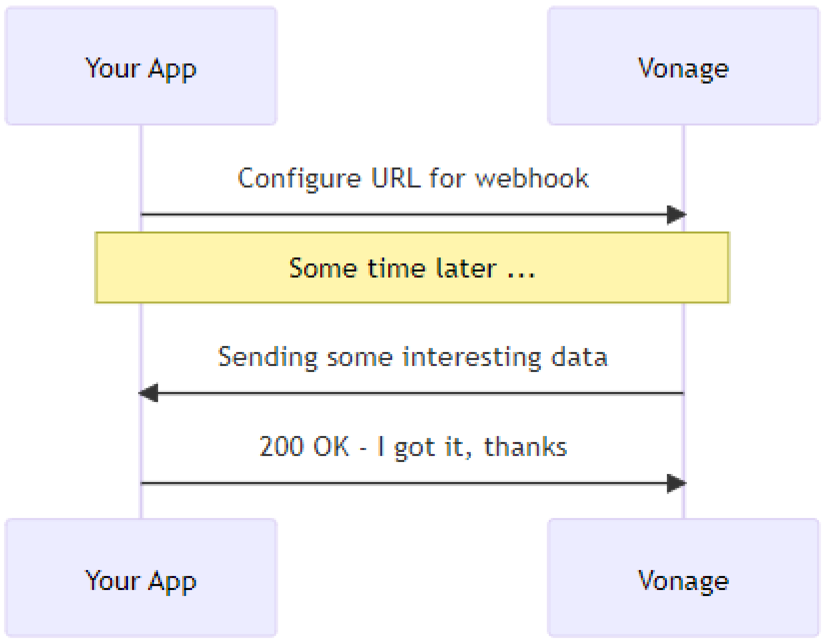
# Vonage Voice API

The Vonage Voice API allows you to connect people around the world and automate voice interactions that deliver a frictionless extension of your brand experience using AI technologies.

- Text to Speech with over 50+ languages
- Create IVRs and Voice Bots
- Speech to text and WebSockets
- Embed calls in web and mobile apps
- Web & Mobile (iOS, Android) SDKs
- Record and store inbound or outbound calls
- Send text-to-speech messages in 40 languages with different genders and accents
- Create conference calls

# Voice API Webhooks


Your App
Vonage
Configure URL for webhook
Some time later ...
Sending some interesting data
200 OK - I got it, thanks
Your App
Vonage

Webhooks provide a convenient mechanism for Vonage to send information to your application for events such as an incoming call or message, or a change in call status.

**Webhooks**

- Answer Webhook
- Event Webhook
- Fallback URL

## Answer Webhook

- ❖ Answer webhook is sent when a call is answered. This is for both incoming and outgoing calls.
- ❖ When an incoming call is answered, an HTTP request is sent to the answer_url.

**Answer webhook data fields**

| **Fields** | **Description** |
|---|---|
| to | The number that answered the call. |
| from | The number that called **to.** This could be a landline or mobile number |
| From_user | The username that called **to** only if the call was made using  the client SDK. |
| uuid | Unique identifier for this call. |
| Conversation_uuid | Unique identifier for this conversation |
| Region_url | Regional API endpoint which should be used to control the call with **REST API**. |
| Custom_data | Custom data object, optionally passed as parameter. |

## Event webhook

- Event webhook is sent for all the events that occur during a call. Your application can log, react to or ignore each event type.

- HTTP requests will arrive at the event webhook endpoint when there is any status change for a call.

- By default the incoming requests are `POST` requests with a JSON body.

### Event webhook data fields

| Fields | Descriptions |
|---|---|
| from | The number the call came from |
| to | The number the call was made to |
| uuid | Unique identifier this call |
| conversation_uuid | Unique identifier this conversation |
| status | Call status |
| direction | Call direction |
| timestamp | Timestamp (ISO format) |

**Status Types**

Started

Ringing

Answered

Busy

Cancelled

Unanswered

Disconnect

Rejection

Failed

Timeout

Human / machine

Completed

Record

Input

Transfer

**Direction Type**

Inbound

Outbound

## Fallback URL

Fallback URL is used when either the Answer or Event webhook fails or returns an HTTP error status.

**example**

```
{
  "reason": "Connection closed.",
  "original_request": {
    "url": "https://api.example.com/webhooks/event",
    "type": "event"
  }
}
```

**NCCO - Nexmo Call Control Objects**

A Call Control Object (NCCO) is represented by a JSON array. You can use it to control the flow of a Voice API call.

The Call event model is asynchronous. When a Call is placed to your number, Vonage makes a synchronous request to the webhook endpoint you set as the answer_url for your number and retrieves the NCCO object that controls the Call.

**NCCO instruction are:**

- Action - something to be done in the Call.
- Option - how to customize an *action*.
- Type - describes an *option*. For example, type=phone for an endpoint option.

**actions you can use in an NCCO are:**

- record - all or part of a call
- conversation - create a standard or hosted conversation
- connect - connect to a connectable endpoint such as a phone number or Vonage Business Cloud extension
- talk - send synthesized speech to a conversation
- stream - send audio files to a conversation
- input - collect digits from the person you are calling, then process them

## Creating a custom call or conversation for each user

When you make an outbound call or accept an inbound call, Vonage makes a request to your webhook endpoint at answer_url and retrieves your NCCO. This request contains the following parameters:

| **Name** | **Description** |
|---|---|
| to | The endpoint being called. |
| from | The endpoint you are calling from. |
| conversation_uuid | Unique ID for this conversation |
| uuid | Unique ID for this call |

**Code examples show how to provide the NCCO that controls your call or conversation**

```python
from flask import Flask, request, jsonify

app = Flask(__name__)

HOST = "localhost"
PORT = 3000

@app.route("/webhooks/answer")
def answer_call():
    call_from = request.args['from']

    # Dynamically build NCCO based on source phone
    if call_from == "447700900000":
        ncco = [{
                "action": "talk",
                "text": "Hi John, we will be with you shortly."
            }]
    elif call_from == "447700900001":
        ncco = [{
                "action": "talk",
                "text": "Hi Jane, we will be with you shortly."
            }]
    else:
            ncco = [{
                "action": "talk",
                "text": "Hello, sorry, we do not recognize your number."
            }]
    return jsonify(ncco)

if __name__ == 'main':
    app.run(host=HOST, port=PORT)
```

## Web Sockets

WebSockets is a computer communications protocol that enables two-way communication over a single, persistent TCP connection without the overhead of the HTTP request/response model.

Using Vonage's Voice API, you can connect phone calls to WebSocket endpoints. This means that any application that hosts a WebSocket server can be a participant in a Vonage voice conversation. It can receive raw audio from and play audio into the call in real time.

**Automating calls with bots to perform tasks such as food ordering or requesting information from field experts.**

The endpoint is addressed via a `uri` parameter which should be a standard websocket URL, starting with either `ws://` for plain HTTP or `wss://` for TLS enabled servers.

WebSockets allow you to connect phone calls to any AI bot engine of your choice.

### Working with WebSockets

when establishing the WebSocket connection.

- Return an NCCO instructing Vonage to connect to your WebSocket endpoint
- Accept this WebSocket connection
- Handle JSON text-based protocol messages
- Handle mixed call audio binary messages

### Connecting to a WebSocket

- Vonage to connect to a WebSocket your application server must return an NCCO when requested from your Vonage Application's answer_url
- NCCO must contain a `connect` action with an `endpoint.type` of `websocket`

Example

```
[
    {
       "action": "connect",
       "endpoint": [
           {
                "type": "websocket",
                "uri": "wss://example.com/socket",
                "content-type": "audio/l16;rate=16000",
                "headers": {
                    "language": "en-GB",
                    "caller-id": "447700900123"
                }
           }
       ]
    }
]
```

The specific data fields for webhooks

| Fields | Descriptions |
|---|---|
| uri | Endpoints of your websocket server that vonage will connect to |
| Content-type | String representing the audio sampling rate, **audio/l16;rate=16000** or **audio/l16;rate=8000** and **audio at 8kHz**. |
| headers | additional optional properties to send to your Websocket server |

## Handling incoming WebSocket messages

The initial message sent on an established WebSocket connection will be text-based and contain a JSON payload, it will have the `event` field set to `websocket:connected` and detail the audio format in `content-type`, along with any other metadata that you have put in the `headers` property of the WebSocket endpoint in your NCCO `connect` action.

**example**

```
{
    "event":"websocket:connected",
    "content-type":"audio/l16;rate=16000",
    "prop1": "value1",
    "prop2": "value2"
}
```

## Binary audio messages

Messages that are binary represent the audio of the call. The audio codec presently supported on the WebSocket interface is Linear PCM 16-bit, with either a 8kHz or a 16kHz sample rate, and a 20ms frame size.

To choose the sampling rate set the Content-Type property to audio/l16;rate=16000 or audio/l16;rate=8000 depending on if you need the data at 16kHz or 8kHz.

| Sampling rate | Number of samples in 20ms | Bytes per message |
|---|---|---|
| 8000 | 160 | 160 * 2 = 320 |
| 16000 | 320 | 320 * 2 = 640 |

# Call Flow

**Inbound** calls are made to the Vonage platform by one of the following methods:

- to a Vonage number from a regular phone,
- from a client application using the Client SDK.

**Outbound** calls are calls made from the Vonage platform to

- a regular phone number,
- client application, or
- WebSocket server.

**Outbound** calls are usually initiated in response to a request made via the REST API to create a new call.

**Two types of call flow**

- **Scripted Call**: when the flow is determined by a sequence of question-answer steps (actions);
- **Live Conversation**: which connects two or more participants in a conversation.

## Scripted Call:

inbound and outbound calls initially follow the same call flow once answered. This call flow is controlled by an NCCO.

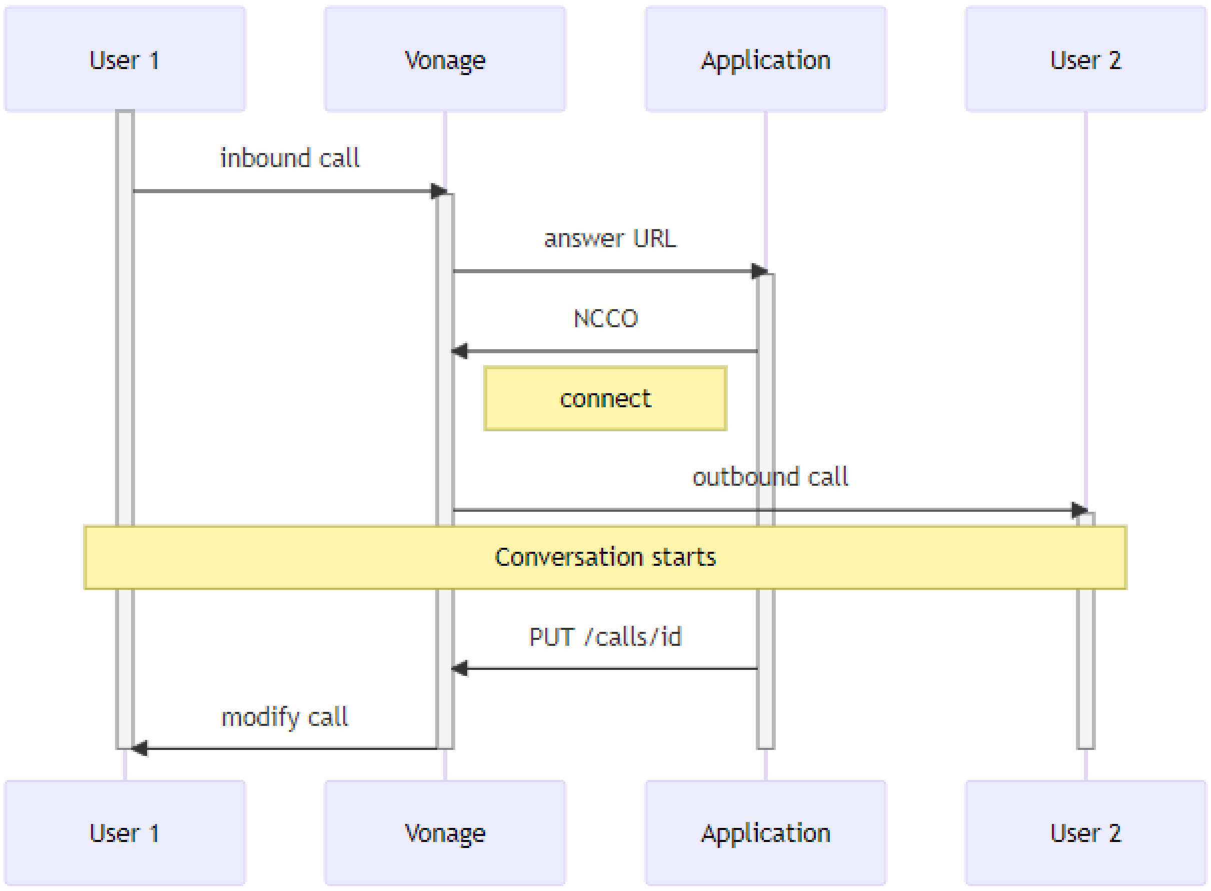
- inbound calls, the `answer_url` is configured in your [Voice Application](#).
- outbound calls, you provide an `answer_url` in the API request that creates the call.

## Live Conversation:

A **private voice communication** use case, you want to connect two or more participants to establish a live conversation.

Each call, inbound or outbound, is automatically added to the new conversation behind the scenes.

- Create a new outbound call with the `connect` action - it will be automatically joined to the same conversation;
- Move the call to an existing (or new) named conversation with the `conversation` action.



Since any type of voice endpoint might be used in the connect action, **the second member is not necessarily a human**: it might be a **voice bot** talking to the user using the media passed through the **WebSocket connection**.

# Text to Speech

Vonage uses text-to-speech engines to allow you to play machine generated speech to your users. This can either be done via an NCCO with the use of the talk action, or by making a PUT request to an in-progress call.

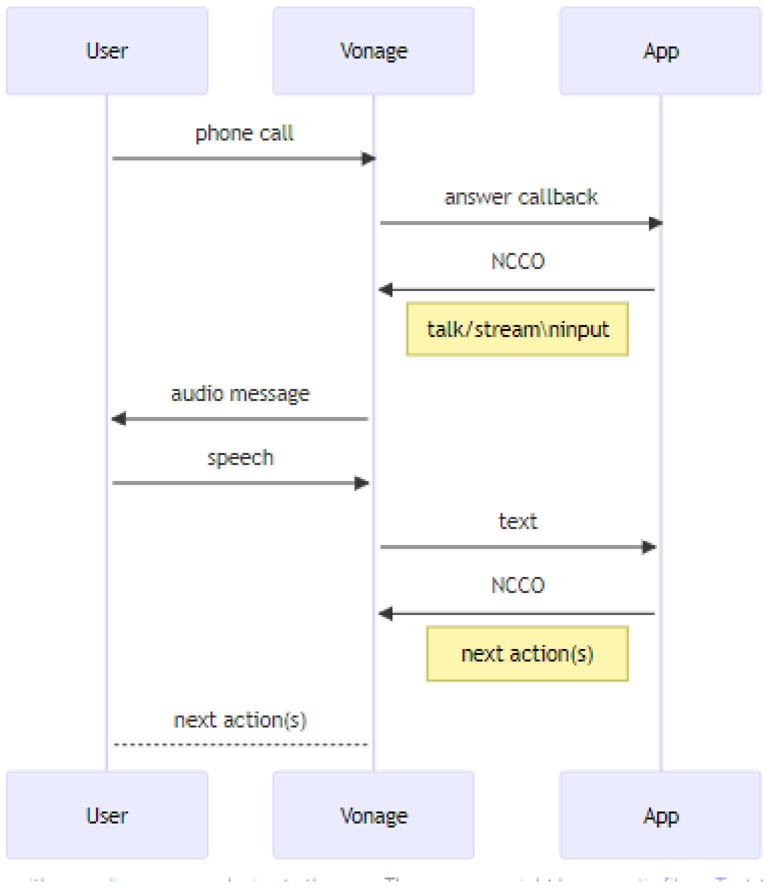
## Example

```
[
  {
    "action": "talk",
    "text": "Thank you for calling. Please leave your message after the tone."
  }
]
```

## Speech to Text

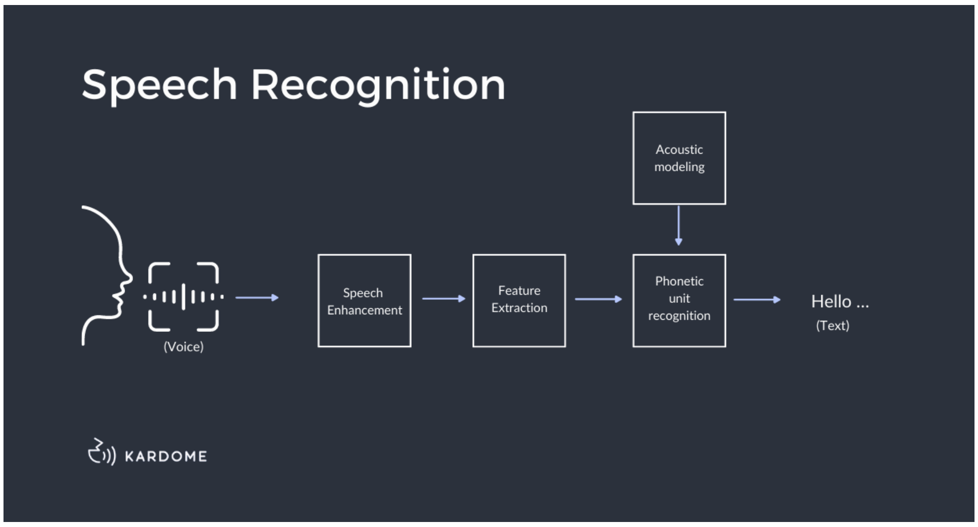
Automatic Speech Recognition (ASR) enables apps to support voice input for such use cases as IVR, identification and different **kinds of voice bots/assistants**. Using this feature, the app gets transcribed **user speech (in the text form)**

### How it works



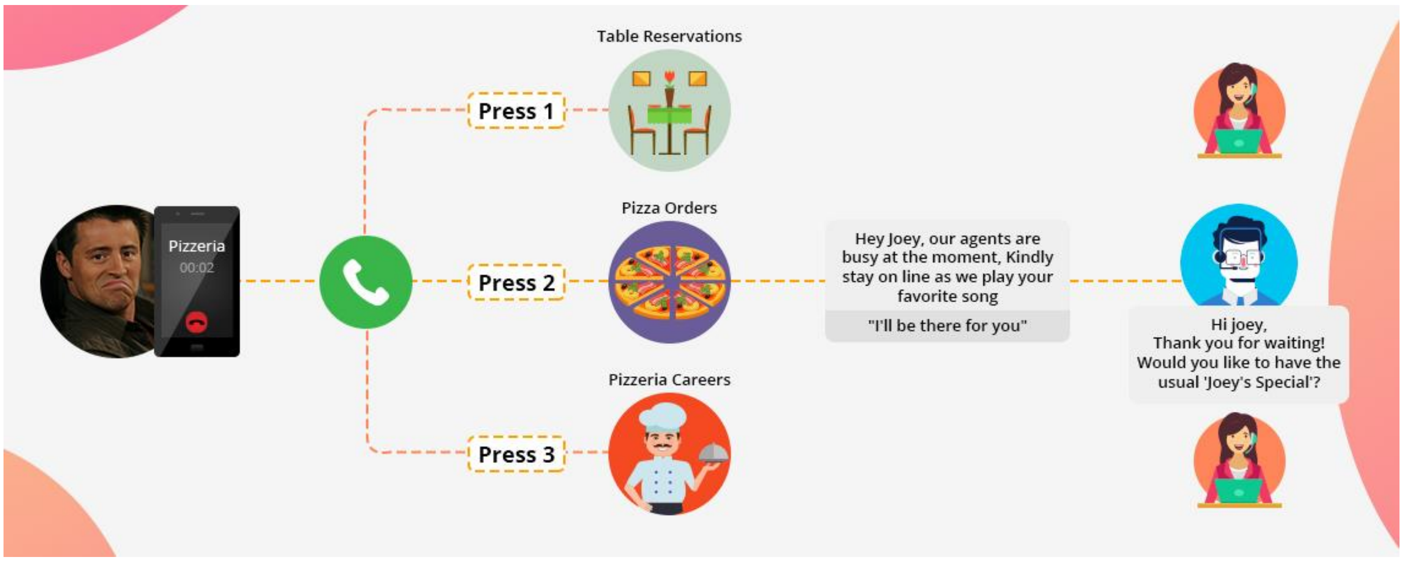
## ASR

ASR, is the use of Machine Learning or Artificial Intelligence (AI) technology to process human speech into readable text.



## IVR

Interactive Voice Response (IVR) is an automated phone system technology that allows incoming callers to access information via a voice response system of pre recorded messages without having to speak to an agent, as well as to utilize menu options via touch tone keypad selection or speech recognition.


Pizzeria
00:02
Table Reservations
Press 1
Pizza Orders
Press 2
Pizzeria Careers
Press 3
Hey Joey, our agents are busy at the moment, Kindly stay on line as we play your favorite song
"I'll be there for you"
Hi joey, Thank you for waiting! Would you like to have the usual 'Joey's Special'?

## Make an outbound call

This code snippet makes an outbound call and plays a text-to-speech message when the call is answered.

```
client = vonage.Client(
    application_id=VONAGE_APPLICATION_ID,
    private_key=VONAGE_APPLICATION_PRIVATE_KEY_PATH,
)

response = client.voice.create_call({
  'to': [{'type': 'phone', 'number': TO_NUMBER}],
  'from': {'type': 'phone', 'number': FROM_NUMBER},
  'answer_url': ['https://raw.githubusercontent.com/nexmo-community/ncco-
examples/gh-pages/text-to-speech.json']
})

print(response)
```

## Make an outbound call with an NCCO

This code snippet makes an outbound call and plays a text-to-speech message when the call is answered. You don't need to run a server hosting an `answer_url` to run this code snippet, as you provide your NCCO as part of the request.

```
client = vonage.Client(
    application_id=VONAGE_APPLICATION_ID,
    private_key=VONAGE_APPLICATION_PRIVATE_KEY_PATH,
)

response = client.voice.create_call({
    'to': [{'type': 'phone', 'number': TO_NUMBER}],
    'from': {'type': 'phone', 'number': VONAGE_NUMBER},
    'ncco': [{
        'action': 'talk',
        'text': 'This is a text to speech call from Nexmo'
    }]
})

pprint(response)
```

## Receive an inbound call

In this code snippet you see how to receive an inbound call.

```python
from flask import Flask, request, jsonify

app = Flask(__name__)


@app.route("/webhooks/answer")
def answer_call():
    for param_key, param_value in request.args.items():
        print("{}: {}".format(param_key, param_value))

    from_ = request.args['from']

    return jsonify([
        {
            "action": "talk",
            "text": "Thank you for calling from " + " ".join(from_)
        }
    ])
```

# Voice API

## Calls

Fetch, Create and Modify voice calls

**Available Operations:**

**GET -** Get details of your calls

https://api.nexmo.com/v1/calls/

**POST -** Create an outbound call

https://api.nexmo.com/v1/calls/

**GET -** Get detail of a specific call

https://api.nexmo.com/v1/calls/:uuid

**PUT -** Modify an in progress call

https://api.nexmo.com/v1/calls/:uuid

## Stream Audio

Start or stop streaming audio in to an active call

**Available Operations**

**PUT -** Play an audio file into a call

https://api.nexmo.com/v1/calls/:uuid/stream

**DELETE -** Stop playing an audio file into a call

https://api.nexmo.com/v1/calls/:uuid/stream

## Play TTS

Start or stop playing Text to Speech in to an active call

**Available Operations**

**PUT** - Play text to speech into a call

https://api.nexmo.com/v1/calls/:uuid/talk

**DELETE -** Stop text to speech in a call

https://api.nexmo.com/v1/calls/:uuid/talk

## Play DTMF

Play DTMF tones in to an active call

**Available Operations**

**PUT** - Play DTMF tones into a call

https://api.nexmo.com/v1/calls/:uuid/dtmf