**WebSocket Documentation for Dhruva Streaming Client**

**Introduction**

This documentation provides a comprehensive guide on using the Dhruva Streaming Client via WebSocket. The Dhruva Streaming Client allows real-time streaming of audio data to a server for automatic speech recognition (ASR), Translation (NMT) and text to speech (TTS). This document covers the usage of the client, its initialization, configuration, and usage in various programming languages.

**WebSocket Connection**

The Dhruva Streaming Client utilizes WebSocket for real-time communication with the server. WebSocket is a protocol that provides full-duplex communication channels over a single TCP connection.

**WebSocket URL**

The WebSocket URL for connecting to the Dhruva service is:

wss://dhruva-api.bhashini.gov.in

The sample code can be executed with the following commands using Python after extracting the zip file to a directory:

pip install -r requirements.txt

python asr\_recorder.py

Once this is done, you can speak over the mic and have streaming responses obtained.

**Authentication**

Authentication is performed using an API key. The API key should be provided during initialization of the client.

**Initialization**

The Dhruva Streaming Client can be initialized with the following parameters:

- `socket\_url`: The WebSocket URL for connecting to the Dhruva service.

- `api\_key`: The API key for authentication with the server.

- `task\_sequence`: A list of tasks to be performed sequentially, each task containing its configuration.

- `response\_callback`: A callback function to handle the responses from the server.

- `auto\_start`: Optional. Boolean indicating whether the client should automatically start streaming audio upon connection.

- `responseTaskSequenceDepth`: 1. This value represents how many tasks in sequence are to be performed. It is set to 1 if only ASR is to be performed. 2 if ASR + Translation, 3 if ASR + Translation + TTS is to be performed

**Task Sequence**

The `task\_sequence` parameter is a list of tasks to be performed sequentially. Each task in the sequence consists of the following:

- `taskType`: The type of task to be performed (e.g., ASR, translation).

- `config`: Configuration parameters specific to the task (e.g., preprocessing, service ID, language settings).

The configuration for each of this task type is specified here: <https://bhashini.gitbook.io/bhashini-apis/>

**Response Callback**

The `response\_callback` function is invoked whenever the server sends a response. It receives two parameters:

- `transcript\_history`: The transcript accumulated so far.

- `current\_transcript`: The current transcript received from the server.

**Usage**

1. **Initialization**: Initialize the Dhruva Streaming Client with the required parameters.

2. **Start Streaming**: If `auto\_start` is set to `True` during initialization, the client will automatically start streaming audio upon connection. Otherwise, call the `start\_transcribing\_from\_mic` method to begin streaming.

3. **Handle Responses**: Implement the response callback function (`response\_callback`) to handle the transcripts received from the server.

4. **Stop Streaming**: To stop streaming audio and disconnect from the server, call the `stop` method.

5. **Error Handling**: Implement error handling for connection failures and disconnections.

**Notes on Intermediate Results**

- `isIntermediateResult`: Indicates whether the received transcript is an intermediate result or the final result.

- True: The transcript is an intermediate result, typically containing partial speech recognition outputs.

- False: The transcript is the final result, usually the complete speech recognition output.

**Conclusion**

This documentation provides a detailed guide on using the Dhruva Speech Streaming Client via WebSocket. By following the provided instructions, developers can effectively integrate real-time speech recognition and related tasks into their applications across different programming languages.