

Mobile ID Integration Guide

OpenID Connect - Version 1.2



Table of contents

1	Introduction	4
1.1	Basic Key Concepts - Terminology	4
1.2	Authorization Code Grant Flow	5
1.2.1	Refresh Token	6
2	Getting Started.....	7
2.1	Endpoint URIs.....	8
2.2	Authorization Code Request.....	9
2.2.1	Scopes and Claims.....	12
2.2.2	Authentication Context Class Reference (ACR).....	13
2.2.2.1	MID SN Check - AL4 Guide	14
2.2.3	Example Authorization Code Request	14
2.3	Authorization Code Response	15
2.3.1	Example Authorization Code Response	15
2.4	Access Token Request.....	16
2.4.1	Example Access Token Request	16
2.5	Access Token Response	17
2.5.1	Authentication Method Reference (AMR)	17
2.5.2	Example Access Token Response	18
2.6	User Info Request	19
2.6.1	Example User Info Request	19
2.7	User Info Response	20
2.7.1	Example User Info Response	20
2.8	Refresh Request.....	21
2.8.1	Example Refresh Request	21
2.9	Refresh Response	22
2.9.1	Example Refresh Response	22
2.10	Error Scheme	23
2.10.1	Error Code Table	24
2.11	Tokens	26
3	Best Practices	27
3.1	Pushed Authorization Request (PAR)	27
3.1.1	Back-channel submission of authorisation parameters.....	27
3.1.2	PAR Endpoint.....	27
3.1.3	PAR Request (POST).....	27
3.1.4	PAR Response	28
3.2	Token and response validation	29
3.2.1	Authentication Request	29
3.2.2	Validate authentication responses	29
3.2.3	Validate token endpoint responses.....	29
3.2.4	Validate ID token.....	29
3.2.5	Protect Client ID and secret	29
3.2.6	Store tokens securely.....	29
3.3	Test Users.....	30
4	Public Cloud Integration Guide	31
4.1	Microsoft Azure ADB B2C	31
4.2	Amazon Cognito.....	31
5	MobileID OIDC - Use Cases	32
5.1	Prompt user for MSISDN.....	32
5.2	RP knows the MSISDN.....	33
5.3	Prompt user for user credentials	34
5.4	RP knows the username.....	35

- 6 Message Formats on the Mobile ID App36
 - 6.2 Classic DTBD 37
 - 6.3 Transaction Approval..... 37
 - 6.4 Examples 38
 - 6.4.1 Pretty JSON 38
 - 6.4.2 Single-line 38
 - 6.4.3 Authorization request (excerpt) 38
 - 6.5 What exactly is signed (classic vs. Transaction Approval)..... 38
 - 6.6 Best practices 38

1 Introduction

This document provides Relying Parties (RPs) with technical guidance and best practices for integrating Mobile ID Open ID Provider (MobileID OP) into their applications.

OpenID Connect

The MobileID OP can be used for both authorization and authentication. It fully complies with the [OpenID Connect specification](#).

OpenID Connect is a simple identity layer on top of the OAuth 2.0 protocol. It allows clients to verify the identity of the End-User based on the authentication performed by an Authorization Server, as well as to obtain basic profile information about the End-User in an interoperable and REST-like manner.

OpenID Connect allows clients of all types, including Web-based, mobile, and JavaScript clients, to request and receive information about authenticated sessions and end-users. The specification suite is extensible, allowing participants to use optional features such as encryption of identity data, discovery of OpenID Providers, and session management, when it makes sense for them.

See <https://openid.net/connect/faq/> for a set of answers to Frequently Asked Questions.

The Mobile ID solution protects access to your company data and applications with a comprehensive end-to-end solution for a strong multi-factor authentication (MFA). Please visit <https://mobileid.ch> for further information. Do not hesitate to [contact us](#) in case of any questions.

1.1 Basic Key Concepts - Terminology

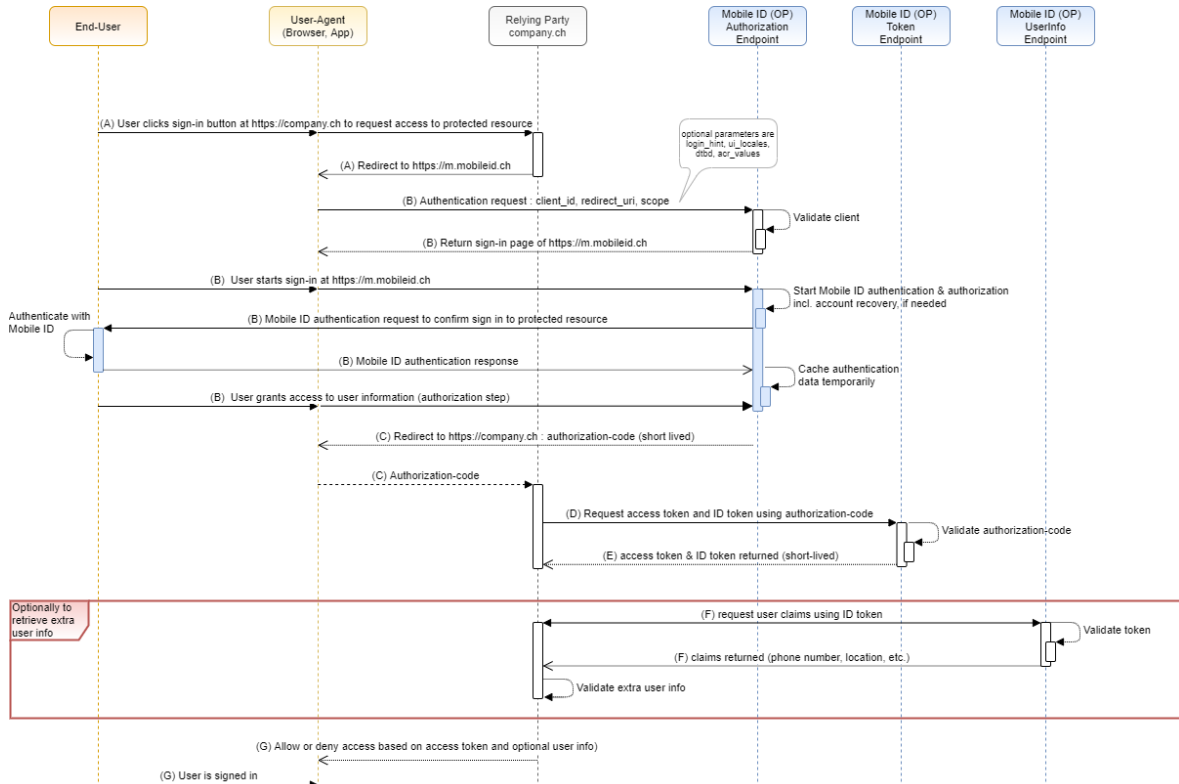
The most basic key concepts are as follows.

- **End-User** is the entity that wants to assert a particular identity, the Mobile ID User (a human being).
- **User-Agent** is the program (such as a browser) used by the End-User to communicate with the Relying Party and OpenID Connect Provider.
- **Relying Party (RP)** is our customer's web site or client application that wants to verify the End-User's identifier. It outsources its user authentication function to an OpenID Connect Provider. It can request claims (e.g., user information) about that End-User.
- **OpenID Connect Provider (OP)** is an authorization server that offers authentication as a service and providing claims to a Relying Party about the authentication event and the End-User.
- **Scopes** are identifiers used to specify what access privileges are being requested.
- **Claims** are simply key & value pairs that contain information about an End-User, as well meta-information about the authentication event. Non-standard claims can be specified as custom claims.
- **Access Token** are credentials used to access protected resources directly. Access tokens usually have an expiration date and are short-lived. They must be kept secret, though security considerations are less strict due to their shorter life.
- **ID Token** is an identity token provided by the OpenID Provider to the Relying Party. The identity token contains a number of claims about that End-User and also attributes about the End-User authentication event, in a standard JWT¹ format and signed by the OpenID Provider (so it can be verified by the intended recipients). It may optionally be encrypted for confidentiality.
- **Refresh Token** carries the information necessary to get a new access token. Refresh tokens can also expire but are rather long-lived.

¹ <https://tools.ietf.org/html/rfc7523>

1.2 Authorization Code Grant Flow

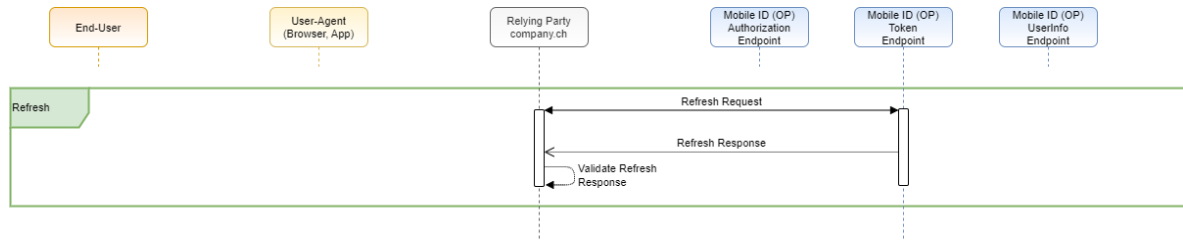
Mobile ID utilizes the Authorization Code Grant Type to obtain an access token to grant application to retrieve user data after authenticating. The Authorization Code Flow, in abstract, follows the following steps:



- A. An End-User requests access to a protected resource of the Relying Party. For example, the End-User clicks on a sign-in button or launches an app. The Relying Party redirects the User-Agent to the Mobile ID Authorization Endpoint at <https://m.mobileid.ch>.
- B. The Mobile ID Authorization Server authenticates the End-User (using an appropriate Mobile ID authentication method) and establishes whether the End-User grants or denies the Relying Party Client's access request. Including access to extra user information that might have been requested in the scope of the authentication request sent by the Relying Party.
- C. The Mobile ID Authorization Server redirects the User-Agent back to the Relying Party's web site or application using the redirection URI provided earlier in step A. The redirection URI includes an authorization code that is valid for a few minutes.
- D. Relying Party requests an Access Token from the Mobile ID Token Endpoint by including the authorization code received in the previous step. When making the request, the Relying Party authenticates with the Mobile ID server. The client includes the redirection URI used to obtain the authorization code for verification.
- E. The Mobile ID server authenticates the client, validates the authorization code, and ensures that the redirection URI received matches the URI used to redirect the client in step C. On success, the Mobile ID server will return a JSON object with the ID Token and an Access Token, and an optional Refresh Token.
- F. Optional: Relying Party requests additional, consented user information (claims). The user info endpoint returns previously consented user profile information to the client. A valid access token is required for that.
- G. Relying Party grants or denies access to the requested service and the user is logged in.

1.2.1 Refresh Token

Because Access Tokens are always short-lived (see chapter 2.11), a Relying Party may want to refresh the Access Token using their long-lived Refresh Token. To refresh an Access Token, the Client must always authenticate at the Token Endpoint. The Relying Party should always validate the Refresh Response.



Note that a Relying Party must be authorized to use a refresh token.

2 Getting Started

Before you can integrate and use Mobile ID OpenID Connect sign-in, the client on-boarding process must have been completed by Swisscom.

For the technical on-boarding, you will be asked to provide Swisscom following information:

What	Quick Description	Ref.
Client Display Name	Your client's name, which is displayed by the authorization server. Example value: <code>iDemo Online Shop</code>	
Redirect URI(s)	Redirection URI(s) to which the response will be sent. Note that TLS (https) is always required and localhost URI is not allowed. Example value: <code>https://app01.idemo-company.ch/oauth2/authresp</code> <code>https://app02.idemo-company.ch/oauth2/authresp</code>	oidc spec
Default ACR	Your default ACR. Must be a value that is available for your selected Mobile ID contract. Example value: <code>mid_al3_any</code>	2.2.2
Client Auth Mode	Your client's authentication method, either basic or post . Example value: <code>client_secret_post</code>	2.4
Always Prompt For Consent	The Mobile ID server default behaviour is to skip the consent step, provided such is already recorded for the given end-user and client. Default: <code>false</code>	
MFA Number Matching	Enable MFA number matching feature for Mobile ID SIM and Mobile ID App authentication. When a user responds to an MFA notification using Mobile ID SIM or Mobile ID App, they'll be presented with a number on their mobile. They need to select that number in the sign-in prompt to complete the approval. This can significantly improve security for MFA. Default: <code>false</code>	video
LDAP Settings	Optional. Mobile ID server can connect to an LDAP(S) to validate user credentials and/or retrieve user attributes from the LDAP, such as: <ul style="list-style-type: none"> - MFA mobile number attribute - Mobile ID Serialnumber attribute (required for ACR <code>mid_al4</code>) - User password attribute 	
CNAME Record	Optional. Mobile ID server can use a custom domain instead of default <code>m.mobileid.ch</code> . Custom Domains are only relevant if <code>prompt=login</code> is used. We will need your record name (e.g. <code>mobileid.acme.com</code>) that routes the traffic to <code>m.mobileid.ch</code> .	

You will get a unique OIDC client identifier and client secret from Swisscom.

If you did not receive your client credentials, it means that your on-boarding process is not finished yet. Please check the state with your commercial contact or via Backoffice.Security@swisscom.com.

2.1 Endpoint URIs

A default Mobile ID OpenID Provider configuration is published on the OIDC discovery endpoint, which allows a client to discover the OAuth 2.0 and OpenID Connect endpoints, capabilities, supported cryptographic algorithms and features.

It is recommended to host a local copy of this file when your application relies on constant availability of this endpoint data.

Endpoint	URL
Discovery	https://openid.mobileid.ch/.well-known/openid-configuration

Additional endpoint URIs:

Endpoint	URL
Authorization	https://m.mobileid.ch/oidc/authorize
Token	https://openid.mobileid.ch/token
User Info	https://openid.mobileid.ch/userinfo
Pushed Authorization Requests	https://openid.mobileid.ch/par

2.2 Authorization Code Request

The authorization code can be obtained by performing a simple HTTP GET request towards the Authorization Code endpoint of the Mobile ID OP. The client secret is not involved yet.

Endpoint	URL
Authorization	https://m.mobileid.ch/oidc/authorize

The Relying Party may trigger the authorization code flow by calling the authorization link (including required request parameters), for example:

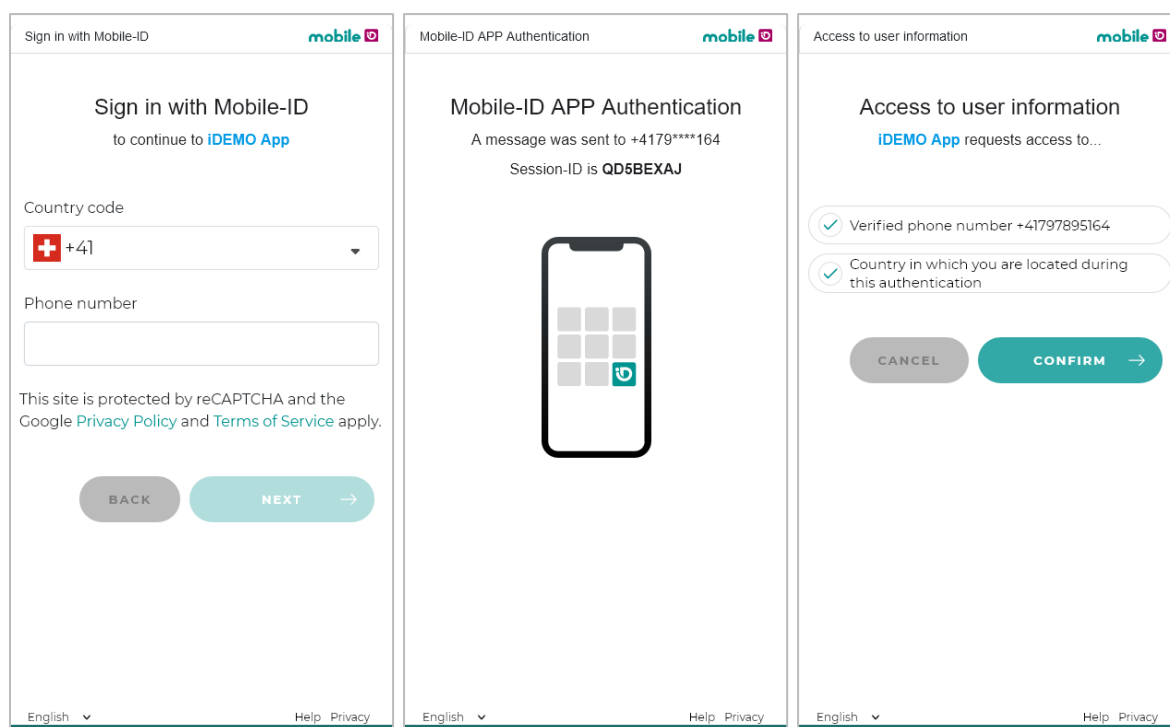
```
<a href="https://m.mobileid.ch/oidc/authorize?response_type=code&scope=openid&client_id=s6BhdRkqt3&state=af0ifjsldkj&redirect_uri=https%3A%2F%2Fcompany.ch%2Fcb" rel="noreferrer">MobileID-sign-in-button</a>
```

For data privacy reasons, it is highly recommended that the Relying Party does not fill in the HTTP referrer header. Typically, the referrer header is populated with the address of the page where the request originated. The HTML hyperlinks should have the `rel` attribute set to `noreferrer`, as shown in the example above.

A button could look like this example:



A click on the button will redirect the user to the mobileid.ch domain, where (s)he can complete the authorization code flow. In the example screenshot below, the user enters the phone number, authenticates with the Mobile ID App and consents to the user information (phone number, current location) that was requested by the Relying Party "iDemo App". Finally, the user is redirected back to the Relying Party's domain.


 Three screenshots of the Mobile ID authentication process.
 1. "Sign in with Mobile-ID": A form to enter a country code (dropdown with "+41" selected) and a phone number. It includes a reCAPTCHA notice and "BACK" and "NEXT" buttons.
 2. "Mobile-ID APP Authentication": A screen showing a message sent to "+4179****164" and a session ID "QD5BEXAJ". It features a graphic of a smartphone with a grid of dots and the Mobile ID logo.
 3. "Access to user information": A screen showing consent for "iDemo App" to access user information. It lists "Verified phone number +41797895164" and "Country in which you are located during this authentication". It has "CANCEL" and "CONFIRM" buttons.

The following request query string parameters are supported:

Parameter	Value	Remarks
scope	MUST contain the value <code>openid</code>	Optionally, the client may request additional scopes as specified in 2.2.1
response_type	MUST be the value <code>code</code>	
client_id	MUST be your client ID	
redirect_uri	MUST be (one of) your redirect URI(s)	
state	MUST be an opaque value	Used to maintain state between the request and the callback
nonce	MUST be a nonce value	
acr_values²	CAN be set to overwrite the client's default ACR	Authentication Level (AL) required by the client (see 2.2.2)
response_mode	SHALL NOT be used	
display	SHALL NOT be used	
prompt	CAN be set to <code>login</code>	
max_age	SHALL NOT be used	
ui_locales	CAN be set to provide the user's preferred language	Supported values are <code>en</code> , <code>de</code> , <code>fr</code> and <code>it</code>
id_token_hint	SHALL NOT be used	
claims	SHALL NOT be used	
login_hint²	<p>CAN be set to provide a login hint to the authorization server about the end-user's phone number(s) or LDAP username. Must be in a JSON format. Valid examples:</p> <pre> {"enableManualInput": true, "hints": [{"msisdn":"+41765XXXXXX"}]} {"enableManualInput": true, "hints": [{"msisdn":"+41765XXXXXX", "default":true}]} {"enableManualInput": true, "hints": [{"msisdn":"+41765XXXXXX", "sn":"+574XXXXXX"}]} {"enableManualInput": true, "hints": [{"msisdn":"+41765YYYYYY"}, {"msisdn":"+41765XXXXXX", "default":true}]} </pre> <p>If an LDAP has been configured for your OIDC client, you can use the following login hints</p> <pre> {"useLDAP": true, "hints": [{"userName":"johndoe"}]} {"useLDAP": true, "hints": [{"userName":"john.doe@amce.com", "userPassword":"plain-secret", "isHashed": false}]} {"useLDAP": true, "hints": [{"userName":"john.doe@amce.com", "userPassword":"hashed-secret", "isHashed": true}]} </pre> <p>You can use the following login hint in combination with the <code>prompt=login</code> parameter:</p> <pre> {"useLDAP": true} </pre>	<p>The "sn" parameter is optional and only required for ACR <code>mid_a14</code></p>

² Your account must be authorized to use this parameter. It requires the use of Pushed Authorisation Rquests (PAR), which keeps the parameters confidential between client and server (see chapter 3.1).

Parameter	Value	Remarks												
dtbd ²	<p>CAN be set to overwrite the default authentication message that is displayed to the end-user if the authentication method is either Mobile ID SIM or Mobile ID App.</p> <p>The message should include these keywords:</p> <table><tr><td>#SESSION#</td><td><p>A unique transaction number.</p><p>In case MFA Number Matching is enabled, this keyword will be replaced with the matching number.</p></td></tr><tr><td>#CLIENT#</td><td>Relying Party Display Name</td></tr></table> <p>The default authentication message is:</p> <table><tr><td>English</td><td>Do you want to login to #CLIENT#? Transaction number #SESSION#</td></tr><tr><td>German</td><td>Möchten Sie sich bei #CLIENT# anmelden? Transaktion Nummer #SESSION#</td></tr><tr><td>French</td><td>Voulez-vous vous connecter à #CLIENT# ? Transaction numéro #SESSION#</td></tr><tr><td>Italian</td><td>Vuoi accedere a #CLIENT#? Transazione numero #SESSION#</td></tr></table>	#SESSION#	<p>A unique transaction number.</p> <p>In case MFA Number Matching is enabled, this keyword will be replaced with the matching number.</p>	#CLIENT#	Relying Party Display Name	English	Do you want to login to #CLIENT#? Transaction number #SESSION#	German	Möchten Sie sich bei #CLIENT# anmelden? Transaktion Nummer #SESSION#	French	Voulez-vous vous connecter à #CLIENT# ? Transaction numéro #SESSION#	Italian	Vuoi accedere a #CLIENT#? Transazione numero #SESSION#	
#SESSION#	<p>A unique transaction number.</p> <p>In case MFA Number Matching is enabled, this keyword will be replaced with the matching number.</p>													
#CLIENT#	Relying Party Display Name													
English	Do you want to login to #CLIENT#? Transaction number #SESSION#													
German	Möchten Sie sich bei #CLIENT# anmelden? Transaktion Nummer #SESSION#													
French	Voulez-vous vous connecter à #CLIENT# ? Transaction numéro #SESSION#													
Italian	Vuoi accedere a #CLIENT#? Transazione numero #SESSION#													

2.2.1 Scopes and Claims

Given below is the list of supported scopes that can be requested during the authorization code request.

Scope	Member (Claims)	Type
openid	sub ³	string
offline_access	-	-
profile	name ⁴	string
phone	phone_number phone_number_verified	string boolean
mid_location ⁵	mid_geo_accuracy mid_geo_country mid_geo_device_confidence mid_geo_location_confidence mid_geo_timestamp	number string number number string
mid_profile	mid_profile_recovery_code_status mid_profile_serial mid_profile_sim_status mid_profile_sim_pin_status mid_profile_sim_mcc mid_profile_sim_mnc mid_profile_sim_network mid_profile_app_status	boolean string string string string string string string
mid_cms	mid_cms_content	string
mid_esign_basic	mid_esign_basic_assurance_level mid_esign_basic_jurisdictions mid_esign_basic_has_valid_evidence	string string boolean

A Relying Party should always respect the user's privacy and keep the requested claims down to the very essential. For example, using scope `openid` only, the user sign-in will be anonymous. Neither the phone number nor any other user information will be passed on to the Relying Party's application.

Example claim values, retrieved from UserInfo endpoint:

```
{
  "mid_geo_accuracy": 0,
  "mid_geo_country": "CH",
  "mid_geo_device_confidence": "1.0",
  "mid_geo_location_confidence": "1.0",
  "mid_geo_timestamp": "2022-03-17T05:49:03.597+01:00",
  "mid_profile_recovery_code_status": true,
  "mid_profile_serial": "MIDCHEYUD1YE4QB1",
  "name": "+41797895164",
  "phone_number": "+41797895164",
  "phone_number_verified": true,
  "sub": "3246d772d2edb20797fe9359cb3d07da6d01df7db2642e14554d241aef1d1d84"
}
```

³ Subject Identifier. A locally unique and never reassigned identifier within the Issuer (Mobile ID) for the End-User, which is intended to be consumed by the Client. See https://openid.net/specs/openid-connect-core-1_0.html#IDToken






⁴ End-User's display name, a unique identifier. If `phone` scope was requested, the value of the `name` claim is the MSISDN, such as "+41791234567". If no `phone` scope was requested, it is an identifier that includes the last 6 digits of the sub, such as "User48e705".

⁵ The scope `mid_location` enables Relying Parties to define geographical boundaries. They can decide who can access what within that barrier, based on the user's location data determined at the moment defined in the `mid_geo_timestamp` claim. More details can be found in the [Mobile ID Reference Guide](#) chapter "Geofencing".

2.2.2 Authentication Context Class Reference (ACR)

Below is an overview of all authentication means offered and supported by Mobile ID OP. The ACR can be requested with an authorization request parameter - or, if the request does not contain such parameter, the client's default ACR is selected.

An ACR can include one or several different authentication methods. The Mobile ID OP will check the user's authentication possibilities and will select an authentication method that complies with the ACR.

Authentication Level (AL)	ACR value	 SIM Card	 Mobile App	 OTP Text SMS	 CH Loc. Check ⁶	 MID SN Check
2	mid_al2_any	✓	✓	✓		
3	mid_al3_any	✓	✓			
	mid_al3_simcard	✓				
	mid_al3_mobileapp		✓			
	mid_al3_any_ch	✓	✓		✓	
4	mid_al4_any	✓	✓			✓
	mid_al4_simcard	✓				✓
	mid_al4_mobileapp		✓			✓
	mid_al4_any_ch	✓	✓		✓	✓

If a user has more than one authentication method available that comply with the requested ACR, the Mobile ID OP will use the following preference (note, all authentication methods are equally billed):

Prio	Condition (User Mobile ID Account Status)	AL "any" choice
1	Active Mobile ID SIM card	Mobile ID SIM
2	Inactive Mobile ID SIM card & Inactive Mobile ID App	Mobile ID SIM (Account Recovery)
3	Unknown SIM ⁷ card & Active Mobile ID App	Mobile ID App
4	Unknown SIM ⁷ card & Inactive Mobile ID App	AL2: One-Time-Password SMS AL3: Mobile ID App (Account Recovery) AL4: Mobile ID App (Account Recovery)

If your client attempts to request a scope that is not available with your current Mobile ID contract, an OIDC error `mid_sec_2020` ("unauthorized acr_values used in request", see chapter 2.10) is returned.

Please check with your commercial contact or via Backoffice.Security@swisscom.com if you are interested in ACR values that are listed in the table above but currently not configured for your Mobile ID OIDC account.

⁶ ANY_CH will verify if the user is in Switzerland, with minimum device- and location confidence score 0.7 and location accuracy <500m

⁷ An unknown SIM card is any SIM card that is not Mobile ID compatible.

2.2.2.1 MID SN Check - AL4 Guide

Using an ACR value of Authentication Level 4 will provide the highest and most secure authentication level.

The use of such ACR will require the Relying Party to always provide the Mobile ID's serial number via the `login_hint` authorization request parameter, if there is no LDAP configured. If an LDAP is configured, the Mobile ID server will lookup the user's serial number on the LDAP. The Mobile ID authorization server will compare the serial number with the serial number of the authentication response.

The authentication will be successful if the serial numbers matched. A serial number mismatch will result in an authentication failure.

An account recovery (e.g., activate Mobile ID App) is supported and will result in a successful authentication if a valid user restore option was available used during the recovery process.

Therefore, the Relying Party must implement its own Mobile ID user registration process to fetch and store the user's current Mobile ID serial number value. Technically, the best way to retrieve the user's current Mobile ID serial number is by starting an authorization request using authentication level 3 (for example `mid_al3_any`) and scope `mid_profile`, so that the Relying Party will retrieve the claim `mid_profile_serial`.

2.2.3 Example Authorization Code Request

```
HTTP/1.1 302 Found
Location: https://m.mobileid.ch/oidc/authorize?
    response_type=code
    &scope=openid%20auth_basic
    &client_id=s6BhdRkqt3
    &state=af0ifjsldkj
    &redirect_uri=https%3A%2F%2Fclient.example.org%2Fcb
```

2.3 Authorization Code Response

The response to the authorize request will be a **HTTP redirect** that results in a HTTP GET request to the `redirect_uri` that was provided for the authorization code request with the following parameters:

Parameter	Value
code	WILL contain the authorization code
state	WILL contain the state
iss	WILL contain the issuer of the authorization code: <code>https://openid.mobileid.ch</code>

Note: The authorization code is by default only valid for **10 seconds** after it has been issued.

2.3.1 Example Authorization Code Response

```
HTTP/1.1 302 Found
Location: https://client.example.org/cb?
    code=Splx10BezQQYbYS6WxSbIA
    &iss=https%3A%2F%2Fopenid.mobileid.ch
    &state=af0ifjsldkj
```

The Relying Party must validate the state parameter and use the code to proceed to the next step: Exchanging the access code for the Access Token (2.4).

2.4 Access Token Request

The obtained authorization code obtained (2.3) can be used to receive an access token. This chapter describes how to format the access token request and which data is returned from the access token endpoint.

The OpenID Connect RFC states that there are 4 possible client authentication methods (used by clients to authenticate to the Authorization Server when using the Token Endpoint).

Mobile ID OP supports `client_secret_basic` or `client_secret_post` as client authentication method.

The access token can be obtained by performing a HTTP **POST** request towards the Token endpoint of the Mobile ID OP.

Endpoint	URL
Token	https://openid.mobileid.ch/token

The following query string parameters need to be embedded in the request:

Parameter	Value
grant_type	MUST be the value <code>authorization_code</code>
code	MUST be the authorization code you received from Mobile ID (see 2.3)
redirect_uri	MUST be the same redirect URI used in the authorization code request

Headers	Value
Authorization	Client ID and client secret encoded according to the HTTP Basic authentication scheme
Content-Type	MUST be the value <code>application/x-www-form-urlencoded</code>

2.4.1 Example Access Token Request

In this example, basic authentication is used to authenticate the client to the Mobile ID server.

```
POST /token HTTP/1.1
Host: openid.mobileid.ch
Content-Type: application/x-www-form-urlencoded
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmFOM2JW
grant_type=authorization_code
&code=Sp1x10BeZQQYbYS6WxSbIA
&redirect_uri=https%3A%2F%2Fclient.example.org%2Fcb
```


2.5 Access Token Response

After receiving and validating a valid and authorized token request from the client, the Authorization Server returns a successful response that includes:

- Access Token
- ID Token

The response is returned in the `application/json` media type.

The ID token is a JWT and is created (and thus signed, RS256 by default) by the Authorization Server itself.

2.5.1 Authentication Method Reference (AMR)

Authentication Method Reference (AMR) is an attribute within the OpenID Connect Identity Token. The AMR claim makes statements about the authentication method that was used (including additional factors such as geolocation).

AMR Value	SIM Auth	App Auth	OTP Auth
mid_app		✓	
mid_geo ⁸	(✓)	(✓)	
mid_hwk	✓	✓	
mid_otp			✓
mid_sim	✓		
mid_sms			✓

The AMR can be helpful in case the client requests an ACR with an “any” value, such as `mid_a13_any` (see chapter 2.2.2). Since there are multiple authentication methods that comply with such ACR, the client will know from the AMR what authentication method the user actually used for the sign-in.

⁸ The AMR `mid_geo` is only set in case of ACR `mid_a13_any_ch` or `mid_a14_any_ch`, because this ACR is based on geofencing data.

2.5.2 Example Access Token Response

```
HTTP/1.1 200 OK
Content-Type: application/json
{
  "access_token": "eyJraWQiOiJDWHVwIiwidH",
  "scope": "phone openid profile mid_location",
  "id_token": "eyJraWQiOiJDWHVwIiwYXnIj",
  "token_type": "Bearer",
  "expires_in": 600
}
```

Note, the content of the access token is meant for consumption by the `/userinfo` endpoint where the client can give the token to get the user consented claim values. The access token is not meant to convey information to the client or be peeked into by the client, only to the accessed protected resources.

2.6 User Info Request

The `/userinfo` endpoint is an OAuth 2.0 protected resource of the Mobile ID server where client applications can retrieve consented claims, or assertions, about the logged in end-user.

Clients must present a **valid access token** (of type bearer) to retrieve the UserInfo claims. Only those claims that are scoped by the token will be made available to the client.

The user info can be obtained by performing a HTTP **GET** or HTTP **POST** request towards the User Info endpoint of the Mobile ID server.

Endpoint	URL
User Info	https://openid.mobileid.ch/userinfo

The following parameters should be added:

Headers	Value
Authorization	MUST contain the Bearer 'Access-Token'

2.6.1 Example User Info Request

```
GET /userinfo HTTP/1.1
Host: openid.mobileid.ch
Authorization: Bearer sa7Aebo6Ookoo0oh
```

2.7 User Info Response

The user info response is a signed JWT token which will contain claims based on the requested scopes in the authorization code request.

If the access token is still valid, the Mobile ID OP will return a JWT user info token.

2.7.1 Example User Info Response

```
HTTP/1.1 200 OK
Content-Type: application/json
{
  "mid_geo_accuracy": 0,
  "mid_geo_location_confidence": "1.0",
  "sub": "3246d772d2edb20797fe9359cb3d07da6d01df7db2642e14554d241aef1d1d84",
  "mid_geo_device_confidence": "1.0",
  "name": "+41791234567",
  "phone_number_verified": true,
  "phone_number": "+41791234567",
  "mid_geo_timestamp": "2022-03-16T13:47:26.466+01:00",
  "mid_geo_country": "CH"
}
```

2.8 Refresh Request

A request to the Token Endpoint can also use a Refresh Token by using the `grant_type` value `refresh_token`.

To refresh an Access Token, the Client MUST authenticate to the Token Endpoint using the authentication method registered for its `client_id`. The Client sends the parameters via HTTP POST to the Token Endpoint using Form Serialization.

Endpoint	URL
Token	https://openid.mobileid.ch/token

The following query string parameters need to be embedded in the request:

Parameter	Value
<code>grant_type</code>	MUST be the value <code>refresh_token</code>
<code>client_secret</code>	MUST be your client secret
<code>refresh_token</code>	MUST be your refresh token
<code>scope</code>	MUST be the scope

Headers	Value
<code>Content-Type</code>	MUST be the value <code>application/x-www-form-urlencoded</code>

2.8.1 Example Refresh Request

```
POST /token HTTP/1.1
Host: openid.mobileid.ch
Content-Type: application/x-www-form-urlencoded

client_id=s6BhdRkqt3
&client_secret=some_secret12345
&grant_type=refresh_token
&refresh_token=8xLOxBtZp8
&scope=openid%20profile
```

2.9 Refresh Response

Upon successful validation of the Refresh Token, the response body is the Token Response.

2.9.1 Example Refresh Response

```
HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
Pragma: no-cache

{
  "access_token": "TlBN45jURg",
  "token_type": "Bearer",
  "refresh_token": "9yNOxJtZa5",
  "expires_in": 3600
}
```

2.10 Error Scheme

The Mobile ID authorization service implements an error code scheme, in which various error are reported back to the Relying Party (client) together with a human-friendly message, a unique error code and the trace-id of the session.

All fault responses contain an `errorCode`-field and a `description`-field. The description contains a structure as follows:

```
mid_<Category>_<Code>_<Trace> - <Message>
```

Example fault response:

```
{
  "errorCode" : "unauthorized_client",
  "description" : "mid_sec_20I0_A9W1GLUM - Unauthorized scopes used in request"
}
```

Description of description structure details:

mid	Schema prefix for codes that belong to Mobile ID
<Category>	One of... req = issues with request validation sec = security issues with the request auth = issues with user authentication or activation sys = system-related issues such as internal errors
<Code>	An error code specifically related to the sec req auth sys category. See 2.10.1.
<Trace>	the trace/session ID uniquely created for that authentication flow. This value is the same as the one printed on all internal log messages + the same ID as the one shown on user interaction (e.g., Session-ID in the authentication message).
<Message>	the human-friendly message explaining the problem

2.10.1 Error Code Table

This table presents all the error codes that are currently implemented.

OIDC error	Code	Human-friendly text / Description
invalid_request	mid_req_1010	Invalid acr_values parameter, expected one single value
invalid_request	mid_req_1020	Invalid value received for acr_values
invalid_request	mid_req_1030	Invalid ui_locales parameter, expected one single value
invalid_request	mid_req_1040	Invalid value received for ui_locales
invalid_request	mid_req_1050	Invalid login_hint, empty hits are not allowed
invalid_request	mid_req_1060	Invalid login_hint, AL4 cannot be used with enabled manual MSISDN input
invalid_request	mid_req_1070	Invalid MSISDN value in login_hint
invalid_request	mid_req_1080	Duplicated MSISDN value in login_hint
invalid_request	mid_req_1090	Invalid SN value in login_hint
invalid_request	mid_req_1100	Invalid login_hint JSON content
invalid_scope	mid_req_1110	Invalid scopes in request
invalid_request	mid_req_1120	AL4 requested but login_hint is empty
invalid_request	mid_req_1130	Invalid request, missing query string
invalid_request	mid_req_1900	Invalid client request, check request parameters
unauthorized_client	mid_sec_2010	Unauthorized scopes used in request
unauthorized_client	mid_sec_2020	Unauthorized acr_values used in request
unauthorized_client	mid_sec_2030	Unauthorized parameters used in request
access_denied	mid_auth_3010	Authentication rejected by resource owner or authorization server
access_denied	mid_auth_3011	Authentication rejected by resource owner or authorization server. (1) Number-Matching successful and (2) MID request cancelled by the user
access_denied	mid_auth_3012	Authentication failed as user did not respond. (1) Number-Matching successful and (2) MID requested timed out.
access_denied	mid_auth_3013	Authentication failed due to number mismatch. (1) Number-Matching failed and (2) MID request successful.
access_denied	mid_auth_3014	Authentication rejected by resource owner or authorization server. (1) Number-Matching failed and (2) MID request cancelled by the user.
access_denied	mid_auth_3015	Authentication failed as user did not respond. (1) Number-Matching failed and (2) MID request timed out
access_denied	mid_auth_3020	Claims sharing rejected by resource owner or authorization server
access_denied	mid_auth_3025	Signature CMS data validation failed
access_denied	mid_auth_3030	Mobile ID serial number validation failed
access_denied	mid_auth_3040	Country (geo-location) validation failed
access_denied	mid_auth_3050	MSISDN ownership verification failed
access_denied	mid_auth_3060	Mobile ID account activation failed
access_denied	mid_auth_3070	Mobile ID SIM card required for this authentication
access_denied	mid_auth_3080	No authentication method available
access_denied	mid_auth_3090	Authentication via SMS OTP failed
access_denied	mid_auth_3300	Authentication failed; user did not respond

access_denied	mid_auth_3310	Authentication failed; user is busy with another authentication
access_denied	mid_auth_3900	Authentication failed for other reasons
server_error	mid_sys_9900	Internal server error

2.11 Tokens

This table provides details on OIDC and OAUTH tokens and how the Relying Party should manage them.

Token	Type	TTL	Revo- cable	Refresh- able	Valida- tion	Stor- able	Usage
Access Token	bearer	60 min	no	yes	yes	no	for calling userinfo
Authorization Code	string	2 min	no	no	no	no	For obtaining the access_token, the id_token and refresh_token
ID Token	JWT	60 min	no	yes	yes	no	Proof of IdP's authentication. Expired ID tokens should never be accepted for processing
Refresh Token	bearer	configurable per client	yes	yes	no	yes	for obtaining new valid refresh_token, access_token and id_token

3 Best Practices

3.1 Pushed Authorization Request (PAR)

3.1.1 Back-channel submission of authorisation parameters

The Pushed Authorisation Request (PAR) endpoint gives OAuth 2.0 clients a back-channel to post the parameters of an authorisation request to the Mobile ID server, to obtain an opaque URI handle for them, and then continue with the frontend redirection to the authorisation endpoint as usual.

Introducing an extra backend call to submit the authorisation parameters has three benefits:

- Frees the authorisation request from any browser URL length limits. They can become an issue with complex requests, such as [RAR](#).
- Keeps the parameters confidential between client and server. Regular requests expose them in the URL query string and hence to the browser, the end-user and logs.
- Confidential OAuth clients will be authenticated up-front, and the request parameters will be checked for errors, before sending the end-user to the authorisation endpoint for login and consent.

PAR is specified in [RFC 9126](#).

3.1.2 PAR Endpoint

It can be found out from the `pushed_authorization_request_endpoint` advertised in the Mobile ID server metadata and has this address:

```
https://openid.mobileid.ch/par
```

3.1.3 PAR Request (POST)

Submits the OAuth 2.0 authorisation request parameters to `/par` to obtain a `request_uri` handle for use at the authorisation endpoint.

Header parameters:

- **[Authorization]** Used for HTTP basic authentication of the client.
- **Content-Type** Must be set to `application/x-www-form-urlencoded`.
- **[Issuer]** The issuer URL
- **[Tenant-ID]** The tenant ID

Body:

- The OAuth 2.0 authorisation request parameters, together with any client authentication parameters.

Success:

- **Code:** `200`
- **Content-Type:** `application/json`
- **Body:** {object} The PAR response.

Example PAR request for a confidential client registered for `client_secret_basic` authentication:

```
POST /par HTTP/1.1
Host: c2id.com
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW
Content-Type: application/x-www-form-urlencoded

response_type=code
&scope=openid%20email
&client_id=fcb5e4f1
&state=af0ifjsldkj
&redirect_uri=https%3A%2F%2Fclient.example.org%2Fcb
```

Example PAR request for a public client with PKCE:

```
POST /par HTTP/1.1
Host: c2id.com
Content-Type: application/x-www-form-urlencoded

response_type=code
&scope=openid%20email
&client_id=fcb5e4f1
&state=af0ifjsldkj
&redirect_uri=https%3A%2F%2Fclient.example.org%2Fcb
&code_challenge_method=S256
&code_challenge=E9Melhoa2OwvFrEMTJguCHaoeK1t8URWbuGJSstw-cM
```

Example response with a request URI to complete the authorisation:

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache

{
  "request_uri" : "urn:ietf:params:oauth:request_uri:OsL1Z3VqIxAT9R77wB7KCw.5-
cQUnt9DygE4XxnYjysnw",
  "expires_in" : 60
}
```

3.1.4 PAR Response

JSON object with members:

- **request_uri** The request URI handle to use at the authorisation endpoint.
- **expires_in** The configured lifetime of the request URI, in seconds.

Example:

```
{
  "request_uri" : "urn:ietf:params:oauth:request_uri:OsL1Z3VqIxAT9R77wB7KCw.5-
cQUnt9DygE4XxnYjysnw",
  "expires_in" : 60
}
```

3.2 Token and response validation

According to the OIDC specification RPs must ensure that the received tokens are valid. The following chapters summarize the Relying Party's obligations when consuming OIDC/OAUTH tokens and responses.

3.2.1 Authentication Request

Provide state and nonce values with sufficient entropy.

- http://openid.net/specs/openid-connect-core-1_0.html#AuthRequest
- http://openid.net/specs/openid-connect-core-1_0.html#NonceNotes

3.2.2 Validate authentication responses

Handle the state parameter correctly.

- http://openid.net/specs/openid-connect-core-1_0.html#AuthResponseValidation

3.2.3 Validate token endpoint responses

Validate the ID Token and proof scopes.

- http://openid.net/specs/openid-connect-core-1_0.html#TokenResponseValidation

3.2.4 Validate ID token

Relying Party must validate ID Tokens.

- http://openid.net/specs/openid-connect-core-1_0.html#IDTokenValidation

3.2.5 Protect Client ID and secret

Relying Party's credentials (for example, client secret) must be stored safely for remaining a secret only known by the Relying Party. Relying Party should inform the Mobile ID OP in case credentials have been compromised.

3.2.6 Store tokens securely

Tokens, especially Refresh Tokens, must be treated as credentials and stored securely in a place where only the End-Users for whom they were issued can access them.

3.3 Test Users

There are test user accounts available for testing and debugging purpose.

Please be aware, due to the strict phone number validation during the Mobile ID authorization flow, these test phone numbers will only be accepted by the Mobile ID server if they are provided via `login_hint` request parameter. However, your account must be authorized to use the `login_hint` parameter and requires the use of Pushed Authorisation Rquests (PAR), which keeps the parameters confidential between client and server (see chapter 3.1).

MSISDN	Auth	Description
+41-700092501	SIM	Robot User (EC key; Swisscom Root CA 2 Certificate)
+41-700092502	SIM	Robot User (RSA key; Swisscom Root CA 2 Certificate)
+41-000092401	SIM	Simulated user to test the Mobile ID Error 401; USER_CANCEL
+41-000092402	SIM	Simulated user to test the Mobile ID Error 402; PIN_BLOCKED
+41-000092403	SIM	Simulated user to test the Mobile ID Error 403; CARD_BLOCKED
+41-000092404	SIM	Simulated user to test the Mobile ID Error 404; NO_KEY_FOUND
+41-000092406	SIM	Simulated user to test the Mobile ID Error 406; PB_SIGNATURE_PROCESS

4 Public Cloud Integration Guide

This chapter covers a few Mobile-ID integration scenario examples with popular public cloud services.

4.1 Microsoft Azure ADB B2C

[Azure Active Directory B2C](#) is a Single-Sign-On (SSO) solution for any API, web, or mobile application. It enables any organization to provide their users with access using identities they already have, such as Mobile-ID.

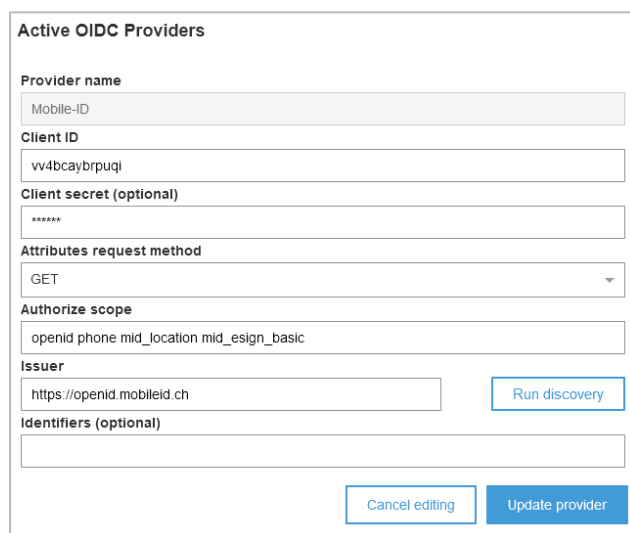
Mobile-ID is officially supported by Microsoft Azure and the setup is described on their [official article](#).

Microsoft will act as the secure front door to any of these applications and they will worry about the safety and scalability of the authentication platform. Azure will handle things like denial of service or brute force attacks, so that organizations can focus on their core business and stay out of the identity business.

4.2 Amazon Cognito

Amazon Cognito lets you add user sign-up, sign-in, and access control to your web and mobile apps quickly and easily. Amazon Cognito scales to millions of users and supports sign-in with social identity providers, such as Apple, Facebook, Google, and Amazon, and enterprise identity providers via SAML 2.0 and OpenID Connect.

[This article](#) explains how to integrate user sign-in with an OpenID Connect IdP such as Mobile-ID.



The screenshot shows the 'Active OIDC Providers' configuration page in the Amazon Cognito console. The configuration is for a provider named 'Mobile-ID'. The fields are as follows:

- Provider name:** Mobile-ID
- Client ID:** vv4bcaybrpuql
- Client secret (optional):** (masked with asterisks)
- Attributes request method:** GET
- Authorize scope:** openid phone mid_location mid_esign_basic
- Issuer:** https://openid.mobileid.ch
- Identifiers (optional):** (empty field)

At the bottom right, there are two buttons: 'Run discovery' and 'Update provider'. At the bottom left, there is a 'Cancel editing' button.

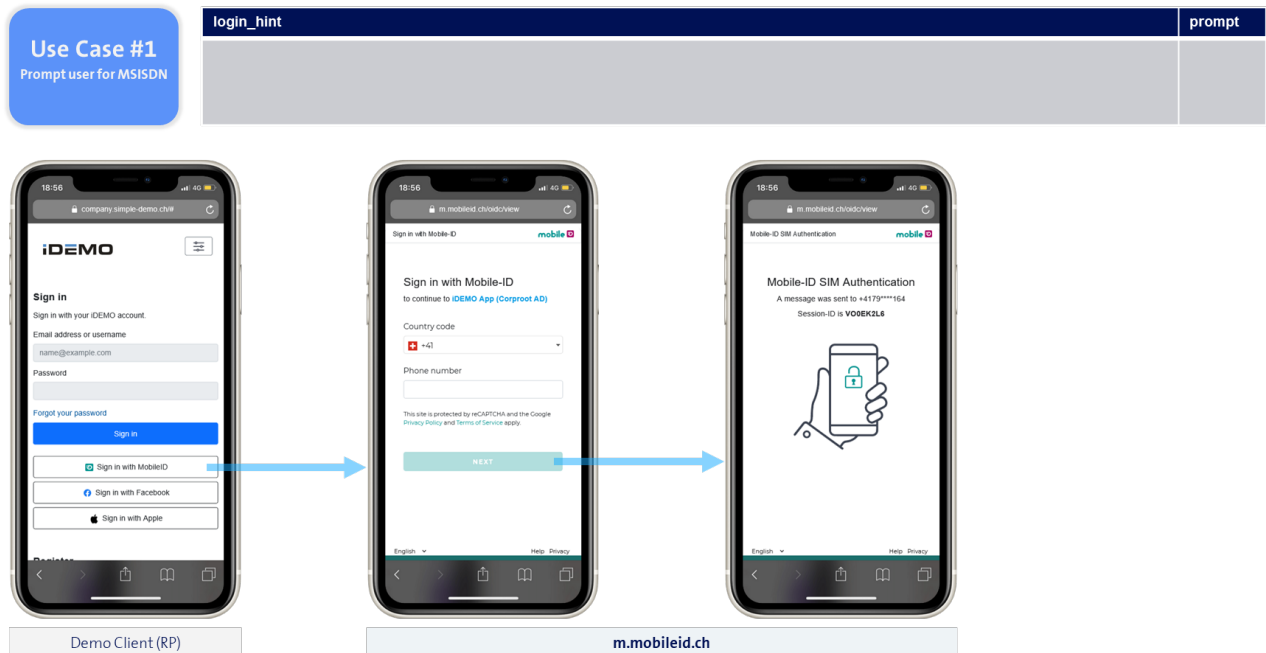
Figure 1: Example Configuration (Amazon)

5 MobileID OIDC - Use Cases

This chapter provides additional guidelines about the various parameter settings in the authorization request.

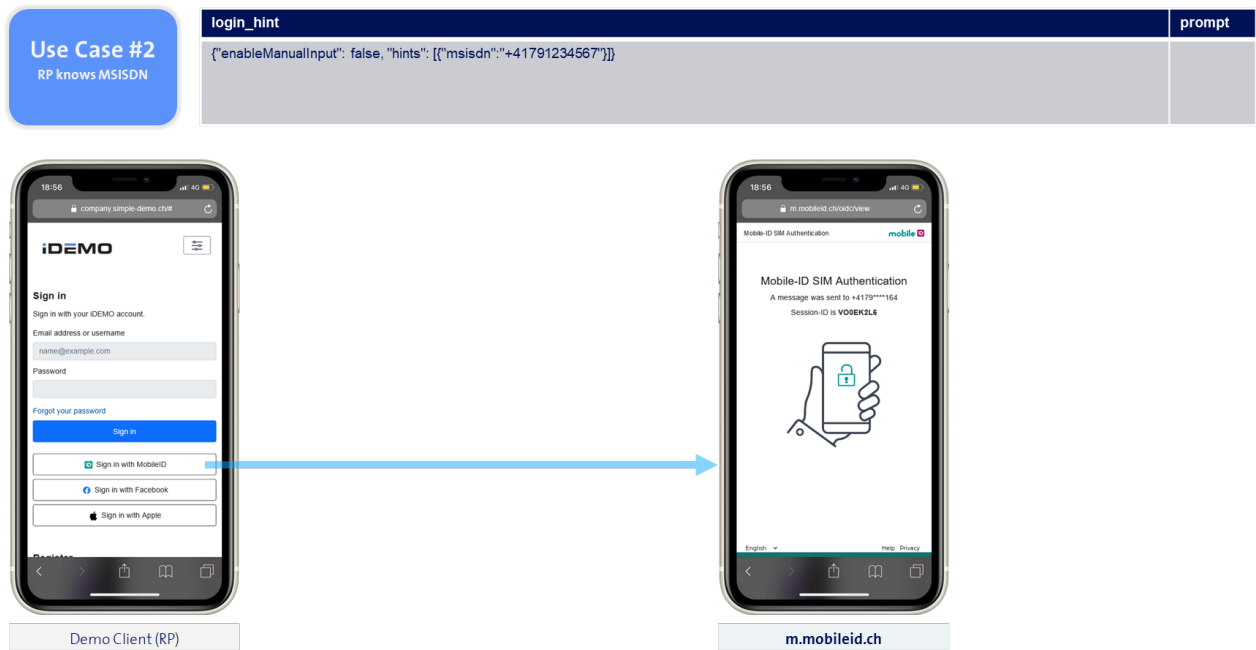
5.1 Prompt user for MSISDN

If the request does not contain a `login_hint` nor `prompt` parameter, the result will be that the user must enter the phone number on the MobileID side, as shown in the figure below. This is a typical B2C scenario, for example the MobileID login to a public web shop. Note that the Relying Party won't know the user's MSISDN unless the user will give his consent.



5.2 RP knows the MSISDN

If the request contains a `login_hint` parameter with the user's phone number, the MobileID authentication can start immediately.

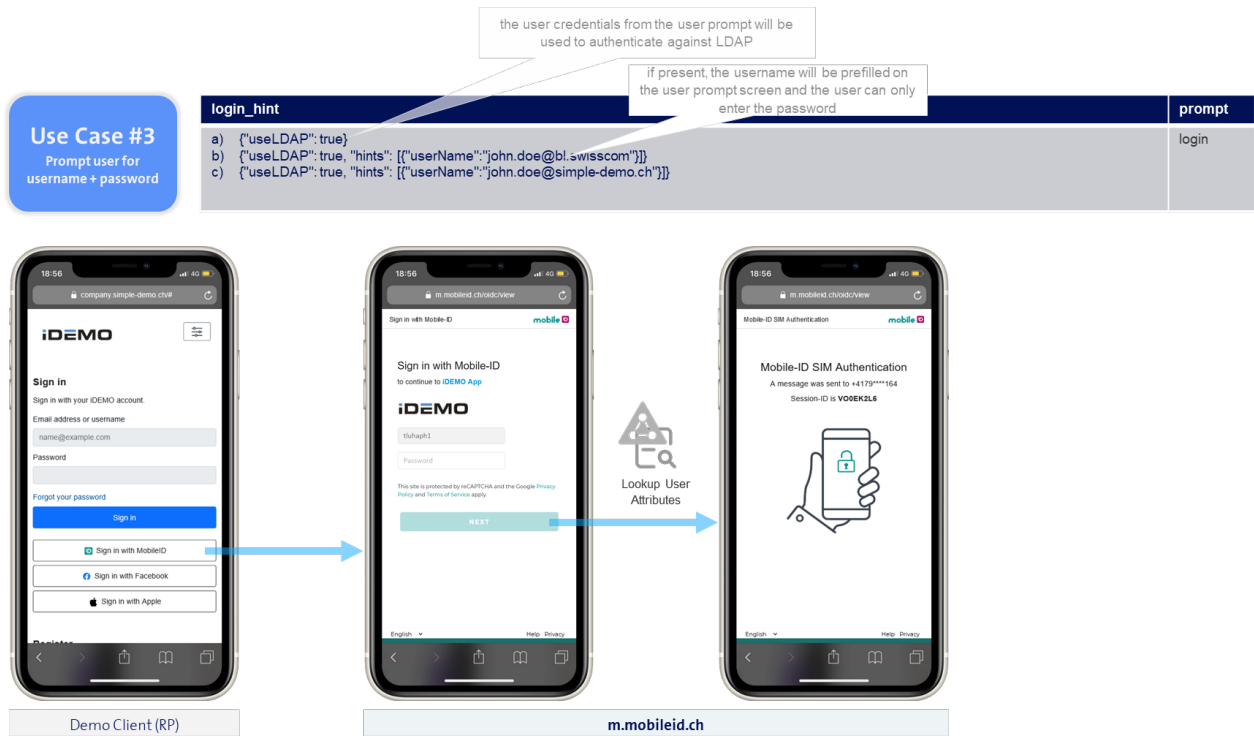


5.3 Prompt user for user credentials

If the request contains a `login_hint` parameter set to `useLDAP:true` and a `prompt` parameter set to `login`, the result will be that the user must enter the user credentials on the MobileID side, as shown in the figure below. This is a typical B2B scenario, for example the MobileID login to a company service.

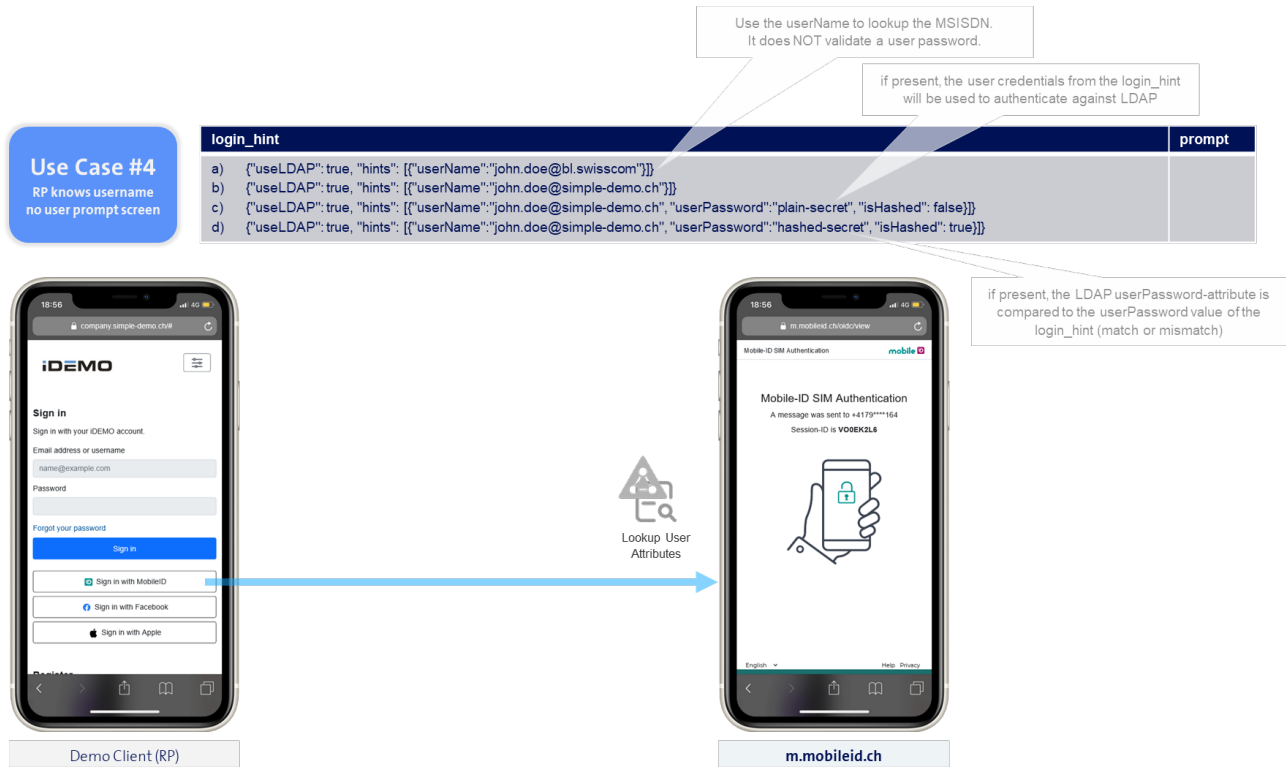
MobileID service will look up the username on the Active Directory (LDAP), verify the user password and retrieve the user's mobile phone number, before it will eventually start the MobileID authentication.

Instead of the MobileID domain (m.mobileid.ch), we can configure your custom domain instead.



5.4 RP knows the username

If the request contains a `login_hint` parameter set to `useLDAP:true` but there is no `prompt` parameter, the result will be that the MobileID service will look up the username on the Active Directory (LDAP) to retrieve the user's mobile phone number, before it will eventually start the MobileID authentication.



6 Message Formats on the Mobile ID App

Mobile ID App screens can present the Data-To-Be-Displayed (DTBD) in two formats.

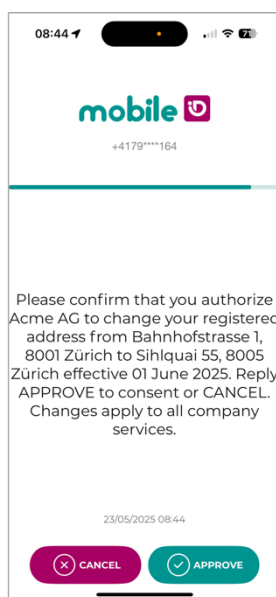
Use **Classic DTBD** for short confirmations and when you must support SIM users. Keep messages concise and always include the “DTBD Prefix” (refer to chapter **Error! Reference source not found.**).

Use **Transaction Approval** when readability matters (e.g., PSD2 payments, contract consent, step-up login verification). Force the App method with Device-LoA4, keep within byte limits, and generate the escaped JSON programmatically.

1. **Classic DTBD** (single text line) uses plain UTF-8 string that is also signed (DTBS).

Supported by SIM and App methods.

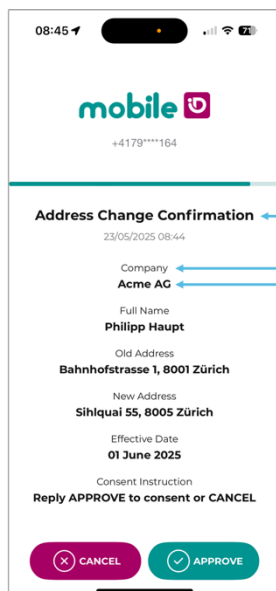
Length limited and no formatting options.



2. **Transaction Approval** (key/value pairs) is a structured App-only format that renders a title (`type`) and one or more `key` and `value` rows for improved readability.

Approve/Cancel becomes active only after the user scrolls to the end if content exceeds one screen.

RPs request this by sending a JSON object in the `dtbd` authorization parameter.



6.2 Classic DTBD

A single UTF-8 string shown on the device. The classic DTBD must include the AP-specific DTBD prefix (e.g., Bank ACME:) and is **supported by both SIM and App methods**.

Keep the DTBD short. Maximum 239 characters; if any character falls outside the GSM 03.38 set, effective maximum is 119 characters.

Parameter: `dtbd` with a plain string. The OP renders the classic one-line message.

Typical request excerpt:

```
...&dtbd=Please%20confirm%20your%20login%20to%20MyBank%20eBanking
```

6.3 Transaction Approval

A structured DTBD that the Mobile ID App renders as a title and rows of key/value pairs. If content overflows, the user must scroll to the bottom; only then are Approve/Cancel enabled.

SIM does not support this format. Always select an App ACR (e.g., `mid_al3_mobileapp`).

How to request it: Send a JSON object via the `dtbd` authorization request parameter; the value must be URL-encoded (percent-encoding). If the decoded value starts with `{` and validates against the schema/limits below and the user has an active App, the OP renders the key/value screen.

```
{
  "type": "<ascii>",
  "dtbd": [
    { "key": "<ascii>", "value": "<ascii>" }
    /* up to 20 pairs */
  ]
}
```

Limits (in bytes)

- `type` ≤ 100
- ≤ 20 pairs
- each `key` ≤ 100, each `value` ≤ 2000
- `sum(keys+values)` ≤ 2000 (bytes; non-ASCII uses 2-4 bytes)
- DTBD Prefix (required): your configured prefix must be in the value of the first pair e.g., `{ "key": "Company", "value": "Acme AG: ..."`

6.4 Transaction Approval Example

6.4.1 Pretty JSON

Build this first.

```
{
  "type": "Address Change Confirmation",
  "dtbd": [
    { "key": "Company", "value": "Acme AG" },
    { "key": "Full Name", "value": "Philipp Haupt" },
    { "key": "Old Address", "value": "Bahnhofstrasse 1, 8001 Zürich" },
    { "key": "New Address", "value": "Sihlquai 55, 8005 Zürich" },
    { "key": "Effective Date", "value": "01 June 2025" },
    { "key": "Consent Instruction", "value": "Reply APPROVE to consent or CANCEL" }
  ]
}
```

6.4.2 Single-line

What you URL-encode as `dtbd=`

```
{"type":"Address Change Confirmation","dtbd":[{"key":"Company","value":"Acme AG"}, {"key":"Full Name","value":"Philipp Haupt"}, {"key":"Old Address","value":"Bahnhofstrasse 1, 8001 Zürich"}, {"key":"New Address","value":"Sihlquai 55, 8005 Zürich"}, {"key":"Effective Date","value":"01 June 2025"}, {"key":"Consent Instruction","value":"Reply APPROVE to consent or CANCEL"}]}
```

Below is another example that includes the keywords `#CLIENT#` and `#SESSION#`, as described in section 2.2.

```
{"type":"LOGIN","dtbd":[{"key":"Company","value":"#CLIENT#"}, {"key":"Session","value":"#SESSION#"}]}
```

6.4.3 Authorization request (excerpt)

Use **PAR** to avoid URL length limits and keep parameters confidential.

Or pass `dtbd` directly on the authorize URL:

```
GET /oidc/authorize?
  response_type=code
  &scope=openid
  &client_id=YOUR_CLIENT_ID
  &state=...
  &redirect_uri=https%3A%2F%2Fclient.example%2Fcb
  &acr_values=mid_a13_mobileapp

&dtbd=%7B%22type%22%3A%22Address%20Change%20Confirmation%22%2C%22dtbd%22%3A%5B%7B%22key%22%3A%22Company%22%2C%22value%22%3A%22Acme%20AG%22%7D%2C...%5D%7D
```

Important: Request an App ACR (e.g., `mid_a13_mobileapp`) if you want to ensure the App method is selected.

6.5 What exactly is signed (classic vs. Transaction Approval)

Classic DTBD: the service signs the **visible text string**.

Transaction Approval: the App signs a **normalized JSON object** of the form:

```
{"format_version":1,"content_string":[{"key":"Company","value":"Test"}]}
```

i.e., the `dtbd` array only; the `type` label is not part of the signed bytes.

6.6 Best practices

- Build → URL-encode → send: generate the JSON with your library, then URL-encode as `dtbd`. Avoid hand-crafted strings. (Use **PAR** for large payloads/confidentiality.)
- **Select App method:** use `acr_values` (e.g., `mid_a13_mobileapp`) when you want to ensure App UX.
- **Prefix rule:** include your DTBD prefix in the **value of the first pair**.
- **Respect byte limits:** limits are in **bytes**, not characters; UTF-8 non-ASCII uses 2-4 bytes.
- **Number-matching & keywords:** `#SESSION#` and `#CLIENT#` keywords are supported in `type`, `key`, or `value` and can be used to implement number matching.