

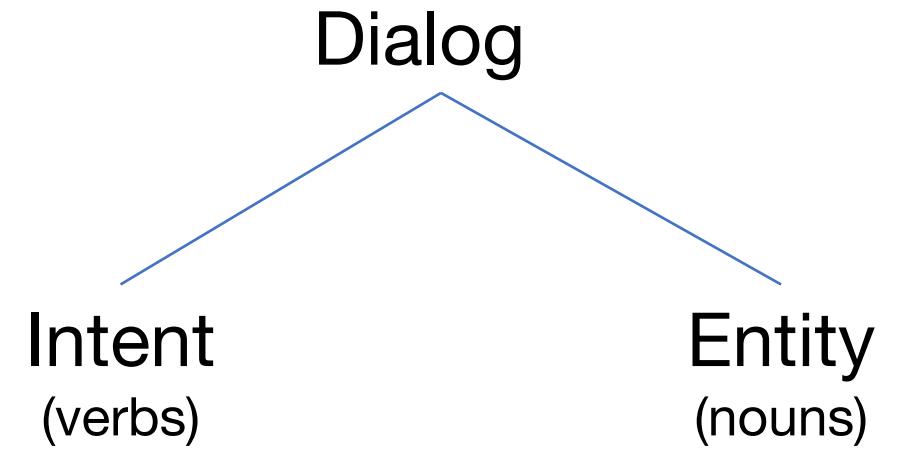
Watson Conversation

Lecture Deck



Components of Training Watson

Create a dialog such as a chat bot, customer service agent, smart product, that responds to natural language



Intents

An *intent* is a purpose or goal expressed in a customer's input, such as answering a question or processing a bill payment.

Provide at least five examples for each intent.

#turn_on

+ Add a new user example...

turn on the wipers and the lights

Turn on the wipers and lights

turn on the strobe lights

turn on the lights too

Turn on the lights please.

Turn on the lights please?

turn on the lights, please

turn on the lights please

turn on the lights place

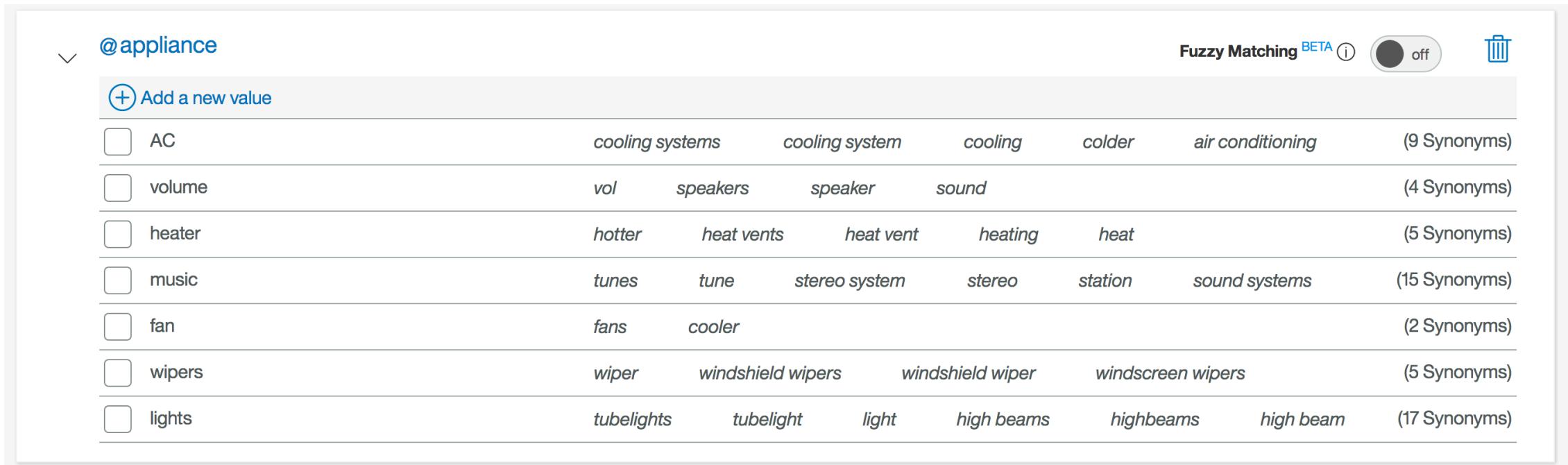
Turn on the lights -- it's too dark outside

Designing Intents

1. Gather as many actual customer questions, commands, or other utterances as possible. Using input from real users gives a better picture of the expected input than having experts create lists of possible utterances. Remember that customers might phrase the same kind of request in many different ways.
2. After you have a list of examples, sort them into categories based on the capabilities you want your application to support. These categories represent the intents you will define, and the examples will help your application to identify the intents in new input.
3. Continue to refine your intents and examples as needed. Do not think of your set of intents as a finished product. It is likely that when you design your dialogs, you will identify additional intents that you need to add.

@ Entities

An *entity* represents a class of object or a data type that is relevant to a user's purpose. By recognizing the entities that are mentioned in the user's input, the Conversation service can choose the specific actions to take to fulfill an intent.



The screenshot shows the IBM Watson Assistant interface with the following details:

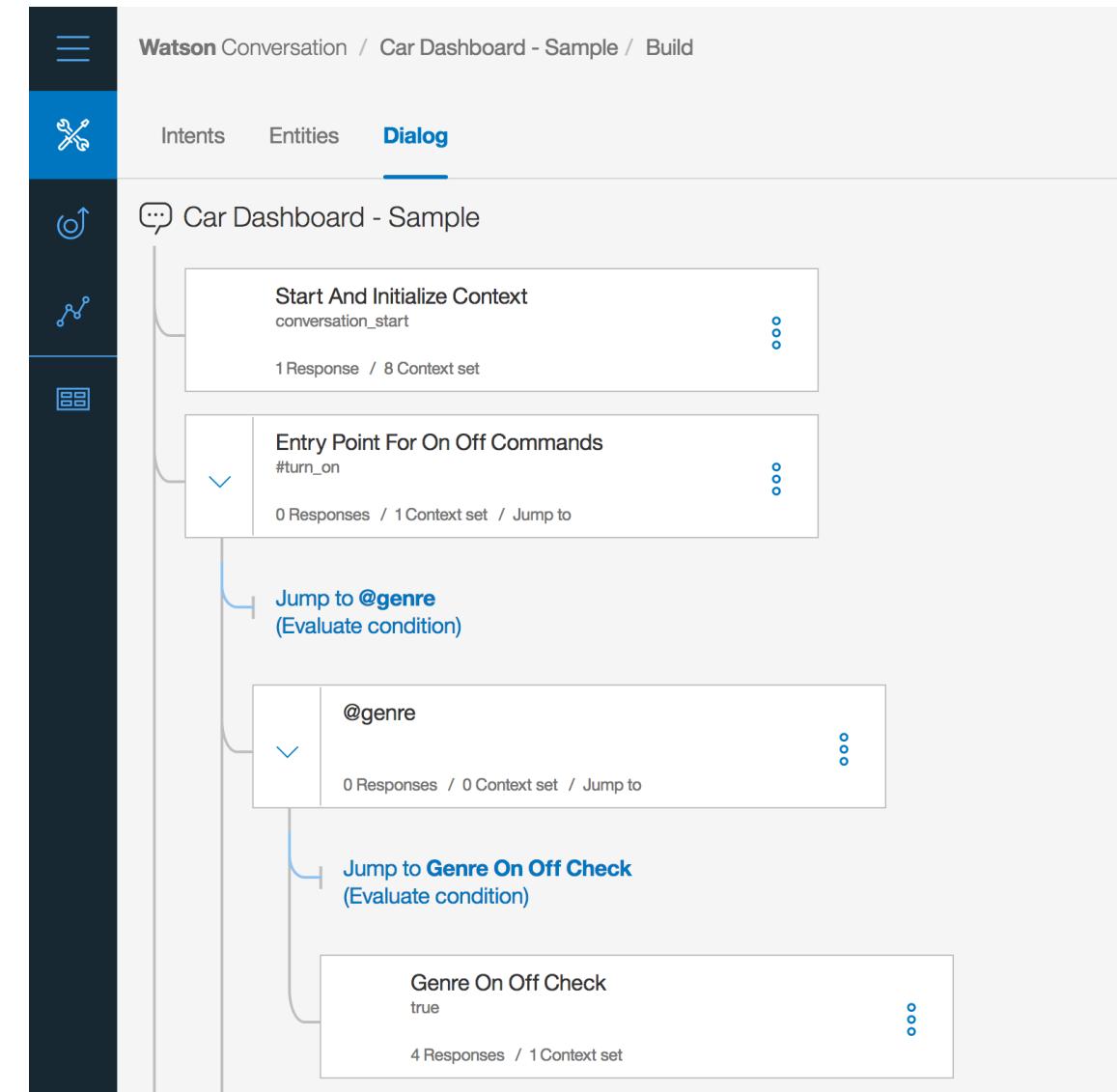
- Entity Name:** @appliance
- Fuzzy Matching:** BETA (off)
- Add a new value:** (+) Add a new value
- Entity Data:**

Value	cooling systems	cooling system	cooling	colder	air conditioning	(# Synonyms)
AC						(9 Synonyms)
volume	vol	speakers	speaker	sound		(4 Synonyms)
heater	hotter	heat vents	heat vent	heating	heat	(5 Synonyms)
music	tunes	tune	stereo system	stereo	station	sound systems (15 Synonyms)
fan	fans	cooler				(2 Synonyms)
wipers	wiper	windshield wipers	windshield wiper	windscreen wipers		(5 Synonyms)
lights	tubelights	tubelight	light	high beams	highbeams	high beam (17 Synonyms)

Dialog

The dialog component of the Conversation service provides responses to users based on the identified intents and examples.

A dialog uses the intent and entity that have been identified, plus context from the application, to interact with the user and ultimately provide a response.



Designing Dialog Conditions

Condition	What it does...
welcome	Evaluated as true during the first dialog turn (when the conversation starts), only if the initial request from the application does not contain any user input .
conversation_start	Initial output. Unlike welcome , it is true whether or not the initial request from the application contains user input .
@cuisine:Chinese	Process node if an entity toppings with value Chinese is present
#{{intent name}}	Process node if intent is present
entities.size() > 1	Test the number of entities input has
true	Catch-all within a child tree level
anything_else	Catch-all within top tree level . We've got a problem we can't solve.

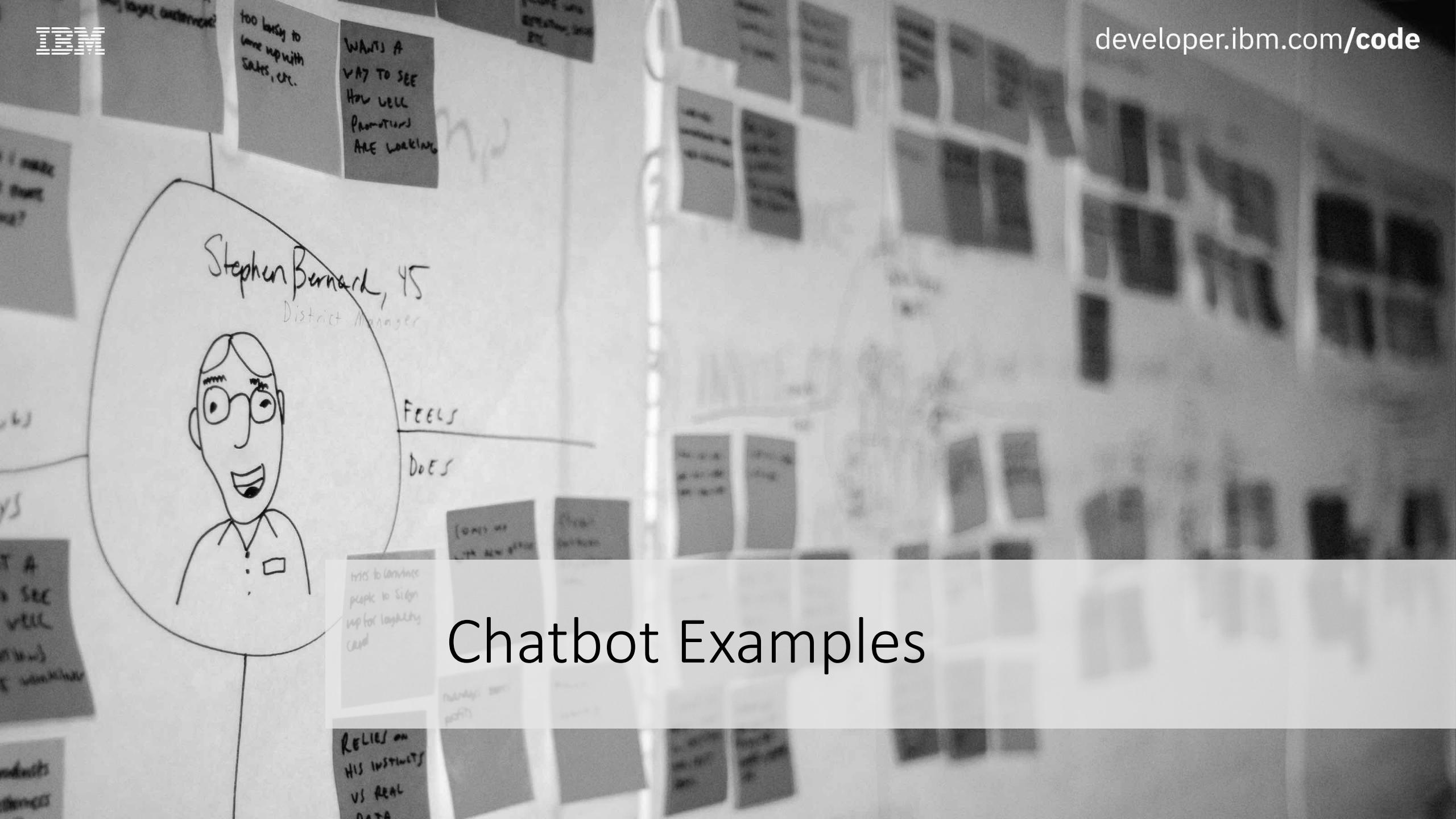
Slots

Add slots to a dialog node to gather multiple pieces of information from a user within that node.

Slots collect information at the users' pace. Details the user provides upfront are saved, and the service asks only for the details they do not.

The screenshot shows the configuration of a 'Book Reservation' dialog node. At the top, there's a section titled 'If bot recognizes:' with the value '#book_reservation'. Below this, under 'Then check for:', there are four entries, each consisting of a slot name, a check-for expression, a save-it-as variable, and a question text. The entries are:

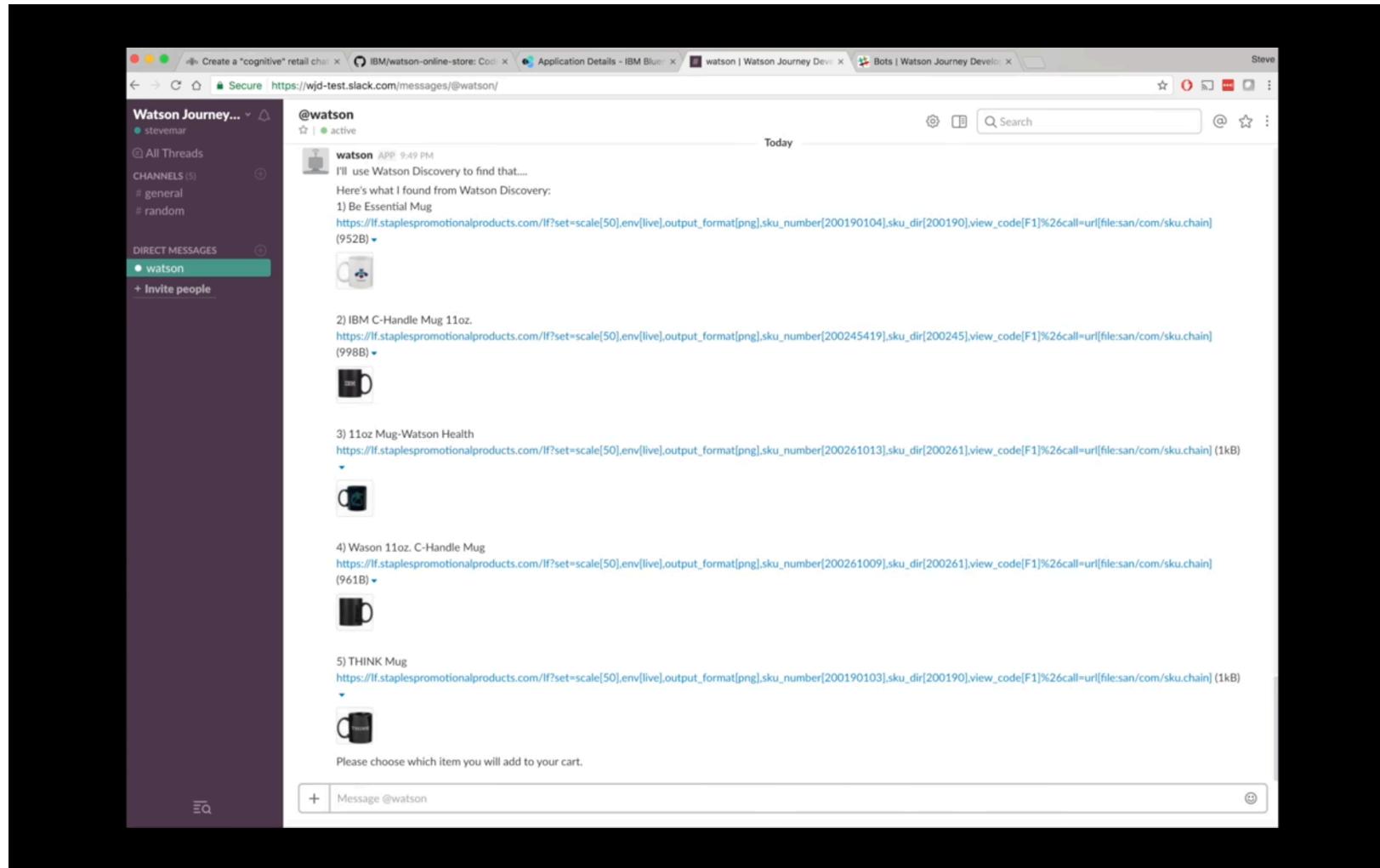
Slot	Check for	Save it as	If not present, ask	Required
1	@cuisine	\$cuisine	What type of cuisine would you like	<input checked="" type="checkbox"/>
2	@sys-date	\$date	What day would you like	<input checked="" type="checkbox"/>
3	@sys-time	\$time	What time would you like	<input checked="" type="checkbox"/>
4	@sys-number	\$number	How many people will be	<input checked="" type="checkbox"/>



Chatbot Examples

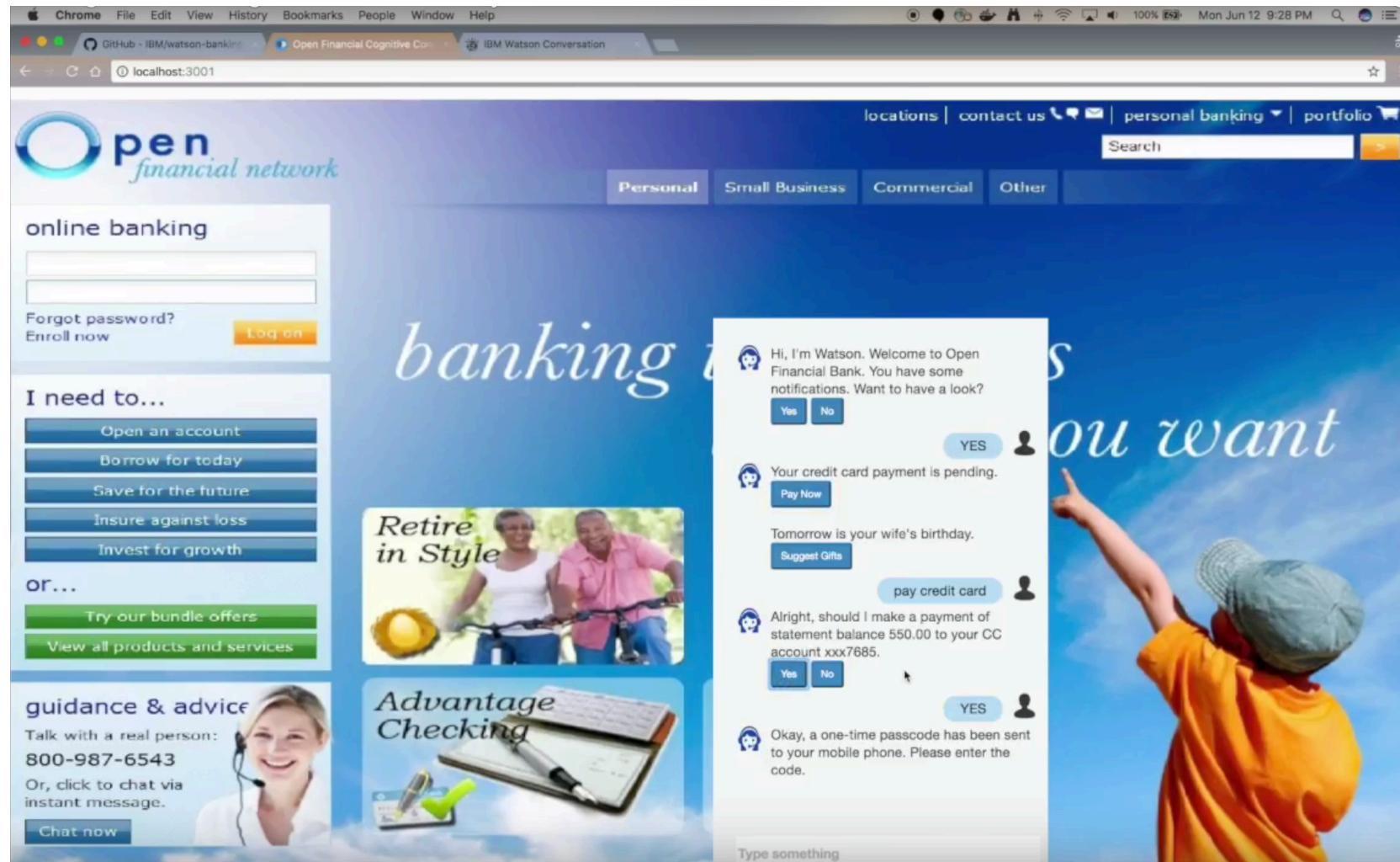
Configurable, retail-ready chatbot

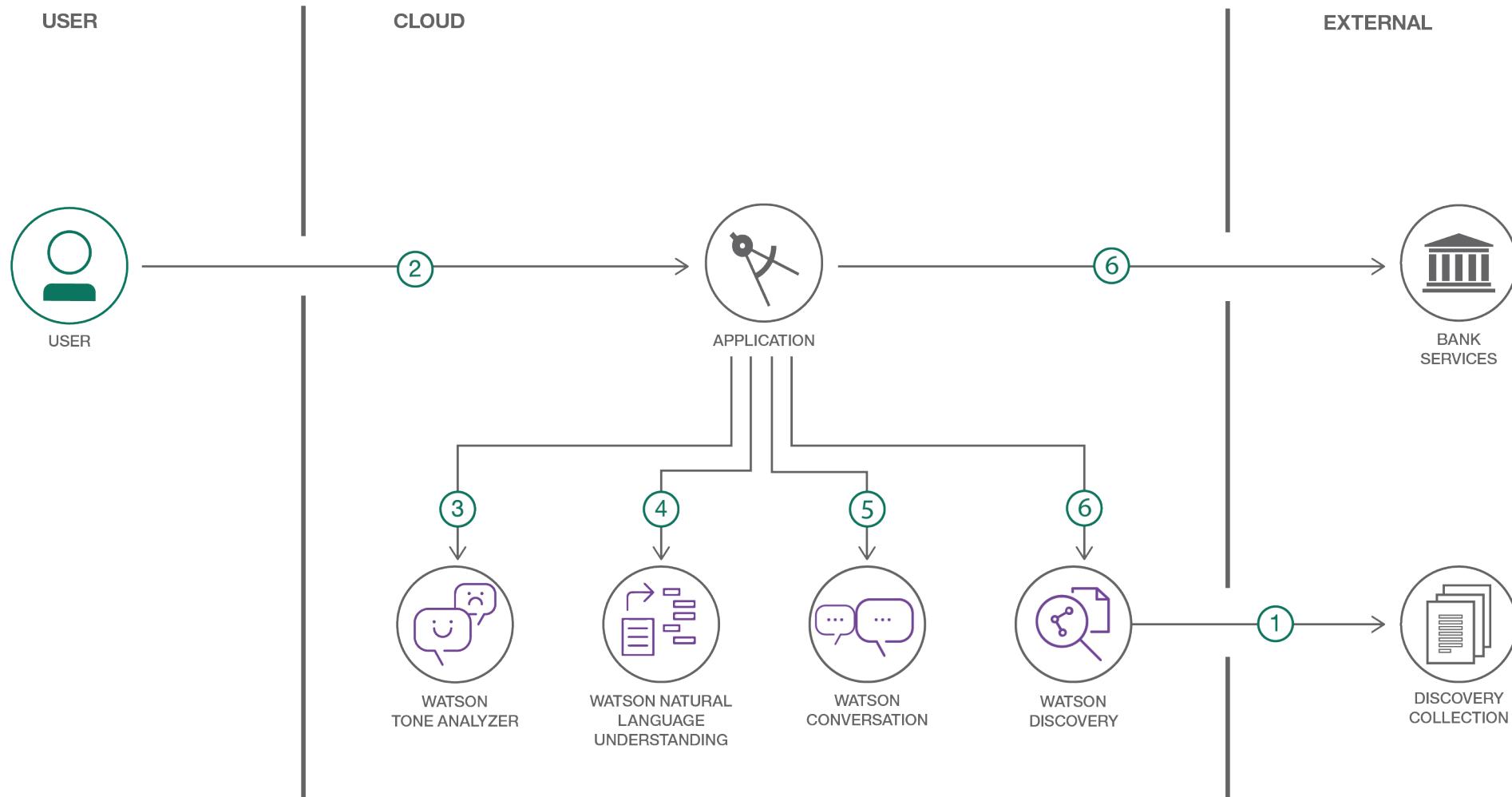
ibm.biz/create-cognitive-retail-chatbot



Create a cognitive banking chatbot

ibm.biz/create-cognitive-banking-chatbot







BotKit Middleware

The screenshot shows the GitHub repository page for `watson-developer-cloud / botkit-middleware`. The repository has 106 commits, 2 branches, 6 releases, and 9 contributors. It has 19 watchers, 87 stars, and 90 forks. The repository description states: "A middleware to connect Watson Conversation Service to different chat channels using Botkit". A link to the npm package is provided: <https://www.npmjs.com/package/botkit-...>.

A middlewarer to connect Watson Conversation Service to different chat channels using Botkit
<https://www.npmjs.com/package/botkit-...>

106 commits	2 branches	6 releases	9 contributors		
Branch: master	New pull request	Create new file	Upload files	Find file	Clone or download
germanattanasio committed on GitHub Merge pull request #82 from Naktibalda/patch-2 ... Latest commit cd79ce4 14 days ago					
examples	Merge pull request #77 from Naktibalda/patch-1	28 days ago			
lib/middleware	updateContext ignores error of storage.users.get	a month ago			
test	updateContext ignores error of storage.users.get	a month ago			
.gitignore	Initial commit 🚀	11 months ago			
.npmignore	ignore the test folder	10 months ago			
.travis.yml	add travis file	10 months ago			
CHANGELOG.md	updateContext ignores error of storage.users.get	a month ago			
LICENSE	rename license file	10 months ago			
README.md	[Readme] Fixed typo in updateContext example	14 days ago			
package-lock.json	1.4.2	28 days ago			
package.json	1.4.2	28 days ago			

Connecting An Application

The screenshot shows a GitHub repository page for `watson-developer-cloud / botkit-middleware`. The repository has 19 watch notifications, 87 stars, and 90 forks. The `Code` tab is selected, showing 17 issues and 0 pull requests. The `.env` file is being viewed, which contains environment variables for Watson Conversation and Slack integration.

Branch: master

`botkit-middleware / examples / simple-bot / .env`

Nakibalda Added CONVERSATION_URL parameter to examples
f642041 on Jul 18

2 contributors

10 lines (8 sloc) | 306 Bytes

```
1 #WATSON
2 CONVERSATION_URL=https://gateway.watsonplatform.net/conversation/api # US-South
3 #CONVERSATION_URL=https://gateway-fra.watsonplatform.net/conversation/api # Germany
4 CONVERSATION_USERNAME=username
5 CONVERSATION_PASSWORD=password
6 WORKSPACE_ID=your_workspace_id
7
8 #SLACK
9 SLACK_TOKEN=your_slack_bot_token
```

Make API calls to third-party data sources, databases, and microservices to get data.

Handles each message received via Slack

If Conversation service returns output from a Dialog node, responds back to the Slack channel

Application specific action to check balance and use it in the response back to the user.

```
function checkBalance(watsonResponse, callback) {
  //middleware.after function must pass a complete Watson response to callback
  watsonResponse.context.validAccount = true;
  watsonResponse.context.accountBalance = 95.33;
  callback(null, watsonResponse);
}

watsonMiddleware.after = function(message, watsonResponse, callback) {
  //real action happens in middleware.after
  if (typeof watsonResponse !== 'undefined' && typeof watsonResponse.output !== 'undefined') {
    if (watsonResponse.output.action === 'check_balance') {
      return checkBalance(watsonResponse, callback);
    }
  }
  callback(null, watsonResponse);
};

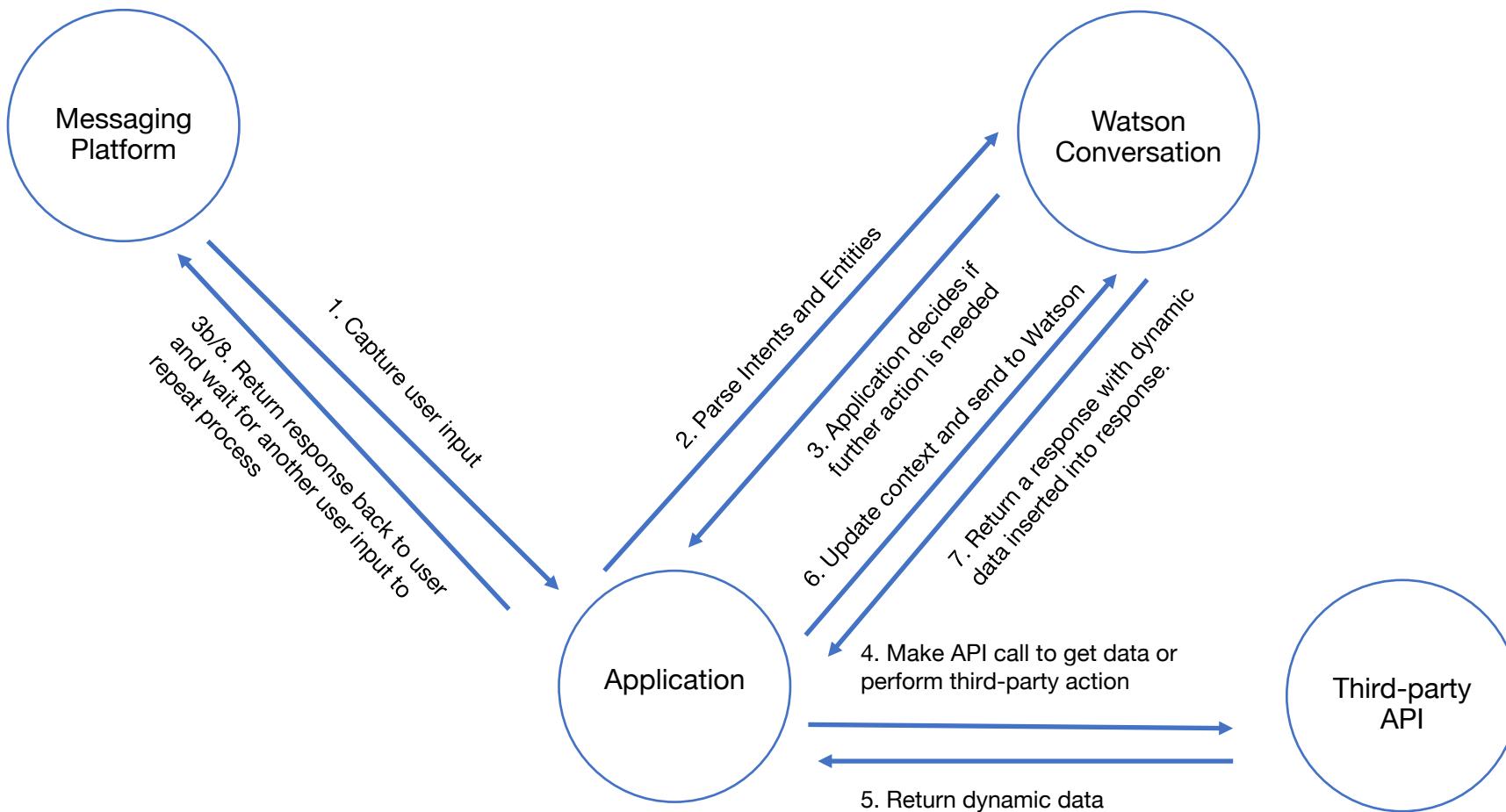
var processWatsonResponse = function(bot, message) {
  if (message.watsonError) {
    return bot.reply(message, "I'm sorry, but for technical reasons I can't respond to your message");
  }

  if (typeof message.watsonData.output !== 'undefined') {
    //send "Please wait" to users
    bot.reply(message, message.watsonData.output.text.join('\n'));

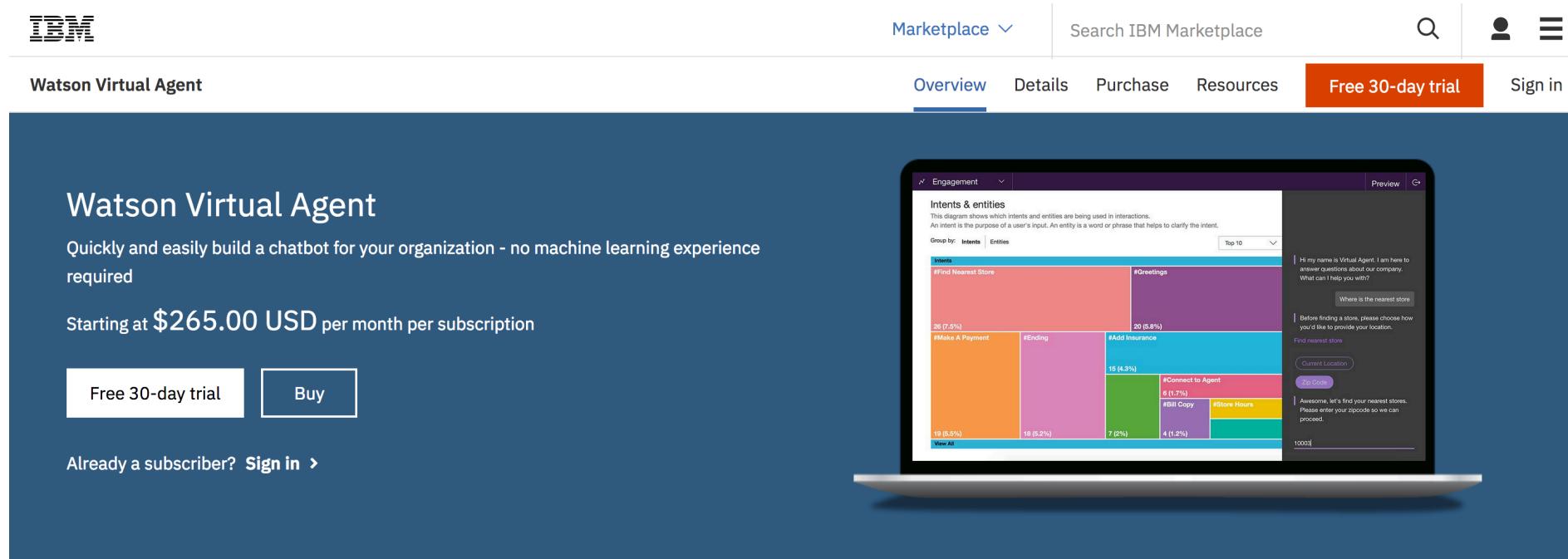
    if (message.watsonData.output.action === 'check_balance') {
      var newMessage = clone(message);
      newMessage.text = 'balance result';
      //send to Watson
      watsonMiddleware.interpret(bot, newMessage, function() {
        //send results to user
        processWatsonResponse(bot, newMessage);
      });
    }
  }
};

controller.on('message_received', processWatsonResponse);
```

Connecting Third-Party APIs



Pre-trained Virtual Agents



The screenshot shows the IBM Marketplace interface for the "Watson Virtual Agent" product. The top navigation bar includes the IBM logo, a search bar labeled "Search IBM Marketplace", and user icons for sign-in and account management. Below the navigation, there are tabs for "Overview", "Details", "Purchase", "Resources", a prominent orange "Free 30-day trial" button, and a "Sign in" link. The main content area features a large blue banner with the product name "Watson Virtual Agent". It highlights that users can quickly build a chatbot with no machine learning experience required, starting at \$265.00 USD per month per subscription. Two buttons are visible: "Free 30-day trial" and "Buy". Below the banner, a message asks if the user is already a subscriber, with a "Sign in >" link. To the right of the banner, a laptop screen displays the Watson Virtual Agent interface, showing an "Engagement" dashboard with a "Intents & entities" chart and a conversation log.

Watson Virtual Agent

Quickly and easily build a chatbot for your organization - no machine learning experience required

Starting at **\$265.00 USD** per month per subscription

Free 30-day trial Buy

Already a subscriber? [Sign in >](#)

Marketplace Search IBM Marketplace

Overview Details Purchase Resources [Free 30-day trial](#) Sign in

Engagement

Intents & entities

This diagram shows which intents and entities are being used in interactions. An intent is the purpose of a user's input. An entity is a word or phrase that helps to clarify the intent.

Group by: Intents Entities

Intents

Intent	Percentage
#Find Nearest Store	28 (7.3%)
#Greetings	20 (5.8%)
#Make A Payment	19 (5.5%)
#Ending	18 (5.2%)
#Add Insurance	15 (4.3%)
#Connect to Agent	8 (2.2%)
#Bill Copy	7 (2%)
#Store Hours	4 (1.2%)

Entities

Entity	Percentage
Where is the nearest store	100000
Before finding a store, please choose how you'd like to provide your location.	
Find nearest store	
Current Location	
Zip Code	
Awesome, let's find your nearest stores. Please enter your zipcode so we can proceed.	

Preview

Hi my name is Virtual Agent. I am here to answer questions about our company. What can I help you with?

Where is the nearest store

Before finding a store, please choose how you'd like to provide your location.

Find nearest store

Current Location

Zip Code

Awesome, let's find your nearest stores. Please enter your zipcode so we can proceed.

100000

What it can do for your business

Watson Virtual Agent is a new way to provide automated services to your customers. It offers a cognitive, conversational self-service experience that can provide answers and take action. You can easily customize your Watson Virtual Agent to fit your specific business needs, provide custom content and match your business brand. Additionally, deep analytics provide insights on your customer's engagement with the Watson Virtual Agent and help with the understanding of your constantly changing customer's needs.



Resources

- <https://www.ibm.com/watson/services/conversation/>
- <https://conversation-demo.mybluemix.net>
- <https://github.com/watson-developer-cloud>
- <https://www.ibm.com/us-en/marketplace/cognitive-customer-engagement>
- <http://developer.ibm.com/code/>