Alexa

Alexa Skills Kit

Alexa Voice Service

Alexa Fund

Steps to Create a Smart Home Skill

Introduction

You create a smart home skill in the Amazon developer portal. Once you've created the skill, you provide skill information, configuration and an endpoint called a *skill adapter*. A skill adapter is an AWS Lambda function (an Amazon Web Services offering) that handles requests from the Smart Home Skill API and communicates with a device cloud. To enable this communication, the device cloud should support the OAuth 2.0 authorization code grant type. You will need information about the

Smart Home Skill API

- Smart Home Skill API Documentation
  - Understanding the Smart Home Skill API
  - **Steps to Create a Smart Home Skill**
  - Smart Home Skill API Reference
  - Smart Home Skill Publishing Guide
  - Developing Smart Home Skills in Multiple Languages
  - Providing Scenes in a Smart Home Skill
  - Linking an Alexa User with a User in Your System

authentication endpoint, client ID and secret to complete the smart home skill registration.

This topic provides the steps for creating a skill in the developer portal, creating a skill adapter as an AWS Lambda function, configuring and testing your skill, and submitting it for certification.

Register as an Amazon Developer and Create a Skill

To configure a new smart home skill, you need an account on the Amazon Developer Portal. If you don't already have an account, go to https://developer.amazon.com/login.html and create an account. Registering is free. Once you've registered:

1. Open the Amazon Developer Portal in a browser and log in.
2. Navigate to the Alexa section by clicking **Apps & Services** and then clicking **Alexa** in the top navigation.
3. In the Alexa Skills Kit box, click **Get Started**.
4. Click the **Add a New Skill** button.
5. On the **Skill Information** page, select **Smart Home Skill API**, and enter the **Name** for your skill. Note that your skill's name must not indicate that it's functionality is limited. For example, you should not use "basic", "reduced" or "simple" in the title of your skill.
6. Select the language for your skill. You can add additional languages later.
7. Click **Save**. and copy the **Application Id** for the skill to the clipboard.

Create a Lambda Function

You will need an AWS account and some basic knowledge of AWS. The skill adapter portion of your smart home skill is hosted as a Lambda

Other Resources

- Smart Home Skill API Kit Forum
- Skills Beta Testing
- Getting Started with the Alexa Skills Kit
- Custom Skills
- Flash Briefing Skills

function on AWS. AWS Lambda is a service that lets you run code in the cloud without managing servers. The Smart Home Skill API sends your skill adapter requests and your code inspects the request, takes any necessary actions and then sends back a response.

To get started, sign in to your AWS Account, and create the Lambda function.To do this:

1. On the AWS Console, under **Compute**, select **Lambda**.
2. Make sure you've selected the **N.Virginia** for English (US) skills or the **EU (Ireland**) region for English (UK) and German skills. The region is displayed in the upper right corner. Providing your Lambda function in the correct region prevents latency issues.
3. Click **Create a Lambda function**.
4. On the **Select blueprint** page, select the **Blank Function**.
5. On the **Configure triggers page,** click the the empty box to configure a new trigger.
6. Select **Alexa Smart Home** entry from the drop-down list that appears. Enter the **Application Id** from the developer portal.
7. Optionally check **Enable trigger**. This enables the Amazon Alexa service to call your Lambda function. It's recommended that you enable the trigger later, once you have completed your implementation and testing in the Lambda portal.
8. Click **Next.**
9. On **Step 3: Configure function**, enter the following:
    - **Name**: A name for your Lambda function. This name is appears in the AWS console and is also returned by the AWS command-line interface (CLI) **ListFunctions** API. It should be unique across your Lambda functions.
    - **Description**: Optionally provide a description.
    - **Runtime**: Select **Python 2.7**. Lambda also supports Node.js and Java, but the example you will work with is in Python.

10. For **Lambda function code**, make sure **Edit code inline** is selected.

11. Copy and paste the following code into the code editor. This code provides a starting point for implementing a skill adapter. The code determines the request type by determining the namespace and name values in the message header. The `namespace` specifies whether the request is a discovery or control request, and either the `handleDiscovery` or `handleControl` function is called. For a control request, the `name` property specifies the kind of request. In this example code, the only request type detected is a `TurnOnRequest`. The response is not fully implemented. You should handle every type of request a user would make to your skill, and provide the correct response headers and payload. For information about all of the request and response messages, see Smart Home Skill API Reference.

```
def lambda_handler(event, context):
    access_token = event['payload']['accessToken']

    if event['header']['namespace'] == 'Alexa.ConnectedHome.Discovery':
        return handleDiscovery(context, event)

    elif event['header']['namespace'] == 'Alexa.ConnectedHome.Control':
        return handleControl(context, event)

def handleDiscovery(context, event):
    payload = ''
    header = {
        "namespace": "Alexa.ConnectedHome.Discovery",
        "name": "DiscoverAppliancesResponse",
        "payloadVersion": "2"
        }

    if event['header']['name'] == 'DiscoverAppliancesRequest':
        payload = {
            "discoveredAppliances":[
                {
                    "applianceId":"device001",
                    "manufacturerName":"yourManufacturerName",
                    "modelName":"model 01",
                    "version":"your software version number here.",
                    "friendlyName":"Smart Home Virtual Device",
                    "friendlyDescription":"Virtual Device for the Sample
                    "isReachable":True,
                    "actions":[
                        "turnOn",
                        "turnOff"
                    ],
                    "additionalApplianceDetails":{
```

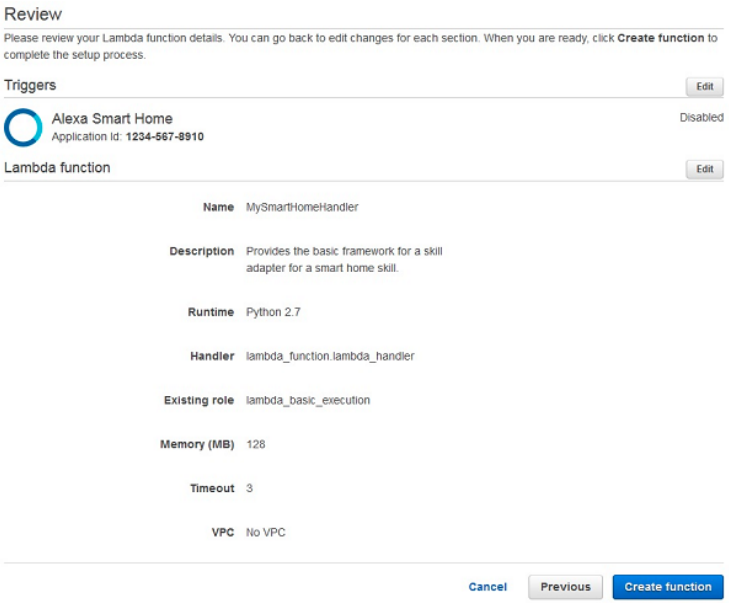
```
                    "extraDetail1":"optionalDetailForSkillAdapterToRe
                    "extraDetail2":"There can be multiple entries",
                    "extraDetail3":"but they should only be used for
                    "extraDetail4":"This is not a suitable place to m
                }
            }
        ]
    }
    return { 'header': header, 'payload': payload }

def handleControl(context, event):
    payload = ''
    device_id = event['payload']['appliance']['applianceId']
    message_id = event['header']['messageId']

    if event['header']['name'] == 'TurnOnRequest':
        payload = { }

    header = {
        "namespace":"Alexa.ConnectedHome.Control",
        "name":"TurnOnConfirmation",
        "payloadVersion":"2",
        "messageId": message_id
        }
    return { 'header': header, 'payload': payload }
```

12. On the Configuration For **Handler**, for this example, leave the default name **lambda_function.lambda_handler.** A function handler is the main entry point for a Lambda function, and you specify it with the format *fileName.entryPointFunctionName*. For this example, the file name is `lambda_function` and the `lambda_handler` function is the entry point in this code.

13. For **Role**, select a role with lambda_basic_execution policy attached. To create a new role see, Create the Execution Role topic in the Lambda documentation

14. Leave the **Advanced settings** set to the defaults and click **Next**.

15. On the **Step 4: Review** page, you should see something similar to the following:



16. Click **Create function.**

17. On the summary page, copy the **Amazon Resource Name (ARN)** for your Lambda function, which is found in the upper right corner, into your clipboard. You will use this value when you configure the smart home skill in the developer portal. The ARN should be similar to the following:

    ```
    ARN - arn:aws:lambda:us-east-1:025363977285:function:myskill
    ```

18. If you want to perform a quick test of the Discovery request in the Lambda code editor, you can click on **Test** and for the **Sample event template**, leave the default **Hello World**. Replace the entire contents in the test editor with the following code.

    ```
    {
      "header": {
        "payloadVersion": "2",
        "namespace": "Alexa.ConnectedHome.Discovery",
        "name": "DiscoverAppliancesRequest"
    ```

```
  },
  "payload": {
    "accessToken": "someaccesstoken"
  }
}
```

19. Select Save and test. If successful, the *Execution* result should be similar to the following:

```
{
  "header": {
    "payloadVersion": "2",
    "namespace": "Alexa.ConnectedHome.Discovery",
    "name": "DiscoverAppliancesResponse"
  },
  "payload": {
    "discoveredAppliances": [
      {
        "friendlyDescription": "Virtual Device for the Sample Hel
        "modelName": "model 01",
        "additionalApplianceDetails": {
          "extraDetail3": "but they should only be used for refer
          "extraDetail2": "There can be multiple entries",
          "extraDetail1": "optionalDetailForSkillAdapterToReferen
          "extraDetail4": "This is not a suitable place to mainta
        },
        "version": "your software version number here.",
        "manufacturerName": "yourManufacturerName",
        "friendlyName": "Smart Home Virtual Device",
        "actions": [
          "turnOn",
          "turnOff"
        ],
        "applianceId": "device001",
        "isReachable": true
      }
    ]
  }
}
```

Finish Registering Your Skill in the Developer Portal

To finish registering your skill, you must configure OAuth 2.0 for your skill, which is used for account linking. You can use Login with Amazon or another provider for this section.

1. Navigate back to your skill in the Developer Portal. To do this, log in, and navigate to the Alexa section by clicking **Apps & Services** and then clicking **Alexa** in the top navigation. Select your skill from the displayed list.
2. You can click past the Interaction model tab. The interaction model, which is what a user can say to invoke the Smart Home Skill API, is predefined so does not need to be specified. For details of the smart home voice interaction model, see the utterances listed with each reference entry in the Smart Home Skill API Reference.
3. On the **Configuration** page, select the language tab for your skill, and in the **Endpoint** field, copy in the ARN number from the Lambda function you created.
4. You must enable **Account Linking** for your skill. This allows end-users to associate their cloud-enabled devices with the Alexa skill, and where you need OAuth 2.0 information for the device cloud your skill communicates with. To learn more about account linking, see Linking an Alexa User with a User in Your System. It's important to note that the OAuth provider must have a certificate signed by an Amazon-approved certificate authority or account linking will fail. In addition, access tokens provided by your system must have a lifetime of at least 6 minutes. This means the `expires_in` parameter of your access token response must be greater or equal to 360.

   The following fields are required:

- **Authorization URL**: This is the URL for your web site's login page. Refer back to Adding Account Linking Support to Your Login Page for information about how this page is used when users link their accounts.
- **Client ID**: An identifier your login page uses to recognize that the request came from your skill. This value is passed to your authorization URL in the client_id parameter. You will use authorization code grant, which means this value is also part of the client credentials that the Alexa service includes when requesting an access token from the **Access Token URI**.
- **Redirect URL**: Enabling account linking displays your Redirect URL. This is the URL to which your login page must redirect the user after they are authenticated. This URL should be white-listed with your OAuth provider. Each language version of your skill will have a different redirect URL.
- **Authorization Grant Type**: **Authorization Code Grant** is preselected, as it's the supported grant type for Smart Home Skill API. For Authorization Code Grant, you must also fill in the following:
    - **Access Token URI**: The URL for the OAuth server that provides the access and refresh tokens, and responds to token requests and token refresh requests.
    - **Client Secret**: A credential you provide that lets the Alexa service authenticate with the **Access Token URI**. This is combined with the Client Id to identify the request as coming from Alexa.
    - **Client Authentication Scheme**: Identifies the type of authentication

Alexa should use when requesting tokens from the Access Token URI.

- **Privacy Policy URL**: A URL for a page with your privacy policy. This link is displayed in the Alexa app and is required for smart home skills.

5. On the **Test** page, make sure that your skill is enabled for testing by choosing **Yes**.

Test Your Skill

When your implementation is complete, and you've tested your skill adapter in Lambda, you can functionally test your smart home skill. You will test your skill with an Alexa-enabled device. To do this:

1. Make sure the trigger is enabled for your skill adapter.
    - In the Lambda console, select your smart home skill, and on the **Triggers** tab, click **Enable** to enable the **Alexa Smart Home** trigger.
2. Find your skill in the Alexa app and enable and account-link your skill to the device cloud it is designed to work with. To find your skill in the Alexa app:
    - Go to **Skills**, and "Your Skills" in the upper right corner. Find your skill in the list. If you want to remove account linking later, you should disable your skill in the **Skills** tab.
3. Discover and manage devices in the **Your Devices** section in the **Smart Home** tab of the Alexa app.
4. Give Alexa commands using the utterances your skill supports with device names you've set for devices in your account-linked device cloud. Make sure and test the skill with valid utterances and invalid ones, and in a variety of conditions, such as with a target device powered off.

5. When you are satisfied with your smart home skill's performance, you can either beta test your skill, or submit it for certification.

Submit Your Skill for Certification

The final step is getting your skill certified. This way it will appear on the skills tab of the Alexa app and the general public can use your skill. To start this process, do the following:

- Go back to your skill in the developer portal. To do this, open the Alexa section and click **Get Started**.
- Select your smart home skill and navigate to the **Publishing Information** page. This information displays with your skill in the Alexa app. For detailed instructions on completing this section, see the Smart Home Skill Publishing Guide.
- **Descriptions**: Complete the **Short Skill** and **Full Skill** descriptions.
- **Category**: This is automatically set to **Smart Home** and cannot be changed.
- **Keywords**: (optional) Add keyword appropriate for your skill.
- **Images**: Add small and large icons per the guidelines specified.
- **Test Instructions**: You must:
    - Provide valid test account credentials (username and password) for your skill so the certification team can complete account
    - Make sure the the device cloud for this test account contains at least one discoverable device for testing purposes.
    - Provide any additional instructions necessary to test your skill.

- Click **Next**.
- On the **Privacy and Compliance** page, answer the questions, and click **Submit for Certification**.

Congratulatons! You've successfully created and submitted a smart home skill for certification. You should hear from the certification team in a few days with the status of your skill; whether it's been accepted or you have some work to do before it meets our certification guidelines.

Submit Changes to a Live Skill

Once your skill is live, any changes to the skill must be certified. For example, your skill must be recertified as part of changing the skill metadata or changing the AWS Lambda function. The certification review varies depending on the changes, and updating the Lambda function may trigger additional review.

When you submit your updated skill to the developer portal, in the **Testing Instructions** field, add a change log that provides a detailed description of all changes.

To Update Your Lambda function

Do the following to update your AWS Lambda function:

1. Create a new Lambda function with a new ARN, and copy your live Lambda function into the new Lambda function.
2. Make your updates in the new Lambda function. Avoid updating the live Lambda function that supports your live skill, and avoid Lambda aliases or versioning because they do not work reliably once your skill is updated.
3. In the development stage for your skill, update the Lambda ARN configuration to point to the new Lambda ARN.

After your skill is recertified and pushed live, your skill points to the new Lambda ARN. You can then deprecate the previous Lambda function or use the function for future development or updates.