Amazon Lex is an AWS service for building conversational interfaces for applications using voice and text. With Amazon Lex, the same conversational engine that powers Amazon Alexa is now available to PAM. Complete documentation is available here.

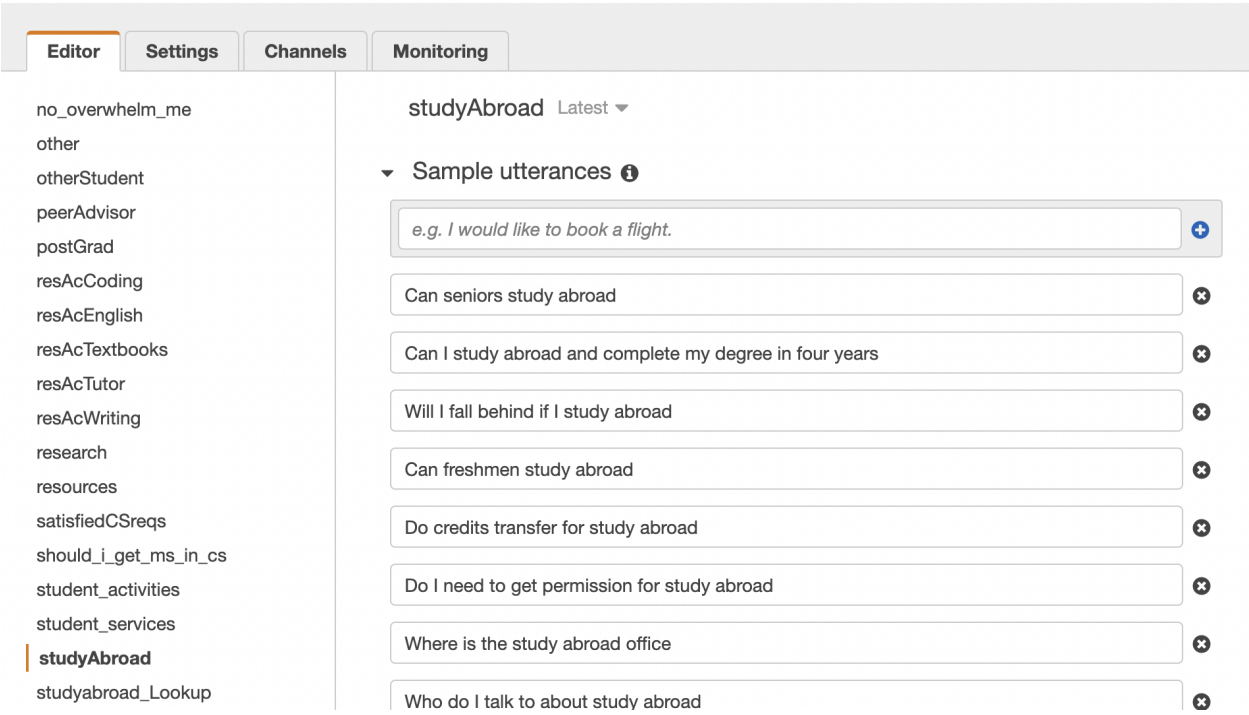
Some of the key features used in PAM include Intents, Slots, Contexts, Response Cards, Error Handling, Confidence Scores and Confirmation Prompts (Code defined in Appendix B).

To programmatically create bots, intents, and slot types, use the model building API operations. You can also use the model building API to manage, update, and delete resources for your bot. The model building API operations include:

- PutBot, PutBotAlias, PutIntent, and PutSlotType to create and update bots, bot aliases, intents, and slot types, respectively.
- CreateBotVersion, CreateIntentVersion, and CreateSlotTypeVersion to create and publish versions of your bots, intents, and slot types, respectively.
- GetBot and GetBots to get a specific bot or a list of bots that you have created, respectively.
- GetIntent and GetIntents to get a specific intent or a list of intents that you have created, respectively.
- GetSlotType and GetSlotTypes to get a specific slot type or a list of slot types that you have created, respectively.
- GetBuiltinIntent, GetBuiltinIntents, and GetBuiltinSlotTypes to get an Amazon Lex built-in intent, a list of Amazon Lex built-in intents, or a list of built-in slot types that you can use in your bot, respectively.
- GetBotChannelAssociation and GetBotChannelAssociations to get an association between your bot and a messaging platform or a list of the associations between your bot and messaging platforms, respectively.
- DeleteBot, DeleteBotAlias, DeleteBotChannelAssociation, DeleteIntent, and DeleteSlotType to remove unneeded resources in your account.

Intents are particular actions that a user may wish to perform. Each bot is built using intents to describe the different kind of actions that can be taken for every user input value. In some ways, intents can be considered the ‘if-then’ function of the language. The bot is able to perform actions (then blocks) only once an intent has been fulfilled (if-statement returns true).

Utterances are sample phrases or exact user values that may be expected to be input into the bot. Intents are built using these utterances as they are able to store a set of utterances upon which they react. Continuing to consider intents as an ‘if-then’ function, utterances can be seen as the values to which the user input must match to create a boolean true result and fulfill the intent.

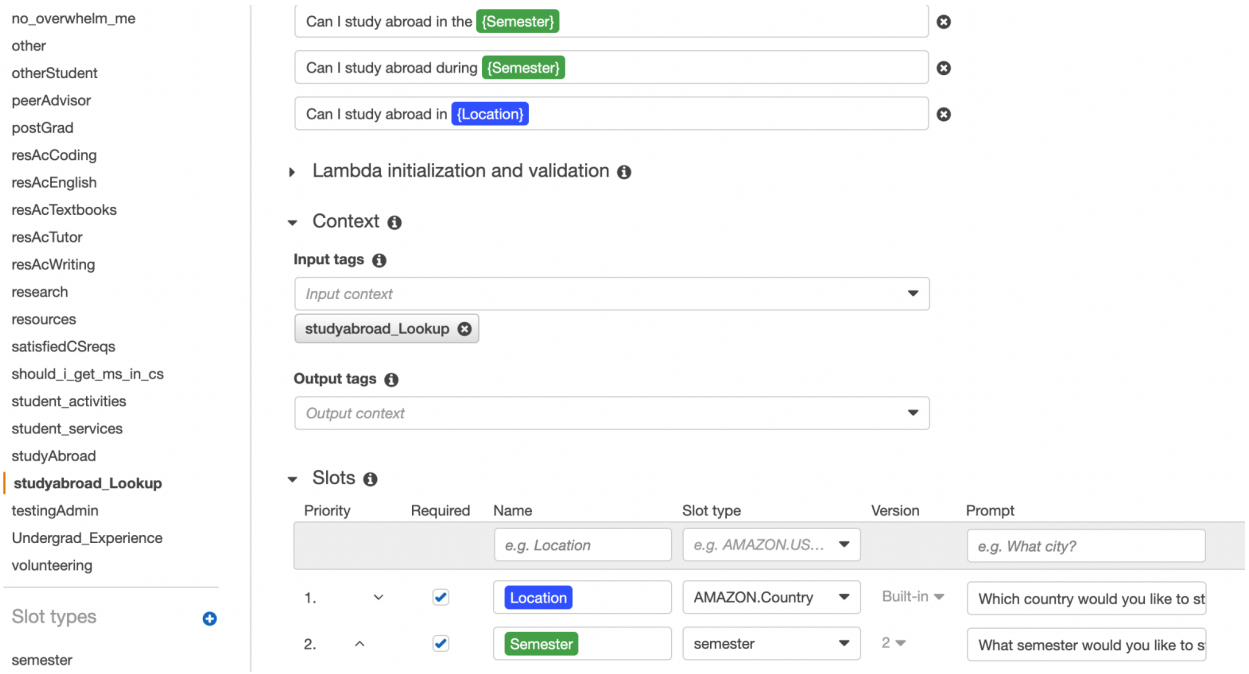


For example, a user is able to input a message such as “Where can I find the Study Abroad office?” in PAM. The Lex bot then stores that input and matches it to all the available utterances in all the intents. It then creates confidence scores for each intent.

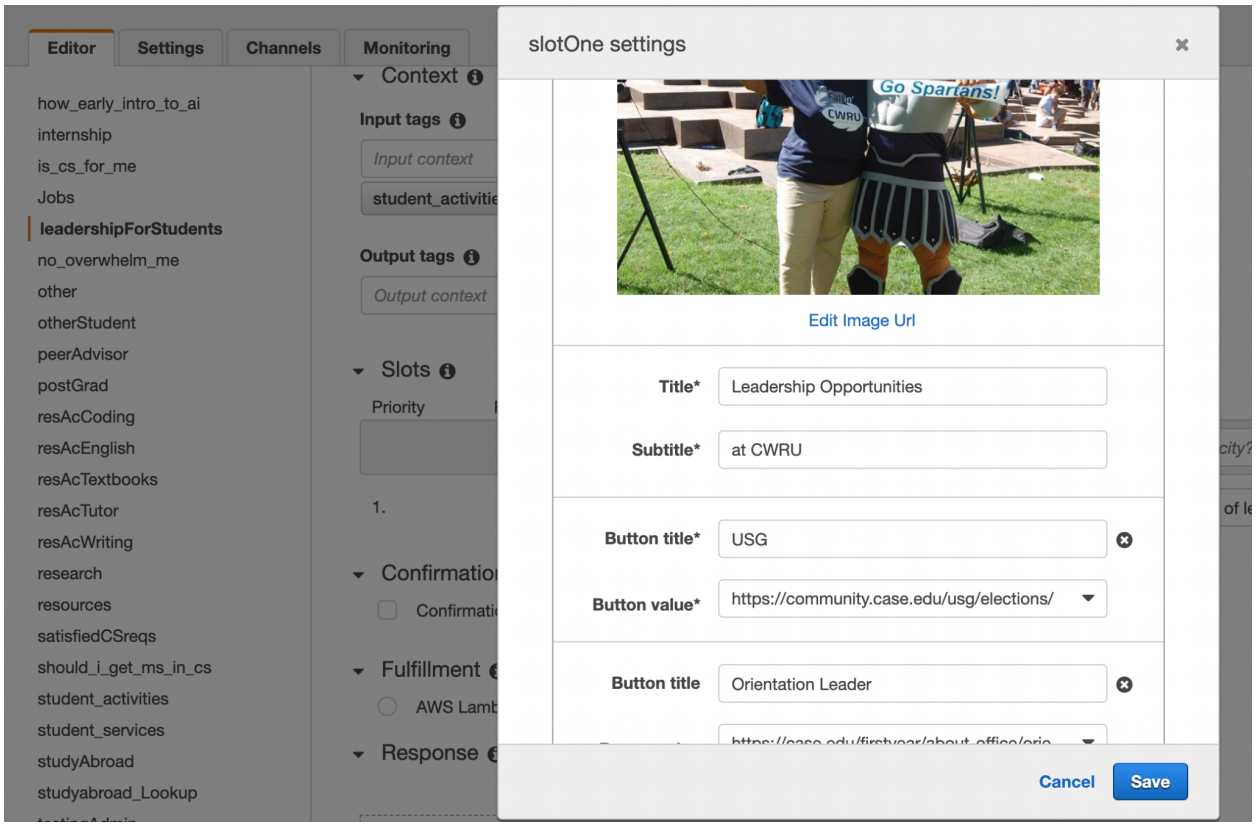
Confidence Scores are the numerically calculated value representations from 0 to 1 of how confident Lex’s natural language processing (NLP) engine is that a certain user input matches a certain utterance within a certain intent. This engine allows users to input values that may be slightly different than the utterances saved in an intent but still fulfill the intent.

Lex uses the highest confidence score to recognize the correct intent (action) that needs to be triggered upon this user input. In our example, the intent in PAM called ‘studyAbroad’ would be fulfilled based on its utterance match and confidence score.

Slots are another feature of intents that help the bot identify certain pieces of user-inputted data that are considered crucial. Intents use slots to catch certain keywords in a user input and store them under formal parameters known as slot type. These values can then be used in generating a response that is specific to the inputted data. In PAM, you can see this is the same ‘studyAbroad’ intent where slot types ‘{Location}’ and ‘{Semester}’ are used to catch user inputs to values for where the user wanted to study abroad and in which semester. Slot can be made required so that the bot asks for these values specifically if the user does not provide them in natural conversation.

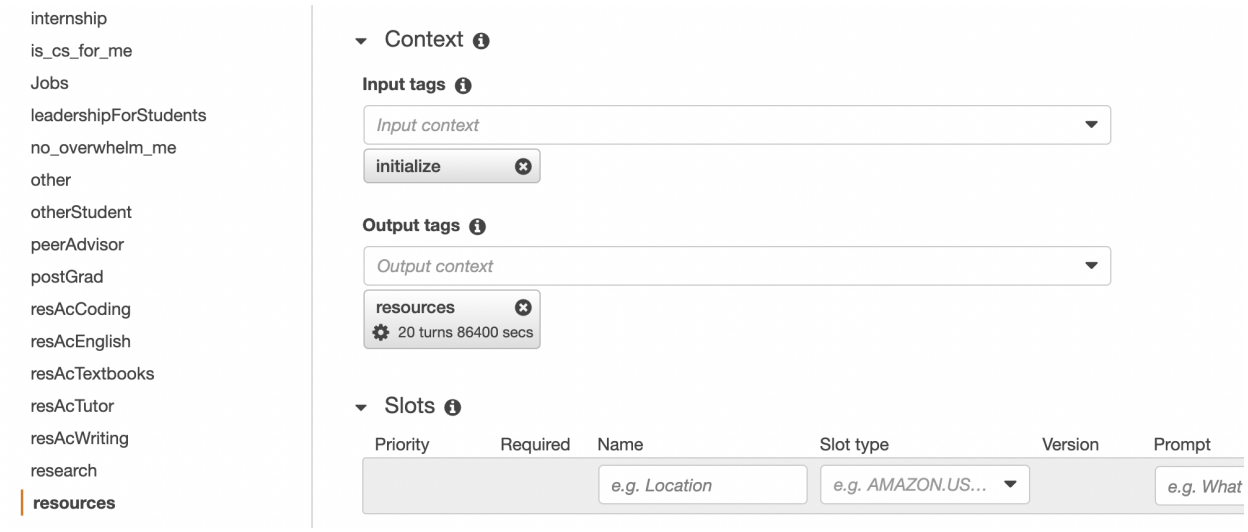


Response Cards are one way PAM is able to respond to user inputs apart from text and voice replies. They are graphical representations and provide buttons that are able to redirect users to different areas of the bot or internet. You can see an example under the intent ‘leadershipForStudents’.



Contexts are state variables that can be associated with an intent when you define a bot. Amazon Lex can trigger intents based on *context*. You configure the contexts for an intent when you create the intent using the console or using the PutIntent operation. You can only use contexts in the English (US) (en-US) locale, and only if you set the `enableModelImprovements` parameter to `true` when you create the bot with the PutBot operation.

Contexts help manage the flow of conversation for the user by guiding them through successive intents. They do this by using input and output context tags on intents. An intent with an input tag is only available for fulfillment once an intent with the same tag as an output context is fulfilled. For example the ‘initialize’ context tag serves as an output tag for the intent ‘hello’ but as an input tag for the context ‘resources’.



An output context is returned to your application in the response from the PostText or PostContent operation, and it is set for the current session. After a context is activated, it stays active for the number of turns or time limit configured when the context was defined.

An *input context* specifies conditions under which an intent can be recognized. An intent can only be recognized during a conversation when all of its input contexts are active. An intent with no input contexts is always eligible for recognition.

Amazon Lex automatically manages the lifecycle of contexts that are activated by fulfilling intents with output contexts. You can also set active contexts in a call to the `PostContent` or `PostText` operation.

You can also set the context of a conversation using the Lambda function for the intent. The output context from Amazon Lex is sent to the Lambda function input event. The Lambda function can send contexts in its response. For more information, see Lambda Function Input Event and Response Format in Appendix B.

In Appendix A you can see ever context currently in PAM and which intents it actvates. Contexts are in green and Intents are in Orange.

Appendix A

Context Trees

1) initialize
    a) hello → eleg_and_req
    b) hello → Undergrad_Experience
    c) hello → career_help
    d) hello → resources
    e) hello → other
    f) hello → peerAdvisor

2) eleg_and_req
    a) eleg_and_req → approveTechnicalelectives
    b) eleg_and_req → course_registration
    c) eleg_and_req → depth_track_diff
    d) eleg_and_req → how_early_intro_to_ai
    e) eleg_and_req → satisfiedCSreqs

3) undergrad_experience
    a) Underfrad_Experience → research
    b) Underfrad_Experience → studyAbroad
    c) Underfrad_Experience → extracurriculars
    d) Underfrad_Experience → student_activities

4) studyabroad_lookup
    a) studyAbroad → studyabroad_lookup

5) student_activities
    a) student_activties → leadershipForStudents
    b) student_activties → volunteering

6) career_help
    a) career_help → Coop
    b) career_help → internship
    c) career_help → postGrad
    d) career_help → Entrepreneurship
    e) career_help → Jobs
    f) career_help → alumni_network

7) resources
    a) resources → academic_resources
    b) resources → otherStudent
    c) resources → testingAdmin
    d) resources → freeTech

8) academic_resources
    a) academic_resources → resAcCoding
    b) academic_resources → resAcEnglish
    c) academic_resources → resAcTextbooks
    d) academic_resources → resAcTutor
    e) academic_resources → resAcWriting

9) other
    a) other → classList
    b) other → differencebwBSBA
    c) other → is_cs_for_me
    d) other → no_overwhelm_me
    e) other → should_i_get_ms_in_cs

Appendix B

Lambda Funtion Input Event Format

```
{
  "currentIntent": {
    "name": "intent-name",
    "nluIntentConfidenceScore": score,
    "slots": {
      "slot name": "value",
      "slot name": "value"
    },
    "slotDetails": {
      "slot name": {
        "resolutions" : [
          { "value": "resolved value" },
          { "value": "resolved value" }
        ],
        "originalValue": "original text"
      },
      "slot name": {
        "resolutions" : [
          { "value": "resolved value" },
          { "value": "resolved value" }
        ],
        "originalValue": "original text"
      }
    },
    "confirmationStatus": "None, Confirmed, or Denied (intent
confirmation, if configured)"
  },
  "alternativeIntents": [
    {
      "name": "intent-name",
      "nluIntentConfidenceScore": score,
      "slots": {
        "slot name": "value",
        "slot name": "value"
      },
```

```
      "slotDetails": {
        "slot name": {
          "resolutions" : [
            { "value": "resolved value" },
            { "value": "resolved value" }
          ],
          "originalValue": "original text"
        },
        "slot name": {
          "resolutions" : [
            { "value": "resolved value" },
            { "value": "resolved value" }
          ],
          "originalValue": "original text"
        }
      },
      "confirmationStatus": "None, Confirmed, or Denied (intent confirmation, if configured)"
    }
  ],
  "bot": {
    "name": "bot name",
    "alias": "bot alias",
    "version": "bot version"
  },
  "userId": "User ID specified in the POST request to Amazon Lex.",
  "inputTranscript": "Text used to process the request",
  "invocationSource": "FulfillmentCodeHook or DialogCodeHook",
  "outputDialogMode": "Text or Voice, based on ContentType request header in runtime API request",
  "messageVersion": "1.0",
  "sessionAttributes": {
    "key": "value",
    "key": "value"
  },
  "requestAttributes": {
    "key": "value",
    "key": "value"
  },
  "recentIntentSummaryView": [
```

```
    {
        "intentName": "Name",
        "checkpointLabel": Label,
        "slots": {
          "slot name": "value",
          "slot name": "value"
        },
        "confirmationStatus": "None, Confirmed, or Denied
(intent confirmation, if configured)",
        "dialogActionType": "ElicitIntent, ElicitSlot,
ConfirmIntent, Delegate, or Close",
        "fulfillmentState": "Fulfilled or Failed",
        "slotToElicit": "Next slot to elicit"
    }
  ],
   "sentimentResponse": {
      "sentimentLabel": "sentiment",
      "sentimentScore": "score"
   },
   "kendraResponse": {
       Complete query response from Amazon Kendra
   },
   "activeContexts": [
        {
            "timeToLive": {
                "timeToLiveInSeconds": seconds,
                "turnsToLive": turns
            },
            "name": "name",
            "parameters": {
                "key name": "value"
            }
        }
    ]

}
```

Note the following additional information about the event fields:

- currentIntent – Provides the intent name, slots, slotDetails and confirmationStatus fields.

nluIntentConfidenceScore is the confidence that Amazon Lex has that the current intent is the one that best matches the user's current intent.

slots is a map of slot names, configured for the intent, to slot values that Amazon Lex has recognized in the user conversation. A slot value remains null until the user provides a value.

The slot value in the input event may not match one of the values configured for the slot. For example, if the user responds to the prompt "What color car would you like?" with "pizza," Amazon Lex will return "pizza" as the slot value. Your function should validate the values to make sure that they make sense in context.

slotDetails provides additional information about a slot value. The resolutions array contains a list of additional values recognized for the slot. Each slot can have a maximum of five values.

The originalValue field contains the value that was entered by the user for the slot. When the slot type is configured to return the top resolution value as the slot value, the originalValue may be different from the value in the slots field.

confirmationStatus provides the user response to a confirmation prompt, if there is one. For example, if Amazon Lex asks "Do you want to order a large cheese pizza?," depending on the user response, the value of this field can be Confirmed or Denied. Otherwise, this value of this field is None.

If the user confirms the intent, Amazon Lex sets this field to Confirmed. If the user denies the intent, Amazon Lex sets this value to Denied.

In the confirmation response, a user utterance might provide slot updates. For example, the user might say "yes, change size to medium." In this case, the

subsequent Lambda event has the updated slot value, PizzaSize set to medium. Amazon Lex sets the confirmationStatus to None, because the user modified some slot data, requiring the Lambda function to perform user data validation.

- alternativeIntents – If you enable confidence scores, Amazon Lex returns up to four alternative intents. Each intent includes a score that indicates the level of confidence that Amazon Lex has that the intent is the correct intent based on the user's utterance.

  The contents of the alternative intents is the same as the contents of the currentIntent field. For more information, see Using Confidence Scores.

- bot – Information about the bot that processed the request.
  - name – The name of the bot that processed the request.
  - alias – The alias of the bot version that processed the request.
  - version – The version of the bot that processed the request.
-
- userId – This value is provided by the client application. Amazon Lex passes it to the Lambda function.

- inputTranscript – The text used to process the request.
  If the input was text, the inputTranscript field contains the text that was input by the user.

  If the input was an audio stream, the inputTranscript field contains the text extracted from the audio stream. This is the text that is actually processed to recognize intents and slot values.

- invocationSource – To indicate why Amazon Lex is invoking the Lambda function, it sets this to one of the following values:

    - DialogCodeHook – Amazon Lex sets this value to direct the Lambda function to initialize the function and to validate the user's data input.

      When the intent is configured to invoke a Lambda function as an initialization and validation code hook, Amazon Lex invokes the specified Lambda function on each user input (utterance) after Amazon Lex understands the intent.

    - FulfillmentCodeHook – Amazon Lex sets this value to direct the Lambda function to fulfill an intent.

      If the intent is configured to invoke a Lambda function as a fulfillment code hook, Amazon Lex sets the invocationSource to this value only after it has all the slot data to fulfill the intent.

- In your intent configuration, you can have two separate Lambda functions to initialize and validate user data and to fulfill the intent. You can also use one Lambda function to do both. In that case, your Lambda function can use the invocationSource value to follow the correct code path.

- outputDialogMode – For each user input, the client sends the request to Amazon Lex using one of the runtime API operations, PostContent or PostText. Amazon Lex uses the request parameters to determine whether the response to the client is text or voice, and sets this field accordingly.

  The Lambda function can use this information to generate an appropriate message. For example, if the client expects a voice response, your Lambda function could return Speech Synthesis Markup Language (SSML) instead of text.

- messageVersion – The version of the message that identifies the format of the event data going into the Lambda function and the expected format of the response from a Lambda function.
- sessionAttributes – Application-specific session attributes that the client sends in the request. If you want Amazon Lex to include them in the response to the client, your Lambda function should send these back to Amazon Lex in the response. For more information, see Setting Session Attributes

- requestAttributes – Request-specific attributes that the client sends in the request. Use request attributes to pass information that doesn't need to persist for the entire session. If there are no request attributes, the value will be null. For more information, see Setting Request Attributes

- recentIntentSummaryView – Information about the state of an intent. You can see information about the last three intents used. You can use this information to set values in the intent or to return to a previous intent. For more information, see Managing Sessions With the Amazon Lex API.

- sentimentResponse – The result of an Amazon Comprehend sentiment analysis of the last utterance. You can use this information to manage the conversation flow of your bot depending on the sentiment expressed by the user. For more information, see Sentiment Analysis.

- kendraResponse – The result of a query to an Amazon Kendra index. Only present in the input to a fulfillment code hook and only when the intent extends the AMAZON.KendraSearchIntent built-in intent. The field contains the entire response from the Amazon Kendra search. For more information, see AMAZON.KendraSearchIntent.

- activeContexts – One or more contexts that are active during this turn of a conversation with the user.

- timeToLive – The length of time or number of turns in the conversation with the user that the context remains active.
- name – the name of the context.
- parameters a list of key/value pairs the contains the name and value of the slots from the intent that activated the context.

Lambda Function Response Format

```
{
  "sessionAttributes": {
     "key1": "value1",
     "key2": "value2"
     ...
  },
  "recentIntentSummaryView": [
     {
        "intentName": "Name",
        "checkpointLabel": "Label",
        "slots": {
          "slot name": "value",
          "slot name": "value"
         },
        "confirmationStatus": "None, Confirmed, or Denied (intent
confirmation, if configured)",
         "dialogActionType": "ElicitIntent, ElicitSlot,
ConfirmIntent, Delegate, or Close",
         "fulfillmentState": "Fulfilled or Failed",
         "slotToElicit": "Next slot to elicit"
     }
  ],
  "activeContexts": [
      {
        "timeToLive": {
           "timeToLiveInSeconds": seconds,
           "turnsToLive": turns
       },
```

```
      "name": "name",
      "parameters": {
        "key name": "value"
      }
    }
  ],
  "dialogAction": {
    "type": "ElicitIntent, ElicitSlot, ConfirmIntent, Delegate,
or Close",
    Full structure based on the type field. See below for
details.
  }
}
```

The response consists of four fields. The sessionAttributes, recentIntentSummaryView, and activeContexts fields are optional, the dialogAction field is required. The contents of the dialogAction field depends on the value of the type field. For details, see dialogAction.

sessionAttributes

Optional. If you include the sessionAttributes field it can be empty. If your Lambda function doesn't return session attributes, the last known sessionAttributes passed via the API or Lambda function remain. For more information, see the PostContent and PostText operations.

```
 "sessionAttributes": {
     "key1": "value1",
     "key2": "value2"

  }
```

recentIntentSummaryView

Optional. If included, sets values for one or more recent intents. You can include information for up to three intents. For example, you can set values for previous intents based on information gathered by the current intent. The information in the summary must be valid for the intent. For example, the intent name must be an intent in the bot. If you include a slot value in the summary view, the slot must exist in the intent. If you don't include the recentIntentSummaryView in your response, all of the values for the recent intents remain unchanged. For more information, see the PutSession operation or the IntentSummary data type.

```
"recentIntentSummaryView": [
    {
        "intentName": "Name",
        "checkpointLabel": "Label",
        "slots": {
          "slot name": "value",
          "slot name": "value"
         },
        "confirmationStatus": "None, Confirmed, or Denied (intent
confirmation, if configured)",
         "dialogActionType": "ElicitIntent, ElicitSlot,
ConfirmIntent, Delegate, or Close",
         "fulfillmentState": "Fulfilled or Failed",
         "slotToElicit": "Next slot to elicit"
    }
  ]
```

activeContexts

Optional. If included, sets the value for one or more contexts. For example, you can include a context to make one or more intents that have that context as an input eligible for recognition in the next turn of the conversation.

Any active contexts that are not included in the response have their time-to-live values decremented and may still be active on the next request.

If you specify a time-to-live of 0 for a context that was included in the input event, it will be inactive on the next request.

For more information, see Setting Intent Context.

dialogAction

Required. The dialogAction field directs Amazon Lex to the next course of action, and describes what to expect from the user after Amazon Lex returns a response to the client.

The type field indicates the next course of action. It also determines the other fields that the Lambda function needs to provide as part of the dialogAction value.

- Close — Informs Amazon Lex not to expect a response from the user. For example, "Your pizza order has been placed" does not require a response.

  The fulfillmentState field is required. Amazon Lex uses this value to set the dialogState field in the PostContent or PostText response to the client application. The message and responseCard fields are optional. If you don't specify a message, Amazon Lex uses the goodbye message or the follow-up message configured for the intent.

```
"dialogAction": {
    "type": "Close",
    "fulfillmentState": "Fulfilled or Failed",
    "message": {
      "contentType": "PlainText or SSML or CustomPayload",
      "content": "Message to convey to the user. For example,
Thanks, your pizza has been ordered."
    },
   "responseCard": {
      "version": integer-value,
      "contentType": "application/vnd.amazonaws.card.generic",
      "genericAttachments": [
          {
```

```
                "title":"card-title",
                "subTitle":"card-sub-title",
                "imageUrl":"URL of the image to be shown",
                "attachmentLinkUrl":"URL of the attachment to be
associated with the card",
                "buttons":[
                    {
                        "text":"button-text",
                        "value":"Value sent to server on button
click"
                    }
                ]
            }
        ]
    }
```

- ```
  }
  ```

- ConfirmIntent — Informs Amazon Lex that the user is expected to give a yes or no answer to confirm or deny the current intent.

  You must include the intentName and slots fields. The slots field must contain an entry for each of the filled slots for the specified intent. You don't need to include a entry in the slots field for slots that aren't filled. You must include the message field if the intent's confirmationPrompt field is null. The contents of the message field returned by the Lambda function take precedence over the confirmationPrompt specified in the intent. The responseCard field is optional.

```
"dialogAction": {
    "type": "ConfirmIntent",
    "message": {
      "contentType": "PlainText or SSML or CustomPayload",
      "content": "Message to convey to the user. For example,
Are you sure you want a large pizza?"
    },
   "intentName": "intent-name",
   "slots": {
      "slot-name": "value",
```
```

```
        "slot-name": "value",
        "slot-name": "value"
    },
    "responseCard": {
        "version": integer-value,
        "contentType": "application/vnd.amazonaws.card.generic",
        "genericAttachments": [
            {
                "title":"card-title",
                "subTitle":"card-sub-title",
                "imageUrl":"URL of the image to be shown",
                "attachmentLinkUrl":"URL of the attachment to be
associated with the card",
                "buttons":[
                    {
                        "text":"button-text",
                        "value":"Value sent to server on button
click"
                    }
                ]
            }
        ]
    }
}
```

-

- Delegate — Directs Amazon Lex to choose the next course of action based on the bot configuration. If the response does not include any session attributes Amazon Lex retains the existing attributes. If you want a slot value to be null, you don't need to include the slot field in the request. You will get a DependencyFailedException exception if your fulfillment function returns the Delegate dialog action without removing any slots.
  The kendraQueryRequestPayload and kendraQueryFilterString fields are optional and only used when the intent is derived from the AMAZON.KendraSearchIntent built-in intent. for more information, see AMAZON.KendraSearchIntent.

```
 "dialogAction": {
    "type": "Delegate",
    "slots": {
       "slot-name": "value",
       "slot-name": "value",
       "slot-name": "value"
    },
    "kendraQueryRequestPayload": "Amazon Kendra query",
    "kendraQueryFilterString": "Amazon Kendra attribute filters"
```

```
    }
```

- ElicitIntent — Informs Amazon Lex that the user is expected to respond with an utterance that includes an intent. For example, "I want a large pizza," which indicates the OrderPizzaIntent. The utterance "large," on the other hand, is not sufficient for Amazon Lex to infer the user's intent.

  The message and responseCard fields are optional. If you don't provide a message, Amazon Lex uses one of the bot's clarification prompts. If there is no clarification prompt defined, Amazon Lex returns a 400 Bad Request exception.

```
{
  "dialogAction": {
    "type": "ElicitIntent",
    "message": {
      "contentType": "PlainText or SSML or CustomPayload",
      "content": "Message to convey to the user. For example,
What can I help you with?"
    },
    "responseCard": {
      "version": integer-value,
      "contentType": "application/vnd.amazonaws.card.generic",
      "genericAttachments": [
          {
             "title":"card-title",
             "subTitle":"card-sub-title",
             "imageUrl":"URL of the image to be shown",
             "attachmentLinkUrl":"URL of the attachment to be
associated with the card",
```

```
                "buttons":[
                    {
                        "text":"button-text",
                        "value":"Value sent to server on button
click"
                    }
                ]
            }
        ]
    }
}
```

-
- ElicitSlot — Informs Amazon Lex that the user is expected to provide a slot value in the response.

  The intentName, slotToElicit, and slots fields are required. The message and responseCard fields are optional. If you don't specify a message, Amazon Lex uses one of the slot elicitation prompts configured for the slot.

```
"dialogAction": {
    "type": "ElicitSlot",
    "message": {
      "contentType": "PlainText or SSML or CustomPayload",
      "content": "Message to convey to the user. For example,
What size pizza would you like?"
    },
   "intentName": "intent-name",
   "slots": {
      "slot-name": "value",
      "slot-name": "value",
      "slot-name": "value"
   },
   "slotToElicit" : "slot-name",
   "responseCard": {
      "version": integer-value,
      "contentType": "application/vnd.amazonaws.card.generic",
      "genericAttachments": [
```

```
            {
                "title":"card-title",
                "subTitle":"card-sub-title",
                "imageUrl":"URL of the image to be shown",
                "attachmentLinkUrl":"URL of the attachment to be
associated with the card",
                "buttons":[
                    {
                        "text":"button-text",
                        "value":"Value sent to server on button
click"
                    }
                ]
            }
        ]
    }
```

- }