

**NOTE: This legacy documentation was last updated in June 2020.
It is no longer maintained and may be outdated.**

Actions on Google integration

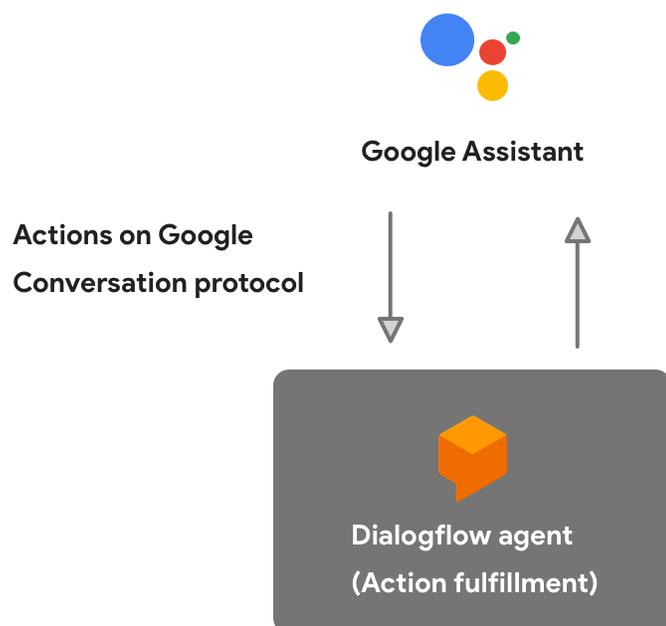
The Actions on Google integration for Dialogflow lets you build conversational fulfillment for Actions by handling communication between your Dialogflow agent and the Google Assistant using the Conversation protocol, which defines a standard way for conversational fulfillment to communicate with the Assistant. The integration lets you use Dialogflow's easy-to-use IDE and best-in-class NLU to build and deploy Actions without having to parse requests and generate responses in the Conversation protocol. In fact, you can build fulfillment for simple Actions completely in Dialogflow, without any code.

The following sections describe how fulfillment works and how Actions on Google intents map to Dialogflow intents when enabling the Actions on Google integration.

Fulfillment

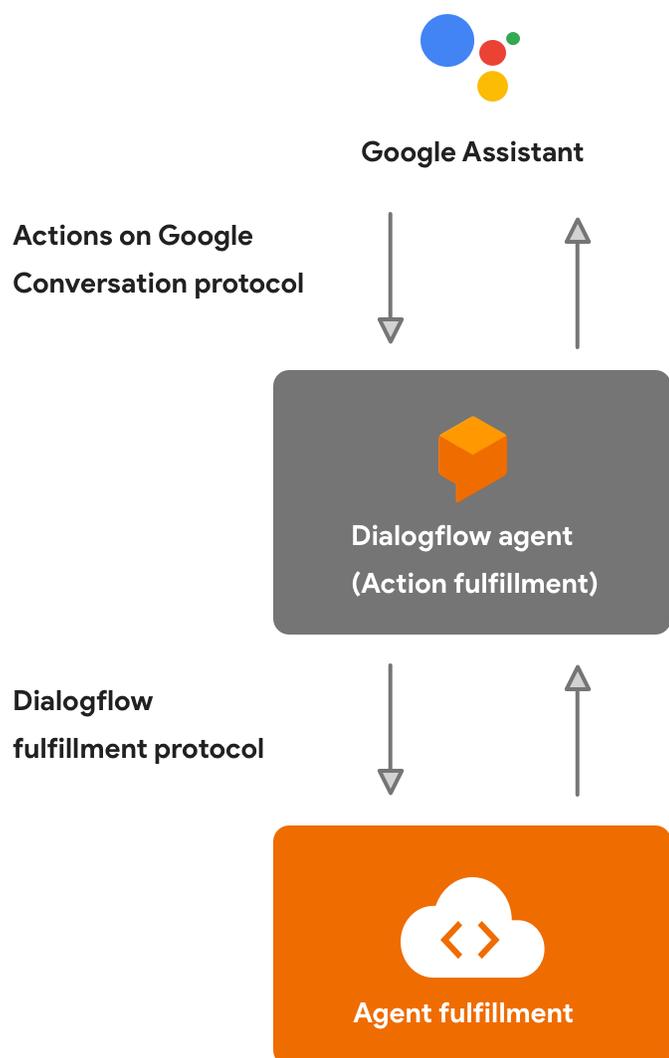
Your Dialogflow agent acts as conversational fulfillment for your Actions. In addition, your actual agent can also call it's own fulfillment to further customize logic for your Actions.

Action fulfillment



When the Assistant invokes your agent, it acts as conversational fulfillment for your Action. The Assistant sends user utterances to your Dialogflow agent to match to an intent and respond back. Your agent matches the utterance to an intent and sends a response. The Assistant renders this response to the user, displaying it appropriately depending on the user's device capabilities (audio and display output).

Dialogflow agent fulfillment



In addition, your Dialogflow agent can call on its own fulfillment (deployed as a webhook) to carry out some logic like calling a REST API or some other backend service and then generate a response to return to the Assistant. In this case, your webhook generates the response using the Dialogflow webhook format. Your agent then translates this response to the Actions on Google conversation protocol so the Assistant can render the response to the user.

You can turn on fulfillment for individual Dialogflow intents, which allows you to specify which intents require extra processing and which can just return responses defined within the Dialogflow console.

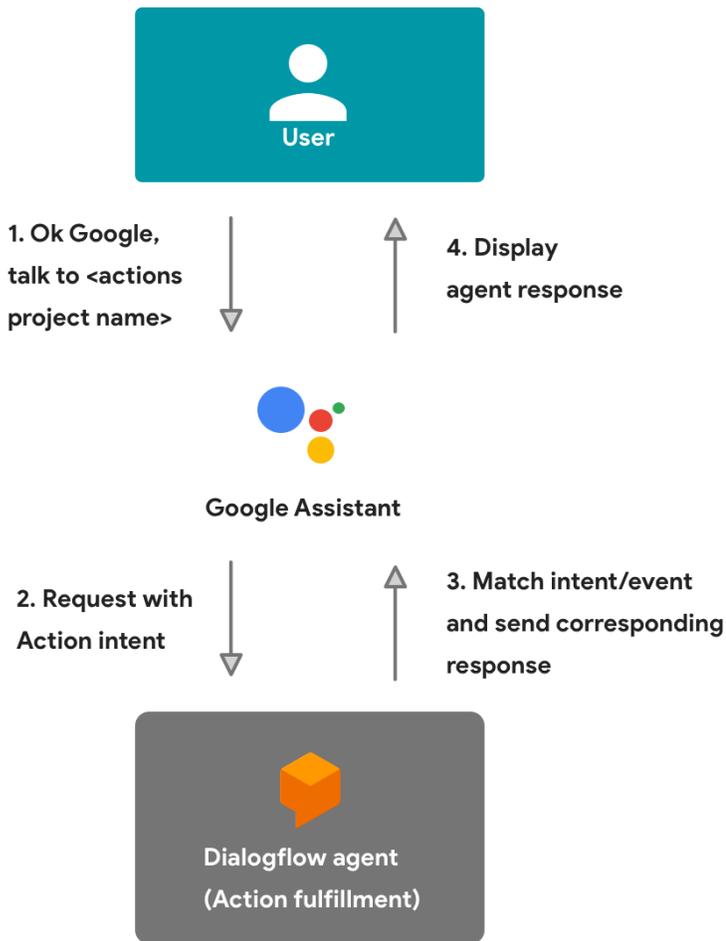
Intents

Intents are used differently in Actions on Google than in Dialogflow, but there is a consistent mapping between the two platforms' intents to allow interoperability. The Conversation protocol requires the Assistant to provide an intent in every request to your Dialogflow agent, along with other data such as the user utterance, so that your Dialogflow agent knows how to respond back. Your Dialogflow agent receives the Actions on Google intent, matches it to one of its own intents, and responds back.

Your Dialogflow agent maps one of its own intents to one of these general types of Actions on Google intents:

- **Action** - Represent tasks that the user wants to carry out. When your agent receives an Action intent, this means that the Assistant wants your agent to fulfill the user's request for an Action and is invoking the agent.
- **Dialog** - Represents a back and forth turn between the Assistant and your fulfillment. The Assistant sends requests with the dialog intent (`actions.intent.TEXT`) to your agent when it receives a user utterance that it wants your agent to process. When your agent receives a dialog intent, it finds a matching Dialogflow intent and responds back accordingly. This happens repeatedly until your agent has enough information to fulfill the Action.
- **Helper** - Represents a dialog that you want the Assistant to carry out on your behalf, such as obtaining the user's email or location. Your agent sends a response to the Assistant with a helper intent to carry out a dialog. The Assistant then returns the result of the helper back to your agent.

Action intents

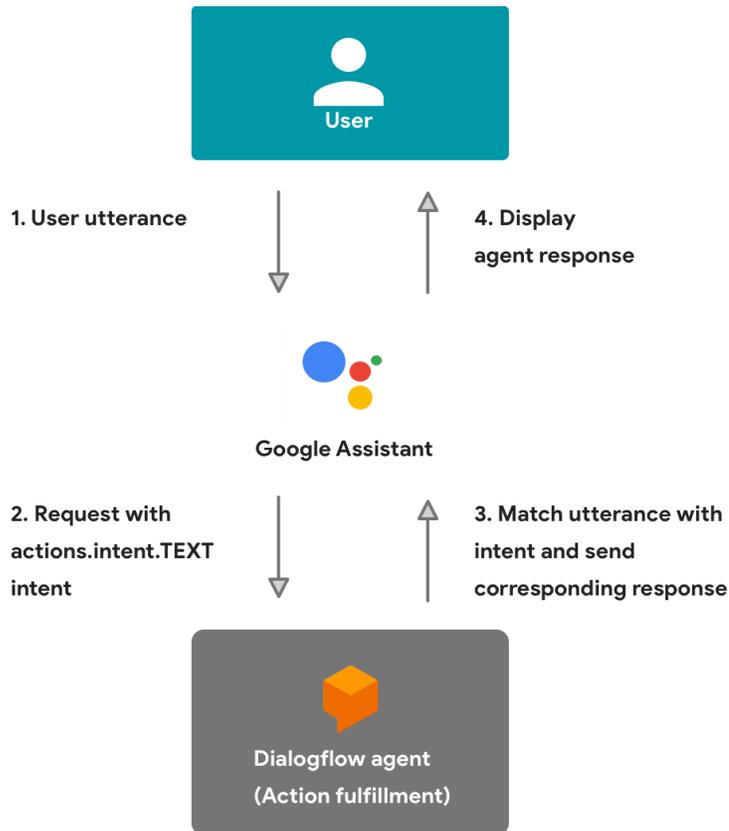


Action intents represent something users want to get done and are passed to your Dialogflow agent when an Action is invoked. Your agent then carries out a conversation to fulfill the Action. There are three types of Action intents:

- **Main intent** - This intent is named `actions.intent.MAIN` and represents the default Action for every Actions project. Your agent must specify one and only one intent as your main intent. When users say the invocation name that is associated with your Actions project, the Assistant makes an HTTP request to the corresponding fulfillment, providing this intent in the request, and hands off the conversation to your Dialogflow agent.
- **Built-in intents** - Google defines these intents and the user utterances that trigger them. You can opt into supporting these intents to support standard invocations such as *"play a game"*. For more information, see the built-in intents in Actions on Google documentation.
- **Custom intents** - You define these intents and the user utterances that trigger them. These intents represent custom Actions that you create. To define a custom action, you specify the following components:
 - Query patterns- that describe what user utterances can trigger this intent.
 - Parameters that you want to parse from the user utterances.
 - A unique name for an intent that typically follows reverse domain name notation, followed by the name of the intent in all caps. For example, `com.mybrand.MY_INTENT`.

For more information on how to specify Actions, see the Build documentation.

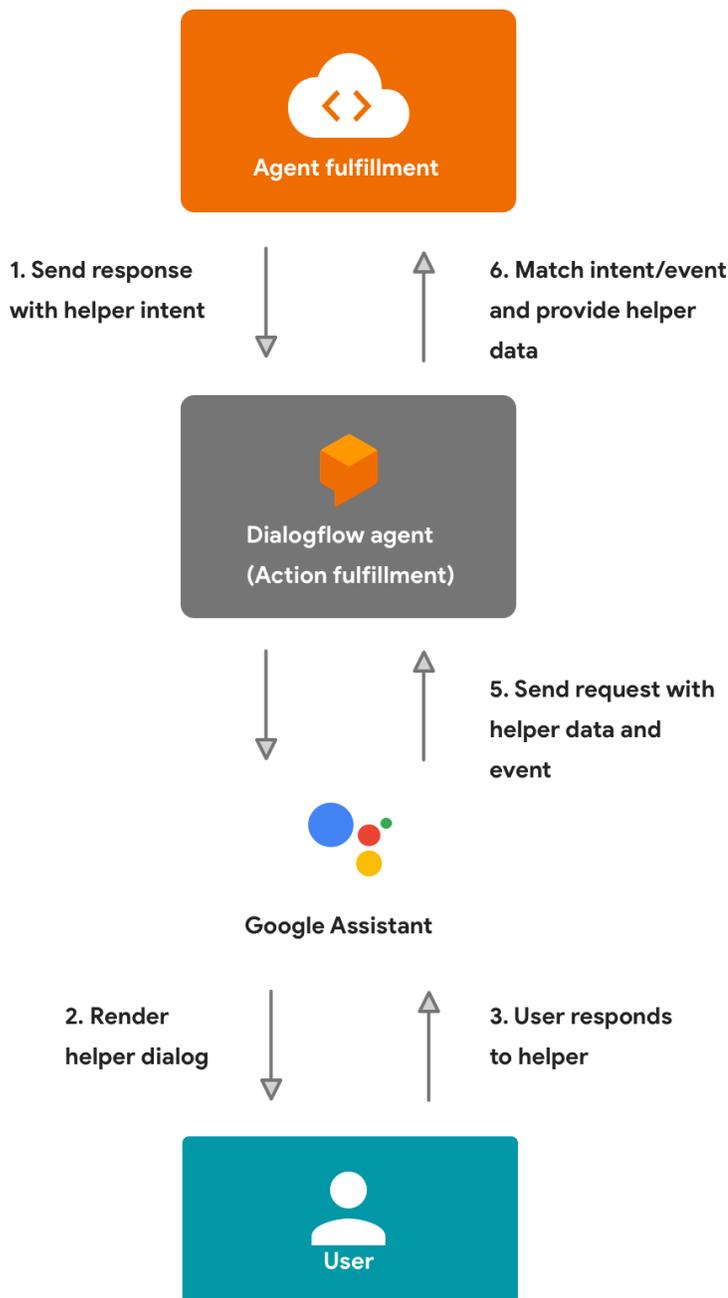
Dialog intents



After your agent receives an initial request to carry out an Action, it must respond back. Typically, a conversation involves many back and forth turns and expects requests to come in after a response. These requests are represented by the `actions.intent.TEXT` intent, which represents a generic user utterance after your agent has been invoked. The user utterances that are associated with these intents are matched to the corresponding Dialogflow intent in your agent. From there, your agent respond back accordingly.

Any intent in your Dialogflow agent can behave like a dialog intent, as long as the user utterance matches the Dialogflow intent's language model. For more information on how to build dialogs, see the Build documentation.

Helper intents



Helpers are small dialogs or tasks that you can ask the Assistant to carry out on your fulfillment's behalf. Helpers let you call on standard logic to carry out common tasks like asking users for their email address or location.

To request a helper, you specify a helper intent in a response back to the Assistant instead of the generic `actions.intent.TEXT` intent. When the Assistant receives this intent, it carries out the corresponding dialog, gets the information it needs from the user, and returns it back to you in the subsequent request to your fulfillment. In this request, a helper event is sent to your agent, along with the data returned from the helper. The corresponding intent is matched and your agent's fulfillment can process the helper data.

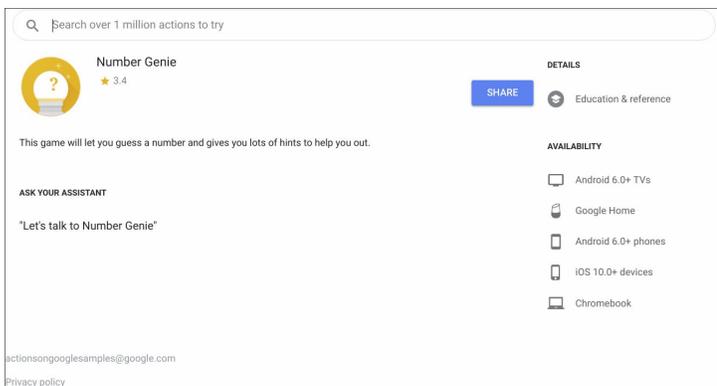
For more information on how to use helpers, see the [Build documentation](#).

Setup

To start building conversational Actions with Dialogflow, you'll first need an Actions project, which lets you manage your Actions and a Dialogflow agent, which lets you build and manage your conversational fulfillment.

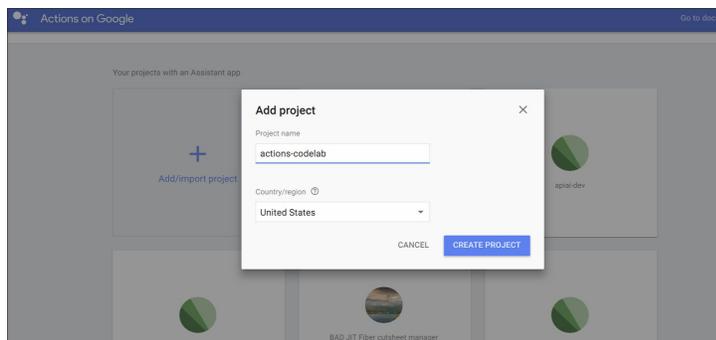
Create an Actions project

Actions projects are containers for your Action(s) with the metadata (name, description, category) that becomes your Actions directory listing.



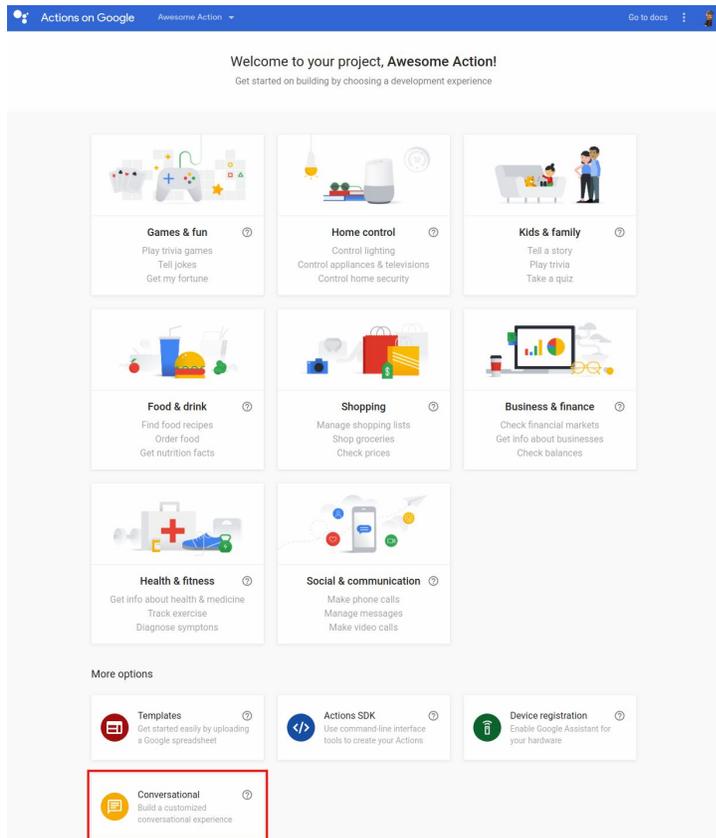
To start building Actions, you'll first need to create an Actions project:

1. Open the Actions Console.
2. Click on **Add/import project**.
3. Type in a **Project name**, like `actions-codelab`. This name is for your own internal reference; later on, you can set an external name for your project.



4. Click **Create Project**.

5. Choose the **Conversational** card at the bottom of the page.

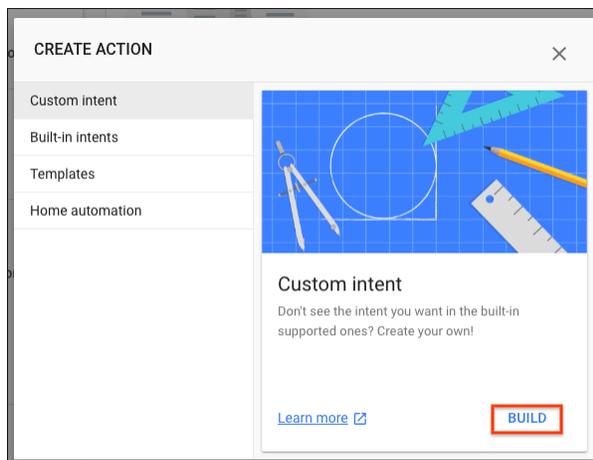


6. Click **Build > Actions** in the left nav.

7. Click **Add your first Action**.

8. Select at least one language for your Action, followed by **Update**. For this codelab, we recommend only selecting English.

9. On the **Custom intent** card, click **Build**. This opens the Dialogflow Console in another tab.



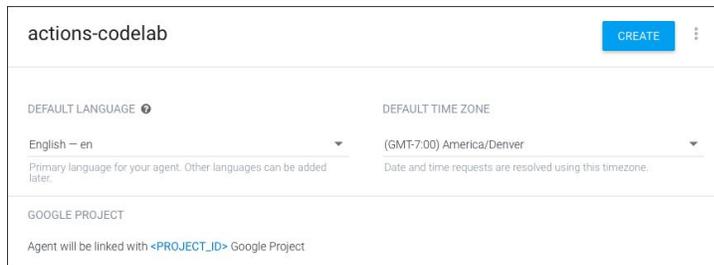
Create a Dialogflow agent

Now that you've built your Actions project, create a Dialogflow agent and associate it with your project:

1. After following the steps above, you should already be in the Dialogflow Console with your Actions project name at the top. You may need to authorize Dialogflow to use your Google account, and accept the Terms of Service.

Note: Not in the Dialogflow Console? Make sure you've completed all the steps in the 'Create an Actions Project' section. Still don't see it? Then you can navigate to the Dialogflow Console and select Create new agent in the left navigation.

2. Click **Create**.



The screenshot shows the 'Create' form in the Dialogflow console for an Actions project named 'actions-codelab'. The form includes a 'CREATE' button and three main sections: 'DEFAULT LANGUAGE', 'DEFAULT TIME ZONE', and 'GOOGLE PROJECT'. The 'DEFAULT LANGUAGE' is set to 'English - en' with a note that other languages can be added later. The 'DEFAULT TIME ZONE' is set to '(GMT-7:00) America/Denver' with a note that date and time requests are resolved using this timezone. The 'GOOGLE PROJECT' section indicates that the agent will be linked with a project ID.

DEFAULT LANGUAGE	DEFAULT TIME ZONE
English - en <small>Primary language for your agent. Other languages can be added later.</small>	(GMT-7:00) America/Denver <small>Date and time requests are resolved using this timezone.</small>

GOOGLE PROJECT
Agent will be linked with <PROJECT_ID> Google Project

If the agent creation is successful, the **Intents** appears. You can now begin customizing how your Dialogflow agent responds to user requests.

Build

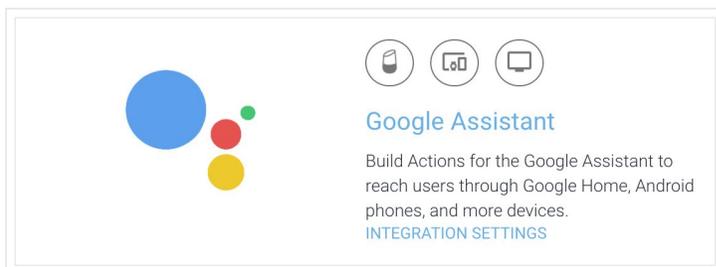
When you are done setting up an Actions project and a Dialogflow agent, you can start building your agent. In Dialogflow, intents represent a mapping between what a user says and the corresponding response. You build your intents based on your conversation design, typically creating an intent for each turn (a back and forth exchange between the user and your agent) of the conversation and mapping Actions to specific Dialogflow intents.

The next sections show you how to create intents that map to specific Actions on Google concepts, such as Actions and helpers. For more information about how Dialogflow intents and Actions on Google intents map to one another, see the Actions on Google integration documentation.

Access the Actions on Google integration

To access the Actions on Google integration:

1. Click on **Integrations** in the left menu. If you don't see the menu, click the menu **menu** button in the upper left corner.
2. Click on the **Google Assistant** integration to open the settings.



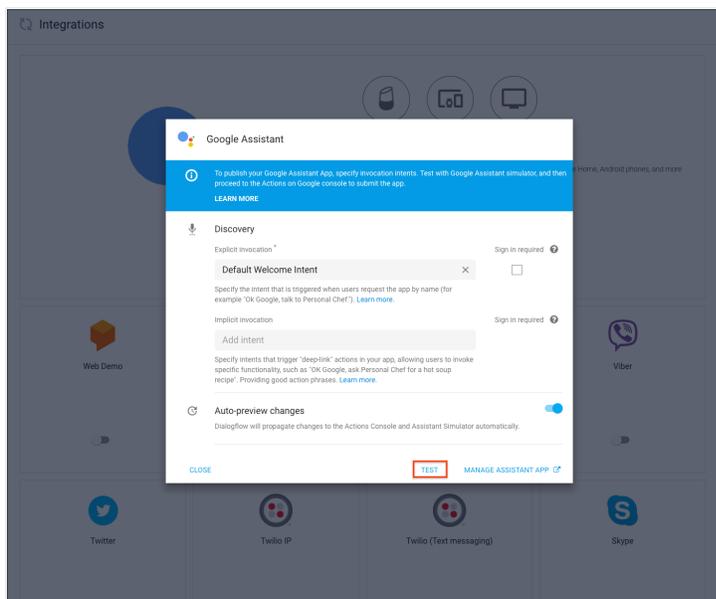
From here, you can specify, test, and manage your Actions.

Specify Actions

To mark specific Dialogflow intents as Action intents, use the Actions on Google integration screen. This screen lets you specify your explicit invocation intent (the intent that is matched when users say the invocation name defined in your Actions on Google project) and any other custom Actions that your Dialogflow agent can support (implicit invocation).

1. Set the explicit invocation intent with the drop-down menu. By default, all agents are created with one **Default Welcome Intent** that is automatically set as the explicit invocation intent.

When users say your invocation name, this intent is matched.



2. Add additional intents to the **Implicit invocation** field. This lets you specify additional Actions that you support outside of the default Action. The training phrases in these intents specify the invocation phrases for your Action. When defining these invocation phrases, follow these guidelines:
 - Don't specify "reserved" wording from name invocation or Action invocation and "trigger phrases". For example, don't use "talk to <action name>", "talk to", or "let me talk to".
 - Don't specify training phrases that contain only a `@sys.any` entity. You should be much more specific, such as "find recipes for `@sys.any`".

Handle unrecognized Actions

When users try to specify an Action that isn't supported, Dialogflow can trigger a specific fallback intent to handle these cases. To enable this behavior, create a fallback intent that sets an input context to be `google_assistant_welcome` and specify the appropriate responses to users:

1. In the left navigation of Dialogflow, click **Intents**.
2. Click the overflow menu icon next to the **Create Intent** button and select **Create Fallback Intent**.
3. In the **Contexts** section, specify `google_assistant_welcome` as the input context. Make sure you hit the Enter key so the context is properly set.
4. Specify responses that are appropriate for an unrecognized Action in the **Response** section.
5. If you want to use fulfillment to handle this intent:
 1. In the **Action** section, specify an Action name.
 2. Click the **Fulfillment** section header and select **Enable webhook call for this intent**.
 3. When Dialogflow triggers this intent, you can call the `getRawInput()` method of the client library to obtain the unrecognized Action phrase. You can then return responses in your fulfillment logic based on the unrecognized Action phrase or other logic.

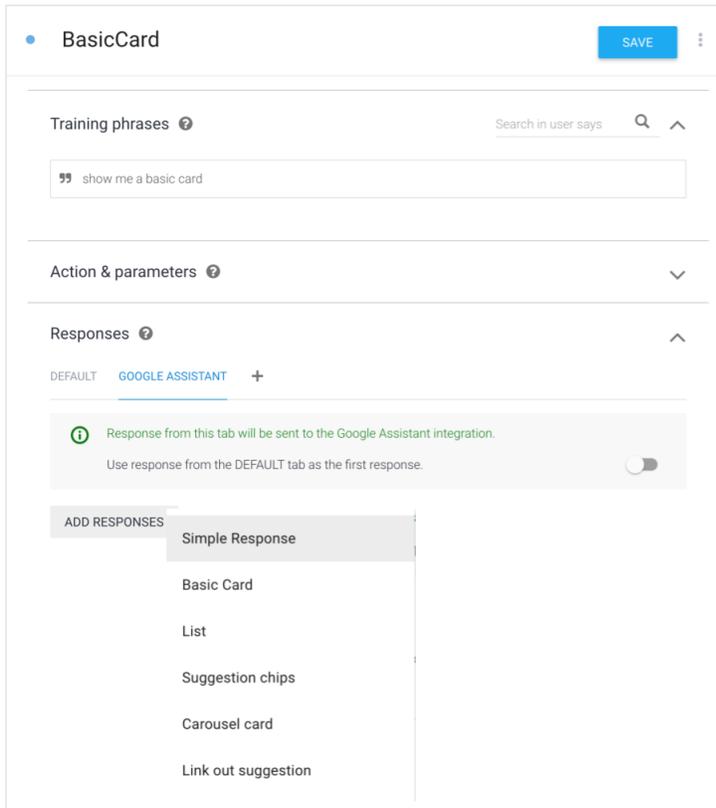
Build dialogs

You build dialogs by creating Dialogflow intents, specifying training phrases to define the grammar or what users need to say to trigger an intent and the corresponding response to the intent. Dialogflow also supports Assistant rich responses, like basic cards, lists, and more. You can create as many intents as you'd like to define your entire conversation's grammar.

To create an intent:

1. Click the **add** sign by the **Intents** menu item in Dialogflow's left navigation. The Intent Editor appears where you can enter the following information:
 - **Intent name** is the name of the intent that's displayed in the IDE. This name is also used to register intent handlers in your fulfillment code if you're using the Actions on Google fulfillment library.
 - **Contexts** let you scope the triggering of the intent to specific cases. This is an advanced feature of Dialogflow, so read the Dialogflow documentation on contexts for more information.
 - **User says** phrases define what users need to say (the grammar) to trigger the intent. Type a few phrases here (5-10) of what users can say to trigger the intent. Dialogflow automatically handles natural variations of the example phrases you provide.
 - **Events** trigger intents without the need for users to say anything. One example event is the `GOOGLE_ASSISTANT_WELCOME` event, which allows the Google Assistant to invoke your Action. This event is used for your agent's default, explicit Action. See the Actions on Google documentation for more information on built-in intents and helper intents, which require you to assign events to the corresponding Dialogflow intents.
 - **Action & parameters** defines what data to pass to fulfillment, if fulfillment is enabled for this intent. This includes data parsed from the user input and the name that you can use in your fulfillment to detect which intent was triggered. You'll use this name later when you create an Action map, which maps an intent to its corresponding fulfillment logic. See the Actions and Parameters for more information about defining Actions.
Note: "Action" refers to this specific component of a Dialogflow intent, and doesn't refer to the larger concept of an Action in the Actions on Google platform.
 - **Response** - The Dialogflow Response Builder lets you define the response to this intent directly within Dialogflow, without calling fulfillment. This feature is useful for responses that are static and don't require fulfillment. You might use this for things such as simple welcome or goodbye messages. However, you will likely use fulfillment to respond to your users for most intents.
 - **Fulfillment** specifies whether or not you want to call your fulfillment when this intent is triggered. You most likely will enable this for most intents in your Dialogflow agent. To see this item in the intent, you must have fulfillment enabled for the agent in the **Fulfillment** menu.

- **Actions on Google** lets you specify platform-specific settings for the intent. Currently, only **End conversation** is supported. This closes the mic and ends the conversation after the corresponding response is returned to the user. This setting is only honored if you don't use fulfillment for the intent.
2. In the **Responses** section, switch to the **Google Assistant** tab and click **Add Responses**. We'll show you how to add a basic card.
 3. From the list, choose **Simple Response**. Actions on Google requires a simple response (text and/or voice only) for each response.



4. In the simple response field, enter a phrase to introduce the card.



5. Click **Add Responses** again and select **Basic Card**.

6. Add information to build the card. The following screenshot shows values for each field and how they are rendered:



- **Image URL** - `https://blog.dialogflow.com/images/logo.svg`
- **Image accessibility text** - Dialogflow logo
- **Title** - Dialogflow
- **Subtitle** - Build natural and rich conversational experiences
- **Text** - This is a basic card!
- **Weblink title** - Dialogflow website
- **Weblink** - `https://dialogflow.com`

7. Click the **Save** button.

Call helpers

For more information on how to use helpers, see the Actions on Google Helpers documentation.

Build Dialogflow agent fulfillment

Your agent already acts as fulfillment for your Actions, but if you need to build fulfillment for your actual Dialogflow agent (for example, to call a REST API or a backend database) you can do that as well. You can turn on fulfillment on a per-intent basis.

Note: Timeout for the webhook with Actions on Google is 10 seconds.

When an intent is triggered that uses fulfillment, you receive a request from Dialogflow that contains information about the intent. You then respond to the request by processing the intent and returning a response. The Dialogflow fulfillment protocol defines the request and response formats.

We highly recommend that you use the Node.js client library to process requests and return responses. Here's the general process for using the client library:

1. Initialize the DialogflowApp object. This object automatically handles listening for requests and parsing them so that you can process them in your fulfillment.
2. Create functions to handle requests. These functions process the user input and other components of the intent and build the response to return to Dialogflow.

Initialize the DialogflowApp object

The following code instantiates `DialogflowApp` and does some boilerplate Node.js setup for Google Cloud Functions:

```
'use strict';

const {dialogflow} = require('actions-on-google');
const functions = require('firebase-functions');

const app = dialogflow({debug: true});

app.intent('Default Welcome Intent', (conv) => {
  // Do things
});

exports.yourAction = functions.https.onRequest(app);
```

Create functions to handle requests

When users speak a phrase that triggers an intent, you receive a request from Dialogflow that you handle with a function in your fulfillment. In this function, you'll generally do following things:

1. Carry out any logic required to process the user input.
2. Build your responses to respond to triggered intents. Take into account the surface that your users are using to construct appropriate responses. See surface capabilities for more information on how to cater responses for different surfaces.
3. Call the `ask()` function with your response.

The following code shows you how to build two TTS responses that handles an invocation intent (`input.welcome`) and a dialog intent (`input.number`) that welcomes the user to your Action and echoes back a number that a user has spoken for a Dialogflow intent with the name:

```
const app = dialogflow();

app.intent('Default Welcome Intent', (conv) => {
  conv.ask('Welcome to number echo! Say a number. ');
});

app.intent('Input Number', (conv, {num}) => {
  // extract the num parameter as a local string variable
  conv.close(`You said ${num}`);
});
```

Instead of having individual handlers for each intent, you can alternatively add a fallback function. Inside the fallback function, check which intent triggered it and do the appropriate thing accordingly.

```
const WELCOME_INTENT = 'Default Welcome Intent';
const NUMBER_INTENT = 'Input Number';
const NUMBER_ARGUMENT = 'num';

// you can add a fallback function instead of a function for individual
// intents
app.fallback((conv) => {
  // intent contains the name of the intent
  // you defined in the Intents area of Dialogflow
  const intent = conv.intent;
  switch (intent) {
    case WELCOME_INTENT:
      conv.ask('Welcome! Say a number. ');
      break;

    case NUMBER_INTENT:
```

```
const num = conv.arguments.get(NUMBER_ARGUMENT);
conv.close(`You said ${num}`);
break;
}
});
```

No-match reprompting

When Dialogflow cannot match any of the input grammars defined in the **User says** portion of your intents, it triggers a fallback intent. Fallback intents typically reprompt the user to provide the necessary input for your Action. You can provide reprompt phrases by specifying them in the **Response** area of a fallback intent or you can use a webhook to provide responses.

Note: By default, Dialogflow automatically creates a **Default Fallback Intent** for every agent. You should modify the responses provided by this intent or delete it and create your own fallback intents to properly handle no-match cases. The default responses in the **Default Fallback Intent** are too general in most cases.

To create fallback intents:

1. Click **Intents** in the left navigation of Dialogflow.
2. Click the menu icon to the right of the **Create Intent** button and select **Create Fallback Intent**. Alternatively, click the **Default Fallback Intent** to edit it.
3. Specify reprompt phrases to speak back to users. These phrases should be conversational and be as useful as possible to the user's current context.

To do this without fulfillment:

Specify phrases in the **Response** area of the intent. Dialogflow randomly chooses phrases from this list to speak back to users until a more specific intent is triggered.

To do this with fulfillment:

1. Select the **Use webhook** checkbox in the **Fulfillment** area of the intent.
2. In your fulfillment logic, handle the fallback intent that gets triggered like with any other intent, as described in creating functions to handle requests.

For example, the following function uses the `app.data` object (an arbitrary data payload that you can use to maintain state) to store a counter that tracks how many times a fallback intent is triggered. If it's triggered more than once, the Action quits. You can then reset the counter when any other intent is triggered.

```
app.intent('Default Fallback Intent', (conv) => {
```

```
conv.data.fallbackCount++;
// Provide two prompts before ending game
if (conv.data.fallbackCount === 1) {
  conv.contexts.set(DONE_YES_NO_CONTEXT, 5);
  conv.ask('Are you done playing Number Genie?');
} else {
  conv.close(`Since I'm still having trouble, so I'll stop here. ` +
    `Let's play again soon.`);
}
});
```

See the Number Genie sample for details on how to implement this.

Using contexts

Use contexts if you want Dialogflow to trigger fallback intents only in certain situations. This is helpful if you want to have different fallback intents for different no-match scenarios.

- If you don't set contexts on a fallback intent, it's considered to be a global fallback intent that Dialogflow triggers when no other intent is matched. You should only have one of these defined if you choose to use one.
- If you set input contexts on a fallback intent, Dialogflow triggers this fallback intent when the following is true:
 - The user's current contexts are a superset of the contexts defined in the intent.
 - No other intent matches.
- This lets you use multiple fallback intents with different input contexts to customize no-match reprompting to specific scenarios.
- If you set an output context on a fallback intent, you keep the user in the same context after the fallback intent is triggered and processed.

See Dialogflow Contexts for more information.

No-input reprompting

When you call the `ask()` method to return responses to users, you can specify an array of prompts that is spoken to users if the Google Assistant can't detect any input. The Google Assistant speaks these prompts in order, and you can specify up to three prompts. For example:

```
app.intent('actions.intent.MAIN', conv => {
  conv.noInputs = [
    'Are you still there?',
    'Hello?',
```

```
    new SimpleResponse({
      text: 'Talk to you later. Bye!',
      speech: 'Talk to you later. Bye!'
    })
  ]
  conv.ask('What's your favorite color?')
})
```

Test and Submit Actions

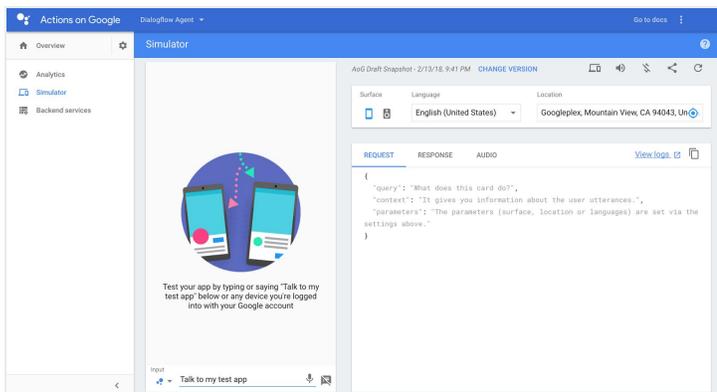
Now that you have your Dialogflow agent and fulfillment deployed, you can test out your Action on real hardware devices or the Actions Console simulator.

Testing on real hardware

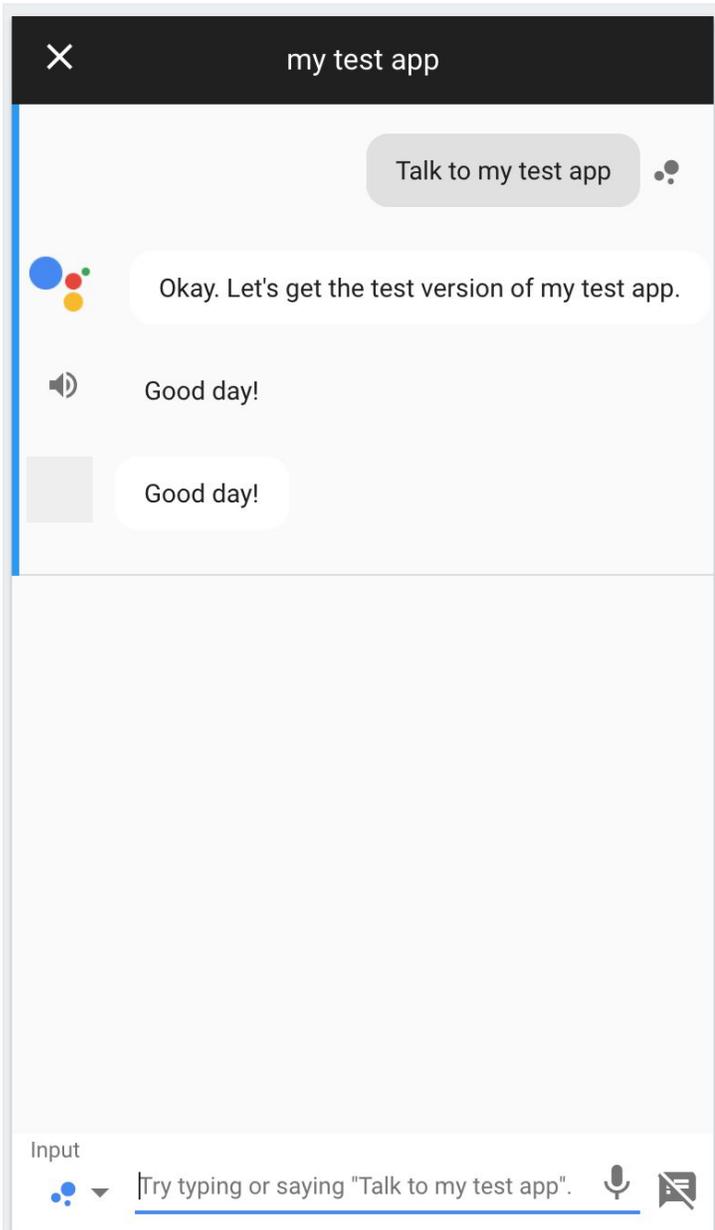
On a supported device, such as a voice-activated speaker or an Android phone, log into the device with the same account that you used to create your Actions on Google project.

Test with the Actions simulator

The Actions console is where you manage all of the information, deployment, and configuration of your actions. In this case, we are building an app for the Google Assistant that consists of multiple actions and is powered by Dialogflow.

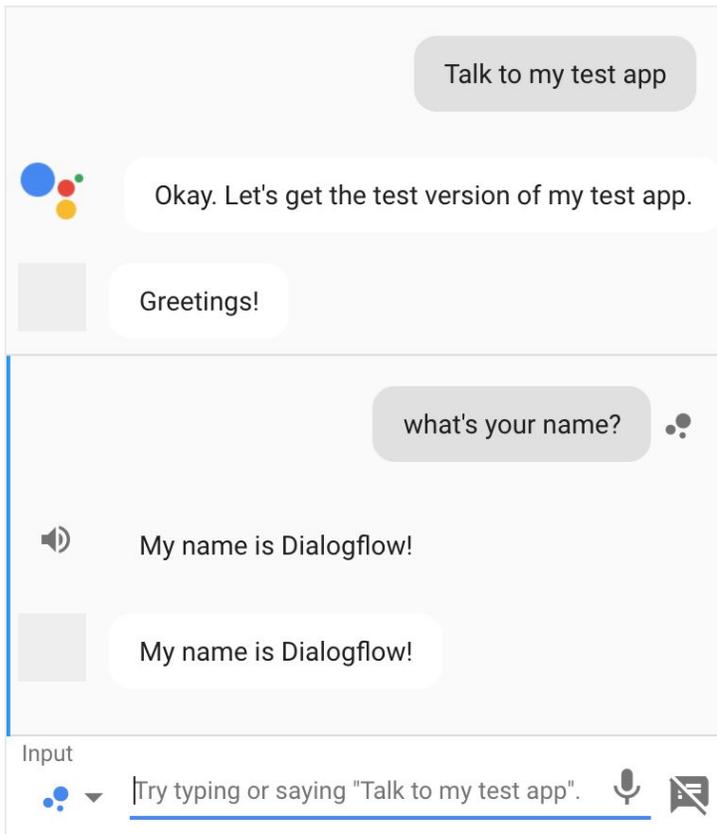


The left panel has information on your actions, like analytics, an overview of information, and how to deploy. The center is where you can talk to and test your app for Google Assistant. The right panel has debugging information on the current conversation and tools to test different situations. You can read more on how to use the Actions console on the Actions on Google developer site.



To test your Assistant app, click the text box that says "Talk to my test app" and press enter. The Google Assistant hands off the conversation to your app by saying "Okay. Let's get the test version of my test app." Then, your Dialogflow agent's Default Welcome intent will be matched, and the resulting response is sent to the Assistant.

After the welcome intent is matched, the user's next query is matched against your agent's intents, just like in the Dialogflow simulator. For instance, if you have an intent that captures when a user asks for your agent's name, when asked "What's your name?", the Actions on Google simulator would look like this:



Submit your Action for approval

When you want to update your developer project with a copy of your agent so you can submit it for review, use the update feature of Dialogflow. This uploads an Action package to your developer project for review.

1. In Dialogflow's left navigation, click **Integrations**, then the **Actions on Google** card.
2. Click on the **UPDATE** button.
3. Go to the Actions on Google Developer Console and select your project. Fill out the information requested to submit your Action for approval. See Registering and Publishing for more information.

Note: Every subsequent time you update the Action package, it has to go through another approval cycle.

Dialogflow versioning

Dialogflow versioning allows you to take a snapshot of your agent to create a **version** that is immutable. You can create as many versions as you want and deploy a specific version to production or alpha/beta environments.

Note: Starting May 8, 2018, any new Action submission will create a snapshot of your Dialogflow agent at the time of submission. This snapshot (or *version*) will not be impacted by any future edits you make to your Dialogflow agent. For edits to go live, you'll need to submit your draft for alpha, beta, or production release in the Actions Console.

With the Dialogflow versioning feature, any edits you make to your Dialogflow agent will not impact your publicly available Actions. You can iterate and improve your Action in draft mode and make it available to users only when you're ready.

You'll also be able to release different versions of your Actions in alpha, beta, or production release environments and roll back to previous versions if necessary.

Publishing to alpha/beta release environments

Dialogflow versioning is enabled automatically for Actions, when you submit an Action that you built with Dialogflow through the Actions Console.

To submit Actions from the Dialogflow Console to a Actions on Google release environment, follow these steps:

1. Click on **Integrations** in the left navigation, click on the **Google Assistant** card and then click **MANAGE ASSISTANT APP** in the dialog.
2. In the Actions Console, follow the steps described in Set up alpha/beta releases.

FAQ

- **How does Dialogflow versioning impact my existing Actions?**
 - Existing Actions submitted to the Actions Console prior to May 8, 2018 will not be impacted. This means that edits you make to your Dialogflow agent will continue to be reflected in the existing Action. If the Action is already published, the edits will continue to be reflected live.
- **Will there be two versions I need to maintain in Action on Google and Dialogflow?**
 - No, you have only one version as shown in your **Manage releases** page. It is unified across Actions on Google and Dialogflow. The only difference is now that we will create a snapshot of your Dialogflow agent at the time of submission. This snapshot will not be impacted by any future edits you make to your Dialogflow intents and entities.
- **Is versioning supported for Dialogflow V1?**
 - Yes, it is supported for Actions on Google projects built with Dialogflow V1 or V2 APIs.
- **Can I create and deploy Actions on Google versions from the Dialogflow Console?**
 - No, you can only do so via the Actions Console.
- **In the Dialogflow page I saw that this is available for users who opt-in to beta, is that applicable for Actions on Google?**

- No, the Dialogflow beta program is for Dialogflow agents that are not on the Actions on Google platform. If you have a Dialogflow agent for the Actions on Google platform, then it is automatically enabled and not in beta.
- **What happens if I enable beta in the Dialogflow Console?**
 - Doing so will not change Actions on Google projects. If you enable beta, you will see an **Environment** tab in the **Dialogflow Agent** settings. In that page, you can view (but not edit) the same versions you created from the Actions Console and load any specific ones into the draft.