2019

OpenGesture Skill

PC SKILLS FRAMEWORK

MOLOTI NAKAMPE & THABO KOEE

Table of Contents

Introduction

Purpose

Hard to believe we started and completed the development of the OpenGesture Skill in 30 days. It's been busy! I certainly feel OpenGesture Skill is working well, and as intended. "If I have seen further than others, it is by standing upon the shoulders of giants" Sir Isaac Newton.

This document provides developer guidance for the OpenGesture Skill using Alexa* with the PC Skills Framework (PCSF) library. The skill allows users to perform hand gesture detection and recognition process and interpret them as text and speech.

In addition, OpenGesture Application is built using Inference Engine of the Intel OpenVINO Toolkit and Unity Game Engine. The inference engine backend is used by default since OpenCV 3.4.2 (OpenVINO 2018.R2) when OpenCV is built with the Inference engine support. Also, the Inference engine backend is the only available option (also enabled by default) when the loaded model is represented in OpenVINO™ Model Optimizer format.

The targeted audiences for this document are system integrators, architects, designers, developers, validation engineers, and scientists.

Scope

The following items are included in scope:

1. OpenGesture Skill PC Architecture
2. OpenGesture Skill Intent and utterance creation requirement
3. OpenGesture Skill client application implementation on Windows 10 PC
4. Reference Code Snippets
5. Screen Shots

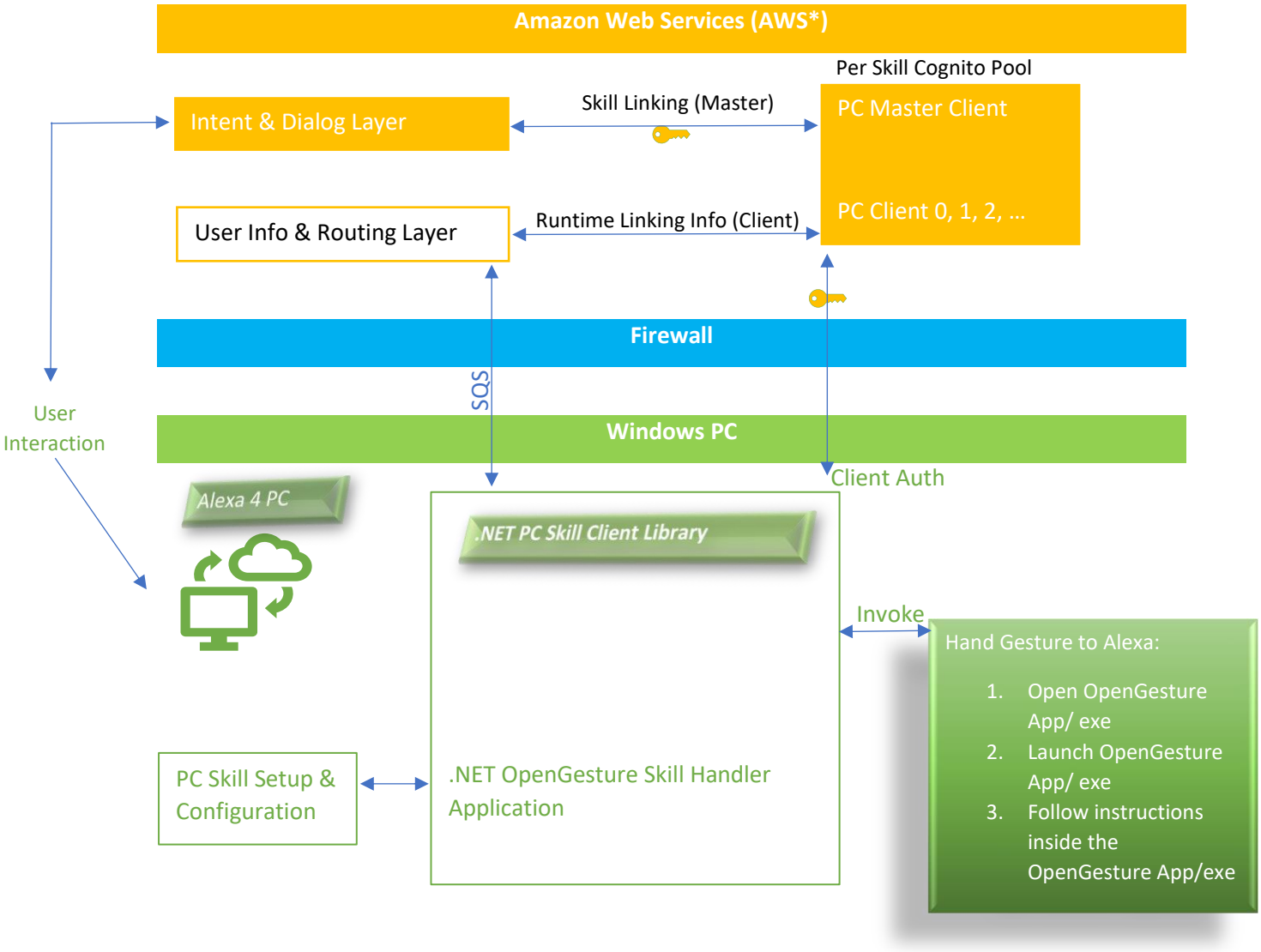
Acronyms

Acronym	Meaning
AWS*	Amazon Web Services
PCSF	PC Skills Framework
SQS	Simple queue service
ASK	Alexa* Skills Kit
A4PC	"Alexa for PC" software for Windows*

Requirements

- OpenGesture App: https://drive.google.com/open?id=1GXccm56IurkrxyEZZcYUBN9DFXaYFL5Q
- AWS Developer Account
- Alexa Development Console Account
- PC with the following installed:
    1. Intel® Core™ i5 processor or higher
    2. Intel OpenVINO Toolkit for Windows
    3. 4GB of RAM
    4. AWS .NET SDK
    5. Web browser

Architecture Diagram



Operational Flow

Below steps walks through the Skill activity and the execution of a skill intent will behave as follows:

1. User invokes OpenGesture skill by using an Alexa device or an A4PC application.

2. User-invoked OpenGesture skill gets mapped with the skill intent and utterances defined in the AWS developer console.

3. The skill uses the routing translation layer to move the intent request to the PC without knowledge of the user or PC involved.

4. The translation and routing component (usually running in AWS Lambda*) uses the master client token to map the originating Alexa device to a specific PC Client SQS queue pair.

5. Translation and routing layer bundles the intent and places it in the outbound queue. Upon deposit, the PC Skill client's long polling returns the bundled intent object as available queued data. The translation and routing layer does a long poll on the return SQS queue so it may be notified immediately upon handling of the intent.

6. The OpenGesture skill handler uses the intent to affect the PC in some manner, or just creates a contextual response.

7. OpenGesture Skill handler function receives the slot values from the request body (AWS Lambda response).

8. Based on the OpenGesture skill and the slot values, search for OpenGesture App is performed by a standard Windows* search using advanced query syntax (AQS). Once the OpenGesture App is found in the users PC, the user should run it and select gesture from the menu scene.

9. AQS is created in string format and opened using Windows Explorer to generate results.

10. The response is placed in the SQS return queue for consumption by the translation and routing layer.

11. The routing and translation layer returns the response retrieved from the return queue to the OpenGesture skill.

12. The skill's response is uttered by the requesting Alexa device or A4PC application.

OpenGesture Skill

Intent

Below steps walkthrough of creating Intents, utterances, slots and slot types in the Amazon developer portal.

Create an intent in the Amazon developer console site (https://developer.amazon.com) that contains the sample utterances. For this skill the Intent is named as “OpenGestureIntent”

Intent Name: OpenGestureIntent

OpenGesture Sample Utterances

Create the following sample utterances for OpenGesture skill
"run application at {appLocation}", "run application at {appLocation} modified {date}", "run application type {appType} ", "run application name {searchQuery}", "run application {searchQuery}", "run application type {appType} at {appLocation}", "run application type {appType} at {appLocation} modified {date}", "start application at {appLocation}", "start application at {appLocation} modified {date}", "start application type {appType} ", "start application name {searchQuery}", "start application {searchQuery}", "start application type {appType} at {appLocation} ", "start application type {appType} at {appLocation} modified {date}", "launch application at {appLocation}", "launch application at {appLocation} modified {date}", "launch application type {appType} ", "launch application name {searchQuery}", "launch application {searchQuery}", "launch application type {appType} at {appLocation} ", "launch application type {appType} at {appLocation} modified {date} ", "execute application at {appLocation} modified {date}", "execute application at {appLocation}", "execute application type {appType} ", "execute application name {searchQuery}", "execute application {searchQuery}", "execute application type {appType} at {appLocation} ", "execute application type {appType} at {appLocation} modified {date} ", "open application at {appLocation} modified {date}", "open application at {appLocation}", "open application type {appType} modified {date}", "open application type {appType} at {appLocation}", "open application type {appType} at {appLocation} modified {date}", "open application name {searchQuery}", "open application {searchQuery}"

** Here {appType}, {appLocation}, {date}, and {searchQuery} are slot values.

Intent Slots

Name: searchQuery
Type: AMAZON.SearchQuery
Refer Amazon developer console for Amazon slot types (https://developer.amazon.com).
Details of custom slots can be found in the following JavaScript* Object Notation (JSON) code snippets. Full code maybe found here: https://github.com/TebogoNakampe/OpenGesture-Skill/blob/master/Intent/OpenGesture.json

Name: appLocation
Type: LIST_OF_APP_LOCATIONS

```
{
            "name": "LIST_OF_APP_LOCATIONS",
          "values": [
            {
              "name": {
                "value": "recent",
                "synonyms": [
                  "last"
                ]
              }
            },
            {
              "name": {
                "value": "user",
                "synonyms": [
                  "my libraries",
                  "my user"
                ]
              }
            },
            {
              "name": {
                "value": "videos"
              }
            },
            {
              "name": {
                "value": "music",
                "synonyms": [
                  "my music"
                ]
              }
            },
            {
              "name": {
                "value": "pictures"
              }
            },
            {
              "name": {
                "value": "everywhere",
                "synonyms": [
                  "root"
                ]
              }
            },
            {
              "name": {
                "value": "downloads",
                "synonyms": [
                  "download",
                  "my downloads"
                ]
              }
            },
            {
              "name": {
                "value": "docs",
                "synonyms": [
                  "document",
                  "documents"
                ]
              }
            }
          ]
        },
```

Name: appType
Type: LIST_OF_APP_TYPES


```
{
          "name": "LIST_OF_APP_TYPES",
          "values": [
            {
              "name": {
                "value": "applications",
                "synonyms": [
                  "application",
                  "exe"
                ]
              }
            },
            {
              "name": {
                "value": "unity",
                "synonyms": [
                  "unity",
                  "exe"
                ]
              }
            },
            {
              "name": {
                "value": "folders"
              }
            },
            {
              "name": {
                "value": "app",
                "synonyms": [
                  "apps",
                  "exe"
                ]
              }
            },
            {
              "name": {
                "value": "program",
                "synonyms": [
                  "programs",
                  "exe"
                ]
              }
            },
            {
              "name": {
                "value": "executable",
                "synonyms": [
                  "exe",
                  "exe"
                ]
              }
            },
            {
              "name": {
                "value": "windowsapp",
                "synonyms": [
                  "xap",
                  "xap"
                ]
              }
            },
            {
              "name": {
                "value": "UniversalWindowsPlatform",
                "synonyms": [
                  "UWP",
                  "APPXs"
                ]
              }
            }
          ]
        },
```

Name: date
Type: LIST_OF_DATE_TYPES


```
{
          "name": "LIST_OF_DATE_TYPES",
          "values": [
            {
              "name": {
                "value": "last year",
                "synonyms": [
                  "past year"
                ]
              }
            },
            {
              "name": {
                "value": "last month",
                "synonyms": [
                  "past month"
                ]
              }
            },
            {
              "name": {
                "value": "last week",
                "synonyms": [
                  "past week"
                ]
              }
            },
            {
              "name": {
                "value": "this week",
                "synonyms": [
                  "current week"
                ]
              }
            },
            {
              "name": {
                "value": "yesterday",
                "synonyms": [
                  "other day"
                ]
              }
            },
            {
              "name": {
                "value": "today",
                "synonyms": [
                  "right now",
                  "now"
                ]
              }
            }
          ]
        }
      ]
    }
  }
}
```

Handler Declaration

Register the handler in which the request is being handled, so that the framework calls the respective intent handler when corresponding intent is triggered. It is declared as follows:

```
private static Dictionary<string, Func<IRCExample, string, string, string, string>>
RequestHandlers =
            new Dictionary<string, Func<IRCExample, string, string, string, string>>()
            {
                .
                .
                { "OpenGestureIntent",  ((thisObj, target, topic, requestbody)
                thisObj.OpenGesture(target,topic,requestbody))}
                .
                .
            };
```

OpenGesture Skill Code Snippets

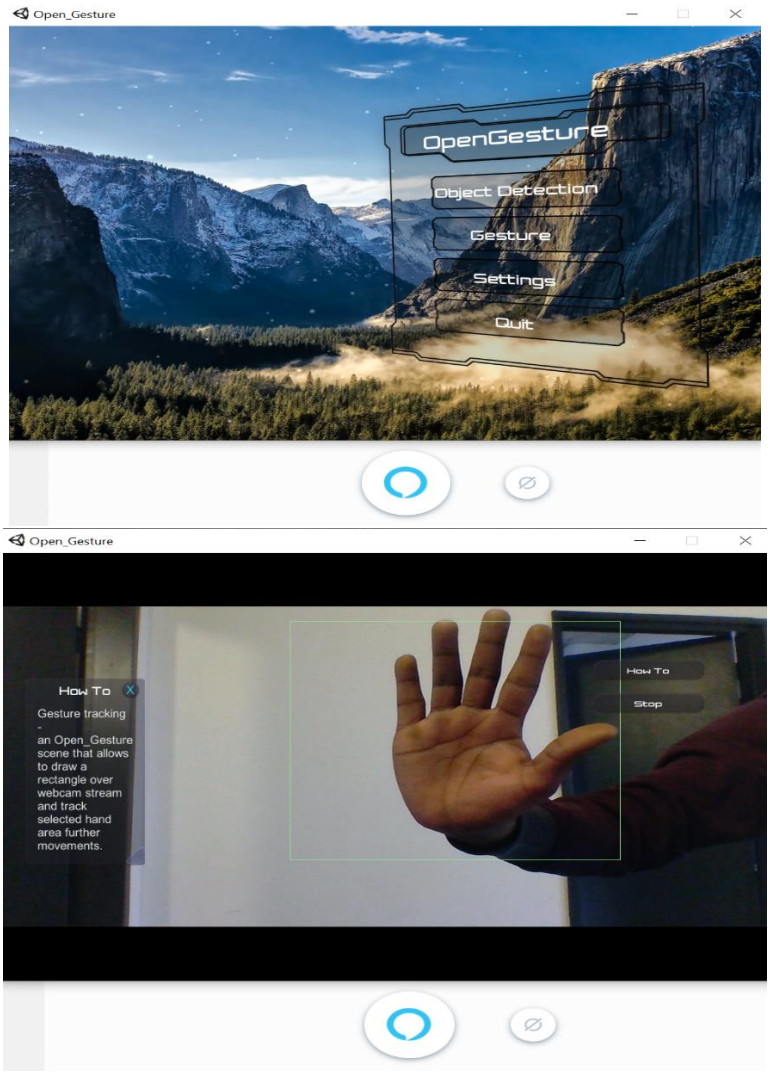
The following code snippet shows how this is implemented, how the slot values are retrieved using the GetSlotString() function, and how these slot values are used to get the required functionality of the OpenGesture Skill. Full code maybe found here: https://github.com/TebogoNakampe/OpenGesture-Skill/blob/master/OpenGestureSkill/IntentHandling.cs

```
#region OpenGesture Code
        private string OpenGesture(string target, string topic, string requestbody)
        {
            object response = irc.ProgressDialog(requestbody);
            if (response == null)
            {
                string appLocationValue = irc.GetSlotString(requestbody, "appLocation");
                string appTypeValue = irc.GetSlotString(requestbody, "appType");
                string searchQueryValue = irc.GetSlotString(requestbody, "searchQuery");
                string dateValue = irc.GetSlotString(requestbody, "date");
                OnLogMessage(string.Format("OpenGestureIntent-> appLocationValue: {0}, appTypeValue: {1}, dateValue: {2},
searchQueryValue: {3}", appLocationValue, appTypeValue, dateValue, searchQueryValue), null, true);
                PerformOpenGesture(appLocationValue, appTypeValue, dateValue, searchQueryValue);
                response = irc.BuildResponse(irc.GetSessionAttributes(requestbody),
                               irc.BuildSpeechletResponse(topic, "Running OpenGesture.", "", true));
            }
            return (new JavaScriptSerializer()).Serialize(response);
        }

        public void PerformOpenGesture(string appLocationValue, string appTypeValue, string dateValue, string searchQueryValue = "")
        {
            appLocationValue = GetFileLocation(appLocationValue);
            StringBuilder queryBuilder = new StringBuilder();
            if (!string.IsNullOrEmpty(searchQueryValue))
            {
                queryBuilder.Append("search-ms:displayname=Searching for query%20" + searchQueryValue + "%20in%20" +
appLocationValue);
                if (!fileTypeValue.Equals("everything"))
                    queryBuilder.Append("&crumb=System.Generic.String%3A" + searchQueryValue + "%20kind%3A%3D" + appTypeValue);
                else
                    queryBuilder.Append("&crumb=System.Generic.String%3A" + searchQueryValue);
            }
            else
            {
                queryBuilder.Append("search-ms:displayname=Search%20Results%20in%20" + appLocationValue);
                if (!fileTypeValue.Equals("everything"))
                    queryBuilder.Append("&crumb=kind%3A%3D" + appTypeValue);
            }
            if (!string.IsNullOrEmpty(dateValue))
                queryBuilder.Append("&crumb=datemodified%3A" + dateValue);
            queryBuilder.Append("&crumb=location:" + appLocationValue);
            Process.Start(queryBuilder.ToString());
        }

        /// <summary>
        /// Get OpenGesture Windows App locaiton
        /// </summary>
        /// <param name="appLocationValue"></param>
        /// <returns></returns>
        private string GetAppLocation(string appLocationValue)
        {
            switch (appLocationValue.ToLower())
            {
                case "documents":
                    appLocationValue = Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments);
                    break;
                case "desktop":
                    appLocationValue = Environment.GetFolderPath(Environment.SpecialFolder.Desktop);
                    break;
```

```
                case "downloads":
                    appLocationValue = System.Convert.ToString(Microsoft.Win32.Registry.GetValue(
                                        @"HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders"
                                        , "{374DE290-123F-4565-9164-39C4925E467B}", String.Empty));
                    break;
                case "user":
                    appLocationValue = Environment.GetFolderPath(Environment.SpecialFolder.UserProfile);
                    break;
                case "music":
                    appLocationValue = Environment.GetFolderPath(Environment.SpecialFolder.MyMusic);
                    break;
                case "pictures":
                    appLocationValue = Environment.GetFolderPath(Environment.SpecialFolder.MyPictures);
                    break;
                case "recent":
                    appLocationValue = Environment.GetFolderPath(Environment.SpecialFolder.Recent);
                    break;
                case "everywhere":
                    appLocationValue = Path.GetPathRoot(Environment.GetFolderPath(Environment.SpecialFolder.System));
                    break;
                default:
                    appLocationValue = Environment.GetFolderPath(Environment.SpecialFolder.UserProfile);
                    break;
            }
            return appLocationValue;
        }
        #endregion
```

Screen Shot

The following screenshot represents results for the OpenGesture Skill:



Conclusion

This document helps users understand OpenGesture Skill and PC Skill architecture and implementation flow using the PC Skills Framework Library (PCSF).