PSA

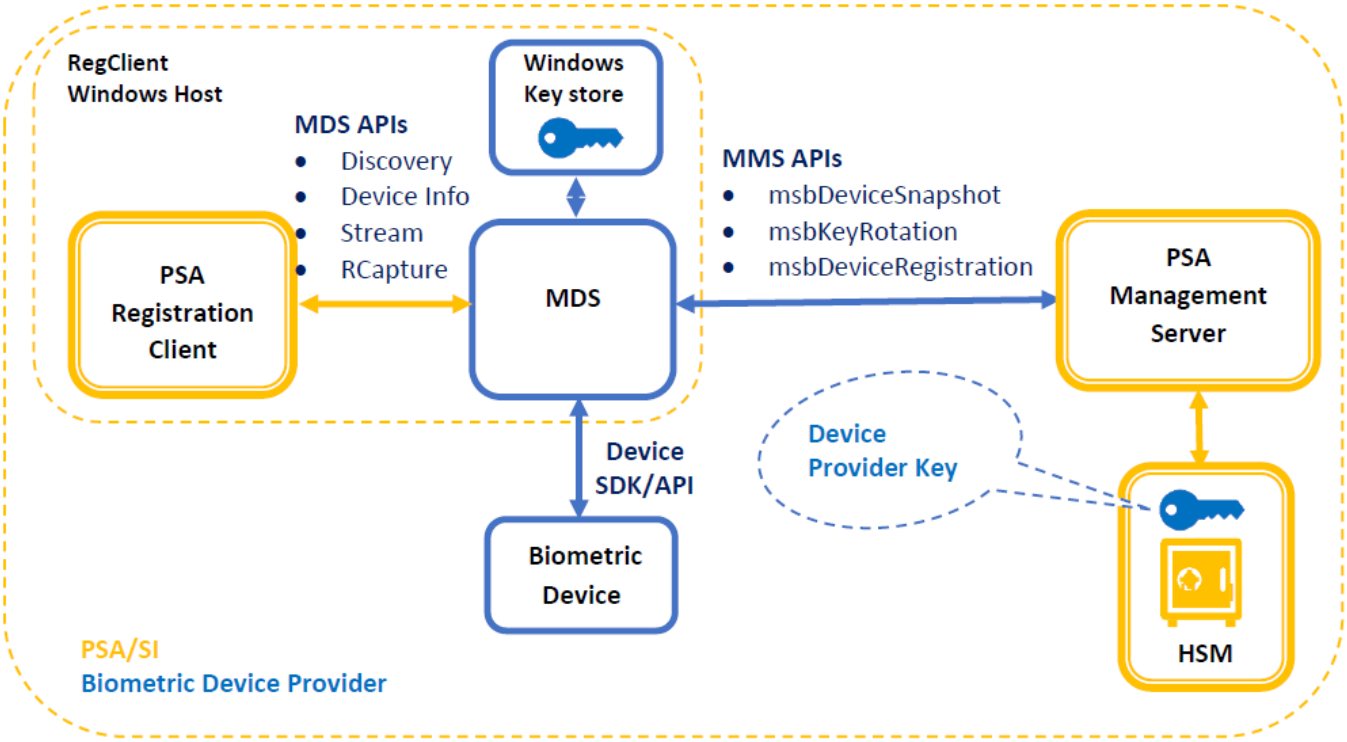
Management Server APIs

Table of Contents

Introduction

RegClient interacts with biometric device through MOSIP Device Service (MDS) provided by biometric device vendor. MDS should adhere to the specifications defined by MOSIP at https://docs.mosip.io/platform/biometrics/mosip-device-service-specification

MDS should interact with Management server to register and to consistently maintain security keys and time sync. This document describes PSA Management server APIs to enable device MDS to register, rotate keys and perform time sync with server.



Management server API expects below JWT (JSON Web Token) format for all the communication with URL ***https://<Public IP or Domain Name>/mms/MSBDeviceRequest/***

Header

```
{
"alg":"RS256",
"typ":"jwt",
"x5c":["MDS vendor certificate, will be used for signature verification. MMS will use first certificate from the array for signature verification."]
}
```

Payload

```
Request/Response specific JSON string
```

Signature

MDS to use self-signed or CA issued certificate for signature. MDS vendor to share its root x509 certificate with SI (System Integrator) to establish trust chain at server. Purpose is to avoid communication from unauthorized MDS/devices.

```
Signature of (base64UrlEncode(header) + "." + base64UrlEncode(payload))
```

https Request/Response body format:

```
{
“Request/Response” :
“base64urlencoded(header).base64urlencoded(payload).base64urlencoded(Signature)”
}
```

1. msbDeviceSnapshot

MDS need to use **msbDeviceSnapshot** request for time-sync and other device status details like mandatory key rotation, registration required etc. MDS is expected to request device snapshot again before **snapshotValidity** expires. Snapshot validity will be based on policy driven interval. Management server may de-register devices that do not time-sync or reconnect within policy driven interval. MDS should verify snapshotValidity before accepting any requests like CAPTURE or RCAPTURE from client.

1.1. Request payload

```
{
        "uniqueTransactionId" : "Unique Transaction ID",
        "timeStamp" : "ISO format datetime with time zone",
        "serialNumber" : "Serial Number of Secure Biometric Device",
        "deviceCodeUUID" : "A unique code given by MOSIP after successful registration.
Empty before registration.",
        "modelID" : "Model ID of Device",
        “deviceProviderId” : “Device provider id”,
        "bioType" : "Finger or Iris or Face",
        "certificationLevel" : "L0 or L1 or L2",
        "devicePurpose" : "Auth or Registration",
         "env": "Staging, Developer, Pre-Production or Production",
        "deviceServiceVersion": "MDS version",
         "specVersion": "MDS spec version",
        “hostos” : ”Windows/Android/Linux”
}
```

https POST

```
 {
   "msbDeviceSnapshot":
"base64urlencoded(header).base64urlencoded(payload).base64urlencoded(signature of
payload)"
 }
```

1.2. Response payload

```
{
        "uniqueTransactionId" : "Unique Transaction ID",
        "timeStamp" : "ISO format datetime with time zone",
        "serialNumber" : "Serial Number of Secure Biometric Device",
        "deviceCodeUUID" : "A unique code given by MOSIP after successful registration.
Empty before registration.",
        "modelID" : "Model ID of Device",
        “deviceProviderId” : “Device provider id”,
        "bioType" : "Finger or Iris or Face",
        "certificationLevel" : "L0 or L1 or L2",
        "devicePurpose" : "Auth or Registration",
         "env": "Staging, Developer, Pre-Production or Production",

        "serverTimeStamp" : "ISO format datetime with time zone of server",
        "isWhiteListed" : "true or false",
        "isDeviceValid" : "true or false",
        "isDeviceRegistered" : "true or false",
        "deviceServiceUpdateRequired" : "true or false",
        "deviceServiceDownloadURL" : "MDS download URL",
        "mandatoryKeyRotation" : "true or false",
        "keyExpiryTimeStamp" : "ISO format datetime with time zone of key expiry",
        "encryptionCertThumbprint" : "MOSIP encryption key Thumbprint",
         “snapshotValidity” : “ISO format datetime with time zone of server",
         “bannerMessage” : “Banner Message”,
         "responseCode" : "0",
         "description" : "Error Description"
}
```

https Response body

```
 {
   "msbDeviceSnapshot":
"base64urlencoded(header).base64urlencoded(payload).base64urlencoded(signature of
payload)"
 }
```

2. msbEncryptionCert

MDS to compare X509 certificate thumbprint of the current encryption certificate with encryptionCertThumbprint from msbDeviceSnapshot response and send msbEncryptionCert request to get latest encryption certificate.

2.1. Request payload

```
{
        "uniqueTransactionId" : "Unique Transaction ID",
        "timeStamp" : "ISO format datetime with time zone",
        "serialNumber" : "Serial Number of Secure Biometric Device",
        "deviceCodeUUID" : "A unique code given by MOSIP after successful registration.
Empty before registration.",
        "modelID" : "Model ID of Device",
        “deviceProviderId” : “Device provider id”,
        "bioType" : "Finger or Iris or Face",
        "certificationLevel" : "L0 or L1 or L2",
        "devicePurpose" : "Auth or Registration",
         "env": "Staging, Developer, Pre-Production or Production",
        "deviceServiceVersion": "MDS version",
         "specVersion": "MDS spec version",
        “hostos” : ”Windows/Android/Linux”

}
```

https POST

```
 {
   "msbEncryptionCert":
"base64urlencoded(header).base64urlencoded(payload).base64urlencoded(signature of
payload)"
 }
```

2.2. Response payload

```
{
        "uniqueTransactionId" : "Unique Transaction ID",
        "timeStamp" : "ISO format datetime with time zone",
        "serialNumber" : "Serial Number of Secure Biometric Device",
        "deviceCodeUUID" : "A unique code given by MOSIP after successful registration.
Empty before registration.",
        "modelID" : "Model ID of Device",
        “deviceProviderId” : “Device provider id”,
        "bioType" : "Finger or Iris or Face",
        "certificationLevel" : "L0 or L1 or L2",
        "devicePurpose" : "Auth or Registration",
         "env": "Staging, Developer, Pre-Production or Production",

        "encryptionCertData" : "Encryption Certificate certificate provided by MOSIP",
        "responseCode" : "0",
        "description" : "Error Description"
}
```

https Response body

```
{
   "msbEncryptionCert":
"base64urlencoded(header).base64urlencoded(payload).base64urlencoded(signature of
payload)"
 }
```

3. msbKeyRotation

MDS to request msbKeyRotation when mandatoryKeyRotation is true in msbDeviceSnapshot response or well before previous key expires.

3.1. Request payload

```
{
        "uniqueTransactionId" : "Unique Transaction ID",
        "timeStamp" : "ISO format datetime with time zone",
        "serialNumber" : "Serial Number of Secure Biometric Device",
        "deviceCodeUUID" : "A unique code given by MOSIP after successful registration.
Empty before registration.",
        "modelID" : "Model ID of Device",
        “deviceProviderId” : “Device provider id”,
        "bioType" : "Finger or Iris or Face",
        "certificationLevel" : "L0 or L1 or L2",
        "devicePurpose" : "Auth or Registration",
         "env": "Staging, Developer, Pre-Production or Production",
        "deviceServiceVersion": "MDS version",
         "specVersion": "MDS spec version",
        “hostos” : ”Windows/Android/Linux”,
        "csrData" : "Base64 encoded CSR with RSA2048 publickey of device"
}
```

https POST:

```
 {
   "msbKeyRotation":
"base64urlencoded(header).base64urlencoded(payload).base64urlencoded(signature)"
 }
```

3.2. Response payload

```
{
          "uniqueTransactionId" : "Unique Transaction ID",
          "timeStamp" : "ISO format datetime with time zone",
          "serialNumber" : "Serial Number of Secure Biometric Device",
          "deviceCodeUUID" : "A unique code given by MOSIP after successful registration.
Empty before registration.",
          "modelID" : "Model ID of Device",
          “deviceProviderId” : “Device provider id”,
          "bioType" : "Finger or Iris or Face",
          "certificationLevel" : "L0 or L1 or L2",
          "devicePurpose" : "Auth or Registration",
           "env": "Staging, Developer, Pre-Production or Production",

          "signedDeviceCertificate": "Base64 encoded X509 certificate of device Key",
          "keyExpiryTimeStamp" : "ISO format datetime with time zone of key expiry",
          "responseCode": "0",
         "description": "Error Description"
}
```

https Response body

```
 {
   "msbKeyRotation":
"base64urlencoded(header).base64urlencoded(payload).base64urlencoded(signature of
payload)"
 }
```

4. msbDeviceRegistration

MDS to request msbDeviceRegistration when isDeviceRegistered is false in msbDeviceSnapshot response.

4.1. Request payload

```
{
        "uniqueTransactionId" : "Unique Transaction ID",
        "timeStamp" : "ISO format datetime with time zone",
        "serialNumber" : "Serial Number of Secure Biometric Device",
        "deviceCodeUUID" : "A unique code given by MOSIP after successful registration.
Empty before registration.",
        "modelID" : "Model ID of Device",
        “deviceProviderId” : “Device provider id”,
        "bioType" : "Finger or Iris or Face",
        "certificationLevel" : "L0 or L1 or L2",
        "devicePurpose" : "Auth or Registration",
         "env": "Staging, Developer, Pre-Production or Production",
        "deviceServiceVersion": "MDS version",
         "specVersion": "MDS spec version",
        “hostos” : ”Windows/Android/Linux”,

        “deviceRegistrationRequest” : “MOSIP registration request JSON string (without
“device provider key” signatures) with signed deviceInfo and signed digitalId as defined at
https://docs.mosip.io/platform/biometrics/mosip-device-service-specification#device-
registration-request “
}
```

https POST

```
 {
   "msbDeviceRegistration":
"base64urlencoded(header).base64urlencoded(payload).base64urlencoded(signature of
payload)"
 }
```

4.2. Response payload

```
{
          "uniqueTransactionId" : "Unique Transaction ID",
          "timeStamp" : "ISO format datetime with time zone",
          "serialNumber" : "Serial Number of Secure Biometric Device",
          "modelID" : "Model ID of Device",
          “deviceProviderId” : “Device provider id”,
          "bioType" : "Finger or Iris or Face",
          "certificationLevel" : "L0 or L1 or L2",
          "devicePurpose" : "Auth or Registration",
           "env": "Staging, Developer, Pre-Production or Production",

          “deviceRegistrationResponse” : “Registration response received as defined at
https://docs.mosip.io/platform/biometrics/mosip-device-service-specification#device-registration-response “

        "responseCode": "0",
        "description": "Error Description"
}
```

https Response body

```
{
   "msbDeviceRegistration":
"base64urlencoded(header).base64urlencoded(payload).base64urlencoded(signature of payload)"
  }
```

5. Response Codes:

Response Code	Description
0	Success
-101	Invalid device code (UUID)
-102	Invalid device. Not whitelisted or validity expired
-103	Registration failed
-104	Encryption certificate not required for registration device
-105	Certificate signing failed
-1001	Unable to generate response

--- End of the document ---