DOCUMENTATION ON "ACPL FM220 RD" APPLICATION

Contents

1. Install Test Applications :

Install “ACPL FM220 RD” application and “ACPL-Aadhaar Auth Demo” application from Google Play store in Android phone.

First open ACPL FM220 RD application and connect device to mobile

- Note : -
  Device connection and registration process are explain in Android RD Service User Manual for that please check this link :-
  https://www.acpl.in.net/assets/pdf/ACPL_FM220_RD_Document_ANDROID.pdf

2. PID Data XML:

String capture (String **pidOptions**):

<!--main capture call which takes PidOptions XML data as input and returns PidData XML as output-->

String getDeviceInfo (); <!-- utility method to obtain DeviceInfo XML from the RD Service-->

<PidOptions ver="">

<!--PidOptions: ver: Version of the PidOtopns spec. currently, it is “1.0”. This is necessary to allow applications to gracefully upgrade even when RD service may be been upgraded. **RD Service must support current version and one previous version** to allow apps to upgrade at different points in time-->

<Opts fCount="" fType="" iCount="" iType="" pCount="" pType="" format="" pidVer="" timeout="" otp="" wadh="" posh="" env=""/> <Demo></Demo>

<!-- Element allows demographic data to be passed to form PID block as per authentication specification-->

<CustOpts>

<!-- no application should hard code these and should be configured on app or AUA servers. These parameters can be used for any custom application authentication or for other configuration parameters. Device providers can differentiate their service in the market by enabling advanced algorithms that applications can take advantage of. -->

<Param name="" value="" />

</CustOpts>

<!-- Allows vendor specific options to be passed. Param element may repeat-->

</PidOptions>

```
/* Opts:

        Int fCount (optional) number of finger records to be captured (0 to 10)
        Int fType (optional) ISO format (0 for FMR or 1 for FIR), 0 (FMR) is default
        Int iCount (optional) number of iris records to be captured (0 to 2)
        Int iType (optional) ISO format (0 for IIR), 0 (IIR) is default
        Int pCount (optional) number of face photo records to be captured (0 to 1). Currently face matching is
                not supported.
        Int pType (optional) face format. Currently face matching is not supported.
        Int format (mandatory) 0 for XML, 1 for Protobuf
        String pidVer (mandatory) PID version
        Int timeout capture timeout in milliseconds
        String env (optional) default value being "P" (prod), other valid values are "PP" (preProd) and
                "S" (Staging).
        String otp (optional) OTP value captured from user in case of 2-factor auth
        String wadh (optional) If passed, RD Service should use this within PID block root element "as-is".
        String posh (optional) if specific positions need to be captured, applications can pass a comma
                delimited position attributes. See "posh" attribute definition in Authentication Specification
                for valid values. RD Service (if showing preview) can indicate the finger using this. If passed,
                this should be passed back within PID block. Default is "UNKNOWN", meaning "any"
                finger/iris can be captured.          */
```

```
<PidData>
        <Resp errCode="" errInfo="" fCount="" fType="" iCount="" iType="" pCount="" pType=""
        nmPoints="" qScore=""/>
        <DeviceInfo />
        <Skey ci="">encrypted and encoded session key</Skey>
        <Hmac>SHA-256 Hash of Pid block, encrypted and then encoded</Hmac>
<Data type="X|P"> base-64 encoded encrypted pid block </pid> </PidData>
```

```
/* Resp:
        Int errCode (mandatory) 0 if no error, else standard error codes
        String errInfo (optional) additional info message in case of error/warning
        Int fCount (mandatory for FP) number of finger records actually captured
        Int fType (mandatory for FP) actual format type – 0 (FMR) or 1 (FIR)
        Int iCount (mandatory for Iris) number of iris records actually captured
        Int iType (mandatory for Iris) actual Iris format (0 for IIR)
        Int pCount (mandatory for Photo) number of face photo records actually captured. Currently face
            matching is not supported
        Int pType (mandatory for Photo) face format. Currently face matching is not supported.
        Int nmPoints (mandatory for FMR capture) Number of minutiae points when FMR is captured.
            Applications may use this for accepting or retrying the capture. If multiple fingers are captured,
            send comma delimited numbers.
        Int qScore (optional) if quality check is done, send a normalized score that is between 0 and 100.
            Device providers may allow configuration within RD service to use specific quality check
```

```
                algorithms to be enabled. Either it can be configured within RD service or applications can
pass those under PidOptionsCustOptsParam.  Skey:
            String skey (mandatory) encrypted session key as per auth spec
           String ci (mandatory) UIDAI public key identifier as per auth spec
Hmac:
            String hmac (mandatory) hmac value as per auth spec
 */
```

```
<DeviceInfo dpId="" rdsId="" rdsVer="" dc="" mi="" mc="" />

/*          dpId – (mandatory) Unique code assigned to registered device provider.
            rdsId – (mandatory) Unique ID of the certified registered device service.
            rdsVer – (mandatory) registered devices service version.  dc – (mandatory)
            Unique Registered device code. mi – (mandatory)
            Registered device model ID. mc – (mandatory) this attribute holds registered
            device public key certificate. This is signed with device provider key.
             */
```

3. Intent Create:

3.1 Create Intent for Device Info:

```
intentInfo = new Intent("in.gov.uidai.rdservice.fp.INFO");
PackageManager  packageManager  = getPackageManager();
List activities =
packageManager.queryIntentActivities(intentInfo,PackageManager.MATCH_DEFAULT_ON
LY
);


boolean isIntentSafe = activities.size() > 0;
            if (!isIntentSafe) {

            //"No Finger Scanner Support available"

             btnAuthenticate.setEnabled(false);
             btnCapture.setEnabled(false);
            }
```

3.2 Create Intent for Capture Call:

```
intentCapture  = new Intent("in.gov.uidai.rdservice.fp.CAPTURE");
packageManager  = getPackageManager();  activities
=
packageManager.queryIntentActivities(intentCapture,PackageManager.MATCH_DEFAULT
_O NLY);

isIntentSafe  = activities.size()  > 0;
        if (!isIntentSafe)  {

        //"No Finger Scanner Support available"

        btnAuthenticate.setEnabled(false);
        btnCapture.setEnabled(false);
        }
```

4. Intent Call:

- The integer argument is a "request code" that identifies your request. When you receive the result Intent, the call back provides the same request code so that application can properly identify the result and determine how to handle it. (For result of this call, refer "Activity Result" code.)

```
int resultCode = 1;
startActivityForResult(intentInfo, resultCode);

pidDataXML = "";

int resultCode  = 13;
startActivityForResult(intentCalim,  resultCode);

int resultCode  = 14;
startActivityForResult(intentRelease,  resultCode);
```

5. Activity Result:

 Activity Result is depends as per value passed in intent call. You can change values as per your requirement.

<!-- The System calls activity's onActivityResult() method when user will be done with subsequent activity and returns. The onActivityResult() method includes three arguments:

1. The request code passed to startActivityForResult().
2. A result code specified by the second activity. This is RESULT_OK if the operation was successful.
3. An Intent that carries the result data. -->

```
protected void onActivityResult(int requestCode, int resultCode, Intent data)  {
super.onActivityResult(requestCode, resultCode, data);
if (resultCode == Activity.RESULT_OK) {
if (requestCode == 1) {
                    String regis_info = data.getStringExtra("RDService");
          if (regis_info != null && regis_info.contains("NOTREADY")) {
                              //"YOUR DEVICE IS NOT CONNECTED.or Other error Display");
     }  else {
                                   String deviceInfoXML = data.getStringExtra("RDService");
            DeviceInfoXml = data.getStringExtra("DEVICE_INFO");
            if (deviceInfoXML == null || deviceInfoXML.equals("") ||  deviceInfoXML.isEmpty()) {
                                             //"Error occurred in DEVICE DRIVER INFO XML"
                              } else {
                                                                       //as per your requirements
                                   }
           }
            } else if (requestCode == 2) {
                                   pidDataXML = data.getStringExtra("PID_DATA");
                         if (pidDataXML != null) {
                                             //"PID DATA XML :- \n" + pidDataXML
                                             if (pidDataXML.equals("") || pidDataXML.isEmpty()) {

                                                       //"Error occurred in PID DATA XML"
                                                       return;
                                             }
                                             if (pidDataXML.startsWith("ERROR:-")) {

                                                       //pidDataXML
                                                       return;
                                             }
DocumentBuilderFactory db =  DocumentBuilderFactory.newInstance();
Document inputDocument =  db.newDocumentBuilder().parse(new InputSource(new
StringReader(pidDataXML)));

NodeList nodes =  inputDocument.getElementsByTagName("PidData");
                                if (nodes  != null) {
                                Element element = (Element) nodes.item(0);
                                NodeList respNode =
inputDocument.getElementsByTagName("Resp");
                                                if (respNode  != null) {
                                                        .....
                                                }
                      }
                                if (pidDataXML != null) {   //deviceInfoXML  != null &&

                                        //"Scan success"
                                } else {
                                        //"Scan Failure"
                                }
                        }

                } else if (requestCode == 3) {

                                //request  for application settings

                }  else if (requestCode ==13) {
                                String value = data.getStringExtra("CLAIM");

                                //usb  connect /disconnect
                }else if (requestCode ==14) {
                                String value = data.getStringExtra("RELEASE");
                                //usb  release
                }else {}

        }
}
```

Following is the XML data format for authentication API:

```
<Auth uid="" rc="" tid="" ac="" sa="" ver="" txn="" lk="">
<Uses pi="" pa="" pfa="" bio="" bt="" pin="" otp=""/>
<Meta udc="" rdsId="" rdsVer="" dpId="" dc="" mi="" mc="" />
<Skey ci="">encrypted and encoded session key</Skey>
<Hmac>SHA-256  Hash of Pid block, encrypted and then encoded</Hmac>
<Data type="X|P">encrypted PID block</Data>
<Signature>Digital  signature  of AUA</Signature>   </Auth>
```

The PIDdata received form RDserviec as below

```
<PidData>
<Resp errCode="" errInfo="" fCount="" fType="" iCount="" iType="" pCount="" pType=""
nmPoints="" qScore=""/>
<DeviceInfo />
<Skey ci="">encrypted and encoded session key</Skey>
<Hmac>SHA-256 Hash of Pid block, encrypted and then encoded</Hmac>
<Data type="X|P"> base-64 encoded encrypted pid block </pid>
</PidData>
```

- Data, Skey and HMac elements from Piddata to Auth are to be used as it is. Deviceinfo in Piddata will provide other elements required for Meta. UDC is as per your choice. It should be present but it is not checked.
- Auth XML is to be signed using UIDAI private key as it was signed in version 1.6.