterance 和 StartConversation 操作的有效维度： • 操作、 BotId、 BotVersion、 InputMode、 LocaleId • 操作、 BotId、 BotAliasId、 InputMode、 LocaleId 其他操作的有效维度： • 操作， BotId， BotVersion， LocaleId • 操作， BotId， BotAliasId， LocaleId 单位：毫秒
RuntimeSystemErrors	指定时间段内的系统错误的数量。系统错误的响应代码范围为 500 到 599。 RecognizeUtterance 和 StartConversation 操作的有效维度： • 操作、 BotId、 BotVersion、 InputMode、 LocaleId • 操作、 BotId、 BotAliasId、 InputMode、 LocaleId 其他操作的有效维度： • 操作， BotId， BotVersion， LocaleId • 操作， BotId， BotAliasId， LocaleId 单位：计数

指标	描述
RuntimeThrottledEvents	受限事件的数目。当 Amazon Lex V2 接收的请求数超出为账户设置的每秒事务数限制时，将对事件进行限制。如果经常超出为账户设置的限制，您可以请求提高限制。要请求提高限制，请参阅 AWS 服务限制。 RecognizeUtterance 和 StartConversation 操作的有效维度： • 操作、 BotId、 BotVersion、 InputMode、 LocaleId • 操作、 BotId、 BotAliasId、 InputMode、 LocaleId 其他操作的有效维度： • 操作， BotId， BotVersion， LocaleId • 操作， BotId， BotAliasId， LocaleId 单位：计数
RuntimeUserErrors	指定时间段内的用户错误的数量。用户错误的响应代码范围为 400 到 499。 RecognizeUtterance 和 StartConversation 操作的有效维度： • 操作、 BotId、 BotVersion、 InputMode、 LocaleId • 操作、 BotId、 BotAliasId、 InputMode、 LocaleId 其他操作的有效维度： • 操作， BotId， BotVersion， LocaleId • 操作， BotId， BotAliasId， LocaleId 单位：计数

Amazon Lex V2 指标支持以下维度。

维度	描述
Operation	生成该条目的 Amazon Lex V2 操作的名称，即 RecognizeText 、RecognizeUtterance 、StartConversation 、GetSession 、PutSession 、DeleteSession 。
BotId	机器人的唯一字母数字标识符。
BotAliasId	机器人别名的唯一字母数字标识符。
BotVersion	机器人的数字版本。
InputMode	机器人输入的类型，即语音、文本或 DTMF。
LocaleId	机器人区域设置的标识符，例如 en-US 或 fr-CA。
Model	表示 Amazon Bedrock 大型语言模型的模型 ID。
ModelType	表示从 Amazon Bedrock 调用的大型语言模型的类型。

使用查看事件 AWS CloudTrail

Amazon Lex V2 与 AWS CloudTrail一项服务集成，该服务提供用户、角色或 AWS 服务在 Amazon Lex V2 中采取的操作的记录。 CloudTrail 将 Amazon Lex V2 的 API 调用捕获为事件。被捕获的调用中包括通过 Amazon Lex V2 控制台的调用以及对 Amazon Lex V2 API 操作的代码调用。如果您创建了跟踪，则可以允许将 CloudTrail 事件持续传输到 Amazon S3 存储桶，包括 Amazon Lex V2 的事件。如果您未配置跟踪，您仍然可以在 CloudTrail控制台的 “事件历史记录” 中查看最新的事件。使用收集的信息 CloudTrail，您可以确定向 Amazon Lex V2 发出的请求、发出请求的 IP 地址、谁提出了请求、何时提出请求以及其他详细信息。

要了解更多信息 CloudTrail，请参阅《AWS CloudTrail 用户指南》。

Amazon Lex V2 中的信息 CloudTrail

CloudTrail 在您创建 AWS 账户时已在您的账户上启用。当 Amazon Lex V2 中发生活动时，该活动会与其他 AWS 服务 CloudTrail事件一起记录在事件历史记录中。您可以在自己的 AWS 账户中查看、搜索和下载最近发生的事件。有关更多信息，请参阅使用事件历史记录查看 CloudTrail 事件。

要持续记录您的 AWS 账户中的事件，包括 Amazon Lex V2 的事件，请创建跟踪。跟踪允许 CloudTrail 将日志文件传输到 Amazon S3 存储桶。默认情况下，在控制台中创建跟踪记录时，此跟踪记录应用于所有 AWS 区域。跟踪记录 AWS 分区中所有区域的事件，并将日志文件传送到您指定的 Amazon S3 存储桶。此外，您可以配置其他 AWS 服务，以进一步分析和处理 CloudTrail 日志中收集的事件数据。有关更多信息，请参阅下列内容：

- 创建跟踪记录概述
- CloudTrail 支持的服务和集成
- 为 CloudTrail 配置 Amazon SNS 通知
- 接收来自多个区域的 CloudTrail 日志文件和接收来自多个账户的 CloudTrail 日志文件

Amazon Lex V2 支持记录模型构建 API V2 中列出的所有操作。

每个事件或日记账条目都包含有关生成请求的人员信息。身份信息有助于您确定以下内容：

- 请求是使用根用户证书还是 AWS Identity and Access Management IAM 用户凭证发出。
- 请求是使用角色还是联合用户的临时安全凭证发出的。
- 请求是否由其他 AWS 服务发出。

有关更多信息，请参阅CloudTrail 用户身份元素。

了解 Amazon Lex V2 日志文件条目

跟踪是一种配置，允许将事件作为日志文件传输到您指定的 Amazon S3 存储桶。 CloudTrail 日志文件包含一个或多个日志条目。事件代表来自任何来源的单个请求，包括有关请求的操作、操作的日期和时间、请求参数等的信息。 CloudTrail 日志文件不是公共 API 调用的有序堆栈跟踪，因此它们不会按任何特定的顺序出现。

以下示例显示了一个演示 “CreateBot别名” 操作的 CloudTrail 日志条目。

```
{
    "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": {
```

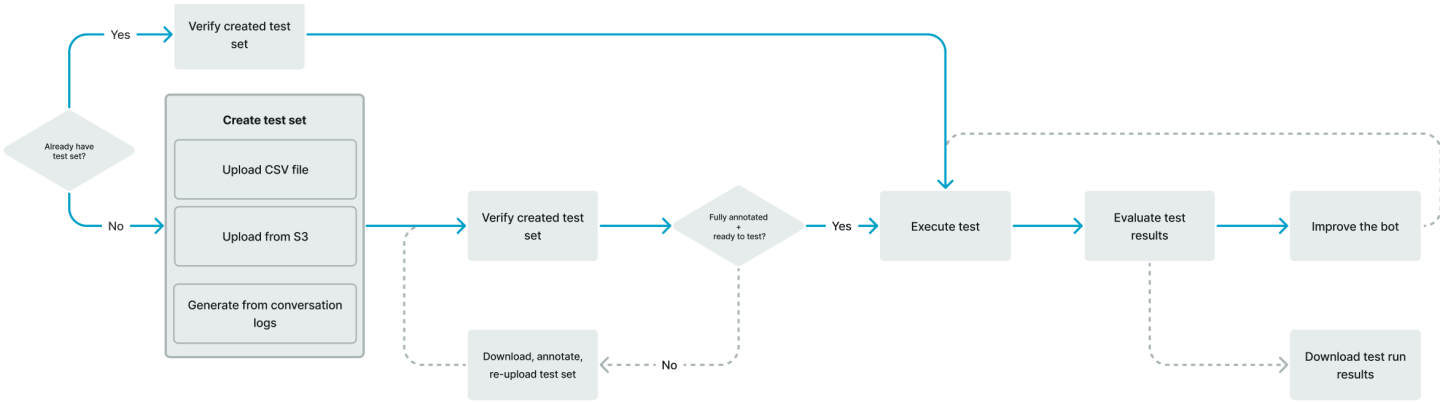
```
            "sessionIssuer": {
                "type": "Role",
                "principalId": "ID of caller",
                "arn": "arn:aws:iam::111122223333:role/role name",
                "accountId": "111122223333",
                "userName": "role name"
            },
            "webIdFederationData": {},
            "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 Test Workbench 评估机器人性能

为提高机器人性能，您可以大规模评估机器人的性能。您的测试评估结果将以简单表格和图表的形式展示。

您可以使用 Test Workbench 创建使用现有转录数据的参考测试集。您可以在部署之前大规模地测试机器人，从而评估性能并查看测试结果细分。



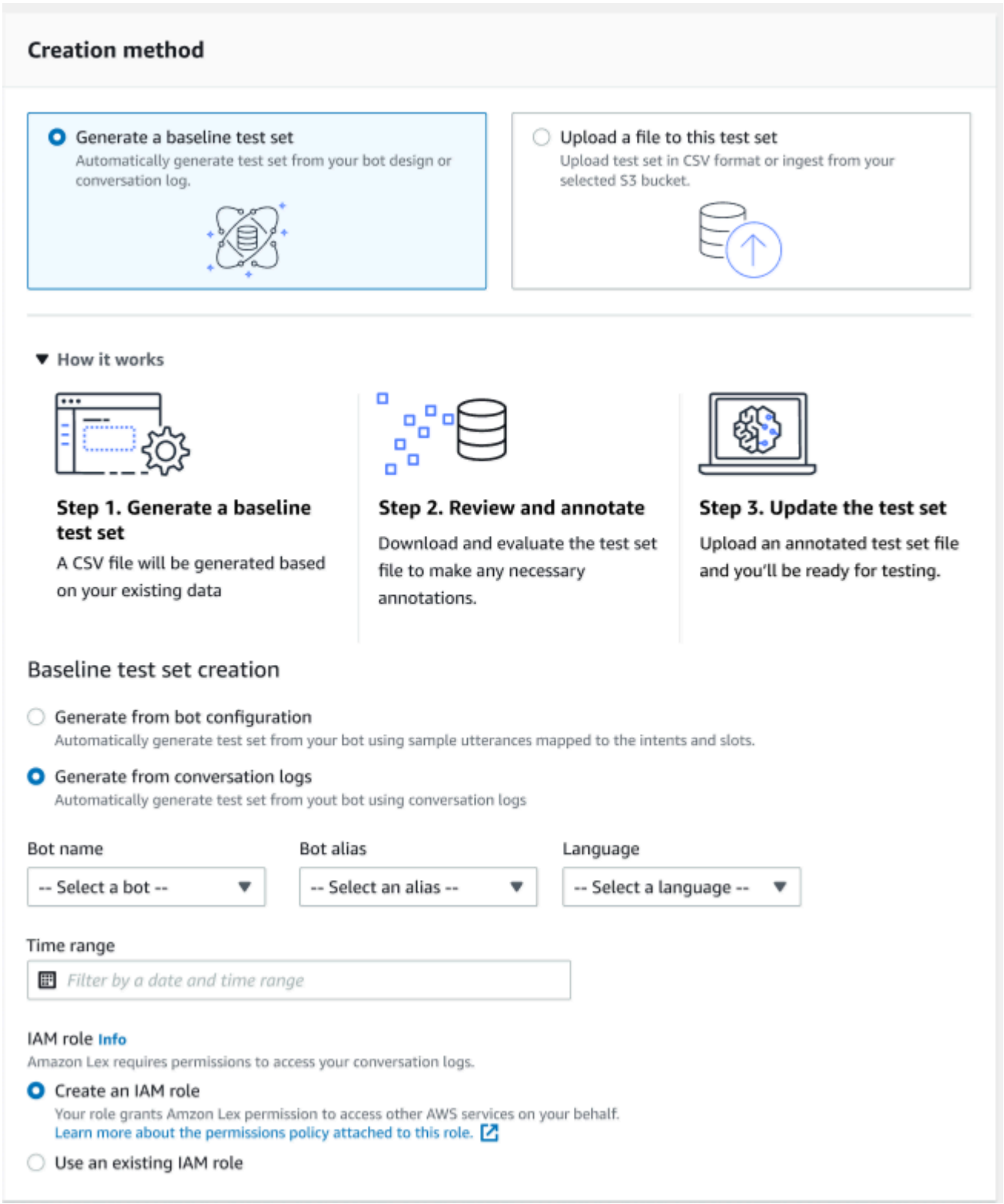
用户可以使用 Test Workbench 来确定机器人的基准性能。该基准性能包括以单一输入或对话形式出现的言语的意图和槽位性能。成功加载测试集后，您可以针对现有的预生产或生产机器人运行该测试集。Test Workbench 可帮助您辨别存在改进空间的槽位填充和意图分类。

主题

- 生成测试集
- 管理测试集
- 执行测试
- 测试集覆盖范围
- 查看测试结果
- 测试结果详细信息

生成测试集

您可以创建测试集来评估机器人的性能。要生成测试机，可上传 CSV 文件格式的测试集或使用对话日志来生成测试集。测试集可以包含音频或文本输入。



如果测试集产生了验证错误，请删除该测试集并将其替换为另一个测试集数据列表，或者使用电子表格编辑程序编辑 CSV 文件中的数据。

要创建测试数据集，请执行以下操作：

1. 登录 AWS Management Console 并打开 Amazon Lex 控制台，网址为 https://console.aws.amazon.com/lex/。
2. 从左侧面板中选择 Test workbench。
3. 从 Test workbench 下的选项中选择测试集。
4. 点击控制台上的创建测试集按钮。
5. 在详细信息中，输入测试集名称和描述（可选）。
6. 选择生成基准测试集。
7. 选择从对话日志生成。
8. 从下拉菜单中选择机器人名称、机器人别名和语言。
9. 要从对话日志生成基准测试，请根据需要选择时间范围和 IAM 角色。您可以使用现有角色，也可以创建基础 Amazon Lex V2 权限的角色。
10. 为您正在创建的测试集选择音频或文本模式。注意：Test Workbench 支持导入的文本文件不得超出 50k，音频文件不得超过 5 小时。
11. 选择用以存储测试结果的 Amazon S3 位置，并视情况添加 KMS 密钥以对输出转录进行加密。
12. 选择创建。

要以 CSV 文件格式上传现有测试集或更新测试集，请执行以下操作：

1. 从左侧面板中选择 Test workbench。
2. 从 Test workbench 下的选项中选择测试集。
3. 在控制台上选择将文件上传到该测试集。
4. 选择从 Amazon S3 存储桶上传或从计算机上传。注意：您可以上传根据模板创建的 CSV 文件。单击 CSV 模板下载包含模板的 ZIP 文件。
5. 选择创建基础 Amazon Lex 权限的角色或者使用角色 ARN 的现有角色。
6. 为您正在创建的测试集选择音频或文本模式。注意：Test Workbench 支持导入的文本文件不得超出 50k，音频文件不得超过 5 小时。
7. 选择用以存储测试结果的 Amazon S3 位置，并视情况添加 KMS 密钥以对输出转录进行加密。
8. 选择创建。

如果操作成功，系统将显示确认消息，表明测试集已准备好进行测试，并且状态将显示为准备进行测试。

成功创建测试集的提示

- 您可以在控制台中为测试工作台创建 IAM 角色，也可以配置您的 IAM 角色 step-by-step。有关更多信息，请参阅为 Test Workbench 创建 IAM 角色。
- 在执行测试之前，点击验证差异按钮验证测试集和机器人定义中是否存在任何不一致之处。如果测试集中使用的意图和槽位命名约定与机器人一致，请继续执行测试。如果发现任何异常，请修改测试集，更新测试集，然后选择验证差异。再次重复此流程，直到没有发现任何不一致之处，然后执行测试。
- Test Workbench 可以使用预期输出槽位列中的不同槽位值格式进行测试。对于任何内置槽位，您可以选择用户输入中提供的值（例如，日期 = 明天），或者提供其绝对解析值（例如，日期 = 2023-03-21）。有关内置槽位及其绝对值的更多信息，请参阅内置槽位。
- 为了 “预期输出槽” 列的一致性和可读性，请遵循 “SlotName = SlotValue”（例如，AppointmentType = cleaning）的惯例，在等号前后留一个空格。
- 如果机器人包含复合槽位，则在预期输出槽位中定义槽位名称的子槽位，以句点隔开（例如，“Car.Color”）。任何其他句法和标点符号均无效。
- 如果机器人包含多值插槽，则在 “预期输出槽” 中提供多个插槽值，用逗号分隔（” FlowerType = roses，lilies”）。任何其他句法和标点符号均无效。
- 确保测试集是根据有效的对话日志创建的。
- Slot:slot 值将位于 CSV 格式的意图列之后的同一列中。
- 来自用户回合的 DTMF 输入被解释为预期的转录，不会列出 Amazon S3 的位置。

在测试集中创建测试用例

Test Workbench 结果取决于机器人定义及其相应的测试集。您可以使用机器人定义中的信息生成测试集，以查明需要改进的方面。您可以基于当前机器人的设计以及您对客户对话的了解来创建一个测试数据集，其中包含您认为（或确定）机器人难以正确解释的示例。

定期根据从生产机器人中了解到的信息来审查意图。继续添加和调整机器人的示例言语和值。考虑使用运行时提示等可用选项来提高槽位分辨率。机器人的设计和开发是一个连续迭代的过程。

以下是关于优化测试集的其他一些提示：

- 在测试集中选择常用意图和槽位的最常见用例。

- 探索客户可能用到您的意图和槽位的不同方式。这可能包括以长短不一的陈述、问题和命令形式进行的用户输入。
- 包括具有不同数量的槽位的用户输入。
- 包括机器人支持的自定义槽位值的常用同义词或缩写（例如，“根管”、“管” 或“RC”）。
- 包括内置槽位值的变体（例如，“明天”、“尽快” 或“第二天”）。
- 通过收集可能被误解的用户输入（例如“墨水”、“脚踝”或“锚点”），检查机器人在对话语模式下的稳健性。

通过 CSV 文件创建测试集

您可以使用 CSV 电子表格编辑器在 Amazon Lex V2 控制台中提供的 CSV 文件模板中直接输入值，以此来创建测试集。测试集是一个逗号分隔值 (CSV) 文件，其中包括记录了单用户言语和多回合对话的以下列：

- 行号：该列逐个递增，以跟踪要测试的已填行总数。
- 对话编号：该列用于跟踪某个对话中的回合数。对于单个输入，此列可以留空，填入“-”或“N/A”。对于对话，对话中的每个回合都将被分配相同的对话编号。
- 来源：该列被设置为“用户”或“代理”。对于单个输入，该列始终被设置为“用户”。
- 输入：该列包括用户言语或机器人提示。
- 预期输出意图：该列用于捕获输入中履行的意图。
- 意图预期输出槽位 1：该列用于捕获用户输入中引发的第一个槽位。测试集应为用户输入中的每个槽位包含一个名为“预期输出槽位 X”的列。

单个输入的测试集示例：

行号	对话编号	来源	输入	预期输出意图	预期输出槽位 1	预期输出槽位 2
1		用户	预约明天的清洁服务	MakeAppointment	AppointmentType = 清洁	Date = 明天
2	不适用	用户	预约 4 月 15 日的清洁服务	MakeAppointment	AppointmentType = 清洁	Date = 4/15/23

行号	对话编号	来源	输入	预期输出意图	预期输出槽位 1	预期输出槽位 2
3	不适用	用户	预约 12 月 1 日	MakeAppointment	日期 = 12 月 1 日	
4	不适用	用户	预约清洁服务	MakeAppointment	AppointmentType = 清洁	
1		用户	能否帮我预约？	MakeAppointment		

对话的测试集示例

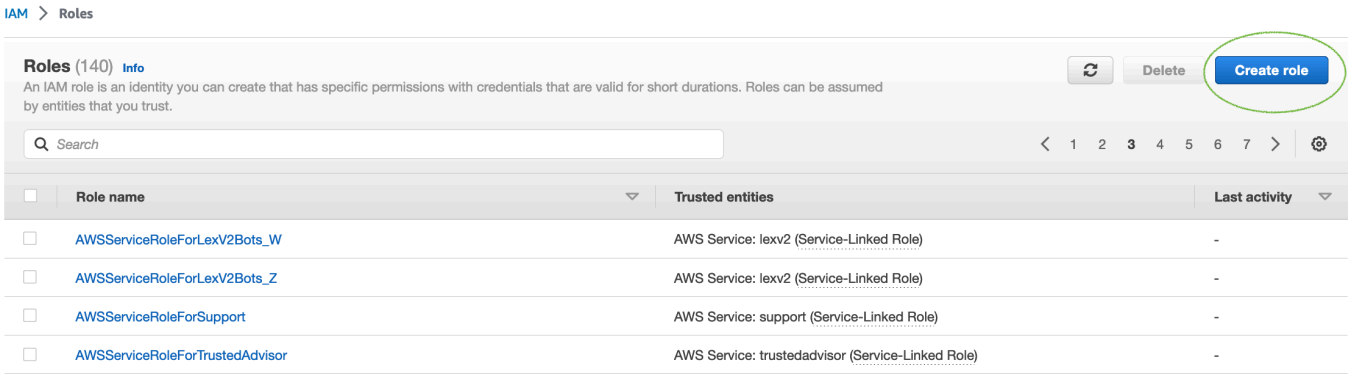
行号	对话编号	来源	输入	预期输出意图	预期输出槽位 1	预期输出槽位 2	预期输出槽位 3
1	1	用户	预约	MakeAppointment			
2	1	座席	您想要安排哪种类型的预约？	MakeAppointment			
3	1	用户	清洁	MakeAppointment	AppointmentType = 清洁		
4	1	座席	您需要预约到什么时候？	MakeAppointment			
5	1	用户	tomorrow	MakeAppointment		Date = 明天	

行号	对话编号	来源	输入	预期输出意图	预期输出槽位 1	预期输出槽位 2	预期输出槽位 3
6	2	用户	预约今天的根管治疗	MakeAppointment	AppointmentType = 根管	Date = 今天	
7	2	座席	您需要预约到什么时候？	MakeAppointment			
8	2	用户	上午十一点	MakeAppointment			Time = 上午十一点

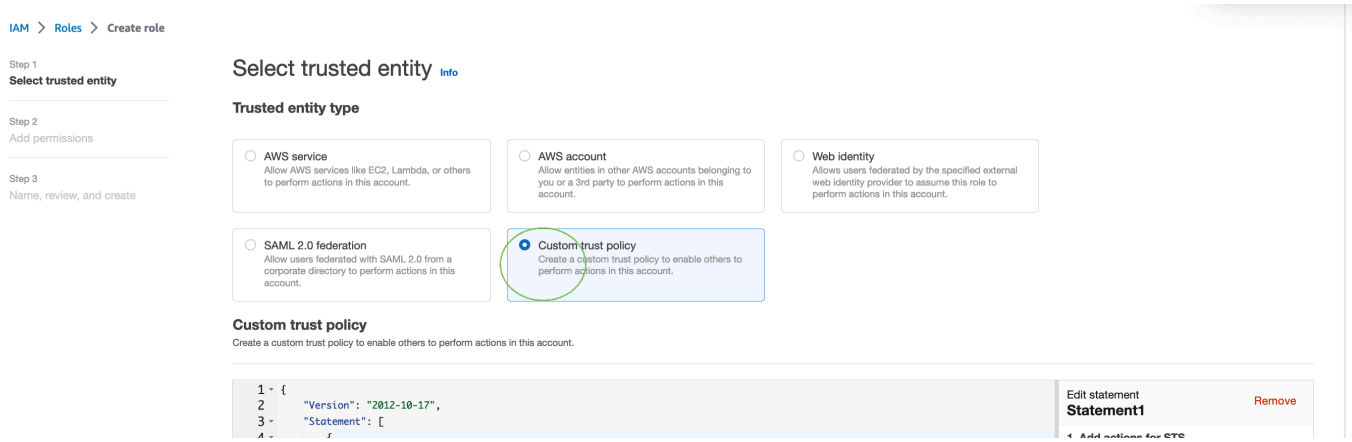
为 Test Workbench 创建 IAM 角色

为 Test Workbench 创建 IAM 角色，请执行以下操作：

1. 按照创建 IAM 用户中的步骤创建可用于访问 Test Workbench 控制台的 IAM 用户。
2. 选择 Create role 按钮。



3. 选择自定义信任策略选项。



4. 输入以下信任策略，然后单击下一步。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "sid4",
      "Effect": "Allow",
      "Principal": {
        "Service": "lexv2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

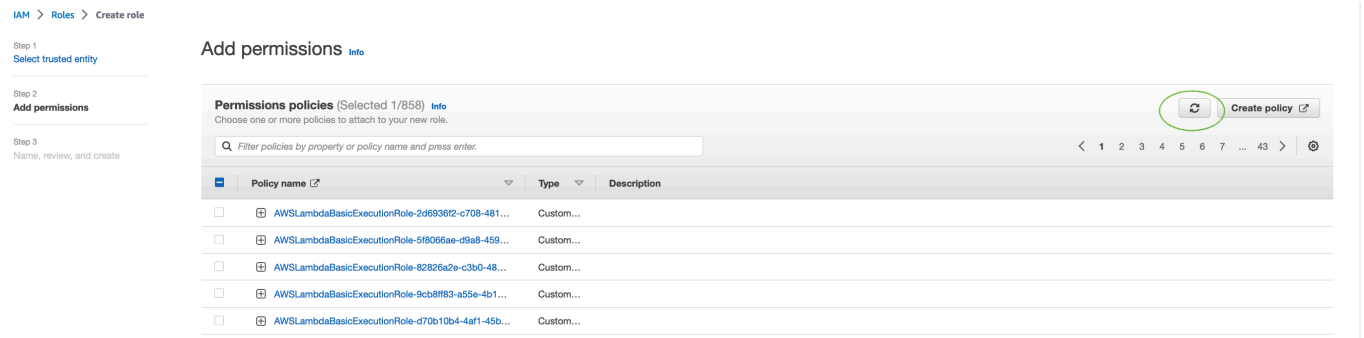
5. 点击创建策略按钮。

6. 浏览器中将打开一个新标签页，请在该页面上输入以下政策，然后单击下一步：标签按钮。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
```

```
            "Action": [
                "logs:FilterLogEvents"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "lex:*"
            ],
            "Resource": "*"
        }
    ]
}
```

7. 输入策略名称，例如 “LexTestWorkbenchPolicy”，然后单击 “创建策略”。
8. 返回浏览器中的上一个标签页，然后单击刷新按钮以刷新策略列表，如下所示。



9. 在策略列表中输入您在第 6 步中使用的策略名称以搜索该策略，然后选择该策略。
10. 点击下一步按钮。
11. 输入角色名称，然后单击创建角色按钮。
12. 在 Amazon Lex V2 控制台中显示 Test Workbench 提示时，请选择您的新 IAM 角色。

管理测试集

您可以从测试集窗口下载、更新和删除测试集，也可以使用可用测试集列表来编辑或手动注释您的测试集文件。然后，由于错误或其他输入问题，请再次上传以重试验证。

要从测试集记录下载测试集文件，请执行以下操作：

1. 从测试集列表中选择测试集的名称。
2. 在测试集记录窗口中，请在屏幕右侧测试输入部分中点击下载按钮。

3. 如果窗口顶部有关于测试集的任何验证错误详细信息，请点击下载按钮。该文件将保存到“下载”文件夹中。您可以通过测试集 CSV 文件中的错误消息来修复测试集中的验证错误。找到在验证步骤中发现的错误，修复该行或将其删除，然后上传文件以重试验证步骤。
4. 如果您成功下载了测试集，则会显示一条绿色横幅消息。

要从测试集列表中下载测试集，请执行以下操作：

1. 在测试集列表中选中要下载的测试集项目旁边的单选按钮。
2. 在右上角的“操作”菜单中选择下载。
3. 如果您成功下载了测试集，则会显示一条绿色横幅消息。该文件将保存到“下载”文件夹中。

查看测试验证错误

您可以对报告验证错误的测试集进行更正。如果测试集尚未准备好进行测试，就会生成这些验证错误。Test Workbench 可以显示测试集输入 CSV 文件中哪些必填列不存在预期格式的值。

要查看测试验证错误，请执行以下操作：

1. 在测试集列表中选择要查看的报告状态为验证错误的测试集的名称。测试集的名称是有效链接，点击该链接可跳转至相关页面以查看有关测试集的详细信息。
2. 测试集记录在屏幕顶部显示验证错误的详细信息。选择查看详细信息以查看验证错误报告。
3. 在错误报告窗口中，依据行号和错误类型来确定发生错误的位置。对于一长串错误，您可以选择下载错误报告。
4. 将测试集输入 CSV 文件中列出的错误与原始测试文件进行比较，以更正所有问题并重新上传测试集。

下表列出了输入 CSV 验证错误消息及对应场景。

场景	错误消息	注意
超出测试集文件大小限制	测试集文件大于 200 MB. Provide smaller file and try your request again.	

场景	错误消息	注意
测试集超过最大记录数	Input file had records more than supported maximum number of 200,000.	
上传的测试集为空	Imported test set is empty. Provide non-empty test set and try your request again.	
列标题名称为空	Column Headers Row: found empty column name in column number 5.	
列标题名称无法识别	Column Headers Row: could not recognize column name 'dummy' in column number 2.	
列标题名称重复	Column Headers Row: found multiple columns 'S3 audio link' and 'S3 audio link' that are same or equivalent. Remove or rename one of those columns.	
多值列名超出限制	Column Headers Row: count of columns for 'Expected Output Slot' exceeded maximum supported count: 6. Remove some columns for 'Expected Output Slot' and try again.	多值列支持的最大列数为 6。

场景	错误消息	注意
文本或音频相关的列标题不存在	Could not find columns for text or audio conversations. For text conversations, use {'Text input'} columns. For audio conversations, use {'S3 audio link', 'Expected transcription'} columns.	必填音频列：{'S3 audio link', 'Expected transcription'}，必填文本列：{'Text input'}
存在与文本和音频均相关的列标题	Found columns for both text and audio conversat ions. You can either use {'Text input'} columns for text conversations, or {'S3 audio link', 'Expected transcription'} columns for audio conversat ions.	必填音频列：{'S3 audio link', 'Expected transcription'}，必填文本列：{'Text input'}
缺少必填列	Could not find mandatory columns ["Expected Output Intent"].	必填列：{"Line #", "Source", "Expected Output Intent"}
存在有数据但无标题的列	Found data in column number 8 for row number 6, but corresponding column did not have a column header.	
必填列无数据	Row=12: no values found for mandatory columns: {"Source" , "Expected Output Intent"}	

场景	错误消息	注意
对话 ID 重复	conversation number '19' was seen for previous conversat ion at row number 39." Make sure that same conversation number has not been provided for two conversations, you can do this by ensuring that all rows for a conversation number are grouped together.	
提供的对话 ID 无效	Found invalid value 'test_con versation' in 'Conversation #' column. Value for this column must be either numeric or N/A (i.e. Not Applicable) for a user row.	
行号列的值为非数字	Found non-numeric value 'test_line' in 'Line #' column. Its value must be numeric.	
代理行中无会话 ID	No value found for 'Conversa tion #' column. It must be provided for an agent row.	
代理行中无数值会话 ID	Found non-numeric value 'test_conversation' in 'Conversation #' column. Its value must be numeric for an agent row.	
S3 位置无效	Invalid value 'bucket/folder' was provided. Valid format is S3://<bucketName>/ <keyName>.	

场景	错误消息	注意
S3 存储桶名称无效	Invalid s3 bucket name 'test_bucket' was provided. Check the bucket name.	
S3 音频位置为文件夹	Provided audio location 'S3://bucket/folder' is invalid. It points to an S3 folder.	
意图名称无效	Invalid characters were present in intent 'intent@name'. Check the intent name.	正则表达式检查：^([0-9a-zA-Z][_-]?)+$
槽名称无效	Invalid characters were present in slot 'Slot@Name'. Check the slot name.	正则表达式：^([0-9a-zA-Z][_-]?)+$，开头或结尾不得为点 (.)
槽位值被赋予父槽位	Slot values were provided for subslot 'Address.City' as well as parent slot 'Address'. Values should be only provided for the subslot.	CST 中的父槽位不得有槽位值
上下文名称中的字符无效	Invalid characters were present in context name 'context@1'. Check the context name.	正则表达式：^([A-Za-z]_?)+$
槽位拼写样式无效	Invalid value 'test' was provided. Make sure that they are all upper case. 有效值为 ["默认”、“字SpellBy母”、“SpellByWord"]。	支持的值 [“默认”、“字SpellBy母”、“SpellByWord”
参与者或来源必须是代理或用户	Invalid value 'bot' was provided. Valid values are ["Agent", "User"].	支持的枚举：“代理”、“用户”

场景	错误消息	注意
行号不得为小数	Invalid value '10.1' was provided. It should be a valid number without any fractions.	
对话编号不得为小数	Invalid value '10.1' was provided. It should be a valid number without any fractions.	
行号应在限定范围内	Invalid value '92233720 368547758071' was provided. It should be greater than or equal to 1 and less than or equal to 922337203 6854775807.	
插入列只能为布尔值	Invalid value 'test' was provided. It should be a valid boolean value such as 'true' or 'false'. Alternatively 'yes' and 'no' can be used.	可能的值：“Tru e”、“true”、“T”、“Yes”、“yEs”、“Y”、“1 lse”、“false”、“F”、“No”、“no”、“N”、“0
预期槽位、会话属性、请求属性应以等于号（=）隔开	Value 'slotName:slotValue' does not have '='. Such value should be provided as a key-value pair in format '<key>=<value>'.	例如：slotName = slotType
预期槽位、会话属性、请求属性应为密钥值对	'=slotValue' does not have a key before '='. Such value should be provided as a key-value pair in format '<key>=<value>'.	例如：slotName = slotType

场景	错误消息	注意
结尾处的引号无效	Found incorrect quoting in 'Lenny's Burger'“. It starts with quote character `'“` but does not end with same quote character.	例如：“Lenny's Burger”、KFC
中间的引号无效	Found incorrect quoting in `'"Lenny's" Burger, KFC`. It contains quote character `'“` inside its content. Values containing single quotes should be wrapped within double quotes and vice-versa.	例如，正确格式应为：“Lenny's Burger”、KFC
缺少引号	`key = Lenny's Burger` contains single-quotes or double-quotes but has not been wrapped inside quotes. Values containing single quotes should be wrapped inside double quotes and vice-versa.	
列中多处存在重复密钥	Key `key1` was repeated in two columns: `Session Attribute 3` and `Session Attribute 1`.	
运行时提示中的格式无效	密钥无效 `BookFlight.Car。“`为运行时提示提供。对于运行时提示，密钥应采用格式 <intentName>.<slotName>。	如果 '.' 必须出现在密钥中间，则无法从该密钥中提取意图名称和插槽名称。格式错误的示例：BookFlight“，”。BookFlight.Car”、“BookFlight.Car。”

场景	错误消息	注意
运行时提示键中的意图名称无效	Found invalid intent `intent@name` for Runtime Hints. Check intent name.	正则表达式检查：^([0-9a-zA-Z][_-]?)+$
运行时提示键中的槽位名称无效	Found invalid slot name in `Slot@Name` for Runtime Hints. Check slot name.	正则表达式：^([0-9a-zA-Z][_-]?)+$，开头或结尾不得为点 (.)

删除测试集

您可以轻松地从测试集列表中删除测试集。

要删除测试集，请执行以下操作：

1. 从左侧菜单转到测试集列表以查看测试集。
2. 在测试集列表中选择要删除的测试集。
3. 点击右上角的操作下拉菜单，然后选择删除。
4. 系统将显示一条消息，确认测试集已删除。

编辑测试集详细信息

您可以在测试集列表中编辑测试集名称和详细信息。测试集的名称或详细信息可以在后期添加或更新。但是，在使用机器人或转录数据运行测试之前，必须更新测试集。

要编辑测试集详细信息，请执行以下操作：

1. 从左侧菜单转到测试集列表以查看测试集。
2. 在测试集列表中，选中要编辑测试集的复选框。
3. 点击右上角的操作下拉菜单，然后选择编辑详细信息。
4. 系统将显示一条消息，确认测试集的编辑已成功保存。

更新测试集

您可以更新、更正、修改或删除测试集中的项目，以优化基准结果，或者更正测试集中可能出现的其他错误

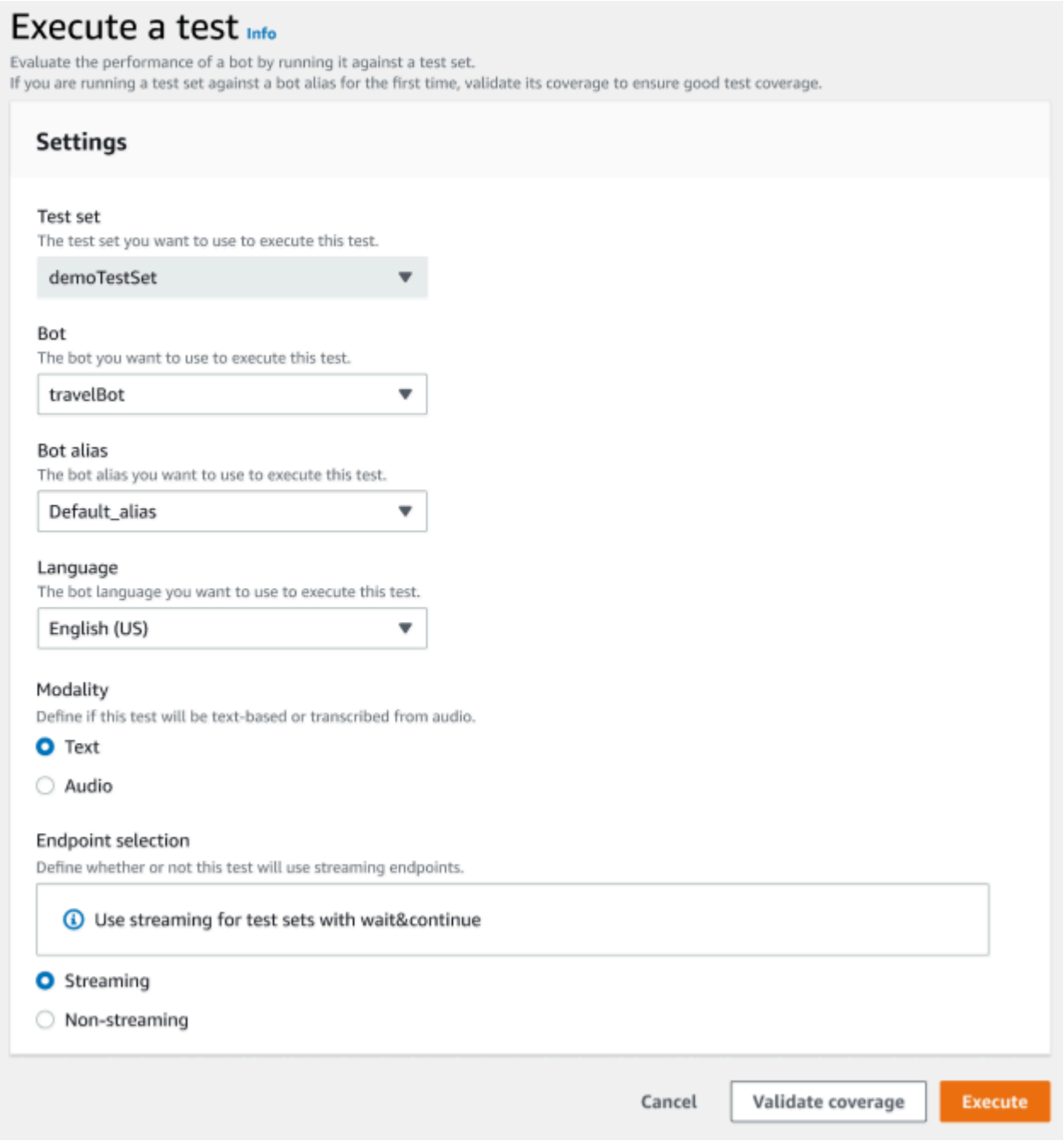
您可以先下载测试集并修复验证错误，然后再上传更正后的测试集。请参见查看测试验证错误。

要更新测试集，请执行以下操作：

1. 在测试集记录中，点击右上角的更新测试集按钮。
2. 选择要从您的 Amazon S3 账户上传的文件，或者从您的计算机上传 CSV 测试文件。注意：更新测试集将覆盖现有数据。
3. 点击更新按钮。
4. 系统将显示一条消息，确认测试集已成功更新。注意：该操作可能需要几分钟完成，具体取决于测试集的复杂性和大小。
5. 系统将显示一条消息，确认测试集已成功更新，同时状态将变为准备进行测试。

执行测试

要执行测试集，必须选择相应的机器人来针对该测试集运行测试。你可以从 Test Set 下的下拉菜单中从你的 AWS 账户中选择一个机器人。该操作将根据经过验证的测试数据测试您选择的机器人，以根据测试集中的基准数据报告性能指标。



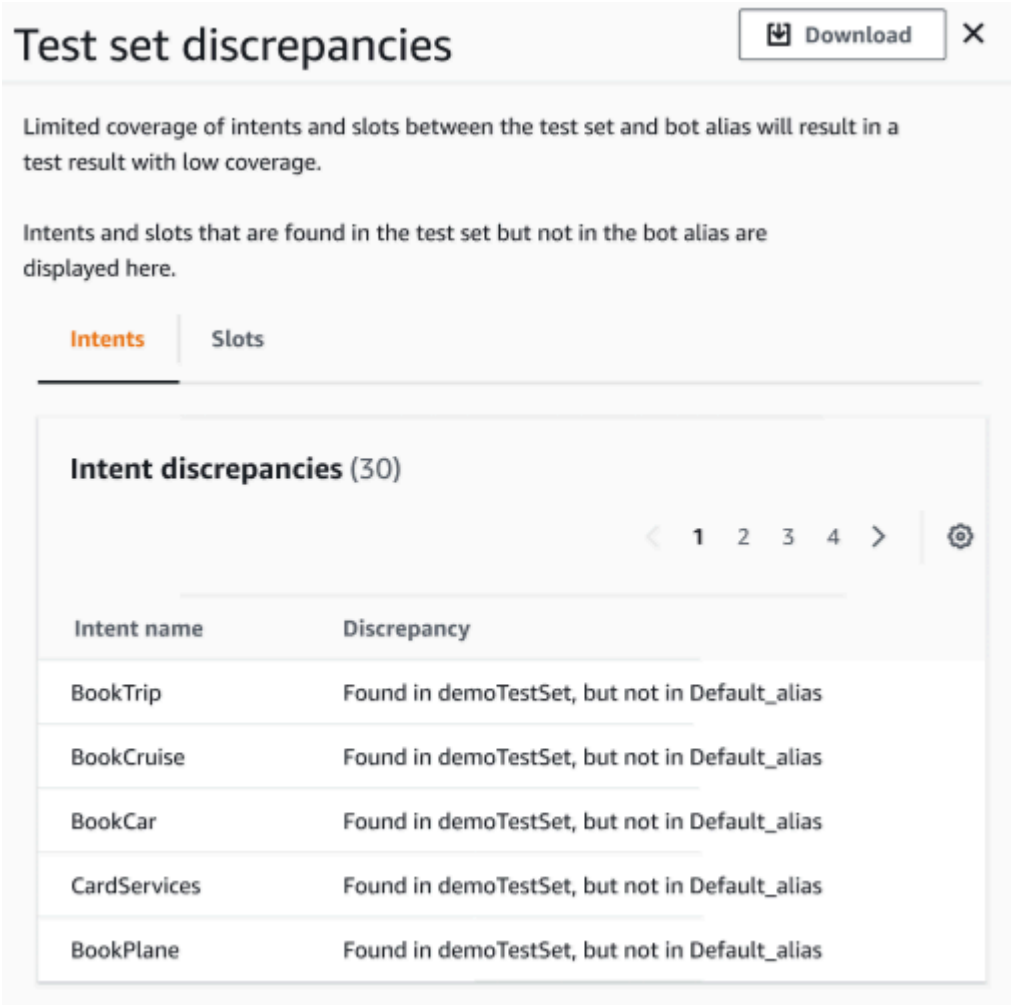
要在 Test Workbench 中执行测试，请执行以下操作：

1. 在测试集记录页中，选择执行测试。
2. 选择要在测试中使用的测试集。
3. 从机器人下拉菜单中选择要在测试中使用的机器人的名称。
4. 在机器人别名下拉菜单中选择机器人别名（如果适用）。
5. 在语言下选择英语版本。

6. 在“模态”下选择文本或音频。
7. 选择您的 Amazon S3 位置。（仅限音频）
8. 选择机器人的端点选择。（仅限流式传输）
9. 点击验证覆盖范围按钮以确认测试已准备就绪。如果验证步骤中存在任何错误，请检查之前的参数并进行更正。
10. 选择执行以运行测试。
11. 系统将显示一条消息，确认已成功执行该测试。

测试集覆盖范围

可通过测试集与机器人之间意图和槽位的有限覆盖范围来产生预期的性能测量值。我们建议您在运行测试之前检查测试集覆盖范围。



要检查验证覆盖范围，请执行以下操作：

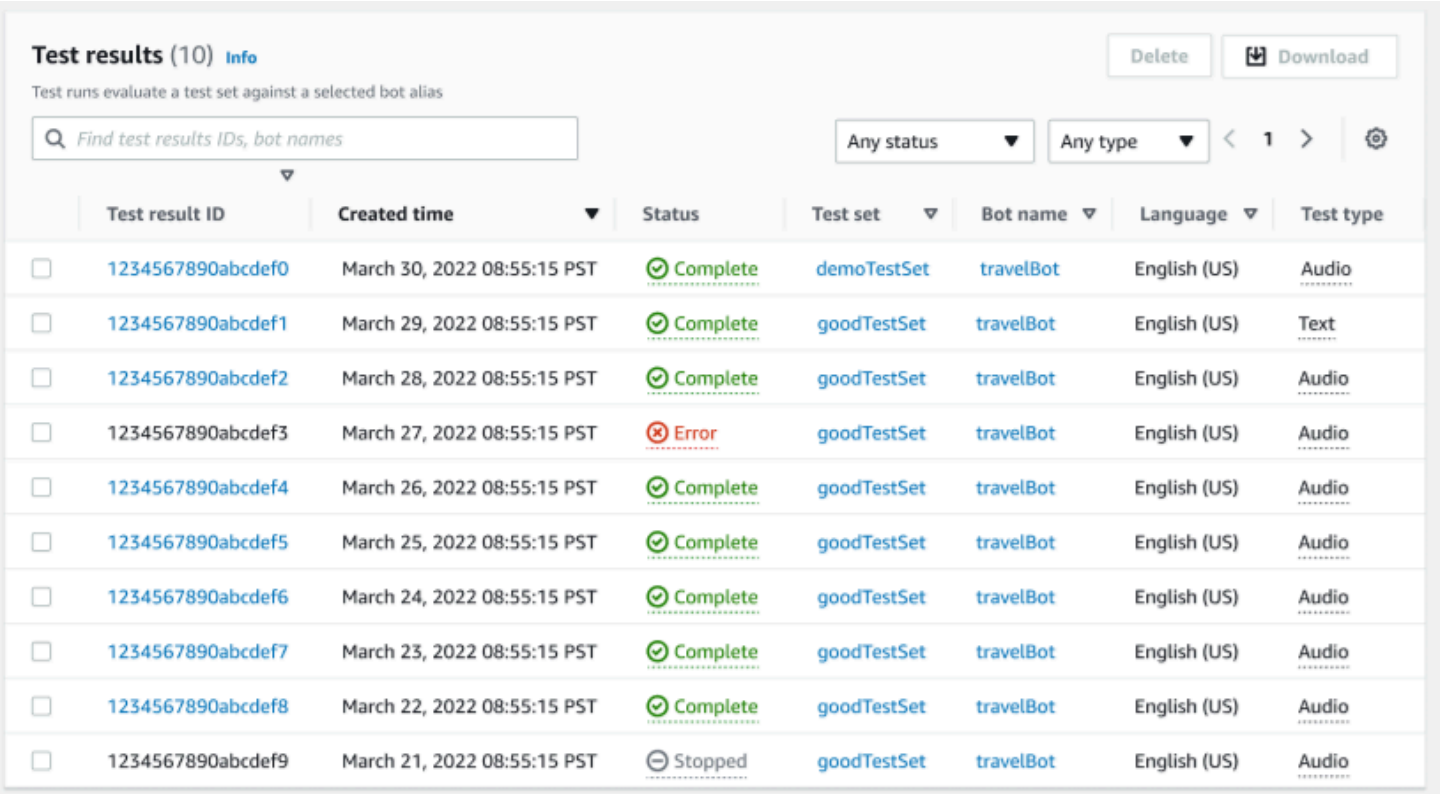
1. 在测试集记录中，点击验证覆盖范围按钮。
2. 系统将显示消息，以指示正在验证测试集与所选机器人之间的覆盖范围。
3. 操作完成后，系统将显示消息，以指示覆盖范围验证成功。
4. 点击窗口底部的查看详细信息按钮。
5. 选择“意图”或“槽位”选项卡以查看对应的测试集差异。您可以点击下载按钮以便以 CSV 格式的文件下载该数据。
6. 查看您的测试集数据、机器人意图和槽位的验证结果。找出问题并相应地更改您的机器人测试集架构以改善结果。更改 CSV 文件后，上传编辑后的测试集和机器人以运行测试。注意：覆盖范围的验证不是针对机器人，而是针对测试集运行的。因此将不涉及机器人中存在、但测试集中不存在的意图。

查看测试结果

解读 Test Workbench 的测试结果，以确定机器人与客户之间的对话可能失败的场景，或者需要客户多次尝试才能履行意图的场景。

通过在测试结果中找到这些问题，您可以使用与实时机器人转录值更一致的不同训练数据或言语来提高意图性能，从而优化机器人的性能。

您可以详细了解存在性能差异的意图和槽位。一旦确定了存在差异的意图或槽位，即可进一步深入研究和检查言语和对话流。



要查看测试结果，请执行以下操作：

1. 通过左侧的菜单转到测试集列表，选择 Test Workbench 下的测试结果选项。注意：如果测试结果为成功，即表明状态为“完成”。
2. 选择要查看测试结果的测试结果 ID。

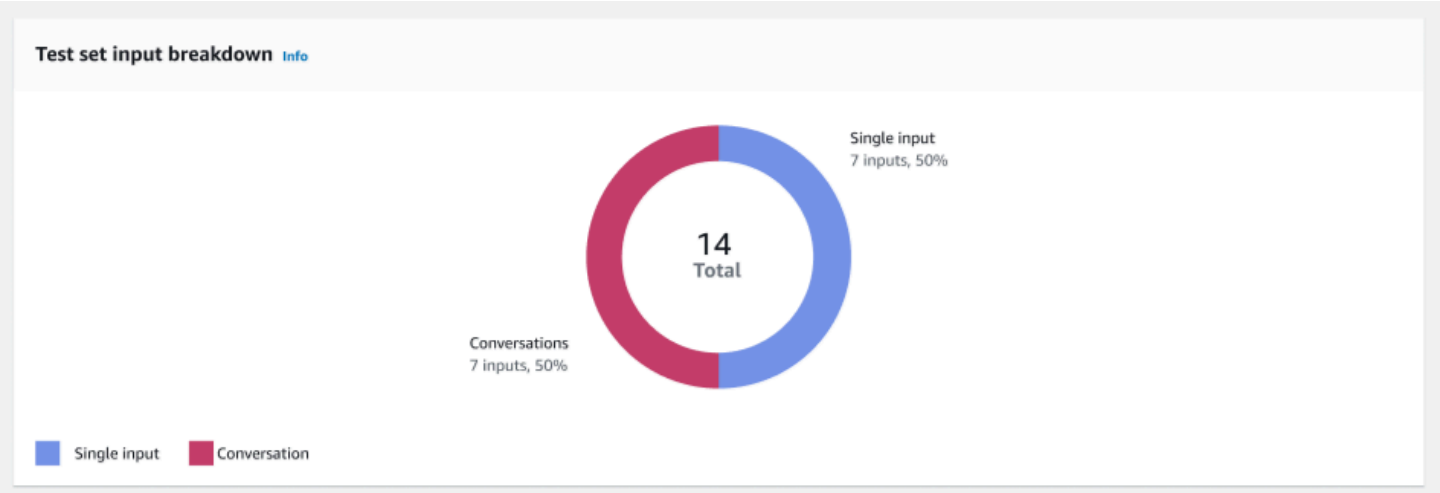
测试结果详细信息

测试结果显示了测试集的详细信息、所使用的意图以及所使用的槽位。其中还提供了总体测试集输入明细，包括总体结果、对话结果、意图和槽位结果。

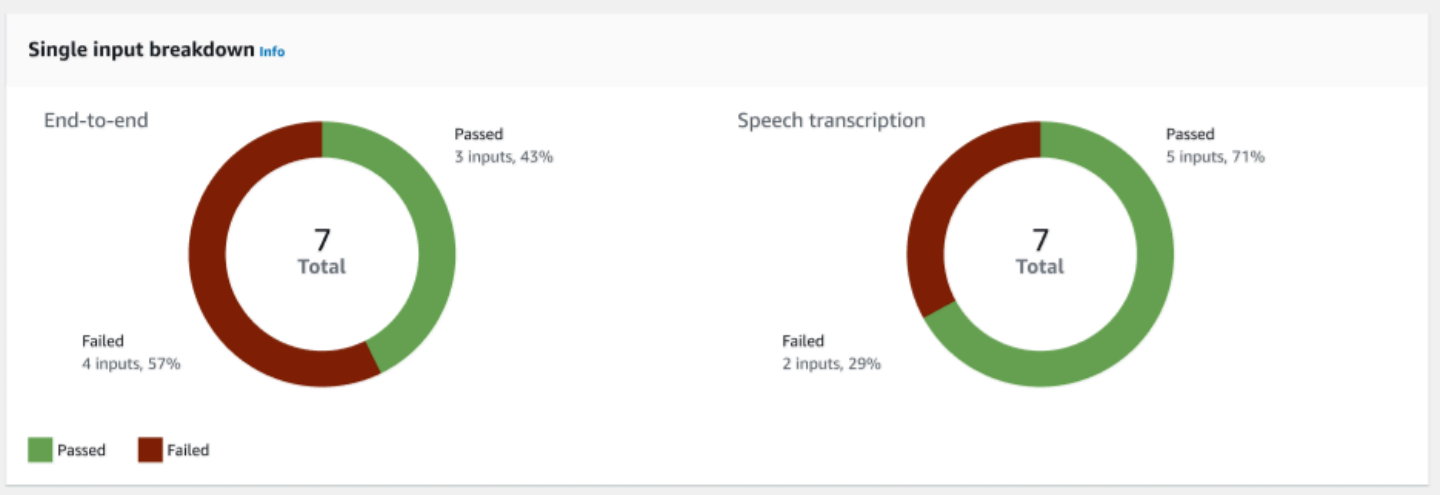
测试结果包括与测试相关的所有信息，例如：

- 测试详细信息元数据
- 总体结果
- 对话结果
- 意图和槽位结果
- 详细结果

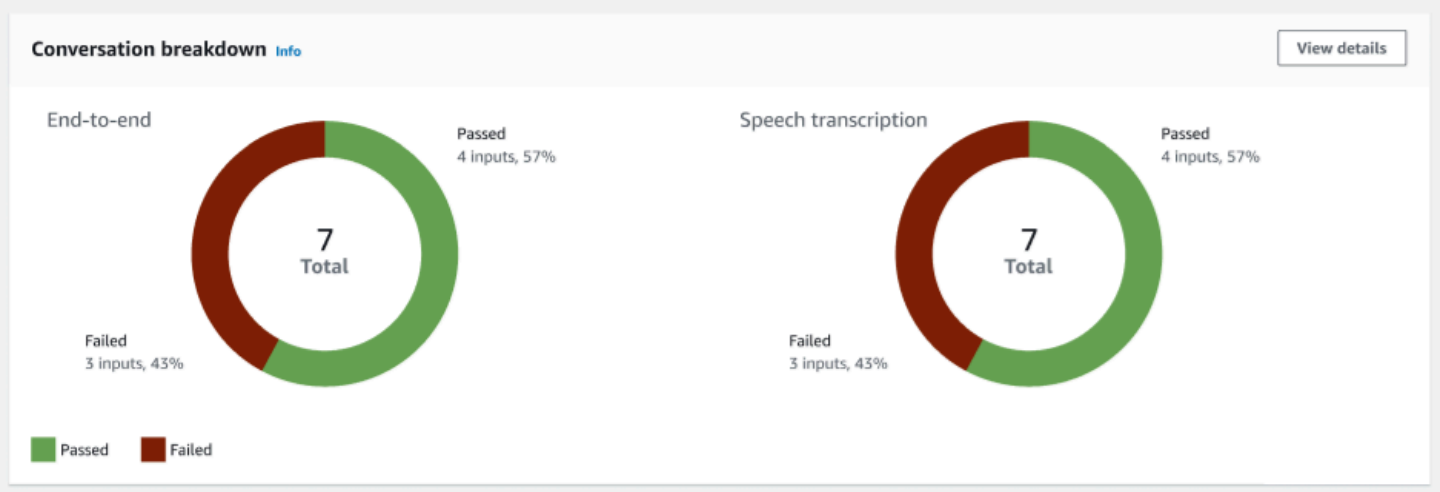
“总体结果”选项卡：



测试集输入细分：该图中细分地展示了测试集中的对话数量和单一输入言语数量。

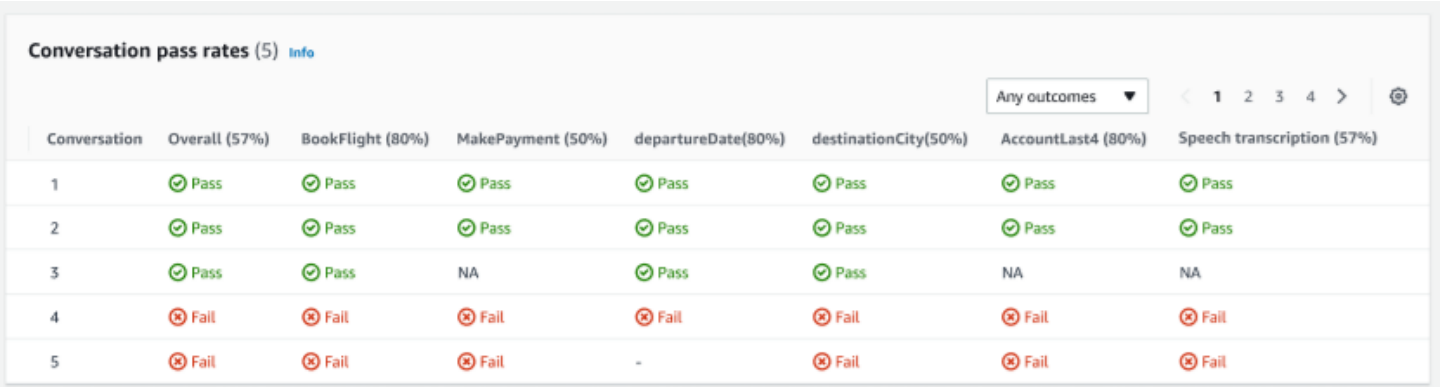


单一输入细分-显示两个图表，其中包括 end-to-end 对话和语音转录。每幅图中均分别展示了通过的输入数量以及失败的输入数量。注意：话语转录图仅适用于音频测试集。

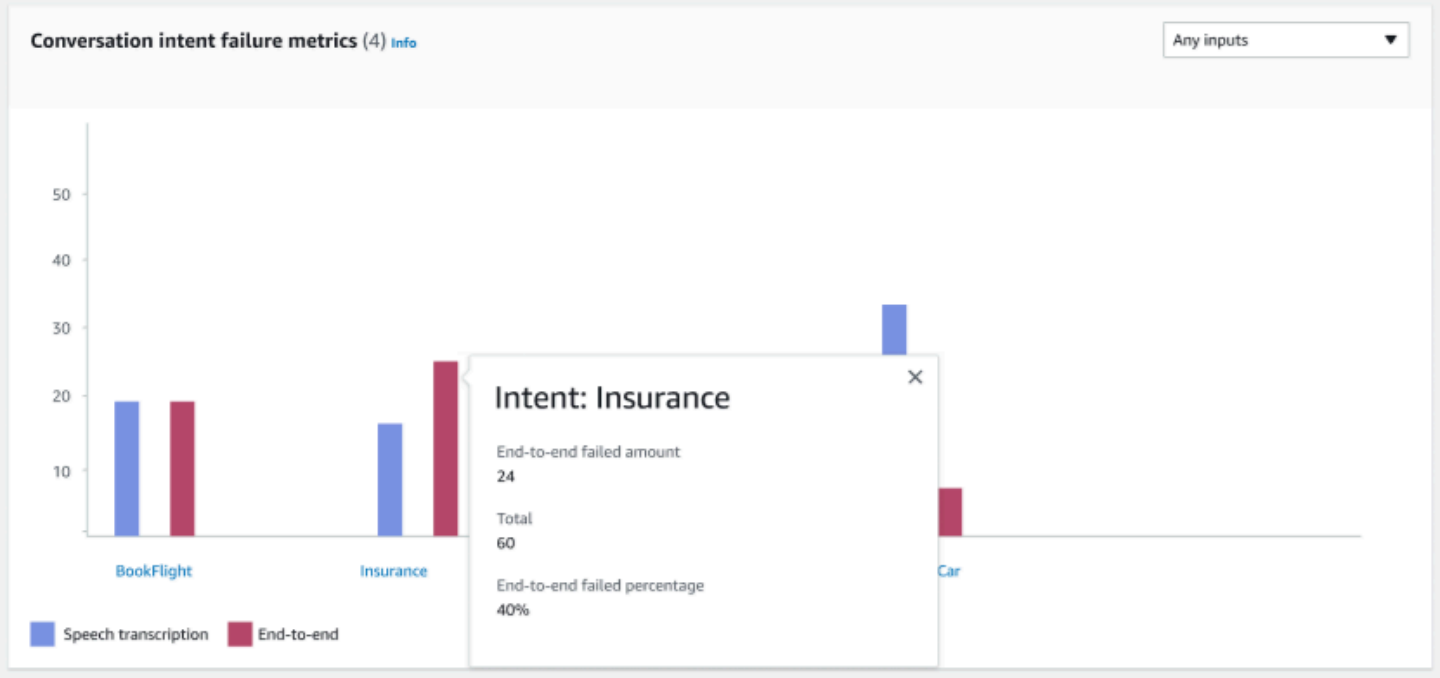


对话细分-显示两个图表，其中包括 end-to-end 对话和语音转录。每幅图中均分别展示了通过的输入数量以及失败的输入数量。注意：话语转录图仅适用于音频测试集。

“对话结果”选项卡：



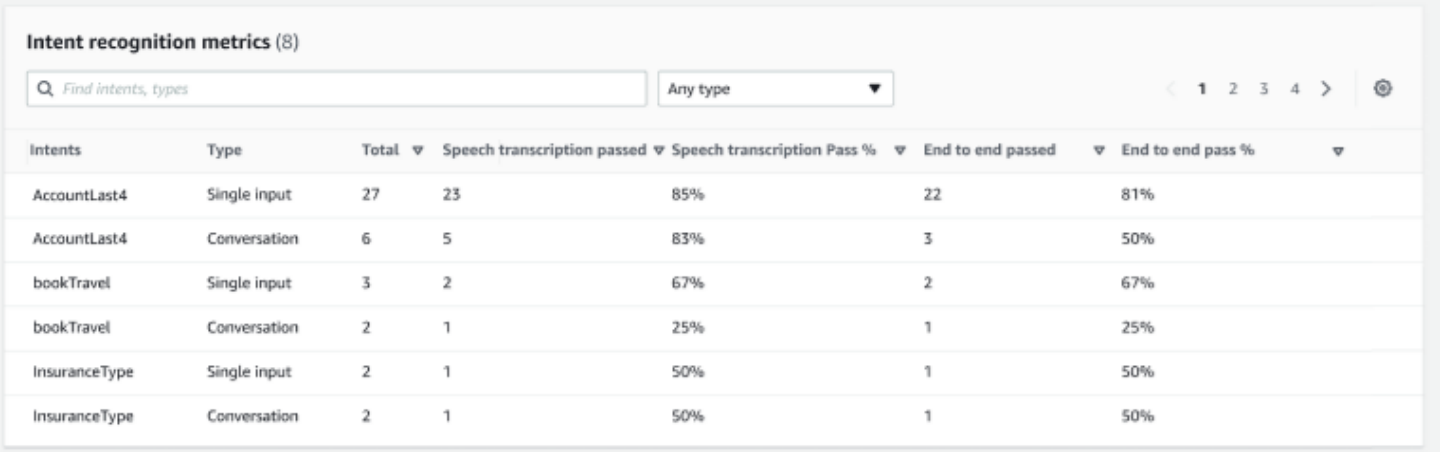
对话通过率：您可在对话通过率表中查看测试集中的每个对话中所使用的意图和槽位。通过查看失败的意图和槽位以及每个意图和槽位的通过百分比，您可以直观地看到对话失败的地方。



对话意图失败指标：该指标显示了测试集中性能最差的 5 个意图。该面板基于机器人的对话日志或转录，以图表的形式展示成功或失败意图的百分比或数量。成功的意图并不意味着整个对话是成功的。这些指标仅适用于意图的值，与意图出现的先后顺序无关。

对话槽位失败指标：该指标显示了测试集中性能最差的 5 个槽位。其中会指示意图中每个槽位的成功率。条形图显示意图中每个时段的语音转录和 end-to-end 对话。

“意图和槽位结果”选项卡：

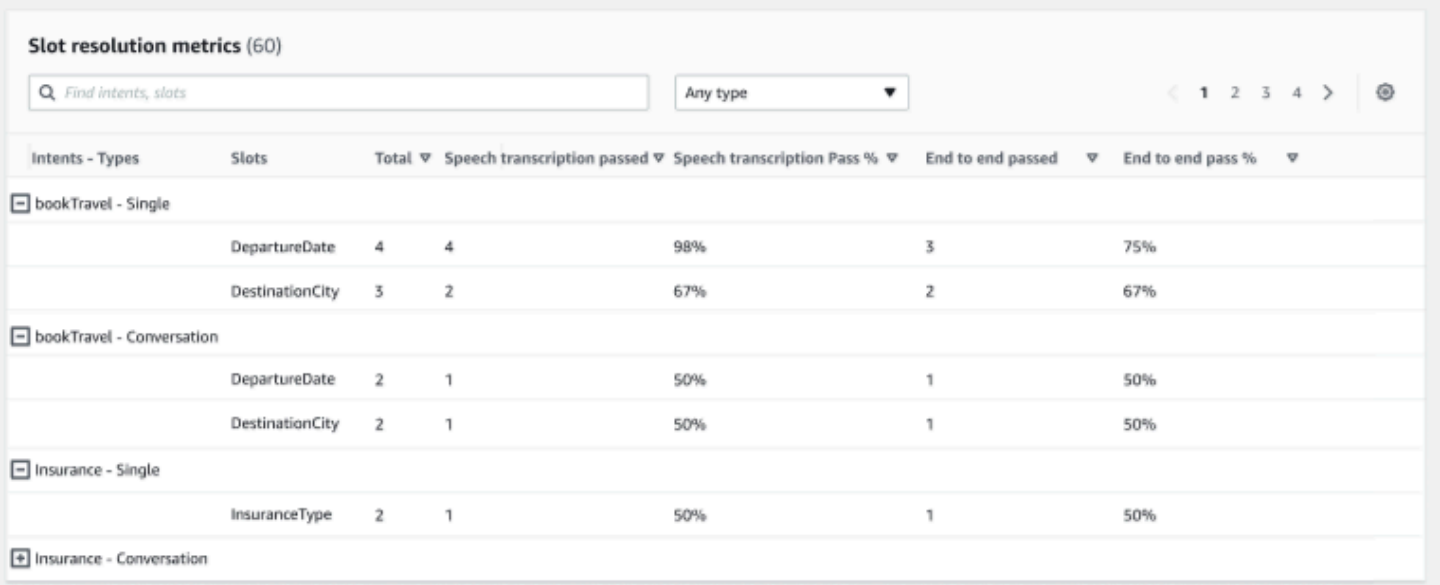


Intent recognition metrics (8)

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%

意图识别指标：以表格形式展示成功识别的意图数量。显示语音转录和 end-to-end 对话的通过率。



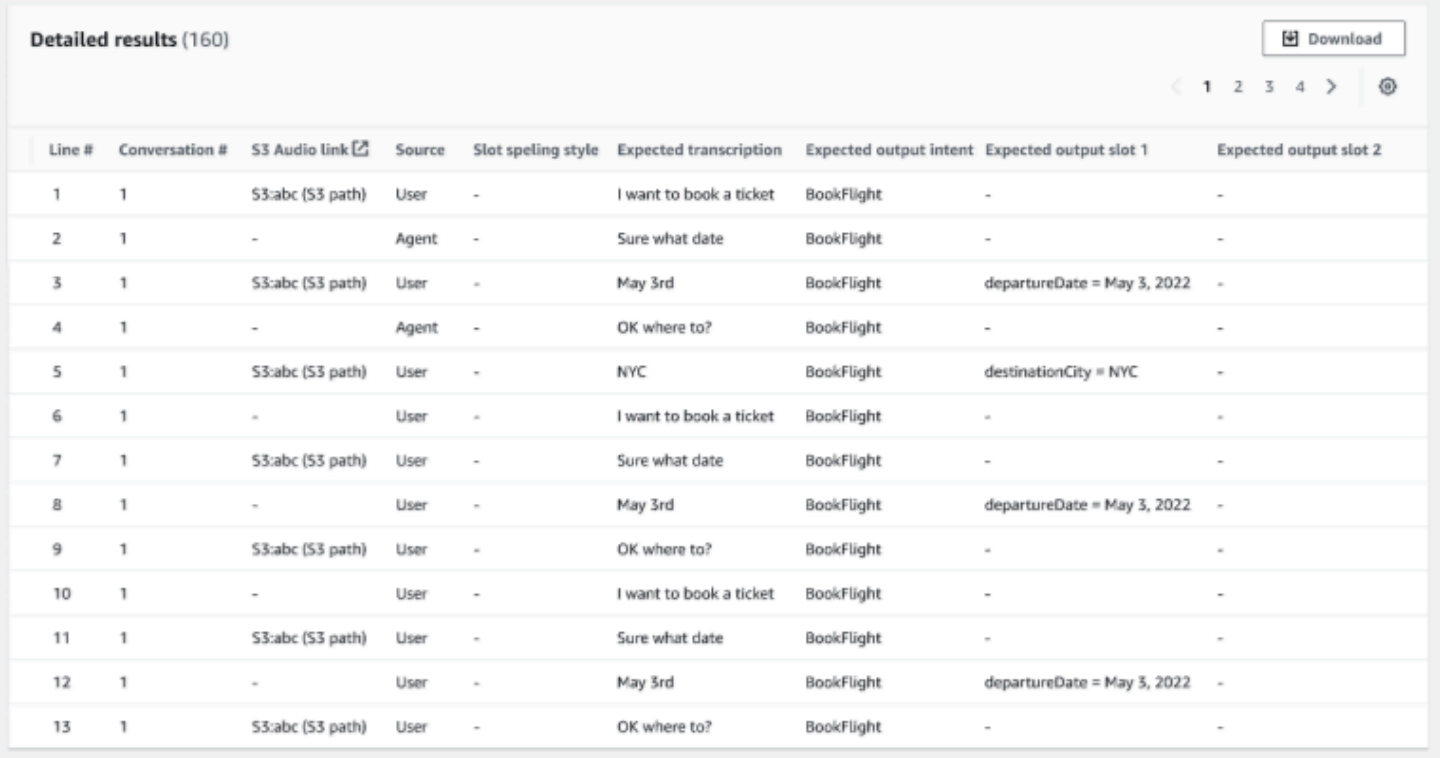
Slot resolution metrics (60)

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						

槽位分辨率指标：分别展示意图和槽位，以及对话或单一输入中所使用的每个意图中每个槽位的成功率和失败率。显示语音转录和 end-to-end 对话的通过率。

“详细结果”选项卡：



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	-	-

详细结果：展示对话日志的详细表格，其中包括用户和代理的言语以及每个槽位的预期输出和预期转录。您可以通过点击下载按钮来下载此报告。

下表列出了结果失败错误消息及对应场景。

场景	错误消息	操作
意图不匹配	预期的 BookFlight 意图，但这是 BookHotel 故意的。	跳过对话中的其他回合
槽位引发不匹配	Expected departureDate slot to be elicited but it was cabinType.	跳过对话中的其他回合
槽位值不匹配	Mismatch between expected and actual slot value.	继续对话中的其他回合
缺少 B ack-to-back 代理提示符	Expected bot to return an agent prompt in this turn but it was not received.	跳过对话中的其他回合

场景	错误消息	操作
转录不匹配	Expected transcription didn't match actual transcription.	继续对话中的其他回合
未引发可选槽位	Expected to elicit cabinType slot in next turn, however current intent fulfilled before that.	跳过对话中的其他回合
槽位无法识别	Expected departureDate slot was not recognized in this turn.	跳过对话中的其他回合
额外的 back-to-back 代理提示	Expected a user turn but it was agent prompt	跳过对话中的其他回合

流式传输到 Amazon Lex V2 机器人

您可以通过 Amazon Lex V2 流式传输 API 在 Amazon Lex V2 机器人与您的应用程序之间启动双向流。启动流后，机器人可以管理机器人与用户之间的对话。机器人会响应用户输入，而无需您编写代码来处理用户的响应。该机器人可以：

- 处理在播放提示时来自用户的中断。有关更多信息，请参阅允许用户打断机器人。
- 等待用户提供输入。例如，机器人可以等待用户收集信用卡信息。有关更多信息，请参阅允许机器人等待用户提供更多信息。
- 将双音多频 (DTMF) 和音频输入放在同一个流中。
- 与通过应用程序管理对话相比，可以更好地处理用户输入中的暂停。

Amazon Lex V2 机器人不仅会响应您的应用程序发送的数据，还会将有关对话状态的信息发送到您的应用程序。您可以通过该信息来更改应用程序响应客户的方式。

Amazon Lex V2 机器人还会监控该机器人与您的应用程序之间的连接，并可以确定连接是否已超时。

要通过该 API 向 Amazon Lex V2 机器人发起流传输，请参阅 开始向机器人流传输。

开始从应用程序向 Amazon Lex V2 机器人启动流传输时，您可以将该机器人配置为接受用户的音频输入或文本输入。您还可以选择用户在对他们输入的响应中是接收音频还是文本。

如果您已将 Amazon Lex V2 机器人配置为接受用户的音频输入，则它无法接受文本输入。如果您已将机器人配置为接受文本输入，则用户只能通过书面文本与其通信。

当 Amazon Lex V2 机器人接收流式音频输入时，机器人会确定用户何时开始说话以及何时停止说话。它可以处理用户的任意暂停或中断。它还可以在同一个流中接受 DTMF（双音多频）输入和语音输入。这有助于用户更自然地与机器人互动。您可以向用户提供欢迎消息和提示，也可以允许用户中断这些消息和提示。

当您启动双向流时，Amazon Lex V2 使用 HTTP/2 协议。您的应用程序和机器人以一系列事件的形式在单个流中交换数据。事件可以是以下之一：

- 用户的文本、音频或 DTMF 输入。
- 应用程序向 Amazon Lex V2 机器人发送的信号，包括表示消息的音频播放已完成，或者用户已断开会话连接。

有关事件的更多信息，请参阅开始向机器人流传输。有关如何对事件进行编码的信息，请参阅 事件流编码。

主题

开始向机器人流传输

您可以通过 StartConversation 操作在应用程序中启动用户和 Amazon Lex V2 机器人之间的流传输。来自应用程序的 POST 请求会在您的应用程序和 Amazon Lex V2 机器人之间建立连接。这样，您的应用程序和机器人就能够开始通过事件相互交换信息。

只有以下 SDK 支持 `StartConversation` 操作：

- 适用于 C++ 的 AWS SDK
- 适用于 Java V2 的 AWS SDK
- 适用于 JavaScript V3 的 AWS SDK
- 适用于 Ruby V3 的 AWS SDK

您的应用程序必须向 Amazon Lex V2 机器人发送的第一个事件是 ConfigurationEvent. 该事件包含诸如响应类型格式之类的信息。可以在配置事件中使用的参数如下所示：

- responseContentType：确定机器人是通过文字还是语音响应用户输入。
- sessionState：与机器人流式会话相关的信息，例如预先确定的意图或对话状态。
- welcomeMessages：指定在用户与机器人对话开始时为其播放的欢迎消息。这些消息在用户提供输入之前播放。要激活欢迎消息，还须指定 `sessionState` 和 `dialogAction` 参数的值。
- disablePlayback：确定机器人是否应等待客户端的提示后才开始监听来电者的输入。默认情况下，播放处于激活状态，因此该字段的值为 `false`。
- requestAttributes：为请求提供其他信息。

有关如何为上述参数指定值的信息，请参阅 StartConversation 操作的 ConfigurationEvent 数据类型。

机器人与您的应用程序之间的每个流只能有一个配置事件。在您的应用程序发送配置事件后，机器人可以从您的应用程序获取其他通信。

如果您已指定您的用户通过音频与 Amazon Lex V2 机器人通信，则您的应用程序可以在对话期间向该机器人发送以下事件：

- AudioInputEvent：包含一个最大 320 字节的音频块。您的应用程序必须通过多个音频输入事件从服务器向机器人发送消息。该流中的每个音频输入事件都必须具有相同的音频格式。
- DTMFInputEvent：向机器人发送 DTMF 输入。每次 DTMF 按键都对应一个事件。
- PlaybackCompletionEvent：通知服务器已向他们播放了用户输入的响应。如果您要向用户发送音频响应，则必须使用播放完成事件。如果配置事件的 `disablePlayback` 取值为 `true`，则无法使用此功能。
- DisconnectionEvent：通知机器人用户已断开对话连接。

如果您已指定用户通过文本与机器人通信，则您的应用程序可以在对话期间向机器人发送以下事件：

- TextInputEvent：从您的应用程序向机器人发送的文本。每个文本输入事件中最多可包含 512 个字符。
- PlaybackCompletionEvent：通知服务器已向他们播放了用户输入的响应。如果您要向用户播放音频，则必须使用此事件。如果配置事件的 `disablePlayback` 取值为 `true`，则无法使用此功能。
- DisconnectionEvent：通知机器人用户已断开对话连接。

您必须以正确的格式对发送给 Amazon Lex V2 机器人的每个事件进行编码。有关更多信息，请参阅事件流编码。

每个事件都有一个事件 ID。为了帮助解决流中可能出现的任何问题，请为每个输入事件分配一个唯一的事件 ID。然后，您可以通过机器人解决任何处理失败问题。

Amazon Lex V2 还为每个事件使用时间戳。除了事件 ID 之外，您还可以使用这些时间戳来帮助解决任何网络传输问题。

在用户与 Amazon Lex V2 机器人对话期间，机器人可以向用户发送以下出站事件作为响应：

- IntentResultEvent：包含 Amazon Lex V2 根据用户言语确定的意图。每个内部结果事件均包括：
  - inputMode：用户言语的类型。有效值为 `Speech`、`DTMF` 或 `Text`。

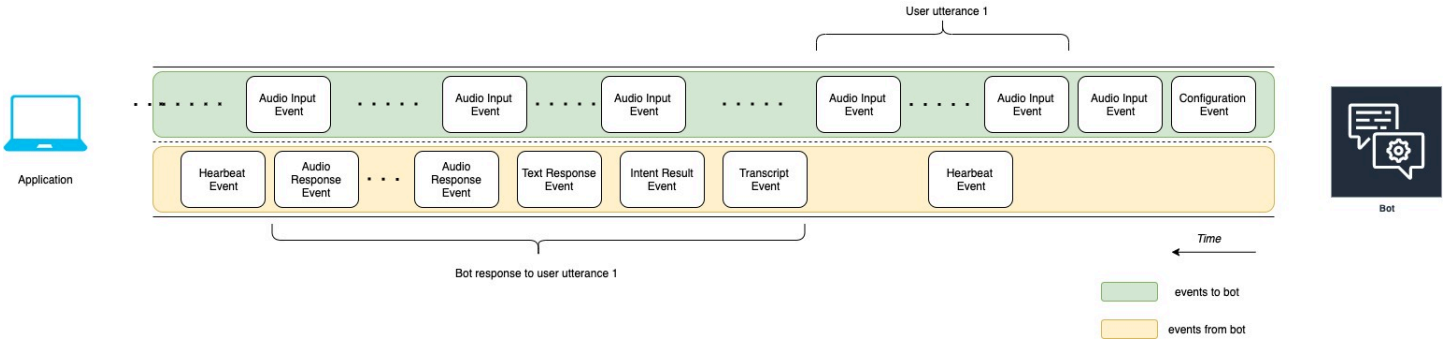
  - interpretations：Amazon Lex V2 根据用户言语确定的解释。
  - requestAttributes：如果您尚未使用 lambda 函数修改请求属性，则这些属性与对话开始时传递的属性相同。
  - sessionId：用于对话的会话标识符。
  - sessionState：用户与 Amazon Lex V2 的会话状态。
- TranscriptEvent：如果用户向您的应用程序提供输入，则此事件包含用户对机器人的言语的转录。如果没有用户输入，您的应用程序不会收到 `TranscriptEvent`。

  发送到应用程序的转录事件的值取决于您是将音频（语音和 DMTF）还是文本指定为对话模式：
  - 语音输入转录：如果用户正在与机器人通话，则该转录事件是用户音频的转录。这是从用户开始说话到他们结束讲话的所有语音的转录。
  - DTMF 输入转录：如果用户通过键盘键入，则该转录事件包含用户在输入中按下的所有数字。
  - 文本输入转录：如果用户提供文本输入，则该转录事件包含用户输入中的所有文本。
- TextResponseEvent：包含文本格式的机器人响应。默认情况下，会返回文本响应。如果您已将 Amazon Lex V2 配置为返回音频响应，则该文本用于生成音频响应。每个文本响应事件都包含机器人返回给用户的消息对象数组。
- AudioResponseEvent：包含根据 `TextResponseEvent` 中生成的文本合成的音频响应。要接收音频响应事件，您必须将 Amazon Lex V2 配置为提供音频响应。所有音频响应事件都具有相同的音频格式。每个事件包含最多 100 字节的音频块。Amazon Lex V2 向您的应用程序发送一个空的音频块，其中 `bytes` 字段设置为 `null`，表示音频响应事件已结束。
- PlaybackInterruptionEvent：当用户中断机器人发送到您的应用程序的响应时，Amazon Lex V2 会触发此事件以停止响应的播放。
- HeartbeatEvent：Amazon Lex V2 会定期发回此事件，以防止您的应用程序与机器人之间的连接超时。

音频对话事件的时序

下列图中展示了用户与 Amazon Lex V2 机器人之间的流式音频对话。应用程序会持续将音频流式传输到机器人，机器人会从该音频中寻找用户输入。在该示例中，用户和机器人都通过语音进行通信。每个图都对应一个用户言语和机器人对该言语的响应。

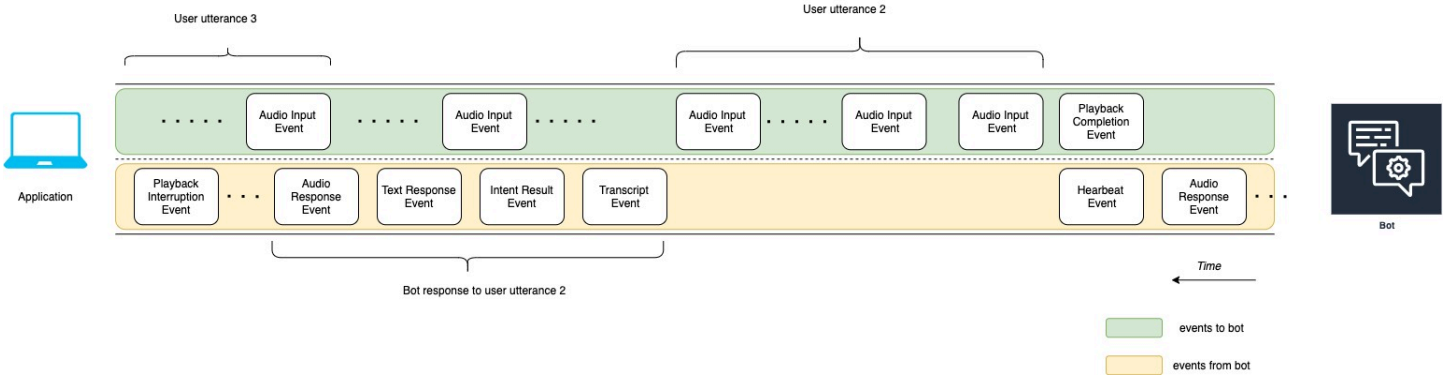
应用程序和机器人之间对话的开始如下图所示。该流传输从零时 (t0) 开始。



上图中的各事件如下所述：

- t0：应用程序向机器人发送配置事件以启动流传输。
- t1：应用程序流传输音频数据。该数据被分解为来自应用程序的一系列输入事件。
- t2：对于用户言语 1，当用户开始说话时，机器人会检测到一个音频输入事件。
- t2：当用户说话时，机器人会发送心跳事件以保持连接。它间歇性发送这些事件，以确保连接不会超时。
- t3：机器人检测到用户言语已结束。
- t4：机器人向应用程序发回一个包含用户语音转录的转录事件。这表示机器人对用户言语 1 的响应的开始。
- t5：机器人发送意图结果事件，以指示用户想要执行的操作。
- t6：机器人开始在文本响应事件中以文本形式提供响应。
- t7：机器人向应用程序发送一系列音频响应事件，以便为用户播放。
- t8：机器人发送另一个心跳事件以间歇性地保持连接。

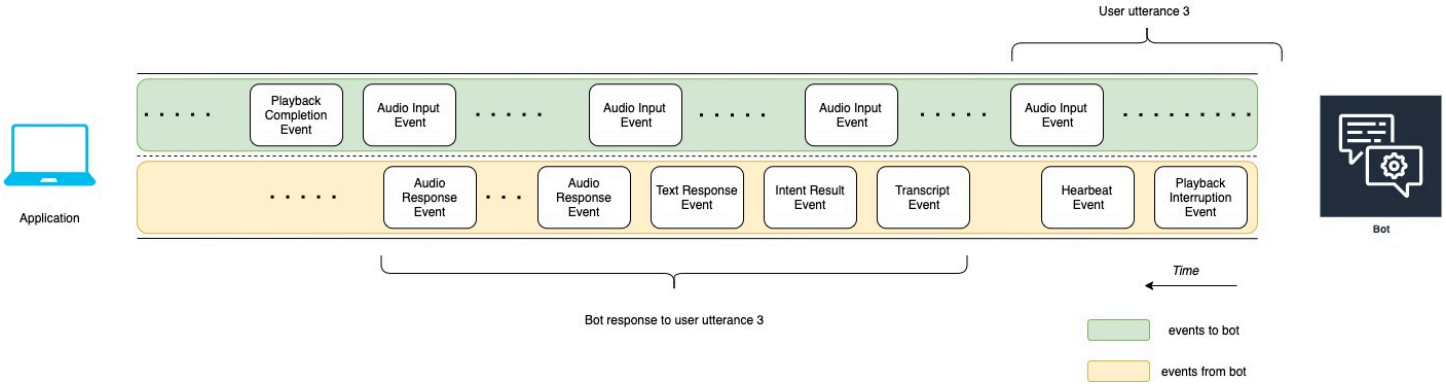
下图是对上图的延续。它显示应用程序向机器人发送播放完成事件，表明它已停止为用户播放音频响应。应用程序向用户播放机器人对用户言语 1 的响应。用户通过用户言语 2 来回应机器人对用户言语 1 的响应。



上图中的各事件如下所述：

- t10：应用程序发送播放完成事件，表示它已完成向用户播放机器人消息。
- t11：应用程序将用户响应作为用户言语 2 发送回机器人。
- t12：关于机器人对用户言语 2 的响应，机器人会等待用户停止说话，然后开始提供音频响应。
- t13：当机器人向应用程序发送机器人对用户言语 2 的响应时，会检测到用户言语 3 的开始。机器人会停止机器人对用户言语 2 的响应，并发送播放中断事件。
- t14：机器人向应用程序发送播放中断事件，表示用户已中断提示。

下图显示了机器人对用户言语 3 的响应，以及机器人响应用户言语后对话仍在继续。



使用 API 开始流传输对话

当您开始向 Amazon Lex V2 机器人发起流传输时，您已完成以下任务：

1. 创建与服务器的初始连接。
2. 配置安全凭证和机器人详细信息。机器人详细信息包括机器人是接受 DTMF 和音频输入，还是文本输入。
3. 向服务器发送事件。这些事件是来自用户的文本数据或音频数据。
4. 处理来自服务器的事件。在此步骤中，您可以确定机器人输出是以文本还是语音形式呈现给用户。

以下代码示例初始化与 Amazon Lex V2 机器人和您的本地计算机的流传输对话。您可以修改这些代码以满足您的需求。

以下代码是通过 AWS SDK for Java 启动机器人连接并配置机器人详细信息和凭证的请求示例。

```
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();
        }
    }
```

```
}
```

以下代码是通过 AWS SDK for Java 向机器人发送事件的请求示例。此示例中的代码表示通过计算机上的麦克风发送音频事件。

```
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();
        }
    }
}
```

以下代码是使用 AWS SDK for Java 处理机器人响应的请求示例。此示例中的代码表示将 Amazon Lex V2 配置为向您播放音频响应。

```
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;
    }

}
```

要将机器人配置为通过音频响应输入事件，您必须先向 Amazon Lex V2 订阅音频事件，然后将机器人配置为对用户的输入事件提供音频响应。

以下代码是向 Amazon Lex V2 订阅音频事件的 AWS SDK for Java 示例。

```
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);
            }
        }
    }
}
```

以下 AWS SDK for Java 示例将 Amazon Lex V2 机器人配置为对输入事件提供音频响应。

```
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();
    }
}
```

事件流编码

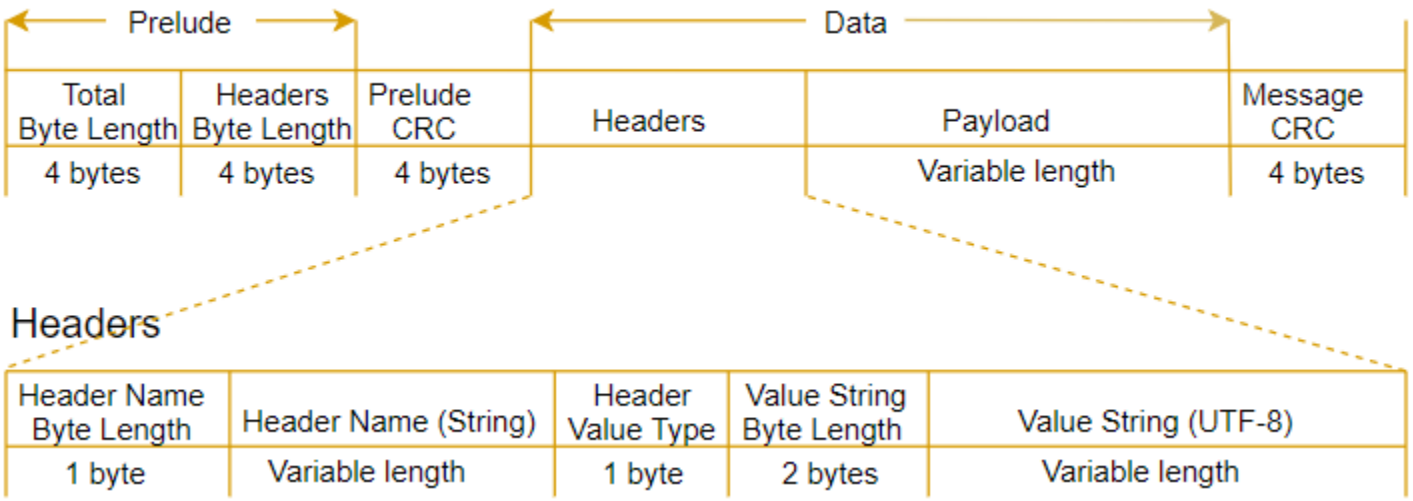
事件流编码在客户端和服务器之间通过消息来提供双向通信。发送到 Amazon Lex V2 流式处理服务的数据帧使用此格式进行编码。来自 Amazon Lex V2 的响应也使用此编码形式。

每个消息都包含两部分：前导信息和数据。前导信息部分包含消息的总字节长度和所有标头的组合字节长度。数据部分包含标头和负载。

每个部分以 4 字节 big-endian 整数 CRC 校验和结尾。消息 CRC 校验和包括前导信息部分和数据部分。Amazon Lex V2 使用 CRC32（通常称为 GZIP CRC32）来计算两个 CRC 结果。有关 CRC32 的更多信息，请参阅 GZIP 文件格式规范版本 4.3。

总消息开销（包括前导信息和两个校验和）为 16 个字节。

下图显示了构成消息和标头的组件。每个消息有多个标头。



每个消息都包含以下组件：

- 前导信息：大小始终固定为 8 字节，由两个 4 字节的字段组成。
  - 第一个 4 字节：总字节长度。这是整个消息的 big-endian 整数字节长度，包括 4 字节长度字段本身。
  - 第二个 4 字节：标头字节长度。这是消息的标头部分的 big-endian 整数字节长度，不包括标头长度字段本身。
- 前导信息 CRC：消息的前导信息部分的 4 字节 CRC 校验和，不包括 CRC 本身。前导信息具有不同于消息 CRC 的 CRC，以确保 Amazon Lex V2 可以立即检测到损坏的字节长度信息，而不会导致缓冲区溢出之类的错误。
- 标头：用于批注消息（如消息类型、内容类型等）的元数据。消息有多个标头。标头是一些键值对，其中的键为 UTF-8 字符串。标头可按任何顺序出现在消息的标头部分中，并且任何给定标头只能出现一次。对于必需的标头类型，请参阅以下部分。
- 负载：发送到 Amazon Lex 的音频或文本内容。
- 消息 CRC：从消息开头到校验和开头的 4 字节 CRC 校验和。该消息中包含除 CRC 本身之外的所有内容。

每个标头都包含以下组件。每个帧有多个标头。

- 标头名称字节长度：标头名称的字节长度。
- 标头名称：指示标头类型的标头名称。有关有效值，请参阅下面的帧描述。
- 标头值类型：指示标头值类型的枚举。
- 值字符串字节长度：标头值字符串的字节长度。

- 标头值：标头字符串的值。此字段的有效值取决于标头的类型。有关有效值，请参阅下面的帧描述。

允许用户打断机器人

当您在 Amazon Lex V2 机器人与您的应用程序之间启动双向音频流时，您可以将机器人配置为在返回提示时监听用户输入。这样，用户就可以在机器人完成播放之前打断提示音。如果用户可能已经知道问题答案，例如当系统提示他们提供 CVV 代码时，您可以使用此配置。

当机器人在您的应用程序可以返回 `PlaybackCompletion` 事件之前检测到用户输入时，它就会知道用户何时中断提示。当用户打断机器人时，机器人会发送 `PlaybackInterruptionEvent`.

默认情况下，用户可以中断机器人正在向您的应用程序流传输的任何提示。您可以通过 Amazon Lex V2 控制台更改此设置。

您可以通过编辑插槽来更改用户对提示的响应方式。插槽是意图的一部分，也是用户向您提供所需信息的手段。每个插槽都会提示用户向您提供该信息。要了解有关插槽的更多信息，请参阅 工作原理。

更改用户是否可以中断提示（控制台）

1. 登录 AWS Management Console 并进入 Amazon Lex V2 控制台。
2. 从机器人下选择一个机器人。
3. 从语言下选择机器人的语言。
4. 选择查看意图。
5. 选择意图。
6. 从插槽列中选择一个插槽。
7. 在高级选项下，选择插槽提示。
8. 选择更多提示选项。
9. 选择或取消选择用户可以在读取提示时打断提示音。

您可以通过创建具有两个插槽的机器人并指定用户不能打断一个插槽的提示来测试此功能。如果您打断了可中断的提示，机器人会发送播放中断事件。如果您打断了不可中断的提示，则会继续播放该提示。

允许机器人等待用户提供更多信息

启动从 Amazon Lex V2 机器人到应用程序的双向流时，您可以配置该机器人等待用户提供更多信息。在某些情况下，用户可能还未准备好回应提示。例如，用户可能还未准备好提供信用卡信息，因为他们的钱包在另一个房间里。

通过 Amazon Lex V2 机器人的等待并继续行为，用户可以说出诸如“稍等片刻”之类的短语，让机器人等待他们找到信息并提供信息。启用此行为后，机器人会定期向用户发送提醒，要求其提供信息。因为没有用户言语可供其转录，它不会返回转录事件。

Amazon Lex V2 机器人会自动管理流传输对话。您无需编写任何其它代码即可启用此功能。当用户提示机器人等待时，`Intent` 的 `state` 为 `Waiting`，`DialogAction` 的 `type` 为 `ElicitSlot`。您可以通过用这些信息来帮助根据需要自定义应用程序。例如，您可以将应用程序配置为在用户寻找信用卡时播放音乐。

您可以为单个插槽启用“等待并继续”行为。要了解有关插槽的更多信息，请参阅 工作原理。

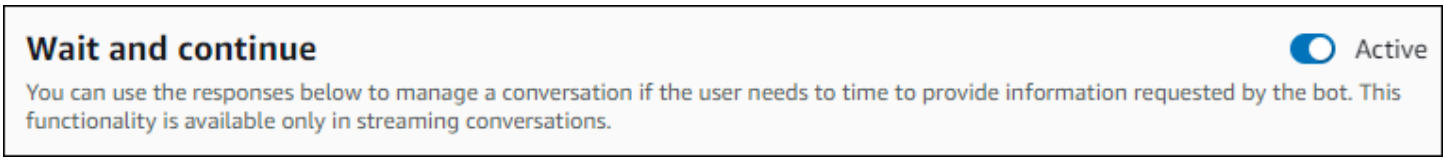
启用“等待并继续”

1. 登录 AWS Management Console 并进入 Amazon Lex V2 控制台。
2. 从机器人下选择一个机器人。
3. 从语言下选择机器人的语言。
4. 选择查看意图。
5. 选择意图。
6. 从插槽下选择一个插槽。
7. 在高级选项下，选择等待并继续。
8. 在等待并继续下指定以下字段：
   - 用户希望机器人等待时的响应：用户要求机器人等待其他信息时的机器人响应方式。
   - 用户需要机器人继续等待时的响应：机器人发送的响应，用于提醒用户它仍在等待信息。您可以更改机器人提醒用户的频率。
   - 用户想要继续时的响应：用户获得所请求的信息时机器人的响应。

对于每个机器人响应，您可以给出响应的多种变体，然后随机向用户呈现一个变体。您还可以选择用户是否可以中断这些响应。

要测试此等待并继续功能，请将您的机器人配置为等待用户输入，然后向 Amazon Lex V2 机器人开始流传输。有关向机器人流传输的信息，请参阅 使用 API 开始流传输对话。

您可能需要关闭等待并继续响应。通过活动开关来设置是否使用“等待并继续”响应。



配置履行进度更新

调用意图的履行 Lambda 函数时，机器人要等到函数完成后才会发送响应。如果 Lambda 函数需要超过几秒钟才能完成，则用户可能会认为机器人没有响应。要解决这个问题，您可以将机器人配置为在履行 Lambda 函数运行时向用户发送更新，以便用户知道机器人仍在处理他们的请求。

当您向某个意图添加履行更新时，机器人会在履行开始时做出响应，并在履行过程中定期做出响应。配置开始响应时，可以指定机器人发送响应之前的延迟。这样，您就可以支持无法快速完成该履行的情况。配置更新响应时，需要指定发送更新的频率。您还可以配置超时以限制履行函数必须运行的时间。

您还可以向机器人添加履行的响应。这样，机器人就能够根据履行成功、失败或超时的结果，发送不同的响应。

仅当通过 StartConversation 操作与机器人互动时，才会进行履行更新。通过 StartConversation、RecognizeText 和 RecognizeUtterance 操作与机器人互动时，可以进行履行后更新。

履行更新

履行更新是在您的 Lambda 函数履行意图时发送的。当开启履行更新时，您将提供在履行开始时发送的开始响应和在履行过程中定期发送的更新响应。

当您指定更新响应时，您还可以指定超时时间，决定该履行函数可以运行多长时间。您可以将超时时长指定为最多 15 分钟（900 秒）。

如果您通过在控制台将 `active` 设置为 false 或执行 CreateIntent 或 UpdateIntent 操作来关闭履行更新，则不会使用为履行更新指定的超时时长，而是使用默认的 30 秒超时时长。

如果履行函数超时，Amazon Lex V2 会执行以下三项操作之一：

- 履行后响应已配置并处于活动状态：返回超时响应。

- 履行后响应已配置但未激活：返回异常。
- 未配置履行后响应：返回异常。

开始响应

在流式对话期间调用履行 Lambda 函数时，Amazon Lex V2 会返回开始响应。它通常会告诉用户实现意图需要一些时间，并建议用户等待。执行 `RecognizeText` 或 `RecognizeUtterance` 操作时，不会返回开始响应。

您最多可以指定五个响应消息。Amazon Lex V2 会选择向用户播放其中一条消息。

您可以在 Lambda 函数被调用和返回开始响应之间配置延迟。如果 Lambda 函数在延迟结束之前完成其工作，则不会返回开始响应。

您可以通过控制台中的 `active` 切换开关或 FulfillmentUpdatesSpecification 结构来开启和关闭开始响应。如果 `active` 为 false，则不播放开始响应。

更新响应

在履行 Lambda 函数运行期间，Amazon Lex 会在流式对话期间定期返回更新响应。当您执行 `RecognizeText` 或 `RecognizeUtterance` 操作时，不会播放更新响应。您可以配置更新响应的播放频率。例如，在履行函数运行期间，您可以每 30 秒播放一次更新响应，让用户知道流程正在运行，应继续等待。

您最多可以指定五个更新消息。Amazon Lex V2 选择要向用户播放的消息。使用多条消息可以防止更新重复。

如果用户在履行 Lambda 函数运行时通过语音、DTMF 或文本提供输入，则 Amazon Lex V2 会将更新响应返回给用户。

如果 Lambda 函数在第一个更新周期结束之前完成其工作，则不会返回更新响应。

您可以通过控制台中的 `active` 切换开关或 FullmentUpdatesSpecification 结构来开启和关闭更新响应。如果 `active` 为 false，则不返回更新响应。

履行后响应

当履行函数结束时，Amazon Lex V2 会返回履行后响应。实现任何意图时都可以使用履行后响应，而不仅仅是在流式对话时。履行后响应会通知用户该函数已完成以及完成结果。

您可以通过控制台中的 `active` 切换开关或 PostFulfillmentStatusSpecification 结构来开启和关闭履行后响应。如果 `active` 为 false，则不播放响应。

有三种类型的履行后响应：

- 成功：当履行 Lambda 函数成功完成其工作时返回。如果履行后响应未激活，Amazon Lex V2 会执行下一个配置的操作。
- 超时：当 Lambda 函数在配置的超时时间过后未完成其工作时返回。如果履行后响应未激活，Amazon Lex V2 将返回异常。
- 失败：当 Lambda 函数在响应中返回状态 `Failed` 或 Amazon Lex V2 在实现意图时遇到错误时返回。如果履行后响应未激活，Amazon Lex V2 将返回异常。

您最多可以为每种类型指定五个消息。Amazon Lex V2 会选择向用户播放其中一条消息。

与履行开始和履行更新响应不同，履行后响应可以为流式和非流式对话回放。

您还可以通过配置 Lambda 函数返回履行后消息来覆盖这些消息。

> **Note**
>
> 如果该意图有结束响应，则会在履行后响应之后返回。

履行后示例

为了更好地了解履行后响应，我们以一个 *`BookTrip`* 机器人为例。该机器人是为帮助计划行程而创建的，具有 *`BookFlight`* 意图，并配置了履行 Lambda 函数，可以为客户预订航班。引发 *`BookFlight`* 的插槽后，Amazon Lex V2 就会调用履行 Lambda 函数。在此履行过程中，可能会出现以下三种结果之一：

- 成功：航班成功预订。
- 超时：预订过程所花费的时间比配置的履行 Lambda 执行时间更长（例如，无法在分配的时间内联系到航空公司）。
- 失败：由于其他原因预订失败。

出现任何结果，您都可以通过履行后响应为客户提供更有意义的响应。每种结果的示例如下：

- 成功响应：“我们成功为您预订了机票，并已向您发送了一封确认电子邮件。如果您有任何疑问，请随时通过该电子邮件中提供的联系信息与我们联系。”
- 超时响应：“由于我们系统繁忙，预订机票所需的时间比预期更长。我们已将您的请求放入队列，并通过电子邮件向您发送了与此请求对应的参考编号。一旦完成机票预订，我们将向您发送预订确认函。如果您有任何疑问，请随时通过该电子邮件中提供的联系信息与我们联系。”

> **Note**
>
> 如果您未配置超时消息，Amazon Lex 会抛出与用例对应的 4XX 错误。

- 失败响应：“很遗憾，我们无法为您预订机票。我们已经通过电子邮件向您发送了帮您预订机票时遇到的问题的详细信息。”

配置捕获用户输入的超时

Amazon Lex V2 流式处理 API 使机器人能够自动检测用户输入中的言语。在创建意图或插槽时，您可以配置言语的各个方面，例如言语的最大持续时间、等待用户输入时的超时或 DTMF 输入的结束字符。您可以根据自己的用例自定义机器人的行为。例如，您可以将信用卡号的位数限制为 16。

您还可以在开始与机器人对话时通过会话属性配置超时，并在必要时在 Lambda 函数中将其覆盖。

属性的配置密钥使用以下语法：

```
x-amz-lex:<InputType>:<BehaviorName>:<IntentName>:<SlotName>
```

`InputType` 可以是 **`audio`**、**`dtmf`** 或 **`text`**。

您可以通过指定 **`*`** 为意图或插槽名称来配置机器人中所有意图或插槽的默认设置。任何特定意图或插槽的设置优先于默认设置。

Amazon Lex V2 提供了预定义的会话属性，用于管理 StartConversation 操作处理对机器人进行文本、语音或 DTMF 输入的方式。所有预定义属性都在 `x-amz-lex` 命名空间中。

您可以将 `*` 指定为意图或插槽名称，为机器人中的所有意图、插槽或子插槽配置默认设置。任何特定意图或插槽的设置优先于默认设置。对以下所有超时使用这些模式。

对于复合插槽的子插槽，您可以通过 **`.`** 分隔，例如：

```
<slotName>.<subSlotName>
```

```
x-amz-lex:allow-interrupt:<intentName>:<slotName>.<subSlotName>
```

表达式	场景
Intent:Slot.SubSlot	仅适用于名为“Slot”的复合插槽中名为“SubSlot”的子插槽
Intent:Slot.*	适用于名为“Slot”的复合插槽中的任何子插槽
Intent:*.SubSlot	仅适用于任何复合插槽中名为“SubSlot”的子插槽
Intent:*.*	适用于任何复合插槽中的任何子插槽

中断行为

您可以为机器人设置中断行为。该属性通过 Amazon Lex V2 定义。

允许中断

```
x-amz-lex:allow-interrupt:<intentName>:<slotName>
```

定义用户是否可以中断 Amazon Lex V2 机器人播放的提示音。您可以选择性地将其关闭。

默认值：True

语音输入超时

您可以通过会话属性为与机器人进行语音交互设置超时值。这些属性通过 Amazon Lex V2 定义。通过这些属性，您可以指定 Amazon Lex V2 在收集输入语音之前等待客户完成讲话的时间。

所有这些属性都在 `x-amz-lex:audio` 命名空间中。

最大言语长度

```
x-amz-lex:audio:max-length-ms:<intentName>:<slotName>
```

定义 Amazon Lex V2 在语音输入被截断且将语音返回到您的应用程序之前所等待的时长。您可以在期望较长的响应或希望为客户留出更多的时间来提供信息时，增加该输入长度。

默认值：13000 毫秒（13 秒）。最大值为 15000 毫秒（15 秒）。

如果您将该 max-length-ms 属性设置为超过 15000 毫秒，则该值将默认为 15000 毫秒。

语音超时

```
x-amz-lex:audio:start-timeout-ms:<intentName>:<slotName>
```

假设客户停止说话之前机器人要等多长时间。如果您希望让客户在发言前有更多时间查找或回想信息，则可以增加该时间。例如，您可能希望为客户提供时间取出信用卡，以便输入该信用卡号码。

默认值：4000 毫秒（4 秒）。

静默超时

```
x-amz-lex:audio:end-timeout-ms:<intentName>:<slotName>
```

机器人在客户停止讲话后要等多长时间，才能假设言语已经结束。如果预计在提供输入时会有一段静默时间，则可以增加该时间。

默认值：600 毫秒（0.6 秒）

允许音频输入

```
x-amz-lex:allow-audio-input:<intentName>:<slotName>
```

您可以启用此属性，以便机器人仅通过音频模式接受用户输入。如果此标志设置为 false，则机器人不接受音频输入。默认情况下，该值设置为 true。

默认值：True

文本输入超时

通过以下会话属性来指定您的机器人在文本对话模式下的行为方式。

该属性位于 x-amz-lex:text 命名空间中。

启动超时阈值

```
x-amz-lex:text:start-timeout-ms:<intentName>:<slotName>
```

机器人要等多长时间才会重新提示客户输入文本。如果您希望让客户在提供文本输入前有更多时间查找或回想信息，则可以增加该时间。例如，您可能希望为客户提供更多时间查找订单详情。或者，您可以降低阈值以更早地提示客户。

默认值：30000 毫秒（30 秒）。

DTMF 输入配置

通过以下会话属性来指定 Amazon Lex V2 机器人在音频对话时如何响应 DTMF 输入。

所有这些属性都在 `x-amz-lex:dtmf` 命名空间中。

删除字符

```
x-amz-lex:dtmf:deletion-character:<intentName>:<slotName>
```

清除累积的 DTMF 数字并立即结束输入的 DTMF 字符。

默认值：*

结尾字符

```
x-amz-lex:dtmf:end-character:<intentName>:<slotName>
```

立即结束输入的 DTMF 字符。如果用户没有按下此字符，则输入将在结束超时后结束。

默认值：#

结束超时

```
x-amz-lex:dtmf:end-timeout-ms:<intentName>:<slotName>
```

机器人应在最后一次 DTMF 字符输入后要等多长时间才能假设输入已结束。

默认值：5000 毫秒（5 秒）。

每句言语的最大 DTMF 位数

```
x-amz-lex:dtmf:max-length:<intentName>:<slotName>
```

每句言语中允许的最大 DTMF 位数。例如，您可以将此值设置为 16，以限制可以为信用卡号输入的字符数。此值无法增加。

默认值：1024 个字符

允许 DTMF 输入

您可以通过会话属性设置机器人可以接受的输入类型。这些属性通过 Amazon Lex V2 定义。

```
x-amz-lex:allow-dtmf-input:<intentName>:<slotName>
```

您可以启用该属性，以便机器人通过 DTMF 模式接受用户输入。如果此标志设置为 false，则机器人不会接受 DTMF 输入。默认情况下，该值设置为 true。

默认值：True

导入和导出

您可以导出机器人定义、机器人区域设置或自定义词汇，然后将其导回以创建新资源或覆盖 AWS 账户中的现有资源。例如，您可以从测试账户导出机器人，然后在生产账户中创建该机器人的副本。您也可以将机器人从一个 AWS 区域复制到另一区域。

在将导出的资源导入之前，您可以对其进行更改。例如，您可以导出一个机器人，然后编辑某个插槽的 JSON 文件，以便在特定插槽中添加或移除插槽值引发言语。编辑完定义后，可以导入修改后的文件。

主题

导出

您可以通过控制台或 `CreatExport` 操作导出机器人、机器人区域设置或自定义词汇。您可以指定要导出的资源，也可以提供一个可选的密码来帮助在开始导出时保护 .zip 文件。下载该 .zip 文件后，必须使用密码访问该文件，然后才能使用。有关更多信息，请参阅导入或导出时使用密码。

导出是一个异步操作。开始导出后，您可以通过控制台或 `DescribeExport` 操作来监控导出进度。导出完成后，控制台或 `DescribeExport` 操作的状态将显示为 `COMPLETED`，控制台会将导出的 .zip 文件下载到您的浏览器。如果您使用 `DescribeExport` 操作，Amazon Lex V2 会提供一个预签名的 Amazon S3 URL，您可以在其中下载导出结果。该下载 URL 仅在五分钟内可用，但您可以通过再次调用该 `DescribeExport` 操作来获取新的 URL。

您可以通过控制台或 `ListExports` 操作查看资源的导出历史记录。结果显示了导出文件及其当前状态。历史记录中的导出有效期为七天。

当您导出机器人或机器人区域设置的 `Draft` 版本时，JSON 文件中的定义可能会处于不一致的状态，因为在导出过程中，机器人或机器人区域设置的 `Draft` 版本可能会发生变化。如果在导出 `Draft` 版本时对其进行了更改，则这些更改可能不会包含在导出文件中。

当您导出机器人区域设置时，Amazon Lex 会导出定义该区域的所有信息，包括区域设置、自定义词汇、意图、插槽类型和插槽。

当您导出机器人时，Amazon Lex 会导出为该机器人定义的所有区域设置，包括意图、插槽类型和插槽。以下内容不随机器人导出：

- 机器人别名
- 与机器人关联的角色 ARN
- 与机器人和机器人别名相关的标签
- 与机器人别名关联的 Lambda 代码挂钩

在导入机器人时，角色 ARN 和标签将作为请求参数输入。如有必要，您需要在导入后创建机器人别名并分配 Lambda 代码挂钩。

您可以通过控制台或 `DeleteExport` 操作删除导出文件和关联的 .zip 文件。

有关通过控制台导出机器人的示例，请参阅 导出机器人（控制台）。

导出所需的 IAM 权限

要导出机器人、机器人区域设置和自定义词汇，运行导出的用户必须具有以下 IAM 权限。

API	• 所需的 IAM 操作	资源
CreateExport	• CreateExport	机器人
UpdateExport	• UpdateExport	机器人
DescribeExport	• DescribeExport • DescribeBot • DescribeCustomVocabulary • DescribeLocale • DescribeIntent • DescribeSlot • DescribeSlotType • ListLocale • ListIntent • ListSlot • ListSlotType	机器人

API	• 所需的 IAM 操作	资源
用于自定义词汇的 DescribeExport	• DescribeExport • DescribeCustomVocabulary	bot
DeleteExport	• DeleteExport	机器人
ListExports	• ListExports	*

有关 IAM policy 示例，请参阅 允许用户导出机器人和机器人区域设置 。

导出机器人（控制台）

您可以从机器人列表、版本列表或版本详细信息页面导出机器人。当您选择一个版本时，Amazon Lex V2 会导出该版本。假设从机器人列表中导出机器人，步骤如下。如果您选择从一个版本开始导出，步骤是相同的。

通过控制台导出机器人

1. 登录 AWS Management Console，并通过以下网址打开 Amazon Lex V2 控制台：https://console.aws.amazon.com/lexv2/home。
2. 从机器人列表中选择要导出的机器人。
3. 从操作中选择导出。
4. 选择机器人版本、平台和导出形式。
5. （可选）输入 .zip 文件的密码。提供密码有助于保护输出存档。
6. 选择导出。

开始导出后，会返回到机器人列表。要监控导出进度，请使用导入/导出历史记录列表。当导出状态为完成时，控制台会自动将 .zip 文件下载到您的计算机。

要再次下载导出文件，请在导入/导出列表中选择导出，然后选择下载。您可以为下载的 .zip 文件提供密码。

导出机器人语言

1. 登录 AWS Management Console，并通过以下网址打开 Amazon Lex V2 控制台：https://console.aws.amazon.com/lexv2/home。

2. 从机器人列表中，选择要导出语言的机器人。
3. 在添加语言中，选择查看语言。
4. 从所有语言列表中，选择要导出的语言。
5. 从操作中选择导出。
6. 选择机器人的版本、平台和格式。
7. （可选）输入 .zip 文件的密码。提供密码有助于保护输出存档。
8. 选择导出。

开始导出后，会返回到语言列表。要监控导出进度，请使用导入/导出历史记录列表。当导出状态为完成时，控制台会自动将 .zip 文件下载到您的计算机。

要再次下载导出文件，请在导入/导出列表中选择导出，然后选择下载。您可以为下载的 .zip 文件提供密码。

导入

要通过控制台导入之前导出的机器人、机器人区域设置或自定义词汇，您需要提供本地计算机上的文件位置以及用于解锁该文件的可选密码。有关示例，请参阅导入机器人（控制台）。

通过 API 导入资源的过程分为三个步骤：

1. 通过 `CreateUploadUrl` 操作创建上传 URL。在使用控制台时，无需创建上传 URL。
2. 上传包含资源定义的 .zip 文件。
3. 通过 `StartImport` 操作开始导入。

上传 URL 是预签名 Amazon S3 URL，具有写入权限。该 URL 在生成后五分钟内可用。如果使用密码保护 .zip 文件，则必须在开始导入时提供密码。有关更多信息，请参阅导入或导出时使用密码。

导入是一个异步过程。您可以通过控制台或 `DescribeImport` 操作监控导入的进度。

当您导入机器人或机器人区域设置时，导入文件中的资源名称与 Amazon Lex V2 中现有资源的名称之间可能存在冲突。Amazon Lex V2 可以通过三种方式处理该冲突：

- 冲突时失败：导入停止，并且未从该 .zip 导入文件中导入任何资源。
- 覆盖：Amazon Lex V2 从该 .zip 导入文件中导入所有资源，并使用导入文件中的定义替换任何现有资源。

- 追加：Amazon Lex V2 从该 .zip 导入文件中导入所有资源，并使用导入文件中的定义将其添加到任何现有资源中。这仅适用于机器人区域设置。

您可以通过控制台或 `ListImports` 操作查看资源导入列表。导入内容在列表中保留七天。您可以通过控制台或 `DescribeImport` 操作来查看有关特定导入的详细信息。

您也可以通过控制台或 `DeleteImport` 操作删除导入内容和关联的 .zip 文件。

有关通过控制台导入机器人的示例，请参阅 导入机器人（控制台）。

导入所需的 IAM 权限

要导入机器人、机器人区域设置和自定义词汇，运行导入的用户必须具有以下 IAM 权限。

API	所需的 IAM 操作	资源
CreateUploadUrl	• CreateUploadUrl	*
用于机器人和机器人区域设置的 StartImport	• StartImport • iam:PassRole • CreateBot • CreateCustomVocabulary • CreateLocale • CreateIntent • CreateSlot • CreateSlotType • UpdateBot • UpdateCustomVocabulary • UpdateLocale • UpdateIntent • UpdateSlot • UpdateSlotType • DeleteBot • DeleteCustomVocabulary • DeleteLocale	1. 要导入新的机器人：机器人、机器人别名。 2. 要覆盖现有的机器人：机器人。 3. 要导入新的区域设置：机器人。

API	所需的 IAM 操作	资源
	• DeleteIntent • DeleteSlot • DeleteSlotType	
用于自定义词汇的 StartImport	• StartImport • CreateCustomVocabulary • DeleteCustomVocabulary • UpdateCustomVocabulary	bot
DescribeImport	• DescribeImport	机器人
DeleteImport	• DeleteImport	机器人
ListImports	• ListImports	*

有关 IAM policy 示例，请参阅 允许用户导入机器人和机器人区域设置 。

导入机器人（控制台）

通过控制台导入机器人

1. 登录 AWS Management Console，并通过以下网址打开 Amazon Lex V2 控制台：https://console.aws.amazon.com/lexv2/home。
2. 从操作中选择导入。
3. 在输入文件中，为机器人命名，然后选择包含定义机器人的 JSON 文件的 .zip 文件。
4. 如果 .zip 文件受密码保护，请输入 .zip 文件的密码。虽然密码保护存档并非必选，但它有助于保护内容的安全性。
5. 创建或输入定义机器人权限的 IAM 角色。
6. 表明机器人是否符合儿童在线隐私保护法 (COPPA) 的要求。
7. 为机器人提供空闲超时设置。如果您未提供具体值，则使用 .zip 文件中的值。如果 .zip 文件不包含超时设置，则 Amazon Lex V2 会使用默认值 300 秒（5 分钟）。
8. （可选）为机器人添加标签。

9. 选择是否在覆盖现有同名机器人时发出警告。如果您启用警告，并且要导入的机器人会覆盖现有的机器人，则会收到警告，并且不会导入该机器人。如果您禁用警告，则导入的机器人将替换同名的现有机器人。
10. 选择导入。

开始导入后，会返回到机器人列表。要监控导入进度，请使用导入/导出历史记录列表。当导入状态为完成时，您可以从机器人列表中选择机器人来修改或构建机器人。

导入机器人语言

1. 登录 AWS Management Console，并通过以下网址打开 Amazon Lex V2 控制台：https://console.aws.amazon.com/lexv2/home。
2. 从机器人列表中，选择要导入语言的机器人。
3. 在添加语言中，选择查看语言。
4. 从操作中选择导入。
5. 在输入文件中，选择包含要导入的语言的文件。要对 .zip 文件进行保护，请在密码中提供密码。
6. 在语言中，选择要导入的语言。语言不必与导入文件中的语言相匹配。您可以将意图从一种语言复制到另一种语言。
7. 在语音中，选择 Amazon Polly 语音用于语音交互，或者为纯文本机器人选择无。
8. 在置信度分数阈值中，输入 Amazon Lex V2 在返回替代意图时插入 `AMAZON.FallbackIntent`、`AMAZON.KendraSearchIntent` 或两者的阈值。
9. 选择是否对覆盖现有语言发出警告。如果启用警告，并且要导入的语言会覆盖现有语言，则会收到警告，并且不会导入该语言。如果禁用警告，则导入的语言将替换现有语言。
10. 选择导入开始导入该语言。

开始导入后，会返回到语言列表。要监控导入进度，请使用导入/导出历史记录列表。当导入状态为完成时，您可以从机器人列表中选择语言来修改或构建机器人。

导入或导出时使用密码

Amazon Lex V2 可以用密码保护您的导出存档，或者通过标准的 .zip 文件压缩功能读取受保护的导入存档。您应始保持对导入和导出的存档进行密码保护。

Amazon Lex V2 会将您导出的存档发送到 S3 存储桶，您可以通过预签名的 S3 URL 访问该存档。该 URL 仅在五分钟内可用。任何有权访问该下载 URL 的人都可以访问该存档。要保护存档中的

数据，请在导出资源时提供密码。如果您需要在 URL 过期后获取该存档，则可以通过控制台或 DescribeExport 操作来获取新的 URL。

如果您丢失了导出存档的密码，则可以通过从导入/导出历史记录表中选择下载或通过 UpdateExport 操作为现有文件创建新密码。如果您在历史记录表中选择下载进行导出，但没有提供密码，则 Amazon Lex V2 会下载一个不受保护的 .zip 文件。

用于导入和导出的 JSON 格式

您可以根据包含描述资源各部分的 JSON 结构的 .zip 文件从 Amazon Lex V2 中导入和导出机器人、机器人区域设置或自定义词汇。当您导出资源时，Amazon Lex V2 会创建 .zip 文件，并通过 Amazon S3 的预签名 URL 将其提供给您。导入资源时，必须创建一个包含 JSON 结构的 .zip 文件并将其上传到 S3 预签名 URL。

当您导出机器人时，Amazon Lex 会在 .zip 文件中创建以下目录结构。导出机器人区域设置时，仅导出该区域设置下的结构。导出自定义词汇时，仅导出自定义词汇下的结构。

```
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
                        ...
```

清单文件结构

清单文件包含导出文件的元数据。

```
{
    "metadata": {
        "schemaVersion": "1.0",
        "fileFormat": "LexJson",
        "resourceType": "Bot | BotLocale | CustomVocabulary"
    }
}
```

机器人文件结构

机器人文件包含机器人的配置信息。

```
{
    "name": "BotName",
    "identifier": "identifier",
    "version": "number",
    "description": "description",
    "dataPrivacy": {
        "childDirected": true | false
    },
    "idleSessionTTLInSeconds": seconds
}
```

机器人区域设置文件结构

机器人区域设置文件包含对机器人的区域或语言的描述。导出机器人时，.zip 文件中可以包含多个机器人区域设置文件。导出机器人区域设置时，该 .zip 文件中只有一个区域设置。

```
{
    "name": "locale name",
    "identifier": "locale ID",
    "version": "number",
```

```
    "description": "description",
    "voiceSettings": {
        "voiceId": "voice",
        "engine": "standard | neural
    },
    "nluConfidenceThreshold": number
}
```

意图文件结构

意图文件包含意图配置信息。.zip 文件中针对特定区域设置的每个意图都有一个意图文件。

BookTrip 机器人中 BookCar 意图的 JSON 结构示例如下。有关意图的 JSON 结构的完整示例，请参阅 CreateIntent 操作。

```
{
    "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
        }
    ]
}
```

插槽文件结构

插槽文件包含意图中插槽的配置信息。.zip 文件中包含一个插槽文件，用于在特定区域设置中为某个意图定义的每个插槽。

插槽的 JSON 结构允许客户在 BookTrip 示例机器人的 BookCar 意图中选择他们想要租的汽车类型，示例如下。有关插槽的 JSON 结构的完整示例，请参阅 CreateSlot 操作。

```
{
    "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,
    }
}
```

复合插槽的 JSON 结构示例如下。

```
{
  "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)"
  }
}
```

插槽类型文件结构

插槽类型文件中包含在某种语言或区域设置中使用的自定义插槽类型的配置信息。对于特定区域设置中的每种自定义插槽类型，.zip 文件中都包含一个插槽类型文件。

以下是插槽类型的 JSON 结构，其中列出了 BookTrip 示例机器人中可用的汽车类型。有关插槽类型的 JSON 结构的完整示例，请参阅 CreateSlotType 操作。

```
{
    "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
    }
}
```

复合插槽类型的 JSON 结构示例如下。

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

以下插槽类型通过自定义语法来理解客户言语。有关更多信息，请参阅语法插槽类型。

```
{
  "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"
      }
```

```
    }
  }
}
```

自定义词汇文件结构

自定义词汇文件中包含单一语言或区域设置的自定义词汇中的条目。在 .zip 文件中，每个具有自定义词汇的区域设置都有一个自定义词汇文件。

用于机器人接收餐厅订单的自定义词汇文件如下。机器人中每个区域设置都有一个文件。

```
{
    "customVocabularyItems": [
        {
            "weight": 3,
            "phrase": "wafers"
        },
        {
            "weight": null,
            "phrase": "extra large"
        },
        {
            "weight": null,
            "phrase": "cremini mushroom soup"
        },
        {
            "weight": null,
            "phrase": "ramen"
        },
        {
            "weight": null,
            "phrase": "orzo"
        }
    ]
}
```

为资源添加标签

为了帮助您管理 Amazon Lex V2 机器人以及机器人别名，您可以将元数据作为标签分配给每个资源。标签是为 AWS 资源分配的标记。每个标签均包含一个键和一个值。

标签让您能够以不同方式（例如，按用途、所有者或应用程序）对 AWS 资源进行分类。标签帮助您：

- 标识和整理您的 AWS 资源。许多 AWS 资源支持标记，因此，您可以将同一标签分配给不同服务中的资源，以指示这些资源是相同的。例如，您可以使用相同标签标记机器人及其使用的 Lambda 函数。
- 分配成本。您可以在 AWS Billing and Cost Management 控制面板上激活标签。AWS 使用标签对您的成本进行分类，并向您提供每月成本分配报告。对于 Amazon Lex V2，您可以通过别名特定的标签为每个别名分配成本。有关更多信息，请参阅 AWS Billing and Cost Management 用户指南中的使用成本分配标签。
- 控制对资源的访问。您可以在 Amazon Lex V2 中使用标签创建策略来控制对 Amazon Lex V2 资源的访问权限。这些策略可以附加到 IAM 角色或用户，以启用基于标签的访问控制。

可以通过 AWS Management Console、AWS Command Line Interface 或 Amazon Lex V2 API 处理标签。

标记您的资源

如果您使用的是 Amazon Lex V2 控制台，则可以在创建资源时标记资源，也可以稍后添加标记。您还可以使用控制台来更新或删除现有标签。

如果您使用 AWS CLI 或 Amazon Lex V2 API，则可以通过以下操作来管理资源的标签：

- CreateBot 和 CreateBotAlias：在创建机器人或机器人别名时应用标签。
- ListTagsForResource：查看与资源关联的标签。
- TagResource：添加和修改现有资源上的标签。
- UntagResource：从资源中删除标签。

Amazon Lex V2 中支持贴标签的资源如下：

- 机器人：使用如下所示的 Amazon 资源名称 (ARN)：
  - `arn:aws:lex:${Region}:${account}:bot/${bot-id}`

- 机器人别名：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` 取值均是由 10 个大写字母和数字组成的字符串。

标签限制

以下基本限制适用于 Amazon Lex V2 资源上的标签：

- 最大密钥数量：50 个使用控制台，200 个使用 API
- 最大键长度：128 个字符
- 最大值长度：256 个字符
- 键和值的有效字符：a-z、A-Z、0-9、空格和以下字符：_ . : / = + - @
- 键和值区分大小写
- 请不要使用 `aws:` 作为键的前缀；它保留为供 AWS 使用

标记资源（控制台）

您可以通过控制台管理机器人或机器人别名上的标签。您可以在创建资源时添加标签，也可以从现有资源中添加、修改或删除标签。

在创建机器人时添加标签

1. 登录 AWS Management Console，并通过以下网址打开 Amazon Lex 控制台：https://console.aws.amazon.com/lex/。
2. 选择创建机器人。
3. 在配置机器人设置的高级设置部分，选择添加新标签。您可以为机器人和 `TestBotAlias` 别名添加标签。
4. 选择下一步，继续创建您的机器人。

在创建机器人别名时添加标签

1. 登录 AWS Management Console，并通过以下网址打开 Amazon Lex 控制台：https://console.aws.amazon.com/lex/。
2. 选择想要为其添加机器人别名的机器人。

3. 在左侧菜单中，选择别名 > 创建别名。
4. 在一般信息中，从标签中选择添加新标签。
5. 选择创建。

添加、删除或修改现有机器人上的标签

1. 登录 AWS Management Console，并通过以下网址打开 Amazon Lex 控制台：https://console.aws.amazon.com/lex/。
2. 选择想要修改的机器人。
3. 在左侧菜单中，选择设置 > 编辑。
4. 在标签中，进行更改。
5. 选择保存来保存对机器人的更改。

添加、删除或修改现有别名上的标签

1. 登录 AWS Management Console，并通过以下网址打开 Amazon Lex 控制台：https://console.aws.amazon.com/lex/。
2. 选择想要修改的机器人。
3. 在左侧菜单中，选择别名，然后从别名列表中选择要修改的别名。
4. 在别名详细信息上的标签中，选择修改标签。
5. 在管理标签中，进行更改。
6. 选择保存，来保存对别名的更改。

Amazon Lex 中的安全性

云安全 AWS 是重中之重。作为 AWS 客户，您可以受益于专为满足大多数安全敏感型组织的要求而构建的数据中心和网络架构。

安全是双方共同承担 AWS 的责任。责任共担模式将其描述为云的安全性和云中的安全性：

- 云安全 — AWS 负责保护在 AWS 云中运行 AWS 服务的基础架构。 AWS 还为您提供可以安全使用的服务。作为AWS 合规计划合规计划合规计划合的一部分，第三方审计师定期测试和验证我们安全的有效性。要了解适用于 Amazon Lex V2 的合规性计划，请参阅合规性计划范围内的 AWS 服务。
- 云端安全-您的责任由您使用的 AWS 服务决定。您还需要对其他因素负责，包括您的数据的敏感性、您公司的要求以及适用的法律法规。

本文档可帮助您了解如何在使用 Amazon Lex V2 时应用责任共担模式。以下主题说明如何配置 Amazon Lex V2 以实现您的安全性和合规性目标。您还会了解如何使用其他 AWS 服务来帮助您监控和保护 Amazon Lex V2 资源。

主题

Amazon Lex V2 中的数据保护

Amazon Lex V2 符合 AWS 共担责任模式，其中包括数据保护的法规和指南。 AWS 负责保护运行所有 AWS 服务的全球基础架构。 AWS 保持对托管在此基础架构上的数据的控制，包括用于处理客户内容和个人数据的安全配置控制。 AWS 作为数据控制者或数据处理者的客户和 APN 合作伙伴应对他们存入 AWS 云端的任何个人数据负责。

出于数据保护的目的，我们建议您保护 AWS 账户凭证并使用 AWS Identity and Access Management (IAM) 设置单个用户账户，以便向每个用户仅提供履行其工作职责所需的权限。我们还建议您通过以下方式保护数据：

- 对每个账户使用多重身份验证（MFA）。
- 使用 SSL/TLS 与资源通信。 AWS
- 使用设置 API 和用户活动日志 AWS CloudTrail。
- 使用 AWS 加密解决方案以及 AWS 服务中的所有默认安全控制。
- 使用高级托管安全服务（例如 Amazon Macie），它有助于发现和保护存储在 Amazon S3 中的个人数据。

我们强烈建议您切勿将敏感的可识别信息（例如您客户的账号）放入自由格式字段（例如名称字段）。这包括您使用控制台、API 或 AWS 软件开发工具包使用 Amazon Lex V2 或其他 AWS 服务时。 AWS CLI您输入到 Amazon Lex V2 或其他服务中的任何数据都可能被选取以包含在诊断日志中。当您向外部服务器提供网址时，请勿在网址中包含凭证信息来验证您对该服务器的请求。

有关数据保护的更多信息，请参阅AWS 安全性博客 上的AWS 责任共担模式和 GDPR 博客文章。

静态加密

Amazon Lex V2 对其存储的用户言语和其他信息进行加密。

主题

- 示例言语
- 会话属性
- 请求属性

示例言语

当您开发机器人时，您可以为每个意图和插槽提供示例言语。您还可以为槽位提供自定义值和同义词。这些信息是静态加密的，仅用于构建机器人并创建客户体验。

会话属性

会话属性包含于 Amazon Lex V2 与客户端应用程序之间传递的特定于应用程序的信息。Amazon Lex V2 将会话属性传递给为机器人配置的所有 AWS Lambda 函数。如果 Lambda 函数添加或更新了会话属性，Amazon Lex V2 会将新信息返回给客户端应用程序。

会话属性在会话存续期内一直存在于加密存储中。您可以将会话配置为在最后一次用户言语之后保持活动最少 1 分钟，最多 24 小时。默认会话持续时间为 5 分钟。

请求属性

请求属性包含请求特定的信息，并仅应用于当前请求。客户端应用程序在运行时使用请求属性向 Amazon Lex V2 发送信息。

您可以使用请求属性传递不需要在整个会话中保留的信息。由于请求属性不会跨属性持久存在，所以不存储它们。

传输中加密

Amazon Lex V2 使用 HTTPS 协议与客户端应用程序进行通信。它使用 HTTPS 和 AWS 签名代表您的应用程序与其他服务（例如 Amazon Polly） AWS Lambda 进行通信。

适用于 Amazon Lex V2 的身份和访问管理

AWS Identity and Access Management (IAM) AWS 服务 可帮助管理员安全地控制对 AWS 资源的访问权限。IAM 管理员可控制能够通过身份验证（登录）和获得授权（拥有权限）来使用 Amazon Lex V2 资源的用户。您可以使用 IAM AWS 服务 ，无需支付额外费用。

主题

- 对 Amazon Lex V2 使用服务相关角色
- Amazon Lex V2 身份和访问问题排查

受众

您的使用方式 AWS Identity and Access Management (IAM) 会有所不同，具体取决于您在 Amazon Lex V2 中所做的工作。

服务用户：如果您使用 Amazon Lex V2 服务来完成任务，则您的管理员会为您提供所需的凭证和权限。随着您使用更多 Amazon Lex V2 功能来完成任务，您可能需要额外的权限。了解如何管理访问权限有助于您向管理员请求适合的权限。如果您无法访问 Amazon Lex V2 中的功能，请参阅 Amazon Lex V2 身份和访问问题排查。

服务管理员：如果您是贵公司 Amazon Lex V2 资源的管理人员，您可能拥有对 Amazon Lex V2 的完全访问权限。在此情况下，您需要确定可被服务用户访问的 Amazon Lex V2 功能和资源。然后，您必须向 IAM 管理员提交请求以更改服务用户的权限。请查看该页面上的信息以了解 IAM 的基本概念。要了解有关贵公司如何将 IAM 与 Amazon Lex V2 搭配使用的更多信息，请参阅 Amazon Lex V2 如何与 IAM 协同工作。

IAM 管理员：如果您是 IAM 管理员，可能需要了解关于如何编写策略以管理对 Amazon Lex V2 的访问权限的详细信息。要查看您可在 IAM 中使用的 Amazon Lex V2 基于身份的策略示例，请参阅 Amazon Lex V2 基于身份的策略示例。

使用身份进行身份验证

身份验证是您 AWS 使用身份凭证登录的方式。您必须以 IAM 用户身份或通过担 AWS 账户根用户任 IAM 角色进行身份验证（登录 AWS）。

您可以使用通过身份源提供的凭据以 AWS 联合身份登录。 AWS IAM Identity Center （IAM Identity Center）用户、贵公司的单点登录身份验证以及您的 Google 或 Facebook 凭据就是联合身份的示例。当您以联合身份登录时，您的管理员以前使用 IAM 角色设置了身份联合验证。当你使用联合访问 AWS 时，你就是在间接扮演一个角色。

根据您的用户类型，您可以登录 AWS Management Console 或 AWS 访问门户。有关登录的更多信息 AWS，请参阅《AWS 登录 用户指南》中的如何登录到您 AWS 账户的。

如果您 AWS 以编程方式访问，则会 AWS 提供软件开发套件 (SDK) 和命令行接口 (CLI)，以便使用您的凭据对请求进行加密签名。如果您不使用 AWS 工具，则必须自己签署请求。有关使用推荐的方法自行签署请求的更多信息，请参阅 IAM 用户指南中的签署 AWS API 请求。

无论使用何种身份验证方法，您可能需要提供其他安全信息。例如， AWS 建议您使用多重身份验证 (MFA) 来提高账户的安全性。要了解更多信息，请参阅《AWS IAM Identity Center 用户指南》中的 多重身份验证和《IAM 用户指南》中的 在 AWS中使用多重身份验证（MFA）。

AWS 账户 root 用户

创建时 AWS 账户，首先要有一个登录身份，该身份可以完全访问账户中的所有资源 AWS 服务 和资源。此身份被称为 AWS 账户 root 用户，使用您创建账户时使用的电子邮件地址和密码登录即可访问该身份。强烈建议您不要使用根用户执行日常任务。保护好根用户凭证，并使用这些凭证来执行仅根用户可以执行的任务。有关要求您以根用户身份登录的任务的完整列表，请参阅《IAM 用户指南》中的需要根用户凭证的任务。

联合身份

作为最佳实践，要求人类用户（包括需要管理员访问权限的用户）使用与身份提供商的联合身份验证 AWS 服务 通过临时证书进行访问。

联合身份是指您的企业用户目录、Web 身份提供商、Identity C enter 目录中的用户，或者任何使用 AWS 服务 通过身份源提供的凭据进行访问的用户。 AWS Directory Service当联合身份访问时 AWS 账户，他们将扮演角色，角色提供临时证书。

要集中管理访问权限，建议您使用 AWS IAM Identity Center。您可以在 IAM Identity Center 中创建用户和群组，也可以连接并同步到您自己的身份源中的一组用户和群组，以便在您的所有 AWS 账户 和应用程序中使用。有关 IAM Identity Center 的信息，请参阅《AWS IAM Identity Center 用户指南》中的什么是 IAM Identity Center？

IAM 用户和群组

I AM 用户是您 AWS 账户 内部对个人或应用程序具有特定权限的身份。在可能的情况下，我们建议使用临时凭证，而不是创建具有长期凭证（如密码和访问密钥）的 IAM 用户。但是，如果您有一些特定的使用场景需要长期凭证以及 IAM 用户，建议您轮换访问密钥。有关更多信息，请参阅《IAM 用户指南》中的 对于需要长期凭证的使用场景定期轮换访问密钥。

IAM 组是一个指定一组 IAM 用户的身份。您不能使用组的身份登录。您可以使用组来一次性为多个用户指定权限。如果有大量用户，使用组可以更轻松地管理用户权限。例如，您可能具有一个名为 IAMAdmins 的组，并为该组授予权限以管理 IAM 资源。

用户与角色不同。用户唯一地与某个人员或应用程序关联，而角色旨在让需要它的任何人代入。用户具有永久的长期凭证，而角色提供临时凭证。要了解更多信息，请参阅《IAM 用户指南》中的何时创建 IAM 用户（而不是角色）。

IAM 角色

I AM 角色是您内部具有特定权限 AWS 账户 的身份。它类似于 IAM 用户，但与特定人员不关联。您可以 AWS Management Console 通过切换角色在中临时担任 IAM 角色。您可以通过调用 AWS CLI 或 AWS API 操作或使用自定义 URL 来代入角色。有关使用角色的方法的更多信息，请参阅《IAM 用户指南》中的使用 IAM 角色。

具有临时凭证的 IAM 角色在以下情况下很有用：

- 联合用户访问 – 要向联合身份分配权限，请创建角色并为角色定义权限。当联合身份进行身份验证时，该身份将与角色相关联并被授予由此角色定义的权限。有关联合身份验证的角色的信息，请参阅《IAM 用户指南》中的为第三方身份提供商创建角色。如果您使用 IAM Identity Center，则需要配置权限集。为控制您的身份在进行身份验证后可以访问的内容，IAM Identity Center 将权限集与 IAM 中的角色相关联。有关权限集的信息，请参阅《AWS IAM Identity Center 用户指南》中的 权限集。
- 临时 IAM 用户权限 – IAM 用户可代入 IAM 用户或角色，以暂时获得针对特定任务的不同权限。
- 跨账户存取 – 您可以使用 IAM 角色以允许不同账户中的某个人（可信主体）访问您的账户中的资源。角色是授予跨账户访问权限的主要方式。但是，对于某些资源 AWS 服务，您可以将策略直接附加到资源（而不是使用角色作为代理）。要了解角色和基于资源的跨账户访问策略之间的区别，请参阅 IA M 用户指南中的跨账户资源访问。
- 跨服务访问 — 有些 AWS 服务 使用其他 AWS 服务服务中的功能。例如，当您在某个服务中进行调用时，该服务通常会在 Amazon EC2 中运行应用程序或在 Simple Storage Service (Amazon S3) 中存储对象。服务可能会使用发出调用的主体的权限、使用服务角色或使用服务相关角色来执行此操作。
  - 转发访问会话 (FAS) — 当您使用 IAM 用户或角色在中执行操作时 AWS，您被视为委托人。使用某些服务时，您可能会执行一个操作，然后此操作在其他服务中启动另一个操作。FAS 使用调用委托人的权限以及 AWS 服务 向下游服务发出请求的请求。 AWS 服务只有当服务收到需要与其他 AWS 服务 或资源交互才能完成的请求时，才会发出 FAS 请求。在这种情况下，您必须具有执行这两个操作的权限。有关发出 FAS 请求时的策略详情，请参阅转发访问会话。
  - 服务角色 - 服务角色是服务代表您在您的账户中执行操作而分派的 IAM 角色。IAM 管理员可以在 IAM 中创建、修改和删除服务角色。有关更多信息，请参阅《IAM 用户指南》中的创建向 AWS 服务委派权限的角色。
  - 服务相关角色-服务相关角色是一种链接到的服务角色。 AWS 服务服务可以代入代表您执行操作的角色。服务相关角色出现在您的中 AWS 账户 ，并且归服务所有。IAM 管理员可以查看但不能编辑服务相关角色的权限。
- 在 A@@ mazon EC2 上运行的应用程序 — 您可以使用 IAM 角色管理在 EC2 实例上运行并发出 AWS CLI 或 AWS API 请求的应用程序的临时证书。这优先于在 EC2 实例中存储访问密钥。要向

EC2 实例分配 AWS 角色并使其可供其所有应用程序使用，您需要创建附加到该实例的实例配置文件。实例配置文件包含角色，并使 EC2 实例上运行的程序能够获得临时凭证。有关更多信息，请参阅《IAM 用户指南》中的使用 IAM 角色为 Amazon EC2 实例上运行的应用程序授予权限。

要了解是使用 IAM 角色还是 IAM 用户，请参阅IAM 用户指南中的何时创建 IAM 角色（而不是用户）。

使用策略管理访问

您可以 AWS 通过创建策略并将其附加到 AWS 身份或资源来控制中的访问权限。策略是其中的一个对象 AWS ，当与身份或资源关联时，它会定义其权限。 AWS 在委托人（用户、root 用户或角色会话）发出请求时评估这些策略。策略中的权限确定是允许还是拒绝请求。大多数策略都以 JSON 文档的 AWS 形式存储在中。有关 JSON 策略文档的结构和内容的更多信息，请参阅 IAM 用户指南中的 JSON 策略概览。

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

默认情况下，用户和角色没有权限。要授予用户对所需资源执行操作的权限，IAM 管理员可以创建 IAM 策略。管理员随后可以向角色添加 IAM policy，用户可以代入角色。

IAM 策略定义操作的权限，无关乎您使用哪种方法执行操作。例如，假设您有一个允许 `iam:GetRole` 操作的策略。拥有该策略的用户可以从 AWS Management Console AWS CLI、或 AWS API 获取角色信息。

基于身份的策略

基于身份的策略是可附加到身份（如 IAM 用户、用户组或角色）的 JSON 权限策略文档。这些策略控制用户和角色可在何种条件下对哪些资源执行哪些操作。要了解如何创建基于身份的策略，请参阅《IAM 用户指南》中的创建 IAM policy。

基于身份的策略可以进一步归类为内联策略或托管策略。内联策略直接嵌入单个用户、组或角色中。托管策略是独立的策略，您可以将其附加到中的多个用户、群组和角色 AWS 账户。托管策略包括 AWS 托管策略和客户托管策略。要了解如何在托管式策略和内联策略之间进行选择，请参阅 IAM 用户指南中的在托管式策略与内联策略之间进行选择。

基于资源的策略

基于资源的策略是附加到资源的 JSON 策略文档。基于资源的策略的示例包括 IAM 角色信任策略和 Amazon S3 存储桶策略。在支持基于资源的策略的服务中，服务管理员可以使用它们来控制对特定资

源的访问。对于在其中附加策略的资源，策略定义指定主体可以对该资源执行哪些操作以及在什么条件下执行。您必须在基于资源的策略中指定主体。委托人可以包括账户、用户、角色、联合用户或 AWS 服务。

基于资源的策略是位于该服务中的内联策略。您不能在基于资源的策略中使用 IAM 中的 AWS 托管策略。

访问控制列表 (ACL)

访问控制列表（ACL）控制哪些主体（账户成员、用户或角色）有权访问资源。ACL 与基于资源的策略类似，尽管它们不使用 JSON 策略文档格式。

Amazon S3 和 Amazon VPC 就是支持 ACL 的服务示例。 AWS WAF要了解有关 ACL 的更多信息，请参阅《Amazon Simple Storage Service 开发人员指南》中的访问控制列表（ACL）概览。

其他策略类型

AWS 支持其他不太常见的策略类型。这些策略类型可以设置更常用的策略类型向您授予的最大权限。

- 权限边界：权限边界是一个高级特征，用于设置基于身份的策略可以为 IAM 实体（IAM 用户或角色）授予的最大权限。您可为实体设置权限边界。这些结果权限是实体基于身份的策略及其权限边界的交集。在 `Principal` 中指定用户或角色的基于资源的策略不受权限边界限制。任一项策略中的显式拒绝将覆盖允许。有关权限边界的更多信息，请参阅《IAM 用户指南》中的 IAM 实体的权限边界。
- 服务控制策略 (SCP)-SCP 是 JSON 策略，用于指定组织或组织单位 (OU) 的最大权限。 AWS Organizations AWS Organizations 是一项用于对您的企业拥有的多 AWS 账户 项进行分组和集中管理的服务。如果在组织内启用了所有功能，则可对任意或全部账户应用服务控制策略 (SCP)。SCP 限制成员账户中的实体（包括每个 AWS 账户根用户实体）的权限。有关 Organizations 和 SCP 的更多信息，请参阅《AWS Organizations 用户指南》中的 SCP 的工作原理。
- 会话策略 – 会话策略是当您以编程方式为角色或联合用户创建临时会话时作为参数传递的高级策略。结果会话的权限是用户或角色的基于身份的策略和会话策略的交集。权限也可以来自基于资源的策略。任一项策略中的显式拒绝将覆盖允许。有关更多信息，请参阅《IAM 用户指南》中的会话策略。

多个策略类型

当多个类型的策略应用于一个请求时，生成的权限更加复杂和难以理解。要了解在涉及多种策略类型时如何 AWS 确定是否允许请求，请参阅 IAM 用户指南中的策略评估逻辑。

Amazon Lex V2 如何与 IAM 协同工作

在使用 IAM 管理对 Amazon Lex V2 的访问权限之前，您需要了解哪些 IAM 功能可与 Amazon Lex V2 配合使用。

可与 Amazon Lex V2 配合使用的IAM 功能

IAM 功能	Amazon Lex V2 支持
基于身份的策略	是
基于资源的策略	是
策略操作	是
策略资源	是
策略条件密钥	否
ACL	否
ABAC（策略中的标签）	是
临时凭证	否
主体权限	是
服务角色	是
服务相关角色	部分

要全面了解 Amazon Lex V2 和其他 AWS 服务如何与大多数 IAM 功能配合使用，请参阅 IAM 用户指南中与 IAM 配合使用的AWS 服务。

Amazon Lex V2 基于身份的策略

支持基于身份的策略	是

基于身份的策略是可附加到身份（如 IAM 用户、用户组或角色）的 JSON 权限策略文档。这些策略控制用户和角色可在何种条件下对哪些资源执行哪些操作。要了解如何创建基于身份的策略，请参阅《IAM 用户指南》中的创建 IAM policy。

通过使用 IAM 基于身份的策略，您可以指定允许或拒绝的操作和资源以及允许或拒绝操作的条件。您无法在基于身份的策略中指定主体，因为它适用于其附加的用户或角色。要了解可在 JSON 策略中使用的所有元素，请参阅《IAM 用户指南》中的 IAM JSON 策略元素引用。

Amazon Lex V2 基于身份的策略示例

要查看 Amazon Lex V2 基于身份的策略的示例，请参阅 Amazon Lex V2 基于身份的策略示例。

Amazon Lex V2 基于资源的策略

支持基于资源的策略	是

基于资源的策略是附加到资源的 JSON 策略文档。基于资源的策略的示例包括 IAM 角色信任策略和 Amazon S3 存储桶策略。在支持基于资源的策略的服务中，服务管理员可以使用它们来控制对特定资源的访问。对于在其中附加策略的资源，策略定义指定主体可以对该资源执行哪些操作以及在什么条件下执行。您必须在基于资源的策略中指定主体。主体可以包括用户、角色、联合用户或 AWS 服务。

您无法对 Amazon Lex 使用跨账户或跨区域策略。如果为具有跨账户或跨区域 ARN 的资源创建策略，Amazon Lex 会返回错误。

Amazon Lex 服务支持附加于机器人或机器人别名上的基于资源的策略，即机器人策略和机器人别名策略。这些策略定义了能够对机器人或机器人别名执行操作的主体。

只能对特定资源使用操作。例如，`UpdateBot` 操作只能被用于机器人资源上，`UpdateBotAlias` 操作只能被用于机器人别名资源。如果您在策略中指定的操作无法用于策略中指定的资源，Amazon Lex 将返回错误。有关操作及其可与之配合使用的资源列表，请参阅下表。

操作	支持基于资源的策略	资源
BuildBotLocal	支持	BotId
CreateBot	否	
CreateBot别名	否	

操作	支持基于资源的策略	资源
CreateBotChannel [仅限许可]	支持	BotId
CreateBotLocal	支持	BotId
CreateBot版本	支持	BotId
CreateExport	支持	BotId
CreateIntent	支持	BotId
CreateResource政策	支持	BotId, BotAliasId
CreateSlot	支持	BotId
CreateSlot类型	支持	BotId
CreateUpload网址	否	
DeleteBot	支持	BotId, BotAliasId
DeleteBot别名	支持	BotAlias我是
DeleteBotChannel [仅限许可]	支持	BotId
DeleteBotLocal	支持	BotId
DeleteBot版本	支持	BotId
DeleteExport	支持	BotId
DeleteImport	支持	BotId
DeleteIntent	支持	BotId
DeleteResource政策	支持	BotId, BotAliasId
DeleteSession	支持	BotAlias我是
DeleteSlot	支持	BotId

操作	支持基于资源的策略	资源
DeleteSlot类型	支持	BotId
DescribeBot	支持	BotId
DescribeBot别名	支持	BotAlias我是
DescribeBotChannel [仅限许可]	支持	BotId
DescribeBotLocal	支持	BotId
DescribeBot版本	支持	BotId
DescribeExport	支持	BotId
DescribeImport	支持	BotId
DescribeIntent	支持	BotId
DescribeResource政策	支持	BotId, BotAliasId
DescribeSlot	支持	BotId
DescribeSlot类型	支持	BotId
GetSession	支持	BotAlias我是
ListBot别名	支持	BotId
ListBotChannels [仅限许可]	支持	BotId
ListBotLocales	支持	BotId
ListBots	否	
ListBot版本	支持	BotId
ListBuiltInIntents	否	
ListBuiltIntSlot类型	否	

操作	支持基于资源的策略	资源
ListExports	否	
ListImports	否	
ListIntents	支持	BotId
ListSlots	支持	BotId
ListSlot类型	支持	BotId
PutSession	支持	BotAlias我是
RecognizeText	支持	BotAlias我是
RecognizeUtterance	支持	BotAlias我是
StartConversation	支持	BotAlias我是
StartImport	支持	BotId, BotAliasId
TagResource	否	
UpdateBot	支持	BotId
UpdateBot别名	支持	BotAlias我是
UpdateBotLocal	支持	BotId
UpdateBot版本	支持	BotId
UpdateExport	支持	BotId
UpdateIntent	支持	BotId
UpdateResource政策	支持	BotId, BotAliasId
UpdateSlot	支持	BotId
UpdateSlot类型	支持	BotId

操作	支持基于资源的策略	资源
UntagResource	否	

要了解如何将基于资源的策略附加到机器人或机器人别名，请参阅 Amazon Lex V2 基于资源的策略示例。

Amazon Lex V2 基于资源的策略示例

要查看 Amazon Lex V2 基于资源的策略的示例，请参阅 Amazon Lex V2 基于资源的策略示例，

Amazon Lex V2 的策略操作

支持策略操作	是

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

JSON 策略的 `Action` 元素描述可用于在策略中允许或拒绝访问的操作。策略操作通常与关联的 AWS API 操作同名。有一些例外情况，例如没有匹配 API 操作的仅限权限 操作。还有一些操作需要在策略中执行多个操作。这些附加操作称为相关操作。

在策略中包含操作以授予执行关联操作的权限。

有关 Amazon Lex V2 操作的列表，请参阅《服务授权参考》中的 Amazon Lex V2 定义的操作。

Amazon Lex V2 中的策略操作在操作前面使用以下前缀：

```
lex
```

要在单个语句中指定多项操作，请使用逗号将它们隔开。

```
"Action": [
      "lex:action1",
      "lex:action2"
```

```
        ]
```

要查看 Amazon Lex V2 基于身份的策略的示例，请参阅 Amazon Lex V2 基于身份的策略示例。

Amazon Lex V2 的策略资源

支持策略资源	是

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。也就是说，哪个主体 可以对什么资源执行操作，以及在什么条件下执行。

`Resource` JSON 策略元素指定要向其应用操作的一个或多个对象。语句必须包含 `Resource` 或 `NotResource` 元素。作为最佳实践，请使用其 Amazon 资源名称 (ARN) 指定资源。对于支持特定资源类型（称为资源级权限）的操作，您可以执行此操作。

对于不支持资源级权限的操作（如列出操作），请使用通配符 (*) 指示语句应用于所有资源。

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

要查看 Amazon Lex V2 的资源类型及其 ARN 的列表，请参阅《服务授权参考》中的 Amazon Lex V2 定义的资源。要了解您可以使用哪些操作指定每个资源的 ARN，请参阅 Amazon Lex V2 定义的操作。

要查看 Amazon Lex V2 基于身份的策略的示例，请参阅 Amazon Lex V2 基于身份的策略示例。

Amazon Lex V2 的策略条件键

支持特定于服务的策略条件密钥	否

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

在 `Condition` 元素（或 `Condition` 块）中，可以指定语句生效的条件。`Condition` 元素是可选的。您可以创建使用条件运算符（例如，等于或小于）的条件表达式，以使策略中的条件与请求中的值相匹配。

如果您在一个语句中指定多个 Condition 元素，或在单个 Condition 元素中指定多个键，则 AWS 使用逻辑 AND 运算评估它们。如果您为单个条件键指定多个值，则使用逻辑OR运算来 AWS 评估条件。在授予语句的权限之前必须满足所有的条件。

在指定条件时，您也可以使用占位符变量。例如，只有在使用 IAM 用户名标记 IAM 用户时，您才能为其授予访问资源的权限。有关更多信息，请参阅《IAM 用户指南》中的 IAM policy 元素：变量和标签。

AWS 支持全局条件密钥和特定于服务的条件密钥。要查看所有 AWS 全局条件键，请参阅 IAM 用户指南中的AWS 全局条件上下文密钥。

要查看 Amazon Lex V2 条件键的列表，请参阅《服务授权参考》中的 Amazon Lex V2 的条件键。要了解您可以对哪些操作和资源使用条件键，请参阅 Amazon Lex V2 定义的操作。

要查看 Amazon Lex V2 基于身份的策略的示例，请参阅 Amazon Lex V2 基于身份的策略示例。

Amazon Lex V2 中的访问控制列表 (ACL)

支持 ACL	否

访问控制列表 (ACL) 控制哪些主体（账户成员、用户或角色）有权访问资源。ACL 与基于资源的策略类似，尽管它们不使用 JSON 策略文档格式。

使用 Amazon Lex V2 基于属性的访问权限控制 (ABAC)

支持 ABAC（策略中的标签）	是

基于属性的访问控制 (ABAC) 是一种授权策略，该策略基于属性来定义权限。在中 AWS，这些属性称为标签。您可以将标签附加到 IAM 实体（用户或角色）和许多 AWS 资源。标记实体和资源是 ABAC 的第一步。然后设计 ABAC 策略，以在主体的标签与他们尝试访问的资源标签匹配时允许操作。

ABAC 在快速增长的环境中非常有用，并在策略管理变得繁琐的情况下可以提供帮助。

要基于标签控制访问，您需要使用 aws:ResourceTag/*key-name*、aws:RequestTag/*key-name* 或 aws:TagKeys 条件键在策略的条件元素中提供标签信息。

如果某个服务对于每种资源类型都支持所有这三个条件键，则对于该服务，该值为是。如果某个服务仅对于部分资源类型支持所有这三个条件键，则该值为部分。

有关 ABAC 的更多信息,请参阅《IAM 用户指南》中的什么是 ABAC？。要查看设置 ABAC 步骤的教程，请参阅《IAM 用户指南》中的使用基于属性的访问权限控制（ABAC）。

使用 Amazon Lex V2 的临时凭证

支持临时凭证	否

当你使用临时证书登录时，有些 AWS 服务 不起作用。有关更多信息，包括哪些 AWS 服务 适用于临时证书，请参阅 IAM 用户指南中的AWS 服务 与 IAM 配合使用的信息。

如果您使用除用户名和密码之外的任何方法登录，则 AWS Management Console 使用的是临时证书。例如，当您 AWS 使用公司的单点登录 (SSO) 链接进行访问时，该过程会自动创建临时证书。当您以用户身份登录控制台，然后切换角色时，您还会自动创建临时凭证。有关切换角色的更多信息，请参阅《IAM 用户指南》中的 切换到角色（控制台）。

您可以使用 AWS CLI 或 AWS API 手动创建临时证书。然后，您可以使用这些临时证书进行访问 AWS。 AWS 建议您动态生成临时证书，而不是使用长期访问密钥。有关更多信息，请参阅 IAM 中的临时安全凭证。

Amazon Lex V2 的跨服务主体权限

支持转发访问会话 (FAS)	是

当您使用 IAM 用户或角色在中执行操作时 AWS，您被视为委托人。使用某些服务时，您可能会执行一个操作，然后此操作在其他服务中启动另一个操作。FAS 使用调用委托人的权限以及 AWS 服务 向下游服务发出请求的请求。 AWS 服务只有当服务收到需要与其他 AWS 服务 或资源交互才能完成的请求时，才会发出 FAS 请求。在这种情况下，您必须具有执行这两个操作的权限。有关发出 FAS 请求时的策略详情，请参阅转发访问会话。

Amazon Lex V2 的服务角色

支持服务角色	是

服务角色是由一项服务担任、代表您执行操作的 IAM 角色。IAM 管理员可以在 IAM 中创建、修改和删除服务角色。有关更多信息，请参阅《IAM 用户指南》中的创建向 AWS 服务委派权限的角色。

> **Warning**
>
> 更改服务角色的权限可能会破坏 Amazon Lex V2 的功能。仅当 Amazon Lex V2 提供相关指导时才编辑服务角色。

Amazon Lex V2 的服务相关角色

支持服务相关角色	部分

服务相关角色是一种与服务相关联的 AWS 服务服务角色。服务可以代入代表您执行操作的角色。服务相关角色出现在您的中 AWS 账户 ，并且归服务所有。IAM 管理员可以查看但不能编辑服务相关角色的权限。

有关创建或管理服务相关角色的详细信息，请参阅能够与 IAM 搭配使用的AWS 服务。在表中查找服务相关角色列中包含 Yes 的表。选择是链接以查看该服务的服务相关角色文档。

Amazon Lex V2 基于身份的策略示例

默认情况下，用户和角色没有创建或修改 Amazon Lex V2 资源的权限。他们也无法使用 AWS Management Console、 AWS Command Line Interface (AWS CLI) 或 AWS API 执行任务。要授予用户对所需资源执行操作的权限，IAM 管理员可以创建 IAM policy。管理员随后可以向角色添加 IAM 策略，用户可以代入角色。

要了解如何使用这些示例 JSON 策略文档创建基于 IAM 身份的策略，请参阅 IAM 用户指南中的 创建 IAM 策略。

有关 Amazon Lex V2 定义的操作和资源类型的详细信息，包括每种资源类型的 ARN 格式，请参阅《服务授权参考》中的 Amazon Lex V2 的操作、资源和条件键。

主题

- 策略最佳实践
- 使用 Amazon Lex V2 控制台
- 允许用户为机器人添加功能
- 允许用户为机器人添加频道
- 允许用户创建和更新机器人

- 允许用户使用自动聊天机器人设计器
- 允许用户使用密 AWS KMS 钥加密和解密文件
- 允许用户删除机器人
- 允许用户与机器人进行对话
- 允许特定用户管理基于资源的策略
- 允许用户导出机器人和机器人区域设置
- 允许用户导出自定义词汇表
- 允许用户导入机器人和机器人区域设置
- 允许用户导入自定义词汇表
- 允许用户将机器人从 Amazon Lex 迁移到 Amazon Lex V2
- 允许用户查看他们自己的权限
- 允许用户在 Amazon Lex V2 中使用可视化对话生成器绘制对话流程
- 允许用户创建和查看机器人副本，但不允许将其删除

策略最佳实践

基于身份的策略可确定相关用户能否创建、访问或删除您账户中的 Amazon Lex V2 资源。这些操作可能会使 AWS 账户产生成本。创建或编辑基于身份的策略时，请遵循以下指南和建议：

- 开始使用 AWS 托管策略并转向最低权限权限 — 要开始向用户和工作负载授予权限，请使用为许多常见用例授予权限的AWS 托管策略。它们在你的版本中可用 AWS 账户。我们建议您通过定义针对您的用例的 AWS 客户托管策略来进一步减少权限。有关更多信息，请参阅《IAM 用户指南》中的 AWS 托管策略 或 工作职能的AWS 托管策略。
- 应用最低权限 – 在使用 IAM policy 设置权限时，请仅授予执行任务所需的权限。为此，您可以定义在特定条件下可以对特定资源执行的操作，也称为最低权限许可。有关使用 IAM 应用权限的更多信息，请参阅《IAM 用户指南》中的 IAM 中的策略和权限。
- 使用 IAM policy 中的条件进一步限制访问权限 – 您可以向策略添加条件来限制对操作和资源的访问。例如，您可以编写策略条件来指定必须使用 SSL 发送所有请求。如果服务操作是通过特定的方式使用的，则也可以使用条件来授予对服务操作的访问权限 AWS 服务，例如 AWS CloudFormation。有关更多信息，请参阅《IAM 用户指南》中的 IAM JSON 策略元素：条件 。
- 使用 IAM Access Analyzer 验证您的 IAM policy，以确保权限的安全性和功能性 – IAM Access Analyzer 会验证新策略和现有策略，以确保策略符合 IAM policy语言 (JSON) 和 IAM 最佳实践。IAM Access Analyzer 提供 100 多项策略检查和可操作的建议，以帮助您制定安全且功能性强的策略。有关更多信息，请参阅《IAM 用户指南》中的 IAM Acess Analyzer 策略验证。

- 需要多重身份验证 (MFA)-如果 AWS 账户您的场景需要 IAM 用户或根用户，请启用 MFA 以提高安全性。若要在调用 API 操作时需要 MFA，请将 MFA 条件添加到您的策略中。有关更多信息，请参阅《IAM 用户指南》中的 配置受 MFA 保护的 API 访问。

有关 IAM 中的最佳实操的更多信息，请参阅《IAM 用户指南》中的 IAM 中的安全最佳实操。

使用 Amazon Lex V2 控制台

要访问 Amazon Lex V2 控制台，您必须具有一组最低的权限。这些权限必须允许您在中列出和查看有关 Amazon Lex V2 资源的详细信息。 AWS 账户如果创建比必需的最低权限更为严格的基于身份的策略，对于附加了该策略的实体（用户或角色），控制台将无法按预期正常运行。

对于仅调用 AWS CLI 或 AWS API 的用户，您无需为其设置最低控制台权限。相反，只允许访问与其尝试执行的 API 操作相匹配的操作。

为确保用户和角色仍可使用 Amazon Lex V2 控制台，用户需要拥有控制台的访问权限。有关创建具有控制台访问权限的用户的更多信息，请参阅 IAM 用户指南中的在您的 AWS 账户中创建 IAM 用户。

允许用户为机器人添加功能

此示例中的策略可允许 IAM 用户为 Amazon Lex V2 机器人添加 Amazon Comprehend、情绪分析和 Amazon Kendra 查询权限。

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

```
}
```

允许用户为机器人添加频道

此示例中的策略可允许 IAM 用户向机器人添加消息收发频道。用户必须先制定此策略，然后才能在消息收发平台上部署机器人。

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

允许用户创建和更新机器人

此示例中的示例策略可允许 IAM 用户创建和更新任何机器人。该策略包括通过控制台或使用 AWS CLI 或 AWS API 完成此操作的权限。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "lex:CreateBot",
        "lex:UpdateBot".
        "iam:PassRole"
      ],
      "Effect": "Allow",
```

```
      "Resource": ["arn:aws:lex:Region:123412341234:bot/*]
    }
  ]
}
```

允许用户使用自动聊天机器人设计器

此示例中的示例策略可允许 IAM 用户运行自动聊天机器人设计器。

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

允许用户使用密 AWS KMS 钥加密和解密文件

此示例显示了一个策略示例，该策略允许 IAM 用户使用 AWS KMS 客户托管密钥来加密和解密数据。

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

允许用户删除机器人

此示例中的示例策略可允许 IAM 用户删除任何机器人。该策略包括通过控制台或使用 AWS CLI 或 AWS API 完成此操作的权限。

```
{
  "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/*"]
    }
```

```
  ]
}
```

允许用户与机器人进行对话

此示例中的示例策略可允许 IAM 用户与任何机器人进行对话。该策略包括通过控制台或使用 AWS CLI 或 AWS API 完成此操作的权限。

```
{
  "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/*"
    }
  ]
}
```

允许特定用户管理基于资源的策略

以下示例授予特定用户管理基于资源的策略的权限。它允许控制台和 API 访问与机器人和机器人别名关联的策略。

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

允许用户导出机器人和机器人区域设置

以下 IAM 权限策略允许用户创建、更新和导出机器人或机器人区域设置。

```
{
    "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/*"]
        }
    ]
}
```

允许用户导出自定义词汇表

以下 IAM 权限策略允许用户从机器人区域设置中导出自定义词汇表。

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

```
        {"Action": [
                "lex:CreateExport",
                "lex:UpdateExport",
                "lex:DescribeExport",
                "lex:DescribeCustomVocabulary"
            ],
            "Effect": "Allow",
            "Resource": ["arn:aws:lex:Region:123456789012:bot/*"]
        }
    ]
}
```

允许用户导入机器人和机器人区域设置

以下 IAM 权限策略允许用户导入机器人或机器人区域以及查看导入状态。

```
{
    "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/*"
            ]
        }
    ]
}
```

允许用户导入自定义词汇表

以下 IAM 权限策略允许用户向机器人区域设置中导入自定义词汇表。

```
{
    "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/*"
            ]
        }
    ]
}
```

允许用户将机器人从 Amazon Lex 迁移到 Amazon Lex V2

以下 IAM 权限策略允许用户开始将机器人从 Amazon Lex 迁移到 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": "*"
        }
    ]
}
```

允许用户查看他们自己的权限

该示例说明了您如何创建策略，以允许 IAM 用户查看附加到其用户身份的内联和托管策略。此策略包括在控制台上或使用 AWS CLI 或 AWS API 以编程方式完成此操作的权限。

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

允许用户在 Amazon Lex V2 中使用可视化对话生成器绘制对话流程

以下 IAM 权限策略可允许用户在 Amazon Lex V2 中使用可视化对话生成器绘制对话流程。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {"Action": [
                "lex:UpdateIntent ",
                "lex:DescribeIntent "
            ],
            "Effect": "Allow",
            "Resource": ["arn:aws:lex:Region:123456789012:bot/*"]
        }
    ]
}
```

允许用户创建和查看机器人副本，但不允许将其删除

您可以向 IAM 角色授予以下权限，使其只能创建和查看机器人副本。通过省略`lex:DeleteBotReplica`，可以防止该角色删除机器人副本。有关更多信息，请参阅 复制机器人和管理机器人副本的权限。

```
{
    "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 基于资源的策略示例

基于资源的策略将附加到诸如机器人或机器人别名等资源上。通过使用基于资源的策略，您可以指定有权访问资源的用户，以及该等用户可以对资源执行的操作。例如，您可以添加基于资源的策略，以允许用户修改特定的机器人，或者允许用户对特定的机器人别名使用运行时操作。

当您使用基于资源的策略时，可以允许其他 AWS 服务访问您账户中的资源。例如，您可以允许 Amazon Connect 访问 Amazon Lex 机器人。

要了解如何创建机器人或机器人别名，请参阅 构建机器人。

主题

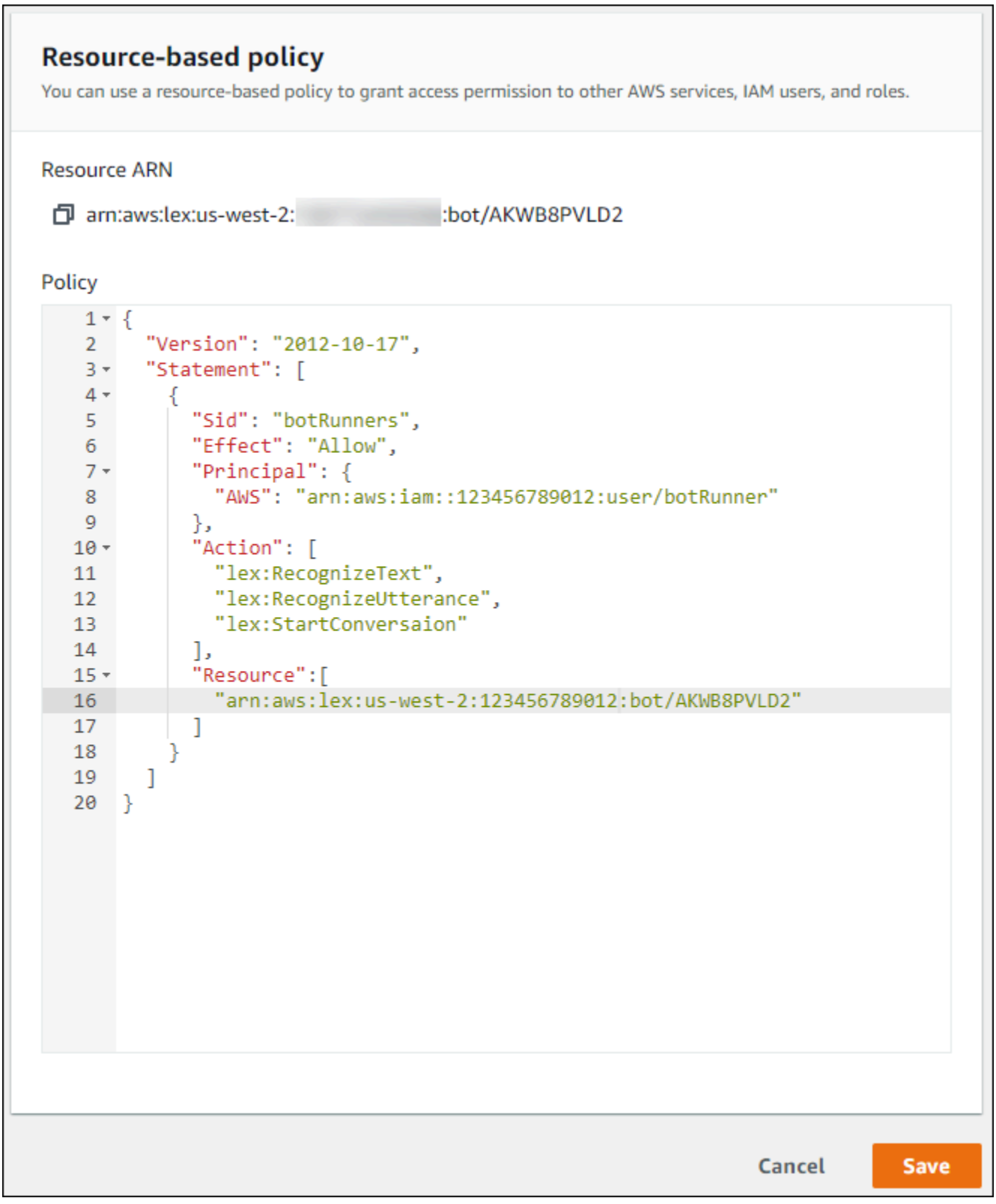
- 使用控制台来指定基于资源的策略
- 使用 API 来指定基于资源的策略
- 允许 IAM 角色更新机器人并列出机器人别名
- 允许用户与机器人进行对话
- 允许 AWS 服务使用特定的 Amazon Lex V2 机器人

使用控制台来指定基于资源的策略

您可以使用 Amazon Lex 控制台来针对机器人和机器人别名管理基于资源的策略。您输入策略的 JSON 结构，控制台会将其与资源关联。如果已有策略与资源关联，则可以使用控制台来查看和修改该策略。

使用策略编辑器保存策略时，控制台会检查策略的句法。如果策略中存在错误，例如用户不存在或存在资源不支持的操作，则会返回错误且不会保存该策略。

下图是针对控制台中机器人的基于资源的策略编辑器。机器人别名的策略编辑器与此类似。


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

要打开机器人的策略编辑器，请执行以下操作：

1. 登录 AWS Management Console 并打开 Amazon Lex 控制台，网址为 https://console.aws.amazon.com/lex/。
2. 从机器人列表中，选择要编辑其策略的机器人。
3. 在基于资源的策略部分中，选择编辑。

要打开机器人别名的策略编辑器，请执行以下操作：

1. 登录 AWS Management Console 并打开 Amazon Lex 控制台，网址为 https://console.aws.amazon.com/lex/。
2. 从机器人列表中，选择包含要编辑别名的机器人。
3. 在左侧菜单中选择别名，然后选择要编辑的别名。
4. 在基于资源的策略部分中，选择编辑。

使用 API 来指定基于资源的策略

您可以使用 API 操作来针对机器人和机器人别名管理基于资源的策略。这些操作包括创建、更新和删除策略的操作。

CreateResource政策

将带有指定策略语句的新资源策略添加到机器人或机器人别名中。

CreateResourcePolicyStatement

将新资源策略语句添加到机器人或机器人别名中。

DeleteResource政策

从机器人或机器人别名中删除资源策略。

DeleteResourcePolicyStatement

从机器人或机器人别名中删除资源策略语句。

DescribeResource政策

获取资源策略和策略修订。

UpdateResource政策

将机器人或机器人别名的现有资源策略替换为新的资源策略。

示例

Java

以下示例说明如何使用基于资源的策略操作来管理基于资源的策略。

```
        /*
         * 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);
```

允许 IAM 角色更新机器人并列出机器人别名

以下示例授予特定 IAM 角色调用 Amazon Lex V2 模型构建 API 操作以修改现有机器人的权限。用户可以列出机器人的别名并更新机器人，但无法删除机器人或机器人别名。

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

允许用户与机器人进行对话

以下示例授予特定用户在机器人的单个别名上调用 Amazon Lex V2 运行时 API 操作的权限。

该用户更新或删除机器人别名的权限被明确拒绝。

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

允许 AWS 服务使用特定的 Amazon Lex V2 机器人

以下示例授予 AWS Lambda 和 Amazon Connect 调用 Amazon Lex V2 运行时 API 操作的权限。

条件块是服务主体所必需的，并且必须使用全局上下文键 AWS:SourceAccount 和 AWS:SourceArn。

AWS:SourceAccount 是调用 Amazon Lex V2 机器人的账户编号。

AWS:SourceArn 是引发对 Amazon Lex V2 机器人别名的调用的 Amazon Connect 服务实例或 Lambda 函数的资源 ARN。

```
{
    "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 的托管策略

AWS 托管策略是由创建和管理的独立策略 AWS。 AWS 托管策略旨在为许多常见用例提供权限，以便您可以开始为用户、组和角色分配权限。

请记住， AWS 托管策略可能不会为您的特定用例授予最低权限权限，因为它们可供所有 AWS 客户使用。我们建议通过定义特定于您的使用场景的客户管理型策略来进一步减少权限。

您无法更改 AWS 托管策略中定义的权限。如果 AWS 更新 AWS 托管策略中定义的权限，则更新会影响该策略所关联的所有委托人身份（用户、组和角色）。 AWS 最有可能在启动新的 API 或现有服务可以使用新 AWS 服务 的 API 操作时更新 AWS 托管策略。

有关更多信息，请参阅《IAM 用户指南》中的 AWS 托管式策略。

AWS 托管策略： AmazonLexReadOnly

您可以将 AmazonLexReadOnly 策略附加到 IAM 身份。

此策略授予只读权限，允许用户查看 Amazon Lex V2 和 Amazon Lex 模型构建服务中的所有操作。

权限详细信息

该策略包含以下权限：

- lex：模型构建服务中对 Amazon Lex V2 和 Amazon Lex 资源的只读访问权限。

```
{
    "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 托管策略： AmazonLexRunBotsOnly

您可以将 AmazonLexRunBotsOnly 策略附加到 IAM 身份。

该策略授予只读权限，允许运行 Amazon Lex V2 和 Amazon Lex 对话机器人。

权限详细信息

该策略包含以下权限：

- lex：对 Amazon Lex V2 和 Amazon Lex 运行时中的所有操作的只读访问权限。

```
{
    "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 托管策略： AmazonLexFullAccess

您可以将 AmazonLexFullAccess 策略附加到 IAM 身份。

该政策授予管理权限，允许用户创建、读取、更新和删除 Amazon Lex V2 和 Amazon Lex 资源，以及运行 Amazon Lex V2 和 Amazon Lex 对话机器人。

权限详细信息

该策略包含以下权限：

- lex：向主体授予对 Amazon Lex V2 和 Amazon Lex 模型构建和运行时服务中的所有操作的读写权限。
- cloudwatch— 允许委托人查看 Amazon CloudWatch 指标和警报。
- iam — 允许主体创建和删除服务相关角色、传递角色以及为角色附加和分离策略。Amazon Lex 操作的权限仅限于“lex.amazonaws.com”，而 Amazon Lex V2 操作的权限仅限于“lexv2.amazonaws.com”。
- kendra — 允许主体列出 Amazon Kendra 索引。
- kms — 允许主体描述 AWS KMS 密钥和别名。
- lambda — 允许主体列出 AWS Lambda 函数并管理附加到任何 Lambda 函数的权限。
- polly：允许主体描述 Amazon Polly 的声音并合成话语。

```
{
    "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 托管策略： AmazonLexReplicationPolicy

您不能将 AmazonLexReplicationPolicy 附加到自己的 IAM 实体。本政策附属于服务相关角色，允许 Amazon Lex V2 代表您执行操作。有关更多信息，请参阅 对 Amazon Lex V2 使用服务相关角色。

此政策授予管理权限，允许 Amazon Lex V2 代表您跨区域复制 AWS 资源。您可以附加此策略以允许角色轻松复制资源，包括机器人、区域设置、版本、别名、意图、插槽类型、插槽类型和自定义词汇表。

权限详细信息

该策略包含以下权限。

- lex— 允许委托人复制其他区域的资源。
- iam— 允许委托人从 IAM 传递角色。这是必需的，这样 Amazon Lex V2 才有权在其他区域复制资源。

```
{
    "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 更新了托 AWS 管策略

查看自该服务开始跟踪这些更改以来对 Amazon Lex V2 AWS 托管政策的更新的详细信息。要获得有关此页面更改的自动提示，请订阅 Amazon Lex V2 Amazon Lex V2 文档历史记录 页面上的 RSS 源。

更改	描述	日期
AmazonLexReadOnly – 对现有策略的更新	Amazon Lex V2 添加了新的权限，允许对机器人资源的副本进行只读访问。	2024年5月10日
AmazonLexFullAccess – 更新了现有策略	Amazon Lex V2 添加了新的权限，允许将机器人资源复制到其他区域。	2024 年 4 月 16 日

更改	描述	日期
AmazonLexFullAccess – 更新了现有策略	Amazon Lex V2 添加了新的权限，允许将机器人资源复制到其他区域。	2024 年 1 月 31 日
AmazonLexReplicationPolicy：新策略	Amazon Lex V2 添加了一项新政策，允许将机器人资源复制到其他区域。	2024 年 1 月 31 日
AmazonLexReadOnly – 更新了现有策略	Amazon Lex V2 添加了新权限，允许对自定义词汇项目列表进行只读访问。	2022 年 11 月 29 日
AmazonLexFullAccess – 更新了现有策略	Amazon Lex V2 添加了新的权限，允许对 Amazon Lex V2 模型构建服务操作进行只读访问。	2021 年 8 月 18 日
AmazonLexReadOnly – 对现有策略的更新	Amazon Lex V2 添加了新的权限，允许对 Amazon Lex V2 自动聊天机器人设计器操作进行只读访问。	2021 年 12 月 1 日
AmazonLexFullAccess – 更新了现有策略	Amazon Lex V2 添加了新的权限，允许对 Amazon Lex V2 模型构建服务操作进行只读访问。	2021 年 8 月 18 日
AmazonLexReadOnly – 对现有策略的更新	Amazon Lex V2 添加了新的权限，允许对 Amazon Lex V2 模型构建服务操作进行只读访问。	2021 年 8 月 18 日
AmazonLexRunBots仅限-更新现有政策	Amazon Lex V2 添加了新的权限，允许对 Amazon Lex V2 运行时服务操作进行只读访问。	2021 年 8 月 18 日

更改	描述	日期
Amazon Lex V2 开始跟踪更改	Amazon Lex V2 开始跟踪对其 AWS 托管式策略的更改。	2021 年 8 月 18 日

对 Amazon Lex V2 使用服务相关角色

Amazon Lex V2 使用 AWS Identity and Access Management (IAM) 服务相关角色。服务相关角色是一种独特类型的 IAM 角色，与 Amazon Lex V2 直接相关。服务相关角色由 Amazon Lex V2 预定义，包括该服务代表您调用其他 AWS 服务所需的所有权限。

服务相关角色可让您更轻松地设置 Amazon Lex V2，因为您不必手动添加必要的权限。Amazon Lex V2 定义其服务相关角色的权限，并且除非另有其他定义，否则只有 Amazon Lex V2 可以代入该角色。定义的权限包括信任策略和权限策略，而且权限策略不能附加到任何其它 IAM 实体。

有关支持服务关联的角色的其他服务的信息，请参阅与 IAM 配合使用的亚马逊云科技服务，并查找 服务相关角色（Service-Linked Role）列设为 Yes（是）的服务。选择是，可转到查看该服务的服务相关角色文档的链接。

您必须配置权限，允许 IAM 实体（如用户、组或角色）创建、编辑或删除服务相关角色。有关更多信息，请参阅《IAM 用户指南》中的服务相关角色权限。

只有在先删除相关资源后，才能删除服务相关角色。这样可以保护您的 Amazon Lex V2 资源，因为您不会无意中删除访问这些资源的权限。

主题

- 为 Amazon Lex V2 创建服务相关角色
- 为 Amazon Lex V2 编辑服务相关角色
- 为 Amazon Lex V2 删除服务相关角色
- Amazon Lex V2 的服务相关角色权限
- Amazon Lex V2 服务相关角色支持的区域

为 Amazon Lex V2 创建服务相关角色

您无需手动创建服务相关角色，因为当您在、或 API 中执行相关操作（详情请参阅Amazon Lex V2 的服务相关角色权限）时，Amazon Lex V2 会为您创建服务相关角色。 AWS Management Console AWS CLI AWS

如果您删除此服务相关角色，然后需要再次创建角色，可以使用相同流程在账户中创建新角色。

为 Amazon Lex V2 编辑服务相关角色

Amazon Lex V2 不允许您编辑服务相关角色。在创建服务相关角色后，您将无法更改角色的名称，因为可能有多种实体引用该角色。但是，您可以使用 IAM 编辑角色的描述。有关更多信息，请参阅《IAM 用户指南》中的编辑服务相关角色。

为 Amazon Lex V2 删除服务相关角色

如果不再需要使用某个需要服务相关角色的功能或服务，我们建议您删除该角色。这样就没有未被主动监控或维护的未使用实体。但是，必须先清除服务相关角色的资源，然后才能手动删除它。

> **Note**
>
> 如果当您试图删除资源时 Amazon Lex V2 服务正在使用该角色，则删除操作可能会失败。如果发生这种情况，请等待几分钟后重试。

要查看在 Amazon Lex V2 中删除特定服务相关角色的资源的步骤，请参阅中有关该角色的特定部分。Amazon Lex V2 的服务相关角色权限

使用 IAM 手动删除服务相关角色

删除与服务相关角色相关的资源后，使用 IAM 控制台 AWS CLI、或 AWS API 删除该角色。有关更多信息，请参见《IAM 用户指南》中的删除服务相关角色。

Amazon Lex V2 的服务相关角色权限

Amazon Lex V2 使用带有以下前缀的服务相关角色。

主题

- AWSServiceRoleForLexV2Bots_
- AWSServiceRoleForLexV2Channels_
- AWSServiceRoleForLexV2Replication

AWSServiceRoleForLexV2Bots_

AWSServiceRoleForLexV2Bots_ 角色授予将您的机器人连接到其他所需服务的权限。此角色包括允许 lexv2.amazonaws.com 服务担任该角色的信任策略，并包括执行以下操作的权限。

- 使用 Amazon Polly 在该操作支持的所有亚马逊 Lex V2 资源上合成语音。
- 如果将机器人配置为使用 Amazon Comprehend 情绪分析，则可以在该操作支持的所有 Amazon Lex V2 资源上检测情绪。
- 如果将机器人配置为将音频日志存储在 S3 存储桶中，请将对象放在指定的存储桶中。
- 如果将机器人配置为存储音频和文本日志，请在中创建日志流并将日志放入指定的日志组。
- 如果将机器人配置为使用 AWS KMS 密钥加密数据，请生成特定的数据密钥。
- 如果将机器人配置为使用该`KendraSearchIntent`意图，请查询对指定 Amazon Kendra 索引的访问权限。

创建角色

每次创建机器人时，Amazon Lex V2 都会在您的账户中创建一个带有随机后缀的新 AWSServiceRoleForLexV2Bots _ 角色。当您向机器人添加其他功能时，Amazon Lex V2 会修改该角色。例如，如果您将 Amazon Comprehend 情绪分析添加到机器人中，Amazon Lex V2 会向服务角色添加操作权限`lex:DetectSentiment`。

删除角色

1. 登录 AWS Management Console 并打开 Amazon Lex 控制台，网址为 https://console.aws.amazon.com/lex/。
2. 在左侧导航窗格中，选择 Bot s，然后选择要删除其服务相关角色的机器人。
3. 选择任何版本的机器人。
4. I AM 权限运行时角色位于版本详细信息中。
5. 返回机器人页面，选择要删除的机器人旁边的单选按钮。
6. 选择 “操作”，然后选择 “删除”。
7. 按照删除服务相关角色中的步骤删除该 IAM 角色。

AWSServiceRoleForLexV2Channels_

AWSServiceRoleForLexV2Channels_ 角色允许列出账户中的机器人以及为机器人调用对话 API。该角色包括一项信任策略，允许channels.lexv2.amazonaws.com服务担任该角色。如果将机器人配置为使用通道与消息服务通信，则 AWSServiceRoleForLexV2Channels _ 角色权限策略允许 Amazon Lex V2 完成以下操作。

- 列出账户中所有机器人的权限。

- 识别文本、获取会话并将会话权限设置为指定的机器人别名。

创建角色

当您创建渠道集成以在消息平台上部署机器人时，Amazon Lex V2 会在您的账户中为每个频道创建一个带有随机后缀的新服务相关角色。

删除角色

1. 登录 AWS Management Console 并打开 Amazon Lex 控制台，网址为 https://console.aws.amazon.com/lex/。
2. 在左侧导航窗格中，选择 “机器人”。
3. 选择一个机器人。
4. 在左侧导航窗格中，选择部署下的渠道集成。
5. 选择要删除其服务相关角色的频道。
6. I AM 权限运行时角色位于常规配置中
7. 选择 “删除”，然后再次选择 “删除” 以删除该频道。
8. 按照删除服务相关角色中的步骤删除该 IAM 角色。

AWSServiceRoleForLexV2Replication

该 AWSServiceRoleForLexV2Replication 角色允许在第二个区域复制机器人。此角色包括允许复制.lexv2.amazonaws.com 服务担任该角色的信任策略，还包括AmazonLexReplicationPolicy AWS 托管策略，该策略允许执行以下操作。

- 将机器人 IAM 角色传递给副本机器人，以重新复制副本机器人的相应权限。
- 在其他地区创建和管理机器人和机器人资源（版本、别名、意图、插槽、自定义词汇表等）。

创建角色

当您为机器人启用全球弹性时，Amazon Lex V2 会在您的账户中创建 AWSServiceRoleForLexV2Replication 与服务相关的角色。确保您拥有正确的权限来授予 Amazon Lex V2 服务创建服务相关角色的权限。

删除使用的 Amazon Lex V2 资源， AWSServiceRoleForLexV2Replication 以便您可以删除该角色

1. 登录 AWS Management Console 并打开 Amazon Lex 控制台，网址为 https://console.aws.amazon.com/lex/。
2. 选择已启用全球弹性的机器人。
3. 在 “部署” 下选择 “全球弹性”。
4. 选择 “禁用全局弹性”。
5. 对所有启用了全球弹性的机器人重复该过程。
6. 按照删除服务相关角色中的步骤删除该 IAM 角色。

Amazon Lex V2 服务相关角色支持的区域

Amazon Lex V2 支持在该服务可用的所有区域中使用服务相关角色。有关更多信息，请参阅AWS 区域和端点。

Amazon Lex V2 身份和访问问题排查

您可以使用以下信息来帮助诊断和修复在使用 Amazon Lex V2 和 IAM 时可能遇到的常见问题。

主题

- 我无权在 Amazon Lex V2 中执行操作
- 我无权执行 iam：PassRole
- 我是管理员并需要允许其他人访问 Amazon Lex V2
- 对用户授予编程访问权限
- 我想允许 AWS 账户之外的用户访问我的 Amazon Lex V2 资源

我无权在 Amazon Lex V2 中执行操作

如果 AWS Management Console 告诉您您无权执行某项操作，则必须联系管理员寻求帮助。管理员是向您提供登录凭证的人。

当 `mateojackson` IAM 用户尝试使用控制台查看有关虚构 *`my-example-widget`* 资源的详细信息，但不拥有虚构 `lex:`*`GetWidget`* 权限时，会发生以下示例错误。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 lex:GetWidget on resource: my-example-widget
```

在这种情况下，Mateo 请求他的管理员更新其策略，以允许他使用 `lex:GetWidget` 操作访问 *my-example-widget* 资源。

我无权执行 iam：PassRole

如果您收到一个错误，指明您无权执行 `iam:PassRole` 操作，则必须更新您的策略以允许您将角色传递给 Amazon Lex V2。

有些 AWS 服务 允许您将现有角色传递给该服务，而不是创建新的服务角色或服务相关角色。为此，您必须具有将角色传递到服务的权限。

当名为 `marymajor` 的 IAM 用户尝试使用控制台在 Amazon Lex V2 中执行操作时，会发生以下示例错误。但是，服务必须具有服务角色所授予的权限才可执行此操作。Mary 不具有将角色传递到服务的权限。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

在这种情况下，必须更新 Mary 的策略以允许她执行 `iam:PassRole` 操作。

如果您需要帮助，请联系您的 AWS 管理员。您的管理员是提供登录凭证的人。

我是管理员并需要允许其他人访问 Amazon Lex V2

要允许其他人访问 Amazon Lex V2，您必须为需要访问权限的人员或应用程序创建一个 IAM 实体（用户或角色）。它们将使用该实体的凭证访问 AWS。然后，您必须将策略附加到该实体，以便在 Amazon Lex V2 中为其授予正确的权限。

要立即开始使用，请参阅《IAM 用户指南》中的创建您的第一个 IAM 委派用户和组。

对用户授予编程访问权限

如果用户想在 AWS 外部进行交互，则需要编程访问权限 AWS Management Console。授予编程访问权限的方式取决于正在访问的用户类型 AWS。

要向用户授予编程式访问权限，请选择以下选项之一。

哪个用户需要编程式访问权限？	目的	方式
人力身份 （在 IAM Identity Center 中管理的用户）	使用临时证书签署向 AWS CLI、 AWS 软件开发工具包或 AWS API 发出的编程请求。	按照您希望使用的界面的说明进行操作。 • 有关的 AWS CLI，请参阅《AWS Command Line Interface 用户指南》 AWS IAM Identity Center中的 “配置 AWS CLI 要使用”。 • 有关 AWS 软件开发工具包、工具和 AWS API，请参阅《软件开发工具包和AWS工具参考指南》中的 IAM 身份中心身份验证。
IAM	使用临时证书签署向 AWS CLI、 AWS 软件开发工具包或 AWS API 发出的编程请求。	按照 IAM 用户指南中的将临时证书与 AWS 资源配合使用中的说明进行操作。
IAM	（不推荐使用） 使用长期凭证签署向 AWS CLI、 AWS 软件开发工具包或 AWS API 发出的编程请求。	按照您希望使用的界面的说明进行操作。 • 有关信息 AWS CLI，请参阅用户指南中的使用 IAM 用户证书进行身份验证。AWS Command Line Interface • 有关 AWS SDK 和工具，请参阅 S AWS DK 和工具参考指南中的使用长期凭证进行身份验证。 • 有关 AWS API，请参阅 IAM 用户指南中的管理 IAM 用户的访问密钥。

我想允许 AWS 账户之外的用户访问我的 Amazon Lex V2 资源

您可以创建一个角色，以便其他账户中的用户或您组织外的人员可以使用该角色来访问您的资源。您可以指定谁值得信赖，可以担任角色。对于支持基于资源的策略或访问控制列表（ACL）的服务，您可以使用这些策略向人员授予对您的资源的访问权。

要了解更多信息，请参阅以下内容：

- 要了解 Amazon Lex V2 是否支持这些功能，请参阅 Amazon Lex V2 如何与 IAM 协同工作。
- 要了解如何提供对您拥有的资源的访问权限 AWS 账户 ，请参阅 IAM 用户指南中的向您拥有 AWS 账户 的另一个 IAM 用户提供访问权限。
- 要了解如何向第三方提供对您的资源的访问权限 AWS 账户，请参阅 IAM 用户指南中的向第三方提供访问权限。 AWS 账户
- 要了解如何通过联合身份验证提供访问权限，请参阅《IAM 用户指南》中的为经过外部身份验证的用户（联合身份验证）提供访问权限。
- 要了解使用角色和基于资源的策略进行跨账户访问的区别，请参阅 IA M 用户指南中的跨账户资源访问。

Amazon Lex V2 中的日志记录和监控

监控是保持 Amazon Lex V2 和您的其他 AWS 解决方案的可靠性、可用性和性能的重要方面。AWS 提供以下监控工具来监控 Amazon Lex V2、在出现错误时进行报告并在适当的时候采取自动化措施：

- Amazon 会实时 CloudWatch监控您的 AWS 资源和您运行 AWS 的应用程序。您可以收集和跟踪指标，创建自定义的控制平面，以及 设置警报以在指定的指标达到您指定的阈值时通知您或采取措施。例如，您可以 CloudWatch 跟踪您的 Amazon EC2 实例的 CPU 使用率或其他指标，并在需要时自动启动新实例。有关更多信息，请参阅 Amazon CloudWatch 用户指南。
- AWS CloudTrail捕获由您的账户或代表您的 AWS 账户进行的 API 调用和相关事件，并将日志文件传输到您指定的 Amazon S3 存储桶。您可以识别哪些用户和帐户拨打了电话 AWS、发出呼叫的源 IP 地址以及呼叫发生的时间。有关更多信息，请参阅 AWS CloudTrail 《用户指南》。

Amazon Lex V2 的合规性验证

作为多项合规计划的一部分，第三方审计师对 Amazon Lex V2 的安全与 AWS 合规性进行评估。Amazon Lex V2 是一项符合 HIPAA 要求的服务。它符合 PCI、SOC 和 ISO 标准。

要了解是否属于特定合规计划的范围，请参阅AWS 服务 “按合规计划划分的范围” ”，然后选择您感兴趣的合规计划。 AWS 服务 有关一般信息，请参阅AWS 合规计划AWS。

您可以使用下载第三方审计报告 AWS Artifact。有关更多信息，请参阅中的 “下载报告” 中的 “ AWS Artifact。

您在使用 AWS 服务 时的合规责任取决于您的数据的敏感性、贵公司的合规目标以及适用的法律和法规。 AWS 提供了以下资源来帮助实现合规性：

- 安全与合规性快速入门指南 — 这些部署指南讨论了架构注意事项，并提供了在这些基础上 AWS 部署以安全性和合规性为重点的基准环境的步骤。
- 在 A@@ mazon Web Services 上构建 HIPAA 安全与合规架构 — 本白皮书描述了各公司如何使用 AWS 来创建符合 HIPAA 资格的应用程序。

> **Note**
>
> 并非所有 AWS 服务 人都符合 HIPAA 资格。有关更多信息，请参阅符合 HIPAA 要求的服务参考。

- AWS 合规资源AWS — 此工作簿和指南集可能适用于您所在的行业和所在地区。
- AWS 客户合规指南 — 从合规角度了解责任共担模式。这些指南总结了保护的最佳实践， AWS 服务并将指南映射到跨多个框架（包括美国国家标准与技术研究院 (NIST)、支付卡行业安全标准委员会 (PCI) 和国际标准化组织 (ISO)）的安全控制。
- 使用AWS Config 开发人员指南中的规则评估资源 — 该 AWS Config 服务评估您的资源配置在多大程度上符合内部实践、行业准则和法规。
- AWS Security Hub— 这 AWS 服务 可以全面了解您的安全状态 AWS。Security Hub 通过安全控件评估您的 AWS 资源并检查其是否符合安全行业标准和最佳实践。有关受支持服务及控件的列表，请参阅 Security Hub 控件参考。
- Amazon GuardDuty — 它通过监控您的 AWS 账户环境中是否存在可疑和恶意活动，来 AWS 服务检测您的工作负载、容器和数据面临的潜在威胁。 GuardDuty 通过满足某些合规性框架规定的入侵检测要求，可以帮助您满足各种合规性要求，例如 PCI DSS。
- AWS Audit Manager— 这 AWS 服务 可以帮助您持续审计 AWS 使用情况，从而简化风险管理以及对法规和行业标准的合规性。

Amazon Lex V2 中的恢复功能

AWS 全球基础设施是围绕 AWS 区域和可用区构建的。 AWS 区域提供多个物理隔离和隔离的可用区，这些可用区通过低延迟、高吞吐量和高度冗余的网络相连。利用可用区，您可以设计和操作在可用区之间无中断地自动实现失效转移的应用程序和数据库。与传统的单个或多个数据中心基础设施相比，可用区具有更高的可用性、容错性和可扩展性。

有关 AWS 区域和可用区的更多信息，请参阅AWS 全球基础设施。

除了 AWS 全球基础设施外，Amazon Lex V2 还提供多项功能来帮助支持您的数据弹性和备份需求。

> **Note**
>
> 有关 Amazon Lex V2 中的全球弹性（允许您按预先确定的配对在第二个区域创建复制的机器人）的更多信息，请参阅全球弹性。

Amazon Lex V2 中的基础设施安全性

作为一项托管式服务，Amazon Lex V2 由 Amazon Web Services：安全流程概览白皮书中所述的 AWS 全球网络安全流程提供保护。

您可以使用 AWS 已发布的 API 调用通过网络访问 Amazon Lex V2。客户端必须支持传输层安全性协议 (TLS) 1.0 或更高版本。建议使用 TLS 1.2 或更高版本。客户端还必须支持具有完全向前保密（PFS）的密码套件，例如 Ephemeral Diffie-Hellman（DHE）或 Elliptic Curve Ephemeral Diffie-Hellman（ECDHE）。大多数现代系统（如 Java 7 及更高版本）都支持这些模式。

此外，必须使用访问密钥 ID 和与 IAM 主体关联的秘密访问密钥来对请求进行签名。或者，您可以使用 AWS Security Token Service（AWS STS）生成临时安全凭证来对请求进行签名。

Amazon Lex V2 和接口 VPC 端点 (AWS PrivateLink)

您可以通过创建接口 VPC 端点在 VPC 与 Amazon Lex V2 之间建立私有连接。接口终端节点由一项技术提供支持 AWS PrivateLink，该技术使您无需互联网网关、NAT 设备、VPN 连接或 Di AWS rect Connect 连接即可私密访问 Amazon Lex V2 API。VPC 中的实例即使没有公有 IP 地址也可与 Amazon Lex V2 API 进行通信。VPC 与 Amazon Lex V2 之间的流量不会脱离 Amazon 网络。

每个接口端点均由子网中的一个或多个弹性网络接口表示。

有关更多信息，请参阅 Amazon VPC 用户指南中的接口 VPC 终端节点 (AWS PrivateLink)。

Amazon Lex V2 VPC 端点注意事项

在为 Amazon Lex V2 设置接口 VPC 端点之前，请务必查看《Amazon VPC 用户指南》中的接口端点属性和限制。

Amazon Lex V2 支持从 VPC 调用其所有 API 操作。

为 Amazon Lex V2 创建接口 VPC 端点

您可以使用亚马逊 VPC 控制台或 AWS Command Line Interface (AWS CLI) 为 Amazon Lex V2 服务创建 VPC 终端节点。有关更多信息，请参阅《Amazon VPC 用户指南》中的创建接口端点。

使用以下服务名称为 Amazon Lex V2 创建 VPC 端点：

- com.amazonaws.*region*.models-v2-lex
- com.amazonaws.*region*.runtime-v2-lex

如果为端点启用私有 DNS，则可以使用其对于该区域的默认 DNS 名称，向 Amazon Lex V2 发送 API 请求，例如 `runtime-v2-lex.us-east-1.amazonaws.com`。

有关更多信息，请参阅《Amazon VPC 用户指南》中的通过接口端点访问服务。

为 Amazon Lex V2 创建 VPC 端点策略

您可以为 VPC 端点附加控制对 Amazon Lex V2 的访问的端点策略。该策略指定以下信息：

- 可执行操作的主体。
- 可执行的操作。
- 可对其执行操作的资源。

有关更多信息，请参阅《Amazon VPC 用户指南》中的使用 VPC 端点控制对服务的访问。

示例：Amazon Lex V2 操作的 VPC 端点策略

以下是适用于 Amazon Lex V2 的端点策略示例。当附加到端点时，此策略会向所有主体授予在所有资源上对列出的 Amazon Lex V2 操作的访问权限。

```
{
```

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

指南和最佳实践

请参阅以下准则和最佳实践，以优化您的机器人的行为以及与客户的互动。

签署请求

API 参考中的所有 Amazon Lex V2 模型构建和运行时请求都使用签名 V4 进行验证。有关验证请求的更多信息，请参阅 AWS 一般参考 中的签名版本 4 签名流程。

保护机密信息

运行时 API 操作 RecognizeText 和 RecognizeUtterance 将会话 ID 作为必填参数。开发人员可将此设置为满足 API 中所述约束的任何值。建议您不要使用此参数发送任何机密信息（如用户登录名、电子邮件或身份证号）。此 ID 主要用于唯一标识与机器人的对话。

从用户言语中获取插槽值

Amazon Lex V2 使用您在插槽类型定义中提供的枚举值来训练其机器学习模型。假设您使用以下示例言语定义了名为 `GetPredictionIntent` 的意图：

```
"Tell me the prediction for {sign}"
```

其中 {sign} 是一个自定义类型 `ZodiacSign` 的插槽，该插槽有 12 个枚举值，从 `Aries` 到 `Pisces`。假设用户说“告诉我对 earth 的预测”：

- 如果您执行以下某种操作，Amazon Lex V2 会推断“earth”是一个 ZodiacSign 值：
  - 通过 CreateSlotType 操作将 `valueSelectionStrategy` 字段设置为 `ORIGINAL_VALUE`
  - 在控制台中选择扩展值
- 如果您通过执行以下操作之一将识别范围限制为所定义的插槽类型的值，Amazon Lex V2 将无法识别“earth”值：
  - 通过 `CreateSlotType` 操作将 `valueSelectionStrategy` 字段设置为 `TOP_RESOLUTION`
  - 在控制台中选择限制使用插槽值和同义词

当您为插槽值定义同义词时，它们会被识别为与插槽值相同。但是，返回的是插槽值而不是同义词。

由于 Amazon Lex V2 会将此值传递给您的客户端应用程序或 Lambda 函数，因此在履行活动中使用插槽值之前，您应检查这些值是否有效。

当 Amazon Lex V2 调用 Lambda 函数或返回与您的客户端应用程序的语音交互结果时，无法保证插槽值的大小写。在文本交互中，槽值的大小写将与输入的文本或槽值匹配 (具体取决于 `valueResolutionStrategy` 字段的值)。

插槽值中的首字母缩略词

定义包含首字母缩略词的插槽值时，请使用以下模式：

- 用句点分隔的大写字母 (D.V.D.)
- 用空格分隔的大写字母 (D V D)

日期和时间内置插槽

AMAZON.Date 和 AMAZON.Time 内置插槽类型可同时捕获绝对和相对日期和时间。相对日期和时间以 Amazon Lex V2 收到请求的时间和日期以及处理请求的区域为准。

对于 `AMAZON.Time` 内置插槽类型，如果用户不指定是在上午还是下午，则时间是不确定的。在这种情况下，Amazon Lex V2 将再次提示用户。建议用户提示引发绝对时间。例如，使用这类提示：“您希望披萨什么时候送到？ 您可以说下午 6 点或晚上 6 点。”

避免机器人的训练数据存在歧义

在机器人中提供易混淆的训练数据将减弱 Amazon Lex V2 理解用户输入的能力。假设您的机器人中有两个意图（`OrderPizza` 和 `OrderDrink`），并将“我想订购”作为示例言语。在您构建机器人时，Amazon Lex V2 无法将这种言语映射到特定的意图。因此，当用户在运行时输入此言语时，Amazon Lex V2 无法高度自信地选择意图。

如果您有两个具有相同示例言语的意图，请通过输入上下文帮助 Amazon Lex V2 在运行时区分这两个意图。有关更多信息，请参阅设置意图上下文。

使用 TSTALIASID 别名

- 您的机器人的 TSTALIASID 别名指向草稿版本，并且应仅用于手动测试。Amazon Lex 限制您向机器人的 TSTALIASID 别名发出的运行时请求数量。
- 更新机器人的草稿版本时，Amazon Lex 将根据机器人的 TSTALIASID 别名终止任何客户端应用程序中正在进行的对话。一般而言，不应在生产环境中使用机器人的 TSTALIASID 别名，因为其草稿版本可能会更新。您应该发布一个版本和别名，然后改用它们。
- 更新别名时，Amazon Lex 需要几分钟使更改生效。修改机器人的草稿版本时，TSTALIASID 别名会立即采用更改。

限额

服务配额，也称为限制，是您的 AWS 账户允许的最大服务资源数量。有关更多信息，请参阅 AWS 一般参考中的 AWS 服务限额。

某些服务限额可以调整或增加。请参阅下表中的可调整列，查看是否可以调整限额，并参阅自助服务列以查看是否可以通过服务限额控制台申请限额调整。联系 AWS Support 以增加可调整的配额，但不能通过自助服务进行调整。增加服务限额可能需要几天时间。要在大型项目中增加限额，请将这段时间添加到您的计划中。

> **Note**
>
> 字符限制按照 Unicode 代码单元的数量计算。在大多数情况下，一个 Unicode 字符等同于一个 Unicode 代码单元。某些特殊字符可能大于一个单元，并且不同编码的计数可能有所不同。有关计算字符串长度的更多信息，请参阅此文档。

构建时限额

创建机器人时，将强制执行以下最大限额。

描述	默认	可调整	自助服务
每个 AWS 账户的机器人数	100	是	是
每个 AWS 账户的机器人频道关联数	5000	否	不适用
每个机器人网络的机器人数	5	否	不适用
每个机器人的机器人网络数	25	否	不适用
每个机器人的版本数	100	否	不适用
每个机器人每个区域的意图数	• en-AU/en-GB/en-US：1000	是	不支持

描述	默认	可调整	自助服务
	• 其他各区域：250		
每个机器人每个区域的槽位数	• en-AU/en-GB/en-US：4000 • 其他各区域：2000	否	不适用
每个机器人区域的自定义槽位类型数	• en-AU/en-GB/en-US：250 • 其他各区域：100	否	不适用
每个机器人各区域的自定义槽位类型值数和同义词数	50000	否	不适用
每个机器人各区域的示例言语中的总字符数	• en-AU/en-GB/en-US：2000000 • 其他各区域：200 000	否	不适用
每个机器人别名的频道关联数	10	否	不适用
每个意图的槽位数	100	否	不适用
每个意图的示例言语数	1500	是	是
每个示例言语的字符数	500	否	不适用
文本响应长度	4,000	否	不适用
每个槽位的示例言语数	10	是	是
每个示例槽位言语的字符数	500	否	不适用

描述	默认	可调整	自助服务
每个槽位的提示数	30	否	不适用
每个自定义槽位类型的值和同义词数	10000	否	不适用
每个自定义槽位类型值的字符数	500	否	不适用
频道关联名称中的字符数	100	否	不适用
每个区域中您账户下所有机器人的并发自动聊天机器人设计器分析任务数	10	否	不适用
自定义语法槽类型 XML 文件的大小	100KB	否	不适用

运行时限额

以下最大限额在运行时强制执行。

描述	默认	可调整	自助服务
输入RecognizeText和的文字大小RecognizeUtterance	1024 个字符	否	不适用
`RecognizeUtterance` 操作的语音输入长度	15 秒	是	不支持
`RecognizeUtterance` 标题的大小	16 KB	否	不适用

描述	默认	可调整	自助服务
RecognizeUtterance 的合并请求和会话标头的大小	12 KB	否	不适用
、或StartConversation 的并发文本模式对话的最RecognizeText RecognizeUtterance 大数量TestBotAlias	2	否	不适用
用于其他别名的 RecognizeText 、RecognizeUtterance 或 StartConversation 的并发文本模式对话的最大数量	50	可以	不支持
的并发语音模式对话的最RecognizeUtterance 大数量TestBotAlias	2	否	不适用
用于其他别名的 RecognizeUtterance 的并发语音模式对话的最大数量	125	是	不支持

描述	默认	可调整	自助服务
的并发语音模式对话的最StartConversation 大数量 TestBotAlias	2	否	不适用
用于其他别名的 StartConversation 的并发语音模式对话的最大数量	200	是	不支持
使用时并发会话管理操作的最大数量（PutSession GetSession 、或DeleteSession ）TestBotAlias	2	否	不适用
使用其他别名时并发会话管理操作（PutSession 、GetSession 或 DeleteSession ）的最大数量	50	可以	不支持
Lambda 函数的最大输入大小	12 KB	否	不适用
Lambda 函数的最大输出大小	50 KB	否	不适用
Lambda 函数输出中会话属性的最大大小（base-64 编码之后）	12 KB	否	不适用

描述	默认	可调整	自助服务
Lambda 函数的最大超时时间	30 秒	是	不支持

Amazon Lex V1 到 V2 迁移指南

Amazon Lex V2 控制台和 API 使机器人构建和管理变得更加容易。通过本指南了解在迁移机器人时对 Amazon Lex V2 API 的改进。

您可以通过 Amazon Lex 控制台或 API 迁移机器人。有关更多信息，请参阅 Amazon Lex 开发者指南中的迁移机器人。

Amazon Lex V2 概述

您可以向机器人添加多种语言，从而可以将它们作为单一资源进行管理。简化的信息架构帮助您高效地管理机器人版本。“对话流程”、部分保存机器人配置和批量上传言语等功能为您提供了更大的灵活性。

机器人的多种语言

您可以通过 Amazon Lex V2 API 添加多种语言。您可以单独添加、修改和构建每种语言。插槽类型等资源在语言级别上进行了范围限定。您可以在不同的语言之间快速切换，以比较和完善对话。您可以通过控制台上的一个控制面板来查看所有语言的言语，以便更快地进行分析和迭代。机器人操作员可以通过一个机器人配置来管理所有语言的权限和日志操作。要与 Amazon Lex V2 机器人对话，必须提供一种语言作为运行时参数。有关更多信息，请参阅Amazon Lex V2 支持的语言和区域设置。

简化的信息架构

Amazon Lex V2 API 遵循简化的信息架构 (IA)，其意图和插槽类型仅限于一种语言。您在机器人级别进行版本控制，因此意图和插槽类型等资源不会单独进行版本控制。默认情况下，机器人通过草稿版本创建，该版本是可变的，用于测试更改。您可以根据草稿版本创建带编号的快照，并选择要包含在版本中的语言。机器人中的所有资源（语言、意图和插槽类型）都作为创建机器人版本的一部分进行存档。有关更多信息，请参阅版本。

生成器效率提高

还有其他生成工具和能力，可帮助您更灵活地控制机器人设计流程。

部分配置保存

通过 Amazon Lex V2 API，您可以在开发过程中保存部分更改。例如，您可以保存引用已删除插槽类型的插槽。这种灵活性使您能够保存工作并稍后再使用。您可以在构建机器人之前解析这些更改。在 Amazon Lex V2 中，部分保存可以应用于插槽、版本和别名。

资源重命名

通过 Amazon Lex V2，您可以在资源创建后对其进行重命名。使用资源名称将用户友好型元数据与每个资源关联起来。Amazon Lex V2 API 为每个资源分配唯一的 10 个字符的资源 ID。所有资源都有资源名称。您可以重命名以下资源：

- 机器人
- 意图
- 槽类型
- 槽
- 别名

您可以根据资源 ID 读取并修改您的资源。如果您正在通过 AWS Command Line Interface 或 Amazon Lex V2 API 使用 Amazon Lex V2，则某些命令需要资源 ID。

Lambda 函数管理简化

在 Amazon Lex V2 API 中，您可以为每种语言定义一个 Lambda 函数，而不是为每种意图定义一个函数。Lambda 函数以该语言的别名进行配置，用于对话和履行代码挂钩。您还可以选择为每个意图单独启用或禁用对话和履行代码挂钩。有关更多信息，请参阅使用 AWS Lambda 函数启用自定义逻辑。

粒度设置

Amazon Lex V2 API 将语音和意图分类置信度分数阈值从机器人转移到语言范围。情绪分析标志从机器人作用域移至别名作用域。机器人作用域的会话超时和隐私设置以及别名作用域的对话日志保持不变。

默认回退意图

Amazon Lex V2 API 在您创建语言时会添加默认回退意图。通过它来为您的机器人配置错误处理，而不是使用特定的错误处理提示。

会话变量更新优化

借助 Amazon Lex V2 API，您可以通过 RecognizeText 和 RecognizeUtterance 操作直接更新会话状态，而无需依赖会话 API。

使用 AWS CloudFormation 创建 Amazon Lex V2 资源

Amazon Lex V2 与 AWS CloudFormation 服务集成，该服务可帮助您对 AWS 资源进行建模和设置，以便缩短创建和管理资源与基础设施所需的时间。您可以创建一个描述所需的全部 AWS 资源的模板（例如您的 Amazon Lex V2 聊天机器人），AWS CloudFormation 将为您预置和配置这些资源。

在您使用 AWS CloudFormation 时，可重复使用您的模板来不断地重复设置您的 Amazon Lex V2 资源。描述您的资源一次，然后在多个 AWS 账户 和区域中反复预置相同的资源。

Amazon Lex V2 和 AWS CloudFormation 模板

要为 Amazon Lex V2 和相关服务预置和配置资源，您必须了解 AWS CloudFormation 模板。模板是 JSON 或 YAML 格式的文本文件。这些模板描述要在 AWS CloudFormation 堆栈中调配的资源。如果您不熟悉 JSON 或 YAML，可以在 AWS CloudFormation Designer 的帮助下开始使用 AWS CloudFormation 模板。有关更多信息，请参阅 AWS CloudFormation 用户指南中的什么是 AWS CloudFormation Designer？。

Amazon Lex V2 支持在 AWS CloudFormation 中创建以下资源：

- AWS::Lex::Bot
- AWS::Lex::BotAlias
- AWS::Lex::BotVersion
- AWS::Lex::ResourcePolicy

有关更多信息（包括这些资源的 JSON 和 YAML 模板示例），请参阅《AWS CloudFormation 用户指南》中的 Amazon Lex V2 资源类型参考。

了解有关 AWS CloudFormation 的更多信息

要了解有关 AWS CloudFormation 的更多信息，请参阅以下资源：

- AWS CloudFormation
- AWS CloudFormation 用户指南
- AWS CloudFormation API 参考
- AWS CloudFormation 命令行界面用户指南

Amazon Lex V2 文档历史记录

- 最新文档更新：2024 年 5 月 10 日

下表介绍每一版 Amazon Lex V2 中的重大更改。要获得本文档的更新通知，您可以订阅 RSS 源。

变更	说明	日期
更新托 AWS 管策略	Amazon Lex V2 为AmazonLexReadOnly托管策略添加了新的权限，允许对已复制到其他区域的机器人资源进行读取访问。	2024年5月10日
更新托 AWS 管策略	Amazon Lex V2 为AmazonLexFullAccess托管策略添加了新权限，允许将复制的机器人资源更新到其他区域。	2024年4月15日
区域扩展	Amazon Lex V2 现已在 AWS GovCloud （美国西部）(-us-gov-west 1) 上市。	2024 年 3 月 22 日
更新托 AWS 管策略	Amazon Lex V2 为AmazonLexReplicationPolicy托管策略添加了新权限，允许将复制的机器人资源更新到其他区域。	2024 年 3 月 7 日
新功能	在 Amazon Lex V2 中，您可以使用 fn.Length () 函数来确定字符串值的值长度。有关更多信息，请参阅条件分支-函数。	2024 年 3 月 4 日
更新到功能	Amazon Lex V2 中用于生成人工智能功能的 qnA 内置插槽现已正式发布。有关更多信息，	2024年2月28日

	请参阅利用生成式人工智能优化机器人的创建和性能。	
更新托 AWS 管策略	Amazon Lex V2 为AmazonLexReplicationPolicy托管策略添加了新权限，允许将复制的机器人资源更新到其他区域。	2024年2月28日
更新托 AWS 管策略	Amazon Lex V2 为AmazonLexFullAccess托管策略添加了新的权限，允许将机器人资源复制到其他区域。	2024 年 2 月 8 日
新的托管策略	Amazon Lex V2 添加了一项托管策略，提供在其他区域复制机器人资源的权限。有关更多信息，请参阅AmazonLexReplicationPolicy。	2024 年 2 月 8 日
新功能	在 Amazon Lex V2 中，您可以使用全球弹性将您的机器人复制到第二个 AWS 区域。有关更多信息，请参阅全球弹性。	2024 年 2 月 8 日
新功能	现在，您可以利用 Amazon Lex V2 中的生成式人工智能功能。有关更多信息，请参阅利用生成式人工智能优化机器人的创建和性能。	2023 年 11 月 29 日
新功能	Amazon Lex V2 可以通过选择性日志记录在意图或槽位级别捕获文本和/或音频。有关更多信息，请参阅选择性日志记录。	2023 年 11 月 8 日

新功能	通过 AMAZON.Confirmation ，Amazon Lex V2 可以使用内置槽位确定响应是“是”、“否”、“可能”还是“不知道”。有关更多信息，请参阅内置槽位类型。	2023 年 8 月 17 日
新功能	除了使用分析控制面板查看其他对话指标外，您还可以查看意图和槽位的性能指标。有关更多信息，请参阅分析。	2023 年 7 月 18 日
新功能	您可以通过 Test Workbench 提高机器人的准确性和履行成功率。有关更多信息，请参阅 Test Workbench。	2023 年 6 月 6 日
新功能	您可以根据模板为一些常用垂直行业创建机器人。有关更多信息，请参阅机器人模板。	2023 年 2 月 23 日
新功能	您可以将多个机器人组合成一个机器人网络，以创建一个集成的客户体验。有关更多信息，请参阅机器人网络。	2023 年 2 月 9 日
新功能	Amazon Lex V2 支持海湾阿拉伯语（阿拉伯联合酋长国）、广东话（香港）、芬兰语（芬兰）、挪威语（挪威）、波兰语（波兰）和瑞典语（瑞典）区域设置。有关更多信息，请参阅 Amazon Lex V2 支持的语言和区域设置。	2022 年 12 月 6 日

已更新为 AWS 托管策略	Amazon Lex V2 为AmazonLexReadOnly托管策略添加了新的权限，允许显示自定义词汇项目。	2022 年 11 月 29 日
新功能	Amazon Lex V2 可以通过控制台或 API 自定义语音转文本输出，从而以替代表示形式显示短语或单词。有关更多信息，请参阅创建语音识别的自定义词汇。	2022 年 11 月 7 日
新功能	Amazon Lex V2 可以为项目元素增加权重属性，用来表示在语音识别过程中提升短语的程度。有关更多信息，请参阅语法权重。	2022 年 10 月 28 日
新功能	通过 AMAZON.FreeFormInput ，Amazon Lex V2 可用于从终端用户获取由单词或字符组成的自由形式输入。有关更多信息，请参阅内置槽位类型。	2022 年 10 月 19 日
新功能	Amazon Lex V2 可以在最终的语音到文本输出中以替代表示形式显示短语或单词。有关更多信息，请参阅创建语音识别的自定义词汇。	2022 年 10 月 19 日
新功能	Amazon Lex V2 支持印地语（印度）和荷兰语（荷兰）区域设置。有关更多信息，请参阅 Amazon Lex V2 支持的语言和区域设置。	2022 年 10 月 14 日

新功能	Amazon Lex V2 管理用户输入的方式有所更新。您可以在对话流程中的任意时候选择 Amazon Lex V2 是否接受文本、音频或 DTMF 输入。有关更多信息，请参阅可配置属性。	2022 年 9 月 22 日
新功能	Amazon Lex V2 管理对话流程的方式有所更新。可视化对话生成器是一个拖放式对话生成器，可轻松设计并可视化对话路径。有关更多信息，请参阅可视化对话生成器。	2022 年 9 月 14 日
新功能	Amazon Lex V2 构建复杂槽位的方式有所更新。您可以在槽位中创建复杂的子槽位，以管理复杂对话设计中的意图。有关更多信息，请参阅构建复合槽位。	2022 年 9 月 9 日
新功能	Amazon Lex V2 管理与用户对话路径流程的方式有所更新。您可以通过对对话中的下一步进行排序来创建复杂的对话路径。有关更多信息，请参阅创建对话路径。	2022 年 8 月 17 日
新功能	Amazon Lex V2 管理与用户对话流程的方式有所更新。您可以通过对提示进行排序来创建复杂对话。有关更多信息，请参阅配置提示。	2022 年 7 月 5 日

新功能	Amazon Lex V2 管理与用户对话流程的方式有所更新。您可以根据条件创建复杂对话。有关更多信息，请参阅了解新的对话流程。	2022 年 5 月 3 日
新功能	为内置语法槽位类型添加了行业语法示例。有关更多信息，请参阅行业语法。	2022 年 3 月 22 日
新功能	添加了有关将 Amazon Lex V2 与 Amazon Chime SDK 集成的文档。有关更多信息，请查看 Amazon Chime SDK。	2022 年 3 月 18 日
新功能	Amazon Lex V2 提供语音转录的置信度分数。根据分数来确定用户的正确响应。有关更多信息，请参阅使用语音转录置信度分数。	2022 年 1 月 27 日
新功能	您可以向槽位添加上下文提示和动态提示，以提高机器人的准确性。有关更多信息，请参阅使用提示提高准确性。	2022 年 1 月 13 日
新功能	Amazon Lex V2 增加了对自定义词汇的支持，以改善对音频输入的语音识别。有关更多信息，请参阅创建自定义词汇表以改善语音识别。	2022 年 1 月 12 日
新功能	Amazon Lex V2 现在支持 AWS PrivateLink。有关更多信息，请参阅 VPC 终端节点 (AWS PrivateLink)。	2022 年 1 月 7 日

新功能	Amazon Lex V2 支持加泰罗尼亚语（西班牙）区域设置。有关更多信息，请参阅 Amazon Lex V2 支持的语言和区域设置。	2022 年 1 月 3 日
新功能	您可以根据自己的自定义语法创建槽位类型。有关更多信息，请参阅使用自定义语法槽位类型。	2021 年 12 月 20 日
新功能	AWS CloudFormation 现在支持 Amazon Lex V2。有关更多信息，请参阅 AWS CloudFormation 资源。	2021 年 12 月 20 日
新功能	Amazon Lex V2 支持葡萄牙语（巴西）、葡萄牙语（葡萄牙）和普通话（中国）区域设置。有关更多信息，请参阅 Amazon Lex V2 支持的语言和区域设置。	2021 年 12 月 16 日
新功能	Amazon Lex V2 提供自动聊天机器人设计器的预览，以帮助您从联络中心的转录开始创建聊天机器人。有关更多信息，请参阅使用自动聊天机器人设计器（预览）。	2021 年 12 月 1 日
新功能	现在，您可以使用 spell-by-letter 和 spell-by-word 样式来输入 Amazon Lex V2 难以理解的插槽值。有关更多信息，请参阅使用拼写方式获取槽位值。	2021 年 11 月 19 日

新功能	您可以通过 Amazon Polly 神经文本转语音 (NTTS) 中的语音与用户进行音频对话。有关更多信息，请参阅 Amazon Polly 中的语音。	2021 年 11 月 19 日
新功能	Amazon Lex V2 支持英语（南非）区域设置。有关更多信息，请参阅 Amazon Lex V2 支持的语言和区域设置。	2021 年 11 月 9 日
新功能	Amazon Lex V2 支持德语（奥地利）区域设置。有关更多信息，请参阅 Amazon Lex V2 支持的语言和区域设置。	2021 年 11 月 5 日
新功能	您可以向用户提供更新消息，这些消息在履行函数开始时播放，并在函数运行时定期播放。您还可以创建消息，在函数完成时通知用户履行状态。有关更多信息，请参阅配置履行进度更新。	2021 年 10 月 7 日
区域扩展	Amazon Lex V2 已在非洲（开普敦）(af-south-1) 和亚太地区（首尔）(ap-northeast-2) 推出。	2021 年 9 月 22 日
新功能	您可以查看用户向您的机器人发送的言语的统计信息。有关更多信息，请参阅查看言语统计信息。	2021 年 9 月 22 日
新功能	Amazon Lex V2 支持韩语（韩国）区域设置。有关更多信息，请参阅 Amazon Lex V2 支持的语言和区域设置。	2021 年 9 月 9 日

新功能	Amazon Lex V2 为英国邮政编码提供了一种内置的槽位类型。有关更多信息，请参阅 AMAZON.UK。PostalCode	2021 年 7 月 27 日
新功能	Amazon Lex V2 支持英语（印度）区域设置。有关更多信息，请参阅 Amazon Lex V2 支持的语言和区域设置。	2021 年 7 月 15 日
新功能	Amazon Lex V2 提供了一种工具，可以将机器人从 Amazon Lex V1 迁移到 Amazon Lex V2 API。有关更多信息，请参阅 Amazon Lex 开发者指南中的迁移机器人。	2021 年 7 月 13 日
新功能	Amazon Lex V2 支持在英语（美国）语言中接受单个槽位的多个值。有关更多信息，请参阅在单个槽位中使用多个值。	2021 年 6 月 15 日
新功能	您可以为英语语言构建更大的机器人。有关更多信息，请参阅 配额。	2021 年 6 月 11 日
新功能	通过 Amazon Lex V2 基于资源的策略来管理对机器人和机器人别名的访问权限。有关更多信息，请参阅 Amazon Lex V2 基于资源的策略。	2021 年 5 月 20 日

新功能	Amazon Lex V2 支持导入和导出机器人和机器人区域设置。您可以使用此功能在账户和区域之间复制机器人和机器人 AWS 区域设置。有关更多信息，请参阅导入和导出。	2021 年 5 月 18 日
区域扩展	Amazon Lex V2 已在加拿大（中部）(ca-central-1) 推出。	2021 年 5 月 17 日
新功能	Amazon Lex V2 支持日语（日本）区域设置。有关更多信息，请参阅 Amazon Lex V2 支持的语言和区域设置。	2021 年 4 月 1 日
新功能	Amazon Lex V2 现在支持三种新的内置槽位类型：AMAZON.City 、AMAZON.Country 和 AMAZON.State 。	2021 年 3 月 12 日
新指南	这是 Amazon Lex V2 用户指南的首次发布。	2021 年 1 月 21 日

API 参考

API 参考现在是一个单独的文档。

AWS 术语表

有关最新的 AWS 术语，请参阅《AWS 词汇表参考》中的 AWS 词汇表。

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。