COGINT.AI

Modern Conversational IVRs with Dialogflow and Voximplant

whitepaper sponsored by

voximplant

written by

Chad Hart
chad@cwh.consulting

December 2019

Table of Contents

Executive Summary

Voicebots – speech-powered artificial intelligence technology – are poised to replace traditional touchtone-based Interactive Voice Response (IVR) systems and will become the predominate method for implementing conversational IVR systems. Major Artificial Intelligence (AI) investments, massive install-bases that drive diverse user data, and recent breakthroughs in deep learning approaches have resulted in highly performant bots that can effectively mimic human interactions. The continued democratization of both voicebot technology and telephony infrastructure is changing the economics and opening the market for Conversational IVRs – Interactive Voice Response systems powered by AI and speech technologies. This whitepaper describes approaches for integrating voicebots with telephony systems and reviews how Voximplant, a long-standing Communications Platform as a Service (CPaaS) vendor – does this with Dialogflow.

Voicebots – speech-powered artificial intelligence technology – are poised to replace traditional IVR systems

Dialogflow, a product acquired by Google, has proven to be among the most popular voicebot development platforms due to its broad ecosystem of partners, leading language support, and mature bot development interface. Dialogflow offers a simplistic, but very easy to use Phone Gateway option for accepting inbound calls from the US. Alternatively, calls can be forwarded from an existing platform into Dialogflow to add some flexibility. However, this approach adds latency, cost, and fails to provide more than superficial interaction between the telephony system and the bot.

A better approach is to use an RTC-Bot gateway to interface between the telephony platform and the Dialogflow bot. A sophisticated Dialogflow gateway could then also easily handle features such as call transfer, recording, DTMF detection, and SMS interaction while supporting Dialogflow's programmatic interaction APIs to allow for natural dialog with features like barge-in and no activity detection.

Voximplant is an established CPaaS vendor with a leading Dialogflow gateway solution. Voximplant provides a distinctive serverless platform that speeds development and minimizes infrastructure maintenance with strong global network coverage and international language support. Voximplant's Dialogflow Connector offers unique support for interfacing with Dialogflow's Graphical User Interface (GUI) options and control APIs, giving it leading coverage of RTC-Bot gateway needs.

Voicebots will Replace Traditional IVRs

Interactive Voice Response (IVR) technology based on Dual Tone Multi Frequency (DTMF) touchtone user entry has served the communications industry well for decades, but its time is coming to an end as superior technologies for automating customer interaction are becoming mainstream. Advances with speech recognition and speech synthesis have allowed for some replacement of DTMF with speech input. However, the rigid, hierarchical logic of these IVRs is still jarring compared to natural human conversation. Conversational IVRs that allow for user input with natural, spoken language without highly-layered menu navigation have also been available for many years. While traditional Conversational IVR approaches have matured, historically this technology has been very expensive and therefore limited in its application. Today, modern voicebot technology is changing that.

Voicebots – speech-powered artificial intelligence technology – have become mainstream in consumer electronics and computing applications. Advances in machine learning and growing troves of user data have brought revolutionary price/performance advantages to not only speech recognition and synthesis, but also Natural Language Understanding (NLU). Consumerization of this technology by industry giants like Apple, Google, Microsoft, and Amazon through voice assistants and smart speakers has made this technology available to more than a billion users on a regular basis.

Ironically, while users frequently interact with voicebots on their phones and smart speakers, it is rare for them to experience this technology when making phone calls to businesses. This is due to several factors that have made significant recent changes:

Factor	How it is changing
Concerns about speech performance with consumer-oriented technologies	Major improvements in language support, recognition rates, synthesized speech vocal quality, and natural language understanding propelled by mass deployment of voicebot technology
Closed systems that make integration difficult or impossible	Movement toward developer-oriented open source and CPaaS telephony platforms

Lack of native telephony support in new voicebot systems	Voicebot gateway options emerging to link voicebots to telephony systems

As the consumer voicebot market matures, modern bot vendors are turning their focus to telephony. Recent examples of this include Google's Contact Center AI initiative, IBM's Voice Gateway for Watson Assistant launch, and Amazon's deeper integrations with Connect – their hosted contact center platform as a service. cogint.ai expects the transition to voicebot-based IVRs to accelerate as the use cases become more public and the technology becomes simplified enough to appeal to broader markets.

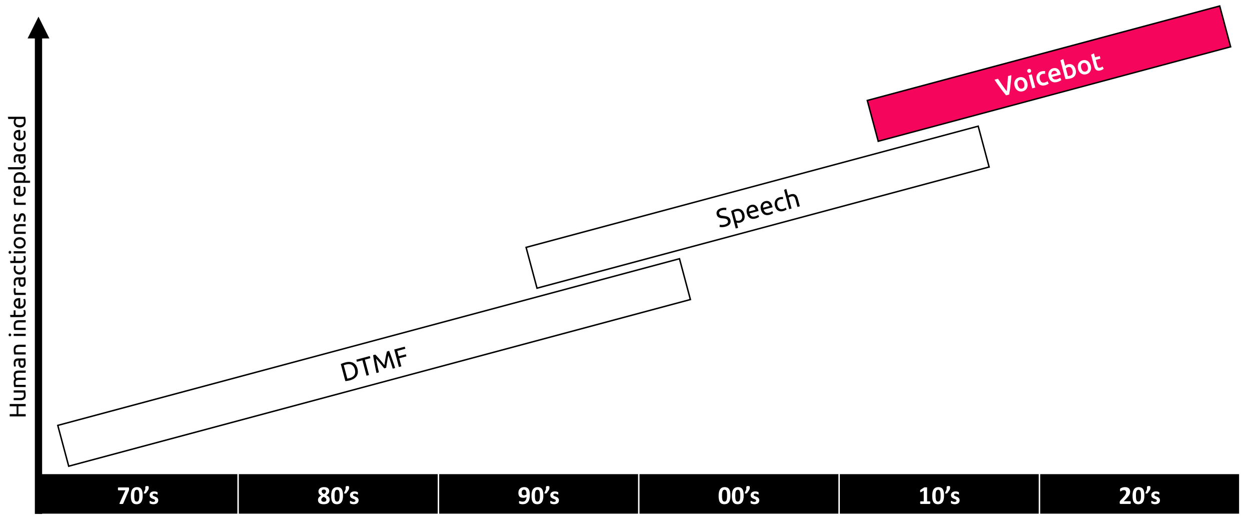


Figure 1 – cogint.ai view of IVR technology trends over time

Why Use Dialogflow in a Voicebot IVR?

Many mature bot platforms exist on the market today. In addition to traditional conversational IVR vendors like Nuance, more general computing and AI giants like Google, Microsoft, Amazon, and IBM are all targeting telephony use cases for their bot platforms. Each of these vendors brings their own portfolio of AI-based speech and natural language tools supported by overlapping, but distinct ecosystems.

Dialogflow's differentiation is due in large part to its unique history as a voicebot platform and Google's leadership in AI products. api.ai – the precursor to Dialogflow - was launched in 2014. It was one of the first bot development frameworks aimed at multiple hardware and messaging platforms. It was also among the few with a focus on voice interaction. Google later bought the company behind api.ai and rebranded the service as Dialogflow. Since then, Google has combined api.ai's core bot technology with their comprehensive speech APIs while continuing to invest in the product as a multi-channel offer. This ecosystem has bolstered Dialogflow's appeal outside of the immediate Google Assistant ecosystem.

While the best bot platform for a given scenario certainly depends on individual needs and requirements, Dialogflow is generally compelling as a voicebot platform for several reasons:

- **Integrations** – Dialogflow's origins as an independent vendor has helped it develop a broad ecosystem of integrations with other messaging and communications apps. Investment in the bot can be easily leveraged across other communications channels, such as web chat or messaging platforms, and investments for other channels can be reused for telephony.
- **Language support** – Dialogflow leads when diverse language support is needed. The core Dialogflow NLU engine currently supports more languages than any other voicebot platform. Dialogflow includes many speech recognition options and a wide variety of voices for use in synthesis, including its advanced WaveNet models derived in part from Google's $500+ million acquisition of DeepMind. These synthesized voices sound significantly more lifelike than previous models.
- **User interface maturity** – Dialogflow's graphical user interface is designed to minimize the need for development and simplify the development process. The interface includes elements like:
  - a Knowledge Connector for ingesting and auto-populating the bot with knowledgebase data,
  - dedicated Channel tabs for interacting with different interfaces like telephony and messaging platforms,
  - small talk interactions that make the bot naturally handle interactions without requiring specific programming,
  - history and performance improvement tools, and even
  - a quick method for running application logic in a serverless environment without leaving the interface.

As a result, Dialogflow has proven to be among the most popular platforms for developers. This is illustrated below by using Stack Overflow questions as a proxy of developer interest.

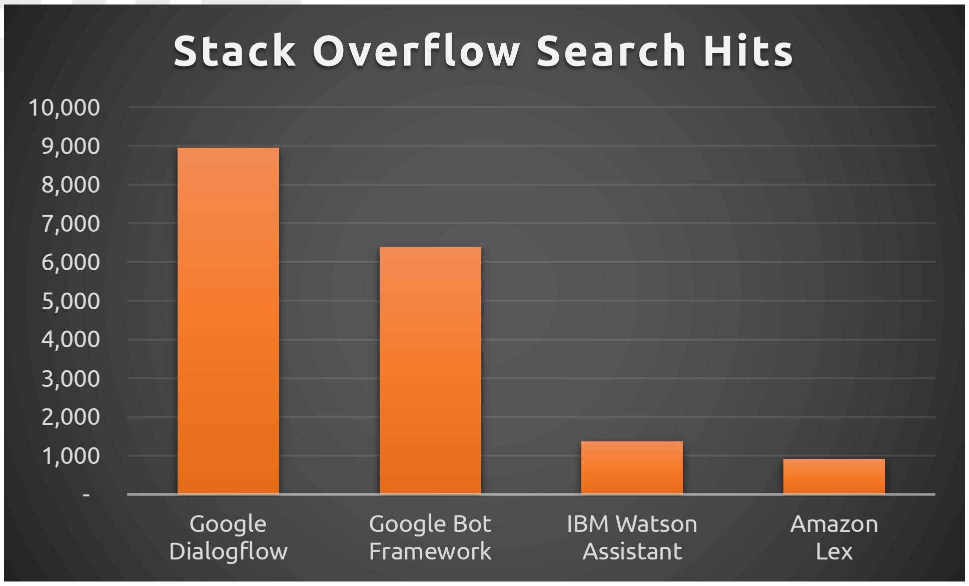


Figure 2- Stack Overflow search results as a proxy for developer interest in each bot platform[1]

Connecting Dialogflow to Phone Calls

At its core, Dialogflow and most other bot systems were originally designed to handle text interaction. There was no underlying audio input and output hardware platform to worry about, but that changed with smart speakers and the growing use of voice assistants. Today, if you are developing for the Google Assistant, Dialogflow abstracts the underlying physical platform that is running the bot so you don't have to worry about the hardware. On the other hand, if you have your own hardware platform you will need to either utilize Dialogflow's standard API interfaces for handling media or figure out your own way to capture microphone input and playback audio. In the case of telephony, the infrastructure that mediates between Dialogflow and the telephony network is an RTC-Bot gateway.

What is an RTC-Bot Gateway?

To connect a phone call into Dialogflow, the RTC-Bot gateway needs to handle both signaling and media conversion.

[1] Results are meant to be directional indicators for developer interest. Due to limitations on Stack Overflow's search interface and variances in topic tagging, search terms varied to exclude irrelevant results while capturing real queries for the bot platform. Actual searches were as follows: Dialogflow, Microsoft Bot Framework / Azure Bot Service, IBM Watson Assistant (+ Assistant with IBM Watson), Amazon Lex

On the signaling side, the gateway needs to take the telephony signaling - which is almost always based on the SIP protocol - and use that to invoke the proper Dialogflow commands to launch and interact with the bot. Signaling also includes handling hang-ups, termination of the call, and other features that will be reviewed later in this paper such as DTMF events.

The media conversion required is slightly more complicated. Dialogflow's interface for real time speech input is gRPC – a full duplex binary transmission technology built for HTTP/2. The gateway needs to convert the media used by the telephony end (RTP or encrypted SRTP) to a gRPC bitstream using Dialogflow-friendly codecs. The gateway also needs to generate the response speech and sound coming from Dialogflow to the user. Often, this just means using the response speech generated by Dialogflow if that option is enabled. Alternatively, it could use its own Text-to-Speech mechanism to vocalize Dialogflow's response text.

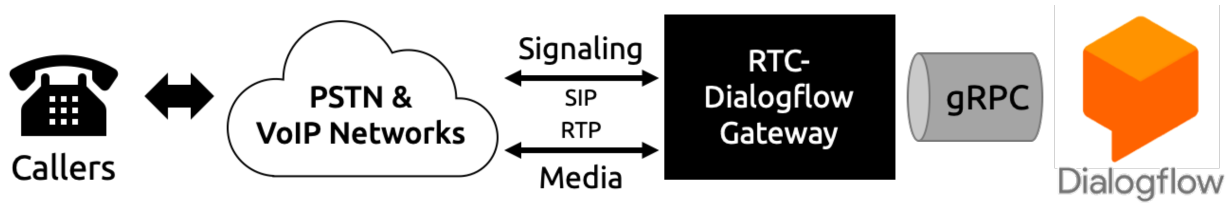


Figure 3 - Illustration of an RTC to Dialogflow Gateway

RTC-Dialogflow Gateway Features

In addition to providing basic call setup and media connectivity, there are many other features described below that could or should be handled by the gateway.

Telephony Interaction

IVR systems and telephony environments introduce new requirements that are not typically handled by native Dialogflow apps. These may include:

- **Call transfer** - In most cases you need to give the user an option to direct the call to a human.
- **DTMF detection** - Even if the goal is to eliminate DTMF menus, sometimes DTMF is needed as a backup option or alternative input method - especially if you are trying to do something like capture a phone number or other long numerical entry.
- **Recording** - Having a full recording of both parties is invaluable for debugging and improving the system. Recording may also be necessary for quality management or compliance requirements.

- **SMS** – Most users have mobile devices, meaning they can receive SMS messages. This is a nice option to have for sending reference information like web links and creates a new possibility for allowing the interaction to continue asynchronously offline.

Dialogflow Interaction

Dialogflow expects there to be an underlying interface platform that provides some control over user input and output – be that a smartphone, webpage, Google Home devices, or other smart device. When interacting with telephony networks, the gateway needs to take on the role of this interface platform.

- **Telephony tab support** – As will be discussed in the next section, Dialogflow includes a native Telephony tab with GUI elements for transferring a call, synthesizing speech, and playing back media. Ideally the gateway can leverage these elements to simplify bot development and debugging.
- **Playback interruption** - Your voicebot should be able to handle situations where the user may barge in and start talking over one of the bot's response prompts. This is more likely to happen when the prompt is long, or the response is something other than what the user was looking for.
- **No activity detection** - If you were in the middle of a conversation and it suddenly went silent, you would say "are you there"? The bot needs the mechanism to do something similar. If you were using Dialogflow to make a Google Assistant bot, they provide an `actions_intent_NO_INPUT` event and mechanisms to setup reprompt intents. The gateway needs to provide something similar.
- **Custom events and contexts** – Dialogflow uses events to handle non-user based inputs and contexts to filter intents and pass information. Additional events and contextual information like a phone number from the gateway can improve the interactivity of the bot.

Current Dialogflow Gateway Approaches

There are currently three main approaches for connecting phone systems to Dialogflow:

1. Use Dialogflow's built-in Phone Gateway
2. Forward a call from a telephony system into Dialogflow's Phone Gateway
3. Directly connect a telephony system to Dialogflow

Each of these methods are reviewed below.

Dialogflow's Phone Gateway

Dialogflow includes a native integration with its own Phone Gateway. This capability has been in Beta since June 2018. In this case Beta appears to mean the product has limited capability since Service Level Agreement (SLA) options do exist.

The integration is extremely easy to use, but comes with many limitations as summarized below:

Advantages	Disadvantages
• Minutes to set up inside the Dialogflow GUI • It is free, subject to quotas unless you want the paid Enterprise Edition version • A `TELEPHONY_WELCOME` event is included for special handling of phone calls • The Dialogflow Intent GUI includes a tab for special handling of telephony responses, including audio playback, special speech synthesis, and call transfer • Easily transfer calls to a single US number	• Only US English supported • Only US numbers supported • Inbound-only call flows • Only one inbound number supported • Extremely basic feature set – only Call Transfer and Telephony Tab support are supported from the previous section • Phone number connectivity-only (no SIP option) • No control of the gateway • No gateway debugging tools

Forwarding Calls to the Dialogflow Phone Gateway

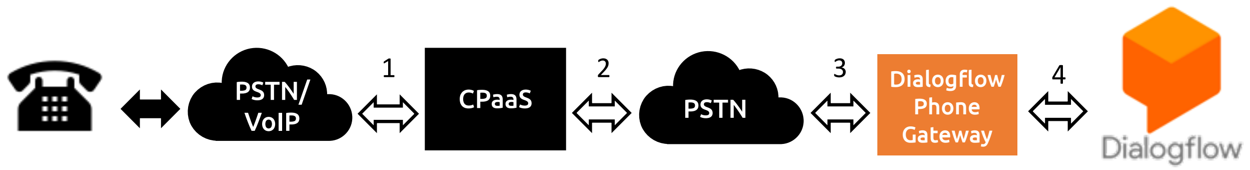


Figure 4 - Illustration of the call forwarding approach for connecting to Dialogflow

As covered in the previous section, Dialogflow does not have a whole lot of its own telephony controls, but it is possible to use another telephony platform to forward calls to the Dialogflow Phone Gateway. This platform could be anything - a commercial PBX or ACD system, an open source telephony platform like Asterisk or FreeSWITCH, or a Communications Platform as a Service (CPaaS) that provides this functionality as a cloud-based service. This approach enables some additional features if you are willing to leverage Dialogflow's webhooks to interact with the telephony platform.

Advantages	Disadvantages
• Use any phone number the platform supports • Support for multiple phone numbers • Easy recording - this is almost always available as a feature • More control over call transfer if your platform can handle incoming webhooks and programmatic control • Handles SIP calls if your platform supports it • Supports inbound and outbound call flows • Advanced call flows - such as conferencing in an agent to listen or help with the voicebot interaction	• No Dialogflow interaction features – there is no way to signal events or send contexts to Dialogflow (without coding those parts yourself) • Extra costs – you need to pay for calls on your existing platform plus Dialogflow (assuming you use their Enterprise plan) • Additional transcoding overhead decreases call quality • Added latency from the extra PSTN routing could decrease performance and call quality • Only some platforms support SMS integration

Direct Connectivity to Dialogflow

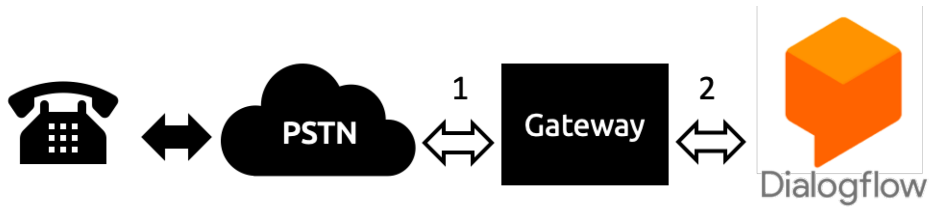


Figure 5 - The direct connectivity approach for interfacing with Dialogflow minimizes latency

Connecting the gateway directly to Dialogflow saves conversion steps. This architecture also introduces new feature options because the gateway can directly signal and control Dialogflow.

Advantages	Disadvantages
• All of the same benefits as in the forwarding methodology • The full feature set mentioned above may be possible depending on the gateway used • Lower cost – cogint.ai analysis has shown this option is less expensive from several vendors than Dialogflow's telephony charges on the Enterprise plan[2] • Better quality - lower latency with fewer intermediary networks that could impact voice quality • Potential use of end-to-end HD audio if connecting with VoIP for better performance/quality	• Relatively few market options available • Platform dependency – these gateways often have unique integration features that may not be supported or require bot reprogramming if changing gateways

[2] See the Voximplant analysis below and additional vendor analysis on cogint.ai here.

Voximplant Dialogflow Connector Review

Voximplant was the first CPaaS provider to offer direct connectivity options for Dialogflow and they continue to be one of the few solutions on the market. Voximplant's platform and development approach is well suited to voicebot integrations, helping them hold a mature position despite the relative newness of Dialogflow with telephony.

Platform and Approach

Serverless Execution

Unlike the VXML-inspired approach used by many platforms, Voximplant has a JavaScript-based execution environment it calls VoxEngine. While not terribly complex, it does require some JavaScript skills if you want to do more than copy and paste example code you may find. cogint.ai finds Voximplant's JavaScript much easier to work with than XML-oriented approaches, especially with their code-completion tools.

This all runs serverless. There is no development and production environment that needs to be set up with corresponding instance installation, load balancing, operating system updates, and all the work involved with maintaining a high-volume server with the ability to run your own software. VoxEngine is very quick to get going, especially if you only need to manage a few lines of code.

VoxEngine is also a web-based Integrated Development Environment (IDE) that includes code completion and debugging tools. Despite these, developers may start to miss their preferred IDE as your VoxEngine scripts grow in complexity. If switching to Voximplant's cloud-based development environment is an issue, developers can - with some additional work - use Voximplant's APIs to synchronize VoxEngine's JavaScript files with their own environment instead.

Dialogflow API Wrappers

Voximplant provides sophisticated control over its interaction with Dialogflow. It has effectively wrapped most of the Dialogflow APIs for interacting with an existing agent (but not the ones for programming an agent).

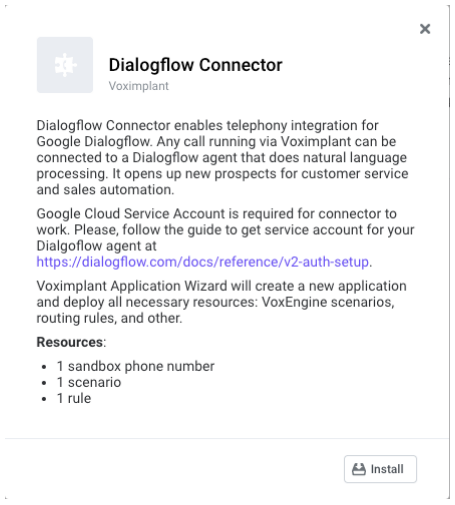
This API includes a method to start a Dialogflow session and a number of events and classes:

Classes	Events
• DialogflowError • DialogflowPlaybackFinished • DialogflowPlaybackMarkerReached • DialogflowPlaybackStarted • DialogflowResponse • DialogflowStopped	• DialogflowEventInput • DialogflowInstance • DialogflowOutputAudioConfig • DialogflowQueryInput • DialogflowQueryParameters • DialogflowResponse • DialogflowResult • DialogflowSettings • DialogflowStreamingRecognitionResult • DialogflowSynthesizeSpeechConfig • DialogflowTextInput • DialogflowVoiceSelectionParams

Some of these events and classes are explored in more detail in the feature review section.

Dialogflow Integration Tool

Voximplant has added some features to make getting started with their Dialogflow Connector quick and easy. In addition to the reference guide for setting up the Dialogflow Connector, Voximplant also has an integration option inside their Marketplace. This Marketplace mechanism provides a guided GUI that walks users through the setup. You can find this off of their main side menu.



Dialogflow Connector
Voximplant

Dialogflow Connector enables telephony integration for Google Dialogflow. Any call running via Voximplant can be connected to a Dialogflow agent that does natural language processing. It opens up new prospects for customer service and sales automation.

Google Cloud Service Account is required for connector to work. Please, follow the guide to get service account for your Dialgoflow agent at https://dialogflow.com/docs/reference/v2-auth-setup.

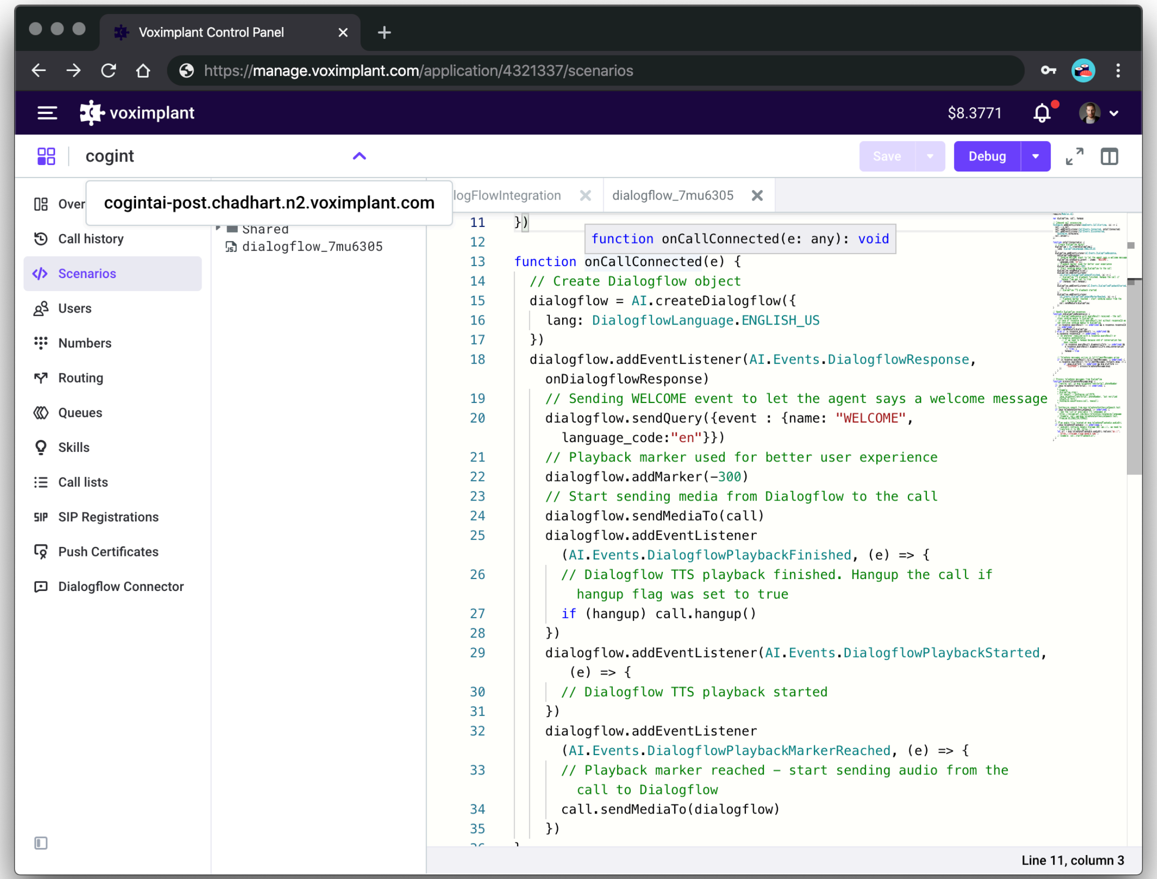
Voximplant Application Wizard will create a new application and deploy all necessary resources: VoxEngine scenarios, routing rules, and other.

Resources:

- 1 sandbox phone number
- 1 scenario
- 1 rule

Figure 6 - Voximplant's Marketplace includes a guided Dialogflow Connector option

From there you will need to go into the Google Cloud Console to get your service account key. The setting up authentication guide from the Dialogflow documentation explains the steps to obtain that JSON key file. Once you upload your key file, Voximplant will create a new application that includes the Agent connection. You should see the default Dialogflow Connector code show up in your scenarios list:



```
})

function onCallConnected(e) {
  // Create Dialogflow object
  dialogflow = AI.createDialogflow({
    lang: DialogflowLanguage.ENGLISH_US
  })
  dialogflow.addEventListener(AI.Events.DialogflowResponse,
    onDialogflowResponse)
    // Sending WELCOME event to let the agent says a welcome message
    dialogflow.sendQuery({event : {name: "WELCOME",
      language_code:"en"}})
    // Playback marker used for better user experience
    dialogflow.addMarker(-300)
    // Start sending media from Dialogflow to the call
    dialogflow.sendMediaTo(call)
    dialogflow.addEventListener
      (AI.Events.DialogflowPlaybackFinished, (e) => {
      // Dialogflow TTS playback finished. Hangup the call if
        hangup flag was set to true
      if (hangup) call.hangup()
    })
    dialogflow.addEventListener(AI.Events.DialogflowPlaybackStarted,
      (e) => {
      // Dialogflow TTS playback started
    })
    dialogflow.addEventListener
      (AI.Events.DialogflowPlaybackMarkerReached, (e) => {
      // Playback marker reached - start sending audio from the
        call to Dialogflow
      call.sendMediaTo(dialogflow)
    })
```

Figure 7 - VoxEngine Dialogflow code template

Alternatively, you can also access, add, and delete an agent in the Dialogflow Connector menu inside your application. Note, it will not automatically create the Scenario code f you add a Dialogflow agent this way.

Voximplant's guide mentions this, but don't forget to set your Text-to-Speech configuration inside Dialogflow's Settings to *MP3* or *Ogg Opus*.

Global Coverage

Voximplant also differentiates itself from other CPaaS competitors in terms of its global server infrastructure.

In real time communications, having telephony media infrastructure close to end users generally helps to reduce latency and increase quality. In this regard, more distinct geographic coverage is better when it comes to data centers. Data center diversity also helps to add redundancy.

An analysis of Voximplant's published IP addresses indicates that they have infrastructure in 15 distinct countries across different data center vendors including six different countries in Europe. Similar analysis of the published IP addresses of other CPaaS providers indicates fewer data center country locations and only a single data center provider:

CPaaS Vendor	Data center countries	Data center providers
Voximplant	12	5
Twilio	8	1
Nexmo	5	1
Messagebird	1	1

Language Support

Voximplant also boasts strong language support for speech recognition (Speech-to-Text) and speech synthesis (Text-to-Speech). Their VoxEngine ASR Module supports more than 80 languages options. For TTS, you can choose from three different transcription engines including Google. Dialogflow offers strong speech recognition support so it is unlikely the ASR module would be needed in that environment, but ASR could be useful with other bot NLU engines and interaction outside of the bot.

Similarly, the VoxEngine Player Module can be used to synthesize over 121 distinct voices for 30 languages with some distinct dialect options. As with speech recognition, this feature is unlikely to be needed with Dialogflow unless you have a preference for one of the non-Google voices.

Voximplant also recently released a WebSocket interface designed to make it easier to with external speech recognition, speech synthesis, and Natural Language Understanding services.

Pricing

Voximplant charges $0.005/minute for inbound traffic to VoxEngine and $0.015/minute for PSTN calls out of VoxEngine. This gives their Dialogflow Connector a 32% discount vs. the call forwarding approach on their same platform. This is also a slight price advantage over Dialogflow's own Phone Gateway with the Enterprise Essentials for more features and connectivity options.

US phone numbers are $1.00/month.

While one can use the Dialogflow Phone Gateway for free, the analysis below assumes production users would pay Google for one of its Enterprise plans for an unlimited quota with an SLA.

Call Forwarding to Dialogflow's Phone Gateway

Line Item	Price
Voximplant Inbound (US) (origination)	0.0050
Voximplant Outbound (US) (termination)	0.0150
Dialogflow Phone Gateway (Enterprise Essentials)	0.0500
TOTAL PRICE PER MINUTE	$ 0.0700

Voximplant's Dialogflow Connector

Line Item	Price
Voximplant Inbound (US) (origination)	0.0050
Voximplant Dialogflow Connector	0.0168
Dialogflow Audio (Enterprise Essentials)	0.0260
TOTAL PRICE PER MINUTE	$ 0.0478

Dialogflow Connector Advantage	**$ 0.0222** **31.7%**

Features

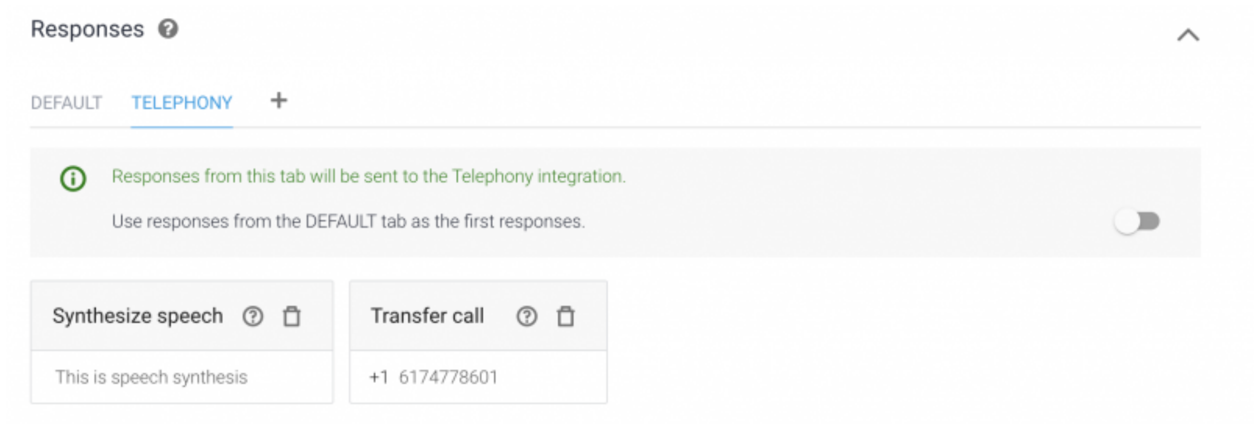
As a mature solution with a gRPC interface, Voximplant provides good coverage of nearly all of the key telephony and Dialogflow interaction features identified earlier.

The sections below provide a high-level review of these features. Developers can refer to this gist for cogint.ai's sample of working VoxEngine code implementing these features.

Telephony Interaction

Call transfer

The bottom part of Voximplant's template code provides examples for handling telephony responses from Dialogflow including call transfer, speech synthesis, and playback of an audio file - all the options available in Dialogflow's Telephony menu. Basically, all you need to do is set your Telephony options there, just like you would do if you were using Dialogflow's Phone Gateway. This also makes it easy to switch from using Dialogflow's Phone Gateway to Voximplant since none of the setup inside the bot for telephony will be lost.



Responses from this tab will be sent to the Telephony integration.

Use responses from the DEFAULT tab as the first responses.

Synthesize speech: This is speech synthesis

Transfer call: +1 6174778601

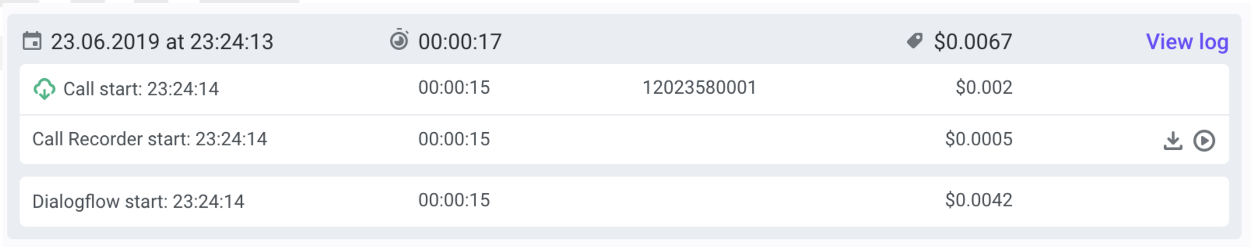
Figure 8 - Dialogflow's built-in telephony options all work with Voximplant

Just uncomment the transfer code and enter one of your phone numbers to dial from:

```
function processTelephonyMessage(msg) {
  // Transfer call to msg.telephonyTransferCall.phoneNumber
  if (msg.telephonyTransferCall !== undefined) {
    dialogflow.stop()
    let newcall = VoxEngine.callPSTN(msg.telephonyTransferCall.phoneNumber,
VOICEBOT_PHONE_NUMBER)
    VoxEngine.easyProcess(call, newcall)
  }
}
```

Recording

Call recording is trivial to do with the `call.record` function. The option to record in stereo is also available – this simplifies debugging since the caller and Dialogflow agent are on separate audio channels.



23.06.2019 at 23:24:13	00:00:17		$0.0067	View log
Call start: 23:24:14	00:00:15	12023580001	$0.002	
Call Recorder start: 23:24:14	00:00:15		$0.0005	
Dialogflow start: 23:24:14	00:00:15		$0.0042	

Figure 9 - Example from Voximplant's call history showing charges including recording

Voximplant lets you put the call recording in the call history tab which is useful for debugging. The recording URL returned by the `CallEvents.RecordStopped` event can be used to transfer the file elsewhere if needed. Voximplant charges $0.0005 / min for standard audio and $0.0030 /min for HD audio recordings. The base charge includes three months of storage which can be extended for additional charges.

DTMF detection

Voximplant has the ability to detect DTMF, but it is up to the developer to send this as an event and handle it inside Dialogflow. Fortunately, this is relatively easy to do.

Voximplant's `call` class has the ability to receive events on DTMF tones. It takes about seven lines of code to pass a single DTMF digit to Dialogflow as an event as each key is pressed. The logic to gather several digits and send them as a single event – i.e. "1234" instead of "1", "2", "3", and "4" would take a few more lines of code depending on your application needs.

You will need to create an event to handle this input and an entity to handle the parameters inside Dialogflow.

SMS

While Voximplant does support SMS messaging, VoxEngine currently has no SMS APIs. It is possible to setup your own server and use webhooks to handle this interaction, but this would require significant development effort and server infrastructure outside of VoxEngine.

Dialogflow Interaction

Telephony tab support

As illustrated in the call transfer section above, Voximplant's default `processTelephonyMessage` code includes support for items entered into

Dialogflow's Telephony Tab. In addition to call transfer, the speech synthesis and media playback GUI elements in that tab are also supported.

```
// Process telephony messages from Dialogflow
function processTelephonyMessage(msg) {
  // Transfer call to msg.telephonyTransferCall.phoneNumber
  if (msg.telephonyTransferCall !== undefined) {
    dialogflow.stop()
    let newcall = VoxEngine.callPSTN(msg.telephonyTransferCall.phoneNumber,
VOICEBOT_PHONE_NUMBER)
    VoxEngine.easyProcess(call, newcall)
  }
  // Synthesize speech from msg.telephonySynthesizeSpeech
  if (msg.telephonySynthesizeSpeech !== undefined) {
    // See the list of available TTS languages at
https://voximplant.com/docs/references/voxengine/language
    if(msg.telephonySynthesizeSpeech.ssml !== undefined)
      call.say(msg.telephonySynthesizeSpeech.ssml, Language.Premium.AU_ENGLISH_MALE)
    else if (msg.telephonySynthesizeSpeech.text !== undefined)
      call.say(msg.telephonySynthesizeSpeech.text, Language.Premium.AU_ENGLISH_MALE)
  }
  // Play audio file located at msg.telephonyPlayAudio.audioUri
  if (msg.telephonyPlayAudio !== undefined) {
    // audioUri contains Google Storage URI (gs://), we need to transform it to URL (https://)
    let url = msg.telephonyPlayAudio.audioUri.replace("gs://",
"https://storage.cloud.google.com/")
    call.startPlayback(url)
  }
}
```

Custom events and contexts

Voximplant supports events and contexts – mechanisms that Dialogflow uses to control the bot outside of direct user input.

Events

In addition to handling text and speech-based inputs to initiate an intent, Dialogflow can also handle incoming events. Using an event allows you to immediately and definitively invoke an intent without machine learning trying to match an utterance from a list of intents.

Voximplant can be used to send a `TELEPHONY_WELCOME` event to Dialogflow after connecting:

```
    // Send a WELCOME event
    dialogflow.sendQuery({event : {name: "TELEPHONY_WELCOME", language_code: "en"}})
```

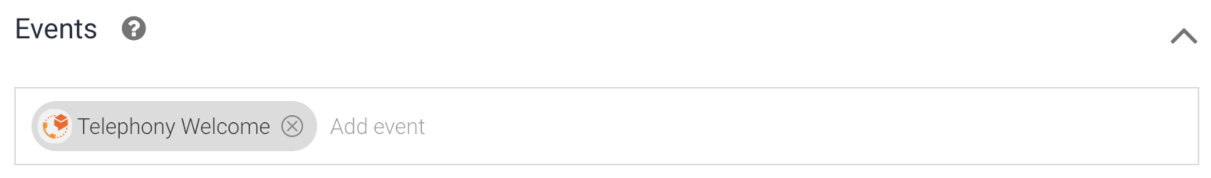


Figure 10 - Telephony Welcome event example inside Dialogflow's console interface

Contexts

In addition, Dialogflow also has a construct known as contexts that allows stateful information to be shared across intents during a bot session. Contexts can also be used as filters when you might want to have different responses for the same intent in different scenarios - like responding differently on a voice call vs. a text message.

As an example, in a customer service environment it is common for the IVR or agent to ask for a callback number in case the call is disconnected. If the bot had the customer's phone number from the caller ID, then it could simply ask the user to verify if that is a good number to return the call. We could implement this in Dialogflow with a `phone` context that keeps the user's caller ID and phone number. VoxEngine has a `dialogflow.setQueryParameters` method for this. To do this we just added the following instead of the above code:

```
// Set a phone context with phone parameters
// ToDo: error handling if returned parameters are null?
phoneContext = {
    name: "phone",
    lifespanCount: 99,
    parameters: {
        caller_id: call.callerid(),
        called_number: call.number()
    }
}

dialogflow.setQueryParameters({contexts: [phoneContext]})

// Send a WELCOME
dialogflow.sendQuery({event : {name: "WELCOME", language_code: "en"}})
```

Playback interruption

Humans in real conversations interrupt each other. Ideally the voicebot could speak and listen at the same time and even interrupt its own responses if needed. With Dialogflow's query API, you send an utterance and it returns a response. Sending several queries back to back is fine, but it does not make sense to playback multiple responses simultaneously over the top of each other. To avoid this many voicebots take a sequential approach:

1. Stream audio to Dialogflow, which listens for an utterance
2. When Dialogflow hears an utterance, it sends a response to the gateway
3. The gateway then stops listening for new audio
4. The gateway plays back the audio provided by Dialogflow (or synthesizes speech from the returned text)
5. Finally, the gateway starts listening again, starting the cycle over again

Voximplant does not have a perfect solution for playback interruption, but they do have a playback marker concept that lets you tell the gateway when to start listening again after playback. Voximplant actually lets you set this to a negative value, so it starts listening again before the playback of the previous intent response has finished playback. If you set this timer appropriately, you can allow some ability to interrupt without worrying about overlapping playback.

Voximplant has a single line of code for this:

```
// Playback marker used for better user experience
dialogflow.addMarker(-300)
```

`addMarker` could potentially be adjusted based on an individual query response parameter or context if you added additional code to look for that.

No activity detection

If the bot does not hear anything it should check if the user is still there and hang-up if needed to avoid the costs of staying on a dead call indefinitely. Voximplant does not have a native function for this, but the behavior can be replicated with some code in VoxEngine. To do this, you can create a timer function inside VoxEngine that sends a `NO_INPUT` event to Dialogflow. Then an intent will need to be created on the Dialogflow side that includes this event and says something like “Are you still there?”

An sample implementation of this for VoxEngine can be seen here: https://cogint.ai/voximplant-dialogflow-connector-2019/#noactivitydetection

COGINT.AI

Feature Compliance Summary

Dialogflow's Phone Gateway is easy to use but is very limited in its capabilities. The forwarding approach is better but has its own drawbacks and requires significant development for features beyond call transfer and recording. Voximplant's Dialogflow Connector provides a more efficient direct connectivity alternative for rapid integration and a deep telephony-to-bot integration with Dialogflow.

Many Dialogflow telephony integration features can be implemented in seconds to minutes with alterations to the provided template. Other features will require more development, but as some of the code samples referenced in this paper illustrate, many of these features can be developed and customized within hours if needed.

Requirement	**Dialogflow Phone Gateway**	**Forwarding to Phone Gateway**	**CPaaS - VoxImplant**
Telephony Interaction			
Call transfer	◑	◕	●
DTMF detection	○	◔	◑
Recording	○	◕	◕
SMS	○	◔	◔
Dialogflow Interaction			
Playback interruption	○	◔	◕
No activity detection	○	◔	◑
Custom events & contexts	○	◔	◕
Telephony tab support	○	◔	●

○ Not possible	◔ Possible with major effort	◑ Supported with effort	◕ Minimal development	● No dev needed

About the Author

Chad Wallace Hart is an analyst and consultant at cwh.consulting, a product management, marketing, and strategy advisory helping to advance the communications industry. Chad's recent experience and projects include authoring an extensive report on the applications of Artificial Intelligence in Real Time Communications, managing a new product incubator program, launching a WebRTC startup, product marketing, and product ownership of various IP communications, WebRTC, and CPaaS offers. Chad is a frequent speaker, blogger/editor at webrtcHacks.com and cogint.ai, and event organizer with Kranky Geek and WebRTC Boston.

About the Sponsor

Voximplant is a serverless communications platform that helps companies develop and implement real-time voice, video, and messaging solutions. The company allows developers to quickly build applications by taking care of the complex infrastructure and technologies that are all too common when adding audio and visual communication into applications. Voximplant customers include businesses of all sizes from start-ups to major brands including Hyundai, Burger King, and Sberbank, one of the largest banks in Europe.

Glossary

Term	Definition
ASR	Automatic Speech Recognition – see STT
Bot	An autonomous program that can mimics human interaction with users
Conversational IVR	An IVR system that offers the ability for users to interact with it via voice in a conversational manner
CPaaS	Communications Platform as a Service – a cloud-based service that exposes Real Time Communications APIs for developers
Deep Learning	A modern form of Machine Learning that leverages a multi-layered artificial neural network topology
Dialogflow Event	Dialogflow allows developers to invoke intents based on something that has happened instead of a direct user communication
Dialogflow Intent	A software construct used to define user intentions that can be used to initiate a programmatic response or action
DTMF	Dual Tone Multi Frequency – the touchtone key tones that are produced by a telephone keypad
gRPC	General Remote Procedure Call – an modern open source client-server protocol is designed to run in any distributed computing environment
IVR	Interactive Voice Response – a program that allows a user to interact using DTMF or voice over the telephone
NLU	Natural Language Understanding - a branch of Artificial Intelligence focused on programmatically handling human language inputs
PSTN	Public Switched Telephone Network – the global telephone network used for placing and receiving phone calls
RTC-Bot Gateway	Controls, converts, and connects the signaling and media of the telephony networks to a format that can be used by bot software
Serverless	Cloud computing method that automatically allocates server resources so the user only needs to consider their software functions
SMS	Short Message System – telephone network-based text messaging system
STT	Speech-to-Text – software transcription of speech into text
TTS	Text-to-Speech – software synthesis of speech based on text input
Voicebot	Autonomous bot programs that leverage STT, NLU, and TTS to interact with humans via speech
VoIP	Voice over Internet Protocol - a computing system that provides voice and sometimes video telephony service over the Internet