developerWorks®

Build an enhanced IT help desk chatbot on IBM i with Watson Assistant

Deploy a chatbot prototype that can understand and automatically solve password-related issues on IBM i

Christophe Lalevée

January 18, 2018
(First published December 20, 2017)

Chatbots are conversational robots that simulate conversation, and can interact with users in natural language. By harnessing both the power of the IBM Cloud Watson services and the power and openness of IBM i, you can open up your chatbot to countless creative applications (such as a virtual agent) that can understand and automatically solve password-related issues. This article guides you to implement a prototype of such a chatbot running on IBM i that links the IBM Watson Assistant service and IBM i to Slack, which is the messaging team collaboration tool.

Watson Conversation is now Watson Assistant. Although some illustrations in this tutorial may show the service as Watson Conversation, the steps and processes will still work.

Introduction

Did you know that as per recent studies, an average of 20% to 50% of all help desk calls are password related? Mundane tasks can be a headache for help desk teams, especially when higher priority jobs need attention. Virtual assistants (also known as *chatbot*) for IT help desk can automatically process tickets that do not need deep expertise such as managing access credentials and password-related issues, allowing your staff to focus on critical issues and value-added tasks. For instance, a chatbot can analyze the root cause and automatically solve password-related issues.

In this article, you will implement a simple chatbot to illustrate this kind of solution.

It can understand user profile-related issues, and, if possible, automatically solve the problem. As the icing on the cake, it can also answer to queries about system utilization, such as processor utilization or the percentage of system auxiliary storage pool (ASP) used.

This chatbot links the IBM® Watson® Assistant service and IBM i (back end) to Slack (front end), which is the messaging team collaboration tool. It will be implemented in the prototyping mode with Node_RED, and will run on IBM i.

You are now going to instantiate and configure the required services, import the Watson Assistant workspace (predefined dialog training) and Node-RED flow (chatbot back-end program), and test it!

What you'll need

To implement this chatbot on your IBM i system, you need:

- An IBM i 7.3 partition with Node-RED installed (if you need to install Node-RED on your IBM i, you can refer to my previous article Running Node-RED on IBM i: Installation and first flow, developerWorks, October 2017).
- An IBM Cloud (earlier known as IBM Bluemix®) account to be able to provision Watson Assistant services.
  Register for your free trial account or log in to IBM Cloud if you already have an account.
- A Twilio account (trial version available), and a Slack account (free version available).

Note:
If you do not want to create a Twilio account, you will still be able to run the chatbot program. In this case, the user will not receive the authentication code by text. The only way to retrieve the code will be to look at the program trace, in the Node-RED user interface, on the **Debug** tab.

Architecture overview

To implement this IT help desk chatbot on IBM i, four components must be deployed and configured. Figure 1 shows the components deployed to support the chatbot solution implemented in this article.

Figure 1. Chatbot architecture overview diagram

The four components are:

1. IBM Watson Assistant service providing natural language capabilities: intents/entities recognition and dialog management.
2. Slack software used as the user interface, in which you will create a Slack bot, a component required to interface Slack with your own chatbot program (#3).

3. Your chatbot back-end program, implemented with Node-RED on your IBM i system. It is the core of the solution, called *application logic*, linking all the components: Watson Assistant service (#1), user interface (#2), and IBM i system and database (#4).
4. IBM i system and the IBM Db2® database for retrieving and updating IBM i user profile (*USRPRF) attributes.
   In this prototype, the Node-RED program can directly query the database or run the program (integrated in #3). But, in a multi-system configuration, this component must be independent and deployed on all systems. Then data and services are available through web services [Representational State Transfer (REST) APIs].

In this article, let us see how to set up these components.

IBM Cloud: Deploy and train Watson Assistant service

In this first part, you will instantiate, train, and test Watson Assistant, (the Watson cognitive service in charge of dialog with a chatbot user).

The IBM Watson Assistant service combines machine learning, natural language understanding, and integrated dialog tools to create conversation flows between your applications and your users.

Perform the following steps to create your own Watson Assistant instance:

1. Log in to IBM Cloud using your IBM Cloud account.
2. Select the region and space you want to work in.
3. Click **Catalog**, then click **Watson** in the left navigation panel. The catalog displays the available Watson services. Click **Assistant** to view the service configuration page.

   Figure 2. IBM Cloud platform, Watson services catalog

4. Provide a unique service name (or use the default service name), and choose plan **Lite** (free). This plan allows 10,000 API calls per month, five workspaces, 25 intents and 25 entities. If you need more, you can subscribe to Standard or Premium plans. Click **Create**.

Figure 3. Assistant instance creation

5. In the new service dashboard that is displayed, click **Launch tool** to launch the web tooling interface where you can define intents, entities, and the dialog for conversation. A new page opens in your browser. Click **Log with your IBMid**.

Watson Assistant: Import a conversation

In this part, you will import a conversation flow to handle a dialog with a help desk chatbot user. The JSON file containing the dialog flow (**iHelpdesk.json**) is provided in the "Downloadable resources" section of this article.

The Watson Assistant tool page allows to create a new workspace or to import an exported one. A *workspace* is a container for the artifacts that define the conversation flow for an application.

Perform the following steps to import a conversation flow:

1. Import the iHelpdesk.json workspace by clicking the **Import** icon (highlighted on Figure 4).

   Figure 4. Watson Assistant tool

2. Click **Choose a file**, and select the iHelpdesk.json file, previously downloaded from "Downloadable resources".
   Then, click **Import**.

Figure 5. Importing a workspace

3. A new workspace is created and opened. To view the conversation dialog, click the **Dialog** tab.

   Figure 6. Watson Assistant workspace

Note:
A dialog consists of the following three types of data: • Intentions (the verb) Intentions represent the purpose of a user's input. You can think of intentions as the actions your users might want to perform in your application.Example: A user wants to know the value of a system resource • Entities (the name) An entity represents a term or an object in the user's text that provides clarification or a specific context for a particular intent. By recognizing the entities that are mentioned in the user's entry, the Watson Assistant service can choose the specific actions to perform to achieve a given intent.Example entity: System resource (such as processor, disk, memory, and so on) • The dialog itself The dialog defines the flow of your conversation as a logical tree. Each node of the tree has a condition that triggers it, depending on the input of the user.The purpose of the dialog is to lead to the answer to a question or the execution of an action or a command.

Watson Assistant: Understanding a conversation

You can have a look at the workspace content to get an idea of the questions you could ask to the chatbot (in the **Intents** and **Entities** tabs) or understand the way the dialog flow has been created (in the **Dialog** tab).

For example, you can try to understand the *Reset Password* flow implementation in the Watson Assistant service. Figure 7 illustrates how the **#lost password** intent dialog has been designed for this demo, where:

- Text within blue boxes are messages sent to a user
- Text within green boxes are actions identified by the Watson Assistant service that the chatbot program will have to run
- Orange squares are conditional statements
- Green circle is dialog entry point for this intent
- Red circles are dialog exit points for this intent

Figure 7. Reset password dialog flow

The chatbot program on IBM i requires the workspace ID and credentials to connect to the Watson Assistant API. To retrieve these values, click the **Deploy** icon and then click **Credentials** (highlighted in Figure 8).
Copy the workspace ID, user name, and password.

Figure 8. Watson Assistant credentials

Your Watson Assistant workspace is now configured. Intends, entities, and dialogs are ready to be used by your IBM i chatbot program.

If you want to learn more about Watson Assistant tool and interface, refer to the Watson Assistant documentation website.

Now, let's configure Slack, a software as a service (SaaS) collaborative solution that will provide the user interface of this chatbot.

Slack

Slack is a cloud-based set of team collaboration tools and services.

Slack works like an Internet RelayChat (IRC), organized in channels corresponding to many topics of discussion. The platform also makes it possible to keep track of all exchanges, allows file sharing within conversations, and integrates external services (such as GitHub, Box, Skype and so on) or your chatbot implemented with Watson Assistant. To integrate your chatbot, you need to use Slack bot, an integration solution that allows to communicate with your chatbot program which in turn communicates with Watson Assistant.

First create a Slack workspace or use an existing one if you have sufficient authorization. If needed, refer to Slack's how-to on creating a new workspace. During this process, you must enter a name that will identify your Slack workspace and define the workspace host name (by default, <WORKSPACE>.slack.com).

In this article, we will use **ihelpdesk** as the workspace name, and **ihelpdesk.slack.com** as the host name (don't forget to use your own details).

Open a new browser page on this host name to get access to your Slack workspace.

Figure 9. Slack Workspace on ihelpdesk.slack.com

Perform the following steps to create a custom bot in your Slack workspace:

1. Open a new page in your browser and navigate to the Slack Custom Integration-Bots creation page using the following URL: https://<*WORKSPACE*>.slack.com/services/new/bot (replace *WORKSPACE* by your own workspace name).

   Figure 10. Slack bot creation interface

2. Name the bot (you could change name after, if needed) and click **Add bot integration**. In this article, we will use **ibot** as the bot name.
3. A new page opens containing the API token (the bot's secret token).
   Token is used to connect your chatbot application (running on Node-RED on IBM i) to Slack servers by using a custom TCP/IP network API called Slack Real Time Messaging (RTM) API. This allows all user interactions with the Slack bot to be sent to your connected chatbot application.

Figure 11. API Token in Slack bot setting page

Copy this token because you will need it to configure Slack nodes in the Node-RED flow.

4. Switch back to the Slack workspace page.
   You can see a new app, the Slack bot (highlighted on Figure 12). It is currently not connected. It will appear as connected after creating the chatbot program with Node-RED (refer to the "Node-RED" section of this article).

   Figure 12. Slack new app available

   Your Slack workspace is configured. The Slack bot application (a virtual user) is now ready to be connected to the Node-RED application you will create.

Twilio

Twilio is a cloud communications platform as a service (PaaS) that allows developers to programmatically integrate voice, messaging, and VoIP into web and mobile apps.

You can access Twilio's services using web service APIs and you will be billed based on the usage.

In this article, we will use Twilio to implement a Two Factor Authentication (2FA) process. In this process, the user is authenticated first by his Slack ID and then by his phone number (a validation code is sent to the user's mobile number).

If you don't have an account yet, Twilio offers a trial version to all customers who sign up, which includes a free balance for you to experiment with. When your balance gets low, Twilio sends you an email with the information required to upgrade your project.

Create a trial account (or connect to Twilio using your existing account) and obtain a SMS-capable phone number.

Check the following references for detailed information about Twilio configuration:

1. Sign up for a Twilio account
2. Get an SMS capable phone number used to send text messages.

After completing the configuration, retrieve and copy the following information that will be required to configure the Node-RED `Twilio` node:

1. Account security ID (SID) and authentication token from the **Dashboard** page.

   Figure 13. Twilio account SID and authentication token

2. An SMS-capable phone number (see Figure 14). To access this, click **All Products & Services** on the home page, and then click **# Phone Numbers**.

   Figure 14. Twilio active numbers

Note: If you want to send text messages to another phone number, you must declare it on the Verified Caller IDs page.

Figure 15. Twilio callers

Your Twilio service is configured and ready to be used for authentication (2FA).

As all cloud external services are ready (Watson Assistant, Slack, and Twilio), you can now create the chatbot program with Node-RED on your IBM i system.

Node-RED

Node-RED is a flow-based programming tool, originally developed by the IBM Emerging Technology Services team (in early 2013) and now a part of JS Foundation.

Traditional development can be very technical, but Node-RED enables you to concentrate on the logic of your workflow, allowing fast prototyping.

If you want to know more about Node-RED and how to get started on IBM i, refer to my previous IBM developerWorks® article, "Running Node-RED on IBM i: Installation and first flow".

Install additional nodes in Node-RED

To implement this chatbot, you need to connect your chatbot program to IBM Watson Assistant (in the IBM Cloud platform), Twilio, Slack, and also locally to the IBM Db2 for i database.

By default, Node-RED doesn't install nodes for Watson software development kit (SDK), Twilio, Slack, or Db2 for i. So, you must now install them.

There are two ways to do that: Using the shell command line or the Node-RED UI.

First, let us install Db2 for i and Watson SDK nodes using the shell command line, then install Twilio and Slack nodes using the Node-RED UI.

1. **Install node-red-contrib-db2-for-i and node-red-node-watson packages**
   The `node-red-contrib-db2-for-i` package provides a node allowing to read and write to a local Db2 for i database from Node-RED on IBM i. Description can be found here.
   The `node-red-node-watson` package provides a collection of nodes for IBM Watson services. Description of packages and nodes, provided for Watson services, can be found here. Of course, it includes Watson Assistant node, which is the service supporting our chatbot dialog.

Using a node package manager (NPM), you can install Node-RED packages globally (thus making it available for all Node-RED users) or locally (making it available for a specific user environment only).
As these packages can be useful for every project. You need to do a global installation using the `npm` shell command.
Start a shell session. It can be Qshell or remote Secure Shell (SSH). I recommend using SSH. Read Jesse Gorzinski's article, "Eight Reasons to Embrace SSH" on IBMSystemsMag.com to be convinced.
To install the packages, run the following commands:

```
$ npm install -g node-red-contrib-db2-for-i
…
$ npm install -g node-red-node-watson
…
```

You can now see the packages installed in the *npm* global root directory:

```
$ ls /QOpenSys/QIBM/ProdData/OPS/Node6/lib/node_modules
node-gyp                        node-red              node-red-node-watson
node-red-contrib-db2-for-i      npm                   websocket-stream
```

Stop and start Node-RED to load new nodes.
Open your browser enter the URLhttp://<YOUR_SERVER_IP_ADDRESS>:1880 to open a page on Node-RED user interface.
In the palette (as shown in Figure 16), you can see the **Db2 for i** node and a new IBM **Watson** category.

Figure 16. Db2 for i and Watson nodes

2. **Install Slack and Twilio packages**

Package `node-red-contrib-slack` provides a Node-RED node to receive from and post messages to Slack. Description can be found here.
You can install it using the Node-RED UI. In this case, new nodes will be available only in the running Node-RED environment.
In Node-RED:

a. Click the menu icon at the upper-right corner and click **Manage Palette**.
b. Click the **Install** tab and search for the word, "slack".
   In the list that appears, search for `node-red-contrib-slack` and click **install**.

   Figure 17. Install Slack nodes

c. Wait for the installation confirmation message.

Next, let us install the Twilio node. The `node-red-node-twilio` package provides a Node-RED node to send SMS messages using the Twilio service. Install it the same way you did for `node-red-contrib-slack` :

a. Click the menu icon at the upper-right corner and click **Manage Palette**.
b. Click the **Install** tab and search for the word, "twilio".
   In the list that appears, search for `node-red-node-twilio` and click **install**.
c. Wait for the installation confirmation message.

Let's now restart Node-RED. When restarted, take a look at the palette. In the **Social** category, three Slack nodes are now available. In the **Mobile** category, one Twilio node is now available.

Figure 18. Slack and Twilio nodes

Import flow and configure nodes

You are now going to import the chatbot flow and configure it to use your own services (Watson Assistant, Slack, and Twilio previously configured).

You need to get the (JSON) code, import it into Node-RED on IBM i, and deploy it.

Get the code.
To get the code, download the Node-Red_chatbot_flow.v1.1.json file from the "Downloadable resources" section of this article, and copy the entire file's content into clipboard.

Import the code.
Go back to the IBM i Node-RED editor in your browser, click the menu at the upper-right corner and click **Import -> Clipboard**. Then, paste the contents of the Node-Red_chatbot_flow.v1.1.json file content that you copied. Click **New flow**, and then click **Import**.

Figure 19. Import chatbot flow

Click somewhere on the blank sheet to paste nodes.

Have a quick look at chatbot flow. Figure 20 highlights the main logic of the flow.

Figure 20. Chatbot flow overview

The numbering in the figure corresponds to the numbering in the following procedure that explains the flow.

1. Get the text entered by the user on Slack.
2. Send it to the Watson Assistant service. This service understands the intent, identifies entities, defines what to answer.
   It sends back an answer for the Slack user and initializes variables in the context part of the JSON object exchanged. The value of `context.Action` variable indicates whether an action has to be executed by the chatbot (and what action) or not.
3. Depending on the `Action` variable value, the switch case node will route to perform one or two steps (from within step 4 to step 8).
4. Answer directly to the Slack user.
5. Retrieve resource usage value from IBM i (by querying Db2 for i services).
6. Generate a temporary password and an authentication code, and send text.
   If you don't want to use SMS to send code, you can configure the `email` node to send it by email.
7. Change the user profile (*USRPRF) IBM i object. First test if password reset is allowed (password not *NONE), and then run the `system` CHGUSRPRF shell command.
8. Retrieve the user profile parameters from IBM i (by querying Db2 for i services) and determine the possible cause of the connection issue.

After performing steps 5, 7 or 8, the result is sent to Watson Assistant to transcript in a natural language sentence.

`Link` nodes (#a, #b) allow to make a connection back to the beginning of the flows (from a to b), without overwriting the flow representation. `Link` nodes allow to set up a mechanism like the symbolic links of UNIX file systems.

Update the code.
You must now configure nodes corresponding to cloud external services previously created. Figure 21 shows the nodes to be configured.

Figure 21. nodes to be configured

1. Watson Assistant
   Double-click the `Conversation-Slack` node (Figure 21, #1), and enter the **user name**, **password** and **workspace ID** (captured in the "Watson Assistant: Understanding a conversation" section of this article.
2. Slack
   Two Slack nodes must be configured: `Slack bot in` and `Slack bot out`, and they both have the same name: `helpdeskbot`.
   Double-click each node (Figure 21, #2), and enter the bot API token (captured in the "Slack" section of this article).
3. Twilio
   To configure the `Twilio` node (refer Figure 21, #3a), double-click it, and select the `External service` value for the **Service** field.

   ## Figure 22. Twilio node configuration

   Then, click the **Edit** button (highlighted in Figure 22) to add a new Twilio API (twilio-api).

Figure 23. Twilio node account configuration

Enter the required values in the **Account SID, From, Token** and **Name** fields (see Figure 23). You should have captured those values in the "Twilio" section of this article.
Then, configure the function node (Figure 21, #3b), named `Build SMS`.
This node build message to be sent by text, and also defined the target phone number (this is only a simple demo, but think about dynamically retrieving this phone number from an enterprise user directory, for instance, Lightweight Directory Access Protocol (LDAP) or IBM Directory Server).
Initialize the `msg.topic` variable with the phone number to which you want to send messages. Remember to replace xxxxxx with this phone number in the code (see Figure 24).

Figure 24. Twilio function node

Note:
The two Db2 for i nodes are configured to use the *Local connection to the IBM i database. User profiles used to launch Node-RED will be used for Db2 authorizations. Maybe, for security reasons, you will have to reconfigure the Db2 for i nodes to declare the credentials of a specific user profile to be used for Db2 connections.

Deploy the code.
Click **Deploy** to deploy and make your application live.
A message, **Successfully deployed**, appears at the top of the window.

You can now test the chatbot.

Test chatbot

Go back to the Slack window, and see that the virtual user is now connected (as shown in Figure 25).

Figure 25. Slack bot connected

This chatbot was trained to answer the following four user intents:

- "Know {resource} level usage" where, {resource} can be a processor or ASP. For example, refer Figure 26

  Figure 26. Query resource usage

- "Change lost password". For example, refer Figure 27.

  Figure 27. Solve connection issue (password related)

- "Solve a connection issue". For example, refer Figure 28.

Figure 28. Solve connection issue (password related)

These are just examples of resolving password-related issues. You can find many ways to improve it by changing the Watson Assistant dialog flow or the Node-RED flow.

Remember that the quality of chatbot is mainly related to the quality of the Watson Assistant training. And, as you could see in the "Watson Assistant: Import a conversation" section, the current training (number of sample sentences by intent) is not sufficient. You can add more sample sentences to train Watson Assistant better.

So, now, it is your turn.

Switch back to the Slack interface.

Click the iBot app on the left pane, and enter your questions.

For exhaustive tests, don't forget to change the parameters of test user profile (*USRPRF on IBM i), for example, password *NONE or expired, user profile *DISABLED due to too many wrong passwords entered, and so on.

Conclusion

In this article, you could see that it is quite easy to create a chatbot to automatically support simple ticket resolution tasks.

This easiness is due to three main factors:

- Watson Assistant in IBM Cloud, providing a powerful tool to design and support dialogs in a natural language
- Node-RED on IBM i, a Node.js development application allowing quick and easy prototyping
- Db2 for i services, simplifying access to many pieces of IBM i system information through a database interface rather than complex commands or APIs.

So, next, what will you request to your chatbot to do? Solve printer issues, the second source of tickets in IT help desk organizations? Provide FAQ interface to your IBM i users?

There are so many ideas and their implementations are now possible on your IBM i. Let's do it!

Related topics

IBM Watson Assistant

- Watson Assistant page
- Watson Assistant documentation
- Creating a chatbot with cognitive technologies, IBM developerWorks - IBM Code

Node-RED

- Node-RED official website
- Running Node-RED on IBM i: Installation and first flow, IBM developerWorks, October 2017, by Christophe Lalevée
- Build your first social media Dashboard in minutes with Node-RED and Db2 for i, IBM developerWorks, November 2017, by Benoit Marolleau

DB2 for i services

- Replacing API Calls With DB2 for i Services, IBM Systems magazine, January 2016, by Jon Paris, Susan Gantner
- Db2 for i services, IBM Knowledge Center IBM i 7.3

Downloadable resources

Description	Name	Size
Watson Conversation workspace	iHelpdesk.zip	8KB
Node-RED chatbot flow for IBM i	Node-Red_chatbot_flow.v1.1.zip	4KB

© Copyright IBM Corporation 2017, 2018
(www.ibm.com/legal/copytrade.shtml)
Trademarks
(www.ibm.com/developerworks/ibm/trademarks/)