Microsoft

Develop natural language processing solutions

Agenda

- Build a question answering solution
- Build a conversational language understanding app
- Custom classification and named entity extraction
- Speech recognition, synthesis, and translation

Build a question answering solution

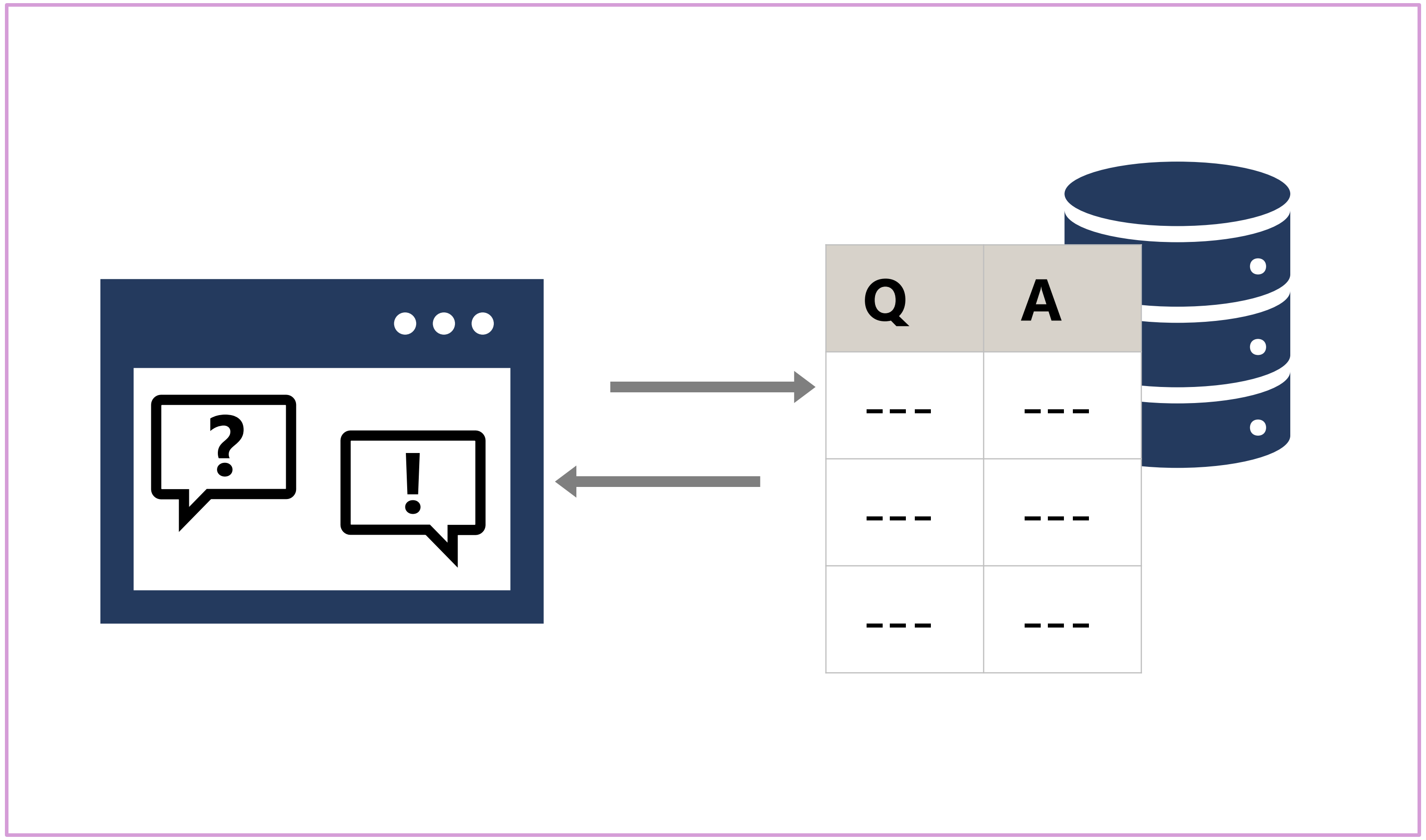
Learning Objectives

After completing this module, you will be able to:

1. Describe the question answering capabilities of Azure AI Language.
2. Describe the differences between question answering and conversational language understanding.
3. Create a knowledge base.
4. Implement multi-turn conversation.
5. Test and publish a knowledge base.
6. Consume a published knowledge base.
7. Implement active learning.

Introduction to Question Answering

- Knowledge base of question and answer pairs with natural language understanding
- Published as a REST endpoint for applications to consume
- Available through language specific SDKs



Question Answering vs Language Understanding

Question answering

- User submits a question, expecting an answer
- Service uses natural language understanding to match the question to an answer in the knowledge base
- Response is a static answer to a known question
- Client application presents the answer to the user

Language understanding

- User submits an utterance, expecting an appropriate response or action
- Service uses natural language understanding to interpret the utterance, match it to an intent, and identify entities
- Response indicates the most likely intent and referenced entities
- Client application is responsible for performing appropriate action based on the detected intent

Creating a Knowledge Base

Use the Language Studio portal

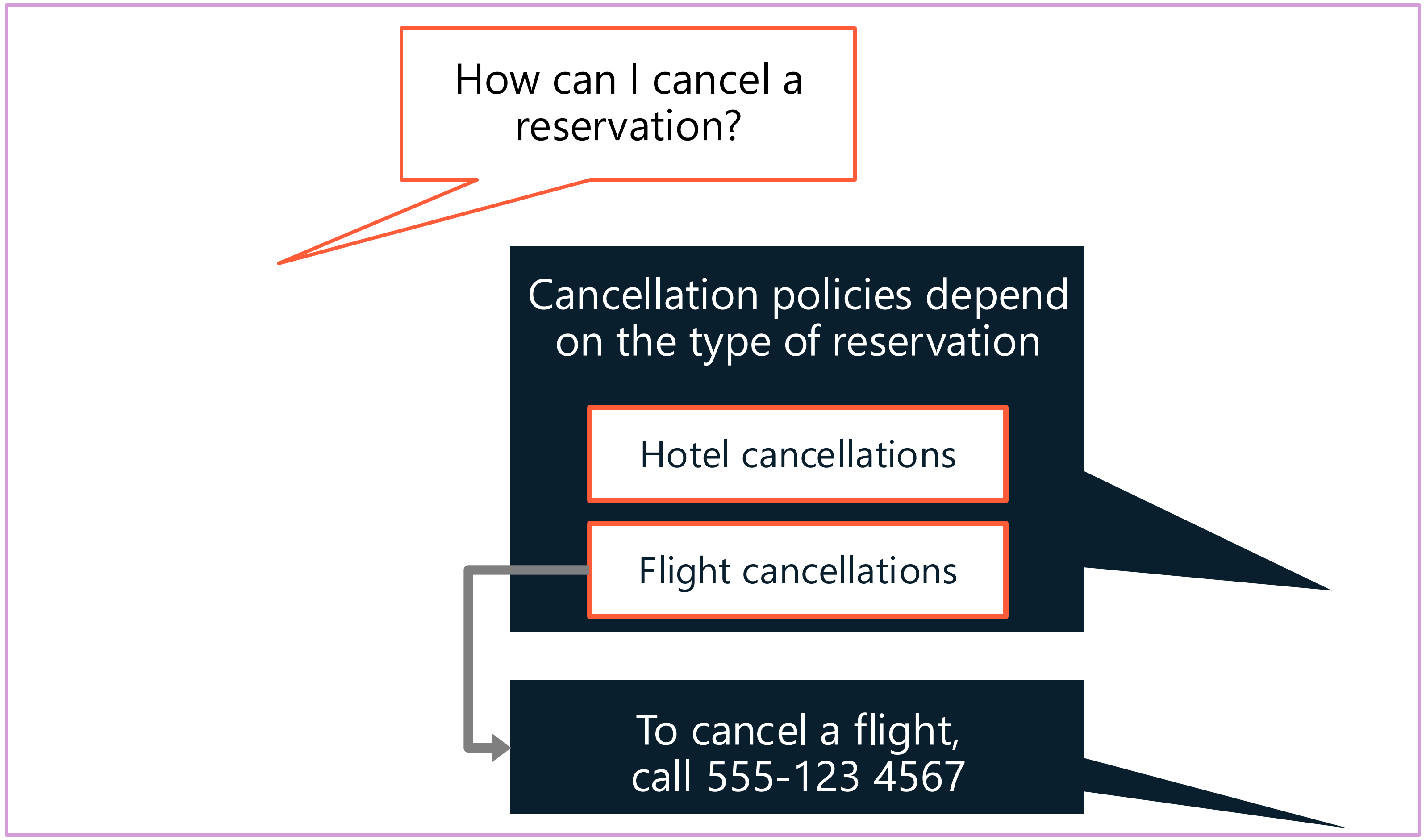
1. Create an **Azure AI Language service** resource in your Azure subscription
2. In Language Studio, select your Azure AI Language resource and **create a Custom question answering** project.
3. Populate the knowledge base:
    - Import from existing FAQ web page
    - Upload document files
    - Add pre-defined “chit-chat” pairs

5. Create the knowledge base and edit question and answer pairs

Multi-turn conversation

Add follow-up prompts to define multi-turn exchanges

- Can reference existing question and answer pairs
- Can be restricted to follow-up responses only


How can I cancel a reservation?
Cancellation policies depend on the type of reservation
Hotel cancellations
Flight cancellations
To cancel a flight, call 555-123 4567

Testing and publishing a Knowledge Base

Test interactively in Language Studio

- Inspect results to see confidence scores
- Add alternative phrases to improve scores as necessary

Publish the trained knowledge base

- Creates an HTTP REST-based endpoint for client apps to consume
- Published knowledge base can be used with SDKs within your app

Creating client apps

- REST Interface or SDKs
- Submit questions to the endpoint

```
{
  "question": "What do I need to do to
cancel a reservation?",
  "top": 2,
  "scoreThreshold": 20,
  "strictFilters": [
    {
      "name": "category",
      "value": "api"
    }
  ]
}
```

```
{
  "answers": [
    {
      "score": 27.74823341616769,
      "id": 20,
      "answer": "Call us on 555 123 4567 to
cancel a reservation.",
      "questions": [
        "How can I cancel a reservation?"
      ],
      "metadata": [
        {
          "name": "category",
          "value": "api"
        }
      ]
    }
  ]
}
```

Confidence score

Answer text

Best question match

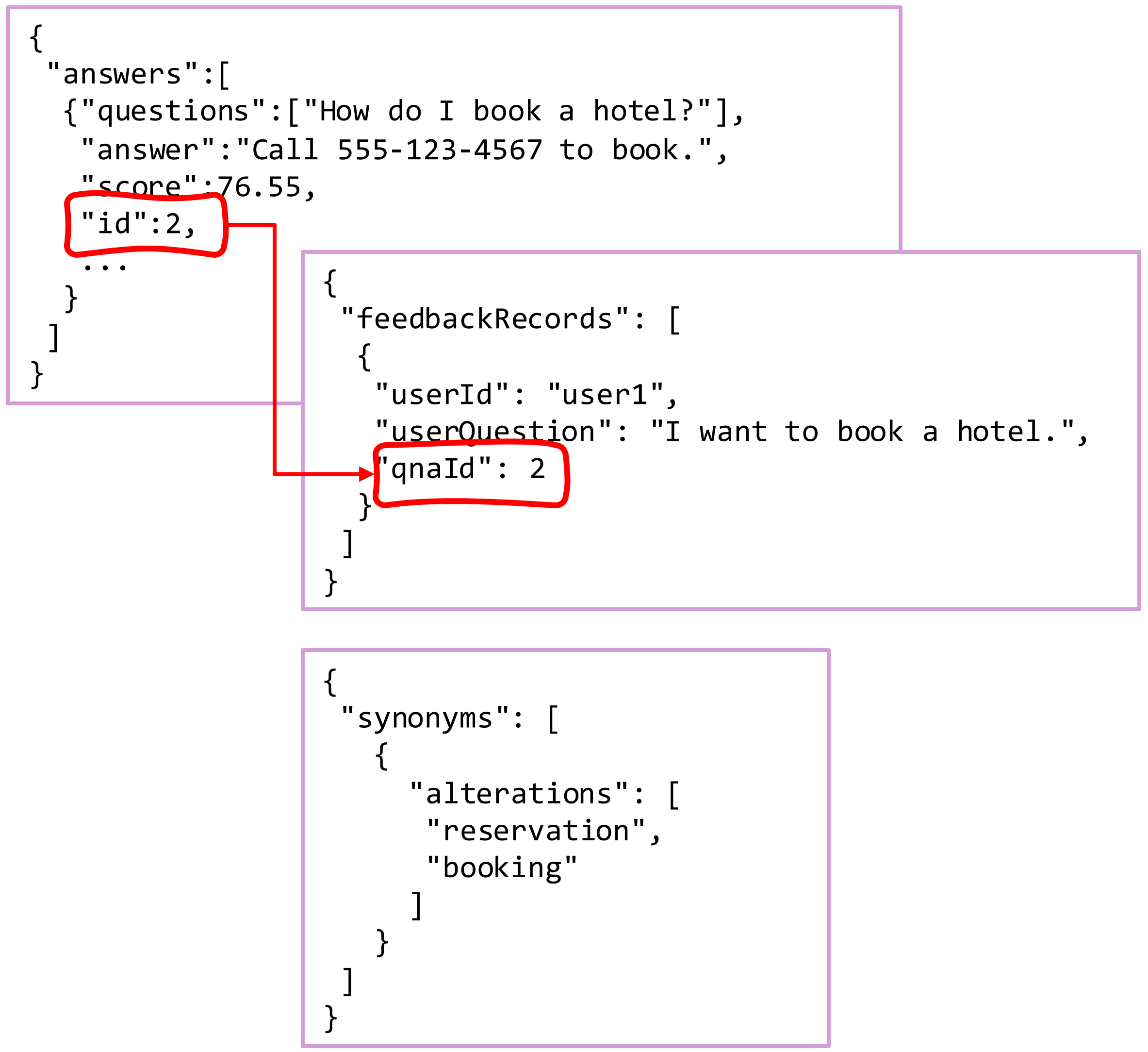
Improving Question Answering Performance

Enable *Active Learning* to suggest alternatives when multiple questions have similar scores for user input

- **Implicit**: The service identifies potential alternative phrases for questions; and presents suggestions in the Language Studio. Periodically review and accept/reject the suggestions.
- **Explicit**: The service returns multiple possible question matches to the user, and the user identifies the correct one. The client app then uses the API to submit feedback items, identifying the correct answer.

Create *Synonyms* for terms with the same meaning

- Add synonyms to the knowledge base through the API or Language Studio interface.



```
{
 "answers":[
  {"questions":["How do I book a hotel?"],
   "answer":"Call 555-123-4567 to book.",
   "score":76.55,
   "id":2,
   ...
  }
 ]
}
```

```
{
 "feedbackRecords": [
  {
   "userId": "user1",
   "userQuestion": "I want to book a hotel.",
   "qnaId": 2
  }
 ]
}
```

```
{
 "synonyms": [
   {
     "alterations": [
      "reservation",
      "booking"
     ]
   }
 ]
}
```

Exercise – Create a Question Answering solution

Create and edit a knowledge base

Train, test, and deploy the knowledge base

Build a conversational language understanding app

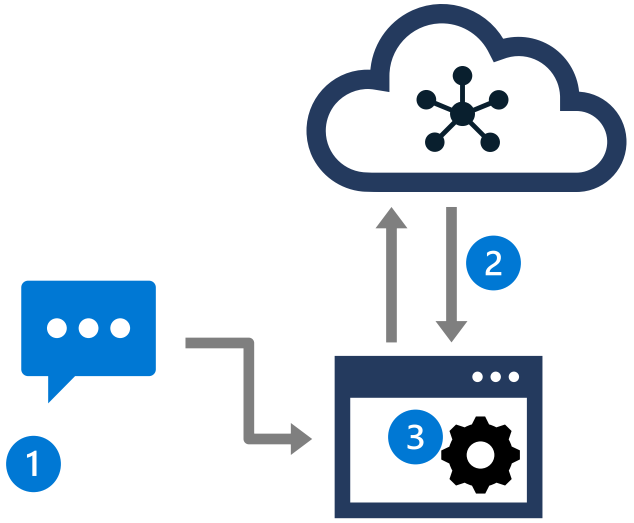
Learning objectives

After completing this module, you will be able to:

1. Provision an Azure AI Language resource
2. Define intents, entities, and utterances
3. Use patterns to differentiate similar utterances and use pre-built entity components
4. Train, test, publish, and review a model
5. Describe Azure AI Language Understanding features

Introduction to language understanding

1. An app accepts natural language input from a user
2. A language model is used to determine semantic meaning (the user's *intent*)
3. The app performs an appropriate action



Natural Language Processing (NLP) requires a language model to interpret user input

Often this activity is referred to as *natural language understanding* (NLU)

Conversational language understanding (CLU) is an Azure service to enable you to build natural language understanding component to be used in an end-to-end conversational application.

Intents and utterances

To train a language understanding model:

- Specify *utterances* that represent expected natural language input
- Map utterances to *intents* that assign semantic meaning

Utterance	Intent
What time is it?	GetTime
Tell me the time.	
What is the weather forecast?	GetWeather
Do I need an umbrella?	
Turn the light on.	TurnOnDevice
Switch on the fan.	
Hello	None

Entities

Define *entities* to add specific context to intents

Utterance	Intent	Entities
What is the time?	GetTime	
What time is it in London?	GetTime	Location (London)
What's the weather forecast for Paris?	GetWeather	Location (Paris)
Will I need an umbrella tonight?	GetWeather	Time (tonight)
What's the forecast for Seattle tomorrow?	GetWeather	Location (Seattle), Time (tomorrow)
Turn the light on.	TurnOnDevice	Device (light)
Switch on the fan.	TurnOnDevice	Device (fan)

Entity types:

Learned	List	Prebuilt
Machine learned through training	Term in a defined list	Common types like numbers and date/times

Prebuilt entity components

Prebuilt components automatically predict common types from utterances:

Quantities

- Age, Number, Percentage, Currency, others…

Datetime

- "June 23, 1976", "7 AM", "6:49 PM", "Tomorrow at 7 PM", "Next Week".

Email

- "user@contoso.com"

Phone number

- US Phone Numbers such as "+1 123 456 7890" or "(123)456-7890".

URL

- "https://learn.microsoft.com/"

Azure AI Language service capabilities

Features fall into two categories:

Preconfigured features – Can be used without labeling or training:

- Summarization
- Named entity recognition
- PII detection
- Key phrase detection
- Sentiment analysis
- Language detection

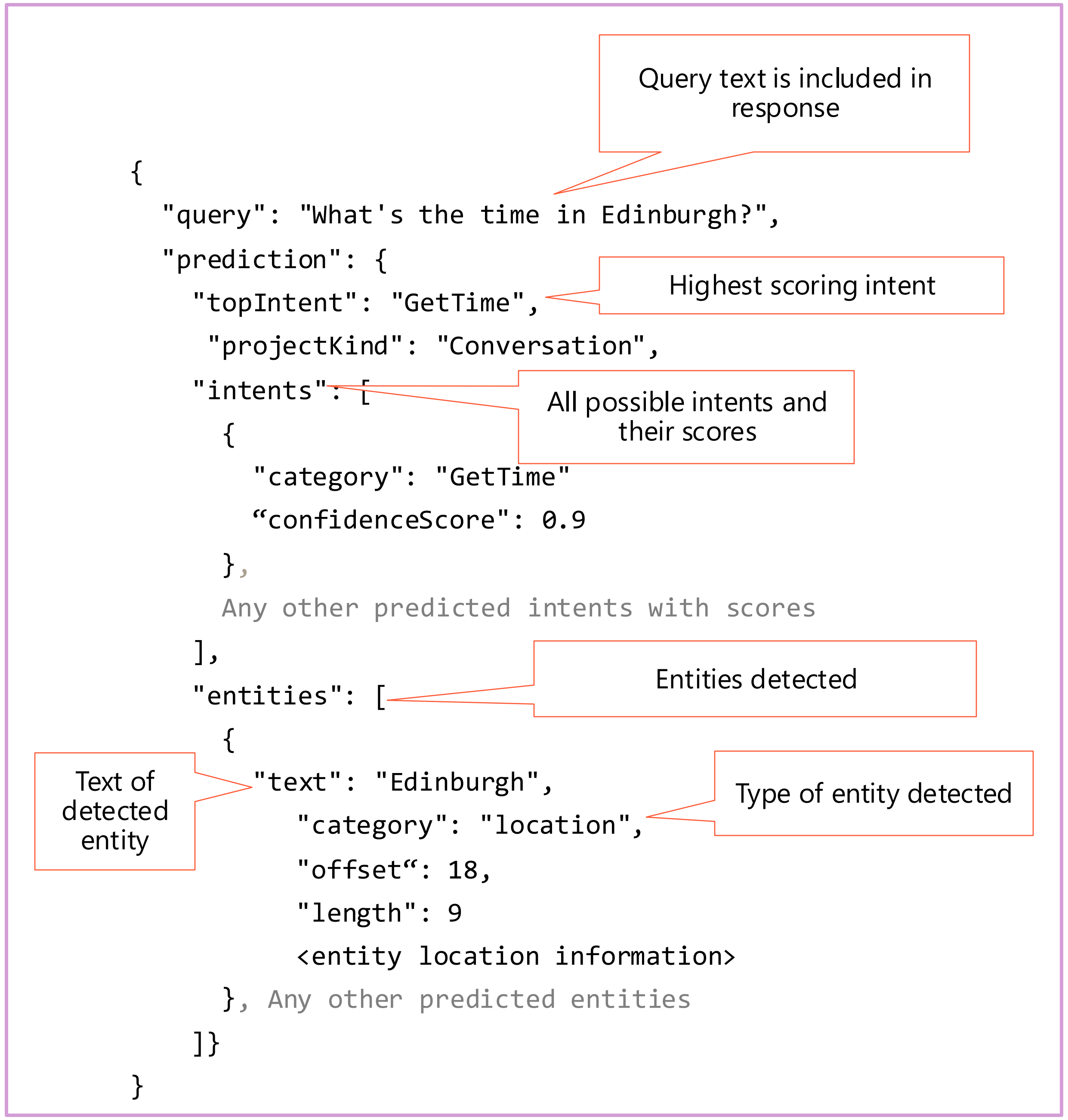
Learned features – Require labeling, training, and deploying to utilize

- Conversational language understanding
- Custom named entity recognition
- Custom text classification
- Question answering

Processing predictions

Submit a request to a published slot, specifying:

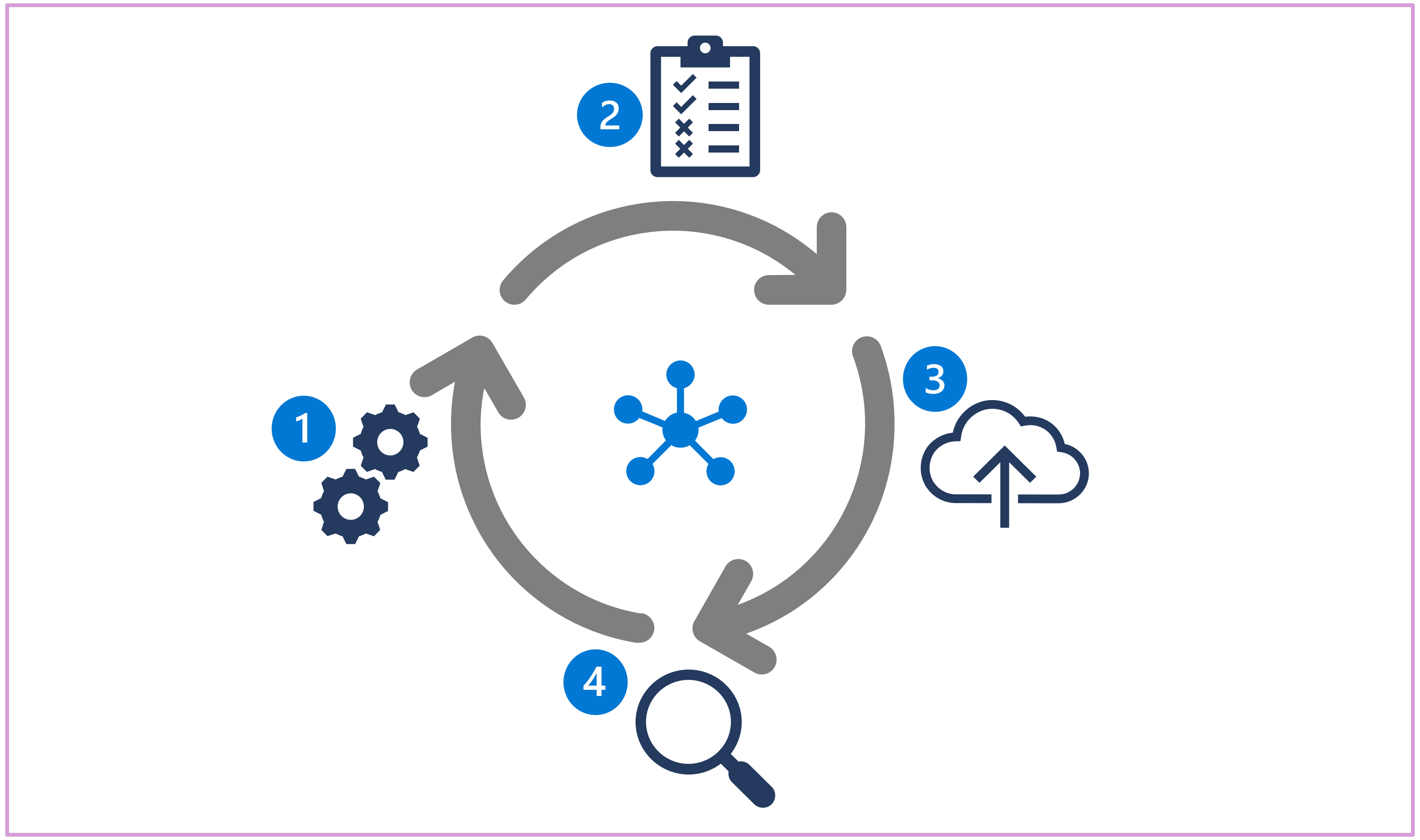
- **Kind –** Indicates which language feature you're requesting. For example, **kind** is defined as *Conversation* for conversational language understanding, or *EntityRecognition* to detect entities
- **Parameters –** Indicates the values for various input parameters. These parameters vary depending on the feature.
- **Analysis input –** Specifies the input documents or text strings to be analyzed by the Azure AI Language service.



```
{
  "query": "What's the time in Edinburgh?",
  "prediction": {
    "topIntent": "GetTime",
     "projectKind": "Conversation",
    "intents": [
      {
        "category": "GetTime"
        “confidenceScore": 0.9
      },
      Any other predicted intents with scores
    ],
    "entities": [
      {
        "text": "Edinburgh",
           "category": "location",
           "offset“: 18,
           "length": 9
           <entity location information>
      }, Any other predicted entities
    ]}
}
```

Training, testing, publishing, and reviewing

1. Train a model to learn intents and entities from sample utterances
2. Test the model interactively or using a testing dataset with known labels
3. Deploy a trained model to a public endpoint so client apps can use it
4. Review predictions and iterate on utterances to train your model



Exercise – Create a conversational language understanding app

Create intents

Create entities

Test and publish a language model

Query your model from a client app

Custom classification and named entity extraction

Learning Objectives

After completing this module, you will be able to:

1. Label documents, train and deploy models for custom classification
2. Understand model performance and see where to improve your model
3. Use your custom model in an app

Custom Text Classification

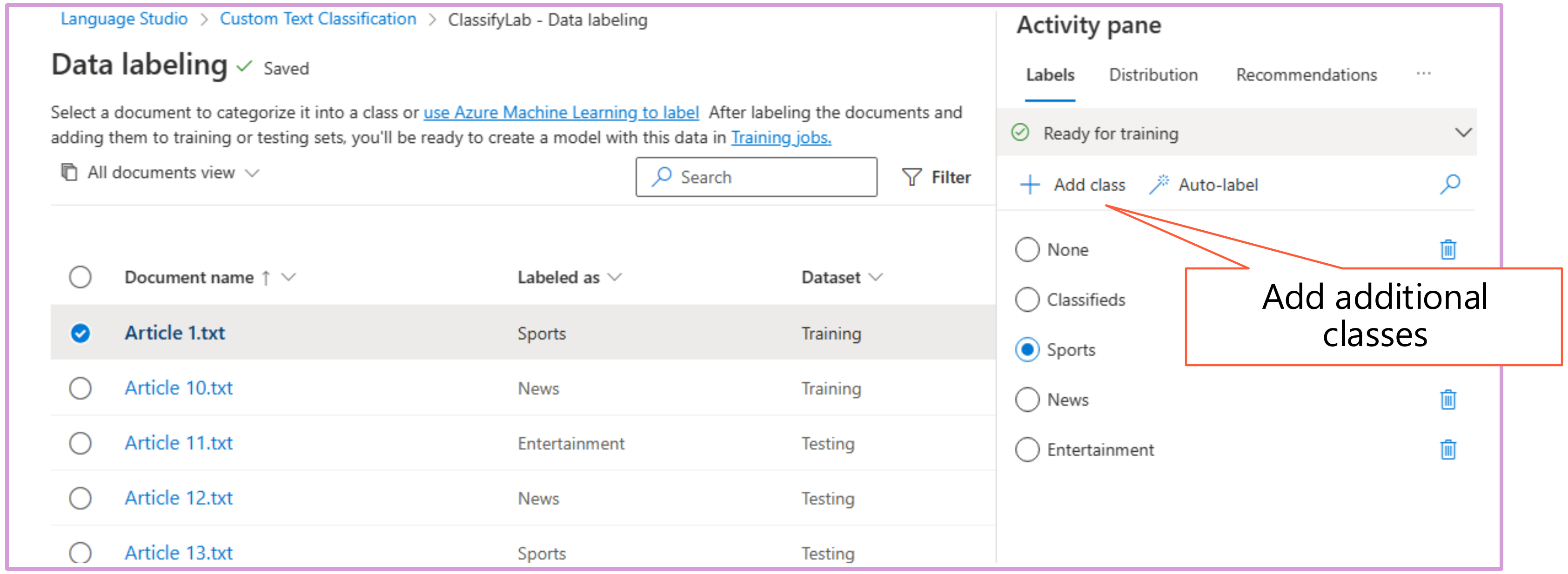
Assign custom labels to documents

1. Connect to documents in Azure
2. Define class labels to assign to your documents
3. Label documents
4. Train your model

Call your model through the Language API

- Specify project and deployment name

Can be single label or multi label projects



Language Studio > Custom Text Classification > ClassifyLab - Data labeling

Data labeling ✓ Saved

Select a document to categorize it into a class or use Azure Machine Learning to label After labeling the documents and adding them to training or testing sets, you'll be ready to create a model with this data in Training jobs.

All documents view | Search | Filter

Document name ↑	Labeled as	Dataset
● Article 1.txt	Sports	Training
Article 10.txt	News	Training
Article 11.txt	Entertainment	Testing
Article 12.txt	News	Testing
Article 13.txt	Sports	Testing

Activity pane

Labels | Distribution | Recommendations

Ready for training

Add class | Auto-label

- None
- Classifieds
- [✓ Sports]
- News
- Entertainment

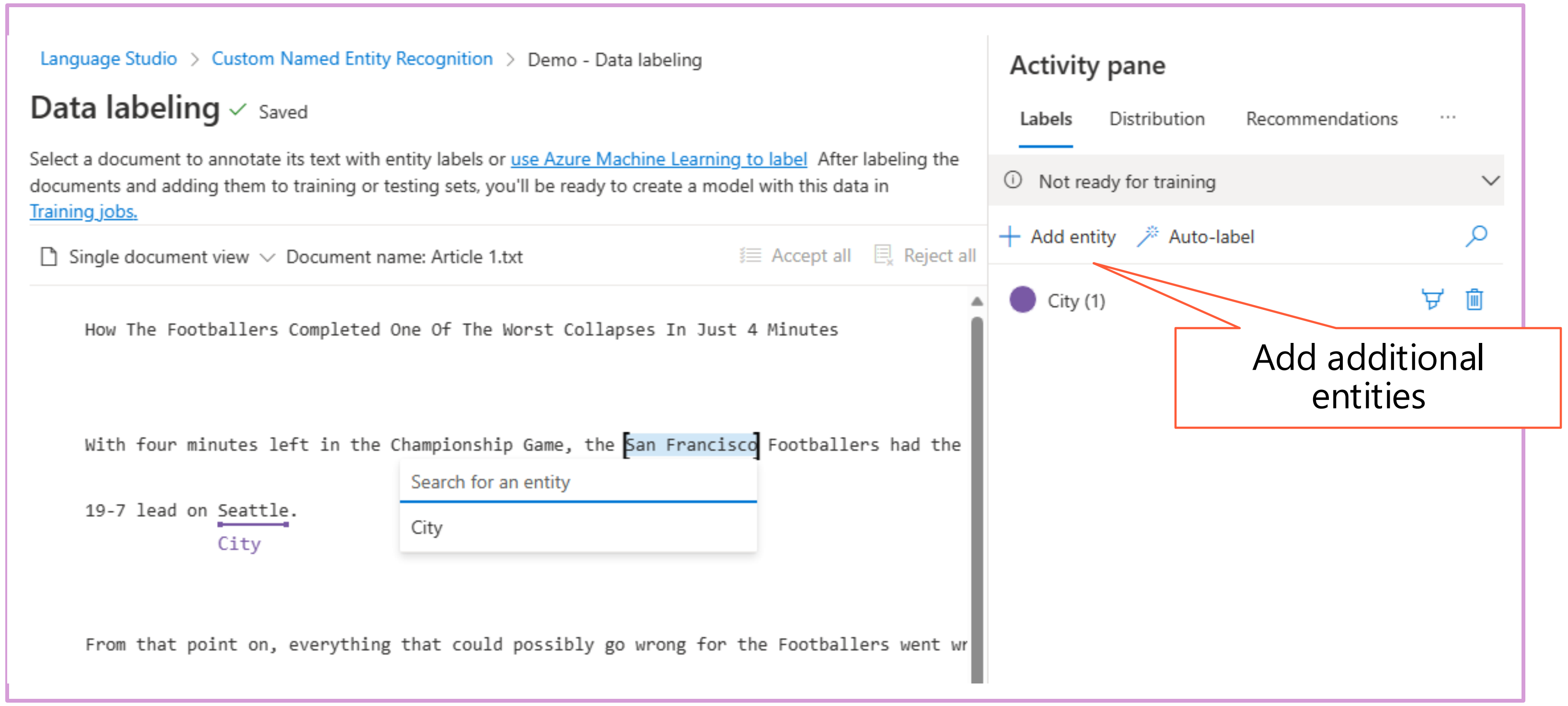
Custom Named Entity Recognition

Assign custom labels to entities in your documents

1. Connect to documents in Azure
2. Define entity labels to assign to your documents
3. Label documents completely and consistently
4. Train your model

Call your model through the Language API

- Specify project and deployment name



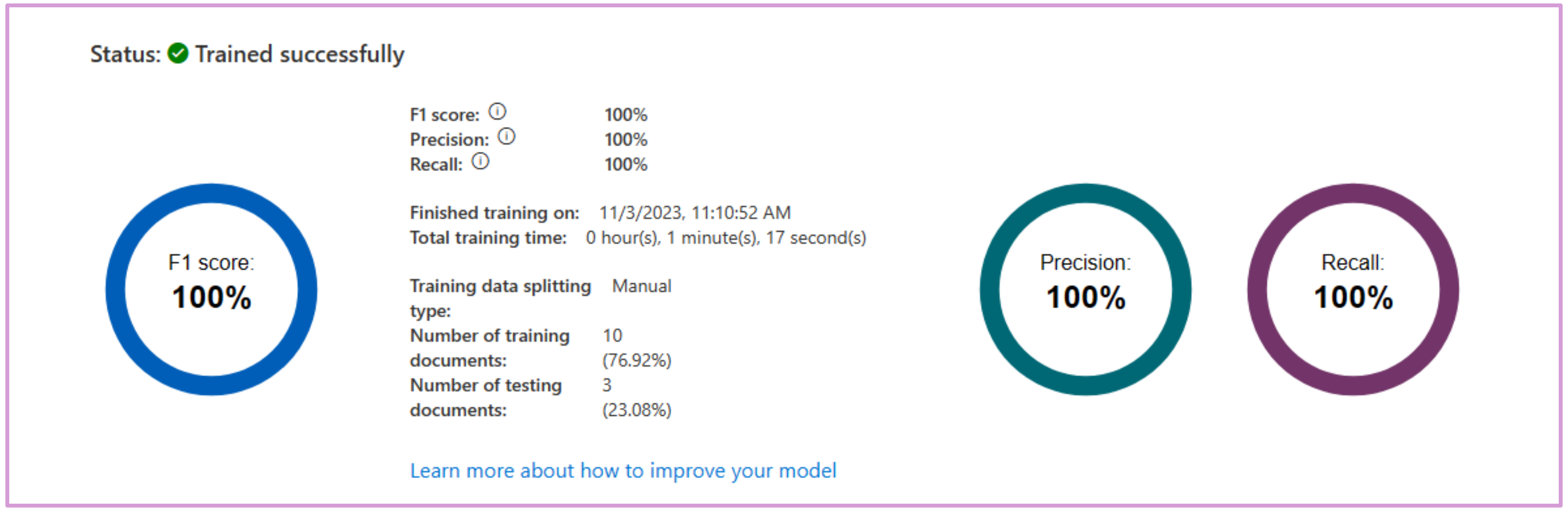
Select a document to annotate its text with entity labels or use Azure Machine Learning to label After labeling the documents and adding them to training or testing sets, you'll be ready to create a model with this data in Training jobs.

Document name: Article 1.txt

```
How The Footballers Completed One Of The Worst Collapses In Just 4 Minutes

With four minutes left in the Championship Game, the San Francisco Footballers had the

19-7 lead on Seattle.
             City

From that point on, everything that could possibly go wrong for the Footballers went wr
```

Review and improve a model

1. Train a model to teach labels or entities
2. Review model performance to determine how to improve performance, including Confusion matrix
3. Determine what cases need to be added to your training data
4. Retrain your model with new data included, and repeat as necessary



Status: Trained successfully

F1 score:	100%
Precision:	100%
Recall:	100%
Finished training on:	11/3/2023, 11:10:52 AM
Total training time:	0 hour(s), 1 minute(s), 17 second(s)
Training data splitting type:	Manual
Number of training documents:	10 (76.92%)
Number of testing documents:	3 (23.08%)

Learn more about how to improve your model

Demo – Extract custom entities

Create a custom named entity recognition project

Create and label entities

Test and publish a custom model

Query your model from a client app

Speech recognition, translation and synthesis

Learning Objectives

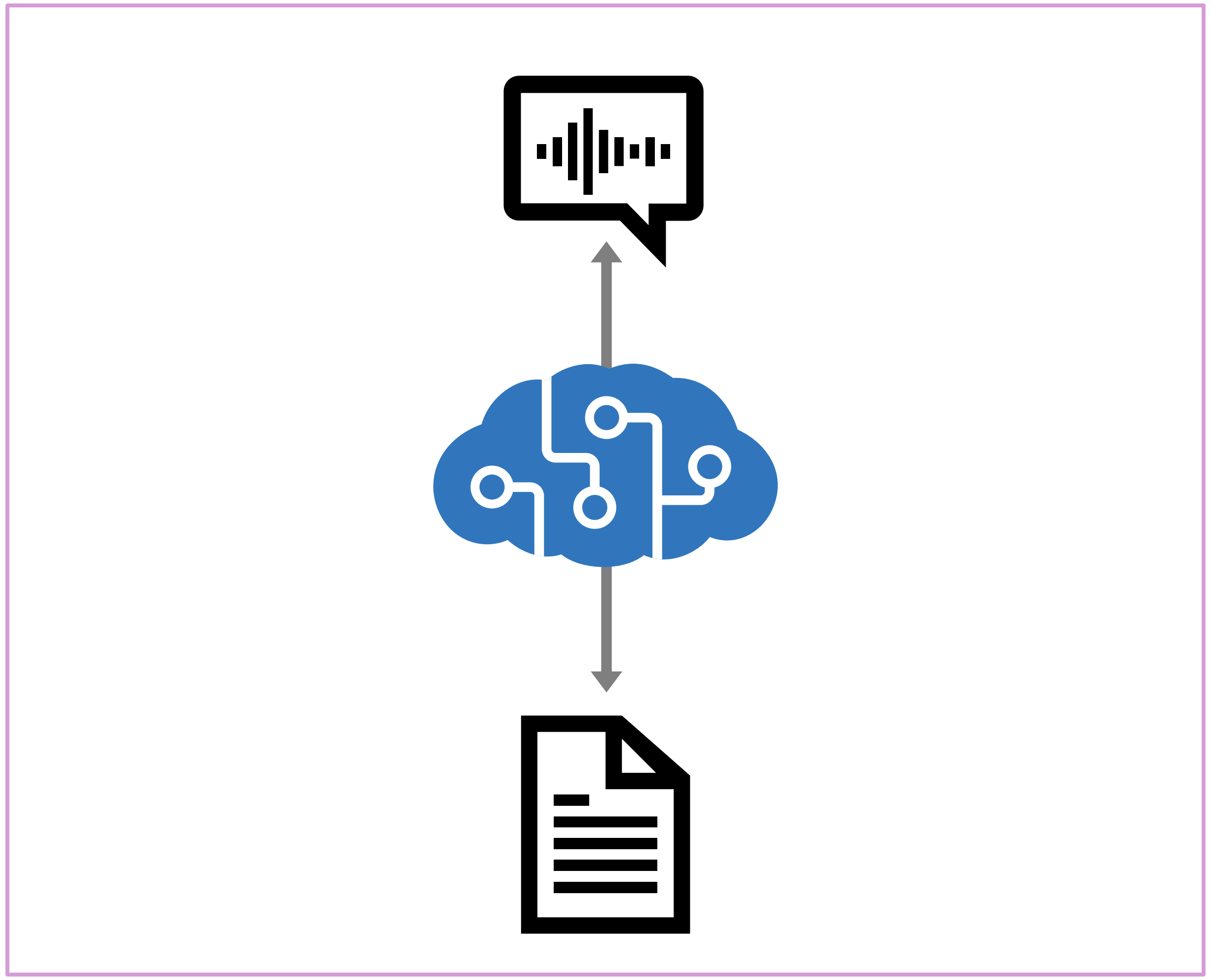
After completing this module, you will be able to:

1. Provision an Azure resource for the Azure AI Speech service
2. Use the Speech to text API to implement speech recognition
3. Use the Text to speech API to implement speech synthesis
4. Configure audio format and voices
5. Use Speech Synthesis Markup Language (SSML)

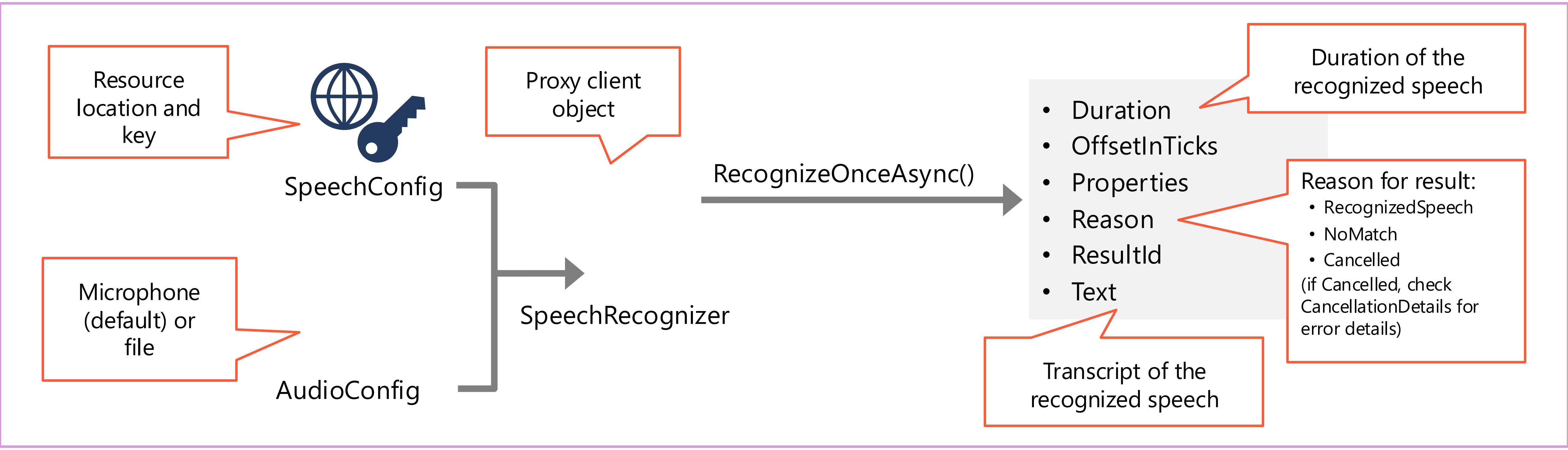
The Speech Service

Speech APIs

- Speech-to-Text API (speech recognition)
- Text-to-Speech API (speech synthesis)
- Speech Translation API
- Speaker Recognition API
- Intent Recognition (uses conversational language understanding)



Speech-to-Text

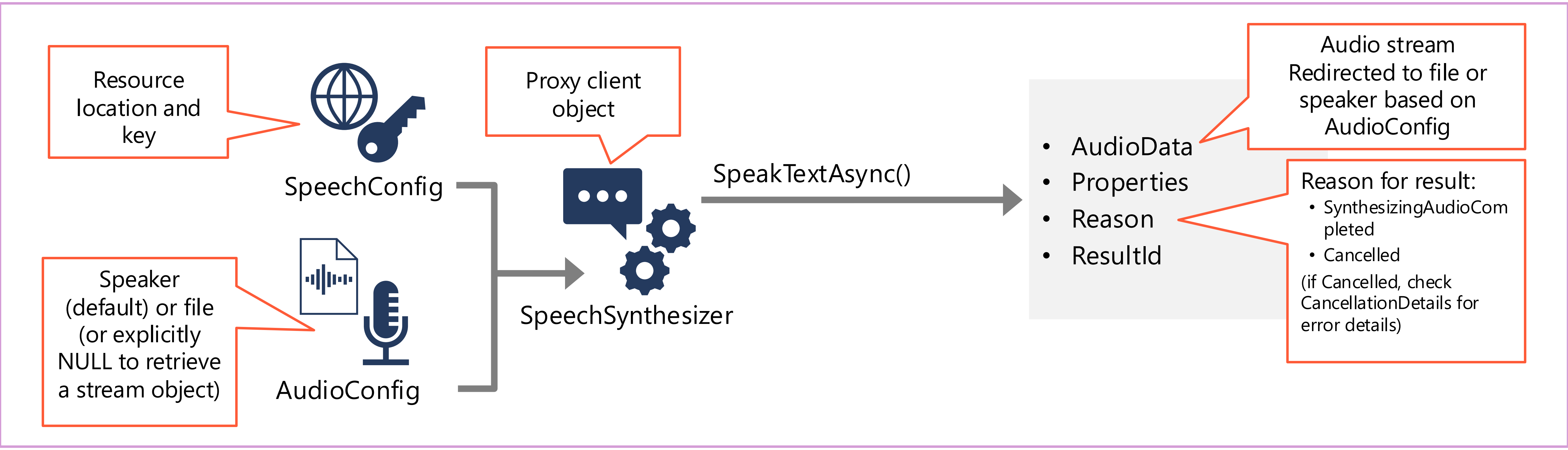


Two REST APIs:

- Speech-to-text API – Used by Azure AI Speech SDK – preferred for most scenarios
- Speech-to-text Short Audio API – Useful for short (up to 60s) of audio

Azure AI Speech SDK (.NET, Python, JavaScript, etc.)

Text-to-Speech


Resource location and key
SpeechConfig
Speaker (default) or file (or explicitly NULL to retrieve a stream object)
AudioConfig
Proxy client object
SpeechSynthesizer
SpeakTextAsync()
• AudioData
• Properties
• Reason
• ResultId
Audio stream Redirected to file or speaker based on AudioConfig
Reason for result:
• SynthesizingAudioCompleted
• Cancelled
(if Cancelled, check CancellationDetails for error details)

Two REST APIs:

- Text-to-speech API – Suitable for most scenarios
- Batch synthesis API – Convert large volumes of text to audio files

Azure AI Speech SDK (.NET, Python, JavaScript, etc.)

Audio Format and Voices

Audio Format

Select an audio format to specify:

- Audio file type
- Sample-rate
- Bit-depth

Voices

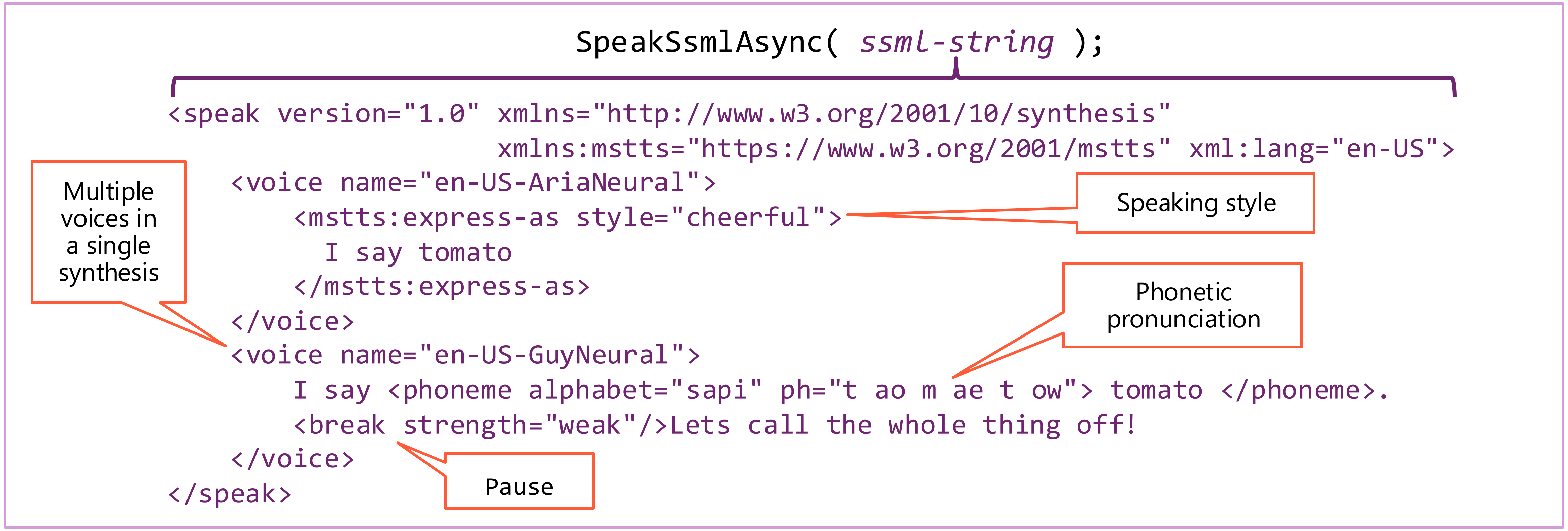
- Standard voices: Synthetic voices created from audio samples
- Neural voices: Natural sounding voices created using deep neural networks

```
speechConfig.SetSpeechSynthesisOutputFormat(SpeechSynthesisOutputFormat.Riff24Khz16BitMonoPcm);

speechConfig.SpeechSynthesisVoiceName = "en-GB-George";
```

Speech Synthesis Markup Language (SSML)

XML-based language with customization options:

- Speaking styles (Neural voices only)
- Pauses and silence
- Phonemes (phonetic pronunciations)
- Prosody (speaking pitch, range, rate, etc.)
- "say-as" (number, date, time, address, etc.)
- Insert recorded speech or background audio



```
SpeakSsmlAsync( ssml-string );
```

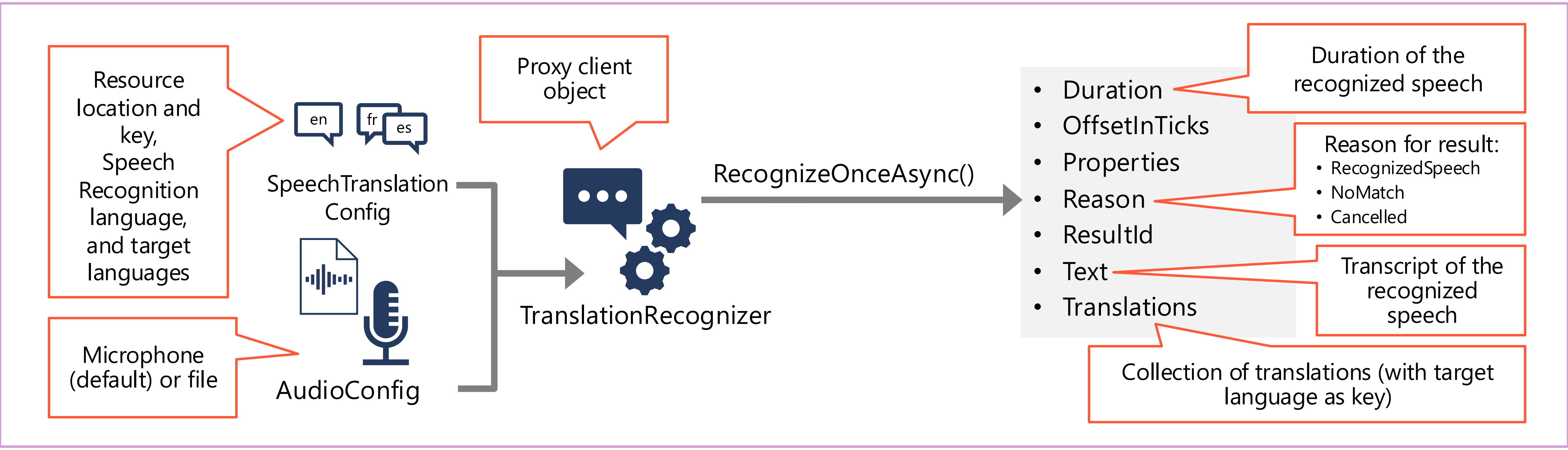
```
<speak version="1.0" xmlns="http://www.w3.org/2001/10/synthesis"
                     xmlns:mstts="https://www.w3.org/2001/mstts" xml:lang="en-US">
    <voice name="en-US-AriaNeural">
        <mstts:express-as style="cheerful">
          I say tomato
        </mstts:express-as>
    </voice>
    <voice name="en-US-GuyNeural">
        I say <phoneme alphabet="sapi" ph="t ao m ae t ow"> tomato </phoneme>.
        <break strength="weak"/>Lets call the whole thing off!
    </voice>
</speak>
```

Exercise – Recognize and Synthesize Speech

Recognize Speech

Synthesize Speech

Translating Speech to Text



Translation builds on speech recognition:

1. Recognize and transcribe spoken input in speech recognition language
2. Return translations for one or more target languages

Synthesizing Translations as Speech

Event-based synthesis

- Only supported for 1:1 translation (single target language)
- Specify desired voice in the **TranslationConfig**
- Use the **Synthesizing** event to retrieve audio stream
- Create an event handler
- Use Result.GetAudio() to retrieve byte stream

Manual synthesis

- Use for multiple target languages
- Translate to text then use Text-to-Speech API to synthesize each translation in the results

Extended interactive exercises

Custom text classification

Translate speech

https://aka.ms/azure-ai-language-lp

Knowledge check

1 **Which object should you use to specify that the speech input to be transcribed to text is in an audio file?**

- ☐ SpeechConfig
- ☑ AudioConfig
- ☐ SpeechRecognizer

2 **You have analyzed text that contains the word “Paris”. How might you determine of this word refers to the French city or the character in Homer’s *The Iliad*?**

- ☐ Use the Azure AI Language service to extract key phrases.
- ☐ Use the Azure AI Language service to analyze sentiment.
- ☑ Use the Azure AI Language service to extract linked entities.

3 **When translating speech, in which cases can you can use the Synthesizing event to synthesize the translations and speech?**

- ☑ topIntent
- ☐ kind
- ☐ query

Knowledge check

4 **Your app must interpret a command to book a flight to a specified city, such as “Book a flight to Paris.” How should you model the city element of the command?**

- ☐ As an intent.
- ☐ As an utterance.
- ✓ As an entity.

5 **Your language model needs to detect an email when present in an utterance. What is the simplest way to extract that email?**

- ☐ Use Regular Expression entities.
- ✓ Use Prebuilt entity components
- ☐ Use Learned entity components.

6 **How should you create an application that monitors the comments on your company’s web site and flags any indication that customers are unhappy?**

- ✓ Use the Azure AI Translator service to detect profanities in comments.
- ☐ Use the Azure AI Language service to perform sentiment analysis of the comments.
- ☐ Use the Azure AI Language service to extract named entities from the comments

Learning Path Recap

In this learning path, we learned to:

- Analyze and translate text
- Build a conversational language understanding model
- Build a question answering solution
- Speech recognition, synthesis, and translation
- Connect an app to Azure AI Language resources

Microsoft