Working with the Stream and Voicebot Applet

Modified on: Mon, 8 Jul, 2024 at 6:37 PM

Types of Streams

There are two types of Streams - Unidirectional and Bidirectional

Unidirectional Streams

Unidirectional Streams allow you to Stream live calls over WebSocket in a single direction, from Exotel to the WebSocket Endpoint. Some of the use cases for this are live transcription, realtime monitoring of agents, realtime coaching etc.

Bidirectional Streams

WebSocketBidirectional Streams allow two-way flow of voice data over a websocket. Exotel would send the voice data of the caller to a websocket endpoint. The endpoint can return back voice data back on the websocket and Exotel would play it out to the caller. The primary use case for this is to enable building intelligent conversational bots that will help you optimize your workforce WebSocket

Chunk size window for Bidirectional streaming use case:

Minimum chunk size: 3.2k [100ms data]
Maximum chunk size: 100k
Chunk size should always be in multiple of 320 bytes.
1. If the size is less than the minimum size, we may face audio issues due to network jitters.
2. If the size is greater than 100k, it might result in timeouts
3. if the size X [for ex 4096] which is not in multiple of 320 Bytes: In this case, the last packet will be of lesser size than 320 Bytes, & platform will wait for 20ms before sending next chunk. i.e. sending less amount of data than wait time, then this might result into audio gaps in between.

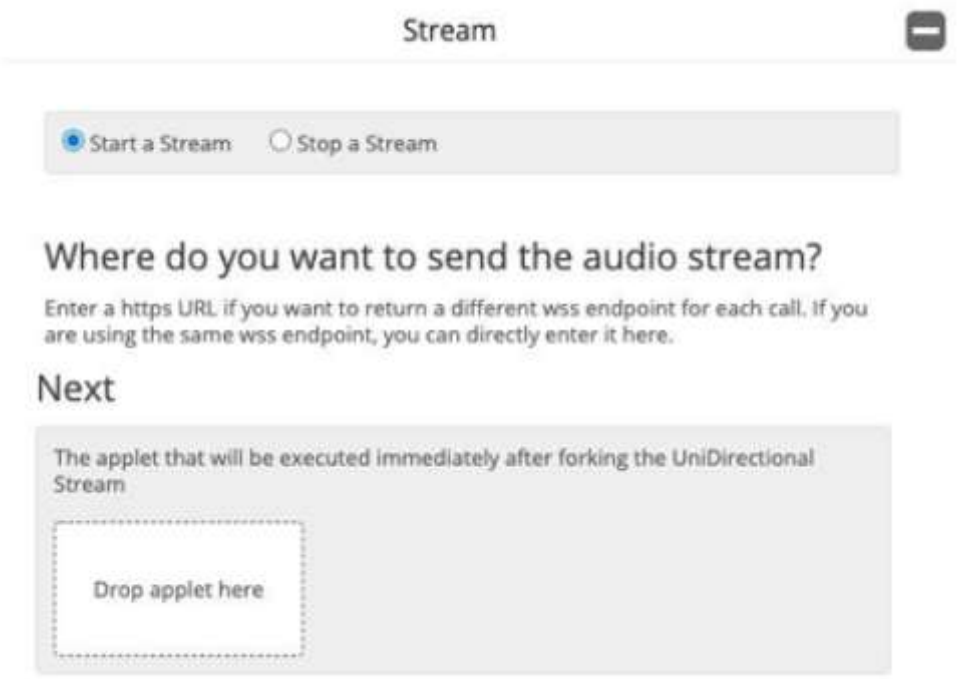
Enabling And Disabling Streams

Unidirectional and Bidirectional Streams can be enabled for a call flow using the “Stream” and “Voicebot” Applet, respectively when creating Custom Apps in the App Bazaar.

This Applet might not be available by default for all accounts. If you are not able to see them in the list of Voice Applets, drop a mail to hello@exotel.in or talk to your Account Manager

Configuring a Unidirectional Stream-Stream applet

You can enable unidirectional streaming on a call flow using the Streaming Applet.



The applet takes 3 parameters :

1. Action - You can either start a new stream or stop a stream that you started earlier in the same call flow. You will use the Stop action if you have started a unidirectional Stream earlier in the same flow. When you choose stop, that is the only input you need to configure

2. URL - This is the URL to which Exotel will stream the voice media. You can either specify a wss endpoint or a https endpoint. If you specify a http/https endpoint, Exotel expects the https endpoint to return a wss url in its response. This is to allow

a. Dynamic endpoints for the same call flow
b. Have dynamic custom parameters that can be passed to the websocket endpoint to handle any application specific customization

When you specify a https endpoint, it must return a json with the key “url”

```
{
        "url" : "wss://streamhandler.yourdomain.com"
}
```

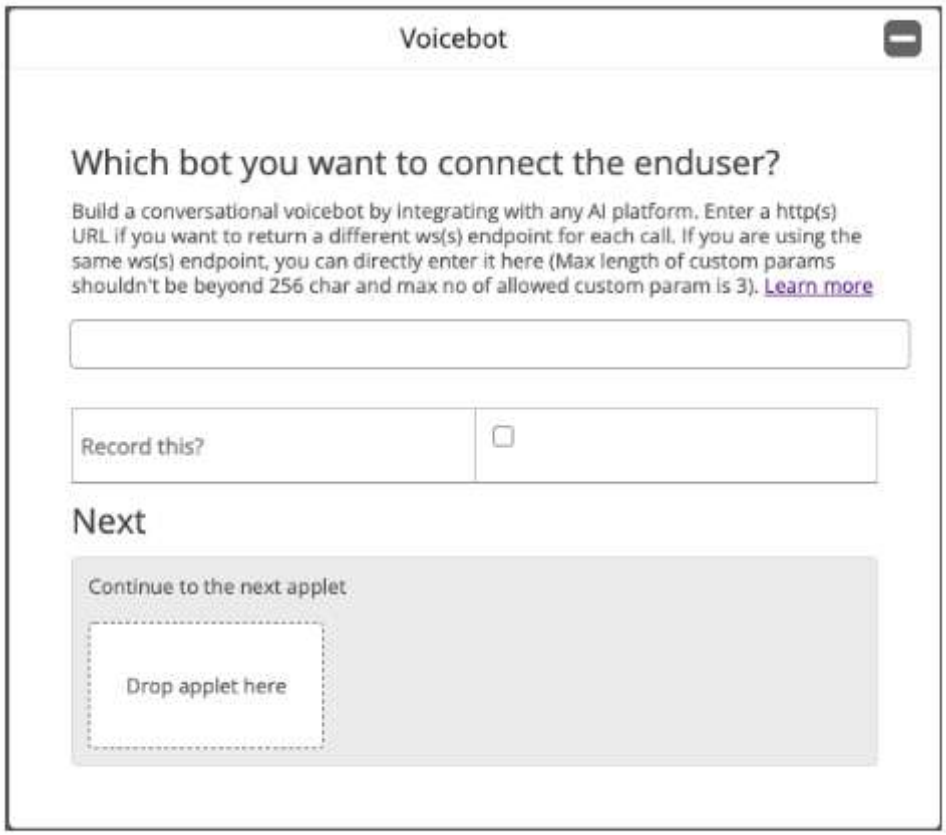
On receiving this, Exotel will initiate a connection to wss://streamhandler.yourdomain.com

3. Next Applet

In the case if Unidirectional Stream, the Stream would be created and the call flow proceeds to the nex d

Configuring a Bidirectional Stream- Voicebot Applet

You can enable bidirectional streaming on a call flow using the Voicebot Applet.



The applet takes 4 parameters:

1. URL - This is the URL to which Exotel will stream the voice media. You can either specify a wss endpoint or a https endpoint. If you specify a http/https endpoint, Exotel expects the https endpoint to return a wss url in its response. This is to allow

a. Dynamic endpoints for the same call flow
b. Have dynamic custom parameters that can be passed to the websocket endpoint to handle any application specific customization.

There are two ways to put this:
a. Static method : ws(s) endpoint can be entered here but it will remain the same for every call that you going to make using this flow. eg: ws://127.0.0.1:5001/media
b. Dynamic method :We can enter http(s) url which can return different ws(s) endpoints based on use-case.
eg: **https://yourdomain.com** (https://yourdomain.com) this URL must return a ws(s) endpoint

2. Custom parameters(Optional) - Custom params along with the endpoint. There are some validation follow while providing custom params.

a. Maximum number of custom params that are allowed is 3.
b. Format of these params will like :

ws://127.0.0.1:5001/media?param1=value1&param2=value2&param3=value3 (In Dynamic case, http(s) should return ws(s) URL in above format)
c. Total length of the params(bold part in above url) shouldn't be more than 256 characters.

3. Record - The checkbox gives an option to record the conversation and generate a recording URL available in passthru applet after voicebot applet.

4. Next Applet - In the case of Bi-Directional Stream. the stream can end if the call is disconnected or the websocket is closed or the stream is explicitly stopped by the client. In the case of Bi-Directional Stream you do not need to add a explicit "Stop" Stream applet since the stream is automatically closed before executing the next Applet. The call flow proceeds to the next applet configured.

Video WalkThrough

You can find a quick walkthrough of a sample flow **here** (https://www.loom.com/share/ae049b2f8c654c3cb983fbfaf58dadbe) .

Protocol

Communication between Exotel and customer endpoint happens over websocket connection.

Websocket messages - From Exotel

Each message in the websocket will be sent/received as a JSON string. Following are the types of messages that are sent:

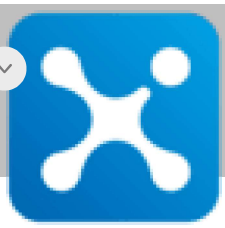
- Connected
- Start
- Media
- DTMF
- Stop
- Mark (Only in Bidirectional)
- Clear

Connected message:

After websocket connection is established, this message will be sent.

```
{
        "event" : "connected",

}
```



Start message:

Start message will contain information about the stream parameters. It will be sent only once, right after the connected message. The custom parameters are picked from the URL configured in the Stream Applet. If you had mentioned the URL as
wss://yourstream.service.com?queuename=premium&product=radio
queuename and product would be passed in as keys with premium and radio as values.

```
{
        "event" : "start",
        "sequence_number" : 1,
        "stream_sid" : "<stream sid>",
        "start" : {
                "stream_sid" : "<>",
                "call_sid" : "",
                "account_sid" : "",
                "from" : "",
                "to" : "",
                "custom_parameters" : {
                        "Key1" : "value1",
                        "key2" : "value2"
                },

                "media_format" : {
                        "encoding" : "<>",
                        "sample_rate" : "<>",
                        “bit_rate” : “<>”
                }
        }
}
```

Media message:

This message encapsulates the audio packets.

```
{
        "event" : "media",
        "sequence_number" : 3,
        "stream_sid" : "<stream sid>",
        "media" : {
                "chunk" : 2,
```

```
            "timestamp" : "10",
            "payload" : "<>"
}
```

media.chunk : chunk of the message
media.timestamp : Timestamp in milliseconds from the start of the stream.

Media in the payloads are sent in raw/slin (16-bit, 8kHz, mono PCM (little-endian)) encoded in base64. The same is expected from the client in the case of bi-directional streams (In the case of a Voice Bot) to be played back to the human.

DTMF message:

DTMF message is sent when the digits are pressed by the user once the connection with websocket is established. This is supported only for bidirectional streaming in voicebot applet.

```
{
        "event" : "dtmf",
        "sequence_number" : 1,
        "stream_sid" : "<stream sid>",
        "dtmf" : {
              "duration" : "<duration in ms>”,
               "digit": <>,
                    }
}
```

Stop message:

Stop message is sent when the stream is stopped or the call has ended.

```
{
         "event" : "stop",
         "sequence_number" : 10,
         "stream_sid" : "<stream sid>",
         "stop" : {
              "call_sid" : “<>”,
"account_sid" : "<>",
              “reason” : “stopped or callended"
                    }
}
```

Mark Message:

Mark message is used only in bidirectional streaming to track media when it is completed.

```
{
        "event" : "mark",
        "sequence_number" : 15,
        "stream_sid" : "<stream sid>",
        "mark" : {
            "name" : "<label>”,
                }
}
```

Websocket messages - To Exotel

These messages will be used only in bidirectional streaming.

Mark Message:

Mark message is used only in bidirectional streaming to track media when it is completed. You can send a mark event message after sending a media message to request a notification when the audio that you have sent has been processed. You'll receive a mark event message with a matching name from Exotel when the audio is processed.

```
{
        "event" : "mark",
        "sequence_number" : 15,
        "stream_sid" : "<stream sid>",
        "mark" : {
            "name" : "<label>”,
                }
}
```

Media message:

This message encapsulates the audio packets.

```
{
         "event" : "media",
        "sequence_number" : 3,
        "stream_sid" : "<stream sid>",
        "media" : {
             "chunk" : 2,
            "timestamp" : "10",
```

```
            "payload" : "<>"
}
```

Clear message:

Clear message is used to clear the audio data that was sent before but not yet played. An example situation wherein it will be useful,

Developing human-like bots which can guess what he/she is going to say and send audio accordingly even before he/she completes it. When the guess goes wrong, we can clear that audio using a clear message.

```
{
        "event": "clear",
         "stream_sid": "<stream sid>",
}
```

Note: For Clear Event to work effectively, it is advisable to send media in smaller chunks. For instance, if two media messages of 5 seconds are sent to us, followed by Clear Event at the 3rd second, and the first message has already been picked up and played, the Clear event will only apply to the second media message. Therefore, sending media in smaller chunks can help prevent confusion and ensure that Clear Event works as intended.

Media Format

Media in the payloads are sent in raw/slin (16-bit, 8kHz, mono PCM (little-endian)) encoded in base64. The same is expected from the client in the case of bi directional streams to be played back to the caller.

Sample Code

https://github.com/exotel/voice-streaming (https://github.com/exotel/voice-streaming)

Simulator to make streaming calls with dummy bot

(https://github.com/exotel/voice-streaming)https://github.com/exotel/voice-streaming/commit/d4696f3cb13fb5d75ac46bed2aaaa2afababa10f#diff-217ed82f87b78b399e304b07a159b27dff327c0f83adf4a2fc30b03bcbf84b01 (https://github.com/exotel/voice-streaming/commit/d4696f3cb13fb5d75ac46bed2aaaa2afababa10f#diff-217ed82f87b78b399e304b07a159b27dff327c0f83adf4a2fc30b03bcbf84b01)

Limitations

1) When you start a UniDirectional Stream, the stream is forked immediately. In case you have a Connect applet post this, the audio stream even when Exotel dials out multiple agents would be relayed (ringing). The client will have to handle this to filter only relevant parts of the stream. This limitation will be fixed in future releases

2) The number of Custom Parameters that can be passed in the START message is 3
3) The stream is sent as a mono channel raw audio format and the client will have to handle speaker diarization

If you have any questions or concerns, please connect with us using the chat widget on your Exotel Dashboard or Whatsapp us on 08088919888.