Native Speech Recognition Unity plugin

Version 1.0

Created by Sergey Okhotnikov

https://okhotnikov.net/

Native Speech Recognition Unity plugin

1. Introduction and Overview 3
2. Package content 3
3. Quick start demo 4
4. Integration Guide 8
5. Requesting permissions 10
6. Handling errors 11
7. Supported languages iOS 12
8. Supported languages Android 14

1. Introduction and Overview

Native Speech Recognition plugin enables you, the Unity developer, to get benefits from native speech recognition functions of iOS and Android platforms.

This solution uses speech recognition classes from iOS (SFSpeechRecognizer) and Android (android.speech.SpeechRecognizer).

Speech recognition is precise and offline for latest iOS and Android versions.

This solution handles user permissions requests for you.

If you have any questions, feedback or having issues, please contact me directly at segey@okhotnikov.net. I will respond to you as quickly as possible.

2. Package content

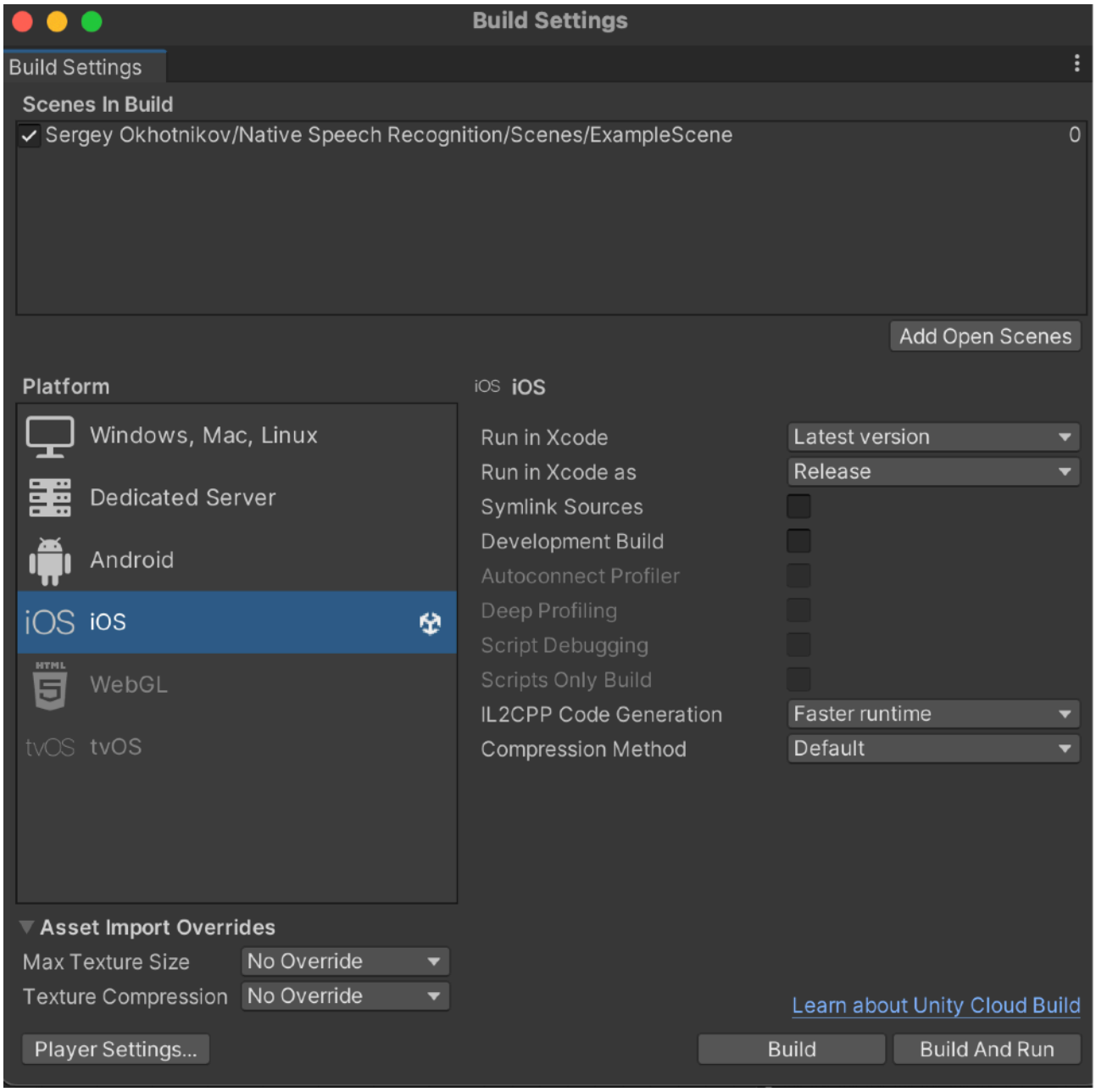
The package contains c# class that communicates with iOS and Android libraries as well as demo scene.

Module	Description
Native Speech Recognition/ Scenes/ExampleScene.unity	Demo scene. You should include it into your iOS or Android build to quick start project demo.
Native Speech Recognition/ Scripts/ SpeechRecognizerExample.cs	This demo script provides interaction between demo scene and Speech Recognition plugin. You should use this script as an example for you own plugin integration.
Native Speech Recognition/ Plugins/SpeechRecognizer/iOS/ SpeechRecognizerIOs.framework	iOS library. You shouldn't edit files in this folder.
Native Speech Recognition/ Plugins/SpeechRecognizer/ Android	Android libraries folder. That folder contains AndroidManifest and java library archive.
Native Speech Recognition/ Plugins/SpeechRecognizer/ SpeechRecognizer.cs	This file contains SpeechRecognizer class that works as a bridge between native and Unity code.
Native Speech Recognition/ Plugins/SpeechRecognizer/Editor/ Postprocessor.cs	This files handles Info.plist file modification upon iOS build. It adds required permission records. You could edit this file to specify your reasons for asking microphone and speech recognition permissions.

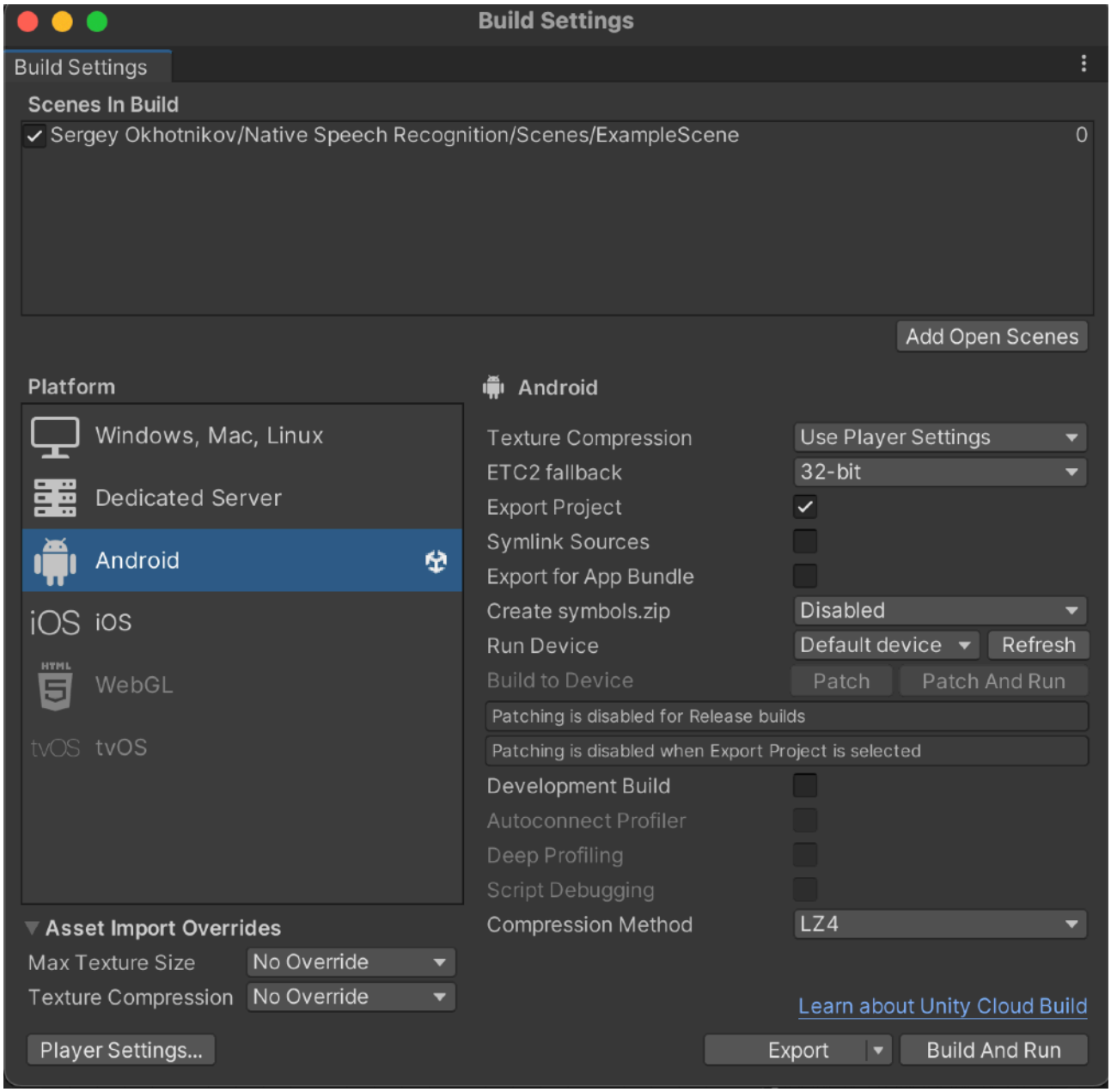
3. Quick start demo

Native Speech Recognition plugin utilizes iOS and Android classes. That's why running project in Unity editor does nothing. You should create iOS or Android build to run test scene.

As a first step you should include ExamplScene into your build.

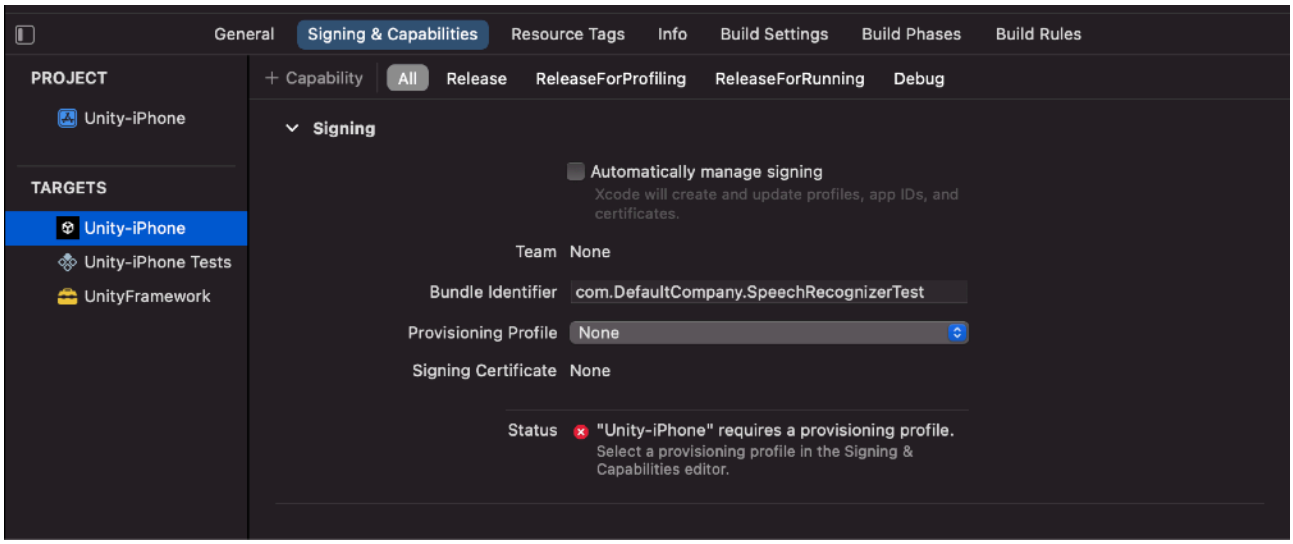


Step 1 iOS: Build a project with ExampleScene.



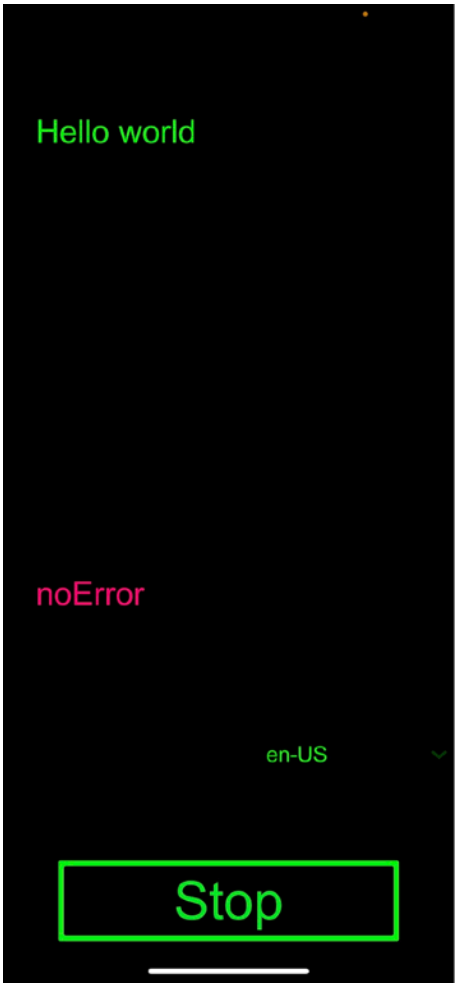
Step 1 Android: Build a project with ExampleScene.

For iOS project additional step required. You should sign your target in Xcode. Select Unity-iPhone target then Signing & Capabilities tab. Choose your project team Provisioning Profile.



Step 2 iOS: Sign the target

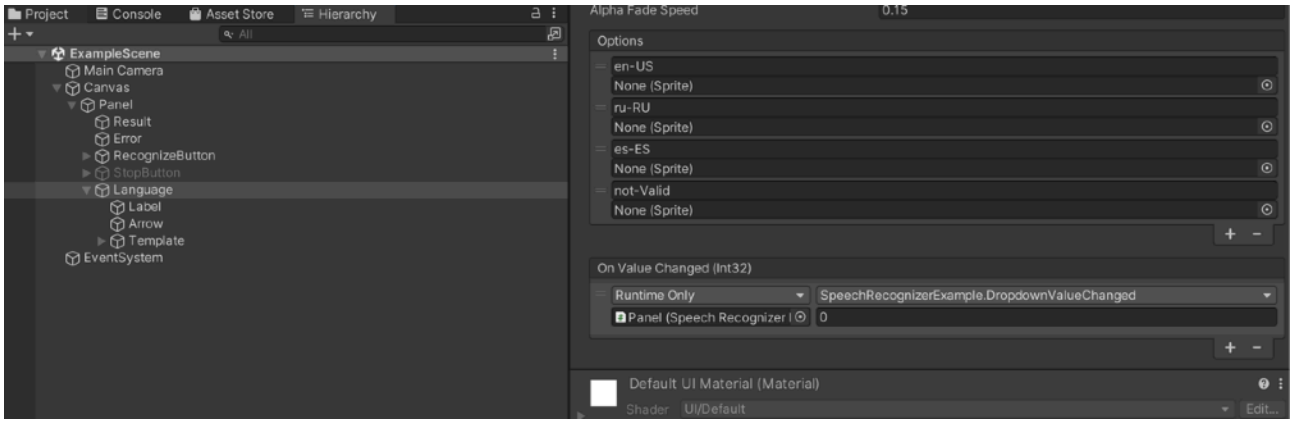
Now your could connect your phone and run the project.



Final step: Run the project

In demo application You could set speech recognition language, start and then stop speech recognition. Recognition results are displayed at the top of the screen. Error messages are displayed in the middle of the screen.

If you need to test some specific language you should add it to the language options list.



Additional step: Add languages

- Open ExampleScene
- Find Language GameObject in the Hierarchy>Canvas>Panel>Language
- Find Dropdown script
- Edit Options list.

You could find full list of supported languages at the end of this tutorial.

4. Integration Guide

The integration process of Native Speech Recognition plugin is easy:

Step 2: Define callbacks for receiving results and error messages

You will need 2 methods for handling callbacks from iOS and Android libraries. One for handling recognition results.

```
private void OnResult(string msg)
{
    ResultText.text = msg;
}
```

Second for handling error messages:

```
private void OnError(string msg)
{
    ErrorText.text = msg;
    if (msg != noError)
    {
        RecognitionNotStarted();
    }
}
```

You should implement custom logic for both methods.

Step 2: Create instance of SpeechRecognizer class

Your only need one instance of SpeechRecognizer at a time so it follows the singleton pattern and has private constructor. To create an instance you need to call static Create method.

```
SpeechRecognizer.Create(en-US,OnResult,OnError);
```

Parameters:

1. string language - Speech recognition language.

2. Action<string> result - callback for receiving recognition results.
3. Action<string> error - callback for receiving platform error messages.

Step 3: Start speech recognition

All you need is to call SpeechRecognizer object's method Start.

```
SpeechRecognizer recognizer = SpeechRecognizer.Create(_language,OnResult,OnError);

recognizer.Start();
```

If everything went well you will receive noError message into your error handling callback.

Result handling callback will start receiving recognition results. It could be called several times. Each time a result will be appended with new portion of a text.

Step 4: Pause speech recognition
Just call SpeechRecognizer object's method Stop.

```
recognizer.Stop();
```

If everything went well you will receive noError message into your error handling callback.

You could reuse SpeechRecognizer object and call Start again when it's required.

Step 5: Stop speech recognition
Don't forget to dispose SpeechRecognizer object by setting variable to null.

```
recognizer.Stop();

recognizer = null;
```

5. Requesting permissions

Native Speech Recognition plugin handles permission requests for your. But there is something that you might want to change.

Step : Change reasons

You could change text that your iOS users will see when your application asks for permissions. To do that find this fragment in Native Speech Recognition/Plugins/SpeechRecognizer/Editor/Postprocessor.cs file:

```
var spKey ="NSSpeechRecognitionUsageDescription";
rootDict.SetString(spKey,The application uses voice input to interact with a user. It requires speech recognition usage for this purposes.);
```

```
var microKey = "NSMicrophoneUsageDescription";
rootDict.SetString(microKey,The application uses voice input to interact with a user. It requires microphone access for this purposes.);
```

Edit text above to mirror your true reasons.

6. Handling errors

You should handle messages received in your error handling method:

1. **noError** - Do nothing. Everything is ok.
2. **unsupportedLanguage** - Your device doesn't support provided language string. Set different language string.
3. **insufficientPermissions** - A user didn't grant permission. You should display a message that this user should set required permission via Settings on his device.
4. **networkError** - Older versions of Android could require internet access for speech recognition. You should notify your user about that.
5. **toManyRequests, recognizerBusy** - Android device gets too many speech recognition request. You shouldn't start and stop speech recognition too frequently to avoid this errors.
6. **failToStartAudioEngine, failToSetAVRecordCategory, failToStartAVSession, failToStopAVSession** - iOS device getting errors switching audio sessions. You shouldn't start and stop speech recognition too frequently to avoid this errors.
7. **audioError, clientError** - Android device audio errors.
8. **notAvailableForTheMoment** - Android device in the middle of downloading speech recognition language.

7. Supported languages iOS

1. nl-NL,
2. es-MX,
3. fr-FR,
4. zh-TW,
5. it-IT,
6. vi-VN,
7. fr-CH,
8. es-CL,
9. en-ZA,
10. ko-KR,
11. ca-ES,
12. ro-RO,
13. en-PH,
14. es-419,
15. en-CA,
16. en-SG,
17. en-IN,
18. en-NZ,
19. it-CH,
20. fr-CA,
21. hi-IN,
22. da-DK,
23. de-AT,
24. pt-BR,
25. yue-CN,
26. zh-CN,
27. sv-SE,
28. hi-IN-translit,
29. es-ES,
30. ar-SA,
31. hu-HU,
32. fr-BE,
33. en-GB,
34. ja-JP,
35. zh-HK,
36. fi-FI,
37. tr-TR,
38. nb-NO,
39. en-ID,

40. en-SA,
41. pl-PL,
42. ms-MY,
43. cs-CZ,
44. el-GR,
45. id-ID,
46. hr-HR,
47. en-AE,
48. he-IL,
49. ru-RU,
50. wuu-CN,
51. de-DE,
52. de-CH,
53. en-AU,
54. nl-BE,
55. th-TH,
56. pt-PT,
57. sk-SK,
58. en-US,
59. en-IE,
60. es-CO,
61. hi-Latn,
62. uk-UA,
63. es-US

8. Supported languages Android

1. af-ZA
2. az-AZ
3. id-ID
4. ms-MY
5. jv-ID
6. su-ID
7. ca-ES
8. cs-CZ
9. da-DK
10. de-DE
11. de-AT
12. et-EE
13. en-AU
14. en-CA
15. en-001
16. en-GH
17. en-IN
18. en-IE
19. en-KE
20. en-NZ
21. en-NG
22. en-PH
23. en-SG
24. en-ZA
25. en-TZ
26. en-GB
27. en-US
28. es-AR
29. es-BO
30. es-CL
31. es-CO
32. es-CR
33. es-EC
34. es-US
35. es-SV
36. es-ES
37. es-GT
38. es-HN
39. es-MX

40. es-NI
41. es-PA
42. es-PY
43. es-PE
44. es-PR
45. es-DO
46. es-UY
47. es-VE
48. eu-ES
49. fil-PH
50. fr-FR
51. fr-CA
52. gl-ES
53. hr-HR
54. zu-ZA
55. is-IS
56. it-IT
57. sw
58. sw-TZ
59. lv-LV
60. lt-LT
61. hu-HU
62. nl-NL
63. nb-NO
64. uz-UZ
65. pl-PL
66. pt-BR
67. pt-PT
68. ro-RO
69. sl-SI
70. sk-SK
71. fi-FI
72. sv-SE
73. vi-VN
74. tr-TR
75. el-GR
76. bg-BG
77. ru-RU
78. sr-RS
79. uk-UA

80. ka-GE
81. hy-AM
82. he-IL
83. ar-IL
84. ar-JO
85. ar-AE
86. ar-BH
87. ar-DZ
88. ar-SA
89. ar-KW
90. ar-MA
91. ar-TN
92. ar-OM
93. ar-PS
94. ar-QA
95. ar-LB
96. ar-EG
97. fa-IR
98. ur-PK
99. ur-IN
100. am-ET
101. hi-IN
102. ta-IN
103. ta-LK
104. ta-SG
105. ta-MY
106. bn-BD
107. bn-IN
108. km-KH
109. kn-IN
110. mr-IN
111. gu-IN
112. si-LK
113. te-IN
114. ml-IN
115. ne-NP
116. lo-LA
117. th-TH
118. my-MM
119. ko-KR

120. cmn-Hans-CN
121. cmn-Hans-HK
122. cmn-Hant-TW
123. yue-Hant-HK
124. ja-JP
125. en-ID
126. en-TH