Table of Contents

Quick Start

Setup

Add an instance of `BhashiniManager` to your scene, and set it up with your ULCA User ID and API key, as detailed in the *Bhashini documentation*.

Pipelines

As per the *Bhashini documentation*:

> ULCA Pipeline is a set of tasks that any specific pipeline supports. For example, any specific pipeline (identified by unique pipeline ID) can support the following:
>
> - only ASR (Speech To Text)
> - only NMT (Translate)
> - only TTS
> - ASR + NMT
> - NMT + TTS
> - ASR + NMT + TTS
>
> Our R&D institutes can create pipelines using any of the available models on ULCA.

Basically, computation (STT, TTS, Translate) is done on a "pipeline". A "pipeline" is set to support a list of tasks, in a defined order, like:

- (input: audio) STT -> Translate (output: text)
- (input: text) Translate -> TTS (output: audio)

In the given examples:

- Case 1 (STT -> Translate): From the given audio clip, the STT model computes text, which is sent automatically to the translate model, and text is returned.
- Case 2 (Translate -> TTS): From the given text, the translate model computes text, which is sent automatically to the TTS model, and audio is returned.

You can have any combination of these tasks, or just individual ones. You can even have tasks like:

- STT -> Translate -> TTS!

Code

So, before we do any computation, we have to set up our pipelines:

```
using Uralstech.UBhashini;
using Uralstech.UBhashini.Data;
using Uralstech.UBhashini.Data.Pipeline;

// This example shows a pipeline configured for a set of tasks which will receive spoken English audio
// as input, transcribe and translate it to Hindi, and finally convert the text to spoken Hindi audio.

BhashiniPipelineResponse pipeline = await BhashiniManager.Instance.ConfigurePipeline(
    new BhashiniPipelineRequestTask(BhashiniTask.SpeechToText, "en"), // Here, "en" is the source language.
    new BhashiniPipelineRequestTask(BhashiniTask.Translation, "en", "hi"), // Here, "en" is still the source language, but "hi" is the target language.
    new BhashiniPipelineRequestTask(BhashiniTask.TextToSpeech, "hi") // Here, the source language is "hi".
);
```

> The Bhashini API follows the *ISO-639* standard for language codes.
>
> The UBhashini throws `BhashiniAudioIOException` and `BhashiniRequestException` errors when requests fail. Be sure to add try-catch blocks in your code!

Now, we store the computation inference data in variables:

```
BhashiniPipelineInferenceEndpoint endpoint = pipeline.InferenceEndpoint;

BhashiniPipelineTaskConfiguration sttTaskConfig =
pipeline.PipelineConfigurations[0].Configurations[0];
BhashiniPipelineTaskConfiguration translateTaskConfig =
pipeline.PipelineConfigurations[1].Configurations[0];
BhashiniPipelineTaskConfiguration ttsTaskConfig =
pipeline.PipelineConfigurations[2].Configurations[0];
```

Here, as we specified the expected source and target languages for each task in the pipeline, it is very likely that the `Configurations` array in `PipelineConfigurations` will only contain one `BhashiniPipelineTaskConfiguration` object. This may not always be the case, so it is recommended to check the array of configurations for the desired model(s). The order of `PipelineConfigurations` is based on the order of the tasks array in the input for `ConfigurePipeline`.

You can also use the following shortcuts to get the task configs.

```
BhashiniPipelineTaskConfiguration sttTaskConfig = pipeline.SpeechToTextConfiguration.First;
BhashiniPipelineTaskConfiguration translateTaskConfig =
pipeline.TranslateConfiguration.First;
BhashiniPipelineTaskConfiguration ttsTaskConfig = pipeline.TextToSpeechConfiguration.First;
```

`SpeechToTextConfiguration`, `TranslateConfiguration` and `TextToSpeechConfiguration` will get the first config matching the task type in the response and `First` gets the first available `BhashiniPipelineTaskConfiguration` from its `Configurations` array.

Computation

Now that we have the inference data and pipelines configured, we can go straight into computation.

Code

```
using Uralstech.UBhashini.Data.Compute;

// The below code records a 10 seconds audio clip from the device microphone.

int sampleRate = AudioSettings.GetConfiguration().sampleRate;
AudioClip audioInput = Microphone.Start(string.Empty, false, 10, sampleRate);

while (Microphone.IsRecording(string.Empty))
    await Task.Yield();

if (!TryGetComponent(out AudioSource audioSource))
    audioSource = gameObject.AddComponent<AudioSource>();

// Now, we send the clip to Bhashini.

BhashiniComputeResponse computedResults = await
BhashiniManager.Instance.ComputeOnPipeline(endpoint,
    new BhashiniInputData(audioInput),
    sttTaskConfig.ToSpeechToTextTask(sampleRate: sampleRate),
    translateTaskConfig.ToTranslateTask(),
    ttsTaskConfig.ToTextToSpeechTask()
);

audioSource.PlayOneShot(await computedResults.GetTextToSpeechResult());
```

`BhashiniInputData` has two constructors:

- `BhashiniInputData(string text = null, string audio = null)`

    - Here, `text` is input for translation and TTS requests, and `audio` is base64-encoded audio for STT requests. Only provide one.
- `BhashiniInputData(AudioClip audio, BhashiniAudioFormat audioFormat = BhashiniAudioFormat.Wav)`
    - This constructor allows you to directly provide an `AudioClip` as input, but requires the `Utilities.Encoder.Wav` or `Utilities.Audio` packages, based on the chosen encoding.

Also, `ToSpeechToTextTask` takes an optional `sampleRate` argument. By default, it is 44100, but make sure it matches with your audio data.

`BhashiniComputeResponse` contains three utility functions to help extract the actual text or audio response:

- `GetSpeechToTextResult`
- `GetTranslateResult` and
- `GetTextToSpeechResult`

You should call them based on the last task in the pipeline's task list. If your pipeline's last task is STT, use `GetSpeechToTextResult`. If the last task is translate, use `GetTranslateResult`. If it's TTS, use `GetTextToSpeechResult`.

`ComputeOnPipeline` and `GetTextToSpeechResult` will throw `BhashiniAudioIOException` errors if they encounter an unsupported format.

And that's it! You've learnt how to use the Bhashini API in Unity!