5. Exposing the sample application to Amazon Alexa

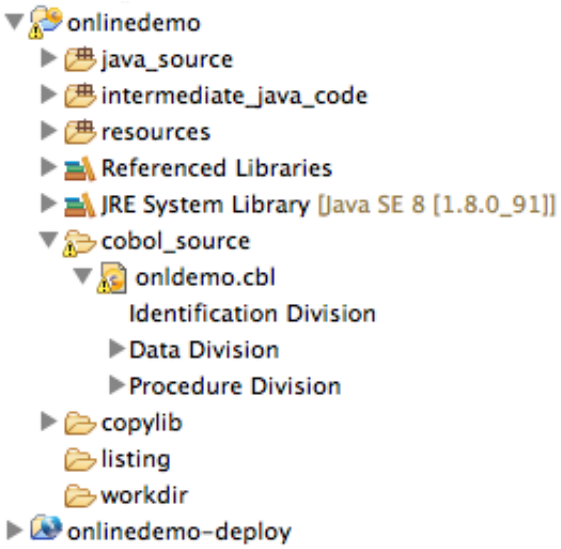
These steps assume you have an Amazon AWS account, access to an Amazon Echo or Dot and are somewhat familiar with the AWS console.
They also presume you have followed the previous 4 documents and configured the sample applications on an Heirloom PaaS instance.

Additional sources for these steps can be found here: http://www.elasticcobol.com/downloads/demo/AlexaDemoSources.zip

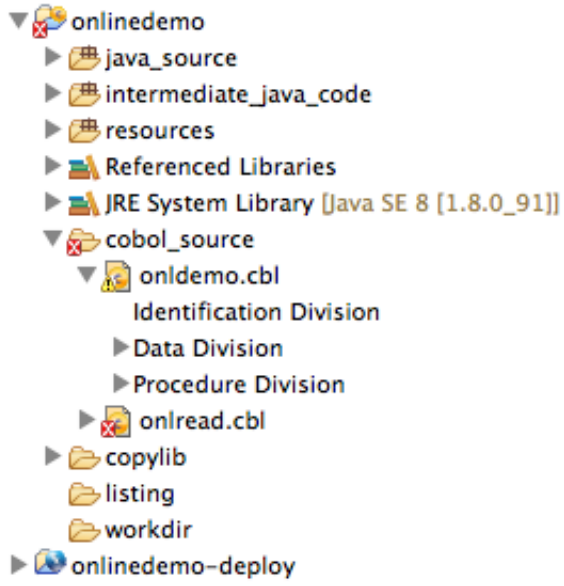
Modify and deploy sample:

Before we can connect and Alexa app to the CICS application we need to add a new screen that can READ the database.

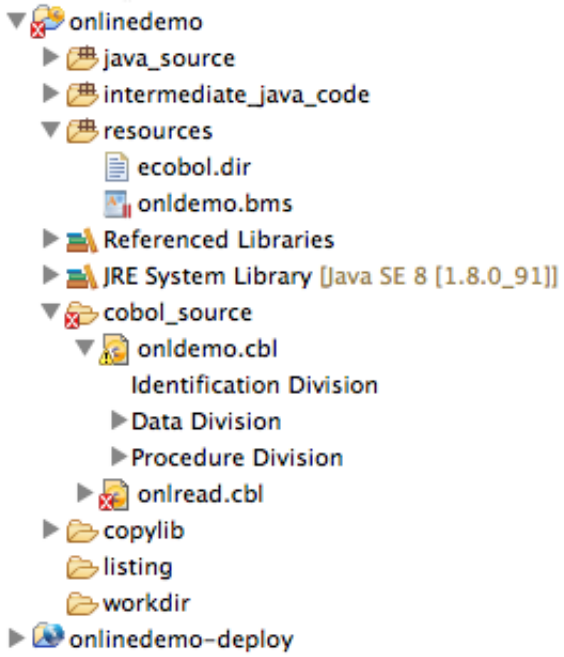
Open Eclipse and expand your onlinedemo project:



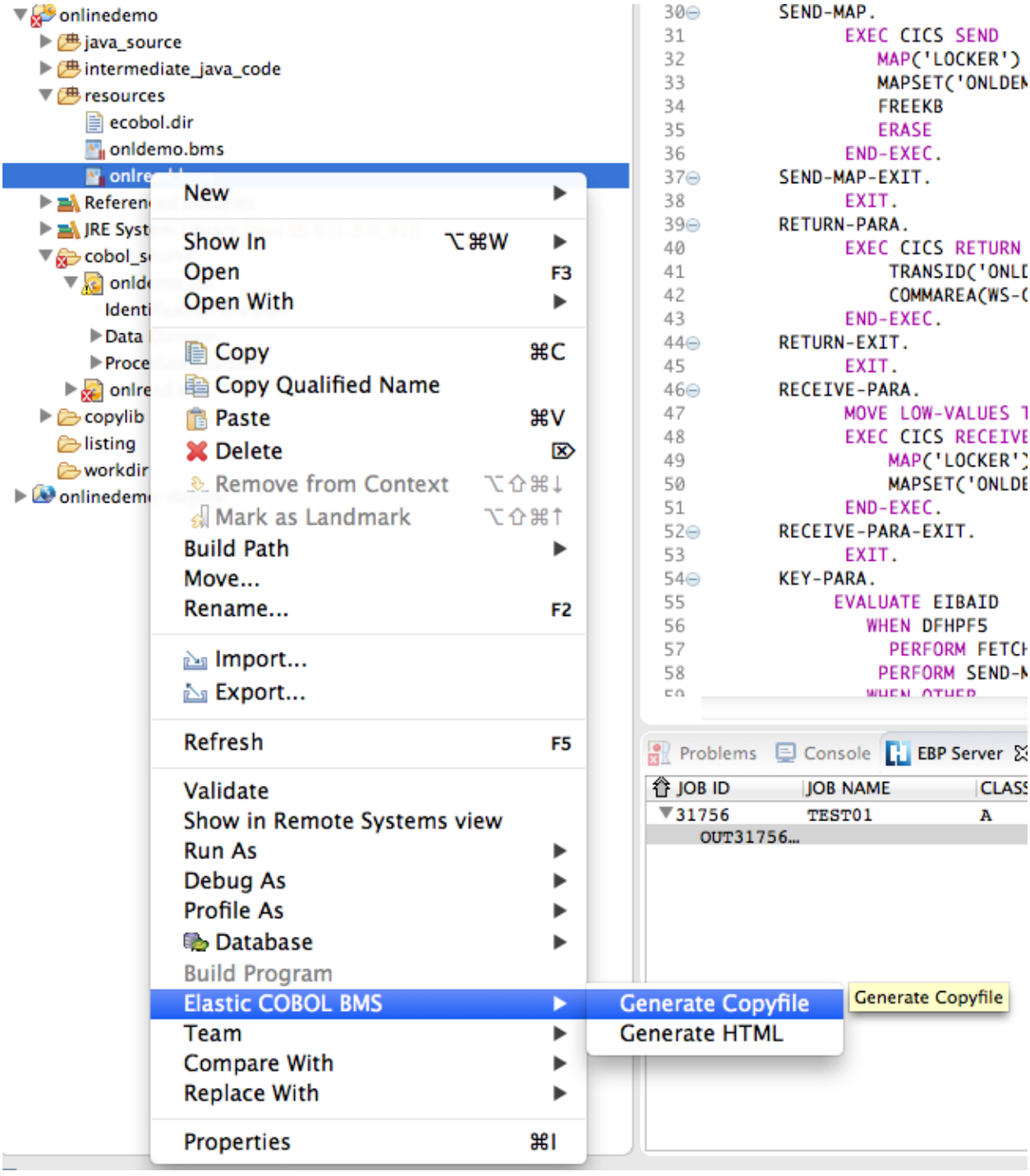
From the source folder, drag and drop **cobol_source/onlread.cbl** onto the cobol_source folder and choose ‘Copy files’ when the dialog appears:



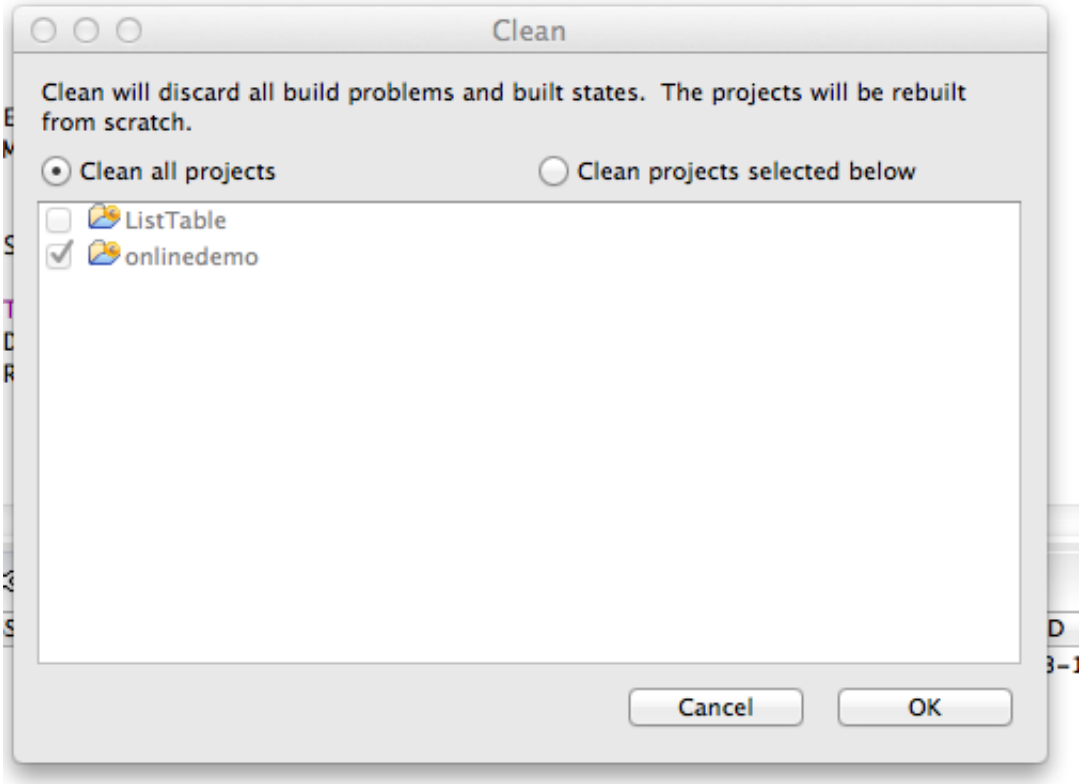
Drag and drop **onlread.bms** from the resources folder of the sources onto the resources folder of your project and again, choose 'Copy files':



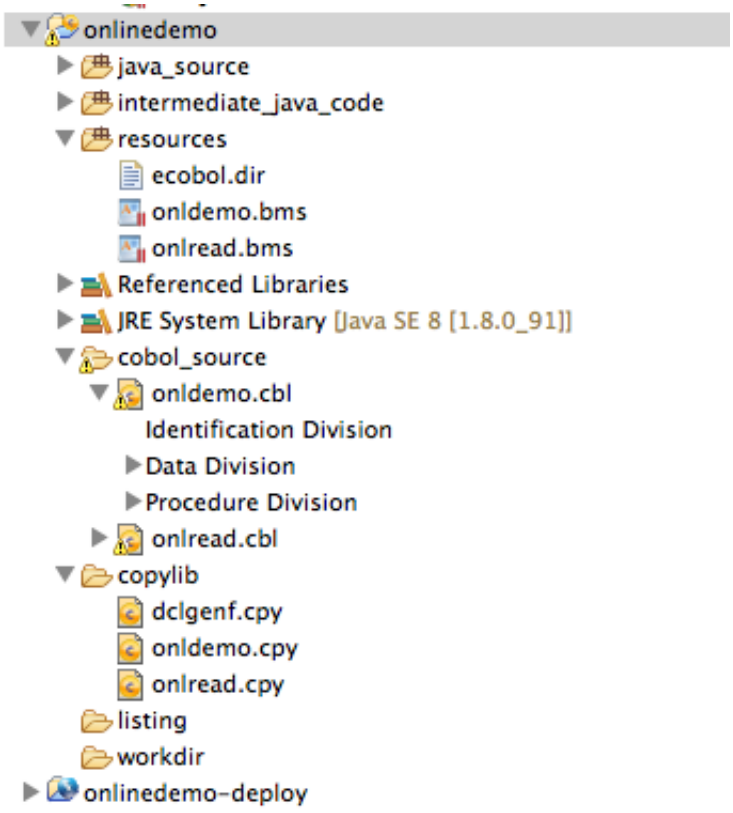
Right click **onlread.bms** and choose 'Generate Copyfile' from the 'Elastic COBOL BMS' menu option:



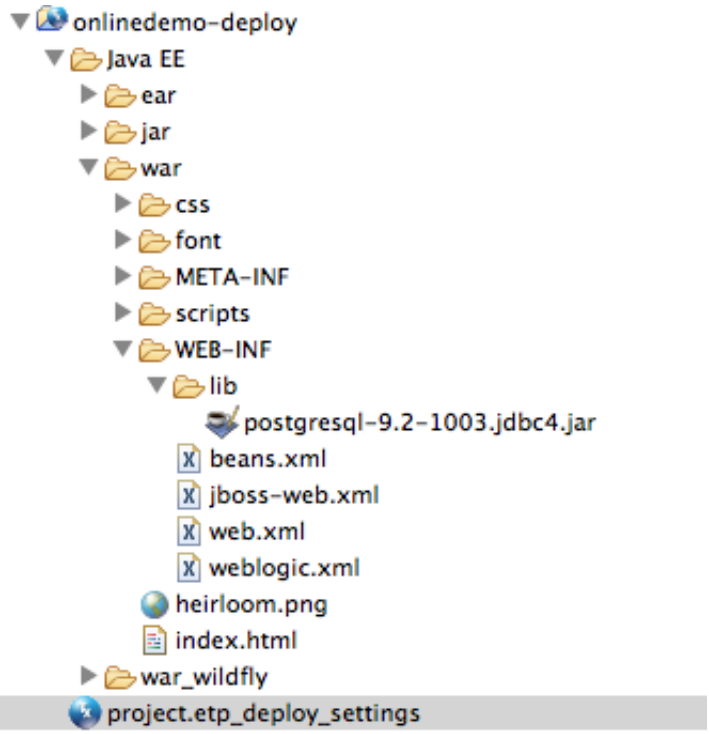
Click the **onlinedemo** project, and then from the Project menu click ‘Clean..’:



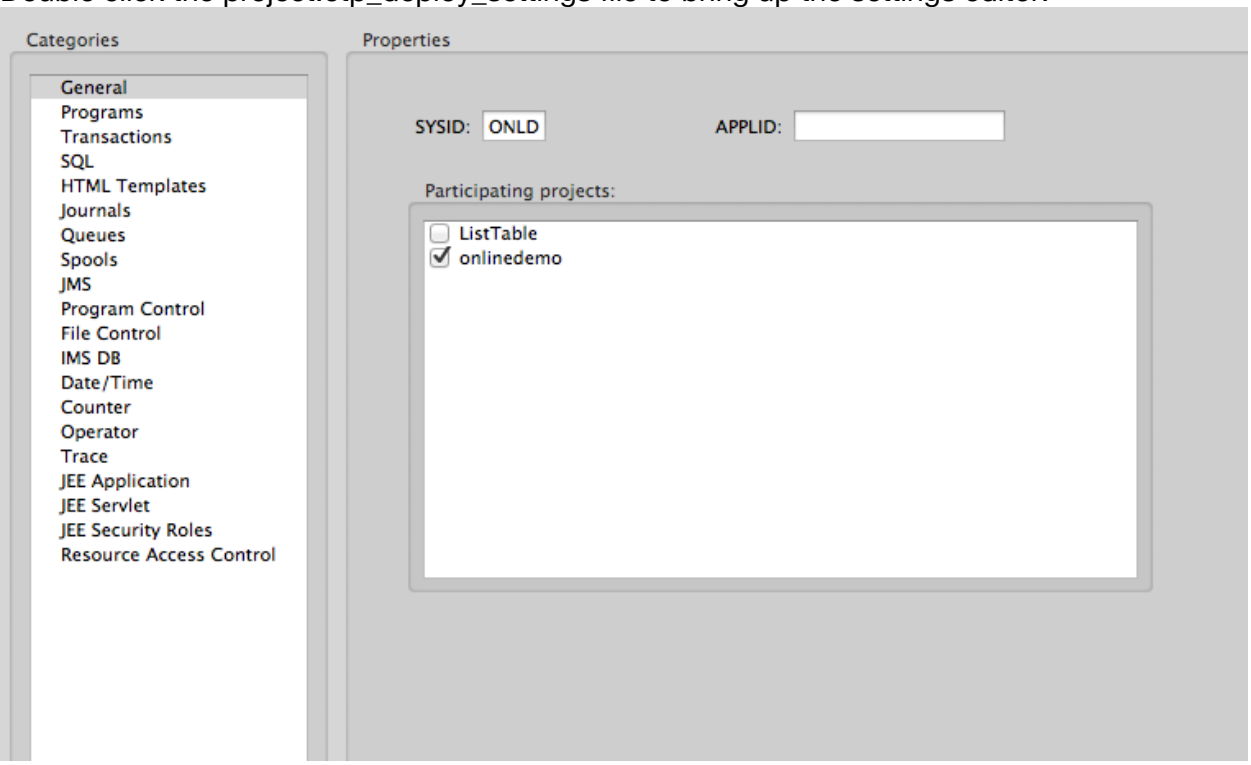
Ensure **onlinedemo** is selected and click OK to rebuild the project:



The project should have no errors listed.
Expand your onlinedemo-deploy project:

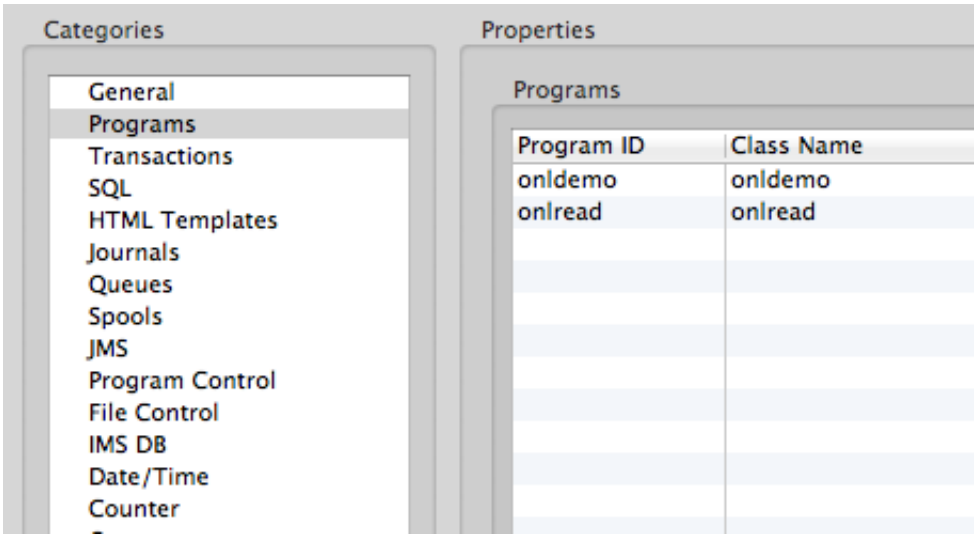


Double click the project.etp_deploy_settings file to bring up the settings editor:

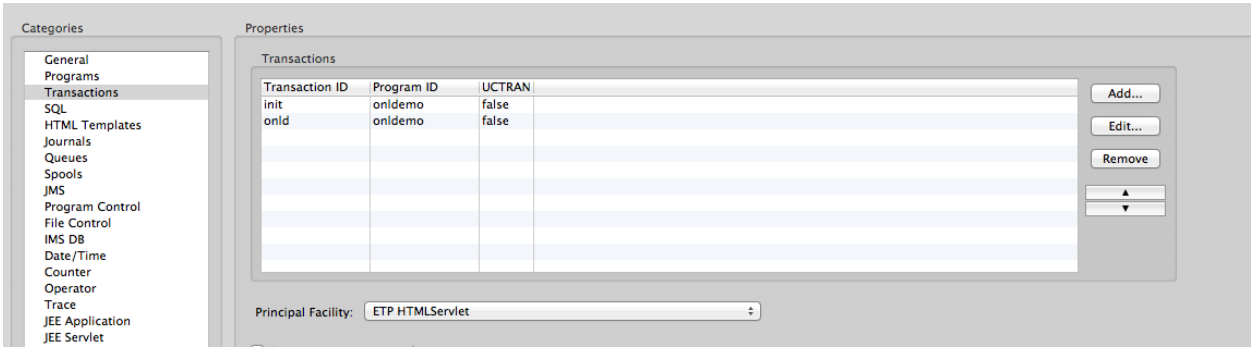


uncheck and recheck the ‘online demo’ project and click ‘Apply’. This will ensure the project

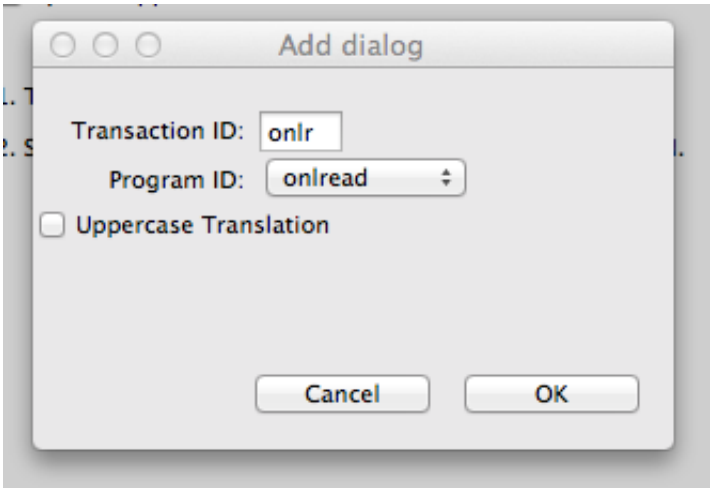
sees all the new programs.
Confirm the new **onlineread** program is there by click ‘Programs’ on the left menu:



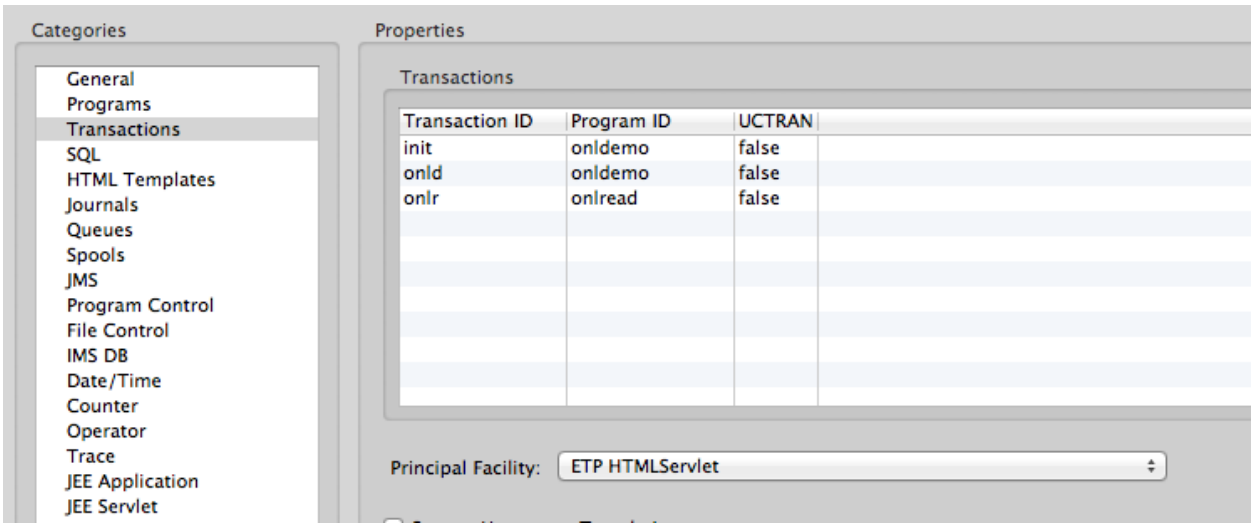
Click ‘Transactions’ in the left menu:



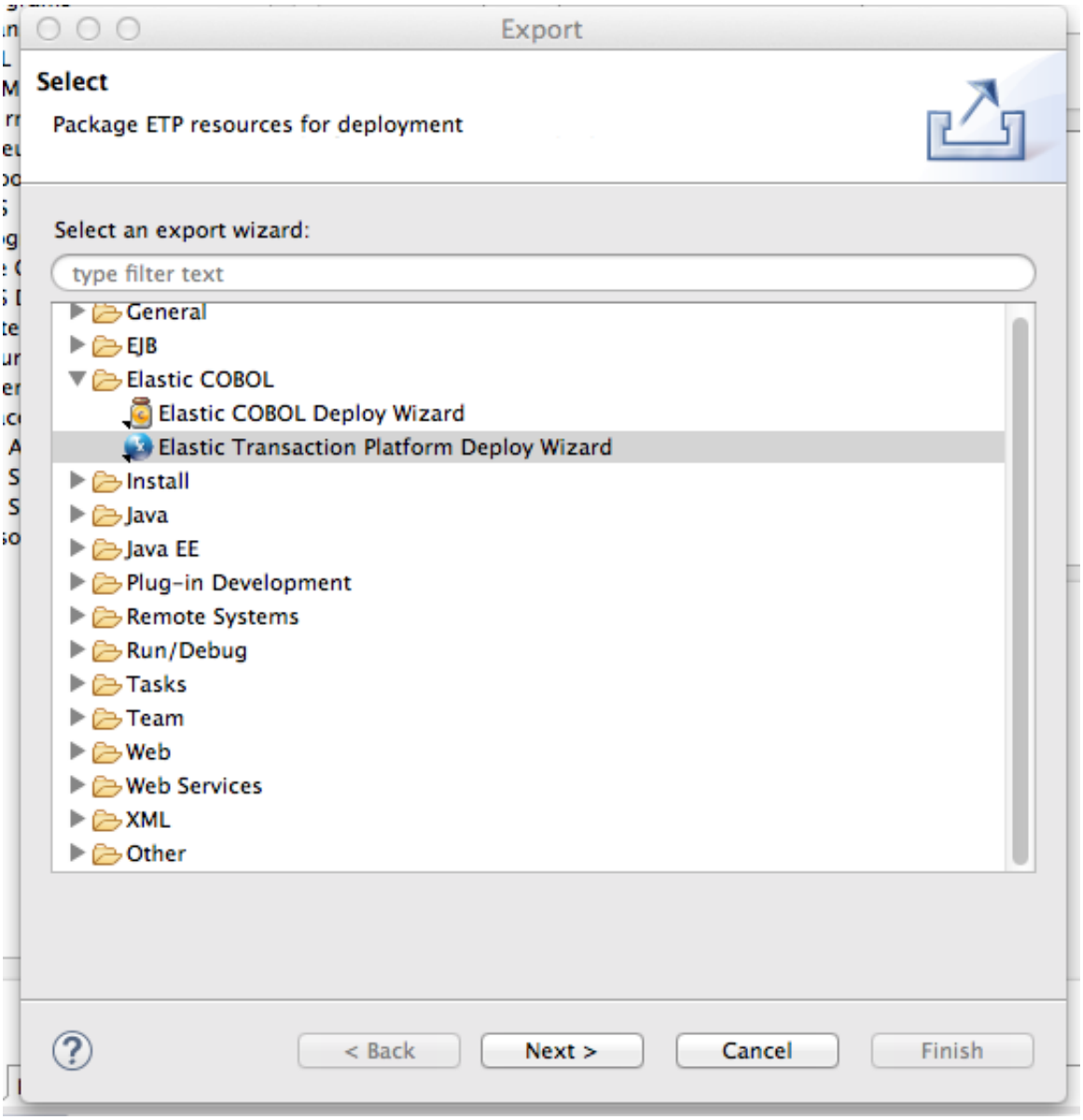
Click ‘Add’ and enter **onlr** and choose the **onlread** program ID:



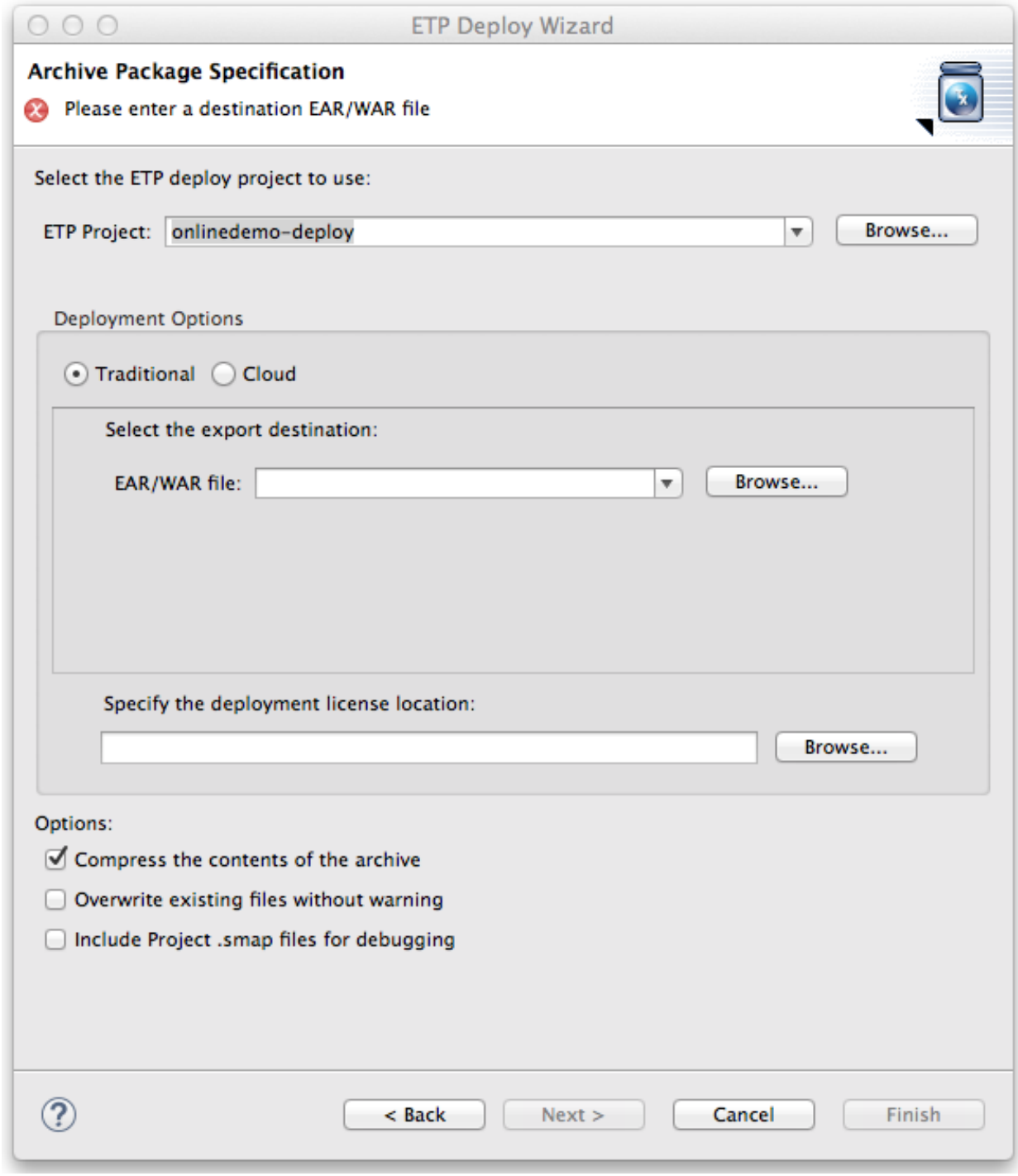
Click ‘OK’:



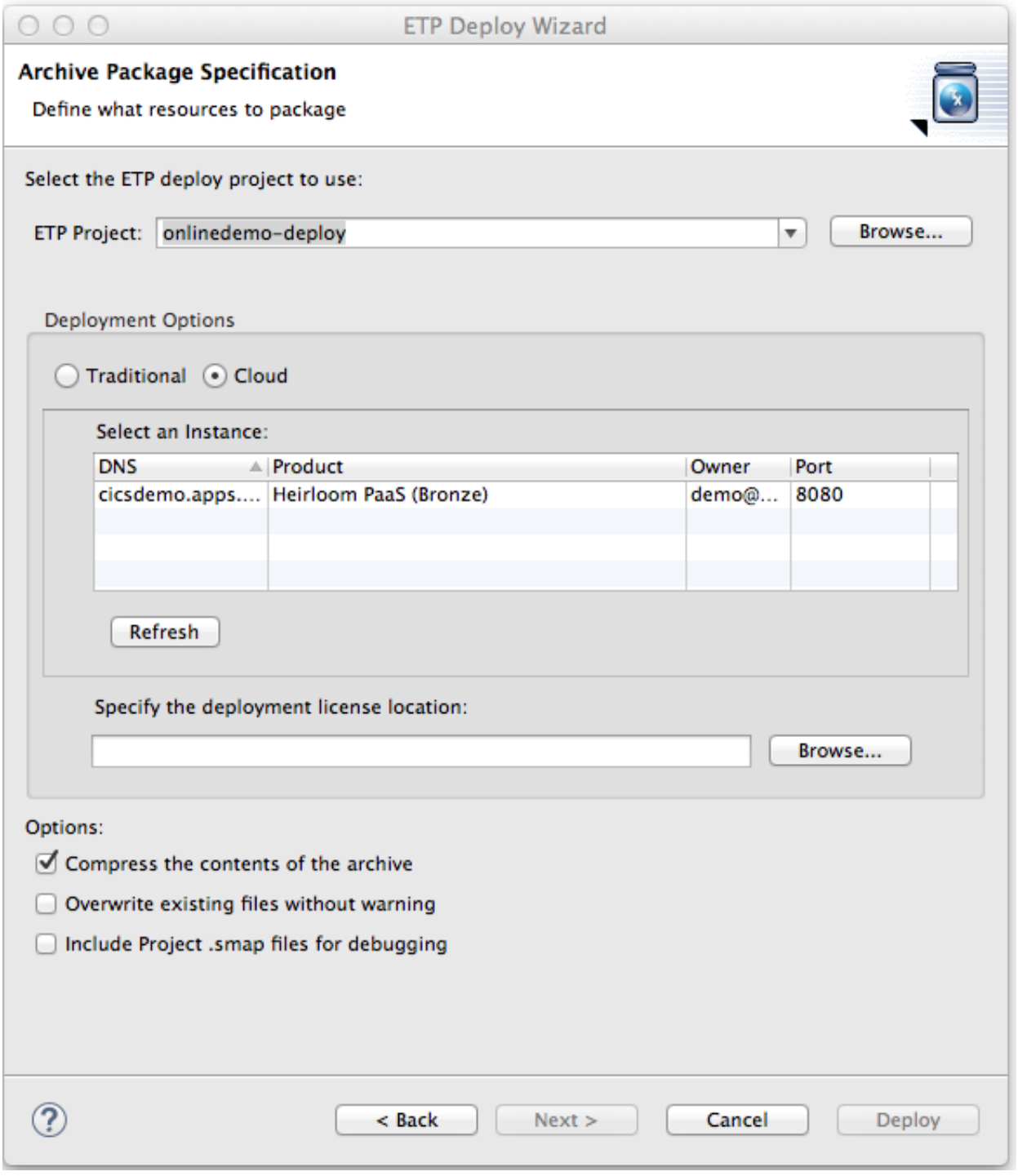
You are now ready to re-export the converted CICS application with read functionality.
Right click the onlinedemo-deploy Project and select Export...:



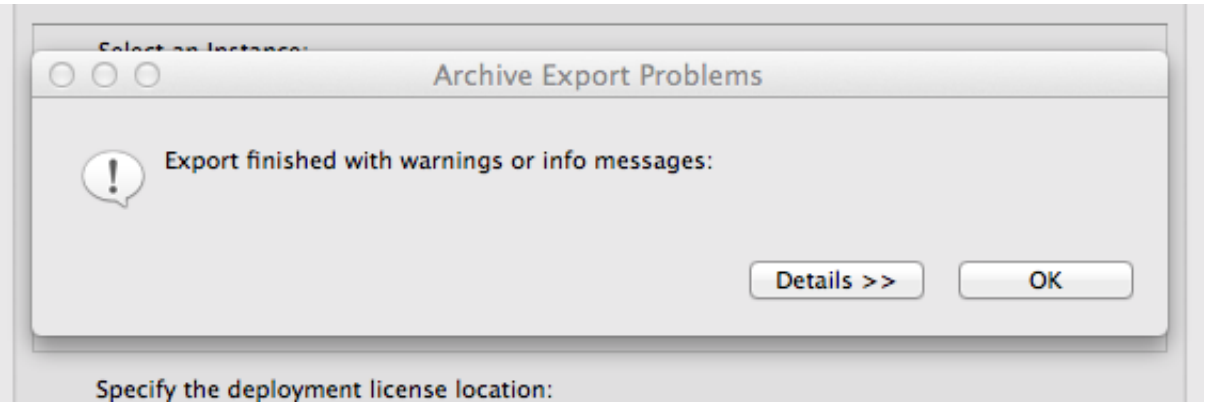
Select the ‘Elastic Transaction Platform Deploy Wizard’ and click Next:



Click the ‘Cloud’ radio button:

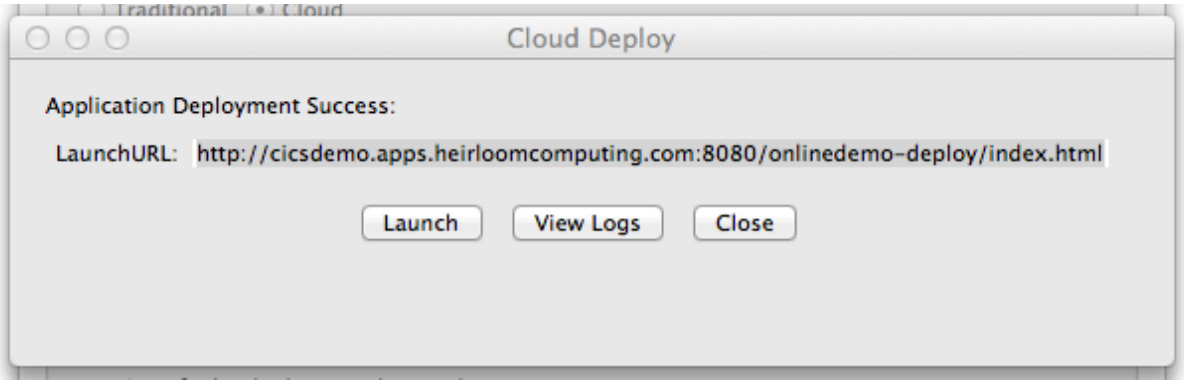


Select your application instance and click Deploy:



Click ‘OK’ to clear the info messages dialog.
The deploy wizard will deploy the application to your PaaS instance, this may take several seconds.

Once it is done you will see this dialog:



Click the ‘Launch’ button:

HEIRLOOM COMPUTING

Your ETP Project Home Page

This is the **index.html** page for your deployed Elastic Transaction Platform project. Change this page or go direct

- http://cicsdemo.apps.heirloomcomputing.com:8080/onlinedemo-deploy/servlet

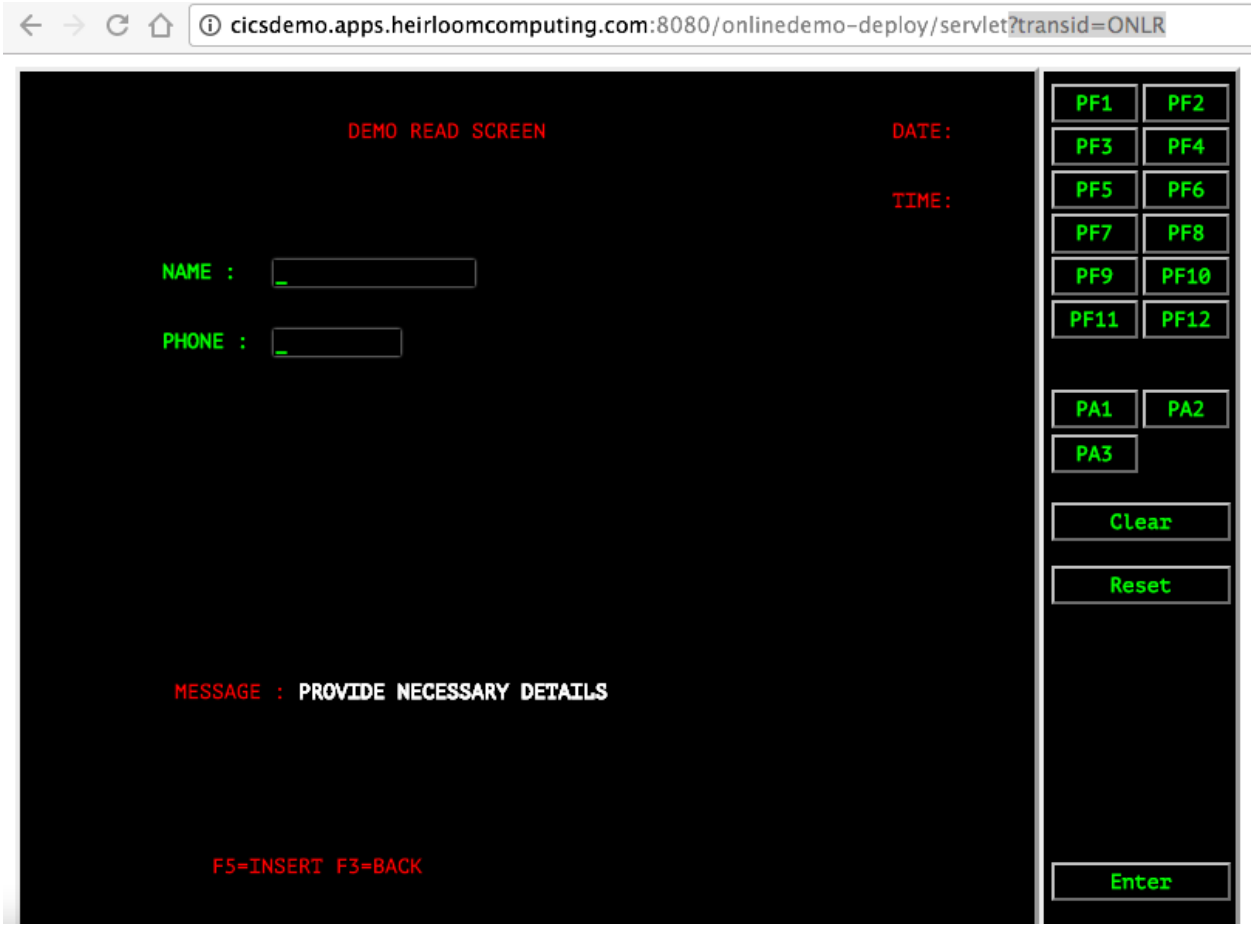
The **index.html** and other static content such as the cascading style sheets that control the look and feel of your sc
/servlet portion of the URL in the ETP Deploy Settings file. For servlet container deployments the URL is determ

Control will transfer to this region's INIT transaction screen in 4 seconds.

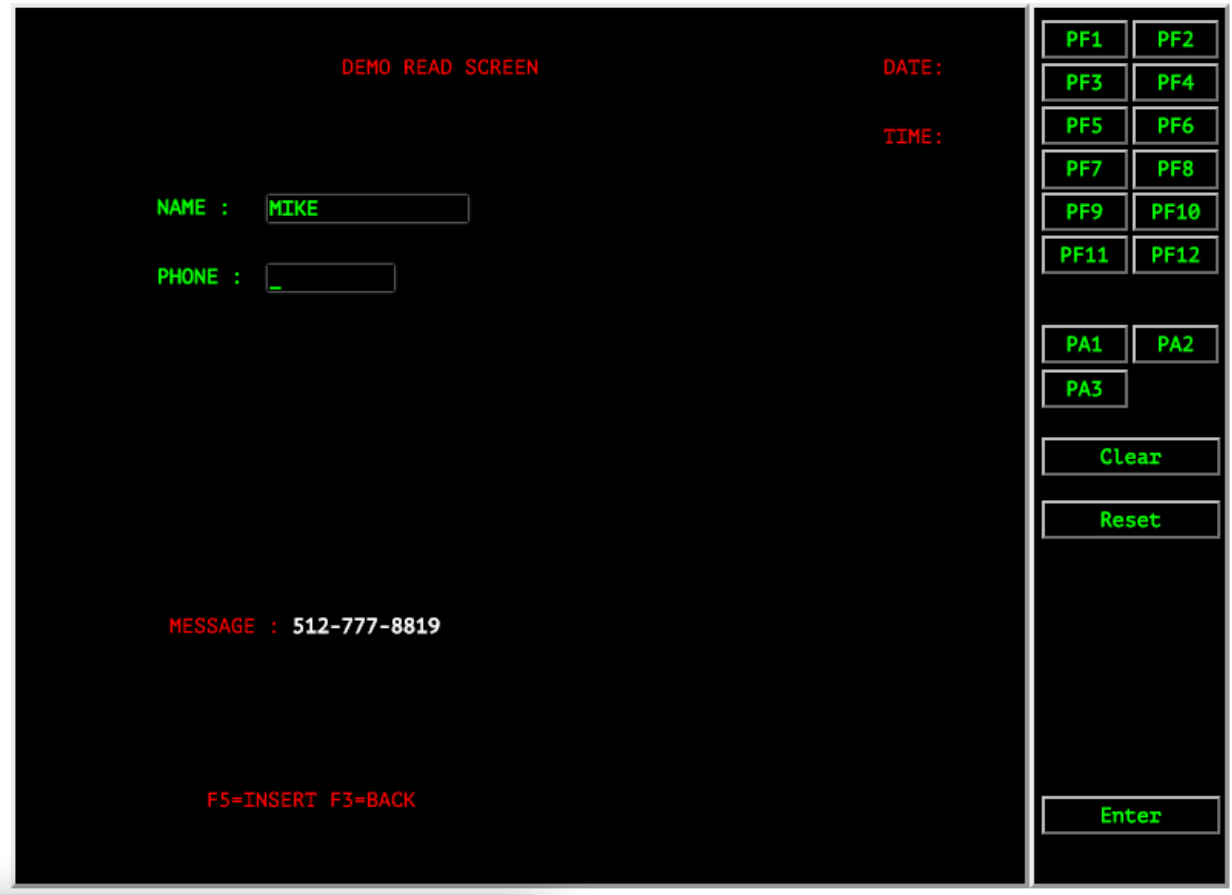
See support.heirloom.cc for more information.

This will bring up a browser window with your applications index.html. You may wait 10 seconds or click the url for the servlet to start the application:

Modify the URL to have ?transid=ONLR on the end of it and press Enter to load the READ screen:



Enter the name of someone you added to the database in the earlier demo and press PF5:

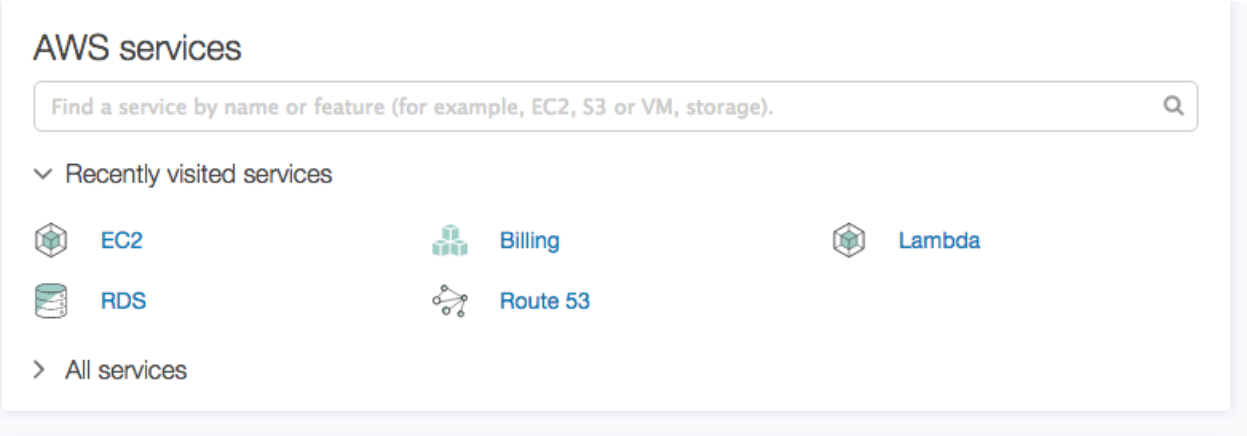


If a phone number appears in the MESSAGE field then our read operation works correct. You can proceed.
If not, review the previous steps to see what you have missed and ensure you have added entries to your database.

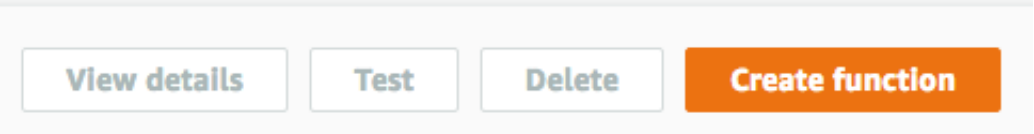
Create and test Alexa application:

Open a browser and log onto your Amazon AWS account:

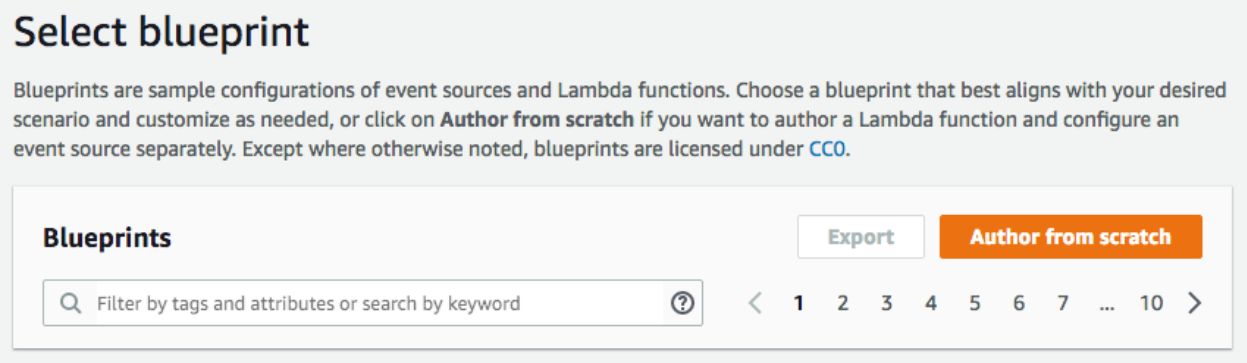
Click ‘Lamda':



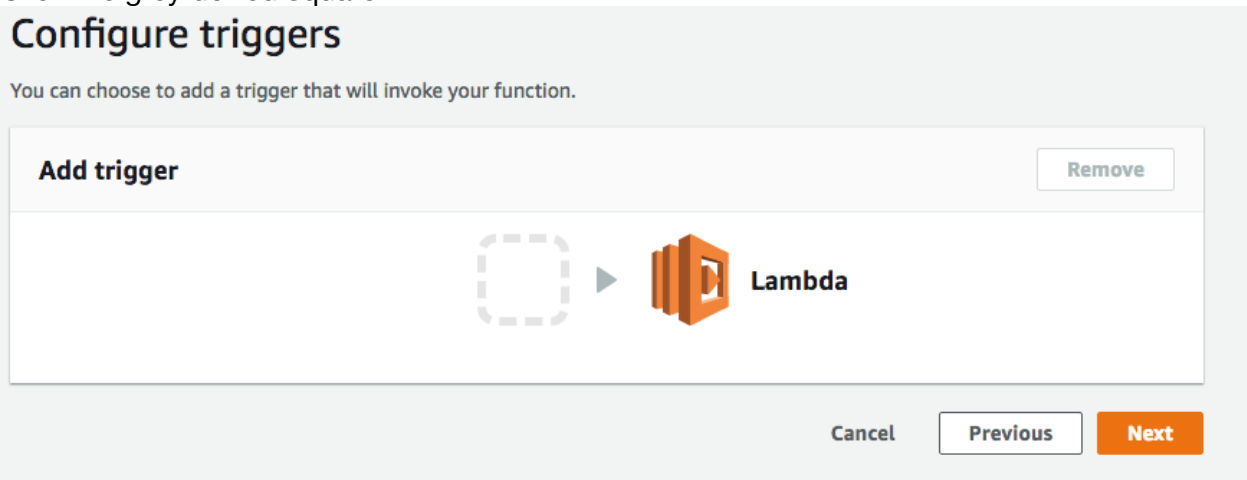
Click the ‘Create function’ button:



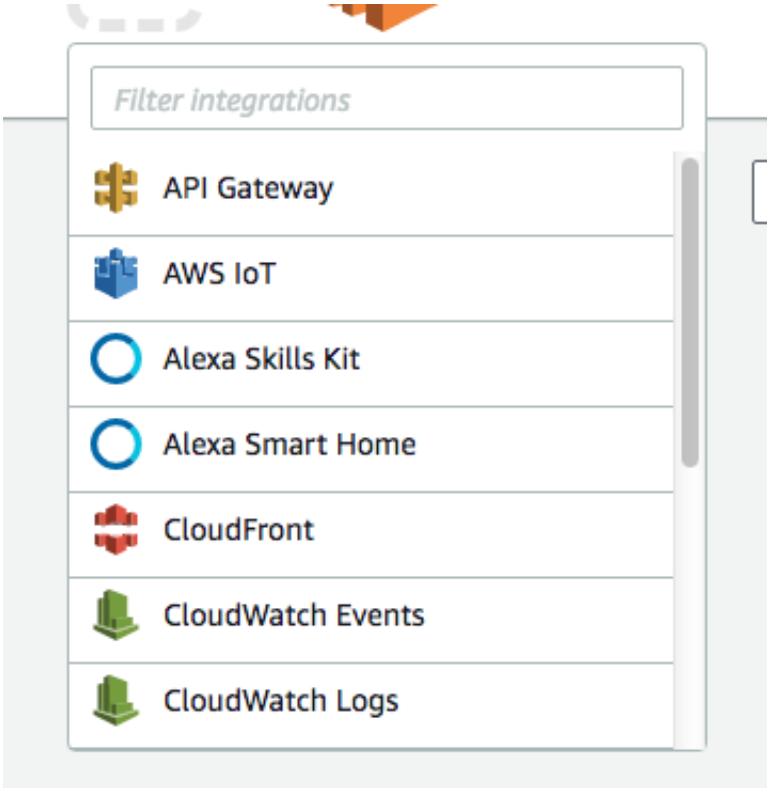
Click ‘Author from scratch’ :



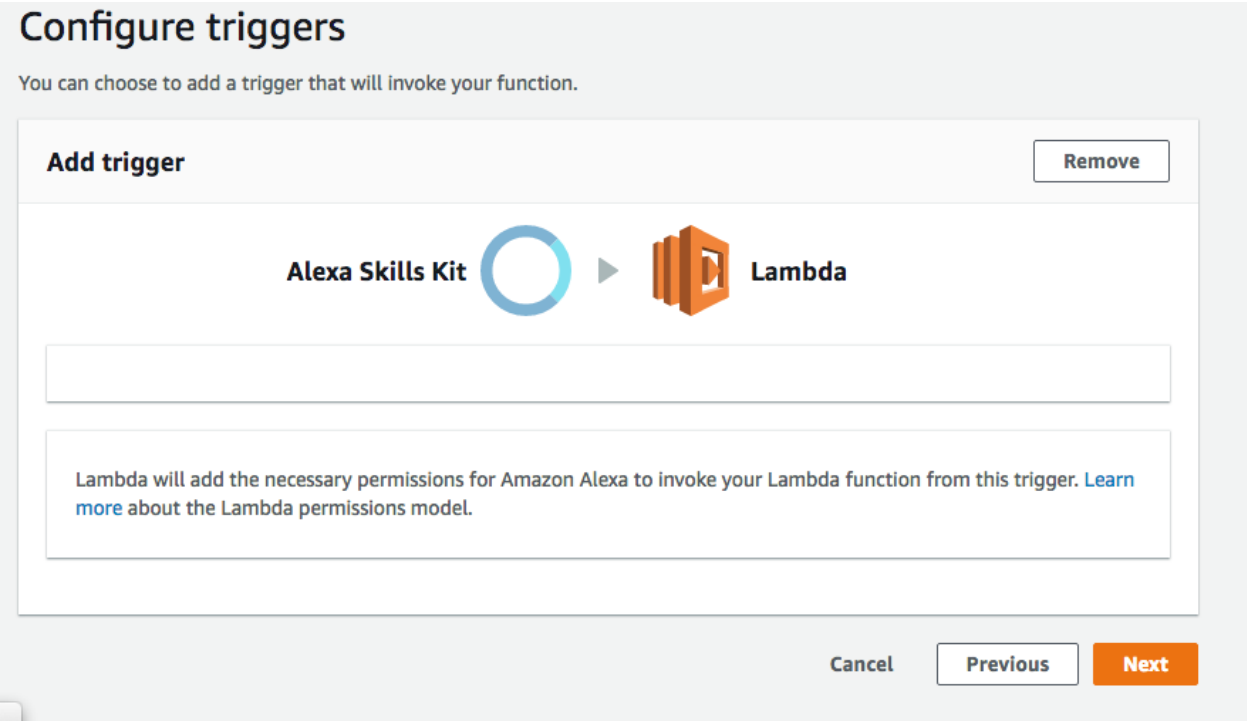
Click the grey dotted square:



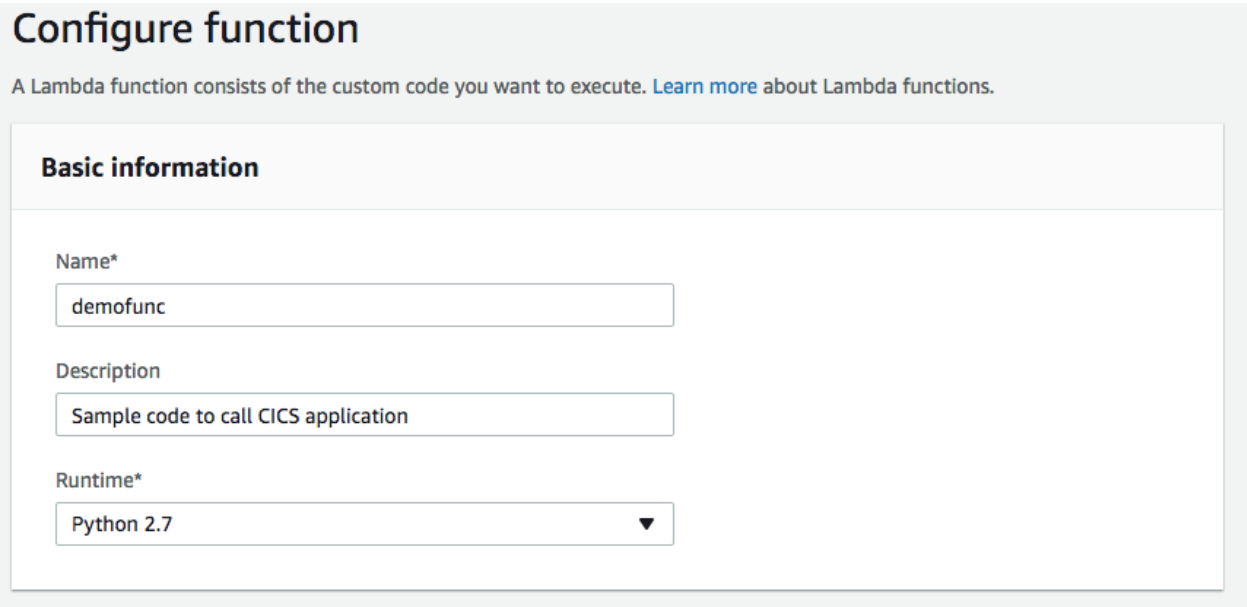
Click ‘Alexa Skills kit’:



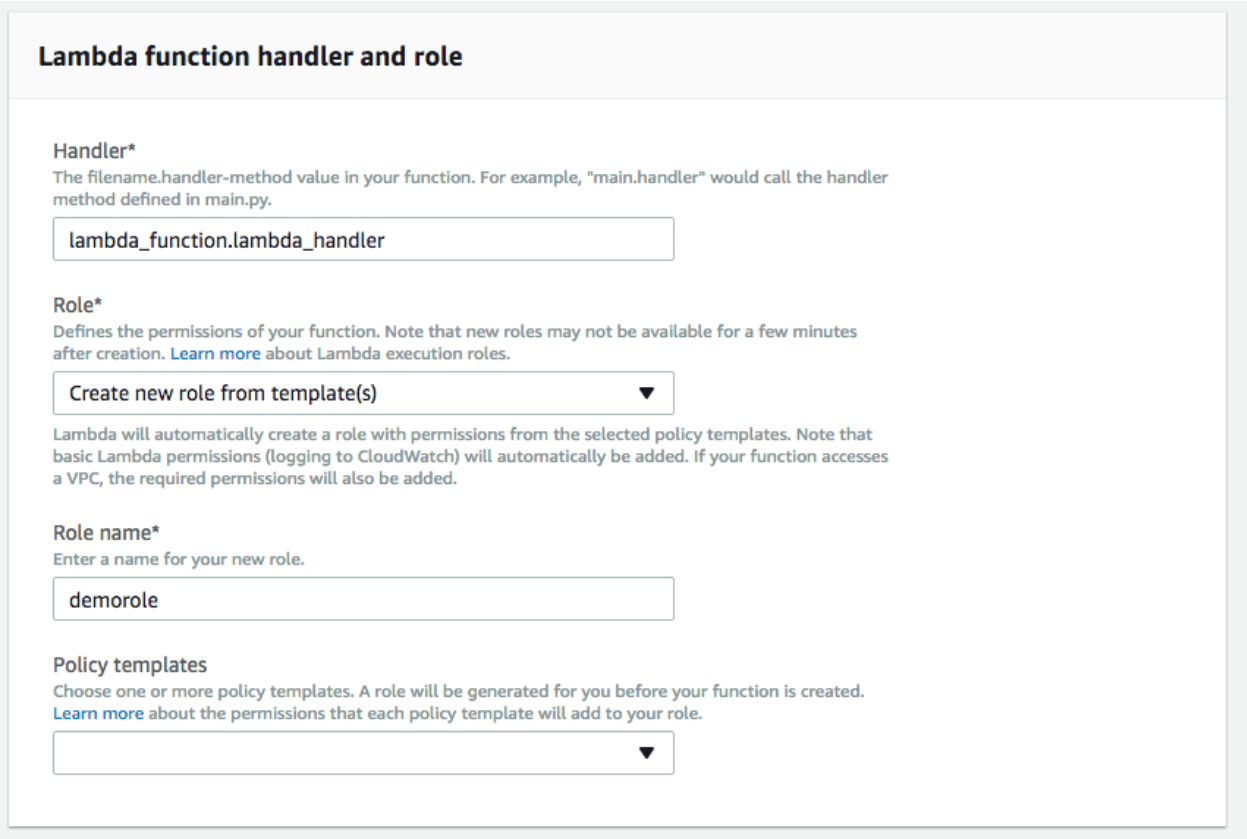
Click ‘Next’:



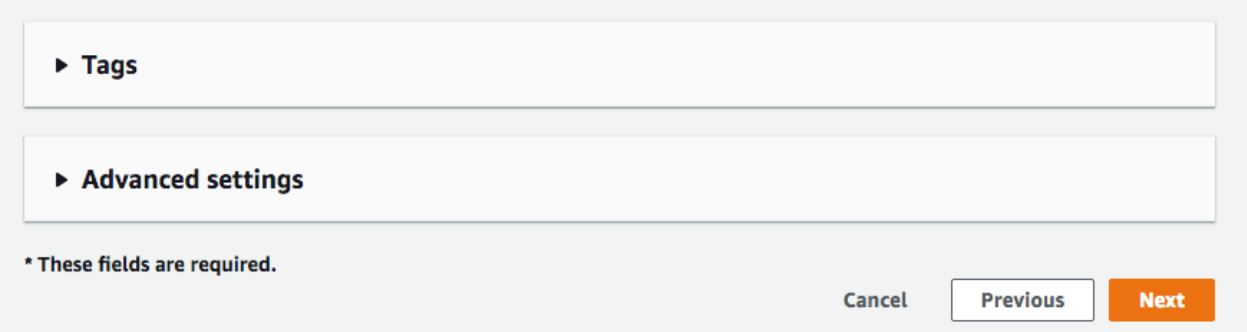
Enter Basic information as follows:



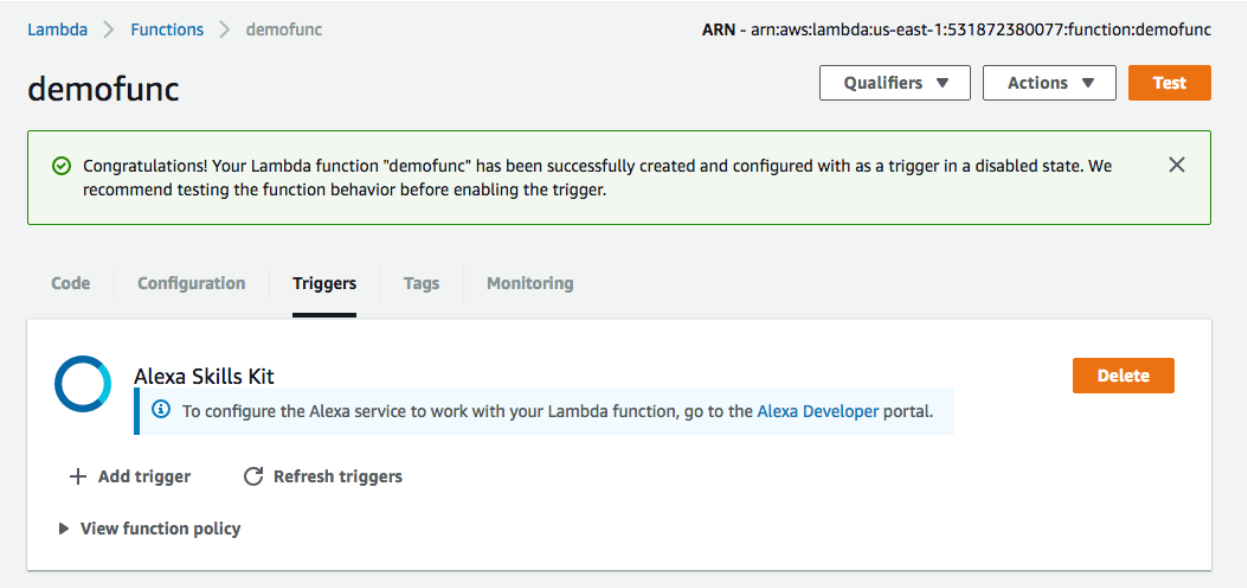
Fill in Lambda function handler and role as follows:



Click Next:

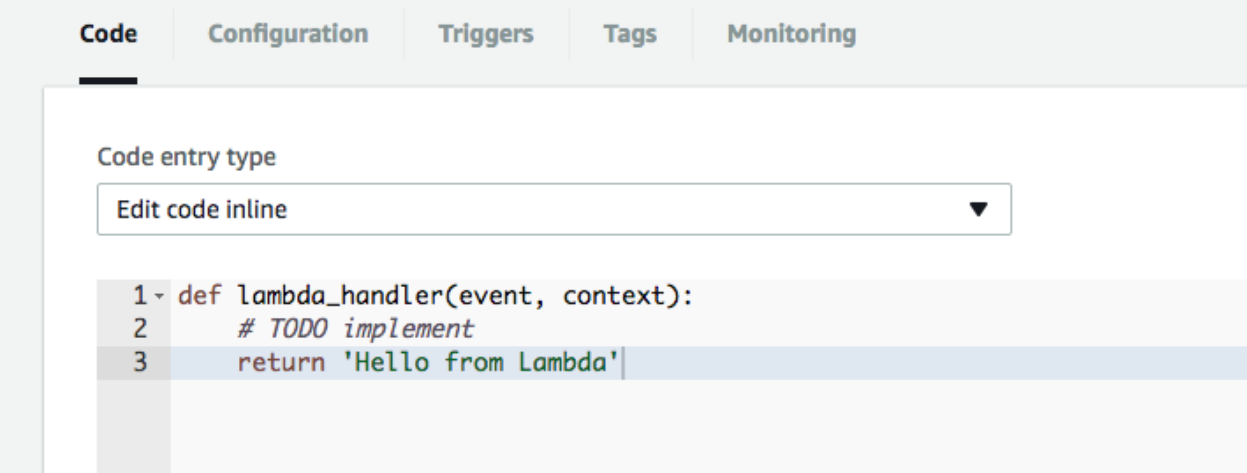


Click ‘Create function’:

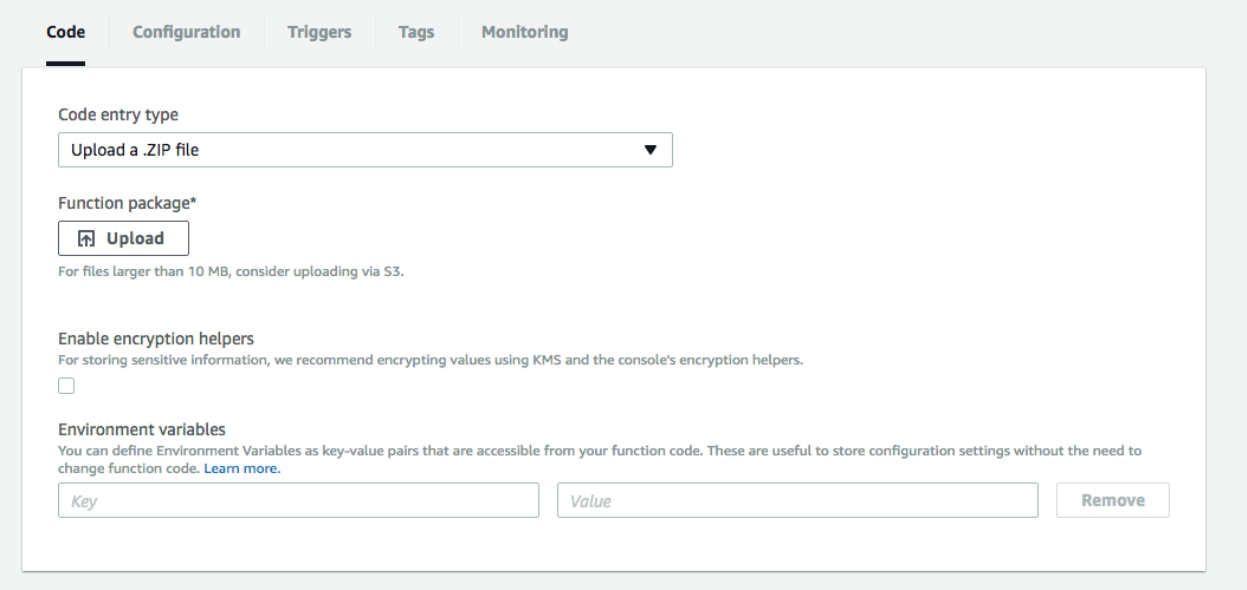


Make a copy of the ARN (top right). We'll need that to configure the Alexa skill shortly.

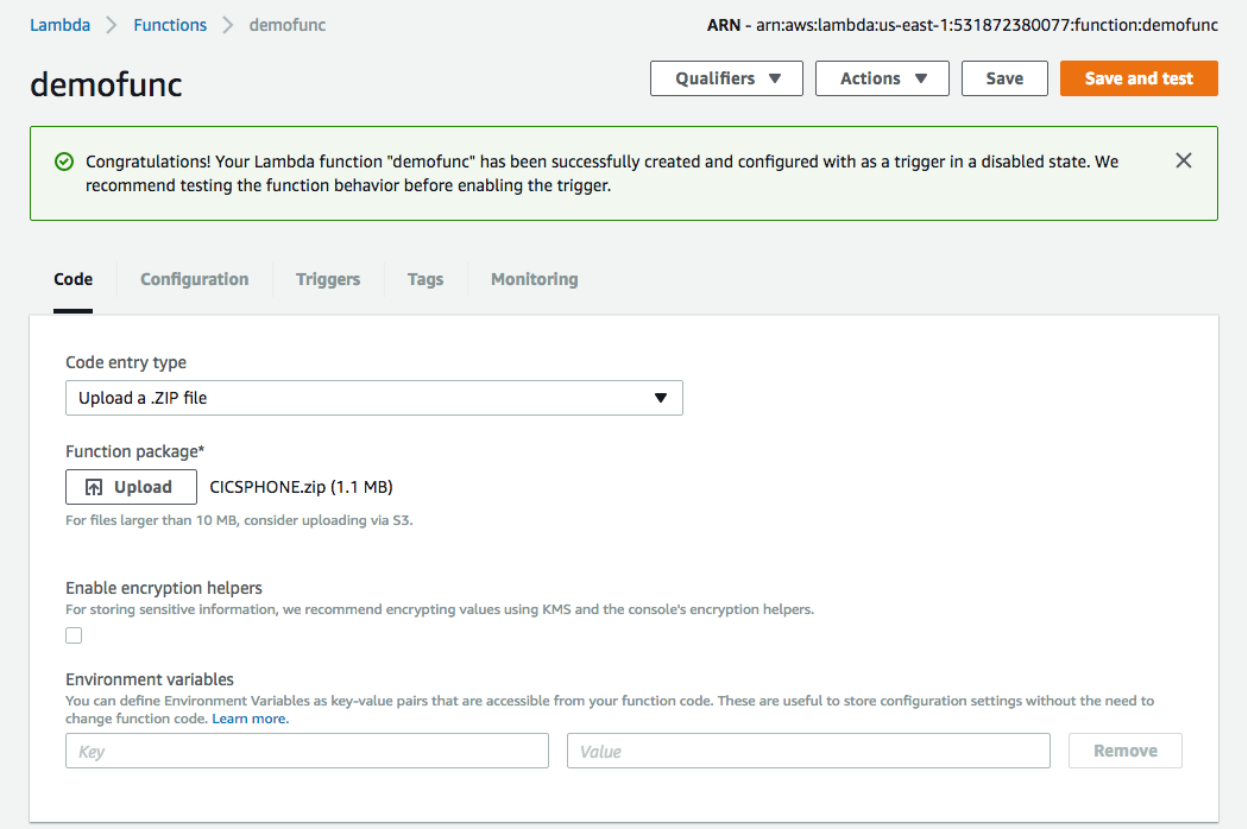
Click the 'Code' tab:



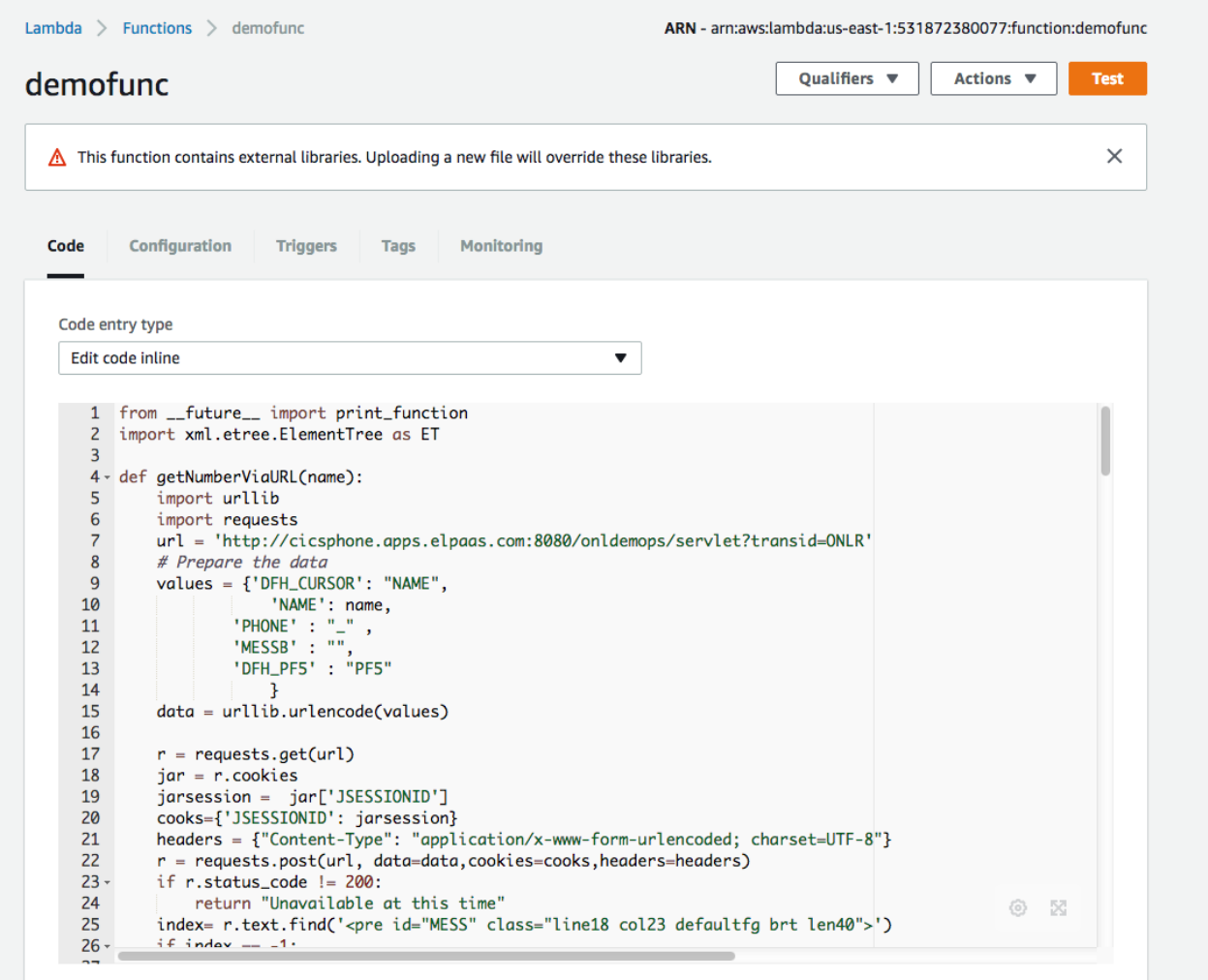
Because the demo makes use of a python library not in lambda by default we'll have to upload a source code bundle.
Change 'Code entry type' to 'Upload a .ZIP file':



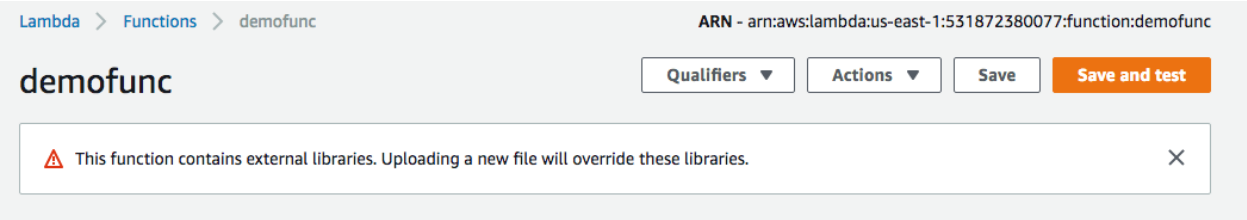
Click ‘Upload’ and select the CICSPHONE.zip from the source folder:



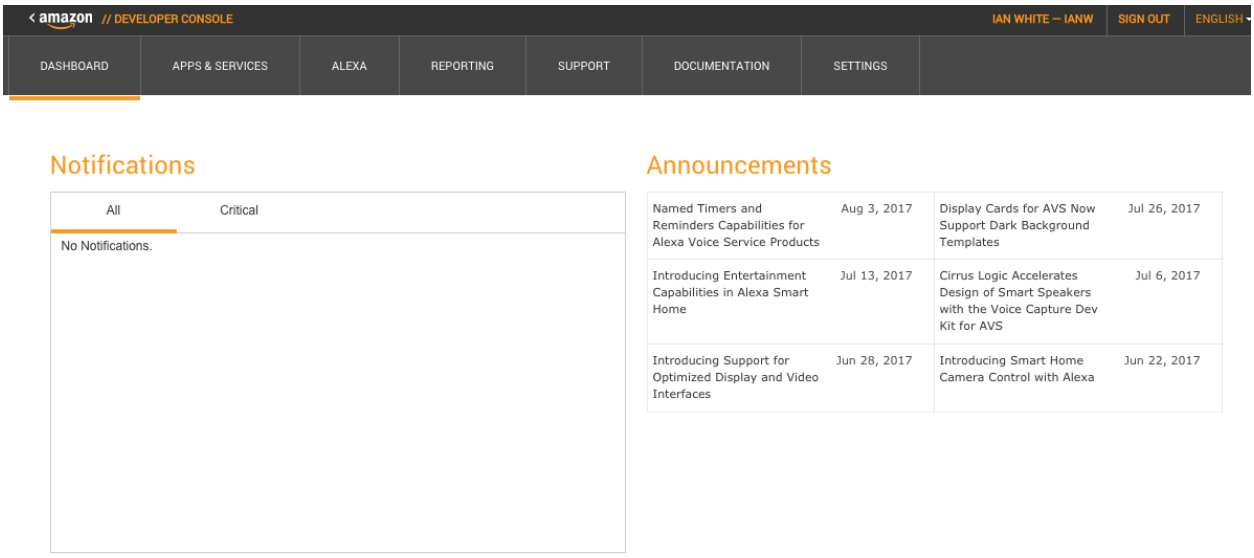
Click ‘Save’:



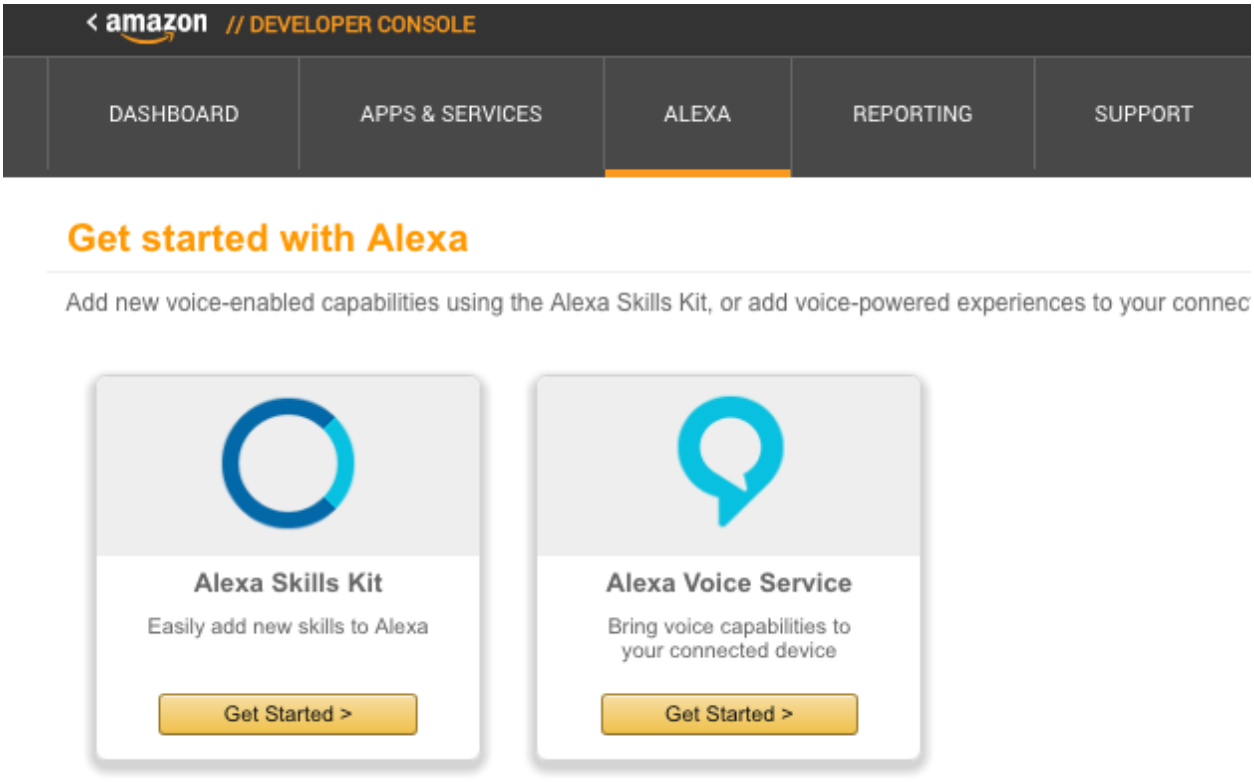
You should now edit line SEVEN (7) to be your applications URL, including the transid=ONLR at the end.
Once done, click ‘Save’:



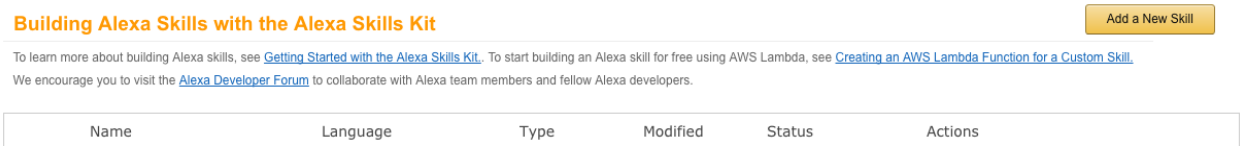
Open a new browser tab and navigate to https://developer.amazon.com/home.html
Sign in if required:



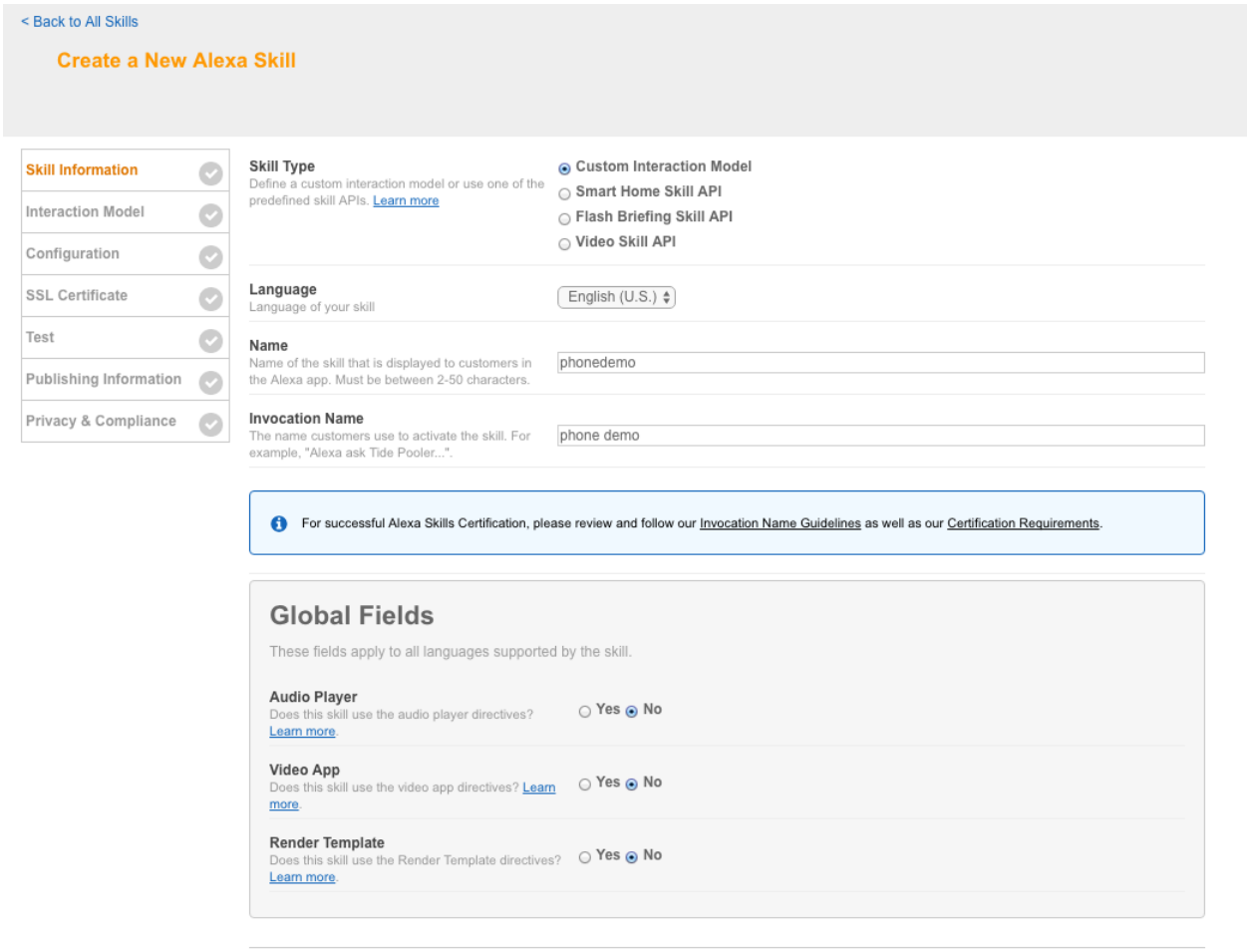
Click ‘Alexa’ in the top menu:



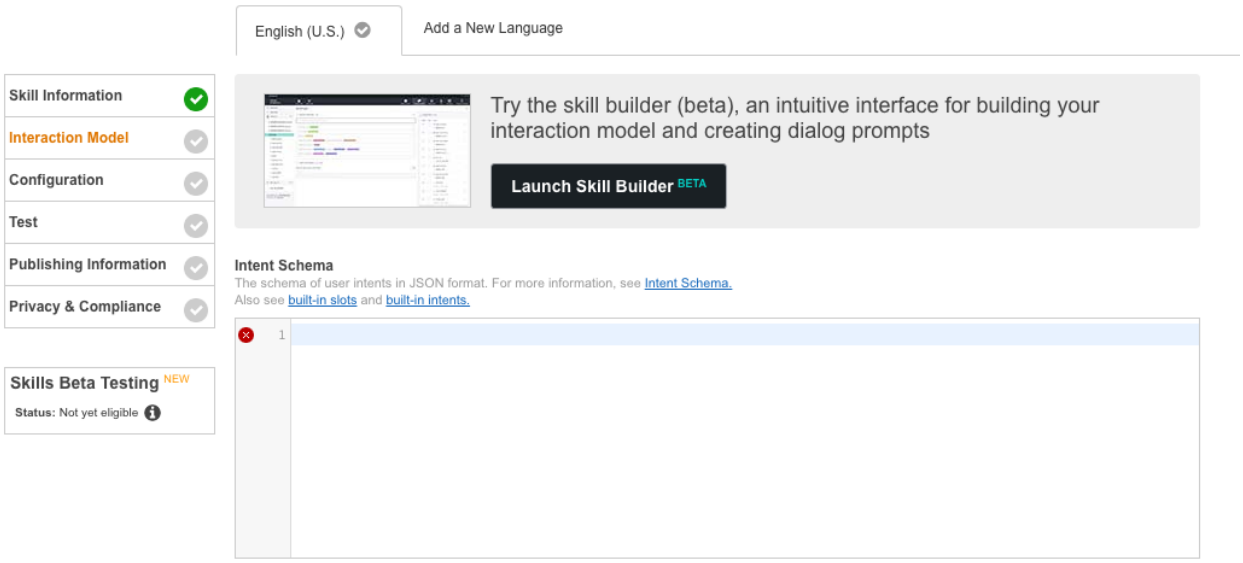
Click the ‘Get Started’ in the Alexa Skills Kit box:



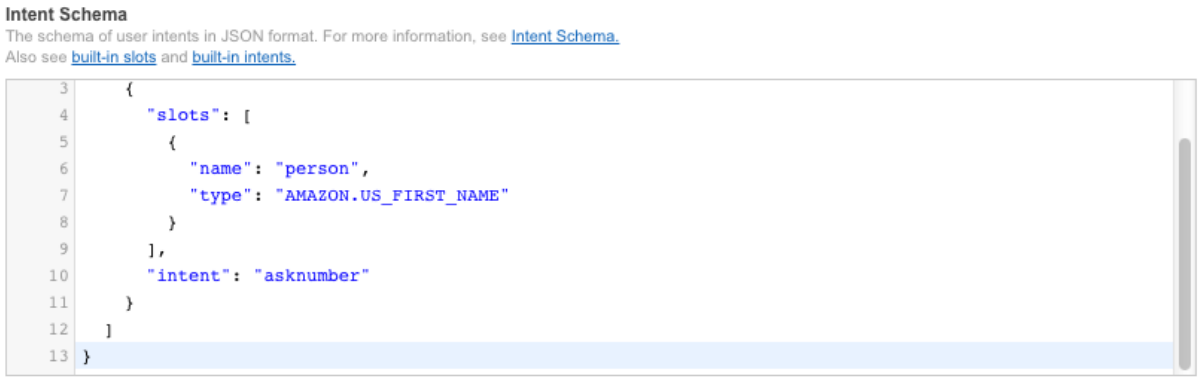
Click ‘Add a New Skill’:



Fill in the skill name and invocation name like the above screenshot and click ‘Save’, then click ‘Interaction Model’ on the left menu:



Copy the text from intent_schema.txt from your source folder into the '**Intent Schema**' field.
Do the same for '**Sample Utterances**' from sample_utterances.txt.
Your fields should look like this:



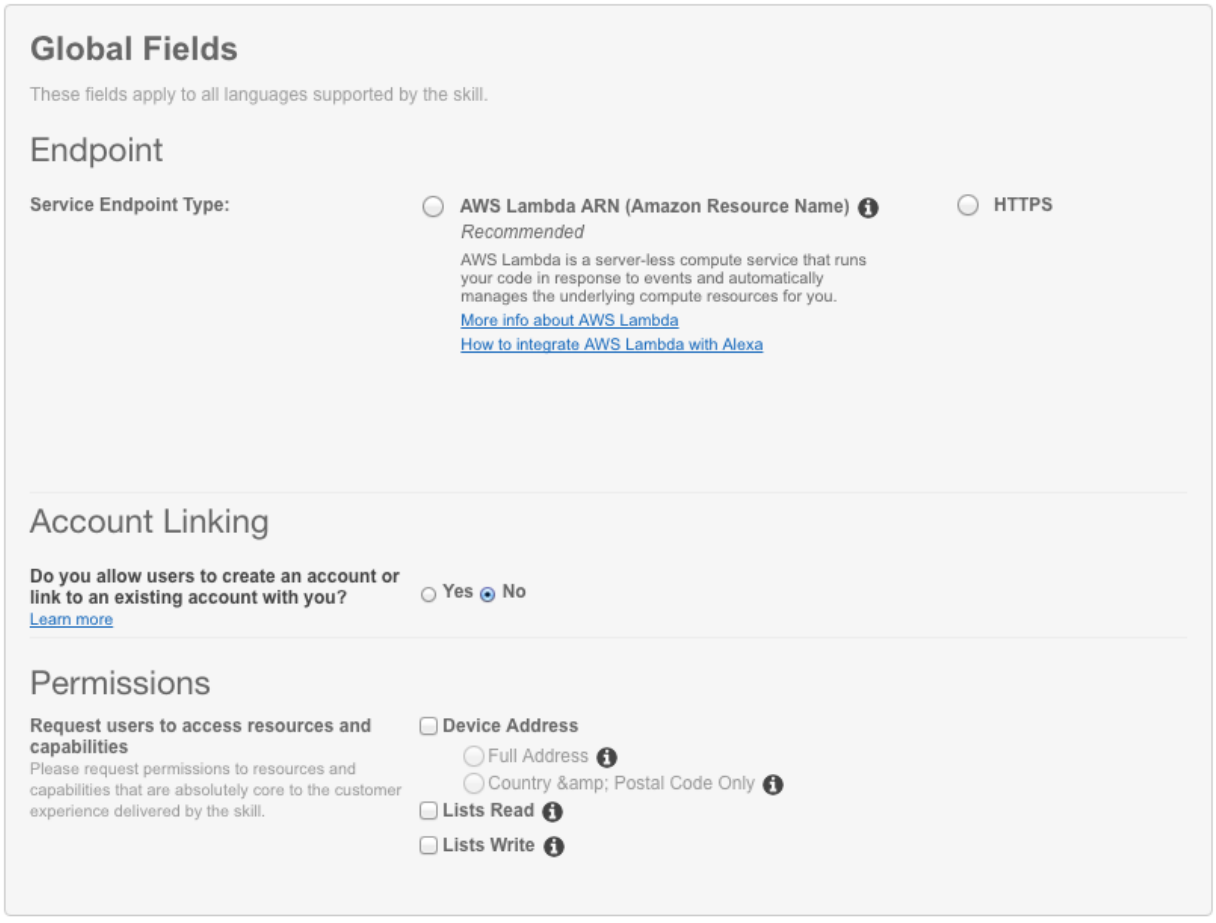
Sample Utterances

These are what people say to interact with your skill. Type or paste in all the ways that people can invoke the intents. Learn more

Up to 3 of these will be used as Example Phrases, which are hints to users.

```
1 asknumber What is the phone number for {person}
2 asknumber what is the number for {person}
3 asknumber for the number for {person}
```

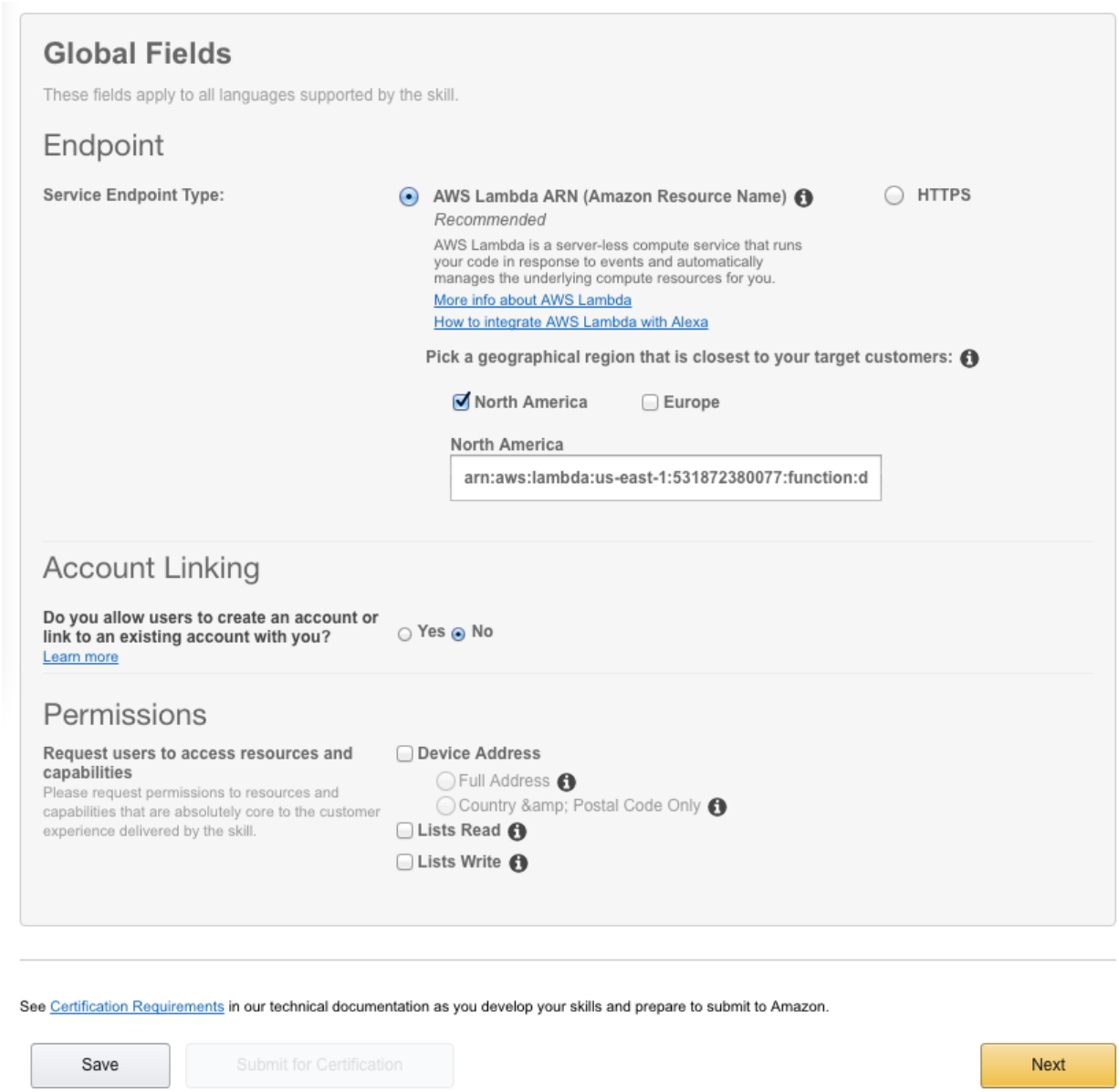
Click ‘Next’:



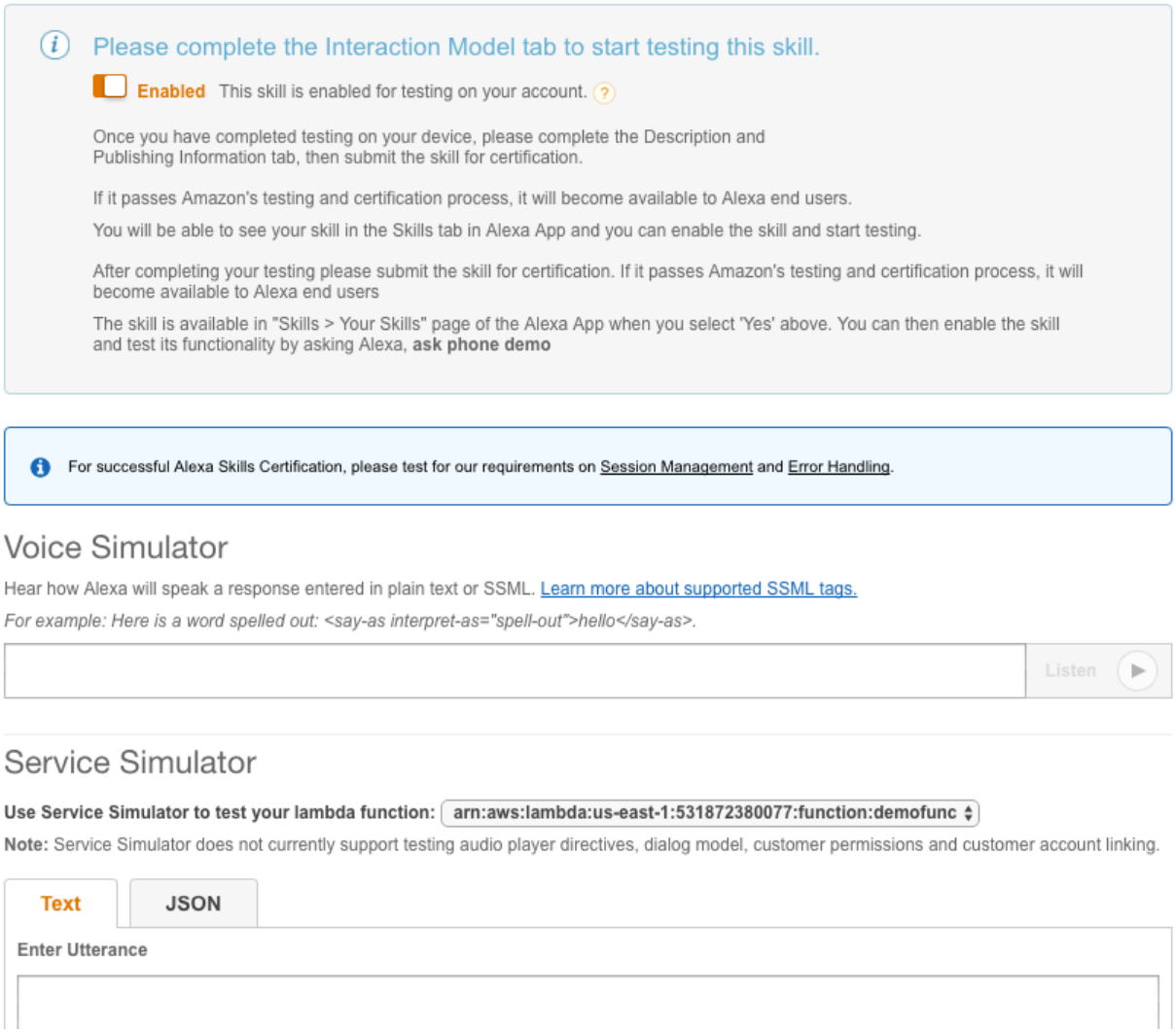
See Certification Requirements in our technical documentation as you develop your skills and prepare to submit to Amazon.

Save | Submit for Certification | Next

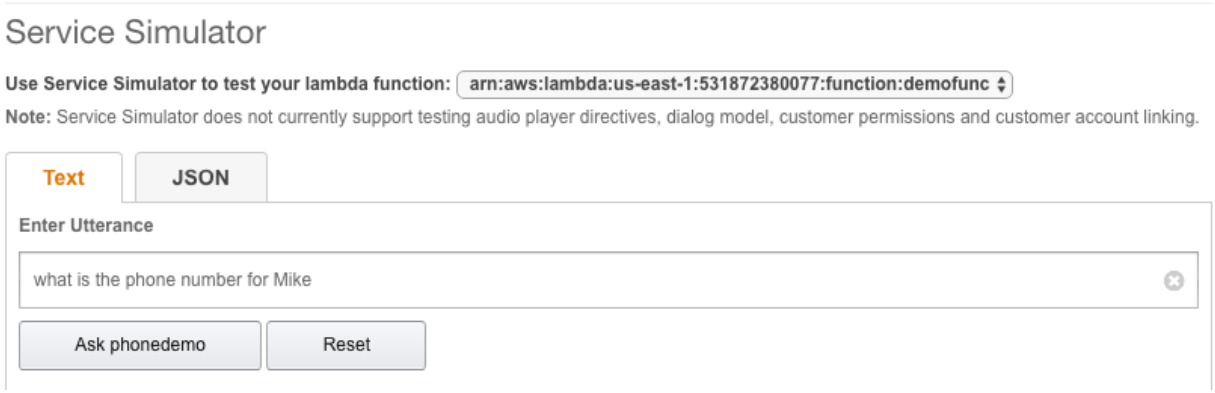
Click the AWS Lambda.. radio button , click the North America checkbox, and enter your ARN into the edit field.
Your screen should look like this:



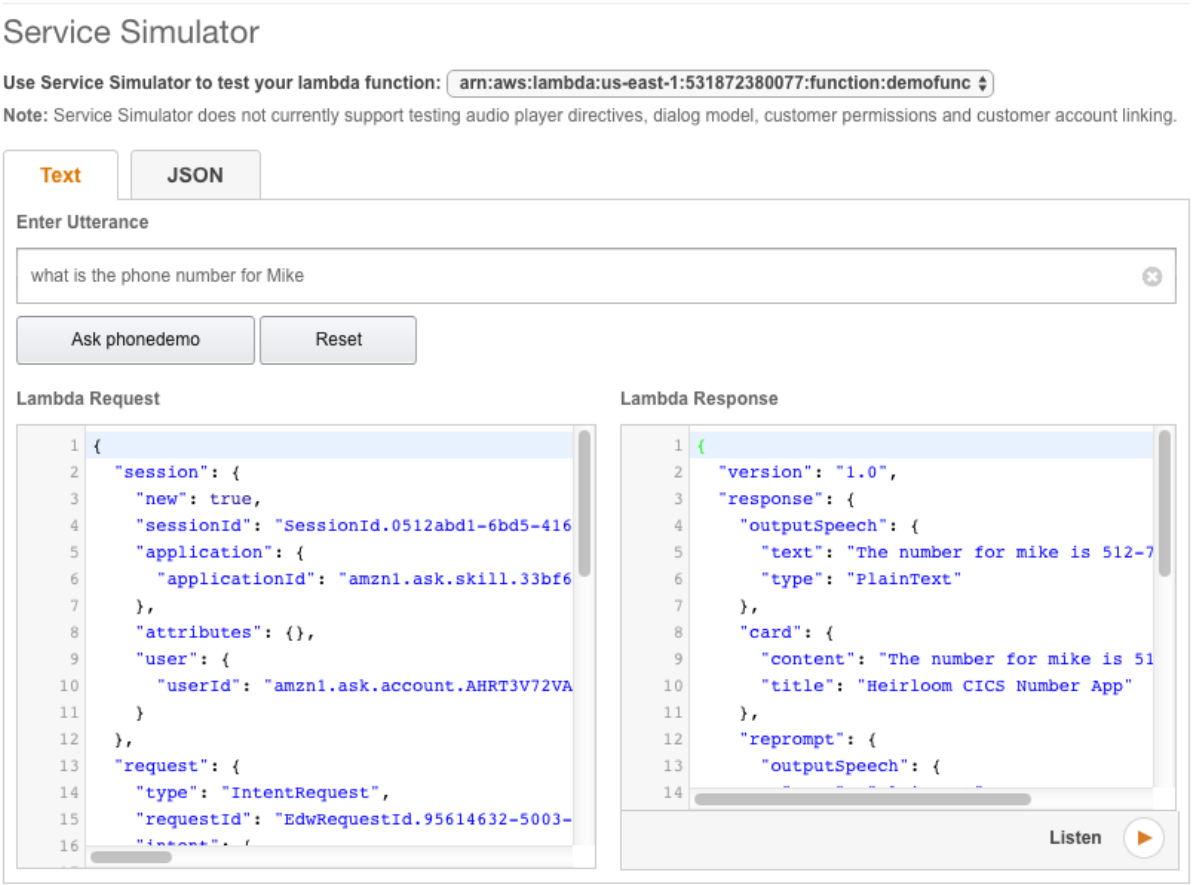
Click ‘Next’:



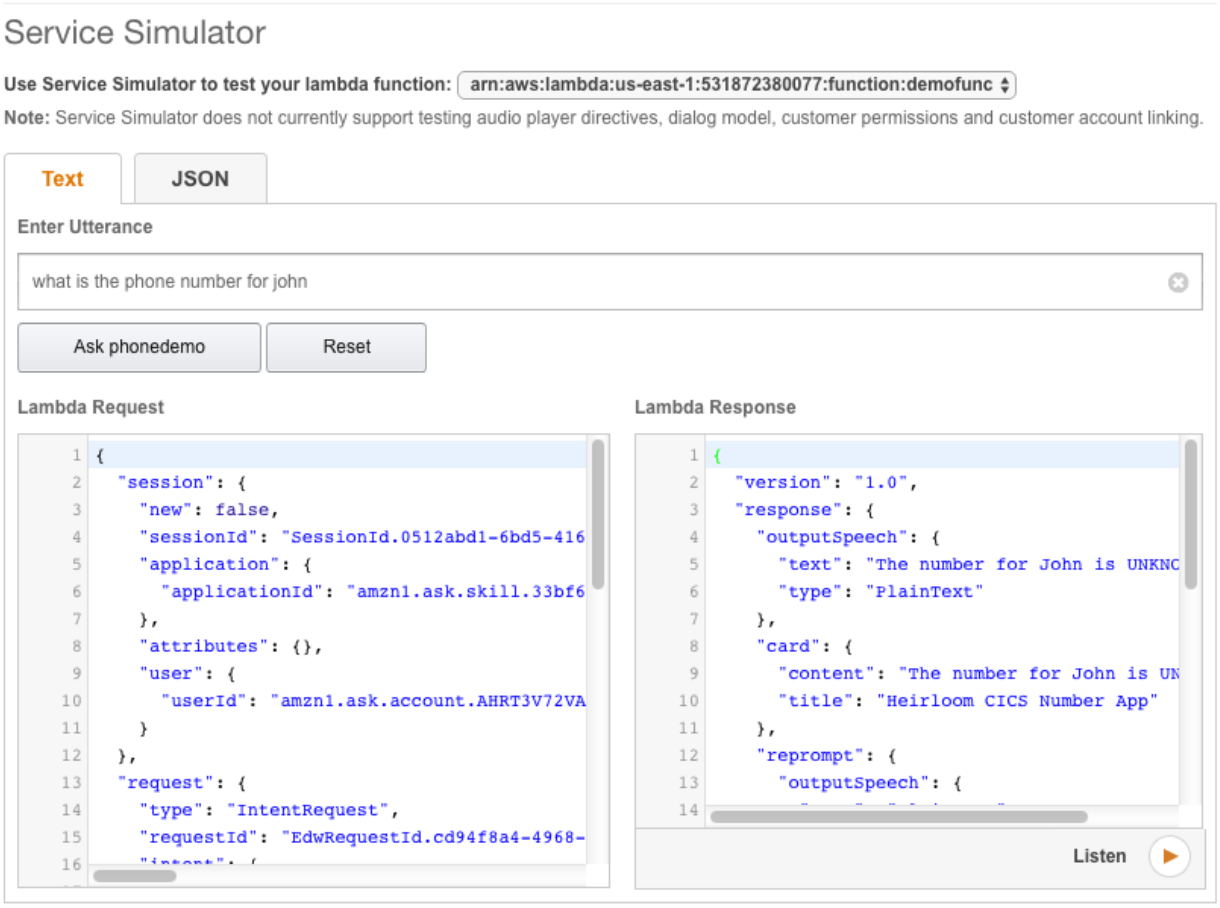
You can now test the service from here.
Enter ‘what is the phone number for’ and then someone you added to the database. For example:



Click ‘Ask phonedemo’:



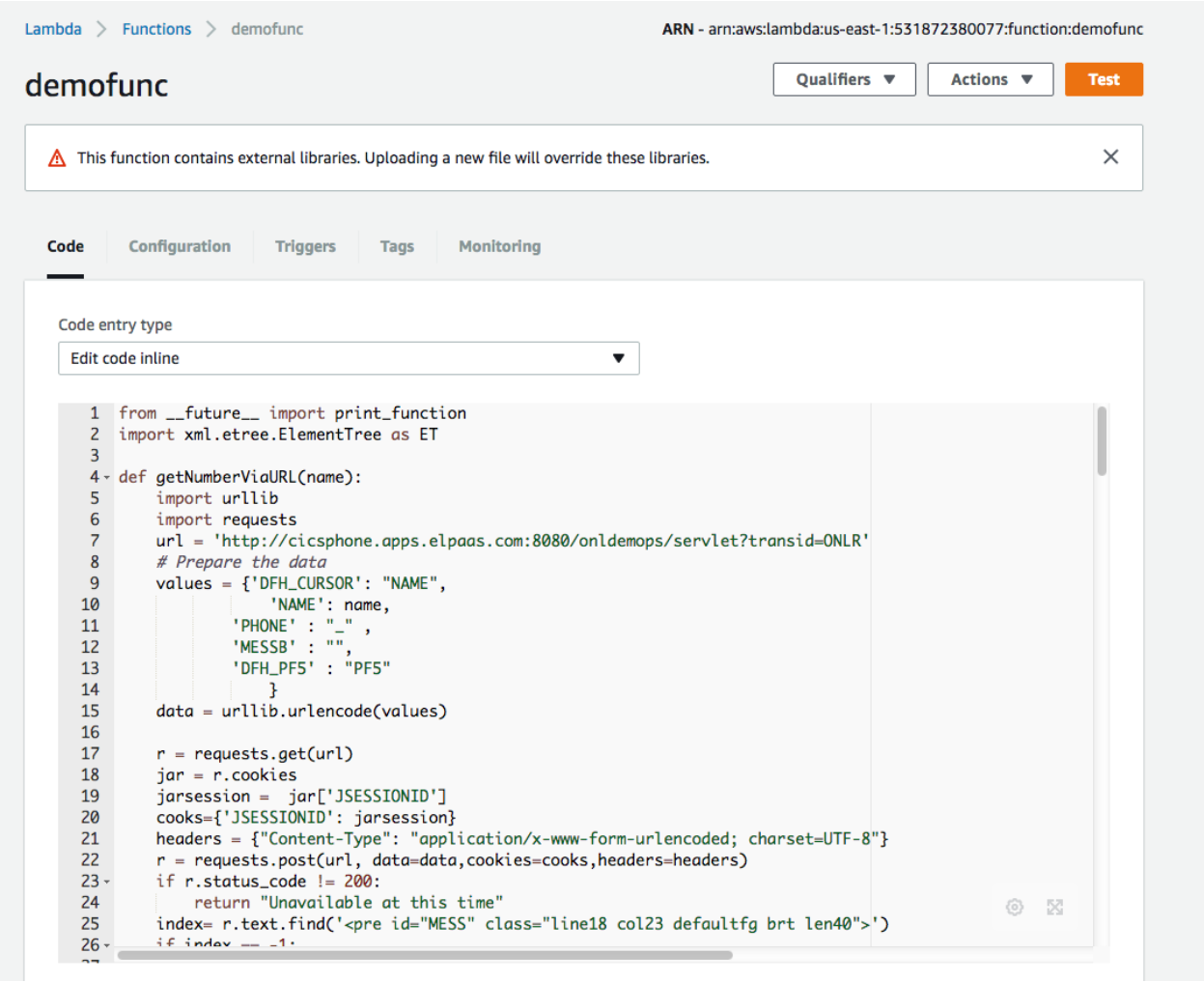
You should see JSON in the response window, and assuming you asked for a person in your DB you’ll see a phone number.
You can change the text to ask for a person you’ve not added and confirm it returns UNKNOWN:



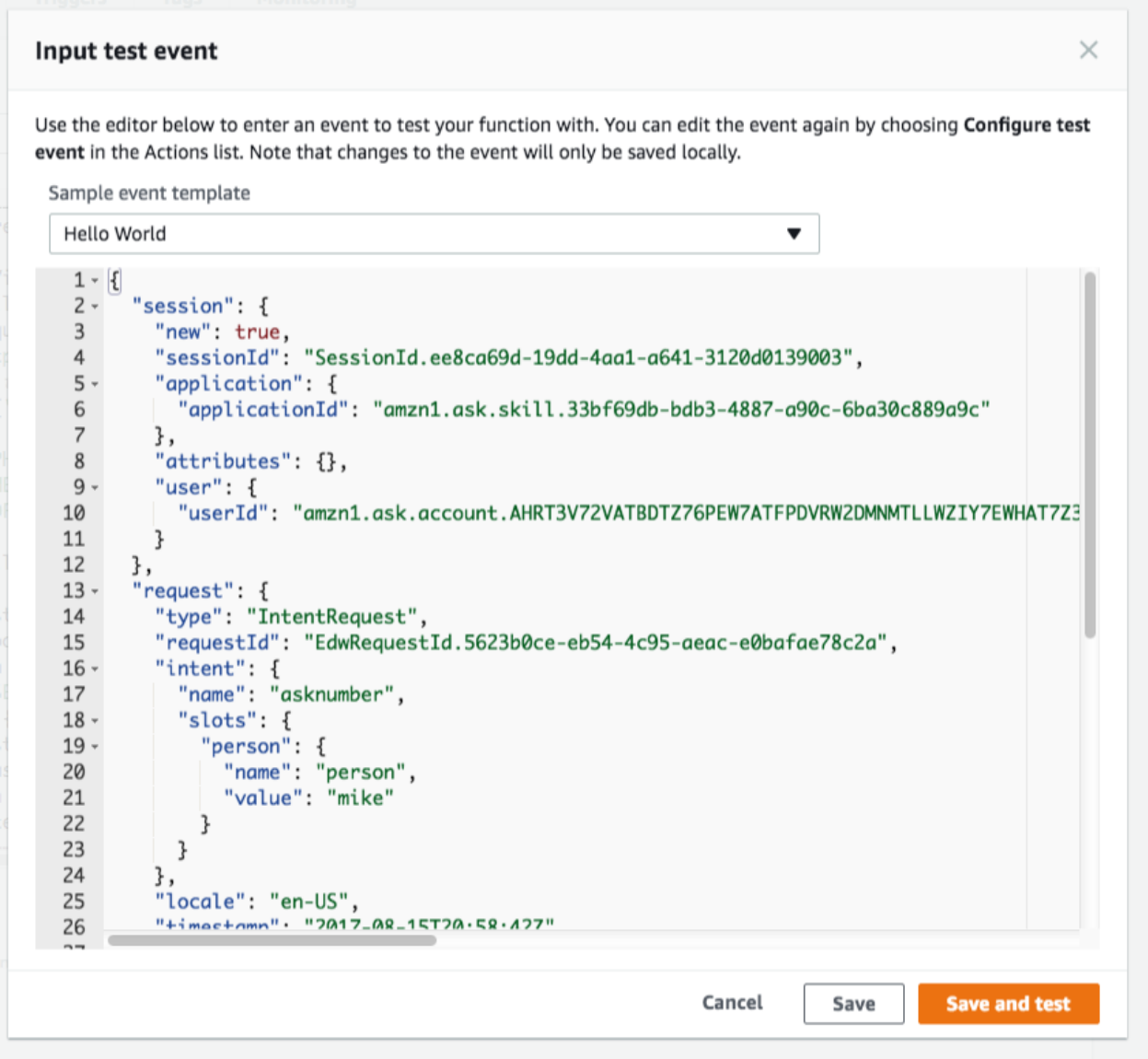
You are now ready to try this via your Echo or Dot.

You can say 'Alexa (or your wake-word) Ask <your app name> what is the number for <person>"
Or you can say ‘Alexa , open <your app name>’ and you’ll hear the welcome message before a prompt for you to ask for a number.

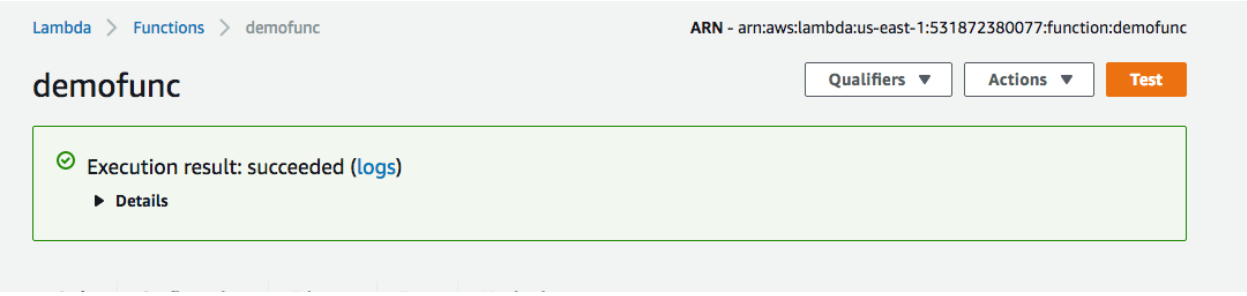
You can explore and change the python code to produce new text or features.
Once you save you can return to the Alexa skill to retest it, or you can take a copy of the ‘Lambda Request’ JSON from the Alexa skill and use it to drive a test in the Lamda function editor.
To do that,copy the text in the Lamba Request field above, and then change browser tabs back to your Lamda code editor window:



Click ‘Actions’ and then ‘Configure New Test Event:

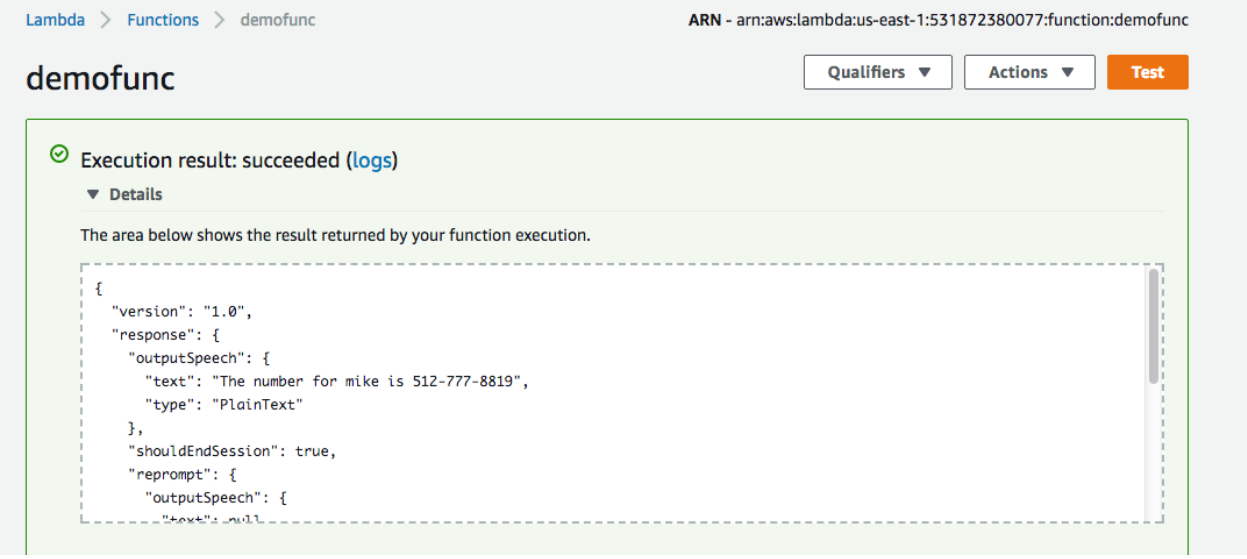


Past in the JSON code you copied as above and click ‘Save and test’:



You’ll see an Execution result entry and you can click the arrow next to Details to show the

resulting JSON (or any errors):



That test is now saved as the default so you can re-run it by clicking the ‘Test’ button after you change and save your code.