RD SERVICE PROGRAMMER MANUAL FOR WINDOWS

You can download sample code from : http://acpl.in.net/RdService.html

1. Introduction: -

As per UIDAI Aadhar API 2.0 specification, RD Service is used to generate encrypted PID data block using captured biometrics. RD Service will sign and encrypt captured biometrics as per UIDAI specifications and then generate encrypt PID block with necessary registered device information.

We will refer sample code to use RD Service functionality in three different programming languages: C#, JAVA, JavaScript.

Here in all examples, 11100th port is only for reference.
RD service can be active on 11100 to 11120.

FOR JAVA PLEASE USE JDK >= 1.8.

- **RD Service Info XML: -**

Using RD Service Info XML, one can determine that device is in READY/NOTREADY/USED status.
Refer sample RD Service Info XML given below for more knowledge.
Refer following snippets for programming guidance,

➔ **C-Sharp: -**

```
string completeUrl = http://localhost:11100;
HttpWebRequest request = (HttpWebRequest)WebRequest.Create(completeUrl);
request.Method = "RDSERVICE";
request.Credentials = CredentialCache.DefaultCredentials;
request.ContentType = "text/xml";
WebResponse response = default(WebResponse);
response = request.GetResponse();
Stream str = response.GetResponseStream();
StreamReader sr = new StreamReader(str);
string finalResponse = sr.ReadToEnd();
```

➔ **JAVA: -**

```
DefaultClientConfig config = new DefaultClientConfig();
config.getProperties().put(URLConnectionClientHandler.PROPERTY_HTTP_URL_CONNECTION_SET
                                _METHOD_WORKAROUND, true);
Client c = Client.create(config);
WebResource r = c.resource("http://localhost:11100");
String reponse = r.method("RDSERVICE", String.class);
```

➔ **JavaScript: -**

```
<!-- HTML button control to call rdservice info call -->
<input type="button" id="btnRDInfo" onclick="rdservice()"/>

<!– JavaScipt to handle reservice info call  -->
<script type="text/javascript">
      function rdservice() {
            var port;
            var urlStr = '';
            urlStr = 'http://localhost:11100/';

            getJSON_rd(urlStr,
                  function (err, data) {
                        if (err != null) {
                              alert('Something went wrong: ' + err);
                        } else {
                              alert('Response:-' + String(data));
                        }
                  }
            );
       }
      var getJSON_rd = function (url, callback) {
            var xhr = new XMLHttpRequest();
            xhr.open('RDSERVICE', url, true);
            xhr.responseType = 'text';
            xhr.onload = function () {
                  var status = xhr.status;
                  if (status == 200) {
                        callback(null, xhr.response);
                  } else {
                        callback(status);
                  }
            };
            xhr.send();
        };

</script>
```

Sample RD Service Info XML: -

```
<RDService xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema" status="READY" info="RD service for
   Startek FM220 provided by Access Computech">
      <Interface id="CAPTURE" path="/rd/capture" />
      <Interface id="DEVICEINFO" path="/rd/info" />
</RDService>
```

- **Device Info XML: -**

  Using Device Info XML, one can get all registered device related information like device certificate, device code, device provider code etc.
  Refer sample Device Info XML given below for more knowledge.
  Refer following snippets for programming guidance,

  ➔ **C-Sharp: -**

```
string completeUrl = "http://localhost:11100/rd/info";
HttpWebRequest request = (HttpWebRequest)WebRequest.Create(completeUrl);
request.Method = "DEVICEINFO";
request.Credentials = CredentialCache.DefaultCredentials;
request.ContentType = "text/xml";
WebResponse response = default(WebResponse);
response = request.GetResponse();
Stream str = response.GetResponseStream();
StreamReader sr = new StreamReader(str);
string finalResponse = sr.ReadToEnd();
```

  ➔ **JAVA: -**

```
DefaultClientConfig config = new DefaultClientConfig();
config.getProperties().put(URLConnectionClientHandler.PROPERTY_HTTP_URL_CONNECTION_SET_
                           METHOD_WORKAROUND, true);
Client c = Client.create(config);
WebResource r = c.resource("http://localhost:11100/rd/info");
String reponse = r.method("DEVICEINFO", String.class);
```

➔ **JavaScript: -**

```
<!-- HTML button control to call device info call -->
<input type="button" id="btnDevInfo" onclick="info()"/>

<!– JavaScipt to handle device info call  -->
<script type="text/javascript">
	function info() {
		var port;
		var urlStr = '';
		urlStr = 'http://localhost:11100/rd/info';
		getJSON_info(urlStr,
			function (err, data) {
				if (err != null) {
					alert('Something went wrong: ' + err);
				} else {
					alert('Response:-' + String(data);

				}
			}
		);
	 }

	 var getJSON_info = function (url, callback) {
		var xhr = new XMLHttpRequest();
		xhr.open('DEVICEINFO', url, true);
		xhr.responseType = 'text';
		xhr.onload = function () {
			var status = xhr.status;
			if (status == 200) {
				callback(null, xhr.response);
			} else {
				callback(status);
			}
		};
		xhr.send();
	 };

</script>
```

Sample Device Info XML: -

```
<DeviceInfo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        dpId="(DEVICE-PROVIDER_ID)"
        rdsId="(RD Service ID)"
        rdsVer="(RD Service Version)" m
        dc="(DEVICE_CODE)" mi="(MODEL_ID)"
        mc="(DEVICE_CERTIFICATE_BASE64_VALUE)"
        error="">
            <additional_info key="" value="" />
</DeviceInfo>
```

- **Capture: -**

  Using Capture call, one can capture biometric data and get the encrypted PID data XML.
  For capture we have to send PID OPTION XML as input.
  Refer example PID OPTION XML.
  Refer sample PID data XML given below for more knowledge.
  Refer following snippets for programming guidance,

  ➔ **C-Sharp: -**

```
string completeUrl = "http://localhost:11100/rd/capture";
HttpWebRequest request = (HttpWebRequest)WebRequest.Create(completeUrl);
request.Method = "CAPTURE";
request.Credentials = CredentialCache.DefaultCredentials;
StreamWriter writer = new StreamWriter(request.GetRequestStream());
string pidOptString = "<PidOptions><Opts fCount=\"1\" fType=\"0\" iCount=\"0\"
pCount=\"0\"    format=\"0\" pidVer=\"2.0\" timeout=\"20000\" otp=\"\"
posh=\"LEFT_INDEX\" env=\"S\" wadh=\"\" /> <Demo></Demo> <CustOpts> <Param
name=\"Param1\" value=\"\" /> </CustOpts> </PidOptions>";
writer.WriteLine(pidOptString);
writer.Close();
WebResponse response = default(WebResponse);
response = request.GetResponse();
Stream str = response.GetResponseStream();
StreamReader sr = new StreamReader(str);
string finalResponse = sr.ReadToEnd();
```

  ➔ **JAVA: -**

```
String pidOpt = "<PidOptions><Opts fCount=\"1\" fType=\"0\" iCount=\"0\" pCount=\"0\"
format=\"0\" pidVer=\"2.0\" timeout=\"20000\" otp=\"\" posh=\"LEFT_INDEX\" env=\"S\"
wadh=\"\" /> <Demo></Demo> <CustOpts> <Param name=\"Param1\" value=\"\" /> </CustOpts>
</PidOptions>";
DefaultClientConfig config = new DefaultClientConfig();
config.getProperties().put(URLConnectionClientHandler.PROPERTY_HTTP_URL_CONNECTION_SET
_METHOD_WORKAROUND, true);
Client c = Client.create(config);
WebResource r = c.resource("http://localhost:11100/rd/capture");
String reponse = r.method("CAPTURE", String.class, pidOpt);
```

➔ **JavaScript: -**

```
<!-- HTML button control to call capture call -->
<input type="button" id="btnCapture" onclick="captureFPAuth()"/>

<!– JavaScipt to handle capture call  -->
<script type="text/javascript">
  function captureFPAuth() {
       var port;
       var urlStr = '';

       urlStr = 'http://localhost:11100/rd/capture';

       getJSONCapture(urlStr,
          function (err, data) {
               if (err != null) {
                    alert('Something went wrong: ' + err);
               } else {
                    alert('Response:-' + String(data);
               }
          }
        );
   }

    var getJSONCapture = function (url, callback) {
       var xhr = new XMLHttpRequest();
       xhr.open('CAPTURE', url, true);
       xhr.responseType = 'text';     //json
       var InputXml = "<PidOptions> <Opts fCount\"1\" fType=\"0\" iCount=\"0\"
       pCount=\"0\" format=\"0\" pidVer=\"2.0\" timeout=\"20000\" otp=\"\"
       posh=\"UNKNOWN\" env=\"S\" wadh=\"\" /> <Demo></Demo> <CustOpts> <Param
       name=\"ValidationKey\" value=\"\" /> </CustOpts> </PidOptions>";

       xhr.onload = function () {
             var status = xhr.status;
                   if (status == 200) {
                         callback(null, xhr.response);
                   } else {
                         callback(status);
                   }
        };
       xhr.send(InputXml);
   };

</script>
```

Sample PID OPTION XML: -

```
<PidOptions>
        <Opts fCount="" fType="" iCount="" iType="" pCount="" pType="" format="" pidVer="" timeout="" otp="" posh="" env="" wadh="" />
        <Demo></Demo>
        <CustOpts>
                <Param name="PARAM_1" value="" />
        </CustOpts>
</PidOptions>
```

/*

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 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.

String env specifies that Authentication operation is going to performed on which environment (Staging(S), Pre-Production(PP),Production(P)).

Demo:

Element allows demographic data to be passed to form PID block as per authentication specification

CustOpts:

Allows vendor specific options to be passed. Param element may repeat.

*/

Sample PID DATA XML: -

```
<PidData xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
        <Resp fType="" iCount="" pCount="" errCode="" errInfo="" fCount="" ts=""
nmPoints="" qScore="" />
        <DeviceInfo/>
        <Skey ci="(CI_VALUE)">(ENCRYPTED_SKEY_VALUE)</Skey>
        <Hmac>(ENCRYPTED_HMAC_VALUE)</Hmac>
        <Data type="X">(ENCRYPTED_PID_BLOCK_VALUE)</Data>
</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 PidOptions→CustOpts→Param.

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
*/
```