

Post-Call Analytics Solution

Installation Process

There are two components to the install process:

- Part 1 - server installation (including system-wide parameters)
- Part 2 - UI installation
- Part 3 - Quicksight configuration

The first two parts are available in two specific ZIP files, which contain their own instructions - in order to install the system you must first follow the steps in the *Install Server.pdf* file, and only once that has completed successfully should you follow the steps in the *Install UI.pdf* file.

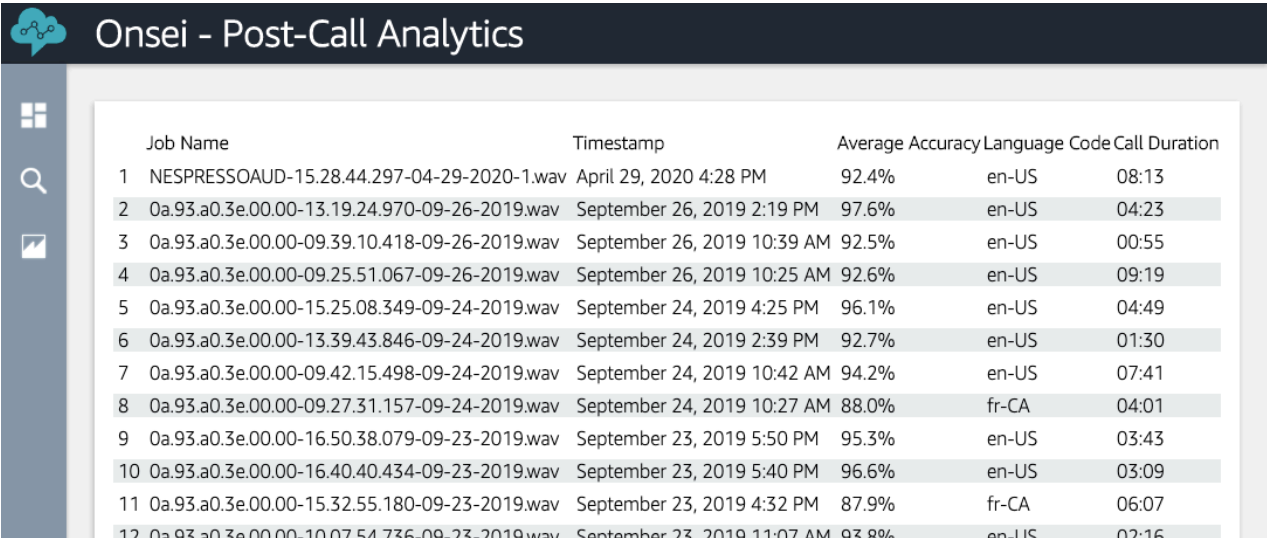
Descriptions of each of the system-wide parameters are shown during the installation process, but they are for reference they are provided later in this document.

Note, you do not need to install the UI in order to make the audio ingestion process completely operational, and only need to follow the Quicksight instructions if you wish to run reporting insights against the processed data.

User Interface Overview

Main Screen

Once you log in to the application, you will see the following main screen, which lists the first 25 calls that are in the system:



	Job Name	Timestamp	Average Accuracy	Language Code	Call Duration
1	NESPRESSOAUD-15.28.44.297-04-29-2020-1.wav	April 29, 2020 4:28 PM	92.4%	en-US	08:13
2	0a.93.a0.3e.00.00-13.19.24.970-09-26-2019.wav	September 26, 2019 2:19 PM	97.6%	en-US	04:23
3	0a.93.a0.3e.00.00-09.39.10.418-09-26-2019.wav	September 26, 2019 10:39 AM	92.5%	en-US	00:55
4	0a.93.a0.3e.00.00-09.25.51.067-09-26-2019.wav	September 26, 2019 10:25 AM	92.6%	en-US	09:19
5	0a.93.a0.3e.00.00-15.25.08.349-09-24-2019.wav	September 24, 2019 4:25 PM	96.1%	en-US	04:49
6	0a.93.a0.3e.00.00-13.39.43.846-09-24-2019.wav	September 24, 2019 2:39 PM	92.7%	en-US	01:30
7	0a.93.a0.3e.00.00-09.42.15.498-09-24-2019.wav	September 24, 2019 10:42 AM	94.2%	en-US	07:41
8	0a.93.a0.3e.00.00-09.27.31.157-09-24-2019.wav	September 24, 2019 10:27 AM	88.0%	fr-CA	04:01
9	0a.93.a0.3e.00.00-16.50.38.079-09-23-2019.wav	September 23, 2019 5:50 PM	95.3%	en-US	03:43
10	0a.93.a0.3e.00.00-16.40.40.434-09-23-2019.wav	September 23, 2019 5:40 PM	96.6%	en-US	03:09
11	0a.93.a0.3e.00.00-15.32.55.180-09-23-2019.wav	September 23, 2019 4:32 PM	87.9%	fr-CA	06:07
12	0a.93.a0.3e.00.00-10.07.54.736-09-23-2019.wav	September 23, 2019 11:07 AM	93.8%	en-US	02:16

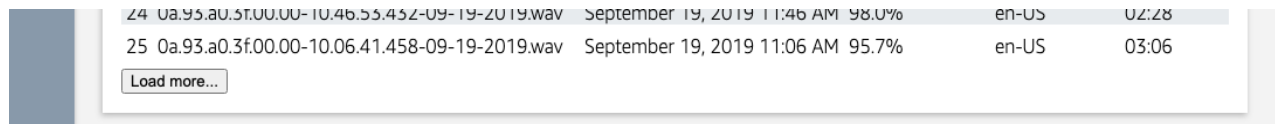
There are 3 navigation buttons to the left that take you to:

- Main Screen
- Search Screen
- Amazon QuickSight

Clicking on any of these buttons at any time will take you to that relevant page, whereas clicking on a call line on the screen will take you to the call details screen.

The columns shown indicate various high-level item data items per file, notably the filename, call timestamp, Amazon Transcribe's estimate of the word-level accuracy score for the call, the language of the call and the duration.

At the very bottom of the screen is an additional button marked **Load More...**

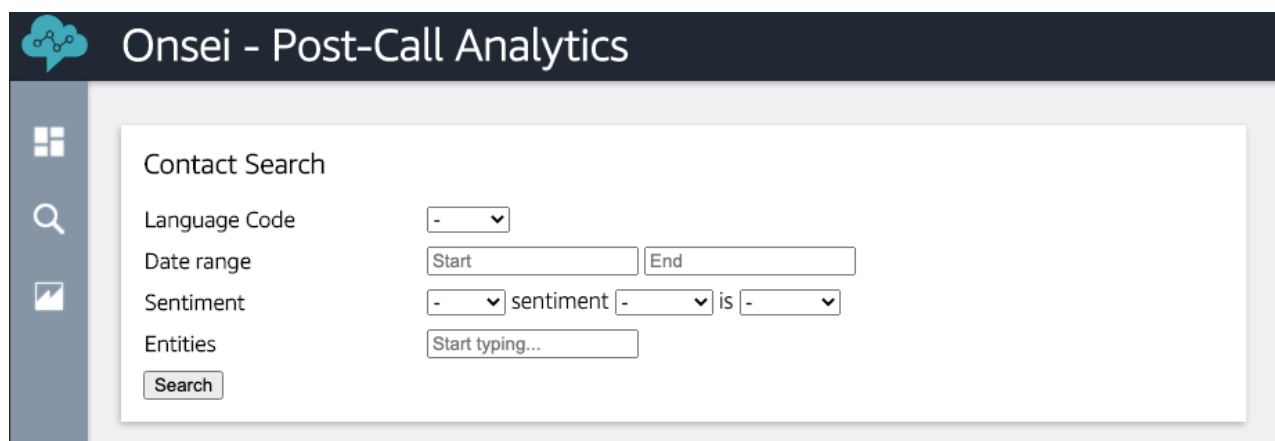


24	0a.93.a0.3f.00.00-10.46.55.452-09-19-2019.wav	September 19, 2019 11:46 AM	98.0%	en-US	02:28
25	0a.93.a0.3f.00.00-10.06.41.458-09-19-2019.wav	September 19, 2019 11:06 AM	95.7%	en-US	03:06
Load more...					

Selecting that button will display another 25 processed files on the screen. However, it is anticipated that the majority of the UI usage will filter the list of viewed calls before any work takes place, and that the majority of the analytics will be within QuickSight.

Search Screen

The search screen will show the following controls, allowing you to show a filtered view of the total available calls.



Onsei - Post-Call Analytics

Contact Search

Language Code:

Date range:

Sentiment: sentiment is

Entities:

Search

There is no facility to search for a specific word in a transcript, but the following items can be searched for:

- **Language Code** - shows options in the drop-down for every language that the application has ingested a file for; e.g., If you have not yet ingested a file in the **es-ES** language then it will not appear as an option, but once you have it will automatically appear
- **Date range** - this will bring up a *Date Picker* window for both the start and end dates that are of interest
- **Sentiment** - you are able to search for specific variations of sentiment using three parameters
 - A specific speaker; e.g., Agent or Caller
 - Average sentiment in a call the chosen speaker, or the trend for that speaker across the call
 - Positive or negative sentiment

- **Entities** - allows the entry of multiple entities for searching. The control knows what entities have been detected in previous calls, and only allows you to enter those as values. As you enter them they appear to the right, and you can clear any previously-entered term just by clicking on the blue box

The screenshot shows a search interface for entities. On the left, there's a 'Search' button. In the center, a search bar contains the text 'kit'. To the right of the search bar is a dropdown menu that is open, showing two options: 'Kit Kat' and 'kit kat'. Further to the right, there is a 'boost' button.

The results will appear once you hit the **search** button, and appear below, and you can interact with the results list in the same way as per the Main Screen.

Call Details Screen

The details screen includes a lot of different information, split into three segments:

The screenshot shows the 'Onsei - Post-Call Analytics' screen. It is divided into several sections:

- Contact Summary:**
 - Date / Time: September 23, 2019 4:40 PM
 - Entity Recognizer Name: mixed-entity-list.csv
 - Language Code: en-US
 - Agent Sentiment: Overall positive; trending up.
 - Caller Sentiment: Overall positive; trending up.
- Source Information:**
 - Type: Voice Transcription
 - Job Name: 0a.93.a0.3e.00.00-16.40.40.434-09-23-2019.wav
 - File Format: wav
 - Call Duration: 03:09
 - Sample Rate: 8000
 - Custom Vocabulary: nci-vocab-en-us
 - Word Accuracy: 96.6%
- Sentiment Trend Graph:** A line graph showing 'Call Sentiment over time' for Agent (yellow) and Caller (blue). The y-axis ranges from -0.6 to 1.0. The Agent's sentiment starts around 0.8, dips to 0.6, and then rises back to 0.8. The Caller's sentiment starts around 0.8, dips to -0.4, and then rises back to 0.8.
- Entities:** A list of detected entities: LOCATION × 3, ISSUE/CONCERN × 7, BRAND × 10, INGREDIENT × 8.
- Recording:** A progress bar showing 0:00 / 3:09.
- Transcript:**
 - Agent 00:01: Thank you for calling. My name is [redacted]. How may I help you today?
 - Caller 00:05: Hi, [redacted] from [redacted] from Edmonton. What? Uh, my name is [redacted]. I'm from Edmonton. Okay. Yeah, actually, like, um, we have received as a gift a box of chocolates. Okay, um, from Nestle. And, uh, it is a mixed Box. Like it has Coffee Crisp, Smarties, Kit Kat and Aero. Okay, um, it contains the regular size Bars and also the Mini Bars. Okay, Uh, now, like, uh, like, we, uh, we need to know, like, uh, any of the ingredients used in the making process is like, have some an email fat or an email in the audience in that other than Milk.

At the very top of the screen there are three buttons - two will navigate to the next or previous call in the call-list - using the whole list or the search results list, depending on how you got here - and the third button will swap the Agent/Caller tags on the call. This is because sometimes on calls Transcribe will attribute the speakers incorrectly - this button simply updates the database with the swapped names.

Top Section - Header-Level Information

The top section is split into 3 areas:

- Call time, along with NLP results for sentiment and the entity detection modes/files used
- Source information, which is Transcribe-specific information including the word-accuracy score and which Transcribe Custom Vocabulary was used when processing this call
- Sentiment trends, showing the agent and call sentiment levels through the call

Middle Section - Custom Entities

The middle section will contain all custom entities that have been identified in this audio transcript. As per Amazon Comprehend's documentation, each entity has a type (such as "Brand") and a value (such as "Aero"). This section summarises which entities exist in this file, and if you hover the cursor over one of the entity type names then it will show which entity values it has found.

Bottom Section - Call Transcript

This is the more complex section. At the top there is an audio control playback, which provides the ability to not only play back the audio but to also to change the volume and scrub your way through the call to get to a particular section.

That is followed by the call transcript itself, which has been split into individual turns on a per-speaker basis. If a single speaker pauses for over 3 seconds then their turn is split up to indicate that pause, and will be shown over multiple lines. Each turn also includes the following:

- **speaker label** - indicates either Agent or Caller
- **sentiment indicator icon** - if you hover the cursor over will display the raw sentiment strength score from Amazon Comprehend
- **call timestamp** - clicking this will cause the audio to playback from that point of the conversation

Each transcript line contains the text as output by Amazon Transcribe, which will be impacted by any Custom Vocabulary that you have defined. Other than the plain text there are a number of on-screen indicators to give you additional information:

- **Word-level confidence** - hover the cursor over a word to see the confidence percentage score. As a visual guide, words lower than 50% confidence are shown in red text, and those lower than 90% are underlined
- **PII redaction** - if PII is enabled, then words identified as such as replaced with [redacted]
- **Entity detection** - any entities that have been detected in the transcript are highlighted with the same colour as its associated entity type from the middle section

Audio Ingestion

Ingestion Basics

Compatible audio files need to be delivered by some means to a folder within an Amazon S3 bucket, as defined by the the following configuration values:

- **Input S3 Bucket** - *InputBucketName*
- **Input Folder** - *InputBucketRawAudio*

The upload of a file will trigger the main processing workflow - see the **Application Architecture** section for a full description of that flow. Once the transcription process has finished, along with any other post-processing of the turns and use of Amazon Comprehend, then a results file is delivered in JSON format to the following configured location:

- **Output S3 Bucket** - *OutputBucketName*

- **Output Folder** - *OutputBucketParsedResults*

Once that JSON file has been delivered then the application database will automatically be updated, and the results will be viewable within the application user interface.

The raw output from Transcribe is delivered to the output S3 bucket and is used by the application post-processing process. They are currently not required after this time, but in the future if the application adds a feature to re-process the original file with a new Custom Entity definition then it could use these entries and not need to re-process the original audio file through Transcribe - hence, such updates could be applied almost instantaneously.

MP3 Playback File Generation

The control used on the user interface to playback the audio is the standard HTML5 control. This can playback many formats, but is one combination that it cannot play - 8Khz WAV files encoded in a specific way. If the application finds such a file then it will create an MP3 version of the original audio and refer to that in the user interface for playback - the original file is still the one used for transcription. The location of this MP3 file is in the standard ingestion bucket, but in this configured folder location:

- **MP3 Playback Folder** - *InputBucketAudioPlayback*

In all other cases, the original audio file is used for playback within the user interface.

File Naming Conventions

There are no restrictions on what the filename of the audio files can be, but currently the application performs some regular expression parsing on the filename in order to extract the date and time of the call. An example filename could be:

CallCentreOutput-09.25.51.067-09-26-2019.wav

Currently, the application attempts to interpret this as **09:25.51am on 26th September 2019**. It assumes that those values appear in that format of DD.DD.DDD-DD-DD-DDDD in the filename, and if it does then it calculates the date and time from that. The filename prefix and any suffix to this date is ignored, but if the date and time cannot be parsed in this way then the literal current date/time at the point of ingest is used as the call time.

NOTE 1: *it is known that customers may want to indicate an Agent Name or other identifier in the filename. This is being worked on as an additional feature, which will be available in the near future.*

NOTE 2: *a known bug exists in that the times are interpreted as being in the UTC timezone of +0 GMT. This is to be changed to be based upon a new configuration value so that the customer can specify the timezone to used at ingest so that the call time always shows the correct time relative to that ingest timezone. For instance, in this example, if the configured timezone was EST (Easter Standard Time) then it would show as 9.25.51am in Toronto or New York, but as 2.25.41pm in London.*

Language Detection

There is a configuration setting named *TranscribeLanguages* that defines how which language is used. It has the following logic:

- If it is set to a single language, such as **en-US**, then language detection is disabled and all ingested files are processed in this language
- If it is set to multiple languages separated by " | ", such as **en-US | fr-CA**, then language detection is enabled

The language detection Lambda functions will use that list of languages to tell Transcribe which languages to use as the basis of its identification, and each language will be assigned a confidence factor of being the dominant language in the audio file - that dominant language is then used for the main transcription.

Alternative Second Language

Language detection does take a short amount of time to do, and does trigger an additional 30-second Transcribe job to be created - this also has a small cost implication, as it effectively increases the billable time for each source audio file by an additional 30 seconds. Hence, there is a secondary mechanism available to deliver files that should always be handled in a fixed second language - this language is defined by the configuration parameter *TranscribeAlternateLanguage*.

This feature is not enabled by default, and you will need to follow these steps in order to enable it. Please refer to the AWS documentation for explanations of each step:

1. Create a new bucket in Amazon S3 named accordingly for "alternative language bucket"
2. In the AWS Lambda console, find the function called **pca-aws-file-drop-trigger** and open it
3. Ensure that the **Designer** frame is open
4. Click on the **Add Trigger** button inside that frame

This will bring up a new dialog, where you should ensure that the following is entered:

- S3 is the trigger source
- The *Bucket* is set to the name of the bucket that you created above

Leave the Event type to be "All object create events", select "Enable trigger", acknowledge the warning and hit the **Add** button. This will now mean that any files dropped into this bucket will also trigger the overall transcription workflow, but it will force transcription to use the language defined by the configuration setting *TranscribeAlternateLanguage*.

Note: if you have defined multiple languages in *TranscribeLanguages* then language detection will be enabled, and files delivered through this alternate mechanism will still go through language detection.

Call Failures

Amazon Transcribe can fail to process for several reasons. Typically, the reason is that Transcribe's Language ID function has failed to find a dominant language due to the file being too short in duration; it needs 30 seconds to positively identify that, so calls shorter than this will be effectively in an unknown and the application will be unable to proceed - hence, these files are marked as FAILED.

Occasionally, Transcribe itself might find an issue with the audio, such as too much background noise or audio interference to be able to make any kind of transcription. Again, these files will be marked as FAILED.

Finally, in the case of any kind of unexpected error during the transcription, the application will retry the transcription attempt up to 2 more times - if it fails for a third time then the whole process is marked as FAILED.

If a call fails then there is no indication on the application user interface, but the application will move the original call audio file from the ingestion location to one that holds all of the failed audio files. This location is the same bucket as the ingestion bucket, but in this configured folder:

- **Failed Call Folder** - *InputBucketFailedTranscriptions*

Bulk Upload

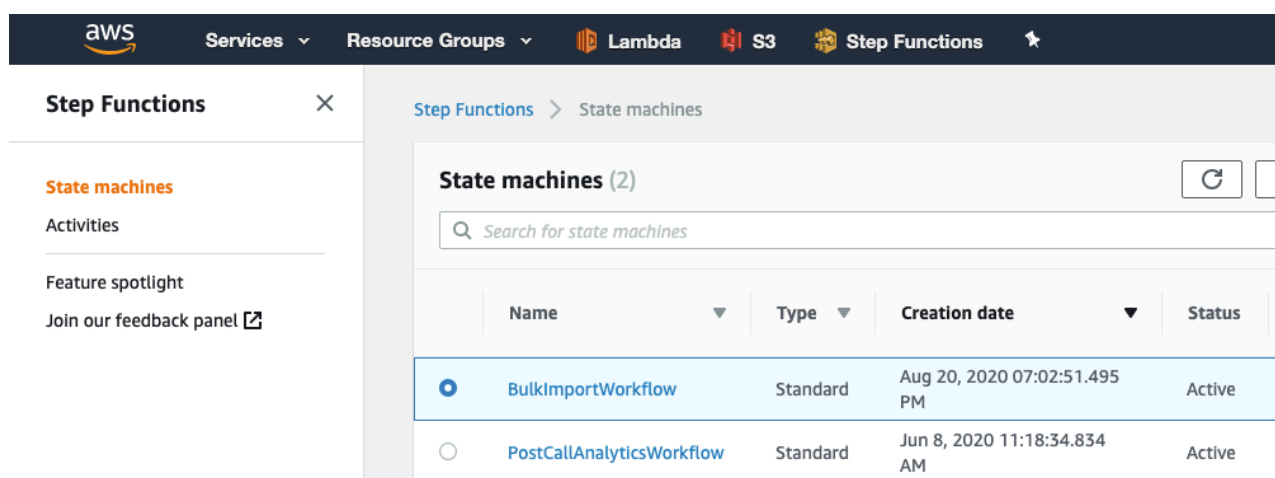
Overview

The bulk upload process is designed to allow you to ingest a large number of files as quickly as possible without interfering with your standard throughput levels of current calls being delivered from your Call Centre solution.

By default, an AWS Account is able to process 100 simultaneous audio file through Transcribe's batch process. If you submit more than this then an automatic queuing mechanism is used, which will queue those jobs within Transcribe until there's processing capacity available in your account.

Enabling Bulk Upload

Upload all files that you wish to bulk upload to the Amazon S3 bucket currently defined in the configuration parameter *BulkUploadBucket*. There is no limitation on the number of files that you can place here. However, the upload process itself will initially be disabled - you need to switch it on via the AWS Console by going to the Step Functions page and selecting the state machine called *BulkImportWorkflow*.



The screenshot shows the AWS Step Functions console. The left sidebar has a 'Step Functions' header and a 'State machines' section. The main content area is titled 'Step Functions > State machines' and shows a list of state machines. The 'BulkImportWorkflow' state machine is selected and highlighted in blue. It is a 'Standard' type, created on 'Aug 20, 2020 07:02:51.495 PM', and is 'Active'. Another state machine, 'PostCallAnalyticsWorkflow', is also listed below it, created on 'Jun 8, 2020 11:18:34.834 AM', and is also 'Active'.

	Name	Type	Creation date	Status
<input checked="" type="radio"/>	BulkImportWorkflow	Standard	Aug 20, 2020 07:02:51.495 PM	Active
<input type="radio"/>	PostCallAnalyticsWorkflow	Standard	Jun 8, 2020 11:18:34.834 AM	Active

This takes you to a screen that lists all previous invocations of this Step Function. On the top-right of this screen is a button marked `Start execution` - click that, and in the resultant dialog click another `Start execution` button in the bottom-right of the screen.

This will start the workflow, which will work its way through each file in the S3 bucket. You can continue to add new files to the bucket whilst the workflow is in progress, but as soon as the workflow spots that the S3 bucket is empty then it will terminate and you will need to re-enable it following these steps.

Processing Method

The workflow simply moves audio files from the bulk upload bucket to the standard location for audio file ingestion - hence, the actual ingest of files via the bulk loading is exactly the same as for any other audio files dropped into that bucket.

There are two key configuration control parameters to this process:

- *BulkUploadMaxTranscribeJobs*
- *BulkUploadMaxDripRate*

The drip-rate parameter defines the maximum number of number of files that the upload process moves at one time. Once it has moved this number of files it will sleep for one minute, and then move another batch of files - this continues until the bucket is empty. However, as an AWS Account has a limit on the number of concurrent Transcribe jobs that could be processed at once, the upload process ensures that it will not move a file if that would mean that more than *BulkUploadMaxTranscribeJobs* audio files are currently being processed. Hence, even if you've set the dip-rate parameter to 50 there will be times when fewer files than this are moved in each of the workflow's 1-minute cycles.

We recommend that the *BulkUploadMaxTranscribeJobs* configuration parameter is set to one half of your account limit for concurrent jobs; e.g., if you have the default limit of 100 then set it to 50, but if you have raised it via AWS Support to 500 then set this parameter to 250. This should leave sufficient space in your account to deal with live incoming calls without any noticeable slowdown of their processing.

Custom Vocabularies

The Custom Vocabulary feature of Transcribe allows customers to make the service pay particular attention to words that they need to be correctly transcribed that are important - typically this is brand names, internal project names, or common colloquial terms.

Custom Vocabulary File Format

The file format for this is a TAB-separated text file. This contains a single header line and four columns - it must be stressed that even if a column value for a particular row is blank then you must still tab to it, and you must not use space characters to line things up visually. An example few lines are shown below, but note that the columns can appear in any order in your file.

Phrase	DisplayAs	SoundsLike	IPA
Los-Angeles	Los Angeles		
F.B.I.	FBI		ɛ f b i aɪ
Etienne		eh-tee-en	

Only the Phrase column is mandatory, you can have one of either SoundsLike or IPA (but never both), and DisplayAs is used if you want the phrase displayed in a way different from Phrase. There are many rules that you have to adhere to:

- Any space character in a multi-word Phrase must be replaced by a -
- Any pronounced letters in a Phrase, such as in acronyms like F.B.I., must have a period after each
- Plurals of acronyms must separate out the 's from the acronym - e.g. FBIs => F.B.I.-s
- Each sound hint in a SoundsLike field must be separated by a -
- IPA characters, or IPA character pairs, must be separated by a space
- There should be no leading or trailing spaces in any column

This guide cannot teach someone how to build phonetic representations - the Amazon Transcribe documentation contains all necessary technical details at this URL: <https://docs.aws.amazon.com/transcribe/latest/dg/how-vocabulary.html>. For further assistance and guidance on building your own vocabulary files, please contact AWS.

Custom Vocabulary Application Configuration

The application lets you configure a base name for your vocabularies in the field *VocabularyName*. This becomes the base name for all vocabulary files in a set, allowing you to define multiple language variants of the same vocabular. For instance, if you configure the base name as **production-vocab** then the application will try and use a Transcribe Custom Vocabulary as follows:

- Audio language = en-US => use vocabulary **production-vocab-en-us**
- Audio language = fr-CA => use vocabulary **production-vocab-fr-ca**

Hence, whenever a file is processed, then the application will attempt to use vocabulary based upon the configured base name and the specified or identified language code, with the language code always appearing in lower-case. This means that for one installation you only have to make a single configuration setting for the vocabulary, and the application will use the appropriate language-specific variant.

If a language variation custom vocabulary does not exist, such as **production-vocab-es-us** not being defined in this case, then audio files ingested in the language **es-US** will be processed without any custom vocabulary - this will be indicated in the Call Details screen for that call, as the *Custom Vocabulary* field will show "---" rather than the name of the vocabulary use.

Creating a Custom Vocabulary

The vocabulary file, with the relevant language suffix, needs to be uploaded to an S3 bucket that you create - there is no restriction on this bucket, but it is recommended that you use the bucket defined in the *SupportFilesBucketName* configuration setting, which is used for other ancillary files such as entity files for entity recognition. Whilst it isn't mandatory, we recommend that you name the files such that they match the vocabulary base name and the target language, so for the Spanish file in the above example we would recommend that the file is called **production-vocab-es-us.txt** - you may end up have multiple language variant files in this bucket, so following this convention will make them easier to manage.

Once uploaded then you need to navigate to the Amazon Transcribe section of the AWS Console and, on the left-hand menu, select *Custom vocabulary*. Hit the **Create vocabulary** button and you will be presented with this screen:

The screenshot shows the 'Create vocabulary' page in the Amazon Transcribe console. The breadcrumb trail at the top is 'Amazon Transcribe > Custom vocabulary > Create vocabulary'. The main heading is 'Create vocabulary' with an 'Info' link. Below this is a 'Vocabulary settings' section. It contains a 'Name' field with the value 'production-vocab-es-us' and a note: 'The name can be up to 200 characters long. Valid characters are a-z, A-Z, and 0-9.' The 'Language' dropdown is set to 'Spanish, US (es-US)'. Under 'Vocabulary input source', 'S3 location' is selected with the note 'Table format only'. The 'Vocabulary file location on S3' section has a text input field containing 's3://pca-custom-source-files/production-vocab-es-us.txt' and a 'Browse S3' button. A note below the input field states: 'File format: txt, maximum size 50 KB.' At the bottom right are 'Cancel' and 'Create vocabulary' buttons.

Enter the parameters as follows:

- **Name** - the name of the vocabulary variant, matching the required naming conventions
- **Language** - the language that this vocabulary variant is in
- **Vocabulary input source** - select *S3 Location*, as table-formatted vocabulary files can only be imported from an S3 source locations
- **Vocabulary file location on S3** - use the **Browse S3** button to file your uploaded file

Amazon Transcribe will then begin the vocabulary generation process, which will take several minutes. If there are any errors in the file, most likely in the IPA sections, then the process will terminate quickly with an error that should tell you roughly what the problem is with your file.

For further assistance with any errors, please contact AWS Support.

Custom Entity Detection

Entity Detection Options

Custom Entity detection via Amazon Comprehend requires a large amount of training data, and initially customers will not have this available. Essentially, it needs a list of customer entities and example sentences where those entities have been seen in context. This takes time to generate, so this application allows you to define just the entity list, and then it will use string-search techniques to identify them in the call transcripts. This will not find the entities in context, rather it will see them wherever they occur, but this means that once a customer has generated lots of call transcript data then it would not take long to train one of the two Comprehend Custom Entity mode typesl.

The following configuration settings apply to entity detection:

- *EntityRecognizerEndpoint* - the Comprehend Custom Entity Recognizer model endpoint
- *EntityStringMap* - base name of a CSV file containing the customer entity values
- *EntityThreshold* - confidence threshold for accepting a Comprehend custom entity

If a *EntityRecognizerEndpoint* is defined and is available for the call language then it is used, and the *EntityThreshold* value is used to decide whether or not to inject the entity data into the call. If the model does not exist for that language then the *EntityStringMap* file is loaded instead. The application requires no further configuration, and will use the relevant method at all times.

Both the endpoint name and the CSV filename are both basenames, very similar to the custom vocabulary naming method. You declare in the configuration a basename, and then the application will append the language code to it and look for that file - again, this lets you define a single name that is applicable to multiple endpoints or files. For example, an English call would make the following interpretation:

- Endpoint name: production-entities => use endpoint **production-entities-en**
- CSV map file: production-entities.csv => use CSV file **production-entities-en.csv**

Note that unlike Transcribe, Comprehend just uses the language code and ignores the regional variant, so **fr-FR** and **fr-CA** would both map to a **-fr** addition to the names.

Simple File-Based Entity Search

There is no additional configuration required to use the simple file-based entity search mechanism - if there is no Comprehend Custom Entity endpoint available for the current language, but a CSV file can be found for the configured basename, then the feature is automatically enabled.

The format of the CSV file, which exactly matches that for the Custom Entities source file, is as follows:

Text	Type
Amazon	CORPORATE
Kindle	BRAND
Fire TV	BRAND
Prime Video	SERVICE

You define your entity types - there is no limit to the number of types that you can create, but we recommend no more than 25 types, as that currently is the limit of entity types on Amazon Comprehend's Custom Entity Detection; if you use more than that limit here then you will not be able to migrate the file to the custom model in the future.

In order to use it the file must be uploaded to the S3 bucket defined in the *SupportFilesBucketName* configuration setting. Once there the application will load it in as and when necessary.

Amazon Comprehend Custom Entities

All information required to create a Custom Entity Detection model is in the AWS documentation for Comprehend. Other than the endpoint naming convention mentioned previously, and the fact that the entity file used by the simple file-based entity search is 100% compatible with the non-annotation based Comprehend model, everything else is as per the documentation.

For the sake of consistency, we recommend that you upload the files required to create a Custom Entity Model to the standard S3 bucket defined by the *SupportFilesBucketName* configuration parameter. If you need further assistance in creating a custom model then please contact AWS for assistance.

User Management

Users are managed via Amazon Cognito within a *User Pool*. Login to the AWS Console and navigate to the Cognito service, then select the **Manage User Pools** button. This will show a screen with all of the configured user pools in your account - one of these is for this application, but you may well have others in your account. Select the relevant user pool to go to its configuration screen.

In this section we will just discuss password policies and how to manage users, but Cognito offers many more powerful user settings, such as using MFA tokens, or linking to external identity providers. Please refer to the Amazon Cognito documentation, or contact your AWS Account Team for further information.

Setting Password Policies

Within the *General settings* area select the **Policies** option to show this screen:

The screenshot shows the AWS IAM console 'What password strength do you want to require?' settings page. On the left is a navigation menu with categories: General settings (Users and groups, Attributes, Policies, MFA and verifications, Advanced security, Message customizations, Tags, Devices, App clients, Triggers, Analytics), App integration (App client settings, Domain name, UI customization, Resource servers), and Federation (Identity providers, Attribute mapping). The 'Policies' option is selected. The main content area has the title 'What password strength do you want to require?'. It includes a 'Minimum length' input field with the value '8'. Below this are four unchecked checkboxes: 'Require numbers', 'Require special character', 'Require uppercase letters', and 'Require lowercase letters'. A blue-bordered box contains a recommendation: 'Amazon recommends requiring passwords with at least 8 characters, lowercase, uppercase, and numbers for greater security.' The next section is titled 'Do you want to allow users to sign themselves up?'. It explains that administrators can choose to only allow administrators to create users or allow users to sign themselves up, with a 'Learn more' link. Two radio buttons are present: 'Only allow administrators to create users' (which is selected) and 'Allow users to sign themselves up'. The final section is titled 'How quickly should temporary passwords set by administrators expire if not used?'. It explains that administrators can choose how long a temporary password expires if not used, including accounts created by administrators. A 'Days to expire' input field has the value '7'.

Here you are able to change many settings, such as password length and strength parameters. Once you have made the changes click the **Save changes** button on the bottom-right on the panel.

User Management

Click on the **Users and groups** option to bring up the user management screen, which will initially show the users currently defined in the application. Click on an existing user to edit them, but select **Create user** to bring up the following new user dialog:

Create user

Username (Required)

☒ Send an invitation to this new user?

☐ SMS (default) ☐ Email

Temporary password

Phone Number

☒ Mark phone number as verified?

Email

☒ Mark email as verified?

Create user

Enter the details that you need for the new user - as a minimum, you should give a temporary password and an email address, sending the invitation via email. The user will receive their login and temporary password, but you will need to inform them of the login URL of the application - please see the *Installation Process* section for information on how to retrieve this.

Installation Process

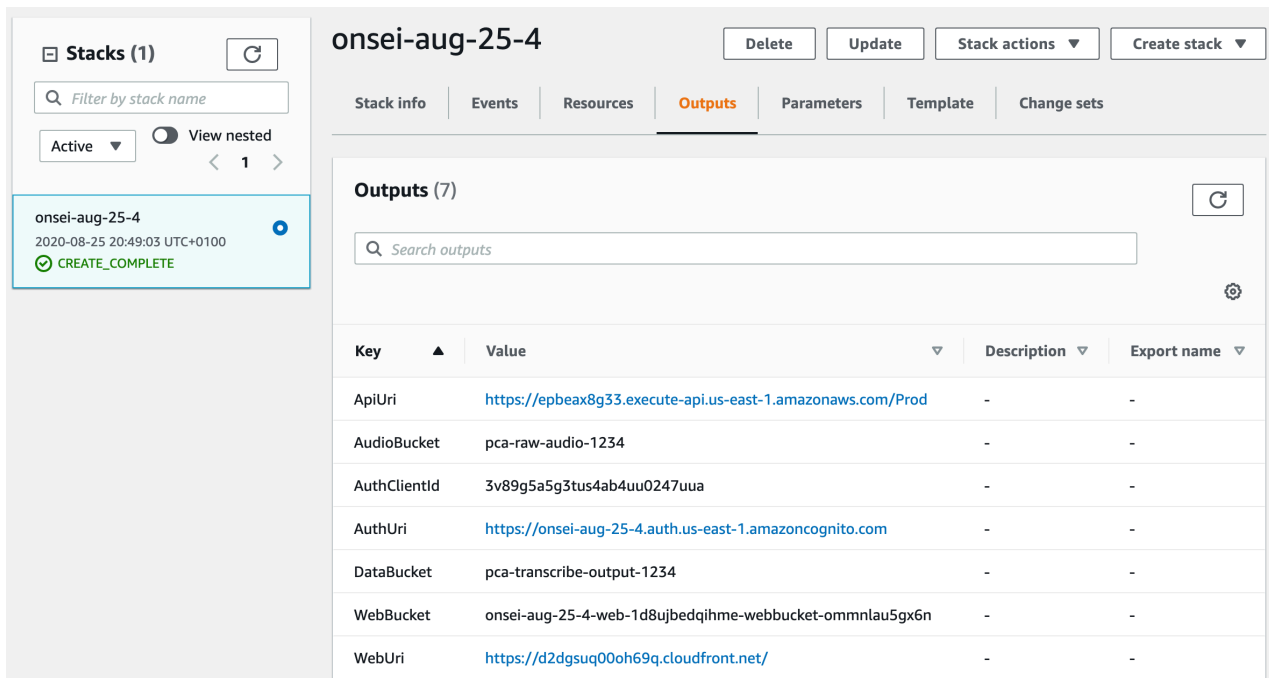
Application Installation

TBD - this will be CloudFormation, possibly with one or two manual steps to prepare some installation parameters. One CloudFormation will start the process to install the back-end components, allowing the user to configure many of the configuration parameters, and that process will then deploy a separate set of CloudFormation templates to install the user interface components.

Note: prior to the full application deployment via CloudFormation there was a single template for the User Interface - this will have the name **onsei-** followed by the date.

Application Login URL

In the AWS Console, navigate to the AWS CloudFormation service, and on the left-hand menu select **Stacks**. This will show all of the stacks currently deployed in the account, which will have the stack names assigned as per the previous section. Find the one used for the user interface and select it.



The screenshot shows the AWS CloudFormation console interface. On the left, a sidebar lists stacks, with 'onsei-aug-25-4' selected and marked as 'CREATE_COMPLETE'. The main panel displays the 'Outputs' tab for this stack, showing a table of 7 output parameters. The 'WebUri' output is highlighted, showing a CloudFront URL.

Key	Value	Description	Export name
ApiUri	https://epbeax8g33.execute-api.us-east-1.amazonaws.com/Prod	-	-
AudioBucket	pca-raw-audio-1234	-	-
AuthClientId	3v89g5a5g3tus4ab4uu0247uua	-	-
AuthUri	https://onsei-aug-25-4.auth.us-east-1.amazonaws.com	-	-
DataBucket	pca-transcribe-output-1234	-	-
WebBucket	onsei-aug-25-4-web-1d8ujbedqihme-webbucket-ommlau5gx6n	-	-
WebUri	https://d2dgsuq00oh69q.cloudfront.net/	-	-

Click on the *Outputs* tab at the top - you will see a list of output parameters, one of which is **WebUri**. Copy this URL, paste it into the address bar of your browser and it will take you to the application login page.

Customising the URL

Amazon Route 53 is a highly available and scalable Domain Name System (DNS) web service. You can use Route 53 to perform domain registration and DNS routing. You can use the Amazon domain registrar, or update your existing DNS records to point a sub-domain to your AWS account. You can then map that to point to the Amazon Cloudfront URL in the previous section.

This will allow you to have a custom URL for the login to the application.

Amazon QuickSight

NOTE: The QuickSight developer is currently writing the piece on how to setup the views and schemas that we provide, and that will be inserted here when available.

Amazon SSM - ParameterStore Settings

The complete list of configuration parameters within the application are listed in the table below. Whilst these can largely be changed by the customer, some of them are to do with informing the application about AWS language coverage and should only be edited if and when AWS extends language support:

- *ComprehendLanguages*
- *ComprehendRedactionLanguages*

During installtion, triggers are setup against two locations within S3 buckets - hence, changing any of the following configuration settings post-install will cause will cause either the file ingest process to cease or the updating of the UI with processed data to cease. These settings are marked in bold in the table, but are the following:

- *InputBucketName*
- *OutputBucketName*
- *InputBucketRawAudio*
- *OutputBucketParsedResults*
- *StepFunctionName*

Every other setting can be editing within the AWS Console, but if you are unsure as to the potential impact then please contact AWS for clarification.

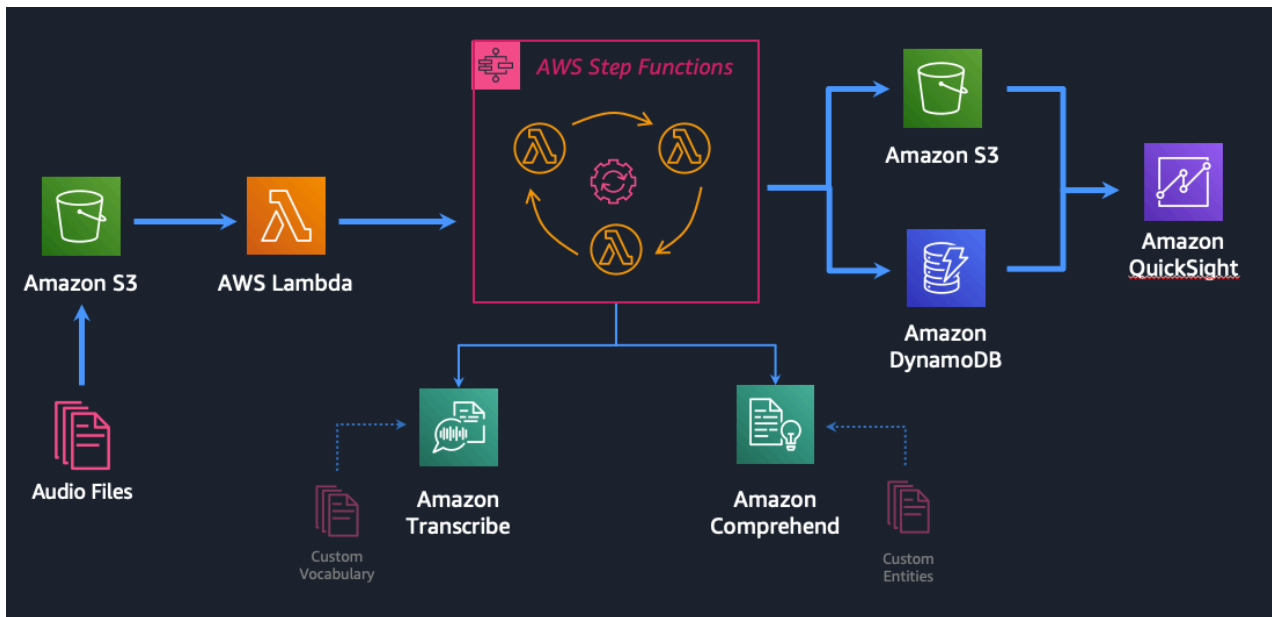
Key	Value	Description
BulkUploadBucket	pca-bulk-upload	Bucket where files can be dropped, and a secondary Step Function can be manually enabled to drip feed them into the system
BulkUploadMaxDripRate	50	Maximum number of files that the bulk uploader will move to the PCA source bucket in one pass
BulkUploadMaxTranscribeJobs	250	Number of concurrent Transcribe jobs (executing or queuing) where bulk upload will pause
ComprehendLanguages	en es fr de it pt ar hi ja ko zh zh-TW	Languages supported by Comprehend's standard calls, separated by " "
ContentRedactionLanguages	en-US	Languages supported by Transcribe's Content Redaction feature, separated by "
ConversationLocation	America/Toronto	Name of the timezone location for the call source - this is the TZ database name from https://en.wikipedia.org/wiki/List_of_tz_database_time_zones
EntityRecognizerEndpoint	undefined	Name of the built custom entity recognizer for Amazon Comprehend (not including language suffix, e.g. -en). If one cannot be found then simple entity string matching is attempted instead
EntityStringMap	simple-entity-list.csv	Basename of a CSV file containing item/Entity maps for when we don't have data for Comprehend Custom Entities (not including language suffix, e.g. -en)
EntityThreshold	0.5	Confidence threshold where we accept the custom entity detection result

InputBucketAudioPlayback	mp3	Folder that holds the audio to playback in the browser
InputBucketFailedTranscriptions	failedAudio	Folder that holds audio files that for some reason failed transcription
InputBucketName	pca-raw-audio-1234	Bucket where where audio files are delivered
InputBucketRawAudio	nci	Folder that holds the raw call audio
MaxSpeakers	2	Maximum number of speakers that are expected on a call
MinSentimentNegative	0.4	Minimum sentiment level required to declare a phrase as having negative sentiment
MinSentimentPositive	0.4	Minimum sentiment level required to declare a phrase as having positive sentiment
OutputBucketName	pca-transcribe-output-1234	Bucket where Transcribe output files are delivered
OutputBucketParsedResults	parsedFiles	Key within the output S3 bucket where parsed results are written to
SpeakerNames	Agent Caller	Default tags used for speaker names, separated by a
SpeakerSeparationType	speaker	Separation mode for speakers, either explicitly <code>speaker</code> or <code>channel</code> , or <code>auto</code> where audio stereo=>Channel and mono=>Speaker
StepFunctionName	PostCallAnalyticsWorkflow	Name of Step Functions workflow that orchestrates this process
SupportFilesBucketName	pca-custom-source-files	Bucket that hold supporting files, such as the file-based entity recognition mapping files
TranscribeAlternateLanguage	fr-CA	Allows files delivered from a non-standard bucket to be based upon this language
TranscribeLanguages [1]	en-US fr-FR	Language to be used for Transcription - multiple entries separated by will trigger Language Detection using those languages; if that fails for any reason then the first language in this list is used for transcription
VocabularyName	nci-vocab	Name of the custom vocabulary to use for Transcribe (not including language suffix, e.g. -en-US)

[1] The *TranscribeLanguages* setting should be **en-US | fr-CA**, but due to an issue in the *Language ID* feature we are finding that some heavily-accented French speakers on English calls are causing some English calls to be declared **fr-CA**. By changing the *TranscribeLanguages* setting to **en-US | fr-FR** we are correctly identifying the English/French language, but then the StepFunction workflow will transcribe the whole audio file in **fr-CA** if it has been identified as **fr-FR**. This may be due to the beta-nature of the *Language ID* feature, so this may be resolved in the future, but for now this is a workaround.

Application Architecture

The server-side architecture is as follows:



Primary Workflow

The main work of the application is carried out by the AWS Step Functions workflow, which orchestrates the entire process - it is kicked off by the delivery of a new audio file in the *InputBucketName* S3 bucket in the *InputBucketRawAudio* folder. The Step Function will perform the following processes, handling error messages, retries and listening for internal AWS events such as when a Transcribe job has completed:

- Is Language Detection needed?
 - Create a 30-second clip of the original audio file and sent to Transcribe
 - Pick out the strongest language candidate and use that for full transcription
- Is Language Detection not needed?
 - Read the required language from the app configuration
- Transcribe the full audio file in the required language, using PII redaction if supported by the language, and any custom vocabulary that has been defined for that language
- Parse the Transcribe output, adding sentiment analysis and entity detection
- Write all results out to S3 - that triggers updates to DynamoDB, and the result is visible in the UI

Failure Handling

The workflow can fail in the following expected ways:

1. Language detection is requested on a file < 30 seconds long, and a dominant language cannot be detected
2. Transcribe itself suffers an internal error - we retry 2 more times, but if it's a fundamental problem with the audio then we will not be able to process it.

In these scenarios, the Step Function will move the original file to the S3 bucket defined in *InputBucketName* into the folder defined by *InputBucketFailedTranscriptions*.

Lambda Functions

Audio Processing Non-Step Function Lambdas

Lambda Function	Description
pca-aws-file-drop-trigger	This responds to the delivery of an audio file to the ingestion S3 bucket. Assuming that it is of a supported format then the ingestion Step Function workflow is initiated for that audio file.
pca-transcribe-eventbridge	This responds to the completion of a Transcribe processing job, either for the full transcription or for language detection. It will identify the Step Function workflow execution that requested this transcription job and signal it to continue processing. Note, if the transcription process ended in an error then this function will indicate if a retry should be made or if the workflow should be failed.

Audio Processing Step Function Lambdas

Lambda Function	Description
pca-aws-sf-language-detection	Extracts a 30-second clip from the source audio and sends that to Transcribe with the Language ID parameter enabled
pca-aws-sf-wait-for-transcribe-notification	When any Transcribe job begins this records tracking codes of the job-id and the current Step Function workflow, so that when it completes we are able to continue from this point without performing any polling
pca-aws-sf-get-detected-language	Extracts the dominant language code identified by Language ID to be used in the main transcription
pca-aws-sf-start-transcribe-job	Performs full transcription of the source audio file, enabling features such as custom vocabularies, PII redaction and others depending upon the application configuration
pca-aws-sf-process-turn-by-turn	Takes the output from Transcribe and processes it into a turn-by-turn conversation, as well as embedding sentiment and entity detection from Comprehend
pca-aws-sf-transcribe-failed	If the Step Function workflow fails at any time this this will clean up any created resources before terminating the workflow

Bulk Upload Step Function Lambdas

Lambda Function	Description
pca-aws-bulk-files-count	Counts the number of files in the Bulk Upload bucket, returning the minimum of that or of the configured file drip-rate
pca-aws-bulk-queue-space	Looks at the number of both IN_PROGRESS and QUEUED Transcribe jobs, and calculates the space left for bulk-load jobs based upon the configured maximum thresholds
pca-aws-bulk-move-files	Moves as many files as possible from the bulk-upload bucket to the standard ingest location based upon previous two Lambda functions's calculations

Installing FFMPEG

Due to licensing limitations, AWS is not able to distribute the open-source **ffmpeg** library with this application, but it is key for the following functions:

- Creating the 30-second audio clip for Transcribe Language ID
- Creating the MP3 playback file if the original audio is an 8Khz WAV file

The application has been built to handle errors if this library does not exist, but the Language ID and MP3 playback features will not function in that case. Hence, we have provided the following instructions for someone to follow to create a library within the Lambda environment that the other functions can use. It should be done after the rest of the application has been deployed.

Creating the Lambda Layer for FFMPEG

In a terminal session create a folder called **lambda_layer**, and in that folder create another one called **python** and navigate into it.

```
> mkdir layer
> cd layer
layer> mkdir bin
```

We now need to download a statically-linked version of the **ffmpeg** modules, which are built and maintained by John Van Sickle. At the time of writing the latest release is version 4.3.1.

```
layer> curl https://johnvansickle.com/ffmpeg/releases/ffmpeg-release-amd64-
static.tar.xz | tar x
layer> mv ffmpeg*/ffmpeg ffmpeg*/ffprobe bin
layer> zip -vr ffmpeg.zip bin -X
```

This results in a package file called **ffmpeg.zip**, which we need to install into a Lambda layer so that other Lambda functions can call it. In the AWS console, go to the S3 service and upload this file to the bucket defined in the configuration parameter *SupportFilesBucketName*.

You should now go to the Lambda service, go to *Layers* and create a new layer. Enter the following parameters:

- **Name** - ffmpeg
- **Description** - FFMPEG for AWS Lambda Layer
- **Upload** - select "Upload a file from Amazon S3" and enter the ARN of your **ffmpeg.zip** file in S3
- **Compatible Runtimes** - select whichever languages you intend to call this from in your Lambda functions. This application only uses Python, so select Python3.8

Hit the *Create* button and the layer should be created.

Configuring the Lambda Functions to use the FFMPEG Lambda Layer

You will need to link this Lambda layer for ffmpeg to two of the Lambda functions - follow the same process for both functions:

- pca-aws-sf-language-detection
- pca-aws-sf-process-turn-by-turn
- pca-aws-sf-start-transcribe-job

In the Lambda service in the AWS console select the function, and then select the *Layers* central button within the *Designer* section. Towards the bottom of the screen there will be a new section panel called **Layers** - click on the `Add a layer` button to bring up the following screen:

Add layer

Choose a layer [Info](#)

Choose from layers with a compatible runtime or specify the Amazon Resource Name (ARN) of a layer version.

☐ **AWS layers**
Choose a layer from a list of layers provided by AWS.

☒ **Custom layers**
Choose a layer from a list of layers created by your AWS account or organization.

☐ **Specify an ARN**
Specify a layer by providing the ARN.

Custom layers
Layers created by your AWS account or organization that are compatible with your function's runtime.

ffmpeg ▼

Version

1 ▲

Cancel Add

Select *Custom layers*, and then in the **AWS layers** drop-down select the **ffmpeg** layer, and then the latest version number of the layer. Click the `Add` button, and the ffmpeg layer will now be callable from the main Lambda function.

Ensure that this step is repeated on both Lambda functions listed above.