## Define the Chatbot Flow Using BotML

In this lab, you will add BotML to your MasterBot. The code will later on support each of your intents. However, we will be focusing on the **Balances** intent where we will add its variables and then add all the states that are needed to complete its actions. When you’re done, you will test the intent to make sure that it work as expected.

## Before You Begin

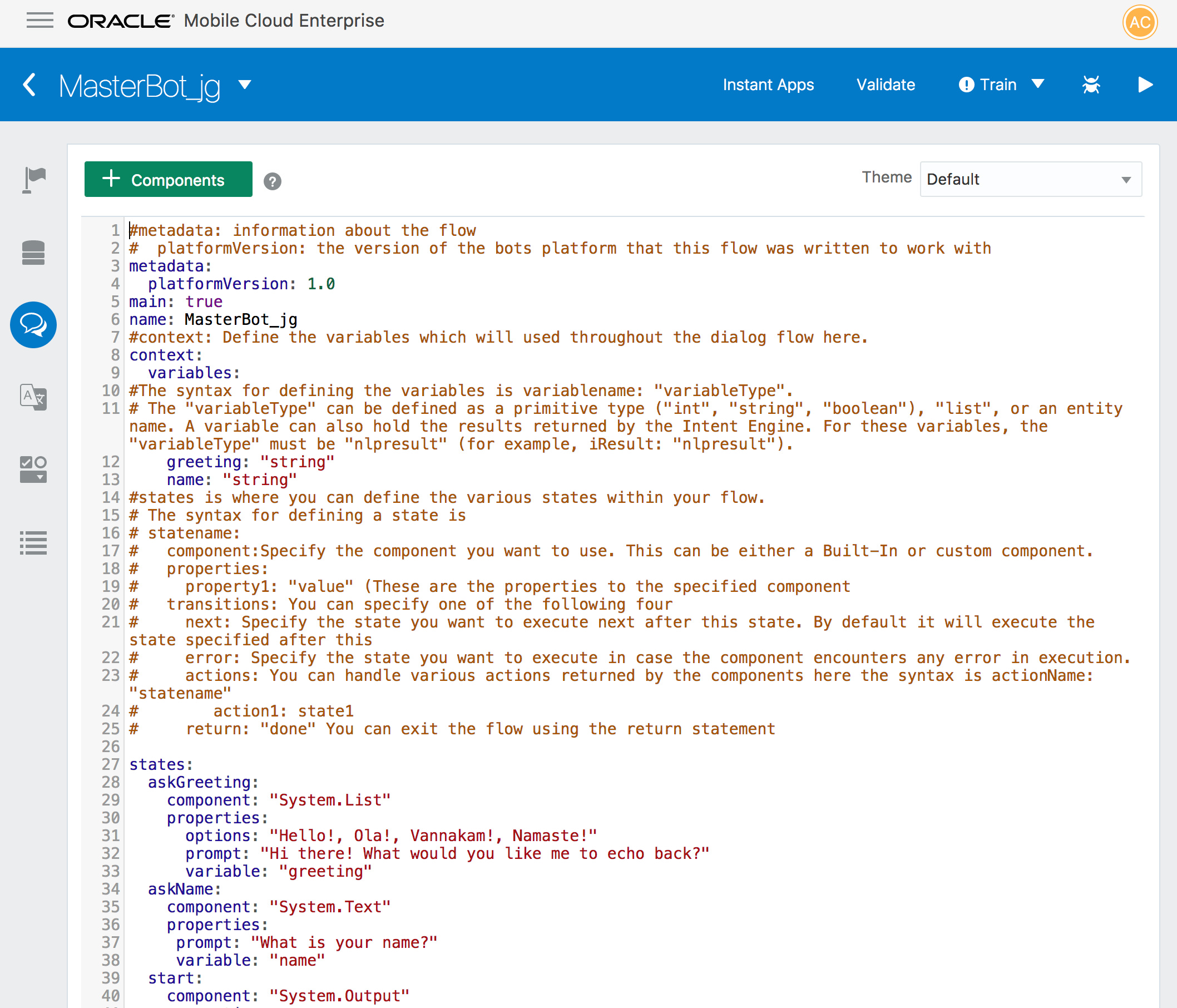
You need the following files for this lab, which are located in the /labfiles/code directory:

* FirstBotYAML-Balances.txt
* FirstBotYAML-Complete.txt

### Step 1: Include the Code that Supports the Balances Intent

In this section, you add code to the flow of your chatbot that supports the Balances intent. We'll examine its components and use it as an example of how you should set up other components.

1. Log in to your MasterBot\_firstNameLastName chatbot, and click on the Flows icon in the left navbar. You will see the default BotML “Hello” code in the editor. We will not be needing any of it, so delete it (click in the editor -> ctrl+a -> Delete).



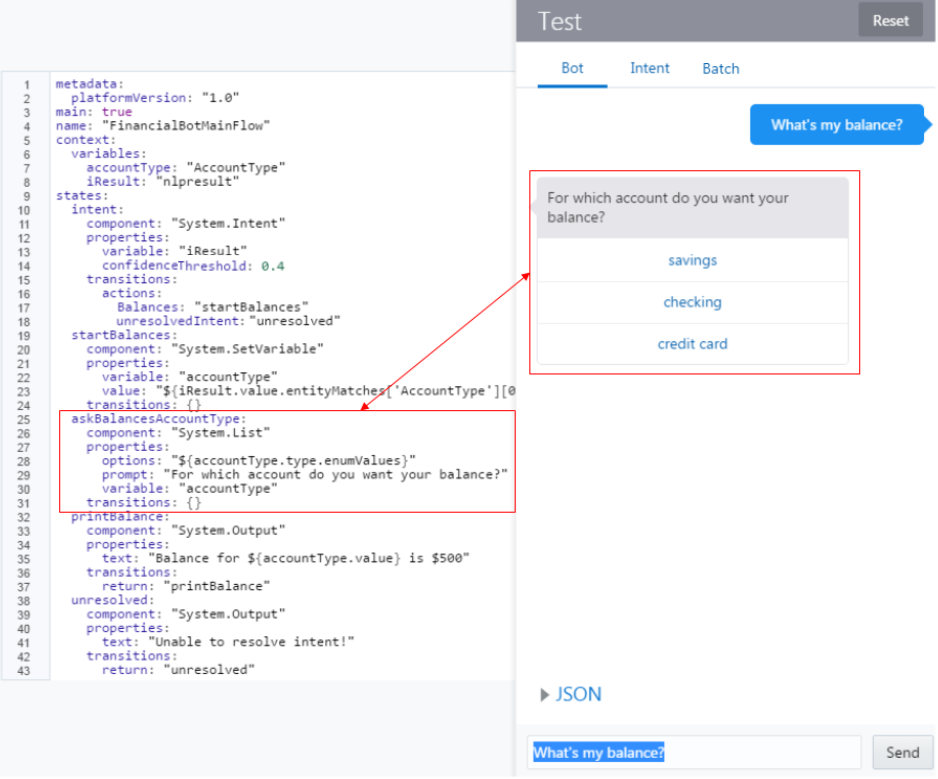
1. If you haven’t already done so, locate the FirstBotYAML-Balances.txt in the /labfiles/code directory, open it, copy its contents into the editor and then click **Validate**.



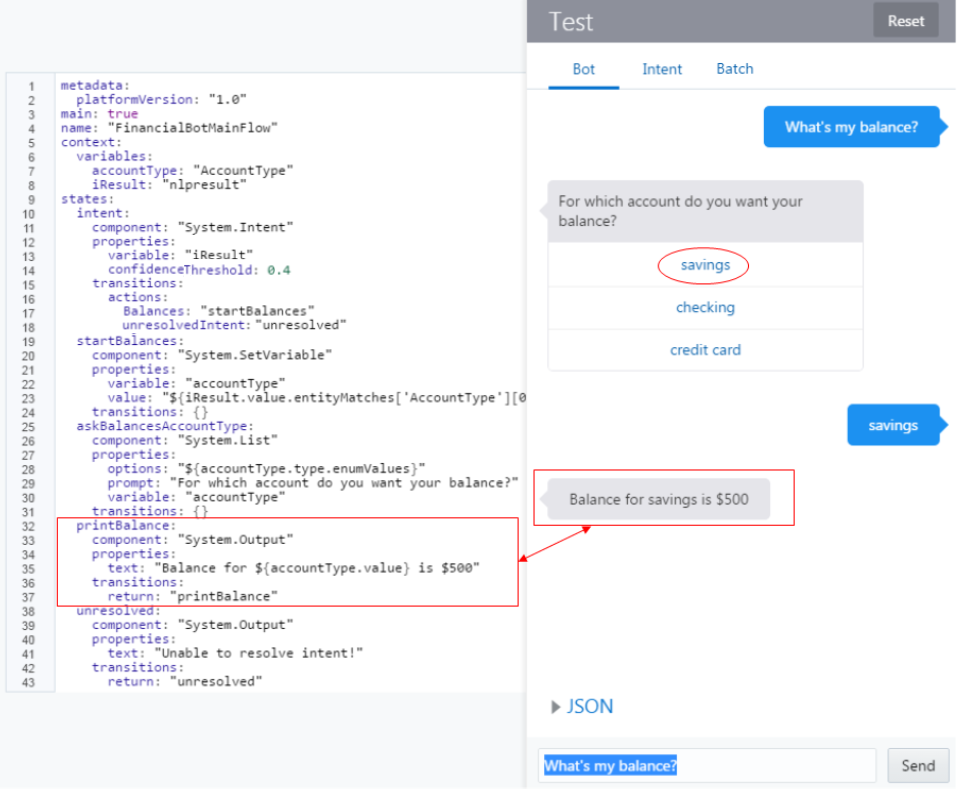
1. Let's look at the code we have and dissect what it represents.
   * Here we see a variety of sections to the code: the header followed by the declaration of content variables and then the intent states.
   * The accountType variable is used for recording and displaying the account for which the balance is requested. It's used in the startBalances state to check if what the user has typed in matches any of the values included in the entity definition. It is also used in the askBalancesAccountType state to store what is entered if the account type is not specified in the startBalances state. Finally, the account type is used in the printBalance state when the balance is displayed to the user.
   * There are five states defined. The intent state is the result of Intent classification and entity resolution that’s provided by the Intent Engine as the nlpresult (of which iResult is a type). In other words, this variable (iResult) holds the result of the Intent Engine (that is, the intent and entity resolution) from the input text provided by the user. The actions show a startBalances state which implements the Balances intent. There is also another state listed that’s used when the intent cannot be resolved: at the end of the flow, notice the unresolved state, which is reached if no other state is fulfilled.
   * Below the intent state are all the other states that the chatbot uses. In our case, the startBalance state is the entry point for Balances, regardless if the accountType variable is set in StartBalance, or if the nlpresult (that is, the iResult variable) provides this value from parsing user input. In this case, askBalancesAccountType does not attempt to set the variable as it’s already been set (and as a result, the list of options will not be displayed). If the value isn’t set in the nlpresult, however, the flow attempts to set it using list of options. If an account type is not mentioned in the user’s message, then a System.List component displays the valid account types as options. Once an account type is set, the control of the flow drops down to the printBalance state, where a System.Output component is used to display what the account type is and the balance amount. The final state is entered if the intent cannot be resolved.



1. To test the flow code, click the **Play** button in the upper right, select the Bot tab in the Tester.
2. Enter *What's my balance?* in the Message area, and then click **Send**. You should see a list of all the accounts included in the System.List component.
3. Notice the prompt and the list of accounts. They originate from the askBalanceAccountType state.



1. Next, select an account. The return should display the System.Output component from the printBalance state and show you the amount in that account. Notice that the displayed value of $500 is hard-coded in the System.Output component (“Balance for ${accountType.value} is $500”).



1. Click **Reset** and then try out some other messages, including some with the account in the message text to see how your chatbot responds. (E.g “How much do I have in savings?”)

### Step 2: Include the Code that Supports the Send Money and Track Spending Intents

Before we finish, we will also insert he BotML to cover our other intents, but we won’t cover them now.

Locate the FirstBotYAML-Complete.txt in the /labfiles/code directory, open it and copy its contents. In the Flow editor, remove ALL current content and paste the copied BotML and then click **Validate**.

Good for you! You have now completed this lab. In the next lab, you’ll learn how to add custom components to your chatbot.