Sep 27, 2017 · 6 min read

How to Integrate API.AI with Chatfuel

When most people think of chatbots, they envision having a helpful, uninterrupted conversation with a flawless virtual assistant.

Unfortunately, this just isn't the case. The reality is that chatbots are only as good as their Natural Language Processing (NLP).

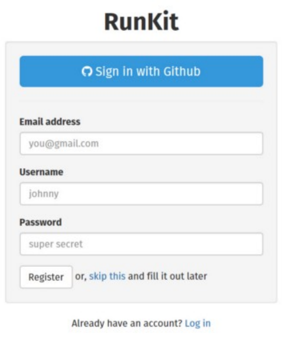
I've built hundreds of menu-based bots, but—despite explicit instructions to do otherwise—users will continue typing messages instead of pressing buttons! This leads to error messages and frustrated users…

Humans will always be humans, so the next best solution is harnessing the power of a robust Artificial Intelligence (AI) engine.

In this tutorial, I'll integrate Google's API.AI platform with Chatfuel.

(NOTE: Chatfuel *does* have its own AI engine. However, if you want to build more advanced AI responses, using a third-party service like API.AI will help.)

Step #1: Create a Runkit account


RunKit

Sign in with Github

Email address: you@gmail.com

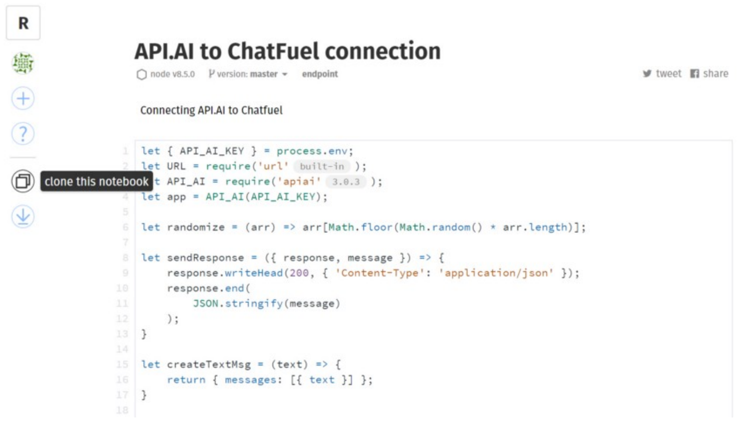
Username: johnny

Password: super secret

Register or, skip this and fill it out later

Already have an account? Log in

After registering your account, clone Edwin's Runkit notebook.



API.AI to ChatFuel connection

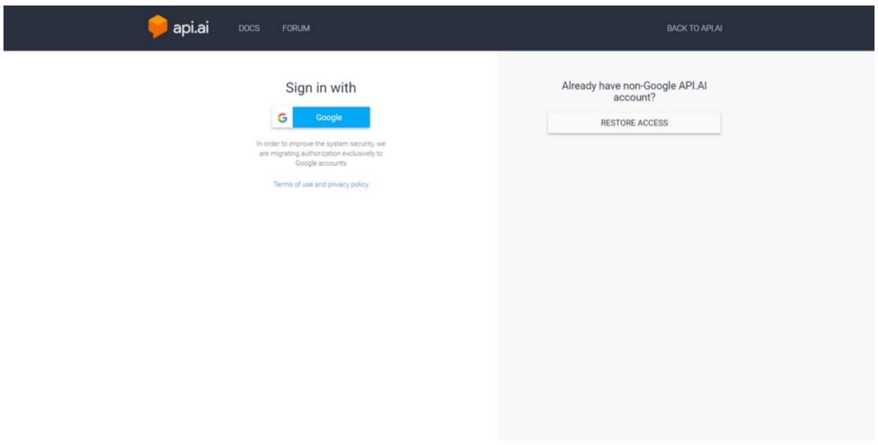
node v8.5.0 version: master endpoint

Connecting API.AI to Chatfuel

```
let { API_AI_KEY } = process.env;
let URL = require('url' built-in );
t API_AI = require('apiai' 3.0.3 );
let app = API_AI(API_AI_KEY);

let randomize = (arr) => arr[Math.floor(Math.random() * arr.length)];

let sendResponse = ({ response, message }) => {
    response.writeHead(200, { 'Content-Type': 'application/json' });
    response.end(
        JSON.stringify(message)
    );
}

let createTextMsg = (text) => {
    return { messages: [{ text }] };
}

```

Step #2: Log in to API.AI

Click here to sign in with an existing Gmail account.



Sign in with

Google

In order to improve the system security, we are migrating authorization exclusively to Google accounts

Terms of use and privacy policy

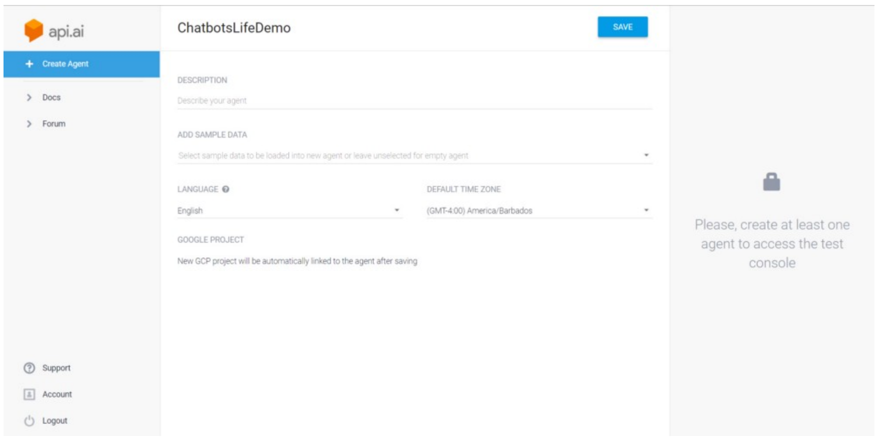
Already have non-Google API.AI account?

RESTORE ACCESS

Step #3: Create an agent

After logging in to API.AI, you'll need to create a new agent. This agent contains the NLP framework that will ultimately communicate with your bot's users via Chatfuel.

Name your agent, then click "Save".



ChatbotsLifeDemo

DESCRIPTION: Describe your agent

ADD SAMPLE DATA: Select sample data to be loaded into new agent or leave unselected for empty agent

LANGUAGE: English

DEFAULT TIME ZONE: (GMT-4:00) America/Barbados

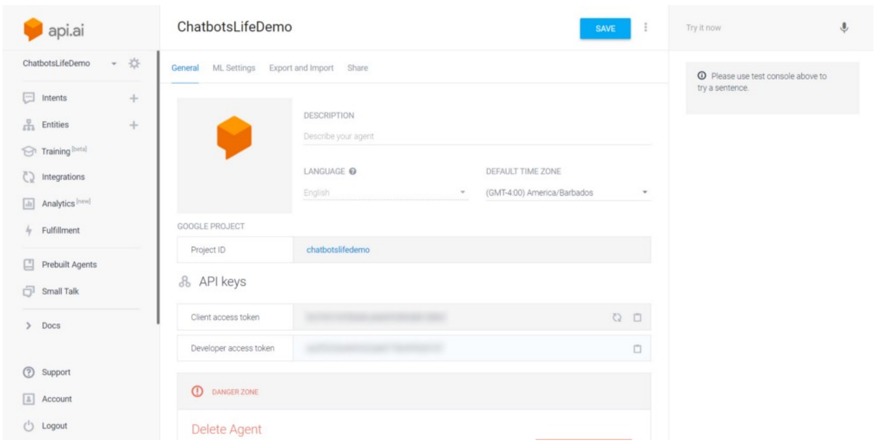
GOOGLE PROJECT: New GCP project will be automatically linked to the agent after saving

Please, create at least one agent to access the test console

Step #4: Connect the agent to Runkit

Now click the gear icon next to the name of your new agent.

Copy the CLIENT ACCESS TOKEN under 'API Keys'.



ChatbotsLifeDemo

DESCRIPTION: Describe your agent

LANGUAGE: English

DEFAULT TIME ZONE: (GMT-4:00) America/Barbados

GOOGLE PROJECT

Project ID: chatbotslifedemo

API keys

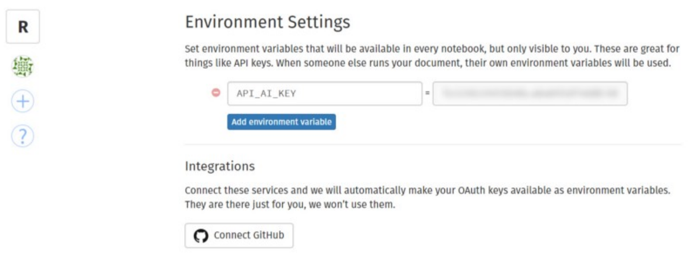
Client access token

Developer access token

Please use test console above to try a sentence.

While logged in to Runkit, edit your 'Environment Settings' to include the token from API.AI.

To do this, add a new 'Environment Variable'. Then replace FOO with API_AI_KEY, and BAR with the CLIENT ACCESS TOKEN you just copied.



Environment Settings

Set environment variables that will be available in every notebook, but only visible to you. These are great for things like API keys. When someone else runs your document, their own environment variables will be used.

API_AI_KEY = [redacted]

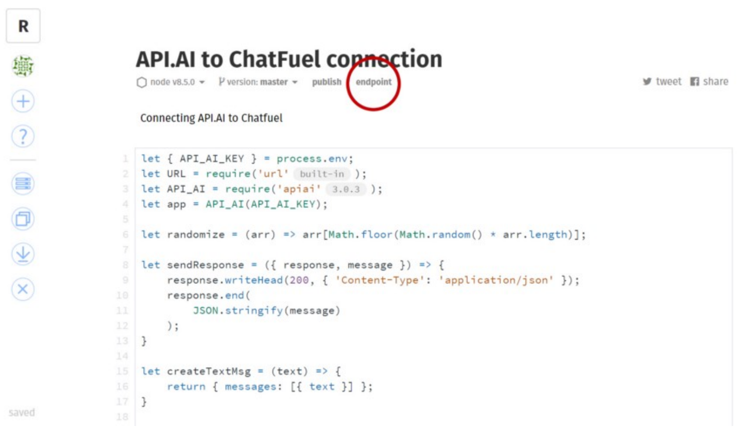
Integrations

Connect these services and we will automatically make your OAuth keys available as environment variables. They are there just for you, we won't use them.

Step #5: Grab the endpoint URL

Next, open the Runkit notebook you copied in Step #1.

Click "endpoint," which will open a new page that displays the JSON code needed for Chatfuel.



API.AI to ChatFuel connection

Connecting API.AI to Chatfuel

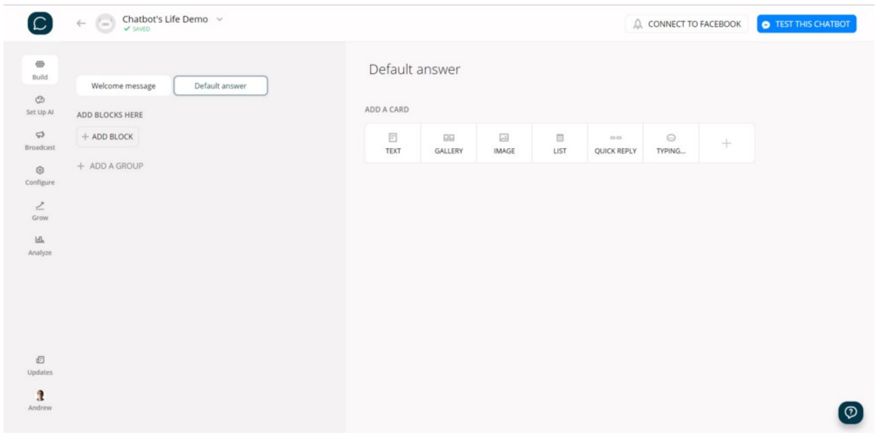
```
let { API_AI_KEY } = process.env;
let URL = require('url' built-in );
let API_AI = require('apiai' 3.0.3 );
let app = API_AI(API_AI_KEY);

let randomize = (arr) => arr[Math.floor(Math.random() * arr.length)];

let sendResponse = ({ response, message }) => {
    response.writeHead(200, { 'Content-Type': 'application/json' });
    response.end(
        JSON.stringify(message)
    );
}

let createTextMsg = (text) => {
    return { messages: [{ text }] };
}
```

Copy the URL of this page. It should be formatted as… *untitled-xxxxxxxxxxxxx-runkit.sh*

Step #6: Connect Runkit to Chatfuel

Now that your JSON code is ready in Runkit, you'll need to loop Chatfuel into the workflow.

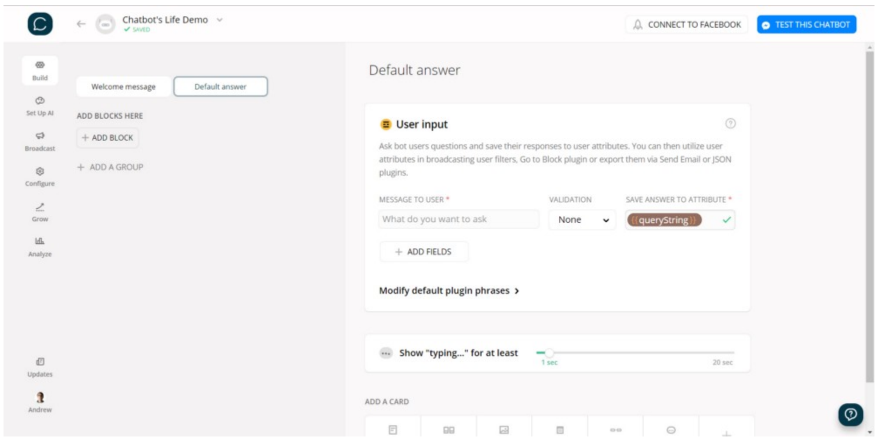
First, create a new bot in Chatfuel (or use your existing chatbot) and navigate to the Default Answer block.



Then add a USER INPUT plugin, leaving blank the 'Message to User' field.

Set 'Save Answer to Attribute' as {{queryString}}.

It's best practice to include a typing animation so that your bot is more conversational, too.



User input

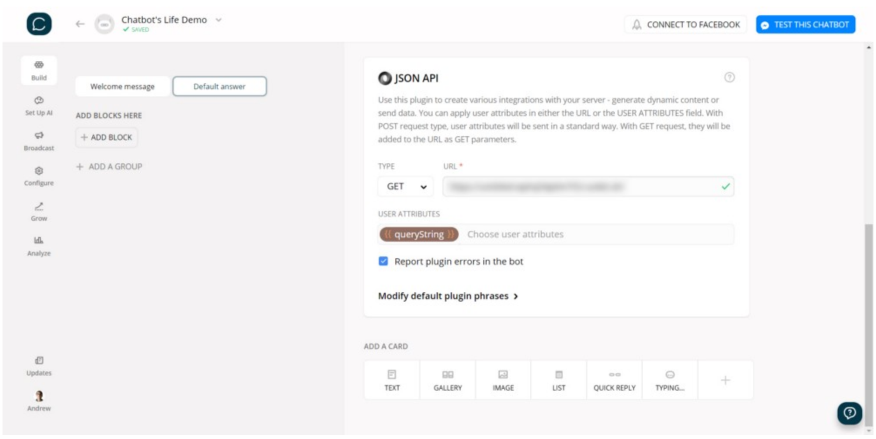
Ask bot users questions and save their responses to user attributes. You can then utilize user attributes in broadcasting user filters, Go to Block plugin or export them via Send Email or JSON plugins.

MESSAGE TO USER *	VALIDATION	SAVE ANSWER TO ATTRIBUTE *
What do you want to ask	None	{{queryString}}

Show "typing..." for at least 1 sec 20 sec

Finally, add a JSON API plugin to your Default Answer.

Paste the endpoint URL you copied from Runkit in Step #5, and populate the {{queryString}} user attribute under USER ATTRIBUTES.



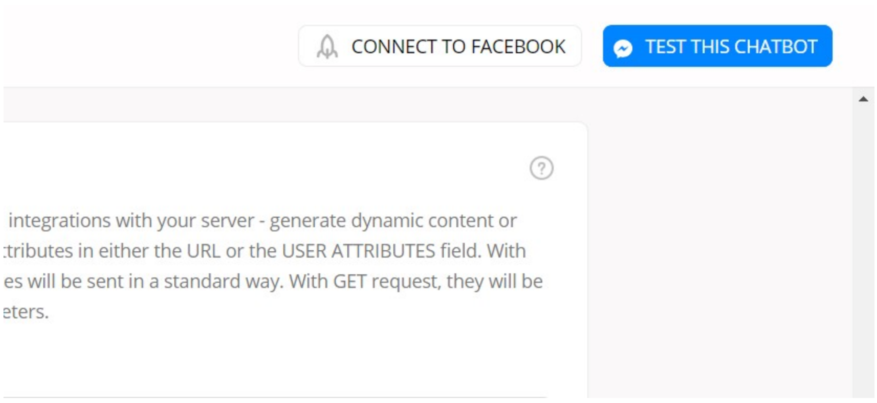
JSON API

Use this plugin to create various integrations with your server - generate dynamic content or send data. You can apply user attributes in either the URL or the USER ATTRIBUTES field. With POST request type, user attributes will be sent in a standard way. With GET request, they will be added to the URL as GET parameters.

TYPE: GET
URL *
USER ATTRIBUTES: {{ queryString }} Choose user attributes
☑ Report plugin errors in the bot
Modify default plugin phrases ›

Step #7: Test via Messenger

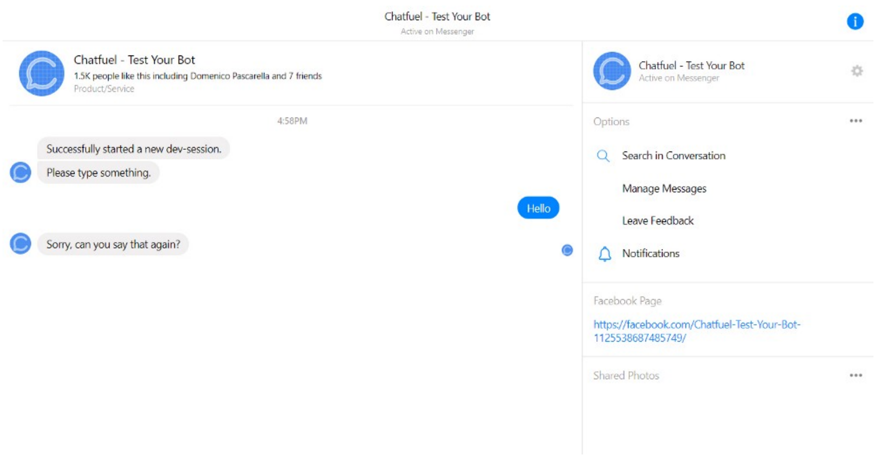
To confirm your workflow is set up correctly, click the TEST THIS CHATBOT button in Chatfuel.



integrations with your server - generate dynamic content or
tributes in either the URL or the USER ATTRIBUTES field. With
es will be sent in a standard way. With GET request, they will be
eters.

Send any response to trigger the Default Answer block.

Your query, per our JSON API workflow in Chatfuel, will be pushed to API.AI (where the text input will be processed using NLP) and then returned to the user via Messenger.



Chatfuel - Test Your Bot
Active on Messenger

Chatfuel - Test Your Bot
1.5K people like this including Domenico Pascarella and 7 friends
Product/Service

4:58PM

Successfully started a new dev-session.
Please type something.

Hello

Sorry, can you say that again?

If your setup is successful, the bot will respond with some version of: *“I didn’t understand you.”*

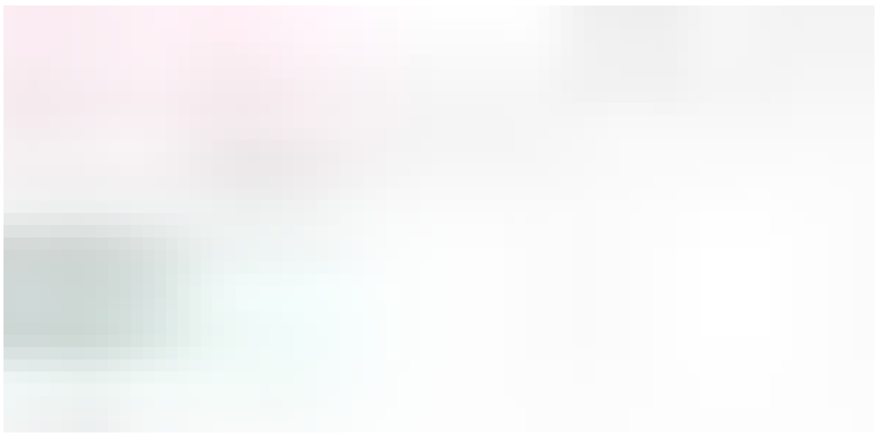
This is normal!

Step #8: Set up “Small Talk”

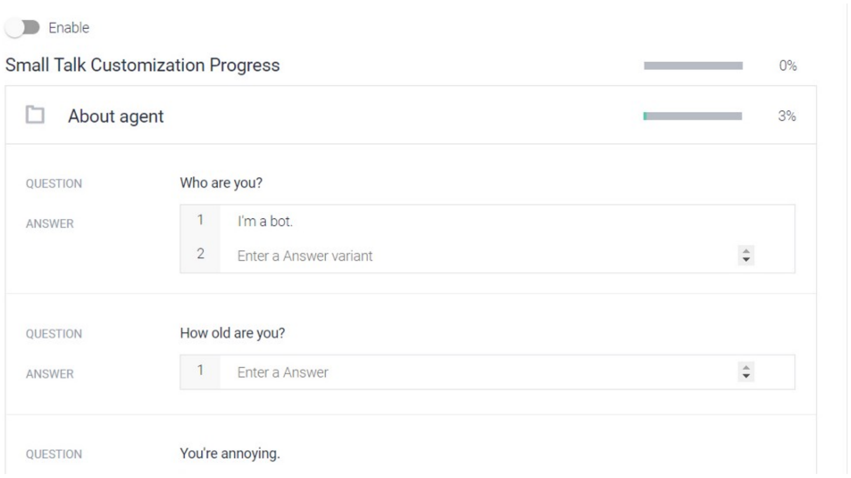
At this point, all the bot can do is randomize error messages. That’s helpful, but let’s make it much better!

API.AI’s “Small Talk” feature will enable users to have simple conversations with the chatbot.

To get started, click “Small Talk” on the left sidebar of API.AI.

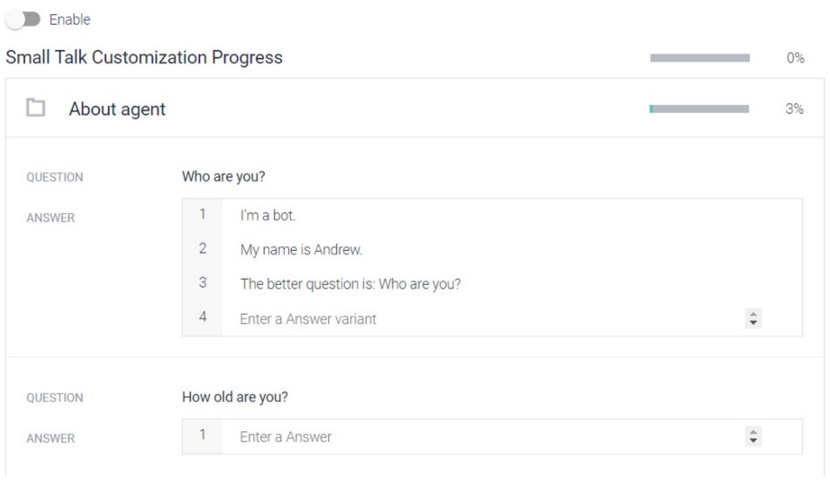


Answer any/all of the questions in each category to enhance your chatbot’s NLP capabilities.



QUESTION	Who are you?
ANSWER	1 I'm a bot.
	2 Enter a Answer variant
QUESTION	How old are you?
ANSWER	1 Enter a Answer
QUESTION	You're annoying.

To add variations of answer, which API.AI will randomize for you, enter an “answer variant”.


Enable

Small Talk Customization Progress 0%

About agent 3%

QUESTION Who are you?

ANSWER
1 I'm a bot.
2 My name is Andrew.
3 The better question is: Who are you?
4 Enter a Answer variant

QUESTION How old are you?

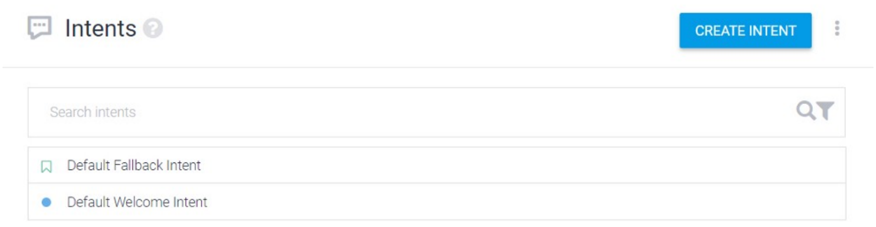
ANSWER
1 Enter a Answer

Click ENABLE and “Save” upon completion.

Step #9: Set up custom responses

If API.AI’s predefined “Small Talk” doesn’t cut it for your use case, “Intents” are the solution!

Click INTENTS on the left sidebar, then click ‘Create Intent’.


Intents

CREATE INTENT

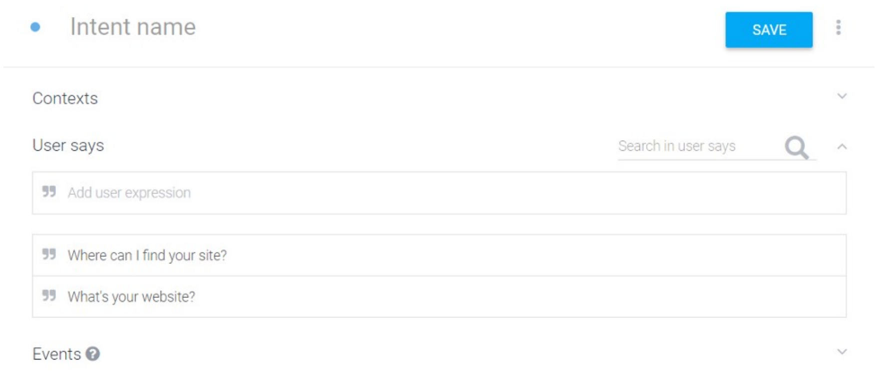
Search intents

Default Fallback Intent

Default Welcome Intent

Under ‘User Says,’ predict the user’s input.

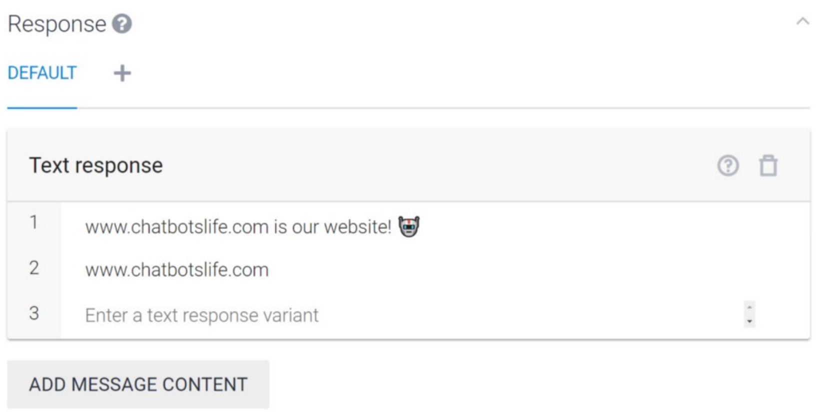
(For example, “What’s your website?”)



Where can I find your site?

What's your website?

Next, scroll down to 'Response' and enter the message you want returned to the user.



1 www.chatbotslife.com is our website! 🐻

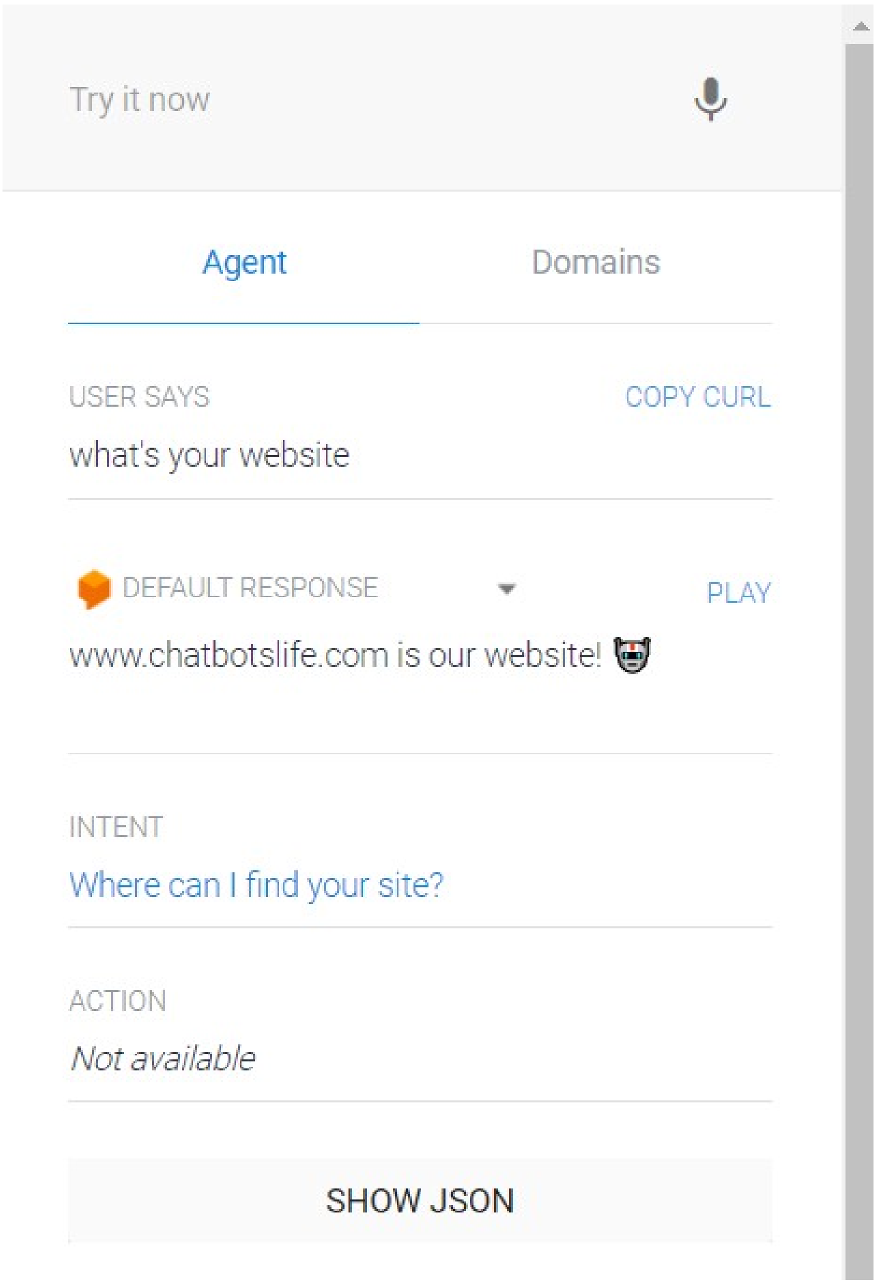
2 www.chatbotslife.com

3 Enter a text response variant

When you're ready for testing, click "Save".

Then use API.AI's chat console to verify that a user's input will trigger your expected response.

If your expected response appears under DEFAULT RESPONSE, it worked!


Try it now

Agent Domains

USER SAYS COPY CURL

what's your website

DEFAULT RESPONSE PLAY

www.chatbotslife.com is our website!

INTENT

Where can I find your site?

ACTION

Not available

SHOW JSON

If not, modify your predicted inputs under 'User Says'.

Step #10 (optional): Respond with rich elements

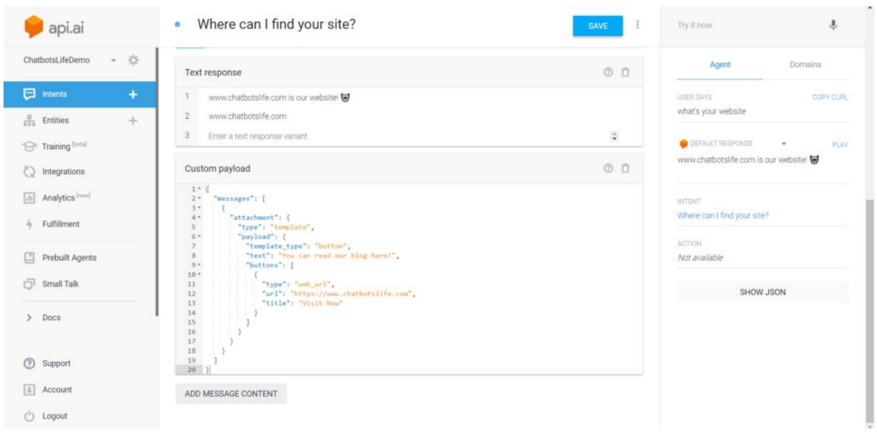
The beauty of integrating API.AI with Chatfuel is that you can send custom payloads to users.

In other words, your chatbot can respond not only with text, but also with rich elements (audio, images, videos, buttons, and carousels—oh my!).

These features are designed to enhance user experience.

To add them, click ADD MESSAGE CONTENT and use CUSTOM PAYLOAD.

Next, use Chatfuel's JSON API documentation to properly format your payload in API.AI.



Where can I find your site?

Text response

1 www.chatbotslife.com is our website! 😈
2 www.chatbotslife.com
3 Enter a text response variant

Custom payload

```
{
  "messages": [
    {
      "attachment": {
        "type": "template",
        "payload": {
          "template_type": "button",
          "text": "You can read our blog here!",
          "buttons": [
            {
              "type": "web_url",
              "url": "https://www.chatbotslife.com",
              "title": "Visit Now"
            }
          ]
        }
      }
    }
  ]
}
```

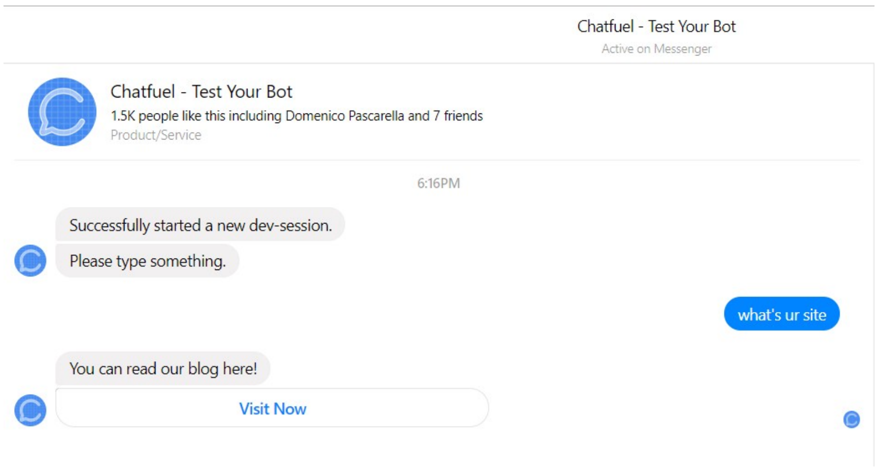
USER SAYS
what's your website

DEFAULT RESPONSE
www.chatbotslife.com is our website! 😈

INTENT
Where can I find your site?

ACTION
Not available

Now—instead of responding to users with an unclickable URL—we can return a more functional and visually appealing response by using a button.



Chatfuel - Test Your Bot
Active on Messenger

Chatfuel - Test Your Bot
1.5K people like this including Domenico Pascarella and 7 friends
Product/Service

6:16PM

Successfully started a new dev-session.

Please type something.

what's ur site

You can read our blog here!

Visit Now

Conclusion

We've covered a lot of ground in this tutorial!

I've shown you how to…
• Clone a Runkit notebook
• Create an API.AI agent
• Edit your Environment Settings
• Set up the JSON API in Chatfuel
• Initiate "Small Talk" and custom responses
• Respond with rich elements using custom payloads

Ultimately, the benefit of integrating with Google's API.AI engine is for your chatbot to better understand your users.

After all, what good is a dumb bot?

Chat soon,
Andrew Demeter

(HUGE thanks to Edwin Reynoso, who generously provided the code and technical insight for this tutorial!)