Konzept:
Entwicklungsleitfaden Funktionstasten

Autor:	Version:	Datum:
Auerswald GmbH & Co. KG	1.2	26.06.2015

Versionsverfolgung

Version	Datum	Änderung	Verfasser
1.0	25.10.2013	Erste Version.	JWe
1.1	12.06.2015	COMfortel 3600 IP hinzugefügt	JWe
1.2	26.06.2015	Anpassungen nach interner Überarbeitung	JWe

Inhaltsverzeichnis

1 Gültigkeit

Dieses Dokument gilt für COMfortel 1400 IP und COMfortel 2600 IP ab Version 2.2A, sowie für COMfortel 3600 IP ab Version 2.2G.

2 Überblick

Die Erweiterbarkeit durch Apps bildet einen Grundbestandteil des Platformgedankens. Besonders die Integration verschiedener Sensoren und Hardwareelemente macht das Erzeugen einer App für gewisse Lösungen erst interessant.

Die Bereitstellung von Funktionstasten und den zugeordneten LEDs durch eine API für externe Entwickler kann in diesem Zusammenhang einen großen Mehrwert der COMfortel IP Telefone darstellen. Besonders Applikationen mit einem großen Bedarf an Tasten oder Anzeigeelementen (z.B. in der Hausautomation) können von einer solchen API profitieren.

In diesem Dokument soll dargestellt werden, welche Schritte notwendig sind, um die Tastenbelegung und Konfiguration in einer App umzusetzen.

3 Konzept

Eine Funktionstasten API besteht aus 3 Komponenten:

1. Telefonseitige Verarbeitung
2. API Bibliothek
3. Applikation

Diese Komponenten besitzen jeweils eine definierte Funktion:

Die telefonseitige Verarbeitung verwaltet eine Zuordnung der verfügbaren Tasten. Sie reagiert auf Tastendrücke und leitet diese über die API Bibliothek an die registrierte Applikation weiter. Weiterhin nimmt sie von der Bibliothek LED-Steuerbefehle für die zugeordneten Tasten an, prüft die Rechte und führt die Kommandos aus.

Die Bibliothek bietet Abstraktionen und definiert Mechanismen, die von Anwendungsentwicklern verwendet werden können, um einfach die gewünschten Funktionen auszuführen. Sie dient primär als Transportschicht zwischen Applikation und System.

Die Applikation nutzt die bereitgestellten Elemente für beliebige Problemstellungen. Hierzu verwendet sie die Bibliothek, die in den Classpath der Applikation eingebunden werden muss.

4 Basiskomponenten

Eine Applikation wird wie andere Funktionstasten im System angemeldet. Um dieses zu erreichen, muss eine Applikation zwei dinge im AndroidManifest deklarieren:

1. Eine Konfigurationsactivity, die auf die Action **CONFIGURE** reagiert.

	Configure-Intent	
Action	**auerswald.intent.action.CONFIGURE**	**M**
Extra(Int)	**key_id**	**m**

2. Einen Broadcastreceiver, der Key Intents empfängt.

	Key-Bind-Intent	
Action	**auerswald.intent.action.KEY_BIND**	**m**
Extra(Int)	**key_id**	**m**
Extra(boolean)	**bind**	**m**

	Key-Press-Intent	
Action	**auerswald.intent.action.KEY_PRESS**	**m**
Extra(Int)	**key_id**	**m**

Die Funktionstasten Konfiguration sucht nach einer Activity, startet diese mit dem Configure Intent. Das gesendete Intent enthält bereits die vom Benutzer gewählte Key ID. Die Konfiguration wartet auf ein Ergebnis der gestarteten Activity. Wird ein RESULT_OK von der App geschickt, wird überprüft ob das data Intent ein Name Extra enthält.

	Configure-Result-Intent	
Extra(String)	**key_name**	**o**

Dieses Extra wird, wenn vorhanden als Beschriftung der Taste verwendet.

Anschließend wird die Einrichtung abgeschlossen. Und ein KEY_BIND an den Receiver der App für die Taste geschickt. Dieses KEY_BIND enthält die für die Applikation spezifische key_id, welche einen seperaten Addressbereich für jede Applikation bereitstellt. Hierdurch wird verhindert, dass beliebige Apps LEDs von Tasten verändern, für die sie keine Berechtigung besitzen. Ein KEY_BIND Intent enthält ein Extra, welches anzeigt ob eine Taste belegt oder gelöscht wurde.

Nachfolgendes Beispiel zeigt ein mögliches AndroidManifest:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="de.auerswald.pack"
    android:versionCode="2"
    android:versionName="1.0" >

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >

        <!-- Configuration Activity -->
        <activity
            android:name="de.auerswald.pack.CustomerConfigurationActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="auerswald.intent.category.KEY_CONFIGURE"/>
            </intent-filter>
        </activity>

           <!-- Key Receiver Implementation extends KeyListener -->
        <receiver android:name=".CustomerKeyListener">
             <intent-filter>
                <action android:name="auerswald.intent.action.KEY_BIND" />
                <action android:name="auerswald.intent.action.KEY_PRESSED" />
            </intent-filter>
        </receiver>

        <!-- Service to bind the Led Service and handle state -->
        <service android:name=".KeyHandlingService" />
    </application>
</manifest>
```

5 Konfiguration

Die Konfigurations-Activity der App sollte eventuell notwendige Daten von dem Nutzen annehmen.

6 Reagieren auf Tastendrücke

Das Framework bietet einen abstrakte Receiver an, der das Entpacken der Broadcasts durchführt und eine onKeyBind- und eine onKeyPressed-Hookmethode aufruft.

In diesen kann somit auf Tastendrücke reagiert werden. Da es sich um einen BroadcastReceiver handelt, der nicht alle Operationen gestattet, ist es unter Umständen notwendig die Tastendrücke weiterzuleiten.

7 Setzen von LEDs

Um LEDs für eine Taste zu setzen, muss mit dem folgenden Intent ein Service gebunden werden.

	Set-Led-Intent	
Action	**auerswald.intent.action.SET_LED**	m

Der zu diesem Service notwendige Service-Stub befindet sich in der bereitgestellten JAR-Datei. Auf diesem Service steht eine setLed Methode bereit, die die KeyId, den Modus und die Farbe der Taste bekommen.

Modus und Farbe befinden sich als Konstanten in der Klasse Constants.

8 Konfiguration über Backup/Provisioning

Die Konfiguration über ein Backup folgt den Einstellungen aus dem auer_settings.xsd.

Hier ist auch definiert, wie externe Tasten in das Telefon Provisioniert werden können.

Der dort verwendete Block nutzt eine spezielle Syntax:

```
<key number="1" level="0">
        <function xsi:type="externApplication">
                <command>$Intent</command>
        </function>
</key>
```

9 Klassendiagramm