

Mobile ID Reference Guide

Technical Documentation - Version 3.3

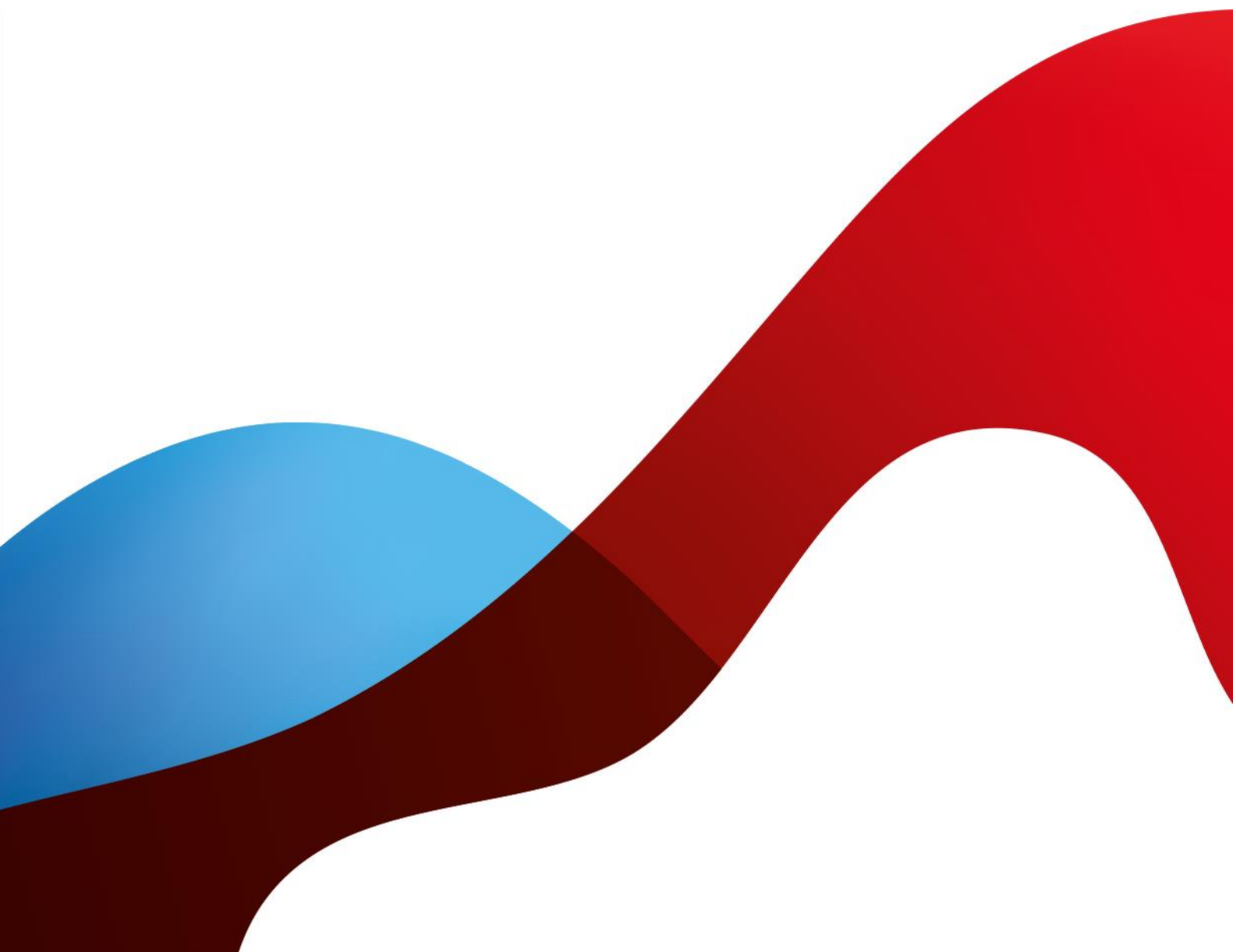


Table of contents

1	Introduction	4
1.1	Terms and Abbreviations	4
1.2	Mobile ID Signature Service (MSS)	5
1.2.1	Mobile ID SIM	5
1.2.1.1	eSIM Support	5
1.2.2	Mobile ID App	6
1.2.3	Authentication Flow	7
2	Application Provider Client Integration	8
2.1	Preconditions	8
2.2	Endpoint Address	8
2.2.1	SOAP Endpoint	8
2.2.2	REST Endpoint	8
2.3	Mutual Authentication	9
3	Mobile ID API	10
3.1	HTTP/1.1 Header	10
3.1.1	HTTP Request	10
3.1.2	HTTP Response	10
3.2	MSS Signature	11
3.2.1	Signature Profiles	11
3.2.1.1	User Scenario Examples (Signature Profile Handling)	12
3.2.1.2	Signature Messaging Mode	13
3.2.2	Synchronous MSS Signature	14
3.2.2.1	Synchronous MSS Signature Request	15
3.2.2.2	Synchronous MSS Signature Response	16
3.2.2.3	Fault Response	17
3.2.3	Asynchronous MSS Signature	18
3.2.3.1	Asynchronous MSS Signature Request	19
3.2.3.2	Asynchronous MSS Signature Response	20
3.2.4	Additional Services (AS)	21
3.2.4.1	User Language	21
3.2.4.2	Geofencing	22
3.2.4.3	App to App - Mobile Only Authentication	28
3.3	MSS Status Query	32
3.3.1	MSS Status Query Request	32
3.3.2	MSS Status Query Response	33
3.4	MSS Receipt	34
3.4.1	Synchronous MSS Receipt	34
3.4.1.1	MSS Receipt Request	35
3.4.1.2	MSS Receipt Response	36
3.4.2	Encrypted MSS Receipts	36
3.5	MSS Profile Query	37
3.5.1	MSS Profile Query Request	38
3.5.1.1	MSS Profile Query Request Extensions	38
3.5.2	MSS Profile Query Response	39
4	Best Practices	41
4.1	MSS Signature	41
4.1.1	Signature Request	41
4.1.2	Signature Response	42
4.1.3	Signature Concurrency Control	43
4.2	Mobile ID Serial Number Validation	44
4.3	Timeout Value	44

4.4	Mobile ID FAQ	44
4.5	Mobile ID Service Health Check	45
4.6	Mobile ID Client Examples	45
5	Auto Activation	46
5.1	Introduction	46
5.2	How to implement this feature	46
5.3	User Perspective	47
6	Status and Fault Codes	48
6.1	Overview	48
6.2	Testing Status and Fault Codes	49
6.2.1	Test-MSISDN Overview	49
7	Root CA Certificates (Trust Anchor)	50
7.1	Mobile ID X509 Server Certificate	50
7.2	Mobile ID User X509 Certificate	50
7.2.1	IMPORTANT: New Swisscom Certificate Authority	50
8	Create X509 Client Certificates.....	51
8.1	OpenSSL	51
8.1.1	Generate Key & Create CSR.....	51
8.1.2	Self-Sign Certificate.....	51
8.1.3	Convert To PKCS#12.....	51
8.2	Java KeyTool	51
8.2.1	Generate Key & Export Certificate	51
8.2.2	Root CA Cert & Intermediate CA Cert Import	51
8.2.3	Useful Commands	51
9	Health Status Microservice	52

Mobile ID is a brand of Swisscom.
 Swisscom (Switzerland) Ltd
 Alte Tiefenastrasse 6
 CH-3050 Bern

1 Introduction

The purpose of this document is to provide technical documentation and guidelines about how to use the Swisscom Mobile ID Authentication API.

The Swisscom Mobile ID authentication solution protects access to your company data and applications with a comprehensive end-to-end solution for two-factor authentication (2FA). Mobile ID can be used in multiple processes: everything from the simple addition of a second factor to an existing login, to password-free two-factor authentication, online signatures, and geolocation. It is suitable for different system landscapes and meets strict regulatory requirements.

Please visit <https://mobileid.ch> for further information. Do not hesitate to [contact us](#) in case of any questions.

The latest version of this document is published on [GitHub](#).

1.1 Terms and Abbreviations

Term	Description
AP	Application Provider
AP_ID	Application Provider Identifier
DTBD	Data-To-Be-Displayed. Message that is displayed on the mobile phone (authentication context).
DTBS	Data-To-Be-Signed. Equal to DTBD. Data that will be signed with the Mobile ID signing key.
JSON	JavaScript Object Notation is a text-based open standard designed for human readable data interchange. Although derived from the JavaScript scripting language it is language independent. The JSON format is often used for serializing and transmitting structured data over a network connection, primarily between a server and a web application, as an alternative to XML.
LAN-I	Swisscom LAN-Interconnect Service. Also known as Enterprise WAN.
MID	Mobile ID platform providing the mobile signature service
MNO	Mobile Network Operator, also known as a wireless service provider, wireless carrier, cellular company, or mobile network carrier.
MSISDN	Number uniquely identifying a subscription in a GSM/UMTS mobile network, aka mobile phone number.
MSSP	Mobile Signature Service Provider, the Swisscom Mobile ID backend application.
OTA	Over-The-Air is a technology used to communicate with and manage a SIM card without being connected physically to the card.
RESTful	Representational State Transfer is a style of software architecture for distributed systems such as the World Wide Web. It is based on the existing design of HTTP/1.0. REST-style architectures consist of clients and servers. Clients initiate requests to servers; servers process requests and return appropriate responses.
SOAP	Simple Object Access Protocol (SOAP) is a protocol specification for exchanging structured information in the implementation of Web Services relying on Extensible Markup Language (XML)
WSDL	The Web Services Description Language (WSDL) is an XML-based language that is used for describing the functionality offered by a Web service.
X509	PKI and Digital Certificates
XML	Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable.

1.2 Mobile ID Signature Service (MSS)

Mobile ID is a cost-efficient, managed authentication service from Swisscom. The customer-facing API is based on open standard ETSI 102 204¹.

Mobile ID authentication is based on a secure hardware token that can either be your Mobile ID compliant SIM/eSIM or your mobile device running the Mobile ID App.

Therefore, a user account could have either the (e)SIM method, the App method or even both methods activated at the same time. However, the AP may select the preferred method and allow both methods or just either one (see section 3.2.1).

1.2.1 Mobile ID SIM

An Application Provider (AP) can request SIM Toolkit (STK) based authentication, hereinafter referred as "SIM method". To utilize the SIM method, the user must have a Mobile ID compliant SIM card or eSIM. Data exchange between the Mobile ID server and the STK application is done with SMS messages using data packets (PDUs), not visible to the end-user. The Mobile ID SIM Toolkit application runs on the SIM card environment and is compliant with all mobile devices.



SIM method key advantages:

- **Strong two-factor authentication**
 - 1st Factor: Physical SIM card or eSIM chip (Possession Factor)
 - 2nd Factor: Personal Mobile ID PIN (Knowledge Factor)
- **High level of security**
 - Tamper-proof key storage based on EAL5+ (ISO/IEC 15408) and Evaluation Level E3 (ITSEC document version 1.2) certified hardware
 - Authentication through separate and secure (encrypted) communication channel
- The Mobile ID STK application is pre-installed on the SIM card (or part of the eSIM Profile)
- Mobile ID SIM cards or eSIM Profiles are supported by most Swiss Mobile Network Operators

1.2.1.1 eSIM Support

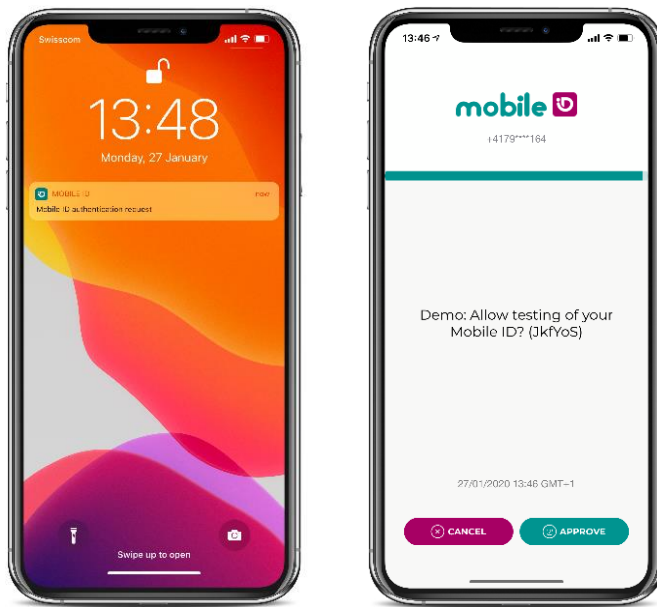
At the time of writing, eSIM Profiles with Mobile ID are available for Apple iPhone models with eSIM support.

¹ [ETSI TS 102 204 V1.1.4 \(2003-08\)](#)

1.2.2 Mobile ID App

An Application Provider (AP) can request Mobile App based authentication, hereinafter referred as "App method". To utilize the App method, the user must have the Mobile ID App installed on a compliant Android or iOS-based smartphone. The app can be downloaded from Google Play Store and Apple App Store.

The Mobile ID App activation can be done within the mobile app (in-app enrolment) or through the selfcare portal² (in latter case, the app must scan a QR code displayed on www.mobileid.ch).



App method key advantages:

- **Strong two-factor authentication**
 - 1st Factor: Smartphone (Possession Factor)
 - 2nd Factor: Passcode (Knowledge Factor) or Biometry (Inherence Factor)
- **High level of security**
 - Authentication through dedicated mobile application (authentication app)
 - Fast and secure (encrypted) communication
- The app is published and available in several countries of the European Union (EU)

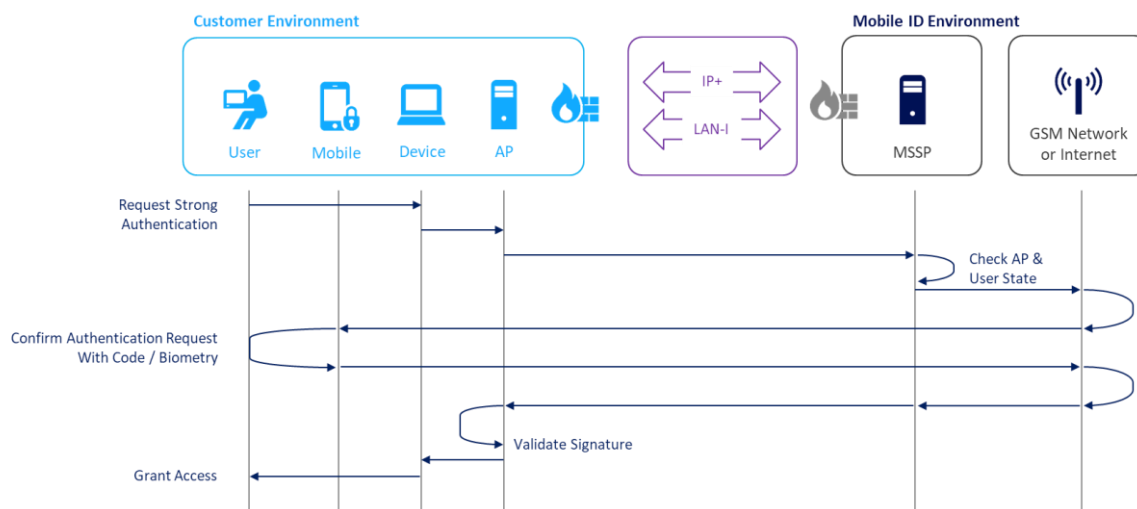
² <https://mobileid.ch>

1.2.3 Authentication Flow

Before going into more technical details, let's have a short look at the main scenario.

Strong Authentication:

The end-user wants to access a corporate application protected by Mobile ID strong authentication.



The main steps performed by the end-user and the mobile signature service are described below:

1. The end-user uses any application relying on Mobile ID for authentication, which sends a mobile signature request through the dedicated web interface (of the AP) including the personal MSISDN as input parameter to login.
2. The AP receives the end-user request, forms the contents to be signed (in accordance with the ETSI TS 102 204³ standard) and forwards the request to the MID service.
3. The MID platform receives the signature request and validates the AP in accordance with the service agreement.
4. The MID platform ensures that the end-user signature request is allowed and forwards the signature request to the end-user's mobile phone.
5. The end-user gets a message on his mobile phone to sign the mobile signature request. The End-User confirms the authentication request either by providing the Mobile ID PIN (SIM method) or Passcode/Biometry (App method).

After the AP has received a valid response, the end-user will be granted access to the corporate application.

³ [ETSI TS 102 204 V1.1.4 \(2003-08\)](#)

2 Application Provider Client Integration

2.1 Preconditions

Before using the Swisscom Mobile ID web service, some initial provisioning steps are required.

1. The Mobile ID customer (your company) has an agreement with Swisscom
 - a. The connectivity (Internet or LAN-I) between the AP and Mobile ID has been established. The AP's public source IP address (or range) must be whitelisted in the Swisscom Firewall.
 - b. The customer delivered the X509 client certificate to Swisscom (section 8).
2. The Mobile ID customer received from Swisscom an AP_ID (Application Provider Identifier) value and a DataToBeDisplayed (DTBD) Prefix value. The DTBD Prefix is an AP specific keyword that must be included as a prefix in every Mobile ID request's text message sent to a Mobile ID user (message displayed on the user's mobile phone). For example: "Bank ACME: ".

2.2 Endpoint Address

The Swisscom Mobile ID web service is accessible through LAN-I⁴ or Internet. If not otherwise specified use the following default access details.

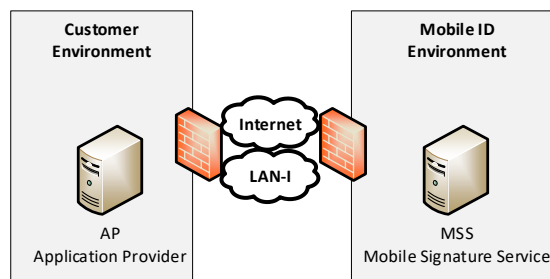
Base-URL

Internet

`https://mobileid.swisscom.com`

Swisscom LAN-I⁵

`https://195.65.233.218`



Service Endpoint

The Mobile ID customer can choose between SOAP or RESTful endpoints.

2.2.1 SOAP Endpoint

A description of this interface is available as WSDL file on GitHub: [mobileid.wsdl](https://github.com/swisscom/mobileid/blob/master/wSDL/mobileid.wsdl).

<code><Base-URL>/soap/services/MSS_SignaturePort</code>	MSS Signature, section 3.2
<code><Base-URL>/soap/services/MSS_StatusQueryPort</code>	MSS Status Query, section 3.3
<code><Base-URL>/soap/services/MSS_ReceiptPort</code>	MSS Receipt, section 3.4
<code><Base-URL>/soap/services/MSS_ProfilePort</code>	MSS Profile Query, section 3.5

2.2.2 REST Endpoint

A description of this interface is available as YAML file on GitHub: [mobileid.yaml](https://github.com/swisscom/mobileid/blob/master/YAML/mobileid.yaml).

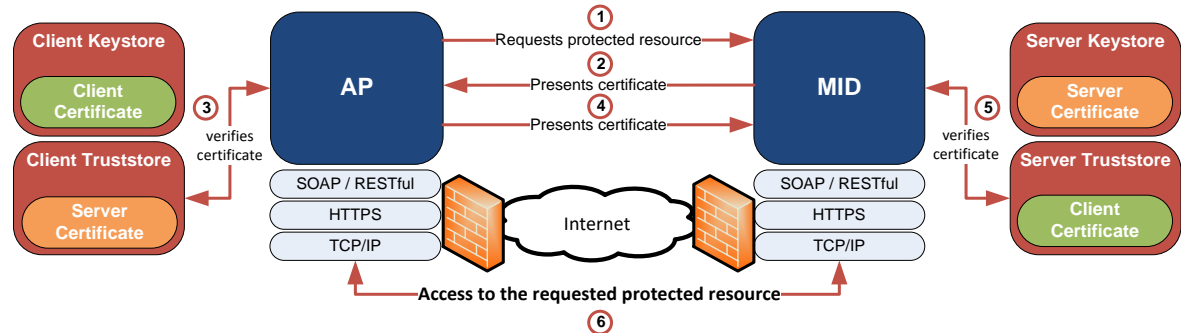
<code><Base-URL>/rest/service/sign</code>	MSS Signature, section 3.2
<code><Base-URL>/rest/service/status</code>	MSS Status Query, section 3.3
<code><Base-URL>/rest/service/receipt</code>	MSS Receipt, section 3.4
<code><Base-URL>/rest/service/profile</code>	MSS Profile Query, section 3.5

⁴ LAN-I (Wide Area Network) <https://www.swisscom.ch/en/business/enterprise/offer/wireline/enterprise-wan.html>

⁵ LAN-I works with IPs only and doesn't know any host names / domains

2.3 Mutual Authentication

A certificate-based mutual authentication when accessing the Mobile ID web service is highly recommended. When using certificate-based mutual authentication, the following actions occur:



1. The client AP requests access to a protected resource on MID.
2. The MID web server presents its server certificate to the client AP.
3. The client AP verifies the MID server certificate.
4. If successful, the client AP sends its client certificate to the MID server.
5. The MID server verifies the AP client credentials.
6. If successful, the MID server grants access to the protected resource requested by the client AP.

Important guidelines for the certificate-based mutual authentication:

- The client shall send only its end entity certificate. Authentication at the MID side does not consider any validation of a client certificate chain or restrictions of the root CA. The authentication is denied in case the client sends a bag with the full certificate chain.
- Enhanced Key Usage value of client certificates must contain Client Authentication (1.3.6.1.5.5.7.3.2). Section 8 contains examples on how to create self-signed certificates.
- It is critically important that all your requests to the Mobile ID service are coming from the server controlled by you. Never make requests to the service directly from the client side (e.g. a Mobile App or JavaScript), as this may compromise your credentials.
- To validate the chain of trust of the Mobile ID server certificate, it will be enough to add the *SwissSign Gold CA - G2⁶* root certificate to the client TrustStore. The intermediate CAs are returned by the MID server and may change. Refer to section 0 for more details.

⁶ Get the root certificate from <https://www.swissign.com/en/support/faq.html>

3 Mobile ID API

Mobile ID exposes a web service based on SOAP or RESTful (JSON). Refer to chapter 2.2 for a more detailed interface description (which includes a link to a WSDL or YAML file on GitHub, which describes the interface).

3.1 HTTP/1.1 Header

3.1.1 HTTP Request

You can POST signature requests via HTTP/1.1 using the SOAP or RESTful interface. The RESTful interface supports JavaScript Object Notation (JSON) as media type.

The header fields should be set as follows:

Header Field	SOAP	RESTful/JSON
Content-Type	text/xml;charset=UTF-8	application/json;charset=UTF-8
Accept	-	application/json

3.1.2 HTTP Response

If the request has succeeded, the Mobile ID server will respond with `HTTP/1.1 200`.

In case of an error while processing the request (including cases such as `USER_CANCEL`, `EXPIRED_TRANSACTION`, `UNKNOWN_CLIENT`, ... see section 6), the Mobile ID server will issue an `HTTP/1.1 500`⁷ response and include a message in the response containing a Fault element (see section) indicating the processing error. This is true for both SOAP and RESTful interfaces.

⁷ https://www.w3.org/TR/soap11/#_Toc478383529

3.2 MSS Signature

3.2.1 Signature Profiles

With every authentication request, an AP can select the SIM or App authentication method by selecting a specific `SignatureProfile`-value in the MSS Signature request.

This will give all the required flexibility for an AP to handle different use cases. For example, an AP may use an MSS Profile Query (chapter 3.5) request to silently check the user's authentication method capabilities before an MSS Signature request (chapter 3.2) is sent. In other scenarios, the AP may want to force a specific authentication method.

Signature Profile value	Description	AP Authorization ¹⁾
<code>http://mid.swisscom.ch/MID/v1/AuthProfile1</code> Deprecated. Should no longer be used.	By default, this signature profile is mapped to <code>http://mid.swisscom.ch/STK-LoA4</code> .	Any AP is authorized
<code>http://mid.swisscom.ch/Any-LoA4</code>	Mobile ID backend will either choose SIM- or App authentication, based on the user's authentication method capabilities: The SIM method is always preferred. The App method is only selected if the App is the only active method for that particular user. As shown in the table at section 3.2.1.1, the response will contain a signature profile value to indicate to the client what authentication method was chosen.	By default, an AP is not authorized to use this signature profile. Please contact Swisscom if you intend to use this signature profile.
<code>http://mid.swisscom.ch/STK-LoA4</code>	Force SIM authentication method.	By default, an AP is not authorized to use this signature profile. Please ask Swisscom if you intend to use this signature profile.
<code>http://mid.swisscom.ch/Device-LoA4</code>	Force App authentication method.	By default, an AP is not authorized to use this signature profile. Please ask Swisscom if you intend to use this signature profile.

¹⁾ In case an AP is not authorized to use a specific signature profile, the request will be rejected with a fault response 109/UNSUPPORTED_PROFILE and more verbose details are provided in the fault detail element.













3.2.1.1 User Scenario Examples (Signature Profile Handling)

The table below lists different user scenarios and how MSSP (the Mobile ID backend) will respond to a specific MSS Signature's signature profile value, based on the following conditions.

Note, in the examples below, the AP has been configured as follows:

- AP is authorized to use all signature profiles
- AP has been configured with a signature profile mapping, so that incoming signature profile `http://mid.swisscom.ch/MID/v1/AuthProfile1` is being mapped to `http://mid.swisscom.ch/Any-LoA4`

This is just a *very specific* example and it may not be applicable to other APs. For example, some APs may not be authorized to use all signature profiles. Other APs may not have a signature profile mapping configured.

MID User	SIM	App	Request's Signature Profile	Response's Signature Profile
  	✗	✗	<code>http://mid.swisscom.ch/MID/v1/AuthProfile1</code>	n/a [Fault 105 / UNKNOWN_CLIENT]
			<code>http://mid.swisscom.ch/Any-LoA4</code>	n/a [Fault 105 / UNKNOWN_CLIENT]
			<code>http://mid.swisscom.ch/STK-LoA4</code>	n/a [Fault 105 / UNKNOWN_CLIENT]
			<code>http://mid.swisscom.ch/Device-LoA4</code>	n/a [Fault 105 / UNKNOWN_CLIENT]
  	✓	✗	<code>http://mid.swisscom.ch/MID/v1/AuthProfile1</code>	<code>http://mid.swisscom.ch/STK-LoA4</code>
			<code>http://mid.swisscom.ch/Any-LoA4</code>	<code>http://mid.swisscom.ch/STK-LoA4</code>
			<code>http://mid.swisscom.ch/STK-LoA4</code>	<code>http://mid.swisscom.ch/STK-LoA4</code>
			<code>http://mid.swisscom.ch/Device-LoA4</code>	n/a [Fault 109 / UNSUPPORTED_PROFILE]
  	✗	✓	<code>http://mid.swisscom.ch/MID/v1/AuthProfile1</code>	<code>http://mid.swisscom.ch/Device-LoA4</code>
			<code>http://mid.swisscom.ch/Any-LoA4</code>	<code>http://mid.swisscom.ch/Device-LoA4</code>
			<code>http://mid.swisscom.ch/STK-LoA4</code>	n/a [Fault 109 / UNSUPPORTED_PROFILE]
			<code>http://mid.swisscom.ch/Device-LoA4</code>	<code>http://mid.swisscom.ch/Device-LoA4</code>
  	✓	✓	<code>http://mid.swisscom.ch/MID/v1/AuthProfile1</code>	<code>http://mid.swisscom.ch/STK-LoA4</code>
			<code>http://mid.swisscom.ch/Any-LoA4</code>	<code>http://mid.swisscom.ch/STK-LoA4</code>
			<code>http://mid.swisscom.ch/STK-LoA4</code>	<code>http://mid.swisscom.ch/STK-LoA4</code>
			<code>http://mid.swisscom.ch/Device-LoA4</code>	<code>http://mid.swisscom.ch/Device-LoA4</code>

Note that an AP can use the MSS Profile Query to find the user capabilities, e.g. what authentication method a specific user will support (see section 3.5).

3.2.1.2 Signature Messaging Mode

The ETSI 102 204⁸ standard has defined the MSS Signature and Swisscom supports both synchronous and asynchronous (client-server) modes⁹.

The MSS_Signature method is used to submit the mobile signature request message (MSS_SignatureReq). The result is provided within the signature response message (MSS_SignatureResp).

The Mobile ID customer (AP) can decide to call either synchronous or asynchronous signature requests. There are different aspects to consider:

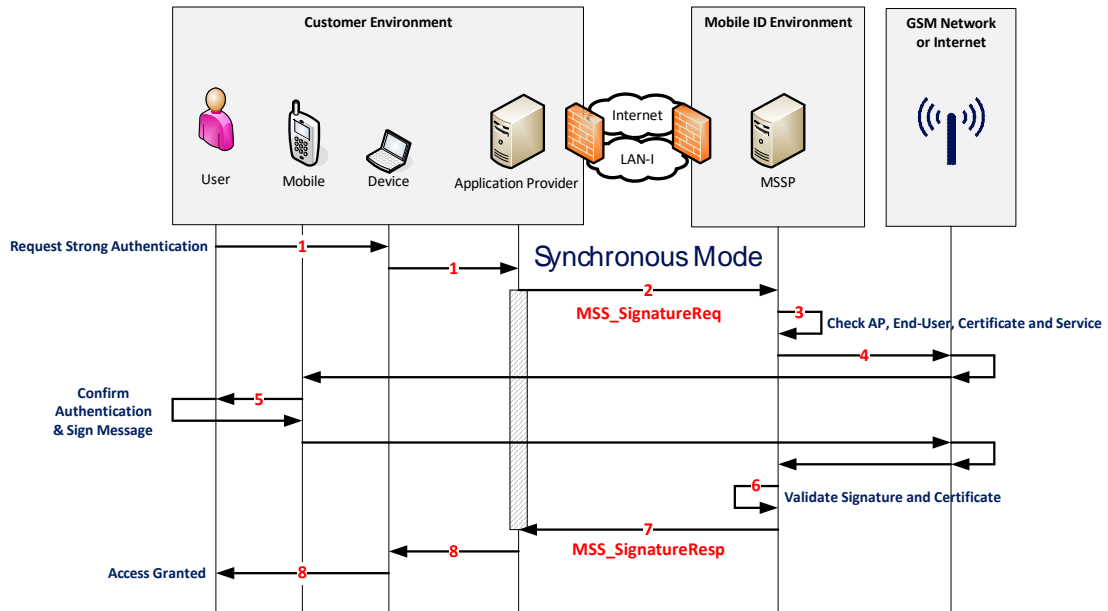
- With the *synchronous* mode, the signature response is immediately processed by the AP after it has been made available by the Mobile ID Service, the overall authentication transaction will be faster. If an Application Provider intends to invoke many signature transactions of different users in parallel, it may require more memory because each thread is waiting for its completion.
- With the *asynchronous client-server* mode, the Application Provider needs to implement a polling mechanism (query the transaction status every x seconds until the signature is has been made available by the Mobile ID Service).

⁸ [ETSI TS 102 204 V1.1.4 \(2003-08\)](#)

⁹ Swisscom Mobile ID does not support the asynchronous server-server mode where the server does a call-back to the client

3.2.2 Synchronous MSS Signature

In the synchronous mode, the AP initiates the signature request by calling MSS_SignatureReq, which is then blocked until the signature has been received or the signing times out occurs as depicted below.



1. End-User uses an application that sends an authentication request.
2. AP receives the request and sends an MSS_SignatureReq message (MessagingMode="synch") to MSSP.
3. MID backend validates the AP credentials.
4. MID sends a signature request to the application on the mobile device (either SIM or App).
5. The mobile application (STK applet or Mobile App) prompts the End-User to enter the Mobile ID secret (PIN, passcode, biometry). End-User confirms with the correct secret and the application digitally signs the request and sends the signature back to the Mobile ID backend (MSSP).
6. Mobile ID backend (MSSP) receives the signature response.
7. Mobile ID backend sends a complete response (signature + certificate containing the end-user's public key) to the AP.
8. Depending on the response of MID the AP may grant or deny access to the End-User.

3.2.2.1 Synchronous MSS Signature Request

SOAP/XML
<pre> <soapenv:Envelope soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"> <soapenv:Body> <MSS_Signature> <mss:MSS_SignatureReq MinorVersion="1" MajorVersion="1" MessagingMode="synch" TimeOut="80" xmlns:mss="http://uri.etsi.org/TS102204/v1.1.2#" xmlns:fi="http://mss.ficom.fi/TS102204/v1.0.0#"> <mss:AP_Info AP_ID="yourAP_ID" AP_PWD="yourAP_PWD" AP_TransID="REF0101120000" Instant="2015-01-01T12:00:00.000+01:00"/> <mss:MSSP_Info> <mss:MSSP_ID> <mss:URI>http://mid.swisscom.ch/</mss:URI> </mss:MSSP_ID> </mss:MSSP_Info> <mss:MobileUser> <mss:MSISDN>MOBILE_NUMBER</mss:MSISDN> </mss:MobileUser> <mss:DataToBeSigned MimeType="text/plain" Encoding="UTF-8">TEXT_TO_BE_SIGNED</mss:DataToBeSigned> <mss:SignatureProfile> <mss:mssURI>SIGNATURE_PROFILE</mss:mssURI> </mss:SignatureProfile> <mss:AdditionalServices> <mss:Service> <mss:Description> <mss:mssURI>http://mss.ficom.fi/TS102204/v1.0.0#userLang</mss:mssURI> </mss:Description> <fi:UserLang>LANGUAGE</fi:UserLang> </mss:Service> </mss:AdditionalServices> </mss:MSS_SignatureReq> </MSS_Signature> </soapenv:Body> </soapenv:Envelope> </pre>
REST/JSON
<pre> { "MSS_SignatureReq": { "AP_Info": { "AP_ID": "yourAP_ID", "AP_PWD": "yourAP_PWD", "AP_TransID": "REF0101120000", "Instant": "2015-01-01T12:00:00.000+01:00" }, "AdditionalServices": [{ "Description": "http://mss.ficom.fi/TS102204/v1.0.0#userLang", "UserLang": { "Value": "LANGUAGE" } }], "DataToBeSigned": { "Data": "TEXT_TO_BE_SIGNED", "Encoding": "UTF-8", "MimeType": "text/plain" }, "MSSP_Info": { "MSSP_ID": { "URI": "http://mid.swisscom.ch/" } }, "MajorVersion": "1", "MessagingMode": "synch", "MinorVersion": "2", "MobileUser": { "MSISDN": "MOBILE_NUMBER" }, "SignatureProfile": "SIGNATURE_PROFILE", "TimeOut": "80" } } </pre>

Note that `MinorVersion` value must be set to “2” in case of a REST/JSON request message.

3.2.2.2 Synchronous MSS Signature Response

SOAP/XML
<pre> <soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"> <soapenv:Body> <MSS_SignatureResponse> <mss:MSS_SignatureResp MajorVersion="1" MinorVersion="1" MSSP_TransID="h4dhx" xmlns:mss="http://uri.etsi.org/TS102204/v1.1.2#" xmlns:fi="http://mss.ficom.fi/TS102204/v1.0.0#"> <mss:AP_Info AP_ID="yourAP_ID" AP_PWD="yourAP_PWD" AP_TransID="REF0101120000" Instant="2015-01-01T12:00:00.000+01:00"/> <mss:MSSP_Info Instant="2015-01-01T12:00:00.100+01:00"> <mss:MSSP_ID> <mss:URI>http://mid.swisscom.ch/</mss:URI> </mss:MSSP_ID> </mss:MSSP_Info> <mss:MobileUser> <mss:MSISDN>MOBILE_NUMBER</mss:MSISDN> </mss:MobileUser> <mss:MSS_Signature> <mss:Base64Signature>MIIIdwYJKoZIhvc...</mss:Base64Signature> </mss:MSS_Signature> <mss:SignatureProfile> <mss:mssURI>SIGNATURE_PROFILE</mss:mssURI> </mss:SignatureProfile> <mss:Status> <mss:StatusCode Value="500"/> <mss:StatusMessage>SIGNATURE</mss:StatusMessage> </mss:Status> </mss:MSS_SignatureResp> </MSS_SignatureResponse> </soapenv:Body> </soapenv:Envelope> </pre>
REST/JSON
<pre> { "MSS_SignatureResp": { "AP_Info": { "AP_ID": "yourAP_ID", "AP_TransID": "REF0101120000", "Instant": "2015-01-01T11:00:00.000Z" }, "MSSP_Info": { "Instant": "2015-01-01T11:00:00.100Z", "MSSP_ID": { "URI": "http://mid.swisscom.ch/" } }, "MSSP_TransID": "h44okl", "MSS_Signature": { "Base64Signature": "MIIIdwYJKoZIhvc..." }, "MajorVersion": "1", "MinorVersion": "1", "MobileUser": { "MSISDN": "MOBILE_NUMBER" }, "SignatureProfile": "SIGNATURE_PROFILE", "Status": { "StatusCode": { "Value": "500" }, "StatusMessage": "SIGNATURE" } } } </pre>

The 'Base64Signature' content is a base 64 encoded CMS¹⁰ (which is an extension of PKCS#7) signature object. It contains the DTBD message that has been signed by the SIM- or mobile application on the mobile device. In addition, it includes the mobile user certificate and all related intermediate certificates. Therefore, the AP will always be able to fully validate the signature response.

Note that the response contains the signature profile value to indicate what authentication method was chosen, which is helpful in case the request signature profile was `http://mid.swisscom.ch/Any-LoA4`.

¹⁰ <https://tools.ietf.org/html/rfc5652>

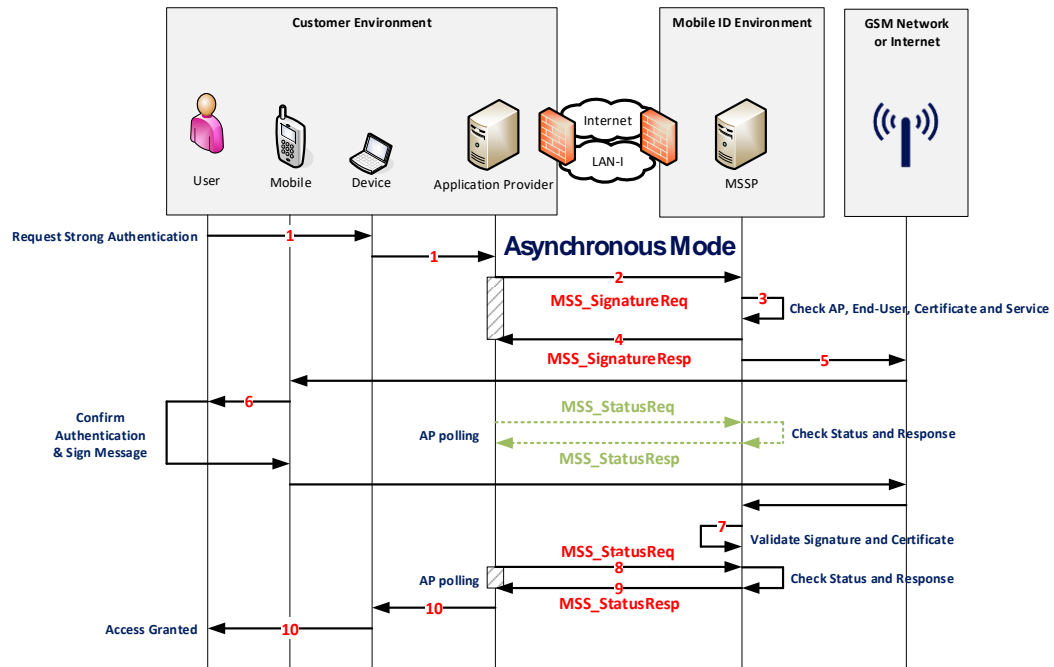
3.2.2.3 Fault Response

This is an example fault response in case of an illegal MSISDN value. Each fault response contains a (sub-)code value, reason text and detail text. Refer to section 6 for a list of status and error codes.

SOAP/XML
<pre> <soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"> <soapenv:Body> <soapenv:Fault> <soapenv:Code> <soapenv:Value>soapenv:Sender</soapenv:Value> <soapenv:Subcode xmlns:mss="http://uri.etsi.org/TS102204/v1.1.2#" xmlns:fi="http://mss.ficom.fi/TS102204/v1.0.0#"> <soapenv:Value>mss:_101</soapenv:Value> </soapenv:Subcode> </soapenv:Code> <soapenv:Reason> <soapenv:Text xml:lang="en">WRONG_PARAM</soapenv:Text> </soapenv:Reason> <soapenv:Detail> <ns1:detail xmlns:ns1="http://kiuru.methics.fi/mssp">Illegal msisdn</ns1:detail> </soapenv:Detail> </soapenv:Fault> </soapenv:Body> </soapenv:Envelope> </pre>
REST/JSON
<pre> { "Fault": { "Code": { "SubCode": { "Value": "_101", "ValueNs": "http://uri.etsi.org/TS102204/v1.1.2#" }, "Value": "Sender", "ValueNs": "http://www.w3.org/2003/05/soap-envelope" }, "Detail": "Illegal msisdn", "Reason": "WRONG_PARAM" } } </pre>

3.2.3 Asynchronous MSS Signature

In the asynchronous mode, the AP initiates the signature request by calling MSS_Signature method and an acknowledgment response is sent back immediately by the MID to AP. The AP client then starts polling using MSS_StatusQuery service to receive the response as depicted below.



1. End-User uses an application that sends an authentication request.
2. AP receives the request and sends an asynchronous MSS_SignatureReq request message to MSSP.
3. MID backend validates the AP credentials.
4. MID responds to AP with MSS_SignatureResp⁷ message (acknowledgment).
5. MID sends a signature request to the application on the mobile device (either SIM or App).
6. The mobile application (STK applet or Mobile App) prompts the End-User to enter the Mobile ID secret (PIN, passcode, biometry). End-User confirms with the correct secret and the application digitally signs the request and sends the signature back to the Mobile ID backend (MSSP).

Meanwhile the AP sends MSS_StatusReq requests to MID. The MID replies with MSS_StatusResp¹¹ (status code "504 OUTSTANDING_TRANSACTION", which means that the AP will need to call again the status method).

7. Mobile ID backend (MSSP) receives the signature response.
8. AP sends next MSS_StatusReq request message to MID.
9. MID sends a complete response (signature + certificate containing the end-user's public key) to the AP as MSS_StatusResp Response.
10. Depending on the response of MID, the AP may grant or deny access to the End-User.

¹¹ In case of an error, MID may replies with a SoapFault containing an error code as described in section 6.

3.2.3.1 Asynchronous MSS Signature Request

In **pink** the differences compared to the sync mode. This value isn't the same for SOAP and REST.

SOAP/XML
<pre> <soapenv:Envelope soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" > <soapenv:Body> <MSS_Signature> <mss:MSS_SignatureReq MinorVersion="1" MajorVersion="1" MessagingMode="asynchClientServer" Timeout="80" xmlns:mss="http://uri.etsi.org/TS102204/v1.1.2#" xmlns:fi="http://mss.ficom.fi/TS102204/v1.0.0#" > <mss:AP_Info AP_ID="yourAP_ID" AP_PWD="yourAP_PWD" AP_TransID="REF0101120000" Instant="2015-01-01T12:00:00.000+01:00"/> <mss:MSSP_Info> <mss:MSSP_ID> <mss:URI>http://mid.swisscom.ch/</mss:URI> </mss:MSSP_ID> </mss:MSSP_Info> <mss:MobileUser> <mss:MSISDN>MOBILE_NUMBER</mss:MSISDN> </mss:MobileUser> <mss:DataToBeSigned MimeType="text/plain" Encoding="UTF-8">TEXT_TO_BE_SIGNED</mss:DataToBeSigned> <mss:SignatureProfile> <mss:mssURI>SIGNATURE_PROFILE</mss:mssURI> </mss:SignatureProfile> <mss:AdditionalServices> <mss:Service> <mss:Description> <mss:mssURI>http://mss.ficom.fi/TS102204/v1.0.0#userLang</mss:mssURI> </mss:Description> <fi:UserLang>LANGUAGE</fi:UserLang> </mss:Service> </mss:AdditionalServices> </mss:MSS_SignatureReq> </MSS_Signature> </soapenv:Body> </soapenv:Envelope> </pre>
REST/JSON
<pre> { "MSS_SignatureReq": { "AP_Info": { "AP_ID": "yourAP_ID", "AP_PWD": "yourAP_PWD", "AP_TransID": "REF0101120000", "Instant": "2015-01-01T12:00:00.000+01:00" }, "AdditionalServices": [{ "Description": "http://mss.ficom.fi/TS102204/v1.0.0#userLang", "UserLang": { "Value": "LANGUAGE" } }], "DataToBeSigned": { "Data": "TEXT_TO_BE_SIGNED", "Encoding": "UTF-8", "MimeType": "text/plain" }, "MSSP_Info": { "MSSP_ID": { "URI": "http://mid.swisscom.ch/" } }, "MajorVersion": "1", "MessagingMode": "asynch", "MinorVersion": "2", "MobileUser": { "MSISDN": "MOBILE_NUMBER" }, "SignatureProfile": "SIGNATURE_PROFILE", "Timeout": "80" } } </pre>

Note that **MinorVersion** value must be set to “2” in case of a REST/JSON request message.

3.2.3.2 Asynchronous MSS Signature Response

SOAP/XML
<pre> <soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"> <soapenv:Body> <MSS_SignatureResponse> <mss:MSS_SignatureResp MajorVersion="1" MinorVersion="1" MSSP_TransID="h4iof" xmlns:mss="http://uri.etsi.org/TS102204/v1.1.2#" xmlns:fi="http://mss.ficom.fi/TS102204/v1.0.0#"> <mss:AP_Info AP_ID="yourAP_ID" AP_PWD="yourAP_PWD" AP_TransID="REF0101120000" Instant="2015-01-01T12:00:00.000+01:00"/> <mss:MSSP_Info Instant="2015-01-01T12:00:00.100+01:00"> <mss:MSSP_ID> <mss:URI>http://mid.swisscom.ch/</mss:URI> </mss:MSSP_ID> </mss:MSSP_Info> <mss:MobileUser> <mss:MSISDN>MOBILE_NUMBER</mss:MSISDN> </mss:MobileUser> <mss:SignatureProfile> <mss:mssURI>SIGNATURE_PROFILE</mss:mssURI> </mss:SignatureProfile> <mss:Status> <mss:StatusCode Value="100"/> <mss:StatusMessage>REQUEST_OK</mss:StatusMessage> </mss:Status> </mss:MSS_SignatureResp> </MSS_SignatureResponse> </soapenv:Body> </soapenv:Envelope> </pre>
REST/JSON
<pre> { "MSS_SignatureResp": { "AP_Info": { "AP_ID": "yourAP_ID", "AP_TransID": "REF0101120000", "Instant": "2015-01-01T11:00:00.000Z" }, "MSSP_Info": { "Instant": "2015-01-01T11:00:00.100Z", "MSSP_ID": { "URI": "http://mid.swisscom.ch/" } }, "MSSP_TransID": "h4iof", "MajorVersion": "1", "MinorVersion": "2", "MobileUser": { "MSISDN": "MOBILE_NUMBER" }, "SignatureProfile": "SIGNATURE_PROFILE", "Status": { "StatusCode": { "Value": "100" }, "StatusMessage": "REQUEST_OK" } } } </pre>

Note that the response contains the signature profile value to indicate what authentication method was chosen, which is helpful in case the request signature profile was `http://mid.swisscom.ch/Any-LoA4`.

3.2.4 Additional Services (AS)

The **MSS Signature** supports additional services that may be requested in the request message. Some of them are mandatory and some are optional.

3.2.4.1 User Language

The UserLang is a mandatory service for all Signature Requests.

One of the supported languages (EN, DE, FR, IT) must be defined. Please refer to section 3.2.2.1 or 3.2.3.1 for an example. It should correspond to the language of the DataToBeDisplayed (DTBT) text message.

Example usage (code snippet from the MSS Signature Request):

SOAP/XML
<pre><mss:AdditionalServices> <mss:Service> <mss:Description> <mss:mssURI>http://mss.ficom.fi/TS102204/v1.0.0#userLang</mss:mssURI> </mss:Description> </mss:Service> </mss:AdditionalServices></pre>
REST/JSON
<pre>"AdditionalServices": [{ "Description": "http://mss.ficom.fi/TS102204/v1.0.0#userLang" }]</pre>

3.2.4.2 Geofencing

Geofencing enables Application Providers to define geographical boundaries. They can decide who can access what within that barrier, based on the user's location data at the moment of the Mobile ID authentication. This technology can help Application Providers to lock an application use to a specific geographic location and block any Mobile ID authentication requests outside of the fencing area. With the geofencing service, any authorized¹² Application Provider (AP) may request the user's location data in an MSS Signature request. Location data is only returned in case of a successful MSS Signature responses.

Geofencing can be requested with the MSS Signature Profile's additional service URI set to <http://mid.swisscom.ch/as#geofencing>¹³.

A successful geofencing service response will be part of the MSS Signature Response and it contains:

- **Current Location as Country Code** (a two-letter code, ISO 3166-1 alpha-2)
Example: `"Country": "CH"`
- **Accuracy** (GPS location data accuracy in meters, integer value)
Example: `"Accuracy": "65"`
- **Timestamp** (GPS location data timestamp, formatting¹⁴ of `yyyy-MM-dd'T'HH:mm:ss.SSS`)
Example: `"Timestamp": "2021-06-16T10:21:20.344+02:00"`
- **Device confidence** (value between 0 and 1, 1 being the highest confidence)
0 = location data is very likely from a rooted or jailbroken device
1 = no abnormalities found in relation to rooted or jailbroken devices
Example: `"DeviceConfidence": "1.0"`
- **Location confidence** (value between 0 and 1, 1 being the highest confidence)
0 = location data is very likely falsified
1 = no abnormalities found in relation to falsified data
Example: `"LocationConfidence": "1.0"`

The geofencing service is supported by the SIM and Mobile App. For both authentication methods, the location data (either based on mobile network data or device GPS location) is always mapped to the corresponding country code. All geofencing related information is given without any guarantee and with the exclusion of any legal liability.

For SIM based Mobile ID authentications, the location data is based on the currently registered mobile network. The accuracy value is always 0. At the time of writing only Swisscom SIM cards provide location data.

For Mobile App based Mobile ID authentications, the location data is based on the device GPS location service. The user must have the geofencing toggle enabled and location services permitted. Both Android and iOS App version 1.2.0 or higher support geofencing.

The confidence score is based on various security checks that the Mobile ID App has integrated. A value of 1 means that no abnormalities were detected. It is up to each Application Provider to decide if and how the confidence score, location accuracy and location timestamp values are considered for a successful user authentication.

¹² An Application Provider (AP) must be authorized to use this additional service. Please contact Swisscom if you are interested in this feature.

¹³ The URI <http://mid.swisscom.ch/as#subscriberInfo> is deprecated.

¹⁴ <https://docs.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html>

Example usage (code snippet from the MSS Signature Request):

SOAP/XML
<pre> <mss:AdditionalServices> <mss:Service> <mss:Description> <mss:mssURI>http://mid.swisscom.ch/as#geofencing</mss:mssURI> </mss:Description> </mss:Service> </mss:AdditionalServices> </pre>
REST/JSON
<pre> "AdditionalServices": [{ "Description": "http://mid.swisscom.ch/as#geofencing" }] </pre>

3.2.4.2.1 MSS Signature Request incl. Geofencing

SOAP/XML
<pre> <MSS_Signature> <mss:MSS_SignatureReq MinorVersion="1" MajorVersion="1" MessagingMode="synch" TimeOut="80" xmlns:mss="http://uri.etsi.org/TS102204/v1.1.2#" xmlns:fi="http://mss.ficom.fi/TS102204/v1.0.0#"> <mss:AP_Info AP_ID="yourAP_ID" AP_PWD="yourAP_PWD" AP_TransID="REF0101120000" Instant="2015-01-01T12:00:00.000+01:00"/> <mss:MSSP_Info> <mss:MSSP_ID> <mss:URI>http://mid.swisscom.ch/</mss:URI> </mss:MSSP_ID> </mss:MSSP_Info> <mss:MobileUser> <mss:MSISDN>MOBILE_NUMBER</mss:MSISDN> </mss:MobileUser> <mss:DataToBeSigned MimeType="text/plain" Encoding="UTF-8">TEXT_TO_BE_SIGNED</mss:DataToBeSigned> <mss:SignatureProfile> <mss:mssURI>http://mid.swisscom.ch/MID/v1/AuthProfile1</mss:mssURI> </mss:SignatureProfile> <mss:AdditionalServices> <mss:Service> <mss:Description> <mss:mssURI>http://mss.ficom.fi/TS102204/v1.0.0#userLang</mss:mssURI> </mss:Description> <fi:UserLang>LANGUAGE</fi:UserLang> </mss:Service> <mss:Service> <mss:Description> <mss:mssURI>http://mid.swisscom.ch/as#geofencing</mss:mssURI> </mss:Description> </mss:Service> </mss:AdditionalServices> </mss:MSS_SignatureReq> </MSS_Signature> </pre>
REST/JSON
<pre> { "MSS_SignatureReq": { "AP_Info": { "AP_ID": "yourAP_ID", "AP_PWD": "yourAP_PWD", "AP_TransID": "REF0101120000", "Instant": "2015-01-01T12:00:00.000+01:00" }, "AdditionalServices": [{ "Description": "http://mid.swisscom.ch/as#geofencing" }, { "Description": "http://mss.ficom.fi/TS102204/v1.0.0#userLang", "UserLang": { "Value": "LANGUAGE" } }], "DataToBeSigned": { "Data": "TEXT_TO_BE_SIGNED", "Encoding": "UTF-8", "MimeType": "text/plain" }, "MSSP_Info": { "MSSP_ID": { "URI": "http://mid.swisscom.ch/" } }, "MajorVersion": "1", "MessagingMode": "synch", "MinorVersion": "1", "MobileUser": { "MSISDN": "MOBILE_NUMBER" }, "SignatureProfile": "http://mid.swisscom.ch/MID/v1/AuthProfile1", "TimeOut": "80" } } </pre>

3.2.4.2.2 MSS Signature Response incl. Geofencing Location Data

SOAP/XML
<pre> <soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"> <soapenv:Body> <MSS_SignatureResponse> <mss:MSS_SignatureResp MajorVersion="1" MinorVersion="1" MSSP_TransID="h4dhx" xmlns:mss="http://uri.etsi.org/TS102204/v1.1.2#" xmlns:fi="http://mss.ficom.fi/TS102204/v1.0.0#"> <mss:AP_Info AP_ID="yourAP_ID" AP_PWD="yourAP_PWD" AP_TransID="REF0101120000" Instant="2021-01-01T11:00:00.000+01:00"/> <mss:MSSP_Info Instant="2021-01-01T11:00:00.000+01:00"> <mss:MSSP_ID> <mss:URI>http://mid.swisscom.ch/</mss:URI> </mss:MSSP_ID> </mss:MSSP_Info> <mss:MobileUser> <mss:MSISDN>MOBILE_NUMBER</mss:MSISDN> </mss:MobileUser> <mss:MSS_Signature> <mss:Base64Signature>MIIIdwYJKoZIhvc...</mss:Base64Signature> </mss:MSS_Signature> <mss:SignatureProfile> <mss:mssURI>SIGNATURE_PROFILE</mss:mssURI> </mss:SignatureProfile> <mss>Status> <mss:StatusCode Value="500"/> <mss:StatusMessage>SIGNATURE</mss:StatusMessage> <mss:StatusDetail> <fi:ServiceResponses> <fi:ServiceResponse> <fi:Description> <mss:mssURI>http://mid.swisscom.ch/as#geofencing</mss:mssURI> </fi:Description> <ns1:GeoFencing country="CH" accuracy="16" timestamp="2021-01-01T11:00:00.000+01:00" deviceconfidence="1.0" locationconfidence="1.0" xmlns:ns1="http://mid.swisscom.ch/TS102204/as/v1.0"/> </fi:ServiceResponse> </fi:ServiceResponses> </mss:StatusDetail> </mss>Status> </mss:MSS_SignatureResp> </MSS_SignatureResponse> </soapenv:Body> </soapenv:Envelope> </pre>
REST/JSON
<pre> { "MSS_SignatureResp": { "AP_Info": { "AP_ID": "yourAP_ID", "AP_TransID": "REF0101120000", "Instant": "2021-01-01T11:00:00.000Z" }, "MSSP_Info": { "Instant": "2021-01-01T11:00:00.000Z", "MSSP_ID": { "URI": "http://mid.swisscom.ch/" } }, "MSSP_TransID": "h44okl", "MSS_Signature": { "Base64Signature": "MIIIdwYJKoZIhvc..." }, "MajorVersion": "1", "MinorVersion": "1", "MobileUser": { "MSISDN": "MOBILE_NUMBER" }, "ServiceResponses": [{ "Description": "http://mid.swisscom.ch/as#geofencing", "Geofencing": { "Accuracy": "16", "Country": "CH", "DeviceConfidence": "1.0", "LocationConfidence": "1.0", "Timestamp": "2021-01-01T11:00:00.000+01:00" } }], "SignatureProfile": "SIGNATURE_PROFILE", "Status": { "StatusCode": { "Value": "500" }, "StatusMessage": "SIGNATURE" } } } </pre>

3.2.4.2.3 MSS Signature Response incl. Geofencing Fault Code

SOAP/XML
<pre> <soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"> <soapenv:Body> <MSS_SignatureResponse> <mss:MSS_SignatureResp MajorVersion="1" MinorVersion="1" MSSP_TransID="h4dhx" xmlns:mss="http://uri.etsi.org/TS102204/v1.1.2#" xmlns:fi="http://mss.ficom.fi/TS102204/v1.0.0#"> <mss:AP_Info AP_ID="yourAP_ID" AP_PWD="yourAP_PWD" AP_TransID="REF0101120000" Instant="2021-01-01T11:00:00.000+01:00"/> <mss:MSSP_Info Instant="2021-01-01T11:00:00.000+01:00"> <mss:MSSP_ID> <mss:URI>http://mid.swisscom.ch/</mss:URI> </mss:MSSP_ID> </mss:MSSP_Info> <mss:MobileUser> <mss:MSISDN>MOBILE_NUMBER</mss:MSISDN> </mss:MobileUser> <mss:MSS_Signature> <mss:Base64Signature>MIIDwYJKoZIhvc...</mss:Base64Signature> </mss:MSS_Signature> <mss:SignatureProfile> <mss:mssURI>SIGNATURE_PROFILE</mss:mssURI> </mss:SignatureProfile> <mss:Status> <mss:StatusCode Value="500"/> <mss:StatusMessage>SIGNATURE</mss:StatusMessage> <mss:StatusDetail> <fi:ServiceResponses> <fi:ServiceResponse> <fi:Description> <mss:mssURI>http://mid.swisscom.ch/as#geofencing</mss:mssURI> </fi:Description> <ns1:GeoFencing errorCode="100" errorMessage="Geofencing feature disabled" xmlns:ns1="http://mid.swisscom.ch/TS102204/as/v1.0"/> </fi:ServiceResponse> </fi:ServiceResponses> </mss:StatusDetail> </mss:Status> </mss:MSS_SignatureResp> </MSS_SignatureResponse> </soapenv:Body> </soapenv:Envelope> </pre>
REST/JSON
<pre> { "MSS_SignatureResp": { "AP_Info": { "AP_ID": "yourAP_ID", "AP_TransID": "REF0101120000", "Instant": "2021-01-01T11:00:00.000Z" }, "MSSP_Info": { "Instant": "2021-01-01T11:00:00.000Z", "MSSP_ID": { "URI": "http://mid.swisscom.ch/" } }, "MSSP_TransID": "h44ok1", "MSS_Signature": { "Base64Signature": "MIIDwYJKoZIhvc..." }, "MajorVersion": "1", "MinorVersion": "1", "MobileUser": { "MSISDN": "MOBILE_NUMBER" }, "ServiceResponses": [{ "Description": "http://mid.swisscom.ch/as#geofencing", "Geofencing": { "ErrorCode": "100", "ErrorMessage": "Geofencing feature disabled" } }], "SignatureProfile": "SIGNATURE_PROFILE", "Status": { "StatusCode": { "Value": "500" }, "StatusMessage": "SIGNATURE" } } } </pre>

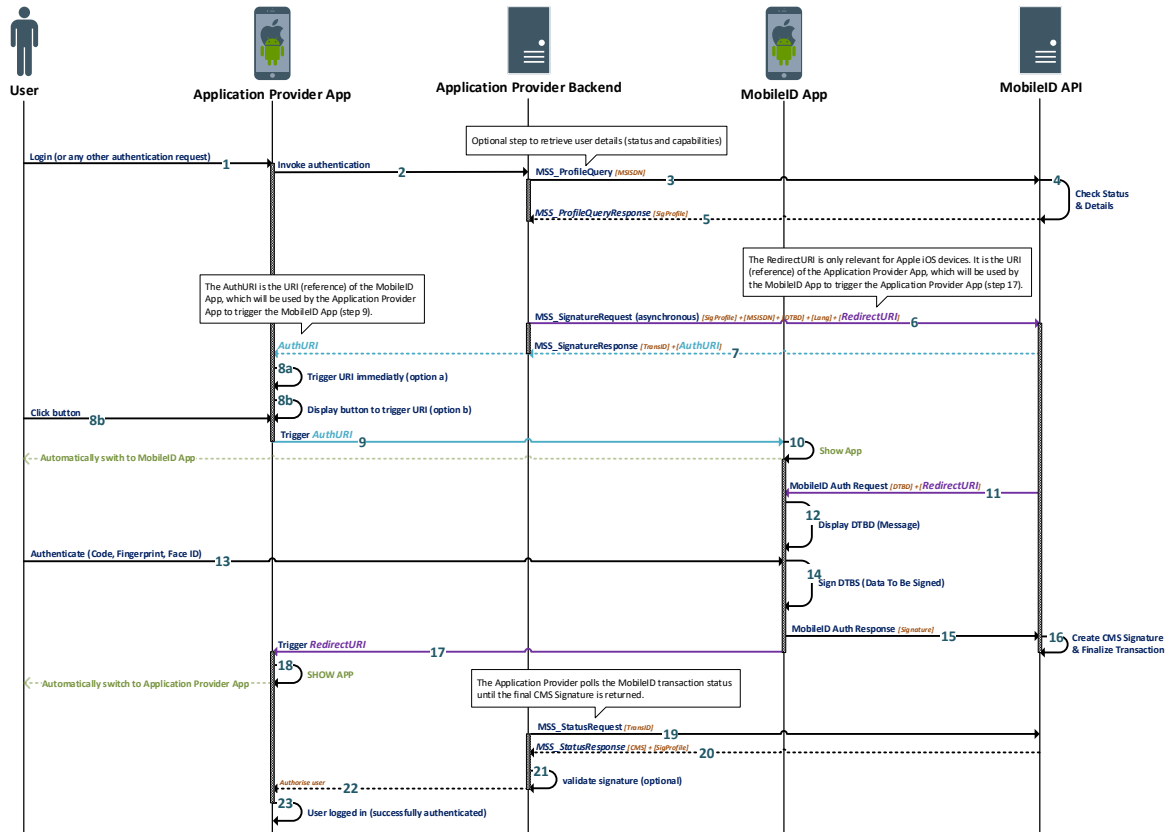
3.2.4.2.4 Geofencing Error Codes

The table below contains all error codes that the geofencing additional service may return. Note that the exact error message may vary for a given error code.

Code	Message
100	The geofencing feature option in the "More" menu is currently disabled.
101	The app failed to retrieve the user's location possibly due to insufficient resources (network, GPS, etc.) or a timeout.
102	The user has not yet responded to the dialog that grants the app permission to access location services.
103	The user has explicitly denied the app permission to access location services.
104	This is on iOS only. The user cannot enable location services possibly due to active restrictions such as parental controls, corporate policy etc. being in place.
105	The user has turned off location services device-wide (for all apps) from the system settings.
106	Location services are unavailable because the device is in Airplane mode.
120	Location failed to the app for an unspecified reason.
121	MSSP internal error (misconfiguration etc.).
122	AP is not authorized to use Geofencing additional service.
123	User has a non-Swisscom SIM card.
200	No location returned from mobile app.
201	App outdated, geofencing not supported.

3.2.4.3 App to App - Mobile Only Authentication

We strongly recommend making use of this service if you intend to invoke the Mobile ID authentication from your own App. The Mobile ID App2App service allows an Application Provider to automatically switch from their App to the Mobile ID App (and the Mobile ID App to automatically switch back to the originating App) as shown in step 10 and 18 in the sequence below. This will greatly improve the usability for the App user.



1. While being in the Application Provider App, the user performs an action that requires authentication
2. The Application Provider App informs the Application Provider backend
3. Optional: The Application Provider backend can request the Mobile ID capabilities of the user
4. Optional: Mobile ID backend checks the user's capabilities and provides the response
5. Optional: Application Provider gets all user details to know if the user has an active Mobile ID App
6. The Application Provider sends an asynchronous signature request that includes the App2App service request. The URI value of the Application Provider App is provided as **RedirectURI** parameter.
7. Mobile ID API responds immediately with a Signature Response, which contains the **AuthURI** parameter. This is the URI value of the Mobile ID App.
8. The Application Provider App can either immediately trigger the Mobile ID App (step 8a) or display a button, which will trigger the Mobile ID App (step 8b)
9. The **AuthURI** is triggered by the Application Provider App
10. **The Mobile ID App is automatically opened on the user's smartphone**
11. The Mobile ID backend invokes the authentication request on the Mobile ID App
12. The Mobile ID authentication message is shown on the Mobile ID App
13. The user confirms the authentication request with his 2nd factor (Passcode or Biometry)
14. The Mobile ID App signs the message with the private key stored on the device
15. The Mobile ID App sends the signed data (signature) to the Mobile ID backend
16. The Mobile ID backend completes the transaction
17. The **RedirectURI** is triggered by the Mobile ID App
18. **The Application Provider App is automatically opened on the user's smartphone**
19. The Application Provider polls the Mobile ID transaction status by sending a MSS Status Request
20. The Status Response returns the final Mobile ID Signature (if outstanding, step 19 is repeated)
21. Optional: The Application Provider validates the CMS Signature to ensure that the Signature is valid
22. The Application Provider forwards the successful Mobile ID authentication result to their App

Example usage (code snippet from the MSS Signature Request):

SOAP/XML
<pre> <mss:AdditionalServices> <mss:Service> <mss:Description> <mss:mssURI>http://mid.swisscom.ch/as#app2app</mss:mssURI> </mss:Description> <mss:App2App> <ns1:RedirectUri xmlns:ns1="http://mid.swisscom.ch/TS102204/as/v1.0">myapp://example</ns1:RedirectUri> </mss:App2App> </mss:Service> </mss:AdditionalServices> </pre>
REST/JSON
<pre> "AdditionalServices": [{ "Description": "http://mid.swisscom.ch/as#app2app", "App2App": { "RedirectUri": "myapp://example" } }] </pre>

Best Practice Guidelines:

- It is strongly recommended for an Application Provider to implement the App2App service as it will greatly improve the usability for any user that uses the Mobile ID App as authentication method
- Both Android and iOS are supported for the automatic App2App switch
- In case of Android, the `AuthURI` is not required. Mobile ID backend will simply ignore the value in such a case. Therefore, the Application Provider doesn't necessarily need to know the user's device type (Android or iOS) and just always provide the `AuthURI` with every App2App service request.
- This service can only be used with the asynchronous MSS Signature. In case a synchronous MSS Signature with App2App service is attempted, Mobile ID API will respond with a fault (`WRONG_PARAM`).
- In case this service is requested, there will be no Mobile ID push notification triggered. That's because a push notification is not required if the Mobile ID App is automatically opened.
- Please refer also to the official documentation from Apple¹⁵ and Android¹⁶ about how to implement custom URL schemes in your app

¹⁵ [Apple Doc](#)

¹⁶ [Android Doc Part 1](#), [Android Doc Part 2](#)

3.2.4.3.1 MSS Signature Request incl. App2App service

SOAP/XML
<pre> <MSS_Signature> <mss:MSS_SignatureReq MinorVersion="1" MajorVersion="1" MessagingMode="asynchClientServer" Timeout="80" xmlns:mss="http://uri.etsi.org/TS102204/v1.1.2#" xmlns:fi="http://mss.ficom.fi/TS102204/v1.0.0#"> <mss:AP_Info AP_ID="yourAP_ID" AP_PWD="yourAP_PWD" AP_TransID="REF0101120000" Instant="2015-01-01T12:00:00.000+01:00"/> <mss:MSSP_Info> <mss:MSSP_ID> <mss:URI>http://mid.swisscom.ch/</mss:URI> </mss:MSSP_ID> </mss:MSSP_Info> <mss:MobileUser> <mss:MSISDN>MOBILE_NUMBER</mss:MSISDN> </mss:MobileUser> <mss:DataToBeSigned MimeType="text/plain" Encoding="UTF-8">TEXT_TO_BE_SIGNED</mss:DataToBeSigned> <mss:SignatureProfile> <mss:mssURI>http://mid.swisscom.ch/Device-LoA4</mss:mssURI> </mss:SignatureProfile> <mss:AdditionalServices> <mss:Service> <mss:Description> <mss:mssURI>http://mss.ficom.fi/TS102204/v1.0.0#userLang</mss:mssURI> </mss:Description> <fi:UserLang>LANGUAGE</fi:UserLang> </mss:Service> <mss:Service> <mss:Description> <mss:mssURI>http://mid.swisscom.ch/as#app2app</mss:mssURI> </mss:Description> <mss:App2App> <ns1:RedirectUri xmlns:ns1="http://mid.swisscom.ch/TS102204/as/v1.0">myapp://example</ns1:RedirectUri> </mss:App2App> </mss:Service> </mss:AdditionalServices> </mss:MSS_SignatureReq> </MSS_Signature> </pre>
REST/JSON
<pre> { "MSS_SignatureReq": { "AP_Info": { "AP_ID": "yourAP_ID", "AP_PWD": "yourAP_PWD", "AP_TransID": "REF0101120000", "Instant": "2015-01-01T12:00:00.000+01:00" }, "AdditionalServices": [{ "Description": "http://mss.ficom.fi/TS102204/v1.0.0#userLang", "UserLang": { "Value": "DE" } }, { "Description": "http://mid.swisscom.ch/as#app2app", "App2App": { "RedirectUri": "myapp://example" } }], "DataToBeSigned": { "Data": "TEXT_TO_BE_SIGNED", "Encoding": "UTF-8", "MimeType": "text/plain" }, "MSSP_Info": { "MSSP_ID": { "URI": "http://mid.swisscom.ch/" } }, "MajorVersion": "1", "MessagingMode": "asynch", "MinorVersion": "2", "MobileUser": { "MSISDN": "MOBILE_NUMBER" }, "SignatureProfile": "http://mid.swisscom.ch/Device-LoA4", "Timeout": "80" } } </pre>

3.2.4.3.2 MSS Signature Response incl. App2App service response

SOAP/XML
<pre> <soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"> <soapenv:Body> <MSS_SignatureResponse> <mss:MSS_SignatureResp MajorVersion="1" MinorVersion="1" MSSP_TransID="h4iof" xmlns:mss="http://uri.etsi.org/TS102204/v1.1.2#" xmlns:fi="http://mss.ficom.fi/TS102204/v1.0.0#"> <mss:AP_Info AP_ID="yourAP_ID" AP_PWD="yourAP_PWD" AP_TransID="REF0101120000" Instant="2015-01-01T12:00:00.000+01:00"/> <mss:MSSP_Info Instant="2015-01-01T12:00:00.100+01:00"> <mss:MSSP_ID> <mss:URI>http://mid.swisscom.ch/</mss:URI> </mss:MSSP_ID> </mss:MSSP_Info> <mss:MobileUser> <mss:MSISDN>MOBILE_NUMBER</mss:MSISDN> </mss:MobileUser> <mss:SignatureProfile> <mss:mssURI>http://mid.swisscom.ch/Device-LoA4</mss:mssURI> </mss:SignatureProfile> <mss:Status> <mss:StatusCode Value="100"/> <mss:StatusMessage>REQUEST_OK</mss:StatusMessage> <mss:StatusDetail> <fi:ServiceResponses> <fi:ServiceResponse> <fi:Description> <mss:mssURI>http://mid.swisscom.ch/as#app2app</mss:mssURI> </fi:Description> <fi:App2App> <ns1:AuthUri xmlns:ns1="http://mid.swisscom.ch/TS102204/as/v1.0"> mobileid://auth?mobile_auth_redirect_uri=myapp%3A%2F%2Fapp.open%23access_token%3DABCD&session_token=32ffc297-0YLukZtgoRVz_wbJfLnViEhZk9aXM8dkHoOnhqf158&user_id=32ffc297-ac33-4be2-b3f4-42588502ea0f </ns1:AuthUri> </fi:App2App> </fi:ServiceResponse> </fi:ServiceResponses> </mss:StatusDetail> </mss:Status> </mss:MSS_SignatureResp> </MSS_SignatureResponse> </soapenv:Body> </soapenv:Envelope> </pre>
REST/JSON
<pre> { "MSS_SignatureResp": { "AP_Info": { "AP_ID": "yourAP_ID", "AP_TransID": "REF0101120000", "Instant": "2015-01-01T11:00:00.000Z" }, "MSSP_Info": { "Instant": "2015-01-01T11:00:00.100Z", "MSSP_ID": { "URI": "http://mid.swisscom.ch/" } }, "MSSP_TransID": "h4iof", "MajorVersion": "1", "MinorVersion": "2", "MobileUser": { "MSISDN": "MOBILE_NUMBER" }, "ServiceResponses": [{ "App2App": { "AuthUri": "mobileid://auth?mobile_auth_redirect_uri=myapp%3A%2F%2Fapp.open%23access_token%3DABCD&session_token=32ffc297-0N5F0yEk2ArMZjFhBWPb4Uifwbk1f3p9ciAURd_QhUQ9e&user_id=32ffc297-ac33-4be2-b3f4-42588502ea0f" }, "Description": "http://mid.swisscom.ch/as#app2app" }], "SignatureProfile": "http://mid.swisscom.ch/Device-LoA4", "Status": { "StatusCode": { "Value": "100" }, "StatusMessage": "REQUEST_OK" } } } </pre>

3.3 MSS Status Query

In case of an asynchronous MSS Signature, the AP needs to poll the status of an on-going signature transaction by sending MSS Status Query requests.

3.3.1 MSS Status Query Request

With the MSS Status Query an AP can poll the final MSS Signature result. At this point in time, AP has received the MSS_SignatureResp with the `MSSP_TransID="h4iof"`. The AP must take the received `MSSP_TransID="h4iof"` and use it in the MSS_StatusReq.

SOAP/XML
<pre> <soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope"> <soapenv:Body> <MSS_StatusQuery> <mss:MSS_StatusReq MinorVersion="1" MajorVersion="1" MSSP_TransID="h4iof" xmlns:mss="http://uri.etsi.org/TS102204/v1.1.2#"> <mss:AP_Info AP_ID="yourAP_ID" AP_PWD="yourAP_PWD" AP_TransID="REF0101120000" Instant="2015-01-01T12:00:00.000+01:00"/> <mss:MSSP_Info> <mss:MSSP_ID> <mss:URI>http://mid.swisscom.ch/</mss:URI> </mss:MSSP_ID> </mss:MSSP_Info> </mss:MSS_StatusReq> </MSS_StatusQuery> </soapenv:Body> </soapenv:Envelope> </pre>
REST/JSON
<pre> { "MSS_StatusReq": { "MajorVersion": "1", "MinorVersion": "1", "AP_Info": { "AP_ID": "yourAP_ID", "AP_PWD": "yourAP_PWD", "Instant": "2015-01-01T12:00:00.000+01:00", "AP_TransID": "REF0101120000" }, "MSSP_Info": { "MSSP_ID": { "URI": "http://mid.swisscom.ch/" } }, "MSSP_TransID": "h4iof" } } </pre>

3.3.2 MSS Status Query Response

Here is an example Status Response with the final MSS Signature result (status 500/SIGNATURE).

SOAP/XML
<pre> <soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"> <soapenv:Body> <MSS_StatusQueryResponse> <mss:MSS_StatusResp MajorVersion="1" MinorVersion="1" xmlns:mss="http://uri.etsi.org/TS102204/v1.1.2#" xmlns:fi="http://mss.ficom.fi/TS102204/v1.0.0#"> <mss:AP_Info AP_ID="yourAP_ID" AP_PWD="yourAP_PWD" AP_TransID="REF0101120000" Instant="2015-01-01T12:00:00.000+01:00"/> <mss:MSSP_Info Instant="2015-01-01T12:00:00.100+01:00"> <mss:MSSP_ID> <mss:URI>http://mid.swisscom.ch/</mss:URI> </mss:MSSP_ID> </mss:MSSP_Info> <mss:MobileUser> <mss:MSISDN>MOBILE_NUMBER</mss:MSISDN> </mss:MobileUser> <mss:MSS_Signature> <mss:Base64Signature>MIIDwYJKoZIhvc...</mss:Base64Signature> </mss:MSS_Signature> <mss:Status> <mss:StatusCode Value="500"/> <mss:StatusMessage>SIGNATURE</mss:StatusMessage> </mss:Status> </mss:MSS_StatusResp> </MSS_StatusQueryResponse> </soapenv:Body> </soapenv:Envelope> </pre>
REST/JSON
<pre> { "MSS_StatusResp": { "AP_Info": { "AP_ID": "yourAP_ID", "AP_TransID": "REF0101120000", "Instant": "2015-01-01T11:00:00.000Z" }, "MSSP_Info": { "Instant": "2015-01-01T11:00:00.100Z", "MSSP_ID": { "URI": "http://mid.swisscom.ch/" } }, "MSS_Signature": { "Base64Signature": "MIIDwYJKoZIhvc..." }, "MajorVersion": "1", "MinorVersion": "1", "MobileUser": MOBILE_NUMBER, "Status": { "StatusCode": { "Value": "500" }, "StatusMessage": "SIGNATURE" } } } </pre>

Note that the Status Query response does not contain the signature profile value, as this value was already returned with the asynchronous MSS Signature response.

A typical status code for this method is 504, which means that the transaction is not yet completed, and the AP must call MSS_StatusReq again (polling). Example for an MSS_StatusResp with a 504 Status Code:

SOAP/XML
<pre> <mss:Status> <mss:StatusCode Value="504"/> <mss:StatusMessage>OUTSTANDING_TRANSACTION</mss:StatusMessage> </mss:Status> </pre>
REST/JSON
<pre> { "Status": { "StatusCode": { "Value": "504" }, "StatusMessage": "OUTSTANDING_TRANSACTION" } } </pre>

3.4 MSS Receipt

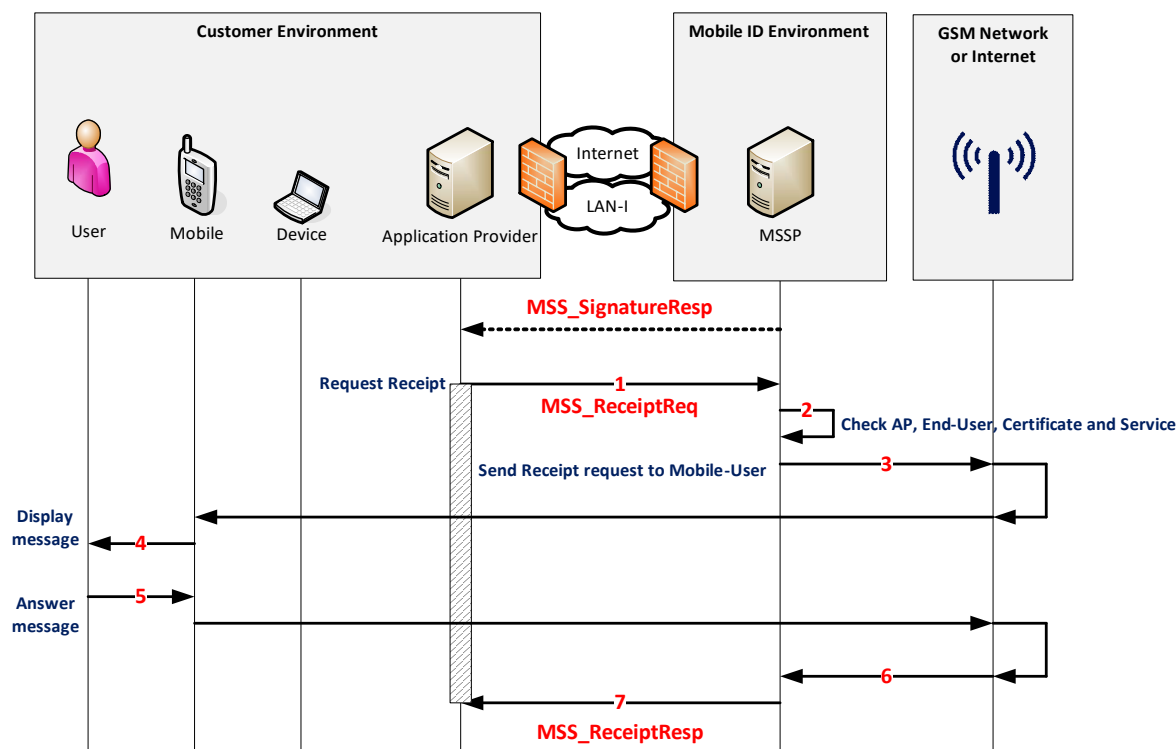
An AP may use this operation after a successful mobile signature request to provide the End-User with a "receipt" that informs him of the mobile signature transaction status.

However, only one receipt per successful signature transaction can be sent.

Only synchronous Signature Receipts are supported. There is no support for asynchronous Signature Receipts.

3.4.1 Synchronous MSS Receipt

In the synchronous mode, the AP initiates the receipt request by calling `MSS_ReceiptReq`, which is then blocked until the signature has been acknowledged, cancelled or the signing times out occurs. The picture depicted below, shows the successful case.



1. For each successful `MSS_SignatureResp` the AP can request an `MSS_ReceiptReq` by passing the same `MSSP_TransID` and same `MSISDN` from the `MSS_SignatureResp` and defining message to be displayed to the End-User.
2. MID backend checks the credentials.
3. MID sends a receipt request to the Mobile ID application on the mobile device.
4. The Mobile ID application displays the message to the End-User.
5. The user press "ok" or "cancel" which trigger MSS Receipt response.
6. The Receipt response is sent to Mobile ID.
7. MID sends `MSS_ReceiptResp` to the AP.

3.4.1.1 MSS Receipt Request

At this point in time, AP has received the MSS_SignatureResp with the `MSSP_TransID="h4iof"`, which is the transaction identifier that need to be set in the MSS_ReceiptReq message.

Moreover, the AP must set the LANGUAGE (one of EN, DE, FR, IT) which is usually the same as used in the preceding signature.

NOTE: The lines highlighted in **pink** is an optional extension to get a more detailed receipt response. This extension is currently supported by the SIM method only! Please remove these lines in case of the App method.

SOAP/XML
<pre> <soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" soap:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"> <soapenv:Body> <MSS_Receipt> <mss:MSS_ReceiptReq MinorVersion="1" MajorVersion="1" MSSP_TransID="h4iof" xmlns:mss="http://uri.etsi.org/TS102204/v1.1.2#" xmlns:sco="http://www.swisscom.ch/TS102204/ext/v1.0.0"> <mss:AP_Info AP_ID="yourAP_ID" AP_PWD="yourAP_PWD" AP_TransID="REF0101120000" Instant="2015-01-01T12:00:00.000+01:00"/> <mss:MSSP_Info> <mss:MSSP_ID> <mss:URI>http://mid.swisscom.ch/</mss:URI> </mss:MSSP_ID> </mss:MSSP_Info> <mss:MobileUser> <mss:MSISDN>MOBILE_NUMBER</mss:MSISDN> </mss:MobileUser> <mss>Status> <mss:StatusCode Value="100"/> <mss>StatusDetail> <sco:ReceiptRequestExtension ReceiptMessagingMode="synch" UserAck="true"> <sco:ReceiptProfile Language="LANGUAGE"> <sco:ReceiptProfileURI>http://mss.swisscom.ch/synch</sco:ReceiptProfileURI> </sco:ReceiptProfile> </sco:ReceiptRequestExtension> </mss>StatusDetail> </mss>Status> <mss:Message MimeType="text/plain" Encoding="UTF-8">RECEIPT_MESSAGE</mss:Message> </mss:MSS_ReceiptReq> </MSS_Receipt> </soapenv:Body> </soapenv:Envelope> </pre>
REST/JSON
<pre> { "MSS_ReceiptReq": { "AP_Info": { "AP_ID": "yourAP_ID", "AP_PWD": "yourAP_PWD", "AP_TransID": "REF0101120000", "Instant": "2015-01-01T12:00:00.000+01:00" }, "MSSP_Info": { "MSSP_ID": { "URI": "http://mid.swisscom.ch/" } }, "MSSP_TransID": "h4iof", "MajorVersion": "1", "Message": { "Data": "RECEIPT_MESSAGE", "Encoding": "UTF-8", "MimeType": "text/plain" }, "MinorVersion": "1", "MobileUser": { "MSISDN": "MOBILE_NUMBER" }, "Status": { "StatusCode": { "Value": "100" }, "StatusDetail": { "ReceiptRequestExtension": { "ReceiptMessagingMode": "synch", "ReceiptProfile": { "Language": "LANGUAGE", "ReceiptProfileURI": "http://mss.swisscom.ch/synch" }, "UserAck": "true" } } } } } </pre>

3.4.1.2 MSS Receipt Response

Note that the `ReceiptResponseExtension` highlighted in pink is an optional extension to get a more detailed receipt response. The response will only contain this part if `ReceiptRequestExtension` was requested in the Receipt Request message. This extension is currently supported by the SIM method only. The table below describes different `ReceiptResponseExtension` related scenarios.

Response scenario	UserAck	User Response	Fault Message
No user response could be received within 80 seconds. It is unknown if the user got the receipt message.	false	N/A	N/A
User accepted the receipt message	true	"OK"	N/A
User cancelled the receipt message	true	"CANCEL"	"User Cancelled the request"
The receipt message was displayed for 60 seconds but user did not respond	true	"TIMEOUT"	"Timeout waiting response"

SOAP/XML

```
<soapenv:Envelope
  xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <MSS_ReceiptResponse>
      <mss:MSS_ReceiptResp MajorVersion="1" MinorVersion="1"
        xmlns:mss="http://uri.etsi.org/TS102204/v1.1.2#"
        xmlns:fi="http://mss.ficom.fi/TS102204/v1.0.0#">
        <mss:AP_Info AP_ID="yourAP_ID" AP_PWD="yourAP_PWD" AP_TransID="REF0101120000"
          Instant="2015-01-01T12:00:00.000+01:00"/>
        <mss:MSSP_Info Instant="2015-01-01T12:00:00.100+01:00">
          <mss:MSSP_ID>
            <mss:URI>http://mid.swisscom.ch/</mss:URI>
          </mss:MSSP_ID>
        </mss:MSSP_Info>
        <mss:Status>
          <mss:StatusCode Value="100"/>
          <mss:StatusMessage>REQUEST_OK</mss:StatusMessage>
          <mss:StatusDetail>
            <ns1:ReceiptResponseExtension ReceiptMessagingMode="synch"
              UserAck="true"
              UserResponse="{&quot;status&quot;:&quot;OK&quot;}"
              xmlns:ns1="http://www.swisscom.ch/TS102204/ext/v1.0.0"/>
            </mss:StatusDetail>
          </mss:Status>
        </mss:MSS_ReceiptResp>
      </MSS_ReceiptResponse>
    </soapenv:Body>
  </soapenv:Envelope>
```

REST/JSON

```
{
  "MSS_ReceiptResp": {
    "AP_Info": {
      "AP_ID": "yourAP_ID",
      "AP_TransID": "REF0101120000",
      "Instant": "2015-01-01T11:00:00.000Z"
    },
    "MSSP_Info": {
      "Instant": "2015-01-01T11:00:00.100Z",
      "MSSP_ID": {
        "URI": "http://mid.swisscom.ch/"
      }
    },
    "MajorVersion": "1",
    "MinorVersion": "1",
    "Status": {
      "StatusCode": {
        "Value": "100"
      },
      "StatusDetail": {
        "ReceiptResponseExtension": {
          "ClientAck": "false",
          "NetworkAck": "false",
          "ReceiptMessagingMode": "synch",
          "UserAck": "true",
          "UserResponse": "{ \"status\": \"OK\" }"
        }
      },
      "StatusMessage": "REQUEST_OK"
    }
  }
}
```

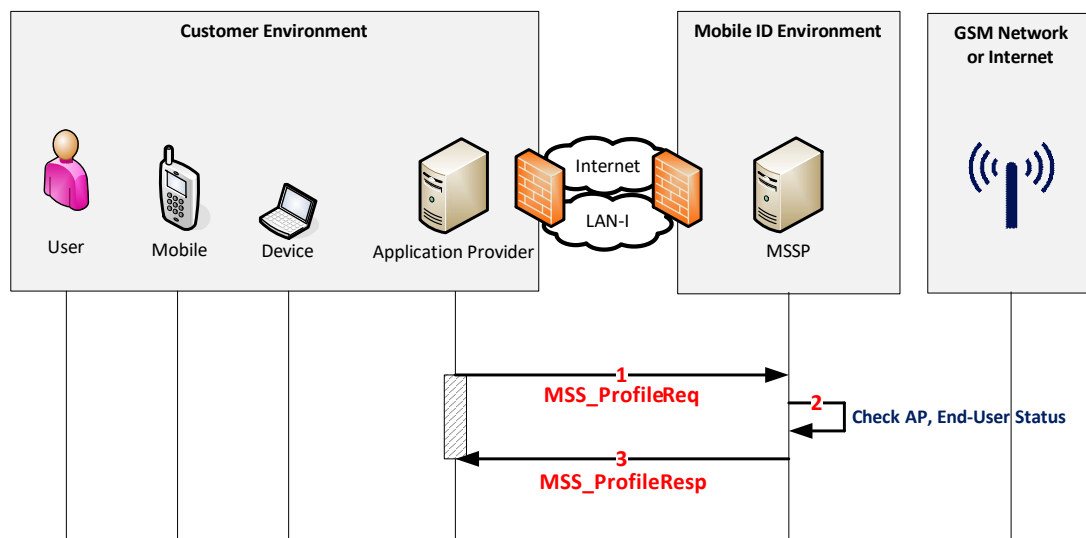
3.4.2 Encrypted MSS Receipts

Marked as obsolete - We currently no longer support encrypted Signature Receipts.

3.5 MSS Profile Query

An AP may use this operation to check the Mobile ID status of a specific user (MSISDN) without any end-user interaction. The AP can use a Profile Query request as depicted below.

The MSS Profile Query considers both SIM- and App method. Therefore, an AP can use the MSS Profile Query to get, among other details (see section 3.5.1.1), a list of supported signature profiles of a specific user. The signature profile is related to the authentication method (SIM or App) supported by a user, see section 3.2.1.



1. Before to send a signature request for authentication, the AP validates the status of a Mobile ID user by sending an MSS_ProfileReq request to Mobile ID
2. MSSP checks the user status
3. In case of an active user, it retrieves the Profiles of the end-user and sends back an MSS_ProfileResp. Otherwise, if the user is not active, the server sends back a fault response that may contain additional details about the user status.

3.5.1 MSS Profile Query Request

The lines highlighted in **pink** are optional Profile Query Extension parameters (see section 3.5.1.1).

SOAP/XML
<pre> <soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope" xmlns:ets="http://uri.etsi.org/TS102204/etsi204-kiuru.wsd1" xmlns:v1="http://uri.etsi.org/TS102204/v1.1.2#"> <soap:Body> <ets:MSS_ProfileQuery> <MSS_ProfileReq MajorVersion="2" MinorVersion="0" xmlns:mss="http://uri.etsi.org/TS102204/v1.1.2#" xmlns:fi="http://mss.ficom.fi/TS102204/v1.0.0#"> <mss:AP_Info AP_ID="yourAP_ID" AP_PWD="yourAP_PWD" AP_TransID="REF0101120000" Instant="2015-01-01T12:00:00.000+01:00"/> <mss:MSSP_Info> <mss:MSSP_ID> <mss:URI>http://mid.swisscom.ch/</mss:URI> </mss:MSSP_ID> </mss:MSSP_Info> <mss:MobileUser> <mss:MSISDN>MOBILE_NUMBER</mss:MSISDN> </mss:MobileUser> <mss:Params>sscds state certs pinstatus rcstatus aastatus</mss:Params> </MSS_ProfileReq> </ets:MSS_ProfileQuery> </soap:Body> </soap:Envelope> </pre>
REST/JSON
<pre> { "MSS_ProfileReq": { "AP_Info": { "AP_ID": "yourAP_ID", "AP_PWD": "yourAP_PWD", "AP_TransID": "REF0101120000", "Instant": "2015-01-01T12:00:00.000+01:00" }, "MSSP_Info": { "MSSP_ID": { "URI": "http://mid.swisscom.ch/" } }, "MajorVersion": "2", "MinorVersion": "0", "MobileUser": { "MSISDN": "MOBILE_NUMBER" }, "Params": "sscds state certs pinstatus rcstatus aastatus" } } </pre>

Note that **MajorVersion** value must be set to “2” and **MinorVersion** value must be set to “0”.

3.5.1.1 MSS Profile Query Request Extensions

Optional Profile Query Extension parameters can be set in the Profile Query request message to get additional information in the Profile Query response message.

Params value	Profile Query response content
sscds	A list of all available <u>S</u> ecure <u>S</u> ignature <u>C</u> reation <u>D</u> evice <u>s</u> of the user
state	User account state details
certs	X.509 mobile user certificate binary content and details, incl. subject name. The subject name contains the user’s unique MID serial number value.
pinstatus	Mobile ID PIN status
rcstatus	Recovery Code status (has it been created: true or false)
aastatus	Auto-Activation status (is it enabled: true or false), see section 5.

3.5.2 MSS Profile Query Response

The lines highlighted in **pink** are optional Profile Query Extension parameters (see section 3.5.1.1).

In this example, the mobile user has both SIM method (state **INACTIVE**) and App method (state **ACTIVE**). The signature profile `http://mid.swisscom.ch/STK-LoA4` is not listed due to the SIM state being inactive and `AutoActivation` set to false. A signature request attempt with `http://mid.swisscom.ch/STK-LoA4` would lead to a fault `404/NO_KEY_FOUND`.

SOAP/XML

```
<soapenv:Envelope xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <MSS_ProfileQueryResponse>
      <mss:MSS_ProfileResp xmlns:mss="http://uri.etsi.org/TS102204/v1.1.2#"
        xmlns:fi="http://mss.ficom.fi/TS102204/v1.0.0#" MajorVersion="2" MinorVersion="0">
        <mss:AP_Info AP_ID="yourAP_ID" AP_PWD="yourAP_PWD" AP_TransID="REF0101120000"
          Instant="2015-01-01T12:00:00.000+01:00" xmlns:mss="http://uri.etsi.org/TS102204/v1.1.2#"
          xmlns:fi="http://mss.ficom.fi/TS102204/v1.0.0#" />
        <mss:MSSP_Info Instant="2020-03-02T13:03:57.456+01:00">
          <mss:MSSP_ID>
            <mss:URI>http://mid.swisscom.ch/</mss:URI>
          </mss:MSSP_ID>
          </mss:MSSP_Info>
          <mss:SignatureProfile>
            <mss:mssURI>http://mid.swisscom.ch/Any-LoA4</mss:mssURI>
          </mss:SignatureProfile>
          <mss:SignatureProfile>
            <mss:mssURI>http://mid.swisscom.ch/Device-LoA4</mss:mssURI>
          </mss:SignatureProfile>
          <mss:SignatureProfile>
            <mss:mssURI>http://mid.swisscom.ch/MID/v1/AuthProfile1</mss:mssURI>
          </mss:SignatureProfile>
          <mss:Status>
            <mss:StatusCode Value="100"/>
            <mss:StatusMessage>REQUEST_OK</mss:StatusMessage>
            <mss:StatusDetail>
              <mcs:ProfileQueryExtension xmlns:mcs="http://www.methics.fi/TS102204/ext/v1.0.0"
                xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
                <mcs:MobileUser RecoveryCodeCreated="true" AutoActivation="false"/>
                <mcs:Sscds>
                  <mcs:Sim>
                    <mcs:State>INACTIVE</mcs:State>
                    <mcs:PinStatus blocked="false"/>
                    <mcs:MobileUserCertificate State="INACTIVE" Algorithm="RSA">
                      <ds:X509Certificate>X509_BASE64_CERT_USER</ds:X509Certificate>
                      <ds:X509SubjectName>X509_BASE64_SUBJECT_USER</ds:X509SubjectName>
                      <ds:X509Certificate>X509_BASE64_CERT_CA</ds:X509Certificate>
                      <ds:X509SubjectName>X509_BASE64_SUBJECT_CA</ds:X509SubjectName>
                    </mcs:MobileUserCertificate>
                    <mcs:MobileUserCertificate State="INACTIVE" Algorithm="EC">
                      <ds:X509Certificate>X509_BASE64_CERT_USER</ds:X509Certificate>
                      <ds:X509SubjectName>X509_BASE64_SUBJECT_USER</ds:X509SubjectName>
                      <ds:X509Certificate>X509_BASE64_CERT_CA</ds:X509Certificate>
                      <ds:X509SubjectName>X509_BASE64_SUBJECT_CA</ds:X509SubjectName>
                    </mcs:MobileUserCertificate>
                  </mcs:Sim>
                  <mcs:App>
                    <mcs:State>ACTIVE</mcs:State>
                    <mcs:PinStatus blocked="false"/>
                    <mcs:MobileUserCertificate State="ACTIVE" Algorithm="RSA">
                      <ds:X509Certificate>X509_BASE64_CERT_USER</ds:X509Certificate>
                      <ds:X509SubjectName>X509_BASE64_SUBJECT_USER</ds:X509SubjectName>
                      <ds:X509Certificate>X509_BASE64_CERT_CA</ds:X509Certificate>
                      <ds:X509SubjectName>X509_BASE64_SUBJECT_CA</ds:X509SubjectName>
                    </mcs:MobileUserCertificate>
                  </mcs:App>
                </mcs:Sscds>
              </mcs:ProfileQueryExtension>
            </mss:StatusDetail>
          </mss:Status>
        </mss:MSS_ProfileResp>
      </MSS_ProfileQueryResponse>
    </soapenv:Body>
  </soapenv:Envelope>
```

REST/JSON

```
{
  "MSS_ProfileResp": {
    "AP_Info": {
      "AP_ID": "yourAP_ID",
      "AP_TransID": "REF0101120000",
      "Instant": "2015-01-01T11:00:00.000Z"
    },
    "MSSP_Info": {
      "Instant": "2015-01-01T11:00:00.100Z",
      "MSSP_ID": {
        "URI": "http://mid.swisscom.ch/"
      }
    },
    "MajorVersion": "2",
    "MinorVersion": "0",
    "SignatureProfile": [
      "http://mid.swisscom.ch/Any-LoA4",
      "http://mid.swisscom.ch/Device-LoA4",
      "http://mid.swisscom.ch/MID/v1/AuthProfile1"
    ],
    "Status": {
      "StatusCode": {
        "Value": "100"
      },
      "StatusDetail": {
        "ProfileQueryExtension": {
          "MobileUser": {
            "AutoActivation": false,
            "RecoveryCodeCreated": true
          },
          "Sscds": {
            "App": [
              {
                "MobileUserCertificate": [
                  {
                    "Algorithm": "RSA",
                    "State": "ACTIVE",
                    "X509Certificate": [
                      "X509_BASE64_CERT_USER",
                      "X509_BASE64_CERT_CA"
                    ],
                    "X509SubjectName": [
                      "X509_BASE64_SUBJECT_USER",
                      "X509_BASE64_SUBJECT_CA"
                    ]
                  }
                ],
                "PinStatus": {
                  "Blocked": false
                },
                "State": "ACTIVE"
              }
            ],
            "Sim": {
              "MobileUserCertificate": [
                {
                  "Algorithm": "RSA",
                  "State": "INACTIVE",
                  "X509Certificate": [
                    "X509_BASE64_CERT_USER",
                    "X509_BASE64_CERT_CA"
                  ],
                  "X509SubjectName": [
                    "X509_BASE64_SUBJECT_USER",
                    "X509_BASE64_SUBJECT_CA"
                  ]
                },
                {
                  "Algorithm": "EC",
                  "State": "INACTIVE",
                  "X509Certificate": [
                    "X509_BASE64_CERT_USER",
                    "X509_BASE64_CERT_CA"
                  ],
                  "X509SubjectName": [
                    "X509_BASE64_SUBJECT_USER",
                    "X509_BASE64_SUBJECT_CA"
                  ]
                }
              ],
              "PinStatus": {
                "Blocked": false
              },
              "State": "INACTIVE"
            }
          }
        }
      },
      "StatusMessage": "REQUEST_OK"
    }
  }
}
```


4 Best Practices

Our [GitHub Repository](#) contains many different examples for a Mobile ID client implementation.

The repo '[mobileid-client-java](#)' is the main Java-based reference implementation for the Mobile ID REST and SOAP API client. The library provided by this repository is for all clients that are developing Java-based projects that need secure authentication and authorization services using the mobile phone. The library works with Java 8+ projects and can be added as a project dependency and used in any scenario that needs to access the Swisscom Mobile ID service.

4.1 MSS Signature

4.1.1 Signature Request

Below are major aspects when a signature request is constructed.

- 1) Define a unique `AP_TransID`¹⁷ (Transaction ID) value.
- 2) Set the current time as value for `Instant` including the time zone information. If the value is too far from MID Service's current time, a fault response with sub-code 101 will be raised.
Example: `2020-01-01T12:00:00.000+01:00`. It shall be compliant with `w3.org xs:dateTime`¹⁸.
- 3) The triplet `AP_ID`, `AP_TransID` and `Instant` shall be unique, otherwise the request will be rejected.
- 4) MSSIDN shall be in international format without spaces. A leading '+'-character is allowed but not mandatory. Example: `+41791234567`
- 5) The `UserLang` value shall be one of the supported languages (`EN`, `DE`, `FR`, `IT`) and shall match with the language of the DataToBeDisplayed (DTBD) text message.
- 6) The DataToBeDisplayed (DTBD) text message, displayed on the mobile phone:
 - Shall be in UTF-8¹⁹
 - Ideally contains a unique transaction ID (e.g. timestamp, customer ID, contract ID, etc.)
 - Shall not exceed 239 characters. In case one of the characters is not present in the GSMDA²⁰ character set (for example the lowercase cedilla 'ç'), the length of the message is reduced to max. 119 characters. We suggest keeping the message as short as possible.

Example DTBD: `Bank ACME: Proceed with the login? (TXN-3D5K)`

¹⁷ Type of the `AP_TransID` is `xsd:NCName`, refer to http://www.datypic.com/sc/xsd/t-xsd_NCName.html

¹⁸ <http://www.w3.org/TR/xmlschema-2/#dateTime>

¹⁹ <http://en.wikipedia.org/wiki/UTF-8>

²⁰ In mobile telephony [GSM 03.38](#) is the standard character set used in short message service.

4.1.2 Signature Response

Below are major aspects of response validation.

- 1) Verify that `AP_TransID` and `MSISDN` value of the response matches with the value of the request
- 2) Validate the mobile user X.509 certificate. Your client should only trust a certificate if it chains up to a trust anchor that matches with your certificate TrustStore. You should only have the relevant root CA certificate (section 7) in your TrustStore.
- 3) Verify the digital signature. The signed content should match with the DTBD value of the preceding signature request. Make sure your client can verify both RSA (2048 Bits) and ECDSA (ECC p256-r1) signatures.
- 4) Optionally, for highest level of assurance and a truly strong two-factor-authentication process, validate the Mobile ID serial number as described in section 4.2.
- 5) Handle all status and fault codes with proper exception handling.

We do not recommend doing certificate revocation checks. Mobile ID user certificates are never revoked - instead it is the Mobile ID service itself that manages the user account state.

4.1.3 Signature Concurrency Control

This section explains what happens if an Application Provider sends an MSS Signature request while there is already a signature transaction on-going for the same MSISDN and same authentication method.

If the SIM method is targeted for both authentication attempts, the 2nd signature request is rejected with fault `406 / PB_SIGNATURE_PROCESS` (see section 6) if the 1st signature transaction is still in progress.

If the App method is targeted for both authentication attempts, the already on-going 1st signature transaction is being cancelled with fault `401 / USER_CANCEL` (see section 6) and the 2nd signature request is displayed on the Mobile ID App.

4.2 Mobile ID Serial Number Validation

With an MSS Signature response, you will get signature data and the mobile user's X.509 certificate, which contains the public key required to validate the signature data.

That X.509 certificate's Subject Name (Distinguished Name) contains a unique Mobile ID serial number. Example subject name with the serial number highlighted in pink:

```
cn=midcheptod58qe59:pn,serialnumber=midcheptod58qe59,pseudonym=midcheptod58qe59
```

For highest level of assurance and a truly strong two-factor-authentication process, an AP should store this value at the first signature and then verify every subsequent signature response if that serial number value still matches.

If the value does not match anymore, it means that the user re-activated his account without a valid recovery option²¹. In such case (serial number mismatch) the 2nd factor (knowledge/inherence) is not assured.

4.3 Timeout Value

Use the reference values below to set an appropriate transaction timeout:

MSS Operation	Transaction Timeout ²²	Client Connection Timeout ²³
Synchronous MSS Signature	80 seconds for Mobile ID SIM 40 seconds for Mobile ID App	90 seconds 50 seconds
Asynchronous MSS Signature	80 seconds for Mobile ID SIM 40 seconds for Mobile ID App	10 seconds
MSS Status Query	-	10 seconds
MSS Receipt	-	90 seconds
MSS Profile Query	-	10 seconds

4.4 Mobile ID FAQ

Please visit <https://mobileid.ch/en/faq> for a list of frequently asked questions about Mobile ID

²¹ Visit <https://mobileid.ch> for further information about account recovery

²² How long the transaction at Mobile ID will be valid. This is set via the "Timeout=" attribute in the request message.

²³ How long your AP client (socket) shall wait for a response message from Mobile ID.

4.5 Mobile ID Service Health Check

The recommended way to check the service health status is to send a synchronous MSS Signature request with the MSISDN value set to `+41000000000` and DTBD set to `Heartbeat`.

The Mobile ID service health check is successful if a fault code `101/WRONG_PARAM` with Detail value `Illegal msisdn` is returned, as shown in the example:

SOAP/XML
<pre> .. <soapenv:Fault> <soapenv:Code> <soapenv:Value>soapenv:Sender</soapenv:Value> <soapenv:Subcode xmlns:mss="http://uri.etsi.org/TS102204/v1.1.2#" xmlns:fi="http://mss.ficom.fi/TS102204/v1.0.0#"> <soapenv:Value>mss:_101</soapenv:Value> </soapenv:Subcode> </soapenv:Code> <soapenv:Reason> <soapenv:Text xml:lang="en">WRONG_PARAM</soapenv:Text> </soapenv:Reason> <soapenv:Detail> <ns1:detail xmlns:ns1="http://kiuru.methics.fi/mssp">Illegal msisdn</ns1:detail> </soapenv:Detail> </soapenv:Fault> .. </pre>
REST/JSON
<pre> { "Fault": { "Code": { "SubCode": { "Value": "_101", "ValueNs": "http://uri.etsi.org/TS102204/v1.1.2#" }, "Value": "Sender", "ValueNs": "http://www.w3.org/2003/05/soap-envelope" }, "Detail": "Illegal msisdn", "Reason": "WRONG_PARAM" } } </pre>

4.6 Mobile ID Client Examples



The GitHub Repository at <https://github.com/MobileID-Strong-Authentication> contains different examples for a Mobile ID client.

The repo '[mobileid-client-java](#)' is the main Java-based reference implementation for the Mobile ID REST and SOAP API client.

The library provided by this repository is for all clients that are developing Java-based projects that need secure authentication and authorization services using the mobile phone.

The library works with Java 8+ projects and can be added as a project dependency and used in any scenario that needs to access the Swisscom Mobile ID service.

5 Auto Activation

5.1 Introduction

Auto Activation is an optional feature for APs, which can greatly improve the user experience for Mobile ID SIM authentications. It is applicable for SIM authentication only and this Auto Activation feature is not relevant for Mobile ID App authentications.

Usually, a brand new Mobile ID SIM card must be activated just once on the [MyMobileID](#) Portal. With this Mobile ID activation process, the user can set his personal Mobile ID PIN (Set & Confirm PIN) and it will create the required signing key on the SIM card. After that, the SIM card is ready to be used for Mobile ID authentications.

Auto Activation is an optional feature that is *disabled by default* but can be enabled per Application Provider. If enabled, a Signature Request to a user with an inactive Mobile ID SIM (which means the user did not yet set his personal Mobile ID PIN) will invoke an implicit activation process during the on-going signature transaction.

So, the Auto Activation will achieve both a successful Mobile ID activation and a successful authentication (Signature Response) at the same time! From now on, that user will be Mobile ID active.

From AP perspective, the user will just have a successful authentication and there is no difference whether the user was auto-activated (during the authentication transaction) or not.

Auto Activation feature can successfully prevent a fault sub-code 404 (this fault code means that the user did not yet activate his Mobile ID SIM card) which can happen when an AP attempts to do a Mobile ID authentication with a user that has an inactive Mobile ID SIM card.

5.2 How to implement this feature

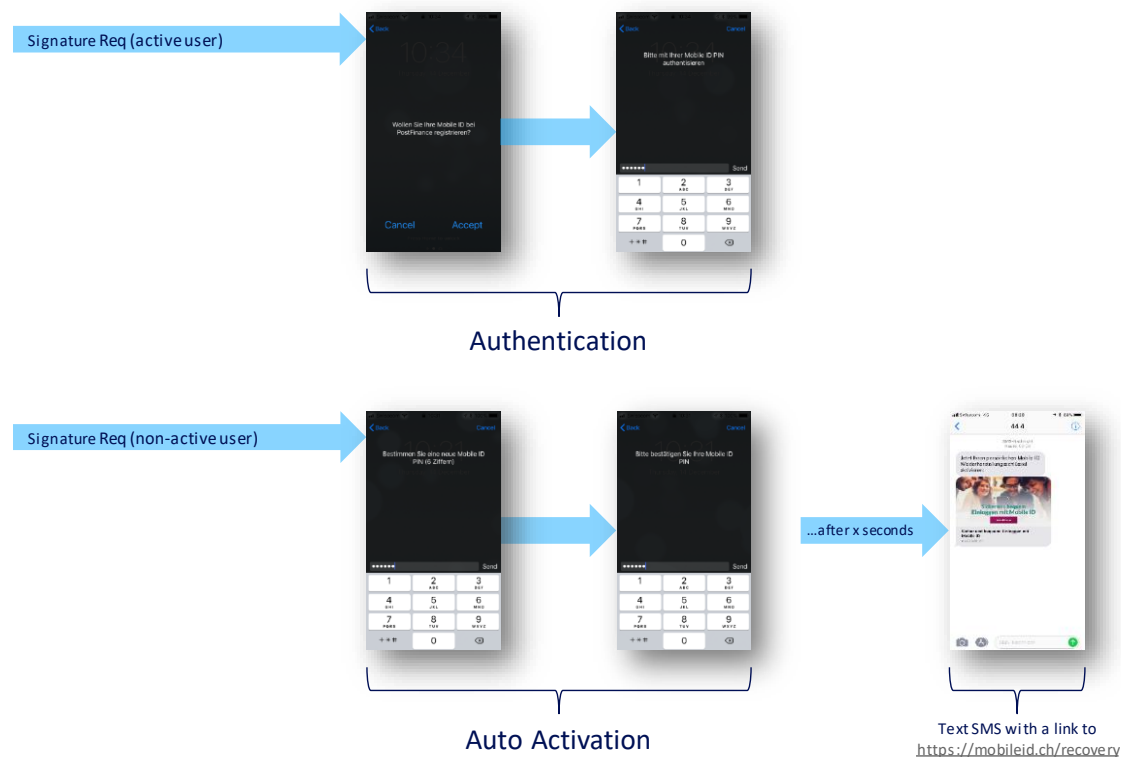
To enable the Auto Activation feature for an Application Provider (AP), the AP must ensure that the user has accepted the Mobile ID specific *terms & conditions* prior to proceed with the auto activation steps.

An AP may use the Profile Query Extensions (see section 3.5) to know if a signature request to a user will invoke auto activation or not. However, an AP does not necessarily need to know that.

Please speak to your Swisscom contact if you wish to get this feature enabled for your AP account. Optionally we can also setup specific test SIM cards, which allows an AP to test this feature (because those test SIM cards will be configured to always trigger the auto activation).

5.3 User Perspective

From a user perspective, the steps displayed on the mobile device look very similar, if you compare the signature process of an active Mobile ID SIM vs. the process of an inactive Mobile ID SIM.



Note that in case of a successful auto activation, the Mobile ID server sends a Text SMS to the user, to remind the user to create a Mobile ID recovery code.²⁴

²⁴ Each time a user completes the Mobile ID activation process, she or he will receive a code that enables her or him to recover Mobile ID and maintain her or his existing pairings to service providers, if you lose your phone, change SIM cards or switch to an e-SIM card. See <https://mobileid.ch/recovery>

6 Status and Fault Codes

6.1 Overview

Status Code	Fault Code	Status Message	Description	MSS Signature	MSS Status Query	MSS Receipt	MSS Profile Query
100		REQUEST_OK	The request from the AP has been accepted.	X		X	X
	101	WRONG_PARAM	The AP's request contains wrong parameters.	X	X	X	X
	102	MISSING_PARAM	The AP's request has missing parameters.	X	X	X	X
	103	WRONG_DATA_LENGTH	The AP's request contains a DTBD message that is exceeding the max. allowed length.	X		X	
	104	UNAUTHORIZED_ACCESS	AP is not authorized to access the Mobile ID API. This is typically due to a wrong AP_ID value or missing X509 client certificate.	X	X	X	X
	105	UNKNOWN_CLIENT	Either the MSISDN is unknown to the MID service and there is no Mobile ID user with that MSISDN. Or the MSISDN is an app only user and the AP is not authorized to use the app method.	X		X	X
	107	INAPPROPRIATE_DATA	The AP's request was not accepted due to inappropriate data. Typically, the DTBD message does not contain the mandatory prefix string (see section 2.19) that is a unique string for each AP.	X		X	
	108	INCOMPATIBLE_INTERFACE	The AP's request contains bad data. Typically, a wrong <code>MajorVersion</code> or <code>MinorVersion</code> value has been set in the request.	X	X	X	X
	109	UNSUPPORTED_PROFILE	Either the AP has specified an MSS signature profile value that the MSSP does not support or the AP is not authorized to use the Signature Profile. See section 3.2.1.	X			
	208	EXPIRED_TRANSACTION	The transaction timed out. The AP may try again.	X	X	X	
	209	OTA_ERROR	A Problem related to the MSSP internal Over-The-Air (OTA) communication with the Mobile ID user's SIM. Typically, there is a temporary problem with SMS communication.	X	X		
	401	USER_CANCEL	The user cancelled the request at the mobile phone.	X	X		
	402	PIN_NR_BLOCKED	The Mobile ID PIN of the SIM method is blocked. The user must re-activate the Mobile ID SIM card on the Mobile ID selfcare portal.	X	X	X	X
	403	CARD_BLOCKED	The Mobile ID user is currently suspended. Please contact Swisscom Support.	X	X	X	X
	404	NO_KEY_FOUND	The Mobile ID user exists but is not in an active state. The user must activate the account on the Mobile ID selfcare portal.	X	X	X	X
	406	PB_SIGNATURE_PROCESS	A signature transaction is already on-going. Please try again later.	X	X		
	422	NO_CERT_FOUND	The Mobile ID user exists but is not in an active state. The user must activate the account on the Mobile ID selfcare portal.	X	X	X	X
500		SIGNATURE	The MSS Signature transaction was successful.	X	X		
501		REVOKED_CERTIFICATE	The Mobile ID user's x509 certificate has been revoked. The user must re-activate the account on the Mobile ID selfcare portal.	X	X		
502		VALID_SIGNATURE	The MSS Signature transaction was successful.	X	X		
503		INVALID_SIGNATURE	The MSS Signature transaction failed due to invalid signature data. The user may try to re-activate the account on the Mobile ID selfcare portal. It may be required to replace the SIM card.	X	X		
504		OUTSTANDING_TRANSACTION	The MSS Signature transaction is outstanding. The AP must try again later.		X		
900		INTERNAL_ERROR	An internal error on MSSP has occurred. Please try again later or contact Swisscom Support, if the problem persists.	X	X	X	X

6.2 Testing Status and Fault Codes

Specific MSISDNs are available to test different type of response status and fault codes. By placing a request to one of this number the related status or fault code will be raised.

The following MSISDN structure is used for testing **fault codes**, which might help a developer to test the error handling of their Mobile ID client.

- `+41000092<faultcode>` - Use one of the 3-digit fault sub-code values listed in section 6.1.

On the other hand, a **successful** Mobile ID signature response can be tested by using one of the MSISDNs below, which might help with CI/CD pipelines that include automated regression testing with Mobile ID authentications:

- `41700092501` - This number returns a successful Mobile ID signature response based on an EC key
- `41700092502` - This number returns a successful Mobile ID signature response based on an RSA key

6.2.1 Test-MSISDN Overview

Request	Fault Response		
Test-MSISDN	Fault Code	Reason Message	Detail Message
41000092101	101	WRONG_PARAM	Error among the arguments of the request
41000092102	102	MISSING_PARAM	An argument in the request is missing
41000092103	103	WRONG_DATA_LENGTH	The DataToBeSigned are too large. Limitations are due to the Mobile Signature technology implemented by the MSSP.
41000092104	104	UNAUTHORIZED_ACCESS	The AP is unknown, or the password is wrong, or the AP asks for an additional service for which it has not subscribed.
41000092105	105	UNKNOWN_CLIENT	MSISDN is unknown
41000092107	107	INAPPROPRIATE_DATA	DTBD matching failed
41000092108	108	INCOMPATIBLE_INTERFACE	The minor version and/or major version parameters are inappropriate for the receiver of the message.
41000092109	109	UNSUPPORTED_PROFILE	The user does not support this Mobile Signature Profile
41000092208	208	EXPIRED_TRANSACTION	Transaction Expiry date has been reached or Time out has lapsed.
41000092209	209	OTA_ERROR	The MSSP has not succeeded to contact the end-user's mobile equipment Bad connection...)
41000092401	401	USER_CANCEL	User cancelled the request
41000092402	402	PIN_NR_BLOCKED	PIN of the mobile user is blocked
41000092403	403	CARD_BLOCKED	Mobile user account has state INACTIVE or no SIM assigned
41000092404	404	NO_KEY_FOUND	Mobile user account needs to be activated
41000092406	406	PB_SIGNATURE_PROCESS	Signature request already in progress.
41000092422	422	NO_CERT_FOUND	Certificate is expired
41000092900	900	INTERNAL_ERROR	Unknown Error

For example, a Profile Query request with MSISDN `+41000092401` will result in a fault `401` as shown below:

REST/JSON
<pre>{ "Fault": { "Code": { "SubCode": { "Value": "401", "ValueNs": "http://uri.etsi.org/TS102204/v1.1.2#" }, "Value": "Receiver", "ValueNs": "http://www.w3.org/2003/05/soap-envelope" }, "Detail": "User cancelled the request", "Reason": "USER_CANCEL" } }</pre>

7 Root CA Certificates (Trust Anchor)

There are two different scenarios, where x.509 certificates are involved. Since they do not have the same root CA, you must ensure that your client's TrustStore contains both root CA certificates.

7.1 Mobile ID X509 Server Certificate

As described in section 2.3, the Mobile ID server's x.509 certificate that is used in the mutual SSL/TLS authentication process is a SwissSign certificate.

You can download the "SwissSign Gold CA - G2" certificate from the SwissSign site:

<https://www.swissign.com/en/support/faq.html>

Key-ID 5B 25 7B 96 A4 65 51 7E B8 39 F3 C0 78 66 5E E8 3A E7 F0 EE

SHA-1 Fingerprint D8 C5 38 8A B7 30 1B 1B 6E D4 7A E6 45 25 3A 6F 9F 1A 27 61

7.2 Mobile ID User X509 Certificate

As described in section 1.2, the main scenario is a strong authentication, where the AP receives a signature response, which includes the signature object and the mobile user's x.509 certificate (public key). The AP should validate the signature as well as the x.509 certificate's trust chain.

The Mobile ID x.509 certificate that is used in the signature process is a Swisscom certificate.

You can download the "Swisscom Root CA 2" certificate from the Swisscom Digital Certificate Service site:

<http://www.swissdigicert.ch>

Key-ID 4D 26 20 22 89 4B D3 D5 A4 0A A1 6F DE E2 12 81 C5 F1 3C 2E

SHA-1 Fingerprint 77 47 4F C6 30 E4 0F 4C 47 64 3F 84 BA B8 C6 95 4A 8A 41 EC

7.2.1 IMPORTANT: New Swisscom Certificate Authority

Swisscom attaches great importance to security standards. For that reason, Swisscom will go live with a new certificate chain of the Mobile ID User certificates. The new certificate authorities meet the latest security requirements and will improve the long-term security of the Mobile ID service.

From **30 June 2022** onwards, Mobile ID users who invoke a new Mobile ID Activation will use a new certificate chain that is built on certificates named `Swisscom Root CA 4` (Root Certificate) and `Swisscom Rubin CA 4` (Intermediate Certificate).

Please ensure that your Mobile ID client's TrustStore contains both the old Root Certificate `Swisscom Root CA 2` as well as the new Root Certificate `Swisscom Root CA 4` by the aforementioned date at the latest.

It is important that you still keep the old certificate(s) in the TrustStore because all Mobile ID users that are already active will keep using the old certificates until it is about to expire in late 2024 (or until they invoke a new activation).

Please refer to the separate document [Swisscom CA 4 Technical Guide](#) for further information and details.

8 Create X509 Client Certificates

Below are examples how to create a self-signed certificate with OpenSSL or Java KeyTool, valid for 5 years.

Please adapt the examples according to your needs. Make sure that you only use SHA256 as shown in the examples below. SHA1 and MD5 are deprecated and shall not be used anymore.

8.1 OpenSSL

8.1.1 Generate Key & Create CSR

```
$ openssl req -new -newkey rsa:2048 -nodes -rand /dev/urandom -keyout mycert.key
-out mycert.csr -sha256 -subj '/CN=mobileid.company.ch/O=Company/C=CH'
```

8.1.2 Self-Sign Certificate

```
$ openssl x509 -req -days 1825 -sha256 -in mycert.csr -signkey mycert.key -out mycert.crt
```

8.1.3 Convert To PKCS#12

Optionally, if you need a PKCS#12 file you can convert the Key and Certificate with this command:

```
$ openssl pkcs12 -export -in mycert.crt -inkey mycert.key -out mycert.p12
```

Provide the `mycert.crt` file to Swisscom and keep your `*.key` file securely stored.

8.2 Java KeyTool

8.2.1 Generate Key & Export Certificate

```
$ keytool -genkey -alias <alias-name> -keyalg RSA -keysize 2048 -validity 1825
-dname 'CN=mobileid.company.ch,O=Company,C=CH' -keystore mycert.jks
$ keytool -export -alias <alias-name> -keystore mycert.jks -file mycert.crt
```

8.2.2 Root CA Cert & Intermediate CA Cert Import

```
$ keytool -keystore truststore.jks -import -file Swisscom_Root_CA_2_der.crt
$ keytool -keystore truststore.jks -import -file Swisscom_Rubin_CA_3_der.crt
```

8.2.3 Useful Commands

```
$ keytool -printcert -v -file mycert.crt
$ keytool -list -v -keystore keystore.jks
$ keytool -list -v -keystore keystore.jks -alias <alias-name>
```

Provide the `mycert.crt` file to Swisscom and keep your `*.jks` file securely stored.

9 Health Status Microservice

The Mobile ID Health Status Service is a microservice where you can retrieve health state details about the MobileID authentication service. The health state is based on real end-to-end authentication tests that are checked every few minutes. It's free of charge!

For more information please visit <https://digital.swisscom.com/products/mobile-id>

Active Probing

Retrieve a detailed health state about the Mobile ID authentication service. The scope of the health status service includes different telecommunications providers, IP+ and LAN-i endpoints and the geofencing service.

The health status service is free of charge!

Real End-To-End Testing

Our health status is based on real end-to-end Mobile ID tests that are responded by special robotic equipment at different physical locations.

MobileID Telecommunications Providers

Checks include Mobile ID SIM cards from Swisscom, Sunrise, Salt and UPC.

MobileID App

Checks include the Mobile ID mobile application.

Health Status Levels

Each test objective will have one of three possible status levels:

- **SUCCESS**
All tests of that objective (e.g. Operator "Swisscom) are currently successful.
- **WARNING**
Some (but not all) tests of that objective failed for 1 or more times in a row. However, some tests of that objective are still working fine.
- **ERROR**
All tests of that objective failed. Mobile ID operation team has been alerted to check the system.