

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.

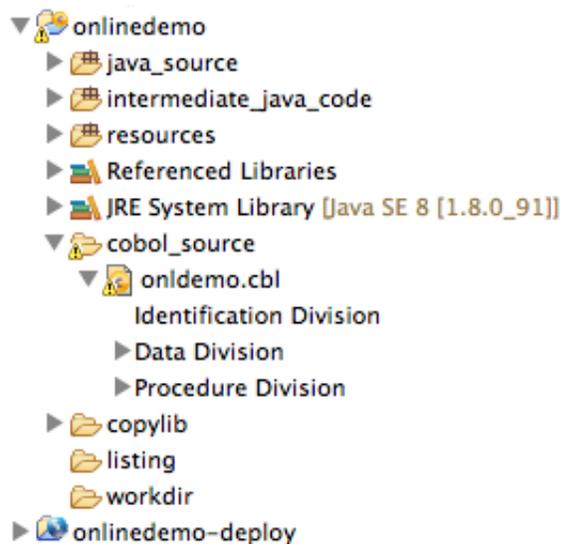
Additional sources for these steps can be found

here: <http://www.elasticcobol.com/downloads/demo/AlexaDemoSources.zip>

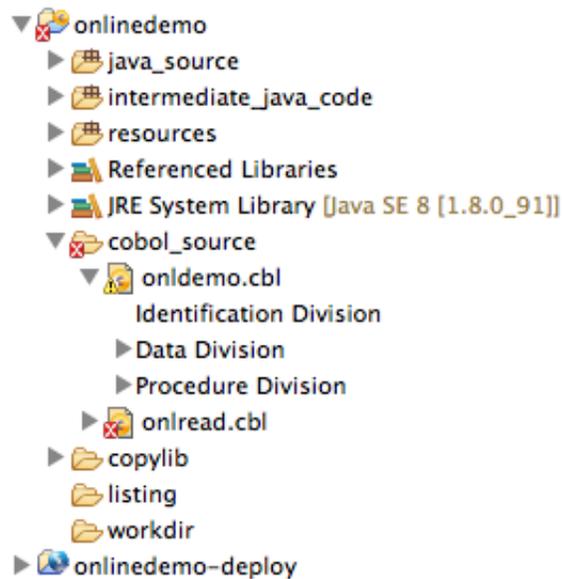
Modify and deploy sample:

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

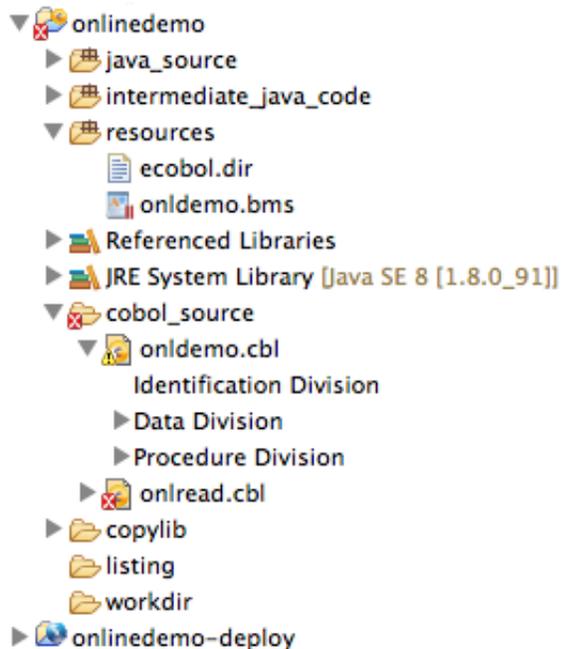
Open Eclipse and expand your onlinedemo project:



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:

The screenshot displays an IDE interface. On the left, a project tree shows the 'onlinedemo' project selected. A context menu is open over it, with 'Elastic COBOL BMS' highlighted. A sub-menu is also open, showing 'Generate Copyfile' and 'Generate HTML' options. The main editor area shows COBOL code with line numbers 30-60. The bottom panel contains a 'Problems' window with a table of job information.

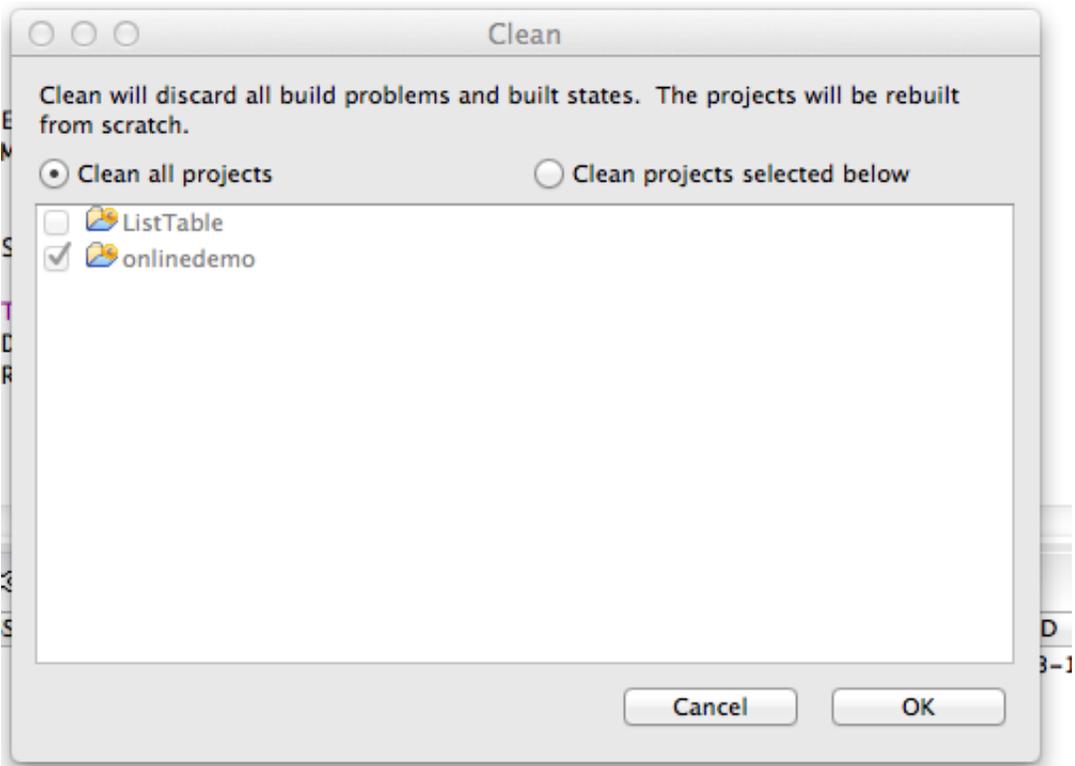
```

30 SEND-MAP.
31     EXEC CICS SEND
32         MAP('LOCKER')
33         MAPSET('ONLDEM
34         FREEKB
35         ERASE
36     END-EXEC.
37 SEND-MAP-EXIT.
38     EXIT.
39 RETURN-PARA.
40     EXEC CICS RETURN
41         TRANSID('ONLE
42         COMMAREA(WS-C
43     END-EXEC.
44 RETURN-EXIT.
45     EXIT.
46 RECEIVE-PARA.
47     MOVE LOW-VALUES 1
48     EXEC CICS RECEIVE
49         MAP('LOCKER')
50         MAPSET('ONLDE
51     END-EXEC.
52 RECEIVE-PARA-EXIT.
53     EXIT.
54 KEY-PARA.
55     EVALUATE EIBAID
56         WHEN DFHPF5
57             PERFORM FETCH
58             PERFORM SEND-M
59         WHEN OTHER

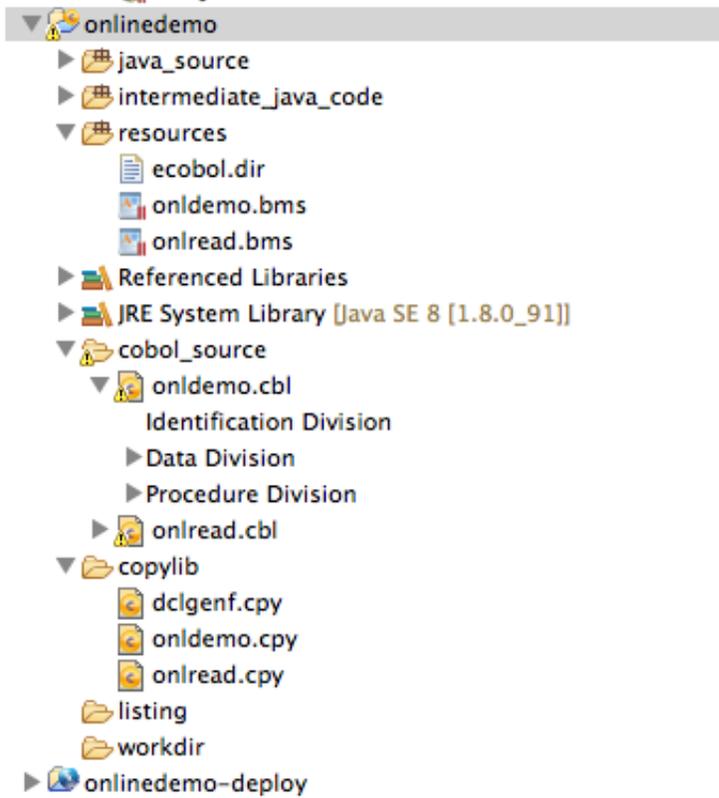
```

| JOB ID | JOB NAME | CLASS |
|-------------|----------|-------|
| 31756 | TEST01 | A |
| OUT31756... | | |

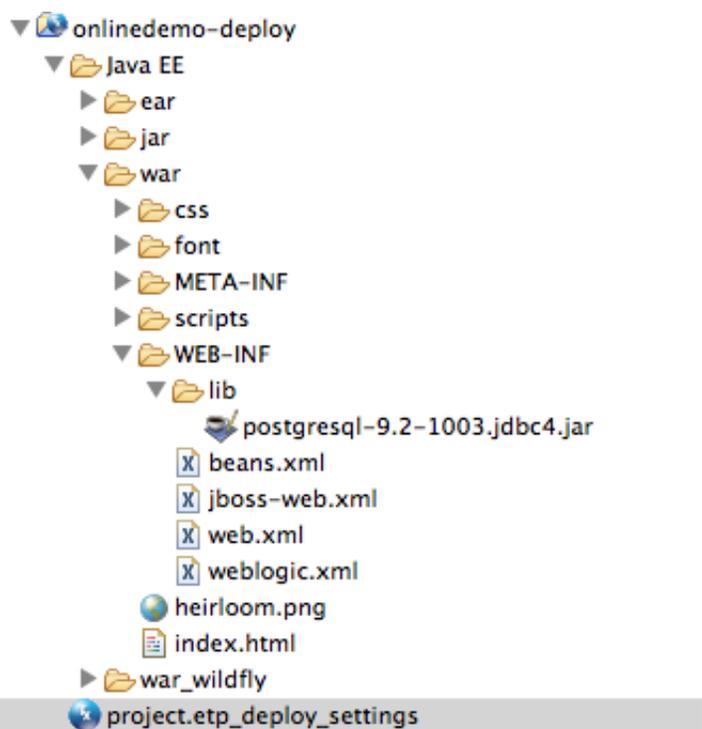
Click the **onlinedemo** project, and then from the Project menu click 'Clean..':



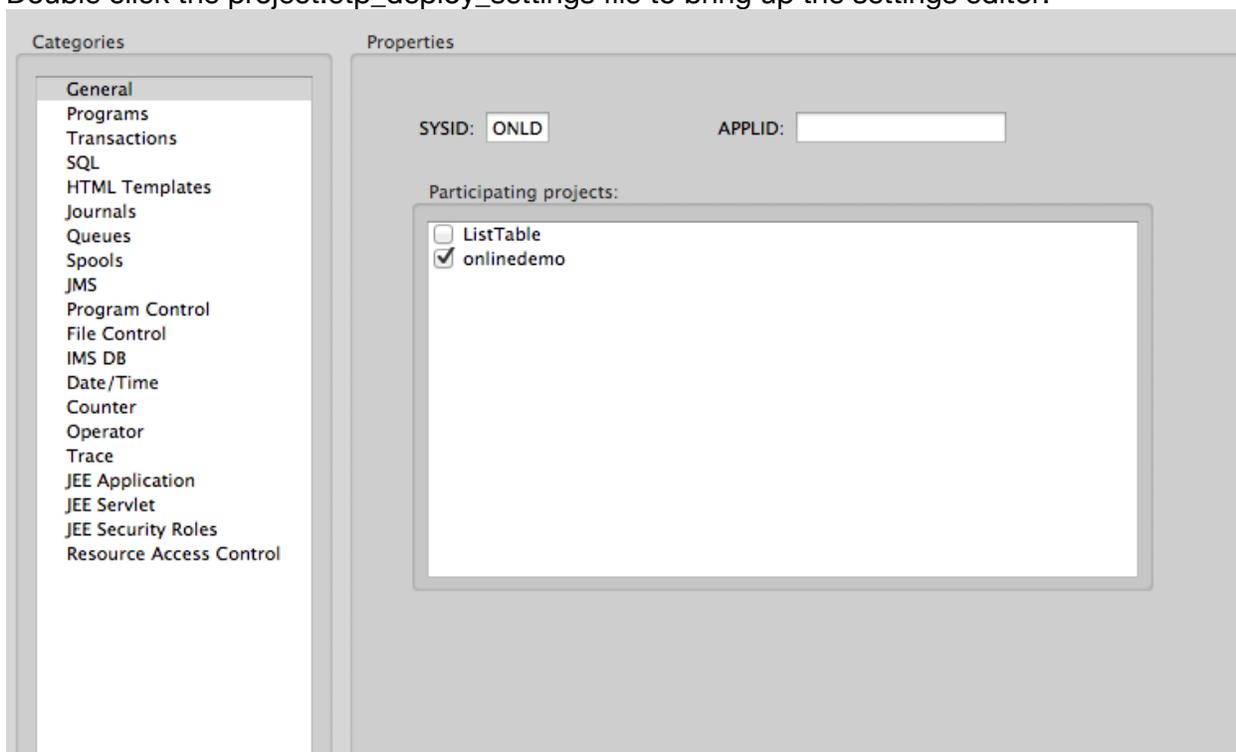
Ensure **onlinedemo** is selected and click OK to rebuild the project:



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



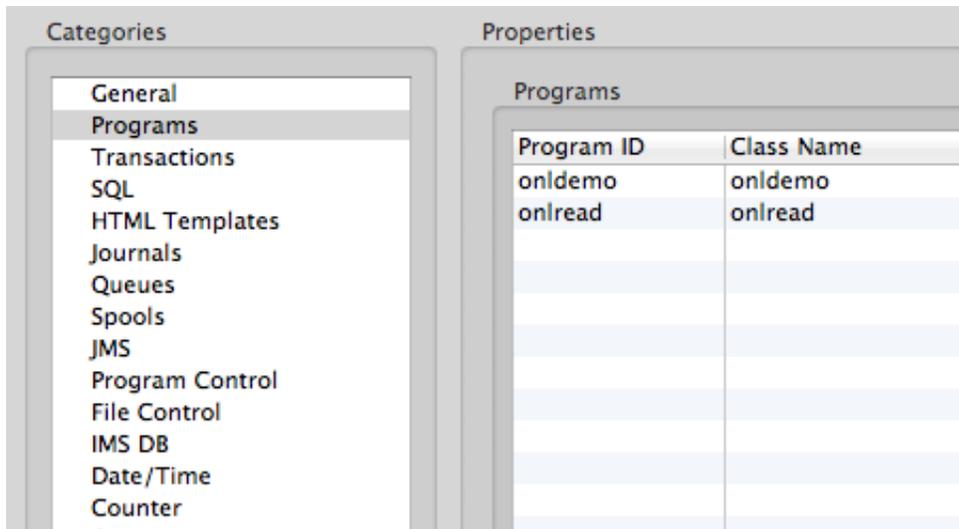
Double click the project.etc_deploy_settings file to bring up the settings editor:



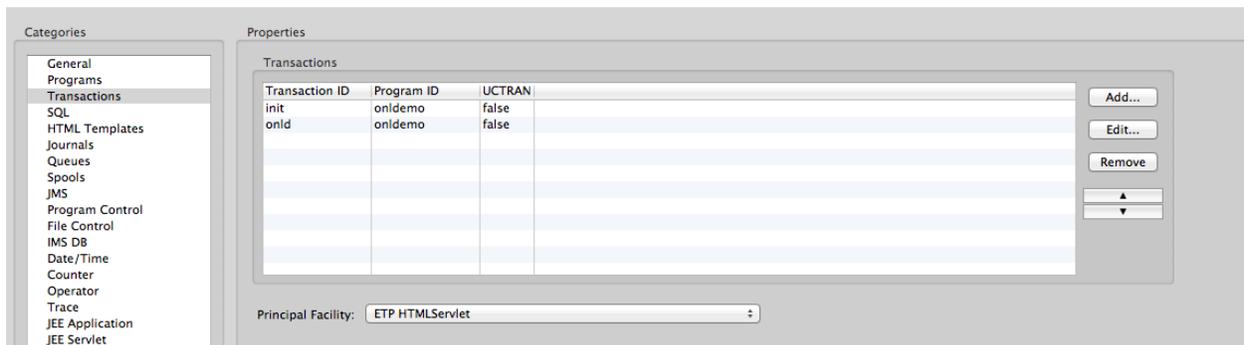
uncheck and recheck the 'online demo' project and click 'Apply'. This will ensure the project

sees all the new programs.

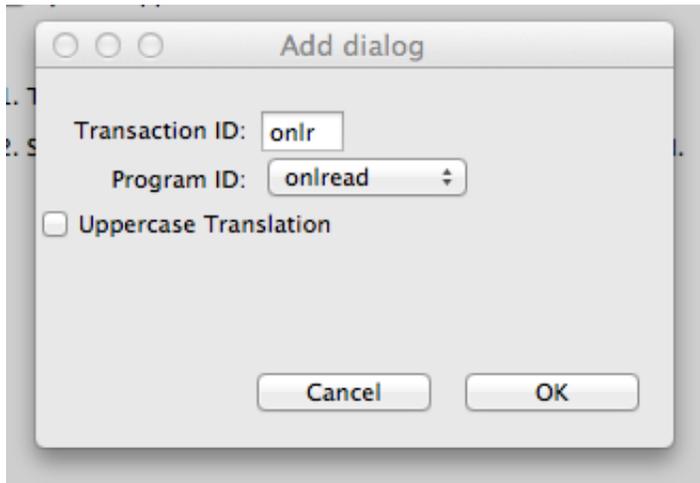
Confirm the new **onlineread** program is there by click 'Programs' on the left menu:



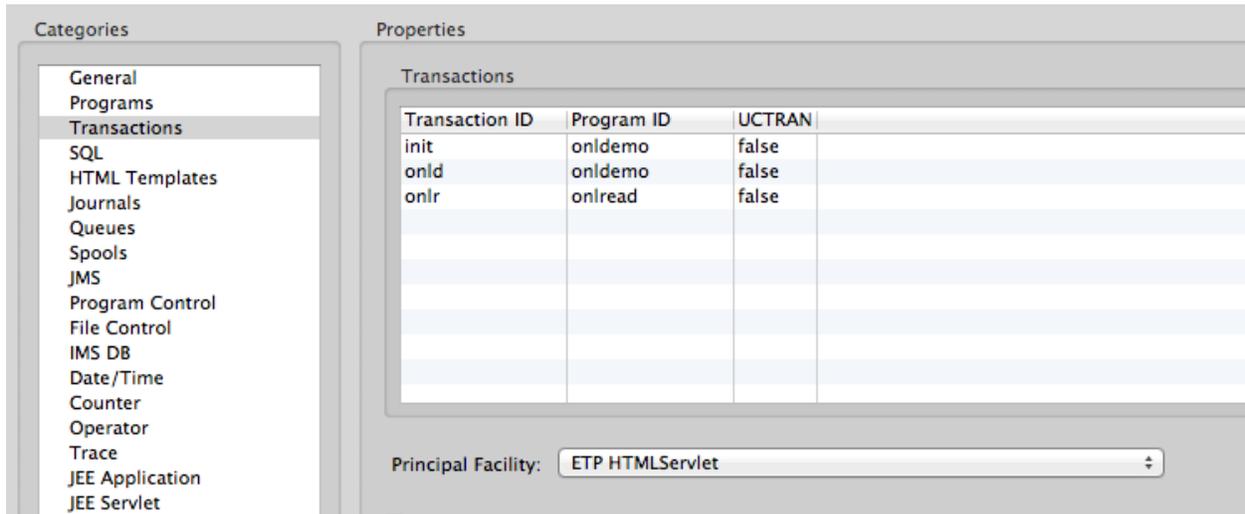
Click 'Transactions' in the left menu:



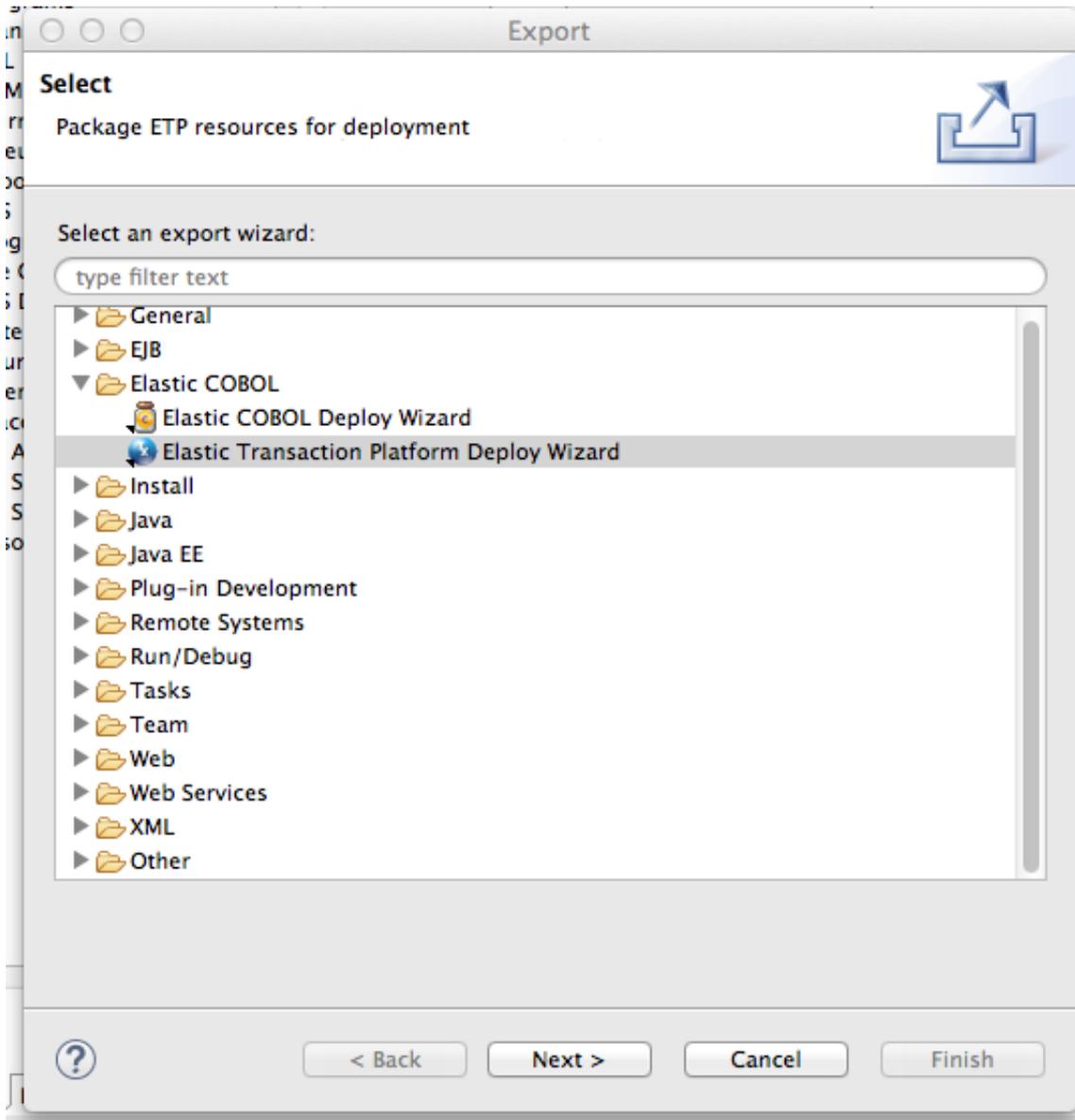
Click 'Add' and enter **onlr** and choose the **onlread** program ID:



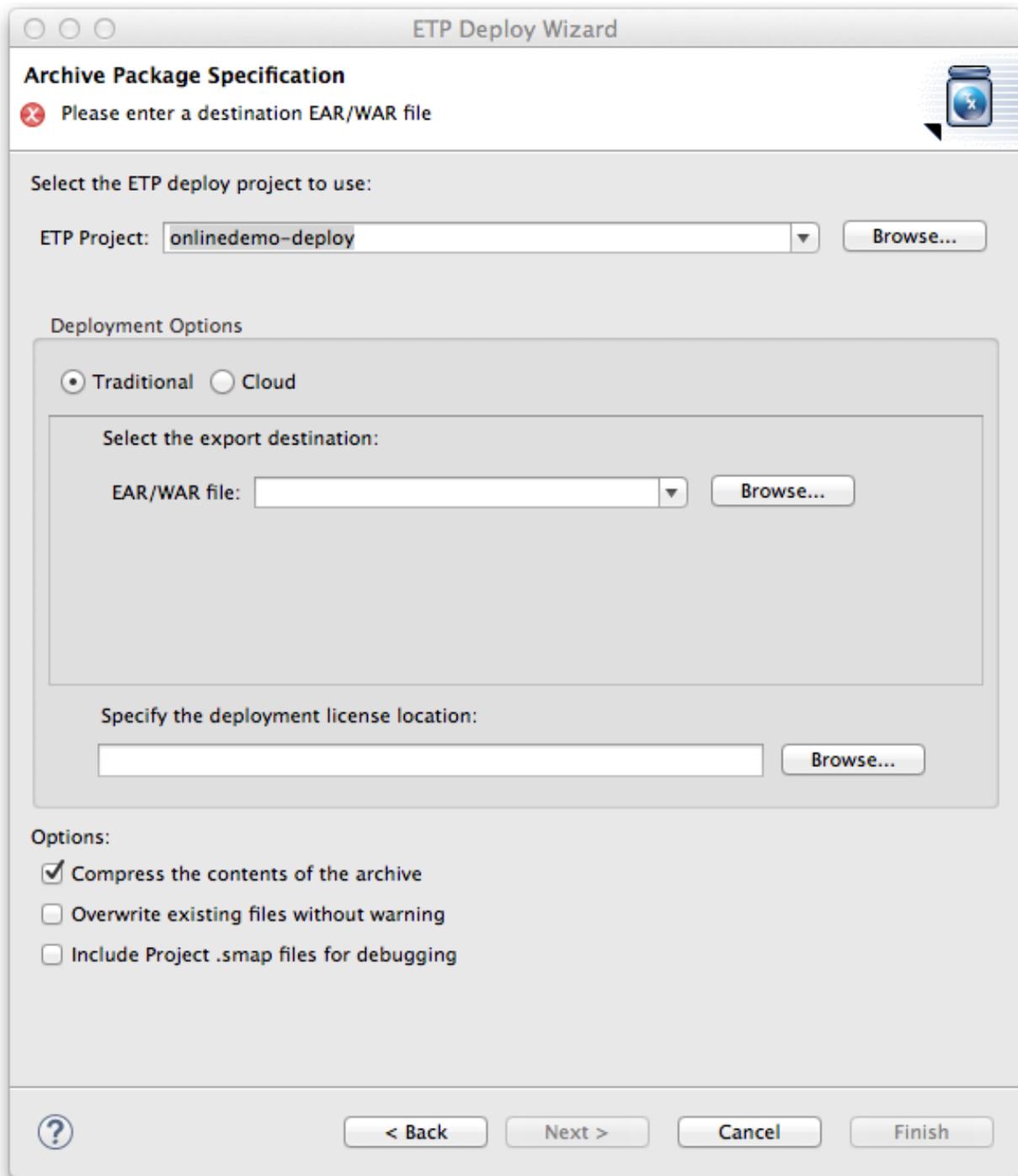
Click 'OK':



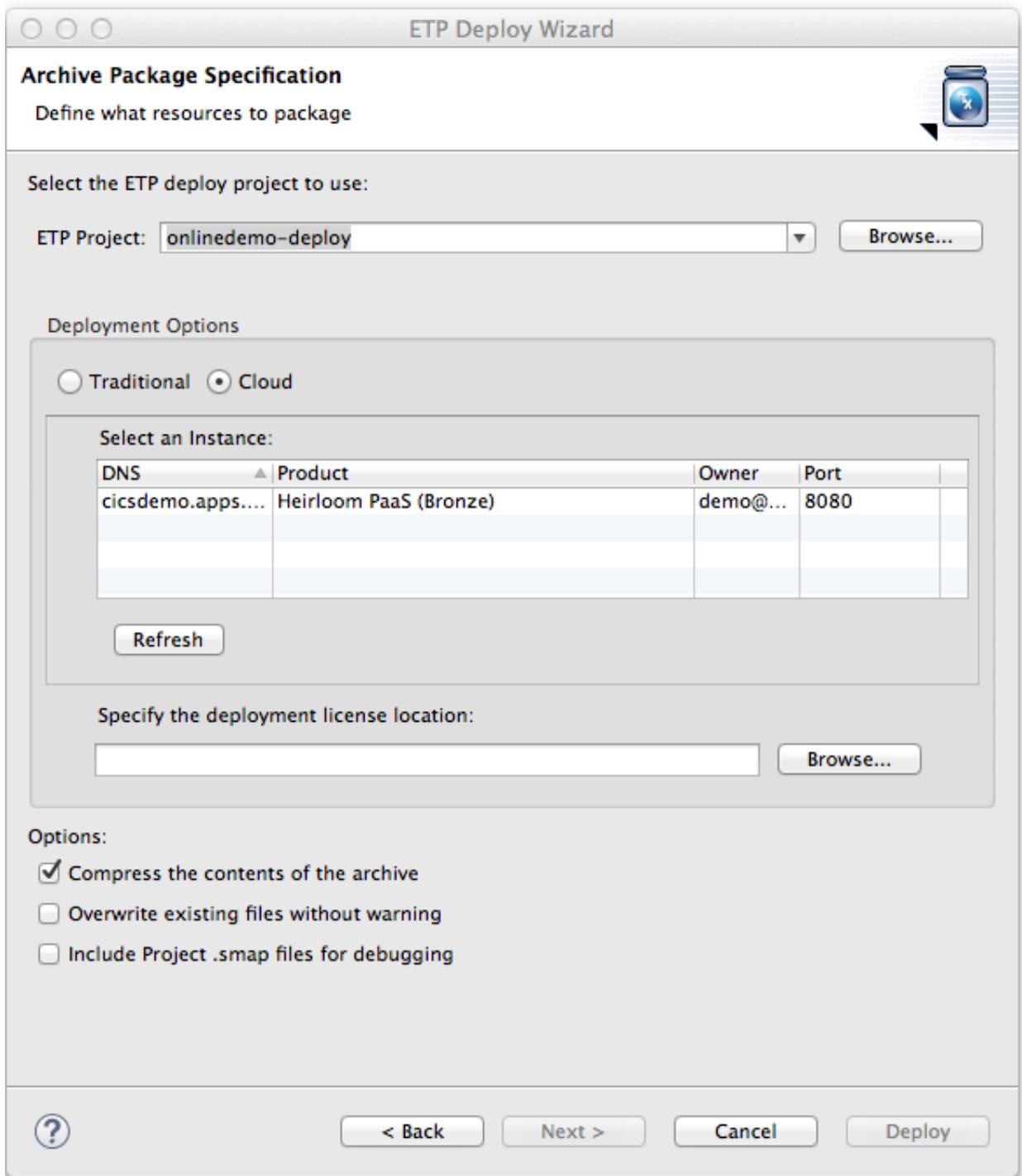
You are now ready to re-export the converted CICS application with read functionality. Right click the onlinedemo-deploy Project and select Export...:



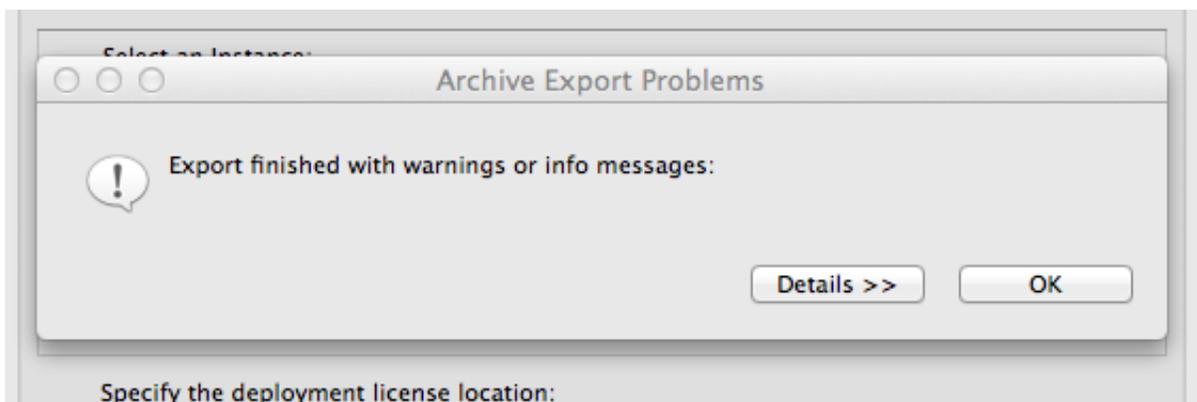
Select the 'Elastic Transaction Platform Deploy Wizard' and click Next:



Click the 'Cloud' radio button:

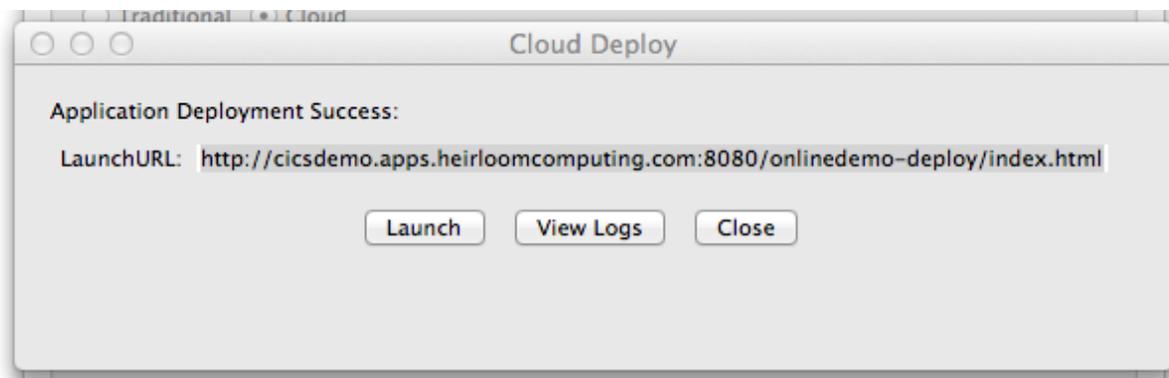


Select your application instance and click Deploy:



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:



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:

The screenshot shows a terminal window with a black background and green text. The main area contains the following text and fields:

- SAMPLE SCREEN** (in red)
- DATE :** (in red)
- TIME :** (in red)
- NAME :** followed by a text input field.
- PHONE :** followed by a text input field.
- MESSAGE : PROVIDE NECESSARY DETAILS** (in red)
- F5=INSERT F3=BACK** (in red)

On the right side of the terminal, there is a vertical control panel with the following elements:

- A grid of 12 function keys: PF1, PF2, PF3, PF4, PF5, PF6, PF7, PF8, PF9, PF10, PF11, PF12.
- A row of three alphanumeric keys: PA1, PA2, PA3.
- A **Clear** button.
- A **Reset** button.
- An **Enter** button at the bottom.

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

← → ↻ 🏠 cicsdemo.apps.heirloomcomputing.com:8080/online-demo-deploy/servlet?transid=ONLR

DEMO READ SCREEN

DATE:

TIME:

NAME :

PHONE :

MESSAGE : PROVIDE NECESSARY DETAILS

F5=INSERT F3=BACK

PF1 PF2
PF3 PF4
PF5 PF6
PF7 PF8
PF9 PF10
PF11 PF12
PA1 PA2
PA3
Clear
Reset
Enter

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

| | | |
|------------------------|-----------------------------------|-------|
| DEMO READ SCREEN | | DATE: |
| NAME : | <input type="text" value="MIKE"/> | TIME: |
| PHONE : | <input type="text" value=""/> | |
| MESSAGE : 512-777-8819 | | |
| F5=INSERT F3=BACK | | |

| | |
|-------|------|
| PF1 | PF2 |
| PF3 | PF4 |
| PF5 | PF6 |
| PF7 | PF8 |
| PF9 | PF10 |
| PF11 | PF12 |
| PA1 | PA2 |
| PA3 | |
| Clear | |
| Reset | |
| Enter | |

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 'Lambda':

AWS services

Find a service by name or feature (for example, EC2, S3 or VM, storage).



Recently visited services



EC2



Billing



Lambda



RDS



Route 53

> All services

Click the 'Create function' button:

View details

Test

Delete

Create function

Click 'Author from scratch' :

Select blueprint

Blueprints are sample configurations of event sources and Lambda functions. Choose a blueprint that best aligns with your desired scenario and customize as needed, or click on **Author from scratch** if you want to author a Lambda function and configure an event source separately. Except where otherwise noted, blueprints are licensed under [CC0](#).

Blueprints

Export

Author from scratch

Filter by tags and attributes or search by keyword



< 1 2 3 4 5 6 7 ... 10 >

Click the grey dotted square:

Configure triggers

You can choose to add a trigger that will invoke your function.

Add trigger

Remove



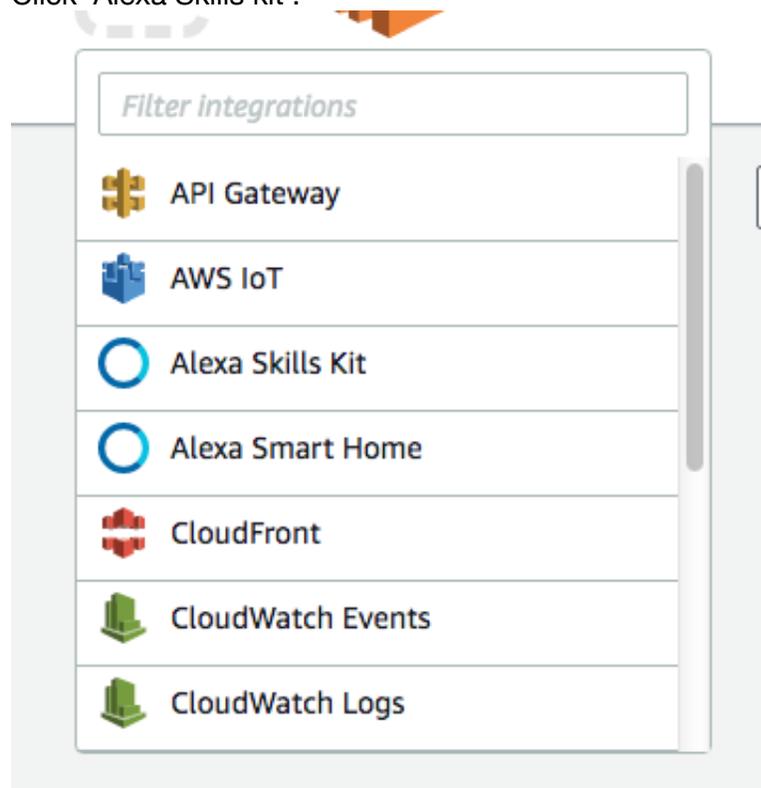
Lambda

Cancel

Previous

Next

Click 'Alexa Skills kit':



Click 'Next':

Configure triggers

You can choose to add a trigger that will invoke your function.

Add trigger Remove

Alexa Skills Kit    **Lambda**

Lambda will add the necessary permissions for Amazon Alexa to invoke your Lambda function from this trigger. [Learn more](#) about the Lambda permissions model.

Cancel Previous Next

Enter Basic information as follows:

Configure function

A Lambda function consists of the custom code you want to execute. [Learn more](#) about Lambda functions.

Basic information

Name*

Description

Runtime*

Fill in Lambda function handler and role as follows:

Lambda function handler and role

Handler*

The filename.handler-method value in your function. For example, "main.handler" would call the handler method defined in main.py.

Role*

Defines the permissions of your function. Note that new roles may not be available for a few minutes after creation. [Learn more](#) about Lambda execution roles.

Lambda will automatically create a role with permissions from the selected policy templates. Note that basic Lambda permissions (logging to CloudWatch) will automatically be added. If your function accesses a VPC, the required permissions will also be added.

Role name*

Enter a name for your new role.

Policy templates

Choose one or more policy templates. A role will be generated for you before your function is created. [Learn more](#) about the permissions that each policy template will add to your role.

Click Next:

▶ Tags

▶ Advanced settings

* These fields are required.

Cancel

Previous

Next

Click 'Create function':

Lambda > Functions > demofunc ARN - arn:aws:lambda:us-east-1:531872380077:function:demofunc

demofunc

Qualifiers ▾ Actions ▾ Test

✔ Congratulations! Your Lambda function "demofunc" has been successfully created and configured with as a trigger in a disabled state. We recommend testing the function behavior before enabling the trigger. ✕

Code Configuration **Triggers** Tags Monitoring

 **Alexa Skills Kit** Delete

ⓘ To configure the Alexa service to work with your Lambda function, go to the [Alexa Developer portal](#).

+ Add trigger 🔄 Refresh triggers

▶ View function policy

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

Click the 'Code' tab:

Code Configuration Triggers Tags Monitoring

Code entry type

Edit code inline ▾

```
1 def lambda_handler(event, context):
2     # TODO implement
3     return 'Hello from Lambda'
```

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':

Code Configuration Triggers Tags Monitoring

Code entry type
Upload a .ZIP file ▼

Function package*
 Upload

For files larger than 10 MB, consider uploading via S3.

Enable encryption helpers
For storing sensitive information, we recommend encrypting values using KMS and the console's encryption helpers.

Environment variables
You can define Environment Variables as key-value pairs that are accessible from your function code. These are useful to store configuration settings without the need to change function code. [Learn more.](#)

| Key | Value | Remove |
|-----|-------|--------|
| | | |

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

Lambda > Functions > demofunc ARN - arn:aws:lambda:us-east-1:531872380077:function:demofunc

demofunc Qualifiers ▼ Actions ▼ Save Save and test

✔ Congratulations! Your Lambda function "demofunc" has been successfully created and configured with as a trigger in a disabled state. We recommend testing the function behavior before enabling the trigger. ✕

Code Configuration Triggers Tags Monitoring

Code entry type
Upload a .ZIP file ▼

Function package*
 Upload CICSPHONE.zip (1.1 MB)

For files larger than 10 MB, consider uploading via S3.

Enable encryption helpers
For storing sensitive information, we recommend encrypting values using KMS and the console's encryption helpers.

Environment variables
You can define Environment Variables as key-value pairs that are accessible from your function code. These are useful to store configuration settings without the need to change function code. [Learn more.](#)

| Key | Value | Remove |
|-----|-------|--------|
| | | |

Click 'Save':

Lambda > Functions > demofunc ARN - arn:aws:lambda:us-east-1:531872380077:function:demofunc

demofunc

Qualifiers ▾ Actions ▾ **Test**

This function contains external libraries. Uploading a new file will override these libraries. ✕

Code Configuration Triggers Tags Monitoring

Code entry type
Edit code inline ▾

```
1 from __future__ import print_function
2 import xml.etree.ElementTree as ET
3
4 def getNumberViaURL(name):
5     import urllib
6     import requests
7     url = 'http://cicsphone.apps.elpaas.com:8080/onldemos/servlet?transid=ONLR'
8     # Prepare the data
9     values = {'DFH_CURSOR': "NAME",
10             'NAME': name,
11             'PHONE' : "-",
12             'MESSB' : "",
13             'DFH_PFS' : "PF5"
14             }
15     data = urllib.urlencode(values)
16
17     r = requests.get(url)
18     jar = r.cookies
19     jarsession = jar['JSESSIONID']
20     cooks={'JSESSIONID': jarsession}
21     headers = {"Content-Type": "application/x-www-form-urlencoded; charset=UTF-8"}
22     r = requests.post(url, data=data, cookies=cooks, headers=headers)
23     if r.status_code != 200:
24         return "Unavailable at this time"
25     index= r.text.find('<pre id="MESS" class="line18 col23 defaultfg brt len40">')
26     if index == -1:
```

You should now edit line SEVEN (7) to be your applications URL, including the transid=ONLR at the end.

Once done, click 'Save':

Lambda > Functions > demofunc ARN - arn:aws:lambda:us-east-1:531872380077:function:demofunc

demofunc

Qualifiers ▾ Actions ▾ Save **Save and test**

This function contains external libraries. Uploading a new file will override these libraries. ✕

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

Notifications

| All | Critical |
|-------------------|----------|
| No Notifications. | |

Announcements

| | | | |
|--|--------------|--|--------------|
| Named Timers and Reminders Capabilities for Alexa Voice Service Products | Aug 3, 2017 | Display Cards for AVS Now Support Dark Background Templates | Jul 26, 2017 |
| Introducing Entertainment Capabilities in Alexa Smart Home | Jul 13, 2017 | Cirrus Logic Accelerates Design of Smart Speakers with the Voice Capture Dev Kit for AVS | Jul 6, 2017 |
| Introducing Support for Optimized Display and Video Interfaces | Jun 28, 2017 | Introducing Smart Home Camera Control with Alexa | Jun 22, 2017 |

Click 'Alexa' in the top menu:

Get started with Alexa

Add new voice-enabled capabilities using the Alexa Skills Kit, or add voice-powered experiences to your connect



Alexa Skills Kit

Easily add new skills to Alexa

Get Started >



Alexa Voice Service

Bring voice capabilities to your connected device

Get Started >

Click the 'Get Started' in the Alexa Skills Kit box:

Building Alexa Skills with the Alexa Skills Kit

Add a New Skill

To learn more about building Alexa skills, see [Getting Started with the Alexa Skills Kit](#). To start building an Alexa skill for free using AWS Lambda, see [Creating an AWS Lambda Function for a Custom Skill](#). We encourage you to visit the [Alexa Developer Forum](#) to collaborate with Alexa team members and fellow Alexa developers.

| Name | Language | Type | Modified | Status | Actions |
|------|----------|------|----------|--------|---------|
|------|----------|------|----------|--------|---------|

Click 'Add a New Skill':

[< Back to All Skills](#)

Create a New Alexa Skill

- Skill Information ✓
- Interaction Model ✓
- Configuration ✓
- SSL Certificate ✓
- Test ✓
- Publishing Information ✓
- Privacy & Compliance ✓

Skill Type
Define a custom interaction model or use one of the predefined skill APIs. [Learn more](#)

Custom Interaction Model
 Smart Home Skill API
 Flash Briefing Skill API
 Video Skill API

Language
Language of your skill: English (U.S.)

Name
Name of the skill that is displayed to customers in the Alexa app. Must be between 2-50 characters.

Invocation Name
The name customers use to activate the skill. For example, "Alexa ask Tide Pooler...".

Global Fields
These fields apply to all languages supported by the skill.

Audio Player
Does this skill use the audio player directives? Yes No [Learn more](#)

Video App
Does this skill use the video app directives? [Learn more](#) Yes No

Render Template
Does this skill use the Render Template directives? Yes No [Learn more](#)

For successful Alexa Skills Certification, please review and follow our [Invocation Name Guidelines](#) as well as our [Certification Requirements](#).

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

English (U.S.)  Add a New Language

- Skill Information 
- Interaction Model 
- Configuration 
- Test 
- Publishing Information 
- Privacy & Compliance 

Skills Beta Testing NEW
Status: Not yet eligible 

 Try the skill builder (beta), an intuitive interface for building your interaction model and creating dialog prompts
Launch Skill Builder BETA

Intent Schema
The schema of user intents in JSON format. For more information, see [Intent Schema](#). Also see [built-in slots](#) and [built-in intents](#).

| | | |
|---|---|--|
|  | 1 | |
|---|---|--|

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:

Intent Schema

The schema of user intents in JSON format. For more information, see [Intent Schema](#). Also see [built-in slots](#) and [built-in intents](#).

```
3   {
4     "slots": [
5       {
6         "name": "person",
7         "type": "AMAZON.US_FIRST_NAME"
8       }
9     ],
10    "intent": "asknumber"
11  }
12 ]
13 }
```

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':

Global Fields

These fields apply to all languages supported by the skill.

Endpoint

Service Endpoint Type:

AWS Lambda ARN (Amazon Resource Name) ⓘ *Recommended* HTTPS

AWS Lambda is a server-less compute service that runs your code in response to events and automatically manages the underlying compute resources for you.

[More info about AWS Lambda](#)
[How to integrate AWS Lambda with Alexa](#)

Account Linking

Do you allow users to create an account or link to an existing account with you? Yes No

[Learn more](#)

Permissions

Request users to access resources and capabilities
Please request permissions to resources and capabilities that are absolutely core to the customer experience delivered by the skill.

Device Address

- Full Address ⓘ
- Country & Postal Code Only ⓘ

Lists Read ⓘ

Lists Write ⓘ

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:

Global Fields

These fields apply to all languages supported by the skill.

Endpoint

Service Endpoint Type: **AWS Lambda ARN (Amazon Resource Name)** ⓘ **HTTPS**

Recommended

AWS Lambda is a server-less compute service that runs your code in response to events and automatically manages the underlying compute resources for you.

[More info about AWS Lambda](#)
[How to integrate AWS Lambda with Alexa](#)

Pick a geographical region that is closest to your target customers: ⓘ

North America **Europe**

North America

Account Linking

Do you allow users to create an account or link to an existing account with you? Yes No

[Learn more](#)

Permissions

Request users to access resources and capabilities

Please request permissions to resources and capabilities that are absolutely core to the customer experience delivered by the skill.

Device Address

Full Address ⓘ
 Country & Postal Code Only ⓘ

Lists Read ⓘ

Lists Write ⓘ

See [Certification Requirements](#) in our technical documentation as you develop your skills and prepare to submit to Amazon.

Save

Submit for Certification

Next

Click 'Next':

 Please complete the Interaction Model tab to start testing this skill.

Enabled This skill is enabled for testing on your account. 

Once you have completed testing on your device, please complete the Description and Publishing Information tab, then submit the skill for certification.

If it passes Amazon's testing and certification process, it will become available to Alexa end users.

You will be able to see your skill in the Skills tab in Alexa App and you can enable the skill and start testing.

After completing your testing please submit the skill for certification. If it passes Amazon's testing and certification process, it will become available to Alexa end users

The skill is available in "Skills > Your Skills" page of the Alexa App when you select 'Yes' above. You can then enable the skill and test its functionality by asking Alexa, **ask phone demo**

 For successful Alexa Skills Certification, please test for our requirements on [Session Management](#) and [Error Handling](#).

Voice Simulator

Hear how Alexa will speak a response entered in plain text or SSML. [Learn more about supported SSML tags.](#)

For example: Here is a word spelled out: `<say-as interpret-as="spell-out">hello</say-as>`.

Listen 

Service Simulator

Use Service Simulator to test your lambda function: 

Note: Service Simulator does not currently support testing audio player directives, dialog model, customer permissions and customer account linking.

TextJSON

Enter Utterance

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:

Service Simulator

Use Service Simulator to test your lambda function: `arn:aws:lambda:us-east-1:531872380077:function:demofunc`

Note: Service Simulator does not currently support testing audio player directives, dialog model, customer permissions and customer account linking.

Text **JSON**

Enter Utterance

what is the phone number for Mike

Ask phonedemo Reset

Click 'Ask phonedemo':

Service Simulator

Use Service Simulator to test your lambda function: `arn:aws:lambda:us-east-1:531872380077:function:demofunc`

Note: Service Simulator does not currently support testing audio player directives, dialog model, customer permissions and customer account linking.

Text **JSON**

Enter Utterance

what is the phone number for Mike

Ask phonedemo Reset

Lambda Request

```
1 {
2   "session": {
3     "new": true,
4     "sessionId": "SessionId.0512abd1-6bd5-416
5     "application": {
6       "applicationId": "amzn1.ask.skill.33bf6
7     },
8     "attributes": {},
9     "user": {
10      "userId": "amzn1.ask.account.AHRT3V72VA
11    }
12  },
13  "request": {
14    "type": "IntentRequest",
15    "requestId": "EdwRequestId.95614632-5003-
16    "intent": {
```

Lambda Response

```
1 {
2   "version": "1.0",
3   "response": {
4     "outputSpeech": {
5       "text": "The number for mike is 512-7
6       "type": "PlainText"
7     },
8     "card": {
9       "content": "The number for mike is 51
10      "title": "Heirloom CICS Number App"
11    },
12    "reprompt": {
13      "outputSpeech": {
14
```

Listen 

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:

Service Simulator

Use Service Simulator to test your lambda function: `arn:aws:lambda:us-east-1:531872380077:function:demofunc` ↓

Note: Service Simulator does not currently support testing audio player directives, dialog model, customer permissions and customer account linking.

The screenshot shows the AWS Service Simulator interface. At the top, there are two tabs: 'Text' (selected) and 'JSON'. Below the tabs is a text input field containing the utterance 'what is the phone number for john'. There are two buttons: 'Ask phonedemo' and 'Reset'. Below the input field, there are two panels: 'Lambda Request' and 'Lambda Response'. The 'Lambda Request' panel shows a JSON object with session information, user information, and request details. The 'Lambda Response' panel shows a JSON object with version, response, output speech, card, and reprompt information. A 'Listen' button with a play icon is located at the bottom right of the 'Lambda Response' panel.

```
1 {
2   "session": {
3     "new": false,
4     "sessionId": "SessionId.0512abd1-6bd5-416",
5     "application": {
6       "applicationId": "amzn1.ask.skill.33bf6"
7     },
8     "attributes": {},
9     "user": {
10      "userId": "amzn1.ask.account.AHRT3V72VA"
11    }
12  },
13  "request": {
14    "type": "IntentRequest",
15    "requestId": "EdwRequestId.cd94f8a4-4968-"
16  }
```

```
1 {
2   "version": "1.0",
3   "response": {
4     "outputSpeech": {
5       "text": "The number for John is UNKNC",
6       "type": "PlainText"
7     },
8     "card": {
9       "content": "The number for John is UN",
10      "title": "Heirloom CICS Number App"
11    },
12    "reprompt": {
13      "outputSpeech": {
14
```

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:

demofunc

Qualifiers ▾

Actions ▾

Test

⚠ This function contains external libraries. Uploading a new file will override these libraries.

✕

Code

Configuration

Triggers

Tags

Monitoring

Code entry type

Edit code inline ▾

```
1 from __future__ import print_function
2 import xml.etree.ElementTree as ET
3
4 def getNumberViaURL(name):
5     import urllib
6     import requests
7     url = 'http://cicsphone.apps.elpaas.com:8080/onldemos/servlet?transid=ONLR'
8     # Prepare the data
9     values = {'DFH_CURSOR': "NAME",
10             'NAME': name,
11             'PHONE' : "-",
12             'MESSB' : "",
13             'DFH_PFS' : "PFS"
14             }
15     data = urllib.urlencode(values)
16
17     r = requests.get(url)
18     jar = r.cookies
19     jarsession = jar['JSESSIONID']
20     cooks={'JSESSIONID': jarsession}
21     headers = {"Content-Type": "application/x-www-form-urlencoded; charset=UTF-8"}
22     r = requests.post(url, data=data,cookies=cooks,headers=headers)
23     if r.status_code != 200:
24         return "Unavailable at this time"
25     index= r.text.find('<pre id="MESS" class="line18 col23 defaultfg brt len40">')
26     if index == -1:
```

Click 'Actions' and then 'Configure New Test Event':

Input test event

Use the editor below to enter an event to test your function with. You can edit the event again by choosing **Configure test event** in the Actions list. Note that changes to the event will only be saved locally.

Sample event template

Hello World ▼

```
1 - {
2 -   "session": {
3 -     "new": true,
4 -     "sessionId": "SessionId.ee8ca69d-19dd-4aa1-a641-3120d0139003",
5 -     "application": {
6 -       "applicationId": "amzn1.ask.skill.33bf69db-bdb3-4887-a90c-6ba30c889a9c"
7 -     },
8 -     "attributes": {},
9 -     "user": {
10 -      "userId": "amzn1.ask.account.AHRT3V72VATBDTZ76PEW7ATFPDVRW2DMNMTLLWZIY7EWHAT7Z3
11 -    }
12 -  },
13 -  "request": {
14 -    "type": "IntentRequest",
15 -    "requestId": "EdwRequestId.5623b0ce-eb54-4c95-aeac-e0bafae78c2a",
16 -    "intent": {
17 -      "name": "asknumber",
18 -      "slots": {
19 -        "person": {
20 -          "name": "person",
21 -          "value": "mike"
22 -        }
23 -      }
24 -    }
25 -  },
26 -  "locale": "en-US",
27 -  "timestamp": "2017-08-15T20:58:42Z"
28 - }
```

Cancel Save Save and test

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

Lambda > Functions > demofunc ARN - arn:aws:lambda:us-east-1:531872380077:function:demofunc

demofunc

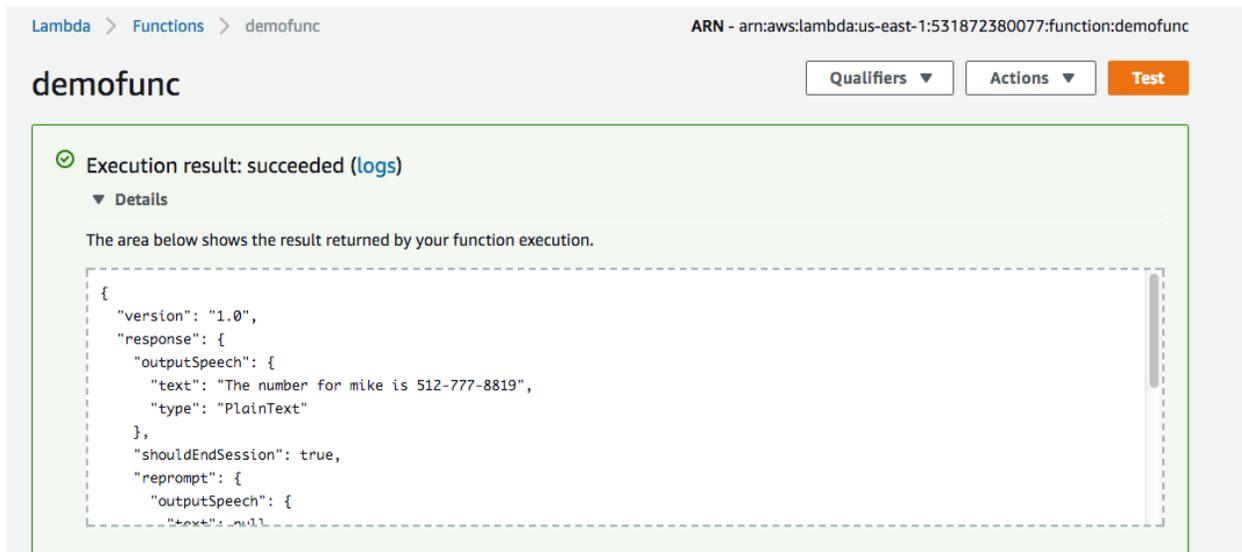
Qualifiers ▼ Actions ▼ Test

✔ Execution result: succeeded ([logs](#))

▶ Details

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

resulting JSON (or any errors):



The screenshot shows the AWS Lambda console for a function named 'demofunc'. The breadcrumb navigation is 'Lambda > Functions > demofunc'. The ARN is 'arn:aws:lambda:us-east-1:531872380077:function:demofunc'. There are three buttons: 'Qualifiers', 'Actions', and 'Test'. The 'Test' button is highlighted in orange. Below the buttons, there is a green box indicating a successful execution. The text reads 'Execution result: succeeded (logs)' with a green checkmark icon. Underneath, there is a 'Details' section with a downward arrow. The text says 'The area below shows the result returned by your function execution.' Below this text is a dashed box containing a JSON object. The JSON object is as follows:

```
{
  "version": "1.0",
  "response": {
    "outputSpeech": {
      "text": "The number for mike is 512-777-8819",
      "type": "PlainText"
    },
    "shouldEndSession": true,
    "reprompt": {
      "outputSpeech": {
        "text": null
      }
    }
  }
}
```

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.