



Introduction to Build AI Chatbots using AWS LEX

[Mariusz Momotko](#)

Senior SDE
Amazon Alexa NLU



Agenda

- Part 1 – Build a chat bot
- Part 2 – Build a useful chat bot
- Part 3 – Build even more useful bot

Workshop materials

- All materials are available on [aws-lex-chatbot GitHub project](#)
 - Lambda functions (+ tests)
 - Bot (definition + deployment)
 - This presentation (PDF)

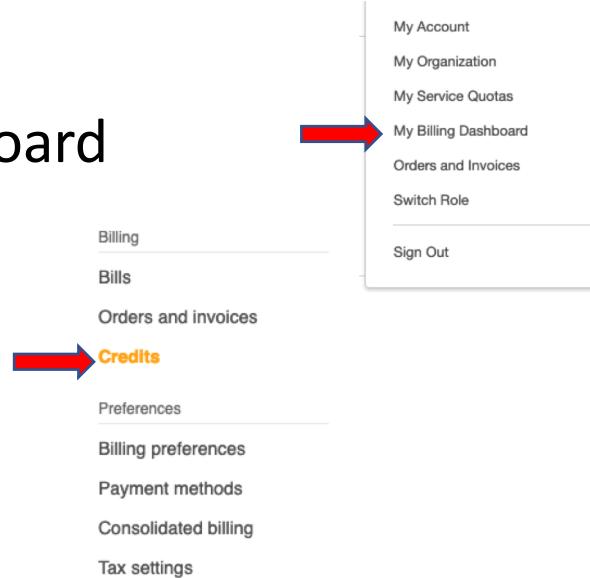
mariusz-momotko	Delete a.txt	Latest commit a115b46 11 minutes ago
docs	Create a.txt	20 minutes ago
lambda	Delete a.txt	11 minutes ago
lex/chatbots	Delete a.txt	21 minutes ago
README.md	Initial commit	1 hour ago



On slides – this symbol indicates that you can use materials if you do exercises on your own.

How to use promo codes

- Log in to AWS Console
- Click My Billing Dashboard
- Choose Billing/Credits
- Type the promo code
- Click Redeem





Part 1 – Build a AWS Lex chat bot

Specify requirements for a bot to support buying a train ticket for SKM

Create a bot

Define intent and slots



A need – I want to buy a train ticket for SKM

Gdańsk Główny	Gdańsk Wrzeszcz	26-06-2019 18:00	Search
---------------	-----------------	------------------	--------

+ Add via station

Connections Gdańsk Główny – Gdańsk Wrzeszcz

Departure	Arrival	Connection	Travel time	Price	
◀ Earlier					
<u>26-06-2019</u>					
18:03	18:09		0:06	3,20 zł	Buy ticket
18:13	18:19		0:06	3,20 zł	Buy ticket
18:23	18:29		0:06	3,20 zł	Buy ticket

https://bilety.skm.pkpl/en/search/gdansk-glowny/gdansk-wrzeszcz/26-06-2019_18:00



AWS Lex

AWS Management Console

AWS services

Find Services
You can enter names, keywords or acronyms.

 X

Amazon Lex
Build Voice and Text Chatbots

Alexa for Business | Amazon Lex
Alexa for Business Provides Tools to Manage Alexa in Your Organization

MediaConnect
Reliable, secure, and flexible transport for live video

AWS Services Resource Groups ★

ConduitAccessClientRole-DO-N... Ireland Support

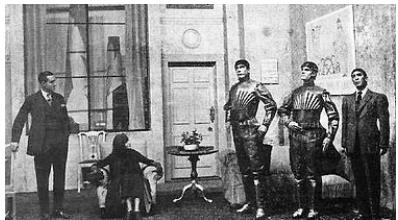
Amazon Lex

Bots

Create Actions ▾

Filter: Filter by Bot name

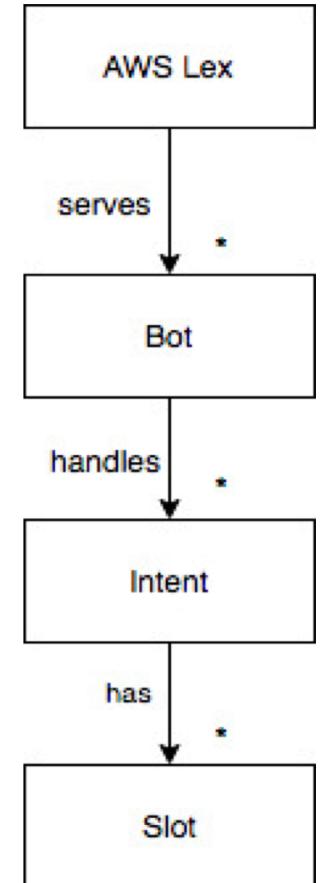
Name	Status	Last updated	Date Created
OrderFlowers	READY	May 10, 2018 at 3:26:54 PM UTC+2	May 10, 2018 at 9:15:11 AM UTC+2
OrderFlowersMOBILEHUB	READY	May 11, 2018 at 10:31:19 AM UTC+2	May 11, 2018 at 10:30:37 AM UTC+2



AWS Lex Bot – Bots, Intents & Slots



- Bot - a ro(bot) able to carry out conversation with a user (in a textual or voice enabled mode)
- Intent – a particular goal that the user wants to achieve
- Slot – data / parameters that the user need to fulfill in order to understand the goal
- Examples
 - *Play Hello by Adele*
 - Intent - PlayMusicIntent
 - Slots: [ArtistName="Adele"], [SongName="Hello"]
 - *Check the weather in Gdańsk*
 - Intent: ForecastedWeatherIntent
 - Slots: [place="Gdańsk"]





Support in ordering a SKM ticket – Lex Bot

AWS Services Resource Groups

ConduitAccessClientRole-DO-N... Ireland Support

Amazon Lex

Bots Intents Slot types

Create Actions

Filter: Q Filter by Bot name

Name	Status	Last updated	Date Created
OrderFlowers	READY	May 10, 2018 at 3:26:54 PM UTC+2	May 10, 2018 at 9:15:11 AM UTC+2
OrderFlowersMOBILEHUB	READY	May 11, 2018 at 10:31:19 AM UTC+2	May 11, 2018 at 10:30:37 AM UTC+2

Create your bot

Amazon Lex enables any developer to build conversational chatbots and dynamically adjusts the response. To get started, you can choose to create a custom bot or one of the many pre-built templates.

CREATE YOUR OWN

Custom bot

Bot name OrderTicket

Language English (US)

Output voice Salli

Type text here to hear a sample

Session timeout 5 min

IAM role AWSServiceRoleForLexBots
Automatically created on your behalf

COPPA Please indicate if your use of this bot is subject to the Children's Online Privacy Protection Act (COPPA). [Learn more](#)

Yes No

Amazon Lex

Bots Intents Slot types

Create Actions

Filter: Q Filter by Bot name

Name	Status
OrderFlowers	READY
OrderFlowersMOBILEHUB	READY
OrderTicket	NOT_BUILT



Ordering a ticket – Intent (1)

The screenshot shows the SKM OrderTicket bot editor interface. At the top, there are search fields for 'Gdańsk Główny' (origin), 'Gdańsk Wrzeszcz' (destination), and a date/timestamp '26-06-2019 18:00'. A red 'Search' button is to the right. Below this is a navigation bar with tabs: Editor (selected), Settings, Channels, and Monitoring. The 'Editor' tab shows sections for 'Intents' (No intents created) and 'Slot types' (No slots created). On the right, there's a 'Getting started with your bot' guide and a blue '+ Create Intent' button. A modal window titled 'Create intent' is open at the bottom left, prompting for a unique name. The input field contains 'OrderTicketIntent'. At the bottom of the modal are 'Previous' and 'Add' buttons.

- Intent – OrderTicket
- Built-in Intents
 - HelpIntent
 - CancelIntent
 - ...



Ordering a ticket – Intent – Sample utterances

- Utterance – spoken or typed phrases that invoke the customer's intent

▼ Sample utterances ⓘ

e.g. I would like to book a flight. +

want a ticket ×

buy ticket ×





Ordering a ticket – Intent – Slots

SKM Search connections | Special Pass 2018 A+

Gdańsk Główny Gdańsk Wrzeszcz 26-06-2019 18:00 Search

Slots 1

Priority	Required	Name	Slot type	Version	Prompt	Settings
		e.g. Location	e.g. AMAZON.US_CITY	Built-in	e.g. What city?	+
1.	✓	fromStation	AMAZON.EUROPE_CITY	Built-in	What is the station you are going from?	⚙️ ✖️
2.	✓	toStation	AMAZON.EUROPE_CITY	Built-in	What is the station you are going to?	⚙️ ✖️
3.	✓	dateOfTravel	AMAZON.DATE	Built-in	What is date of your travel?	⚙️ ✖️
4.	✓	timeOfTravel	AMAZON.TIME	Built-in	What is time of your travel?	⚙️ ✖️

- Slot type
 - Built-in slots [Alexa Skills Kit](#)
 - Date, Time, Numbers, Cities
 - ...

⚠️ Remember to save changes made for the intent.

Slots

- fromStation
- toStation
- dateOfTravel
- timeOfTravel



Ordering a ticket – a first try

- First try (after Build)
 - Interaction
 - Summary (ReadyForFulfillment)
 - Inspect response
- Bot which was built can be published
- Bot is versioned

Build Publish ?

> Test bot (Latest) Ready. Build complete.

want ticket

What is the station you are going from?

Gdańsk Wrzeszcz

What is the station you are going to?

Tczew

What is date of your travel?

today

What is time of your travel?

now

Intent OrderTicketIntent is ReadyForFulfillment:
dateOfTravel:2019-06-24
fromStation:Gdańsk Wrzeszcz
timeOfTravel:09:13 toStation:Tczew

Inspect response

Dialog State: ReadyForFulfillment

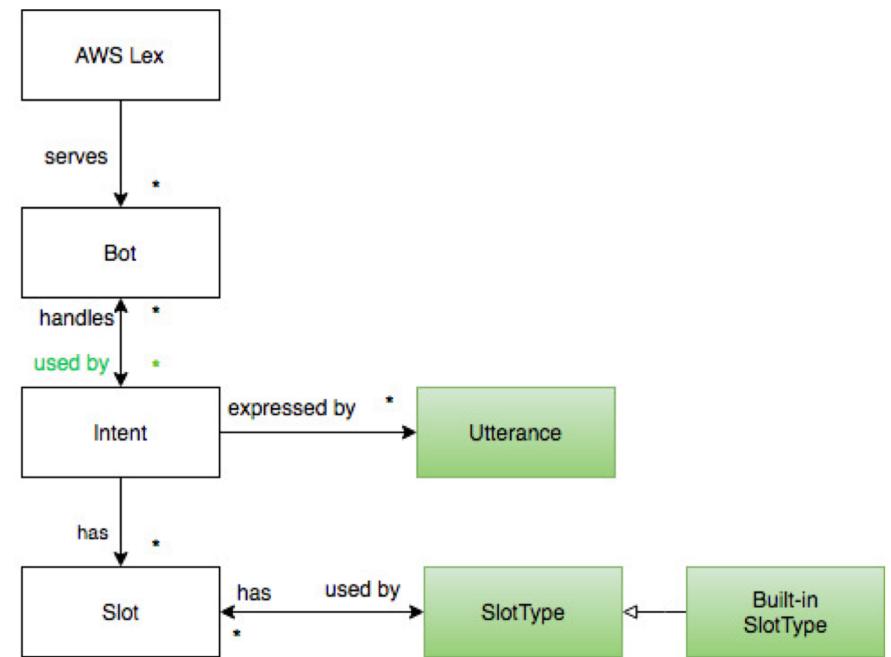
Summary Detail

Intent: OrderTicketIntent

Slots	(4/4)
dateOfTravel	2019-06-24
fromStation	Gdańsk Wrzeszcz
timeOfTravel	09:13
toStation	Tczew

Part 1 - Summary

- We have a working chat bot
 - Recognize customer utterances
 - Ask for customer choices (slots)
 - Provide final response
- BUT
 - Customer can provide incorrect train stations, we cannot validate them
 - It is not possible to cancel customer order
 - Final information is not readable enough





Part 2 – Build a useful AWS Lex chat bot

Adding confirmation prompts

Validation via customized slot type

Formatting the bot's response

Error Handling

Adding confirmation prompts

- Confirm – a question to confirm your intent and choices (slots). It appears at the end of the conversation.
- Cancel – a message which will appear when you cancel your intent.

▼ Confirmation prompt ⓘ

Confirmation prompt

Confirm

Are you sure to order this ticket? 

Cancel (if the user says "no")

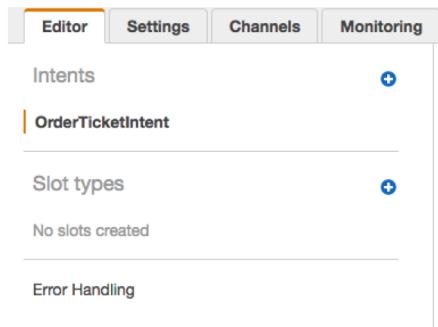
Okay, your order will not be placed. 



⚠ Before testing - remember to save the intent definition and build it.

Define new slot type for SKM stations

- Customized slot type for SKM stations
 - Via Lex GUI (synonyms only in English)
 - Via Export/Import feature



Edit slot type

SKMStationType Latest ▾

List of all SKM stations.

Slot Resolution

Expand Values ⓘ

Restrict to Slot values and Synonyms ⓘ

Value ⓘ

e.g. Small Enter Synonym +

Press Tab to add a synonym

sopot-kamienny-potok	Sopot Kamienny Potok,	x
gdynia-chylonia	Gdynia Chylonia, Chylonia	x
gdynia-orlowo	Gdynia Orlowo, Orlowo	x
sopot-wyscigi	Sopot Wyścigi, Wyścigi	x
gdansk-politechnika	Gdańsk Politechnika, Gdańsk	x

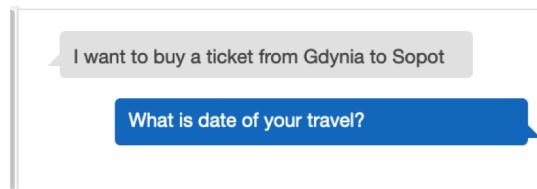
```
{
    "metadata": {
        "schemaVersion": "1.0",
        "importType": "LEX",
        "importFormat": "JSON"
    },
    "resource": {
        "description": "List of all SKM stations.",
        "name": "SKMStationType",
        "version": "1",
        "enumerationValues": [
            {
                "value": "gdansk-srodmiescie",
                "synonyms": [
                    "Gdańsk Śródmieście",
                    "Gdansk Śródmieście",
                    "Śródmieście"
                ]
            },
            {
                "value": "gdansk-glowny",
                "synonyms": [
                    "Gdańsk",
                    "Gdansk"
                ]
            },
            {
                "value": "gdansk-politechnika",
                "synonyms": [
                    "Gdańsk Politechnika",
                    "Gdansk Politechnika",
                    "Politechnika"
                ]
            }
        ]
    }
}
```

⚠ Remember to select proper version of the slot type in slots definition.

-cat To import already defined features for this slot type – use [this file](#).

Simplify slot fulfillment

- Adding slots directly to sample utterances



▼ Sample utterances ⓘ

e.g. I would like to book a flight.

I want buy a ticket from {fromStation} to {toStation}

want a ticket

buy ticket

 Names used in the message must match the names defined for the slots

Formatting the bot's response

- Response
 - final element of a bot's intent, and are displayed to users after the fulfillment of the intent is complete
- Textual response
 - Slot values: {slotName}, e.g. {fromStation}
 - Session attributes value [attribute] (will discuss it a bit later)



The screenshot shows the 'Response' configuration section in the Amazon Lex console. It displays two message preview cards:

- Message:** A card with a blue header containing 'Message' and 'Custom Markup' radio buttons, and a trash icon. It contains the text: "One of these messages will be presented at random." Below this are two message slots:
 - e.g. *Thank you. Your {Drink_Name} has been ordered.*
 - Thank you. If you want to order a SKM ticket use this link: https://bilety.skm.pkpl/en/search/{fromStation}/{toStation}/{dateOfTravel}_{timeOfTravel}
- Custom Markup:** A card with a blue header containing 'Preview' and a trash icon. It contains the same two message slots as the Message card.

At the bottom left, there is a blue 'Add Message' button. At the top right, there are 'Refresh response' and 'Exit preview' buttons.

Error handling

Error handling

Clarification prompts

e.g. Sorry, can you please repeat that?



Sorry, can you please repeat that?



Maximum number of retries

5

Hang-up phrase

e.g. Sorry, I could not understand. Please contact customer support.



Sorry, I could not understand. Goodbye.





Publish the bot

Publish OrderTicket

Your bot is published! You can now connect to your mobile app or continue to chatbot deployment.

Bot Name	OrderTicket
Bot Version	1
Alias	OrderTicket

What to do next?

Here are some resources to help you progress once your bot is published.

[How to connect to your mobile app](#)
Learn how to connect to your bot to your mobile app.
[Download connection info](#)

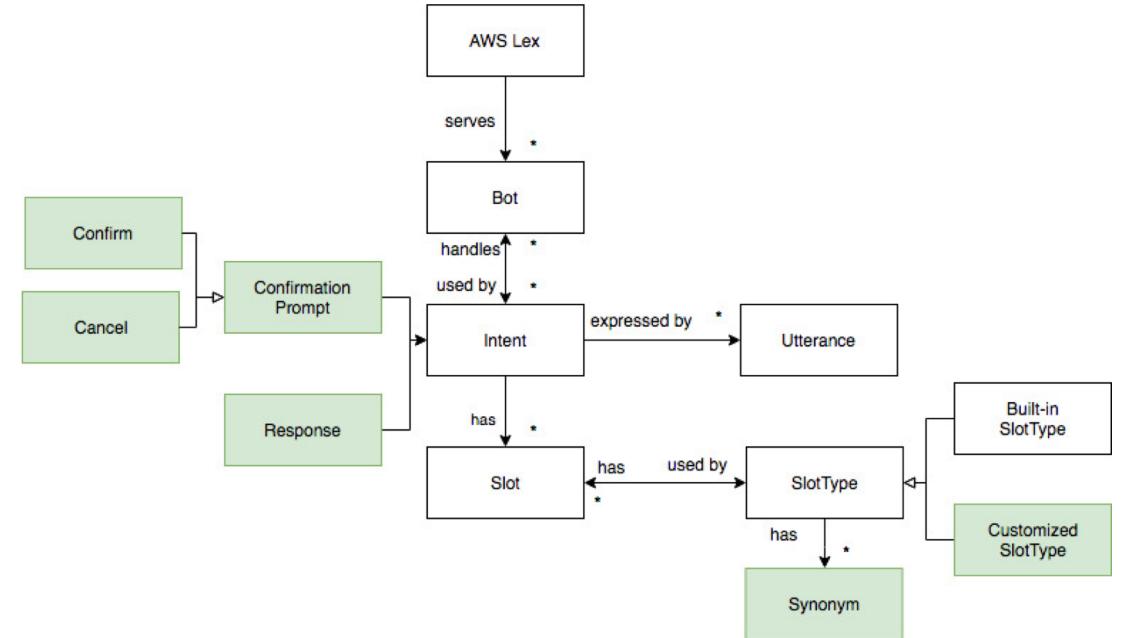
[Integrate with Mobile hub.](#)
Please create a project and choose the Conversational Bots feature in Mobile Hub

[How to deploy your bot to other services](#)
Learn how to deploy your bot to other services like Facebook Messenger, Slack, Twilio, and Kik.
[Go to channels](#)

[Close](#)

Part 2 - Summary

- We are able to:
 - provide proper from / to SKM stations
 - ask for the final confirmation of the customer intent
 - Provide final message (SKM url)
 - handle errors
- BUT
 - Customer may provide the same from and to stations
 - The final url is not correct (date format)





Part 3 – Building even more useful bot

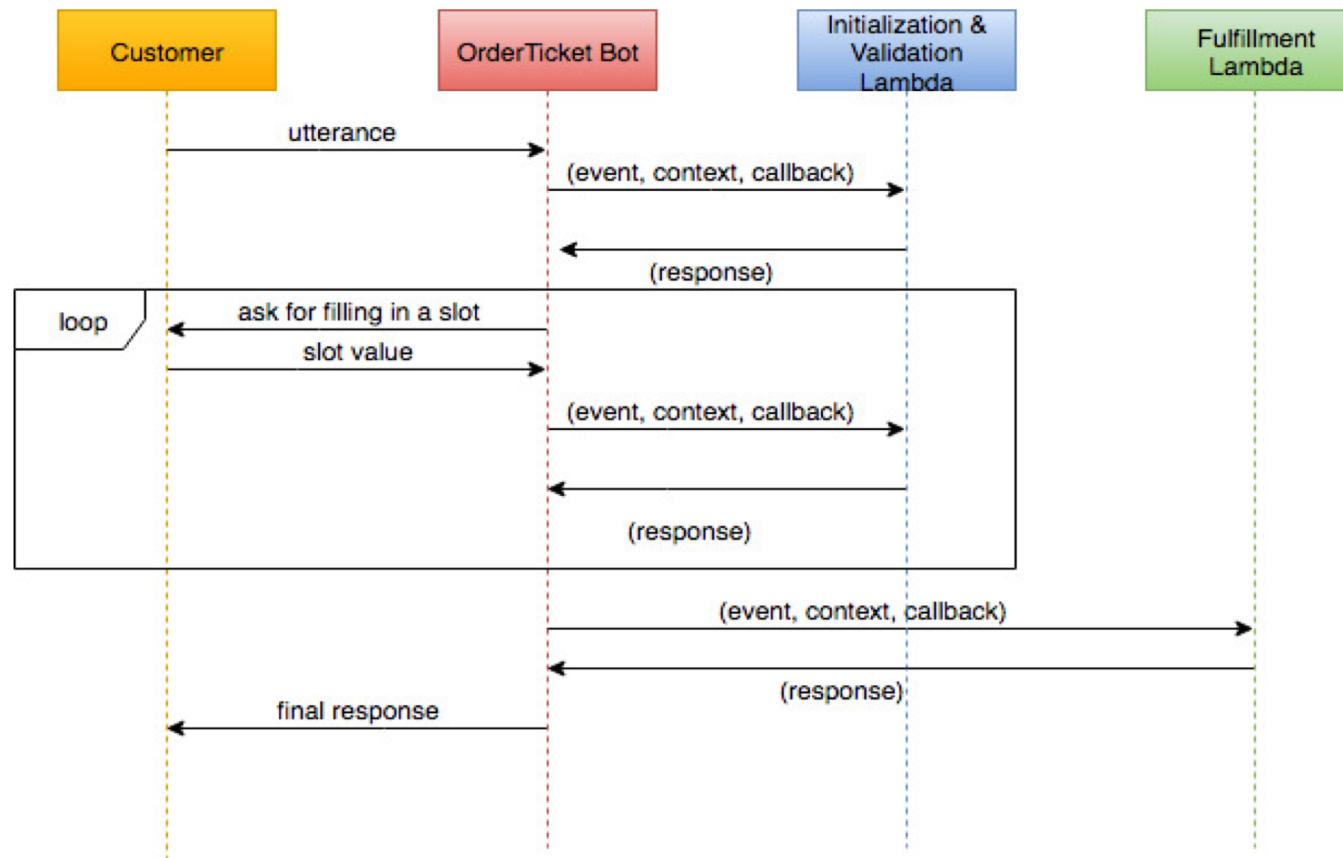
Advanced validation (lambda)

Advanced response formatting (lambda)

Deploy a demo application



Lex Bot Interaction Model



Lex Data Structures - Input

- CurrentIntent
 - Slots – if not filled in → empty
- Bot metadata
- Request metadata
- Session attributes
- Request attributes

```
{
  "currentIntent": {
    "name": "intent-name",
    "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"
  }
}
```

Lex Data Structures - Output

- (selected) Dialog actions
 - Close – not expecting a response from the user
 - ConfirmIntent
 - Delegate – continue conversation with the user
 - ElicitSlot – the user is expected to provide a value for a given slot

```
{  
    "sessionAttributes": {  
        "key1": "value1",  
        "key2": "value2"  
        ...  
    },  
    "dialogAction": {  
        "type": "ElicitIntent, ElicitSlot, ConfirmIntent, Delegate, or Close",  
        Full structure based on the type field. See below for details.  
    }  
}
```



Lambda function implemented in Node.js

- Handler – the method in Lambda function that processes events
 - Types
 - Async handler
 - Non-async handler (recommended by Lex)
 - Inputs
 - Event – chatbot data compliant with the Lex input structure
 - Callback – a handler to call Lex
 - Output
 - Chatbot command and other chatbot data compliant with the Lex output structure



Create a Lambda function

The screenshot shows two main parts of the AWS Lambda console.

Left Panel (AWS Lambda Console):

- Header: AWS Services ▾ Resource Groups ▾
- Left sidebar: History, Lambda, Console Home, Amazon Lex, Amazon SageMaker, S3, EC2.
- Main content area:
 - Lambda** (selected): Run Code without Thinking about Servers.
 - Amazon Lex: Build Voice and Text Chatbots.
 - CodeBuild: Build and Test Code.
 - IoT 1-Click: Trigger AWS Lambda functions from simple devices.

Right Panel (Create function wizard):

- Header: Lambda > Functions > Create function
- Create function** Info: Choose one of the following options to create your function.
- Author from scratch** (selected): Start with a simple Hello World example.
- Basic information**
 - Function name: OrderTicketValidation
 - Runtime Info: Choose the language to use to write your function. Node.js 10.x
- Permissions Info**: Lambda will create an execution role with permission to upload logs to Amazon CloudWatch Logs. You can choose or create an execution role.



Initialization and validation using Lambda OrderTicketValidation

- Step 1 - Import lambda function – OrderTicketValidation.zip
- Step 2 – analyze validation rule for checking from/to stations
 - Get the request
 - Get *fromStation* slot value (empty)
 - Get *toStation* slot value (empty)
 - Compare from/to stations
 - Return proper action
 - Delegate – if everything is OK
 - ElicitSlot – if *toStation* slot needs to be corrected
- Step 3 – test the function



To import lambda code of this function – use [this file](#).
Tests can be imported separately – they are available [here](#).

▼ Lambda initialization and validation ⓘ

Initialization and validation code hook

Lambda function

[View in Lambda console](#)

Version or alias



Test the function

- Provide input with slots
- Verify the result

Version: \$LATEST ▾ Actions ▾ OrderTicketValPositive ▾ Test Save

OrderTicketValidation:\$LATEST

Execution result: succeeded (logs)

Details

The area below shows the result returned by your function

```
{  
  "sessionAttributes": {},  
  "dialogAction": {  
    "type": "Delegate",  
    "slots": {  
      "fromStation": "gdansk-glowny",  
      "toStation": "gdynia"  
    }  
  }  
}
```

Configure test event X

A function can have up to 10 test events. The events are persisted so you can switch to another computer or web browser and test your function with the same events.

Create new test event
 Edit saved test events

Saved test event G

OrderTicketValPositive

```
1 {  
2   "messageVersion": "1.0",  
3   "invocationSource": "DialogCodeHook",  
4   "userId": "John",  
5   "sessionAttributes": {},  
6   "bot": {  
7     "name": "OrderTicket",  
8     "alias": "$LATEST",  
9     "version": "$LATEST"  
10 },  
11  "outputDialogMode": "Text",  
12  "currentIntent": {  
13    "name": "OrderTicketIntent",  
14    "slots": {  
15      "fromStation": "gdansk-glowny",  
16      "toStation" : "gdynia"  
17    },  
18    "confirmationStatus": "None"  
19  }  
20 }
```

Delete Cancel Save



Formatting the result using Lambda OrderTicketFulfillment

To order SKM ticket we need to generate proper url, an example:

<https://bilety.skm.pkpl/en/search/gdansk-glowny/gdansk-wrzeszcz/26-06-2019 18:00>

The expected pattern:

`https://bilety.skm.pkpl/en/search/<fromStation>/<toStation>/<DD>-<MM>-<YYYY>_<HH:MI>`

Fulfillment

AWS Lambda function Return parameters to client

Lambda function: OrderTicketFulfillment

View in Lambda console

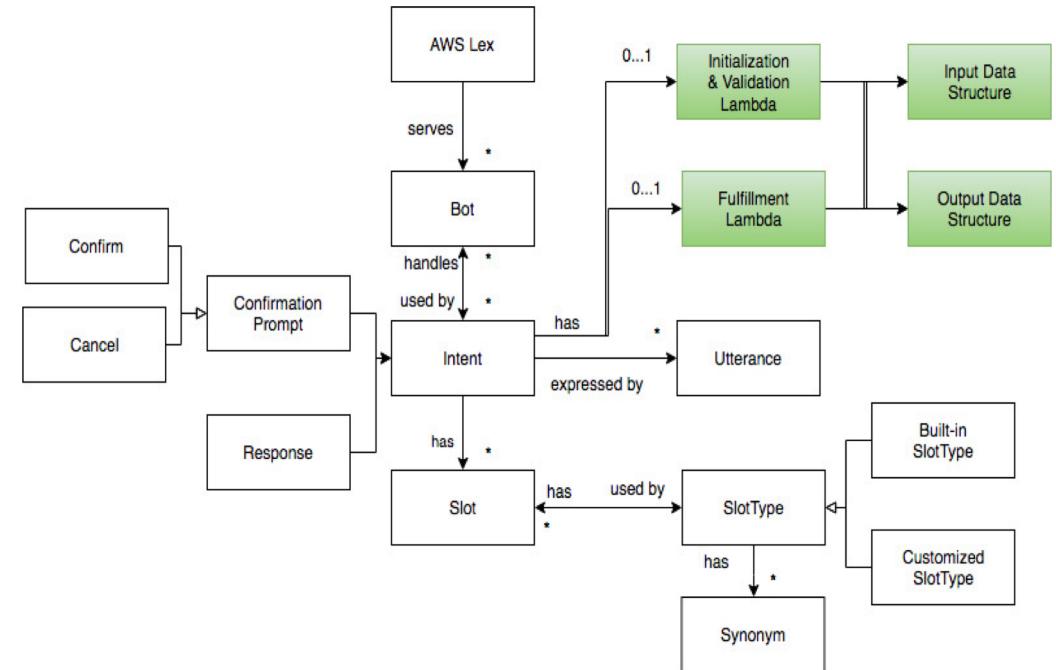
Version or alias: Latest



To import lambda code of this function – use [this file](#).
Tests can be imported separately – they are available [here](#).

Part 3 - Summary

- We are able to
 - Provide data and validate them
 - Get proper url that routes us to SKM web page
- BUT
 - This work for a single ticket only
 - Still there are many parts of ticket ordering which have to be done manually





CloudFormation - Deploy a WebUI application

Create stack

Prerequisite - Prepare template

Prepare template
Every stack is based on a template. A template is a JSON or YAML file that contains configuration information about the AWS resources you want to include in the stack.

Template is ready Use a sample template Create template in Designer

Specify template
A template is a JSON or YAML file that describes your stack's resources and properties.

Template source
Selecting a template generates an Amazon S3 URL where it will be stored.

Amazon S3 URL Upload a template file

Amazon S3 URL
<https://s3.amazonaws.com/aws-bigdata-blog/artifacts/aws-lex-web-ui/artifacts/templates/master.yaml>

Amazon S3 template URL

S3 URL: <https://s3.amazonaws.com/aws-bigdata-blog/artifacts/aws-lex-web-ui/artifacts/templates/master.yaml> [View in Designer](#)



A preconfigured template is available [here](#).

Lex Bot Configuration Parameters

BotName

Name of an existing Lex Bot to be used by the web application.
[OrderTicket](#)

WebAppConfBotInitialText

First bot message displayed in the chatbot UI
[You can ask me for help ordering a SKM ticket. Just type "Buy tickets" or click on the mic and say it.](#)

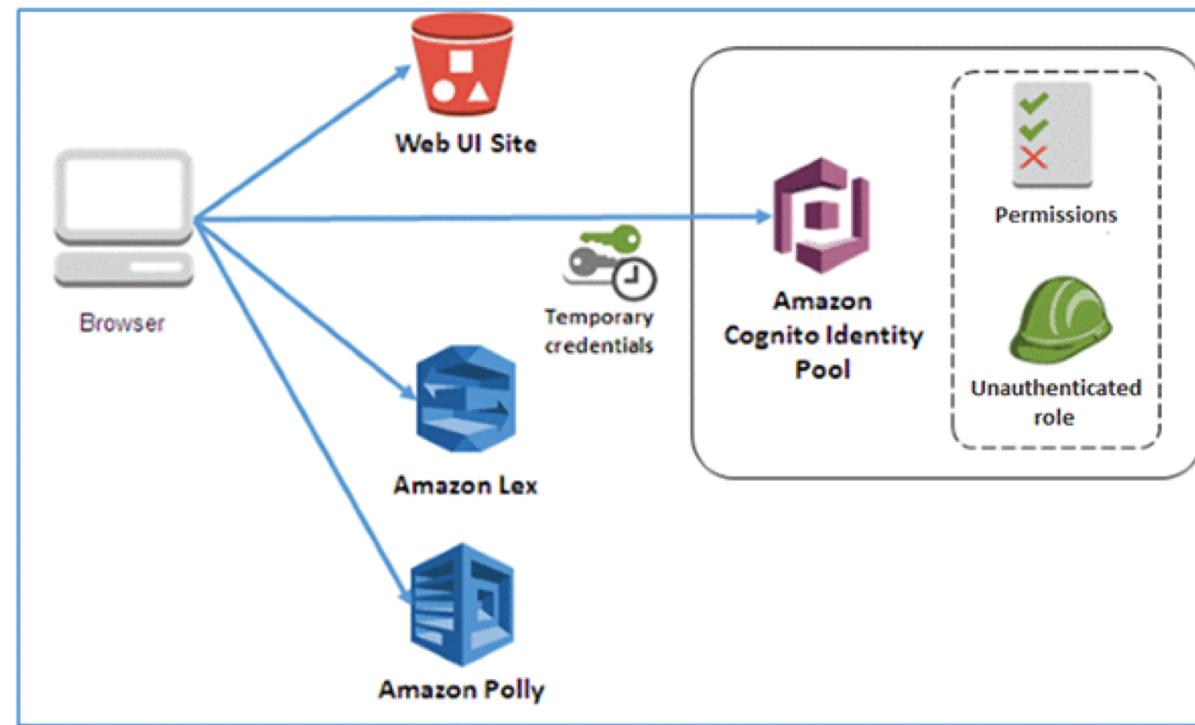
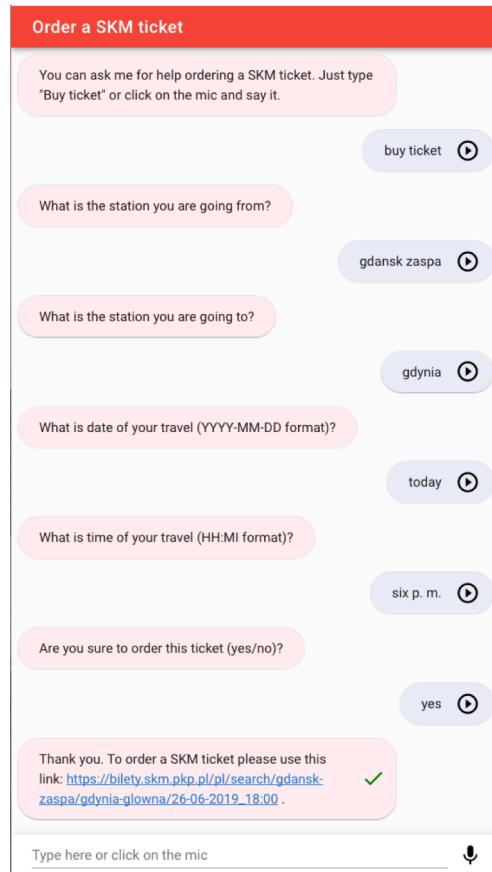
WebAppConfBotInitialSpeech

Message spoken by bot when the microphone is first pressed in a conversation
[Say 'Buy tickets' to get started.](#)

WebAppConfToolbarTitle

Title displayed in the chatbot UI toolbar
[Order a SKM ticket](#)

Demo - voice and textual interfaces





Part 3 - Summary

- We are able to:
 - Understand the intent and all required slots
 - Validate the provided slots
 - Provide final message (SKM url)
 - Handle errors
- BUT
 - We do not cope with other types of train tickets (e.g. seasons tickets)
 - We do not have a mobile interface
 - ...

BUT for today – the plan we had for this workshop has been implemented ☺



Thank you for your attention