Introduction
The Zoom Media Speech-to-text real-time API provides customers with a WebSocket interface in order to process their media streams in a real-time manner. The protocol for this real-time session is described in this document and should help developers with implementing the solution in their products.

Please be aware that a WebSocket connection is mostly asynchronous and for best results relies on either coroutines or threads. This implicitly implies that development of this interface is harder than our batch API.

Example code in multiple languages can be found in our GitHub repository at https://github.com/zoom-media/realtime-examples.

Before getting started
Make sure you have a real-time token ready. This token type can be created at https://api.zoommedia.ai/tokens.

Start connections
You can reach our API service by using the WebSocket Secure (WSS) protocol. The endpoint is:

wss://api.zoommedia.ai/realtime?language=nl-nl

Request Headers

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x-zoom-s2t-key</td>
<td>YOUR API TOKEN</td>
<td>API key of type real-time needed for authorization</td>
</tr>
</tbody>
</table>

Path parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>language</td>
<td>xx-yy</td>
<td>Language identifier for speech to text session</td>
</tr>
</tbody>
</table>

Messages
All control messages are exchanges in JSON format; the following control messages are supported.

Client messages

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>&quot;action&quot;: &quot;start&quot;</td>
</tr>
</tbody>
</table>
Connection with the Speech to Text engine is stopped. The service will return the remaining blob that has to be processed. Once the connection with the Speech to Text engine is closed you will receive a "state: stopped" message. It is not possible to start a new session once you've stopped it.

Server responses
Server responses will be sent upon successfully executing an action or to provide the client with a status update, the following messages are defined.

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>{&quot;state&quot;: &quot;listening&quot;} Connection with the Speech to Text engine is initialised. If the connection is successful and the websocket is ready to process data, you will receive back a &quot;state: listening&quot; message. After receiving this message, you are able to send audio.</td>
</tr>
<tr>
<td>2</td>
<td>{&quot;state&quot;: &quot;stop&quot;} Connection with the Speech to Text engine is stopped. The service will return the remaining blob that has to be processed. Once the connection with the Speech to Text engine is closed you will receive a &quot;state: stopped&quot; message. It is not possible to start a new session once you've stopped it.</td>
</tr>
<tr>
<td>3</td>
<td>{&quot;error&quot;: &quot;Session not started&quot;} The client sent binary data (audio stream) but did not start a session yet. Data will not be processed.</td>
</tr>
<tr>
<td>4</td>
<td>{&quot;error&quot;: &quot;backend already listening&quot;} Client tried to start a new session while there is already a backend session running</td>
</tr>
<tr>
<td>5</td>
<td>{&quot;error&quot;: &quot;restarting of sessions is not supported&quot;} Client tried to start a new session after finishing a previous one. This is not supported, disconnect after finishing the session.</td>
</tr>
<tr>
<td>6</td>
<td>{&quot;error&quot;: &quot;unable to start backend&quot;} The real-time engine was unable to connect to a backend system. This could be due to multiple reasons not defined further. Contact support if the problem persists.</td>
</tr>
<tr>
<td>7</td>
<td>{&quot;error&quot;: &quot;engine_not_responding&quot;} This error is raised after a request for a session in the backend system is not responding. Contact support if this problem persists.</td>
</tr>
</tbody>
</table>
The real-time engine will throw this message to let you know it will shut down at a specific time in the future. This message is sent one hour before shutting down and will give you enough time (one hour by default) to either finish within time or stop gracefully and restart the engine.

Encoding your audio stream

All audio streams provided should be encoded as PCM WAVE, 16khz mono. The following ffmpeg command can be used to convert your input before sending it to our API:

```bash
ffmpeg -y -i "${input_file}" \
    -ac 1 -acodec pcm_s16le -ar 16000 \ 
    -f wav "${output_file}"
```

This command can also be used in a pipe in which `${input_file}` and `${output_file}` can be set to "-" for respectively stdin and stdout.

It may be possible that other flags are suitable for this type of configuration. But adding/enabling those is out of scope of this document. If you need any support with encoding, please contact us as support@zoommedia.ai.

Sending audio and response times

Quality of results and response times relate to each other, in order to receive a timely response, it is important to send small chunks of data. But for the quality of your results it is better to send larger chunks. This all relates to the contextual awareness of the system. Zoom Media advises to send in chunks between 8KB and 64KB in size. Smaller chunks do work but may result in lesser quality, larger chunks work too, but result in larger response times. With the conversion used above (16Khz PCM Wave) 1 second of audio corresponds to 32KB of data.

Data models

Partial result
The first result sent after receiving a binary blob is a partial. The partial contains the spoken text currently detected and may be subject to change.

```
{
    "partial": "Je hoort natuurlijk zeker"
}
```
Full result
When the real-time engine is confident about a full result it will send it in the following format:

```
{  
    "result": [  
        [ "Je", 12046, 12286, 1 ],  
        [ "hoort", 12286, 12526, 1 ],  
        [ "natuurlijk", 12526, 12796, 1 ],  
        [ "zeker", 12796, 13096, 1 ],  
        [ "in", 13096, 13156, 1 ],  
        [ "'s-gravenhage", 13156, 13666, 1 ],  
        [ "verhalen", 13666, 14055, 0.999713 ]],  
    "text": "Je hoort natuurlijk zeker in 's-gravenhage verhalen"
}
```

The result is an object with two keys, the result and text. The result consists of the words spoken with metadata. The metadata is formatted as following:

```
[ "word", time_start, time_stop, confidence ]
```

The time_start and time_stop are timepoints in the audio processed. This does not correspond with any position in the session itself. This makes it possible to send audio up to twice the real-time speed. The confidence is a float corresponding to a percentage of confidence the result is accurate.

The text key contains a full representation of all words detected.