法人向け mazec テクニカルガイド

外部アプリケーション連携

（Android 版）

第6版

はじめに

本書では、法人向け mazec for Business の概要や、外部アプリケーションと連携して使うための仕様について説明しています。
本書の記載は、Java、XML(Android 開発言語)、HTML 等に関する基本知識を持っていることを前提としています。

コードの表記

- コードの記述例は、囲みの中に記述します。
- 1 行に収まらない場合は、折り返すか、途中で改行して記述する場合があります。

```
<EditText ... android:privateImeOptions="input_mode=2&amp;filter=8" />
```

目次

第 1 章　法人向け mazec とは

法人向け mazec とは、コンシューマー向け製品とは異なり、次の法人向け機能が用意されています。

- アプリケーション連携
- 異体字サポート
- Android Tablet対応UIサポート

アプリケーション連携

ユーザーが行う mazec の操作の一部をお客様が開発するアプリケーションから実行することができます。
例えば、入力モードの切り替えや認識文字種の変更などをアプリケーションから指定することができます。

異体字サポート

異体字とは、標準の字体と同じ意味・発音を持つが、表記に差異がある漢字のことです。法人向け mazec では、姓名に対してよく使われる異体字をサポートしています。
例えば、「渡辺」と手書きすると、かな漢字変換候補に「渡辺」「渡邉」「渡邊」と表示されます。
そのほかに「高田」「髙田」、「斉藤」「齋藤」「齊藤」などの異体字がサポートされます。

※ 機種依存により、異体字の表記には次の制約があります。
・異体字がほかの文字に変換される場合があります。(例：□、スペース、その他の文字など)
・異体字が標準の字体で表記される場合があります。
ただし、mazec からは正しい文字コードに変換しています。

Android Tablet対応UI

Android Tablet 向けの UI は、法人向け mazec のみ提供されるものです。
5 インチ以上のタブレット向けに快適な操作ができる UI が提供されます。

第 2 章　mazec for Business

2.1. mazec for Business のアプリケーション外部連携概要

mazec for Business は、下記の2つのアプリケーションから制御することができます。

- Android OSネイティブアプリケーション
- 標準ブラウザによるWebアプリケーション

Android OSネイティブアプリケーションからの制御

お客様が開発するAndroid OSネイティブアプリケーションから mazec の下記の動作を制御できます。

- 入力モードの指定
- 認識文字種の指定

この指定方法には、下記の2つの方法（タイミング）があります。

- 入力コントロールごとに指定する
- mazec を起動させて指定する

標準ブラウザによるWebアプリケーションからの制御

標準ブラウザによるWebアプリケーションから mazec の下記の動作を制御できます。

- 入力モードの指定

標準ブラウザの `input` 要素の `type` 属性に応じて、mazec の入力モードが決定されます。

2.2. mazec for Business の機能

2.2.1　入力モードの切り替え

アプリケーションから入力モードを指定して mazec を起動することができます。
mazec による入力方法には、下記の3つのモードがあります。

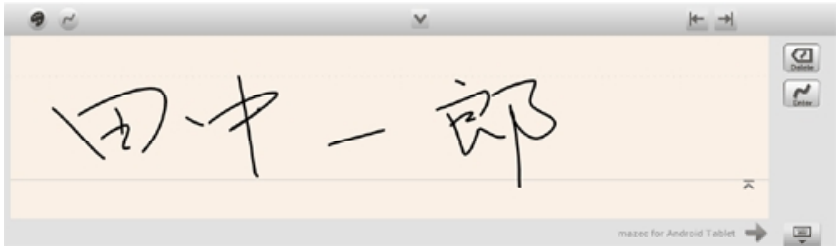
- 手書き認識による交ぜ書き入力モード
- ソフトウェアキーボード入力モード
- イメージ出力モード

入力項目の内容に適した入力モードで mazec を起動（表示）できます。
例えば、住所や氏名の入力欄は手書きによる交ぜ書き入力モード、電話番号の入力欄にはソフトウェアキーボード入力モード、署名の入力欄にはイメージ出力モードを指定することで、入力欄に応じた最適な入力方法を提供することができます。

図：交ぜ書き入力モード（スタンダードモード）

図：ソフトウェアキーボード入力モード（スタンダードモードのかな漢字変換時時）

※　あいう/ABCキーで、直接入力(半角英字入力)とかな漢字変換を切り替えて利用できます。

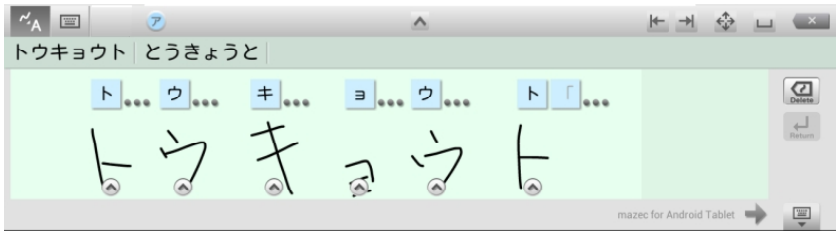


図：イメージ出力モード（スタンダードモード）

※　ストロークの色や太さを変更できます。

2.2.2　認識文字種の設定

入力欄ごとに認識文字種を指定できます。特定の文字の種類を入力する場合、認識文字種を指定すると認識率が高くなります。
例えば、フリガナ用の入力欄に対して認識文字種をカタカナに設定すると、変換候補にはカタカナが優先的に表示されます。



図：認識文字種の設定（認識文字種：カタカナに設定）

入力モードと認識文字種を組み合わせて指定すると、より効率的な文字入力を実現できます。

2.3. プログラムインターフェースの仕様

2.3.1 入力コントロールへの入力モードと認識文字種の指定

入力コントロールに対して、入力モードや認識文字種を指定します。mazec は指定された入力モードと認識文字種で起動します。指定しない場合、以前の状態で表示されます。
例）

- 入力コントロールA：入力モード=交ぜ書き、認識文字種=漢字を指定
- 入力コントロールB：入力モード=交ぜ書き、認識文字種=ひらがなを指定

上記の場合、フォーカスを入力コントロールAから入力コントロールBに移動すると、入力モードが交ぜ書き、認識文字種がひらがなの状態で mazec が表示されます。
また、mazec 以外のIMEから mazec に変更すると、指定した入力モードと認識文字種で mazec が表示されます。ただし、フォーカスを移動せずに mazec をいったん閉じて再表示した場合は、以前の状態で表示されます。

EditText で指定する場合

例）`layout.xml`

```
<EditText ... android:privateImeOptions="input_mode=2&amp;filter=8" />
```

独自のビューで指定する場合

`View.onCreateInputConnection` をオーバーライドし、引数 `EditorInfo` のメンバ `privateImeOptions` で指定します。
例）

```
outAttrs.privateImeOptions = "input_mode=2&filter=8";
```

input_mode / filter の値

プロパティ	値
input_mode	2：交ぜ書き 3：キーボード 4：イメージ出力
filter	0：All 8：漢字 1：ひらがな 2：カタカナ 3：アルファベット 4：数字 7：記号

イメージデータの入力方法については、次をご覧ください。
▶ 3.3.4 イメージデータの出力

2.3.2　mazec を起動させて入力モードと認識文字種を指定

アプリケーションから mazec（IME）を起動（表示）させるタイミングで、入力モードと認識文字種を指定することができます。IMEが mazec になっている状態で、入力コントロールにフォーカスがある場合に指定したパラメーターにしたがって、mazec（IME）を起動（表示）させることが可能です。
InputMethodManager クラスの下記のメソッドを使用します。

※　詳しくは、Android Developer リファレンスをご参照ください。

- **使用するメソッド**

```
public void sendAppPrivateCommand (View view, String action, Bundle data)
```

入力モードの変更

- **アクション名** *1

```
com.metamoji.mazec.action.set_input_mode
```

- **データ**

キー名	型	値
data	int	2：交ぜ書き 3：キーボード

認識文字種の変更

- **アクション名** *1

```
com.metamoji.mazec.action.set_filter
```

- **データ**

キー名	型	値
data	int	0：All 8：漢字 1：ひらがな 2：カタカナ 3：アルファベット 4：数字 7：記号

※1　お客様が作成した下記を含む旧来のコードは、そのまま利用することができます。

com.sevenknowledge.mazec.action.set_○○

2.3.3 input 要素 type 属性による入力モードの切り替え

標準ブラウザの input 要素の type 属性の指定に応じて、mazec の入力モードが変わります。例えば、数字だけを入力する 1 行入力フィールドに対し、type 属性に"number"を指定すると、ソフトウェアキーボードで mazec を表示させることができます。
input 要素の type 属性の指定による mazec の入力モードは、次のようになります。

type 属性	mazec の入力モード
text	交ぜ書き
search	交ぜ書き
tel	ソフトウェアキーボード
url	交ぜ書き
number	ソフトウェアキーボード
password	ソフトウェアキーボード
email	ソフトウェアキーボード

制限事項

端末によっては、input 要素の type 属性に"email"を指定したとき、mazec の入力モードがソフトウェアキーボードではなく、交ぜ書きになる場合があります。

2.3.4 イメージデータの入力

出力イメージサイズの指定

input_mode の値に"4"が指定されている場合、mazec がイメージ出力モードで起動します。その際、下記のメソッドを使用して mazec から出力イメージのサイズを問い合わせます。

```
inputConnection.performPrivateCommand(String action, Bundle data)
```

アプリケーション側ではフレームワークから下記のメソッドが呼び出されます。

```
public boolean onPrivateIMECommand (String action, Bundle data)
```

- アクション名

```
com.metamoji.mazec.action.query_image_size
```

- データ

キー名	型	値
data	null	-

出力イメージのサイズを指定（返答）するには、下記のメソッドを使用します。

```
public void sendAppPrivateCommand (View view, String action, Bundle data)
```

- **アクション名**

```
com.metamoji.mazec.action.reply_image_size
```

- **データ**

キー名	型	値
width	int	画像の幅（ピクセル）
height	int	画像の高さ（ピクセル）

※ 出力イメージのサイズが指定されていない場合は、手書き入力領域の大きさに正規化して出力します。
※ タイミングによっては正しく設定されない場合があるため、出力サイズの指定（返答）は mazec からの問い合わせがあってから行ってください。

データの出力

イメージ出力モードで確定された場合、下記のメソッドを使用して、通常のテキストではなく入力されたストロークイメージをBitmapオブジェクトとしてアプリケーション側に通知します。

```
inputConnection.performPrivateCommand(String action, Bundle data)
```

アプリケーション側ではフレームワークから下記のメソッドが呼び出されます。

```
public boolean onPrivateIMECommand (String action, Bundle data)
```

- **アクション名**

```
com.metamoji.mazec.action.commit_image
```

- **データ**

キー名	型	値
bitmap	Bitmap	画像

アプリケーションは、受け取った Bitmap オブジェクトを利用して、画面に表示したりファイルに保存したりできます。

Appendix A　サンプルコード

【ご注意】　本サンプルコードはアプリケーションの動作保証をするものではありません。
お客様の責務にて、Android OS のリファレンスにしたがって、開発をお願い致します。

EditTextにて入力コントロールに入力モードと認識文字種を指定するサンプルです。

```
<RelativeLayout  xmlns:android="http://schemas.android.com/apk/res/android"
 xmlns:tools="http://schemas.android.com/tools"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 tools:context=".MainActivity" >

<EditText
 android:id="@+id/editText1"
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:layout_alignParentLeft="true"
 android:layout_alignParentTop="true"
 android:ems="10"
 android:inputType="textMultiLine"
 android:privateImeOptions="input_mode=2&amp;filter=0">
</EditText>

</RelativeLayout>
```

Appendix B　サンプルコード

アプリケーション側から mazec にイメージ出力モードを指定し、mazec から出力されたイメージを表示するサンプルです。

- EditText 派生のテキストボックスをタップするとイメージ出力モードで mazec が起動します。
- 出力イメージのサイズは 640 × 240 を指定します。
- 出力されたイメージをイメージビューに描画します。

main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:background="@android:color/white"
  >

  <com.metamoji.mazec.signature.test.MyEditText
    android:id="@+id/myEditText"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="タップして画像入力"
    android:singleLine="true"
    android:privateImeOptions="input_mode=4"
    />

  <ImageView
    android:id="@+id/imageView"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:background="@android:color/white"
    android:onClick="buttonclick"
  />
</LinearLayout>
```

MyActivity.java

```
  public class MyActivity extends Activity
  {
   ...

    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.main);

        final MyEditText myEditText = (MyEditText)
findViewById(R.id.myEditText);
final ImageView imageView = (ImageView)findViewById(R.id.imageView);

        myEditText.setCommitImageListener(new MyEditText.CommitImageListener() {
          public void commitImage(Bitmap image) {
            imageView.setScaleType(ImageView.ScaleType.CENTER);
            imageView.setImageBitmap(image);
            imageView.invalidate();
          }
        });
    }
}
```

MyEditText.java

```
public class MyEditText extends EditText
{
  public boolean onPrivateIMECommand (String action, Bundle data) {
    boolean handled = false;

    if (action.equals("com.metamoji.mazec.action.commit_image")) {
      Bitmap bitmap = (Bitmap)data.get("bitmap");
      if (null != m_commitImageListener) {
        m_commitImageListener.commitImage(bitmap);
      }
      handled = true;

    } else if (action.equals("com.metamoji.mazec.action.query_image_size")) {
      InputMethodManager imm =
(InputMethodManager)getContext().getSystemService(Context.INPUT_METHOD_SERVICE);
      if (null != imm) {
        Bundle bundle = new Bundle();
        bundle.putInt("width", 640);
        bundle.putInt("height", 240);
        imm.sendAppPrivateCommand(this,
"com.sevenknowledge.mazec.action.reply_image_size", bundle);
        handled = true;
      }
    }
    return handled ? true : super.onPrivateIMECommand(action, data);
  }

  public static interface CommitImageListener {
    public void commitImage(Bitmap image);
  }
  private CommitImageListener m_commitImageListener;
  public void setCommitImageListener(CommitImageListener listener) {
    m_commitImageListener = listener;
  }
}
```