2019

GameHub Skill

PC SKILL FRAMEWORK

THABO KOEE & MOLOTI NAKAMPE

Table of Contents

Introduction

Purpose

I had the chance to build an Alexa for PC skill and I was pleasantly surprised by the quality of the voice interaction. Over the past 30 days I started exploring how to build skills for Alexa on PC and, at the beginning, I found myself outside my comfort zone. From a technical perspective, the PC Skills Framework (PCSF) library is an excellent developer tool that lets Alexa skill writers create a PC Skill—a hybrid cloud/PC model where vocabulary, utterance, and intent models are run by the Alexa Voice Service.

This document provides developer guidance for the GameHub Skill using Alexa* with the PC Skills Framework (PCSF) library.

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

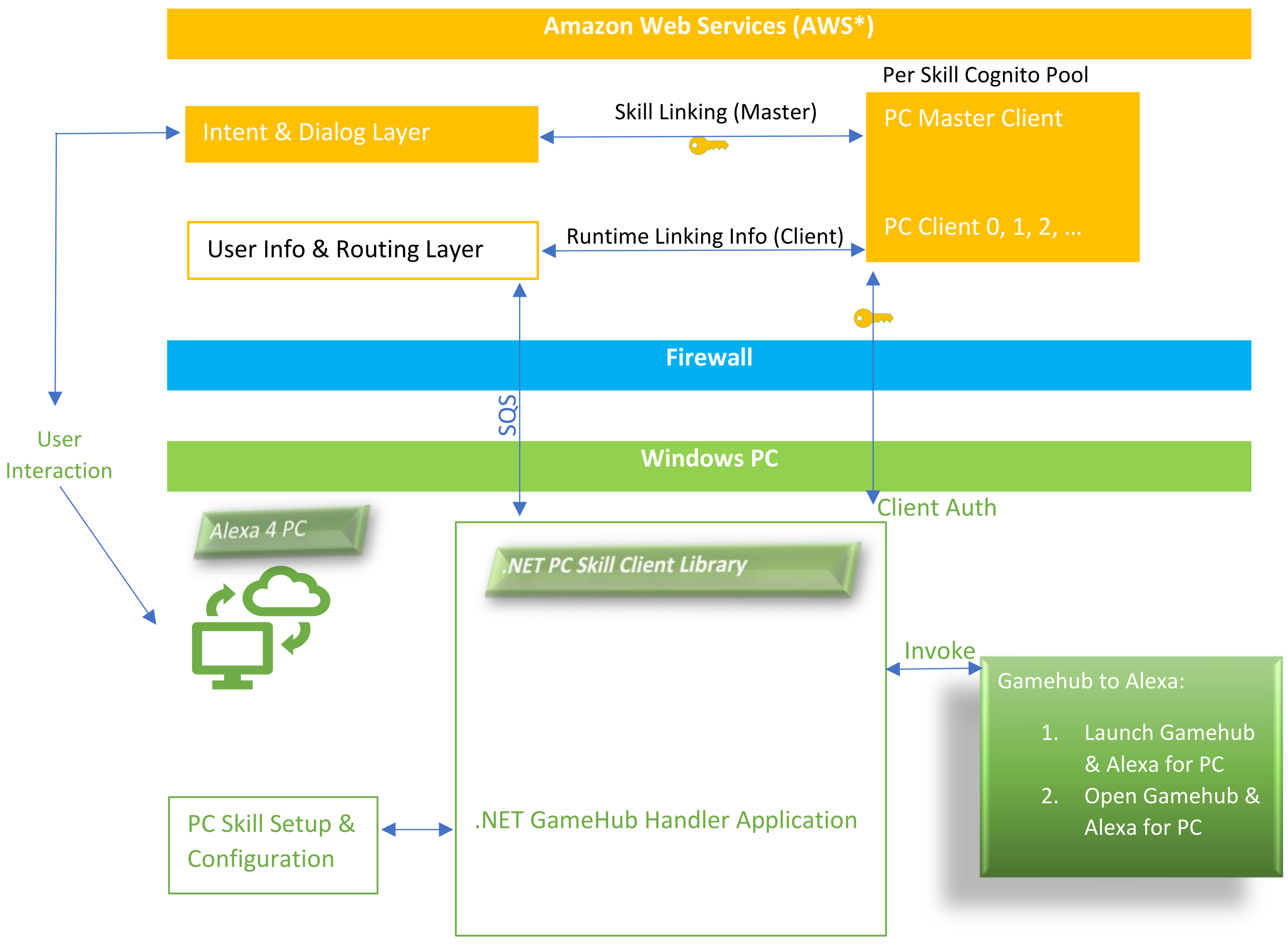
Acronyms

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

- GameHub App: https://drive.google.com/open?id=16A4Renp09Zfpv4aWXwKLLRsgL2yfy96C
- 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


Amazon Web Services (AWS*)
Per Skill Cognito Pool
Intent & Dialog Layer
Skill Linking (Master)
PC Master Client
User Info & Routing Layer
Runtime Linking Info (Client)
PC Client 0, 1, 2, …
Firewall
SQS
User Interaction
Windows PC
Client Auth
Alexa 4 PC
.NET PC Skill Client Library
Invoke
Gamehub to Alexa:
1. Launch Gamehub & Alexa for PC
2. Open Gamehub & Alexa for PC
PC Skill Setup & Configuration
.NET GameHub Handler Application

Operational Flow

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

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

2. User-invoked GameHub 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 GameHub handler uses the intent to affect the PC in some manner, or just creates a contextual response.

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

8. The name of the application/filename received from the request body is used to launch Gamehub application.

9. Based on the command given (Up and Down/left to right/Four by Four), the new position of the Gamehub and Alexa for PC windows is calculated, and the positioning information is applied to the selected windows.

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 GameHub.

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

GameHub

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 “GameHubIntent”

Intent Name: GameHubIntent

GameHub Sample Utterances

Create the sample utterances shown below where {app}=Gamehub, {appA}=Gamehub/Alexa4PC, {appB}= Gamehub/Alexa4PC, and {action}=leftandright.

```
            "run {action}",
            "run {appA} and {appB} {action}",
            "run {app} {action}",
            "execute {action}",
            "execute {appA} and {appB} {action}",
            "execute {app} {action}",
            "launch {action}",
            "launch {appA} and {appB} {action}",
            "launch {app} {action}",
            "open {action}",
            "open {appA} and {appB} {action}",
            "open {app} {action}"
```

Intent Slots

Name: appA Type: AMAZON.SoftwareApplication
Name: appB Type: AMAZON.SoftwareApplication
Name: app Type: AMAZON.SoftwareApplication

Name: action Type: LIST_OF_WIN_ACTIONS
LIST_OF_WIN_ACTIONS is a custom slot and is defined as follows

```
{
          "name": "LIST_OF_WIN_ACTIONS",
          "values": [
            { "name": { "value": "four by four" } },
            { "name": { "value": "up and down" } },
            { "name": { "value": "left and rights" } }
          ]
        }
```

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>>()
            {
                .
                .
                { "GameHubIntent",  ((thisObj, target, topic, requestbody)
                thisObj.Gamehub(target,topic,requestbody))}
                .
                .
            };
```

GameHub 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 GameHub.
https://github.com/TechTouchABI/GameHub/blob/master/GameHub/IntentHandling.cs

```
#region Gamehub code

public enum GamehubActions
{
DoFourbyFour,
DoUpAndDown,
DoLeftAndRights
}

private string Gamehub(string target, string topic, string requestbody)
{
object response = irc.ProgressDialog(requestbody);
if (response == null)
{
string speechText = "";
string action = irc.GetSlotString(requestbody, "action");
string app = irc.GetSlotString(requestbody, "app");
string appA = irc.GetSlotString(requestbody, "appA");
string appB = irc.GetSlotString(requestbody, "appB");

#if DEBUG
irc.LogMessage(string.Format("Arrange - action = {0}, app={1}, appA={2}, appB={3}",
action, app, appA, appB));
#endif

if (string.Equals("four by four", action, StringComparison.OrdinalIgnoreCase))
{
speechText = PerformGamehubActions(app, appA, appB, GamehubActions.DoFourbyFour);
}
else if (string.Equals("up and down", action, StringComparison.OrdinalIgnoreCase))
{
speechText = PerformGamehubActions(app, appA, appB, GamehubActions.DoUpAndDown);
}
else if (string.Equals("left and rights", action,
StringComparison.OrdinalIgnoreCase) ||
string.Equals("left and rights", action, StringComparison.OrdinalIgnoreCase))
{
speechText = PerformGamehubActions(app, appA, appB, GamehubActions.DoLeftAndRights);
}

response = irc.BuildResponse(irc.GetSessionAttributes(requestbody),
irc.BuildSpeechletResponse(topic, speechText, "", false));
}
return (new JavaScriptSerializer()).Serialize(response);
}

private string PerformGamehubActions(string application, string appA, string appB,
GamehubActions actionName)
{
int result = 0;
string speechText = "";
if (!string.IsNullOrEmpty(appA) && !string.IsNullOrEmpty(appB))
{
string[] appNames = { appA, appB };
result = winCTL.ArrangeWindows(actionName, appNames);
if (result > 0)
speechText = string.Format(" " + result + " " + appA + " and " + appB + "
windows.");
else
speechText = string.Format("I'm sorry.  Please make sure Gamehub application is on
system ");
}
else
{
string applicationName = "";
if (!string.IsNullOrEmpty(application))
applicationName = application;
else if (!string.IsNullOrEmpty(appA))
applicationName = appA;
else if (!string.IsNullOrEmpty(appB))
applicationName = appB;

result = winCTL.ArrangeWindows(actionName, applicationName);
if (result > 0)
speechText = string.Format("Executed " + result + " " + applicationName + "
windows.");
else
speechText = string.Format("I'm sorry.  Please make sure Gamehub application is on
system ");
}
return speechText;
}
#endregion
}
}
```

ControlAppWindows Class

The main functions from the ControlAppWindows class include:

ArrangeWindows ()

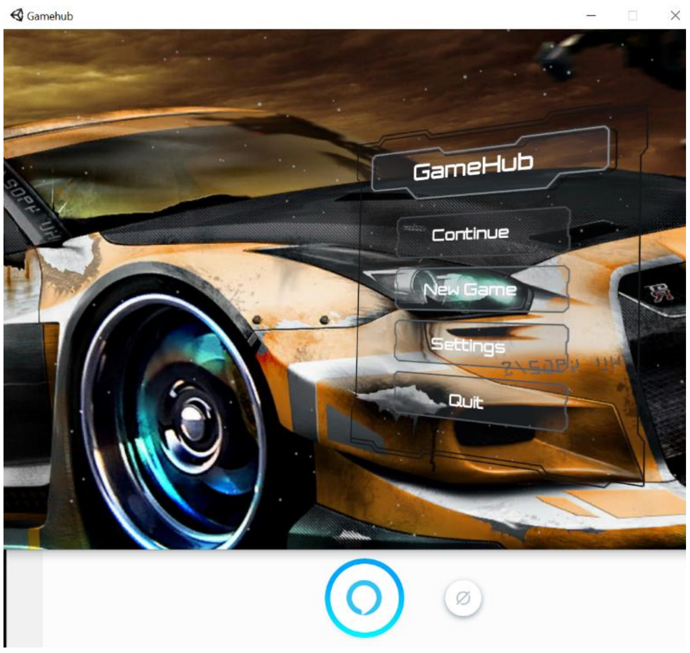
1. Four by Four: Finds the Gamehub and Alexa for PC windows by the input parameter app and arranges its top four instances into four sides. If appName is empty, it arranges the top four applications into four sides. It returns the number of windows arranged in four sides.

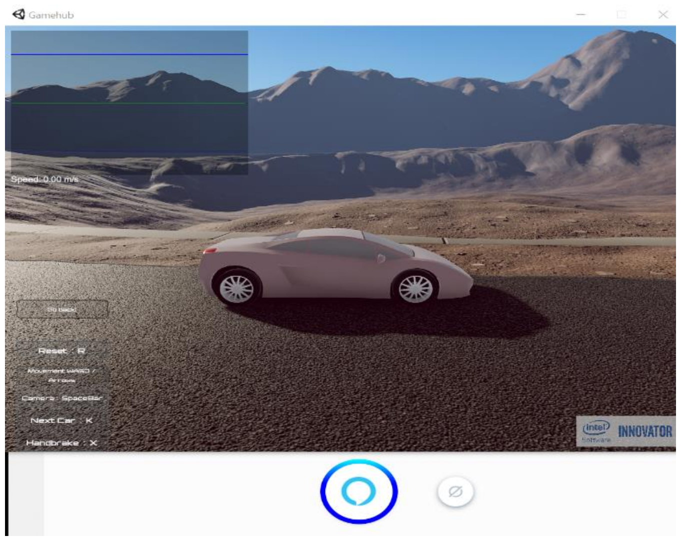
2. Left To Right: Finds the Gamehub and Alexa for PC windows by input parameter app and arranges its top four instances side by side. If app is empty, it arranges the top four application side by side. It returns the number of windows arranged left and right.

3.Up And Down: Finds the Gamehub and Alexa for PC windows by input parameter app and arranges its top four instances Up and Down. If app is empty, it arranges the top four applications Up and Down. It finds the windows specified by the input parameter appA and appB and arranges the top two instances per application up and down. It returns two of windows of Gamehub Application and Alexa* for PC arranged up and down.

Screenshot

The following screen shots illustrate Gamehub App and Alexa* for PC organized on a Desktop for the user-provided query such as "Up and down"





Conclusion

This guide helps user/skill writers to comprehend Gamehub PC Skill Architecture and implementation flow using PC Skills Framework Library (PCSF).