Implementing GPII/Cloud4All Support for Mobile Accessibility for Android

Ferran Gállego

Code Factory S.L., Terrassa, Spain
ferran.gallego@codefactory.es

Abstract. Mobile Accessibility for Android is a combination of a suite of accessible apps and a screen reader which provide accessibility on Android devices for blind and visually impaired users. Main functionality of Android devices is made available to the user through Mobile Accessibility's voice and Braille based UI. This paper describes the process of integrating this commercial product with GPII/Cloud4All online architecture, providing auto-configuration based on user's online profile and NFC user identification.

Keywords: Access to mobile interaction, Cloud4All, GPII.

1 Introduction

Mobile Accessibility (MA) for Android is a suite of 12 accessible apps with a simplified User Interface (UI) that has been specially designed for blind and visually impaired users [1], it also implements an Android Accessibility Service that offers screen reading functionality, allowing the blind user to interact with the native Android UI similarly as the native Android Screen Reader (TalkBack) does.

Mobile Accessibility Input/Output interface is voice based. The product also offers Braille support for different grades and languages, compatible with the most popular Bluetooth Braille devices.

Assistive Technologies like Mobile Accessibility are highly configurable, offering a number of settings that allow the user to adjust the interface behavior based on his preferences, experience and other factors. Adjusting settings based on the needs and preferences of a given user is not an inmediate task, it requires the user to navigate several menus and configuration options related to voice settings, text processing, Braille input/output preferences, screen reader options and more. Moreover, modern mobile devices are 100% touch based, so this configuration process is still more complicated because adapted touch interfaces for blind users are not as productive as keyboard based UIs.

Auto configuration functionality is definitely a huge step in order to allow users to set up new phones, share devices with other members of the family that have different needs and lots of situations where there's no time enough to manually configure the tool to fit the needs and preferences of the user.

C. Stephanidis and M. Antona (Eds.): UAHCI/HCII 2014, Part II, LNCS 8514, pp. 50–57, 2014.

Moreover, since the Near Field Communication (NFC) technology has been widely adopted by phone manufacturers, this features offer the opportunity to develop user identification by touching the device with easily wearable NFC tags integrated on cards, rings, etc.

2 Mobile Accessibility Role in Cloud4All Project

Cloud4All/GPII offer a cloud infrastructure where needs and preferences of users are stored. When a user is identified in a Cloud4All compatible technology, it requests the cloud infrastructure for the preferences of the new person and applies it automatically.

This integration can be done at platform level, where the Operating System (OS) is responsible of the interaction with Cloud4All/GPII infrastructure, providing APIs that would allow third party developers for this platform to take benefit of auto-configuration features supported by the OS.

When Cloud4All/GPII integration is not present at platform level, individual applications could individually integrate Cloud4All/GPII, implementing the communication with online infrastructure and offering auto-configuration as a feature implemented inside the app. This is the case covered by Mobile Accessibility in Cloud4All project (A304.2: Auto-configuration of assistive solutions in mobile phones), so the implementation of Cloud4All/GPII for this solution doesn't assume any platform level integration on Android. Other parts of the project take care of the integration of Cloud4All/GPII at OS level.

3 Mobile Accessibility Settings and Cloud4All/GPII Common Terms

Mobile Accessibility offers a high number of settings that allow the user to adjust the behavior of the app on different aspects: Text-To-Speech (TTS), text input preferences, details about interaction with external Bluetooth Braille devices, screen reader configuration, call management options and more.

A subset of configuration options available on Mobile Accessibility has been selected in order to implement auto-configuration features offered by Cloud4All/GPII:

- *Access_commonprefs_speechrate*: TTS speech rate. Type: Numeric values from 0 to 10.
- *Access_commonprefs_speechpitch*: voice pitch. Type: Numeric values from 0 to 10.
- *Settings_phone_usesystemcallscreens*: this setting allows the user to determine how to manage incoming calls, this can be done through a specific call screen offered by Mobile Accessibility or the default Android call screen. Type: Boolean.

- *Access_commonprefs_turnofflistnumbering*: allows the user to activate/deactivate list numbering information (current item number and total number of items) when navigating lists. Type: Boolean.
- *Access_commonprefs_editingspeakdelchar*: through this setting the user can decide if he wants to get voice output announcing the deleted character while typing text. Type: Boolean.
- *Access_commonprefs_srspeaksystemnotifications*: this setting determines if system notifications should be spoken aloud by the TTS when they arrive. Type: Boolean.
- *Access_commonprefs_numberprocessing*: indicates how long numbers are spoken by the TTS. Type: Numeric from 0 to 3. Values can be: none(0): read the entire number, single(1): read the number by digits, pairs(2): read the number by pairs of digits, triplets(3): read the number by triplets of digits.
- *Access_commonprefs_editingkeyboardecho*: keyboard echo while typing text. Type: numeric from 0 to 3. Values can be: none(0), characters(1), words(2), characters and words (3).
- *Access_commonprefs_editingsecretmode*: allows to determine what keyboard echo to get while typing text in password fields. Type: numeric from 0 to 2. Values can be: say star(0: the TTS says "star" for each character), speak characters (1: the TTS speaks the introduced character -the field will still look like a password field visually, this only affects text output-), silent (2: no speech output while typing on password fields).
- *Settings_phone_usevolumekeysforcalls*: for incoming calls and also while in a call, volume keys can keep its default behavior and allow to adjust the volume or, if the user prefers, it can be used to accept the call (volume up), reject the call or hang up the current one (volume down). This setting allows the user to choose how he wants to use volume keys in this scenario. Type: Boolean.
- *Access_commonprefs_spellphonetically*: how the characters of a word will be spelled. Type: Boolean. Values can be true: character names: a, b, c, d...; false: use NATO phonetic alphabet: Alpha, Bravo, Charlie, Delta...
- *Access_commonprefs_punctuation*: indicates how many punctuation symbols will be spoken when reading text. This is very common between different screen reader solutions. Punctuation characters will take effect on the spoken text, so commas, dots etcetera will produce pauses and so. Anyway, some users will prefer to hear the name of the punctuation symbol when it appears. This is commonly needed for non-advanced users or when they need to read carefully. Type: Numeric from 0 to 3. Possible values are: none(0: no punctuation symbols are spoken), some(1: only some non common punctuation symbols are spoken), most (2: most of punctuation symbols are spoken but not the most common ones such as dot, comma, etc.), all (3: all punctuation symbols are spoken).
- *Access_commonprefs_capitalization*: allows the user to determine whether or not capitalization should be announced when reading text by characters. Type: Boolean. Values can be true (capitalization should be spoken), false (ignore capitalization, just speak character names regardless its upper or lower case condition).

4 Common Terms

Mobile Accessibility application-specific terms (settings described in point 3) have correspondences with Cloud4All/GPII common terms, which define stable definitions of settings or preferences that would apply across many applications or devices. For example, some common terms that match Mobile Accessibility specific terms are:

- *speechRate*: defined as number of words per minute.
- *pitch*: floating point value from 0.0 to 1.0.
- *keyEcho*: type of speech output to get when typing characters. Type: Boolean (true: provide TTS output on key presses, false: no TTS output for characters).
- *wordEcho*: type of speech output to provide after completing each word while typing text. Type: Boolean (true: read the entire word after typing it, false: no TTS output for words).
- *announceCapitals*: same as *access_commonprefs_capitalization* Mobile Accessibility specific setting.
- *punctuationVerbosity*: same as *access_commonprefs_punctuation* Mobile Accessibility specific setting.

5 Transformations

Needs and preferences sets stored in the online Cloud4All/GPII architecture are defined using common terms which will be applied across different technologies and solutions. In order to translate this common terms in application specific terms we need to define transformation rules.

Several transformation functions are available for this puposes. They are documented in the entry Architecture - Available transformation functions from Cloud4All/GPII Wiki site [2].

To do this, we need to add a solution entry for Mobile Accessibility so the architecture knows about our application. This entry will also describe the transformations that need to be perfomed by the online architecture, so the preferences sets sent to Mobile Accessibility are defined in app-specific terms.

Solution entries with transformation definitions are stored in the Cloud4All/GPII online architecture and they are implemented in JSON language. Transformations for Mobile Accessibility's specific terms are defined with the following structure:

```
<MA_specific_term>:{
   "transform": {
      "type": <type_of_transformation>,
      <transformation_parameter_1>: <value_1>,
      ...
      <transformation_parameter_n>: <value_n>
   }
}
```

The following transformation examples show how to transform the common terms described in point 4 to convert it in the corresponding Mobile Accessibility application-specific terms.

5.1 Speech Rate

Speech rate common term defines it as words per minute, while Mobile Accessibility specific terms needs values from 0 to 10. A good approach for this transformation is defining

```
  MA_speech_rate = speechRate_common_term / 40
This transformation can be implemented as follows:
"access_commonprefs_speechrate": {
       "transform": {
          "type": "fluid.transforms.binaryOp",
          "leftPath": "display.screenReader.speechRate",
          "operator": "/",
          "right": 40
       }
    }
```

Here the transformation is of type flu-id.transforms.binaryOp, that means a simple binary operation that needs 3 parameters to define the left operand (speechRate common term in this case), operator and right operand for the transformation: leftPath, operator, rightPath.

5.2 Speech Pitch

Same as for the speech rate, speech pitch common term can be converted to Mobile Accessibility's specific term for pitch with a simple arithmetic operation

```
  MA_speech_pitch = 10 * pitch_common_term
```

This transformation is a binary operation same as 5.1 and can be implemented as follows:

```
"access_commonprefs_speechpitch": {
       "transform": {
          "type": "fluid.transforms.binaryOp",
          "leftPath":
"diplay.textReadingHighlight.pitch",
          "operator": "*",
          "right": 10
       }
    }
```

5.3 Keyboard Echo

This case needs more attention. Mobile Accessibility specific term for this is defined as a numeric value between 0 and 3, where each value means

- 0: no keyboard echo.
- 1: character echo.
- 2: words echo.
- 3: characters and words echo.

There's no exact common term correspondence for this definition. Instead, there are 2 different common terms that combine this user preferences:

- keyEcho: boolean that defines if the user wants characters echo or not.
- wordEcho: boolean that defines if the user wants words echo or not.

So the transformation here must combine both keyEcho and wordEcho common terms in a single Mobile Accessibility specific terms.

This can be implemented by nesting some conditional transformations:

```
"access_commonprefs_editingkeyboardecho": {
   "transform": {
      "type": "fluid.transforms.condition",
      "conditionPath":
"display.screenReader.-provisional-keyEcho",
      "true": {
         "transform": {
            "type": "flu-id.transforms.condition",
            "conditionPath":
"display.screenReader.-provisional-wordEcho",
            "true": "3",
            "false": "1"
          }
      },
      "false": {
         "transform": {
            "type": "flu-id.transforms.condition",
            "conditionPath":
"display.screenReader.-provisional-wordEcho",
            "true": "2",
            "false": "0"
         }
      }
   }
}
```

Conditional transformations accept 3 parameters: conditionPath (path the boolean value to be evaluated, in this case it'll be keyEcho and wordEcho common terms), true (value to return when the condition is true), false (value to return when the condition is false).

The combined Mobile Accessibility specific setting is obtained by first checking wordEcho common term, then we evaluate keyEcho common term for each possible value of wordEcho, returning the combined value that Mobile Accessibility needs in each case.

The previous JSON fragment represents what in pseudo-code would be

```
if keyEcho_common_term then
    if wordEcho_common_term then
       MA_keyboard_echo_setting = 3
    else
       MA_keyboard_echo_setting = 1
else
    if  wordEcho_common_term then
       MA_keyboard_echo_setting = 2
    else
       MA_keyboard_echo_setting = 0
    end if
end if
```

5.4 Punctuation Verbosity and Capitalization

Mobile Accessibility specific terms for this preferences exactly coincide with the definition of its corresponding common terms. This makes trivial its transformations, that can be implemented like this:

```
"access_commonprefs_punctuation":
"display.screenReader.-provisional-punctuationVerbosity",

"access_commonprefs_capitalization":
"display.screenReader.-provisional-announceCapitals"
```

In both cases we're simply indicating that the Mobile Accessibility specific term can take the exact value coming from its corresponding common term.

6 Online Flow Manager

Once the online servers of Cloud4All/GPII architecture have the information about Mobile Accessibility and how to transform common terms in specific terms for our app, we can request the Online Flow Manager (part of the architecture in charge of

receiving requests for preferences sets for a given user ID, communicate with other architecture components and provide the preferences back to the caller according to user's needs and preferences and transformations defined for each solution) for a set of preferences for a given user. The requested preference set will be retrieved by the Flow Manager and translated to Mobile Accessibility specific terms.

This call to the online Flow Manager can be done via HTTP GET request in the form defined in Flow Manager documentation from GPII wiki page[3].

Preferences sets from the online Flow Manager are provided also in JSON format, which can be easily parsed using org.json components natively available on Android.

7 NFC User Listener

User identification for Mobile Accessibility has been based on NFC, so the user can request Mobile Accessibility to be auto configured according to his needs and preferences by simply touching the back side of the device with a NFC token. This token has to provide the userID in plain text format, future implementations could change according to security requirements, for example.

NFC support natively provided by Android API through android.nfc library.

References

1. Code Factory website, http://www.codefactory.es
2. Architecture - Available Transformation Functions documentation, GPII Wiki site: http://wiki.gpii.net/index.php/Architecture_-_Available_transformation_functions
3. Flow Manager API - Cloud based deployment, GPII Wiki site: http://wiki.gpii.net/index.php/Flow_Manager_API#Cloud_Based_Deployment