Mobile Connect MNO Implementation Requirements

Version 0.1

[Publication Date]

This is a Non-binding Permanent Reference Document of the GSMA

Security Classification: Non-confidential

Copyright Notice

Disclaimer

Antitrust Notice

Table of Contents

1 Introduction

1.1 Overview

This document provides a set of mandatory requirements that must be implemented within Mobile Connect. This document can be referred to together with other documents defined by the CPAS workstream for the product definition and technical aspects of the Mobile Connect.

1.2 Scope

The scope of this document is to provide mandatory requirements that must be implemented by operators to implement Mobile Connect. The scope covers:

- Identity Gateway (ID GW) implementation.
- Authenticator selection based on LoA (Level of Assurance).
- Pseudonymous Customer Reference (CR) creation.
- OpenID Connect (OIDC) implementation.
- Security requirements.

1.3 Abbreviations

Term	Description
ACR	Anonymous Customer Reference
CRUD	Create, Read, Update and Delete
ID GW	Identity Gateway
IDP	Identity Provider
IMSI	International Mobile Subscriber Identity
JWT	JSON Web Token
LDAP	Lightweight Directory Access Protocol
LoA	Level of Assurance
MCC	Mobile Country Code
MLS	Message Layer Security
MNC	Mobile Network Code
MSISDN	Mobile Station International Subscriber Directory Number
OIDC	OpenID Connect
OTA	Over-the-air
PCR	Pseudonymous Customer Reference
RDBMS	Rational Database Management System
SMSC	Simple Message Service Centre
TLS	Transport Layer Security
UI	User Interface
URI	Universal Resource Identifier
SDK	Software Development Kit

Term	Description
DTBS	Data to be signed

1.4 References

Ref	Doc Number	Title
[1]	CPAS 15	Mobile Connect Implementation Guidelines
[2]	CPAS	Mobile Connect Architecture
[3]	CPAS 5	CPAS 5 OpenID Connect Mobile Connect Profile
[4]	OpenID Connect Specification	OpenID Connect Authorization Code Flow http://openid.net/specs/openid-connect-core-1_0.html#CodeFlowAuth
[5]	ETSI 102 204	http://webapp.etsi.org/workprogram/Report_WorkItem.asp?WKI_ID=16183
[6]	RFC 5646	https://tools.ietf.org/html/rfc5646
[7]	ISO/IEC 29115	https://www.iso.org/obp/ui/#iso:std:iso-iec:29115:ed-1:v1:en
[8]	RFC 6749	https://tools.ietf.org/html/rfc6749

2 Mobile Connect Architecture Overview

The Mobile Connect logical architecture reuses many of the operator assets and introduces a few new key components in order to deliver the Mobile Connect services in accordance with the guiding principles outlined previously.

The following diagram illustrates the key logical components that will need to be provided for, or impacted by, the deployment of Mobile Connect services.

NOTE: It should be noted that this is a logical architecture identifying the functional components. The actual implementation choices (e.g., mapping of functionality to physical components) is left to the operator.

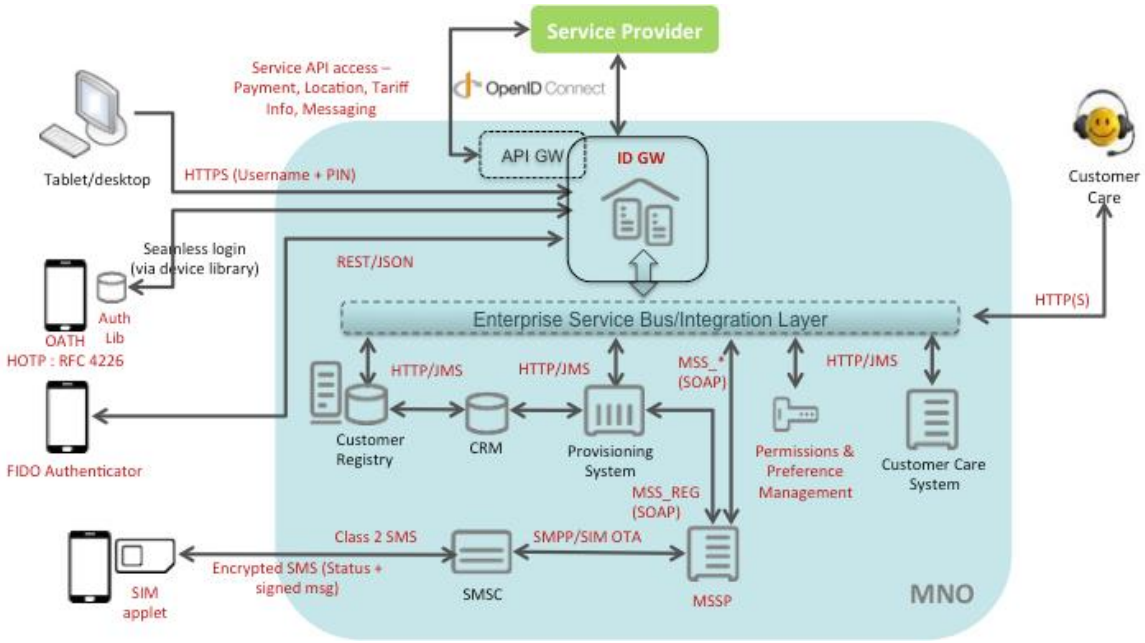

Figure 1: Mobile Connect architecture

Some key principles of the Mobile Connect architecture are:

- Pluggable authenticators.
- Standard definition of LoA.
- OpenID Connect as the protocol interface to the service provider.
- Authenticator selection based on the SP configuration during registration or based on the context of the request like (LoA, SP client_Id + Policy Routing), etc.
- Single point of contact for operator discovery using API Exchange .
- Single point of contact to accept T&Cs for all participating operators.
- Loose coupling between participating operators.
- The service provider only connects to Mobile Connect ID GW of the service operator.
- The service provider only needs to implement the OIDC protocol to connect to ID GW.
- The ID GW supports various authenticators and provides appropriate interface and protocol to connect to authenticators.
- Single point to maintain security, throttling, auditing, reporting, etc.

3 High Level Operator Requirements

3.1 Operator Implementation Requirements

Following are the high level tasks that an operator needs to implement to support Mobile Connect.

- Operators need to expose the OIDC endpoint from their ID GW components of Mobile Connect.
- Operators should register with API Exchange and provide all details including the OIDC endpoint URLs, the IP address range, certificates or Mobile Country coude (MCC)/Mobile Network Code (MNC) details.

- Operators should also develop the authenticator connectors from ID GW to their MSSP compliant with ETSI TS 102 204 [5] for SIM applet-based authentication. This is only needed for SIM-based authentication.
- Operators need to implement a pluggable authenticator mechanism to support various authenticators as per requirements.
- Operators should take care of all security aspects during the ID GW implementation, including Transportation Layer Security (TLS) or Message Layer Security (MLS), as per the Mobile Connect architecture.
- Operators should also provide a portal for end users to register and accept the Terms and Conditions for Mobile Connect.
- Operators may also provide over-the-air (OTA) and Simple Message Service Centre (SMSC) to reach SIM applet.
- Operators should also expose endpoints for UserInfo and PremiumInfo to access end user claims as per OIDC specification.
- Operators may need to implement connectors to get access to third party data source to get end user claim.

4 Operator Onboarding to API Exchange

Mobile Connect uses API Exchange to find out the endpoint of the serving operator. The API Exchange is a central component to manage discovery and enable federation across the identity provider (IDP). The API Exchange exposes a simple REST based Discovery API ultimately returning “Discovered” resources in a JSON object.

API Exchange utilises the mobile context to provide the discovery service. It uses (MCC/MNC) or network IP address to find out the serving operator. In case the client initiate the discovery over off net or off channel, the API presents the user a form to select the home operator.

The operator needs to provide the MCC/MNC, IP Address range, OIDC endpoints, and certificate details to the API Exchange during the onboarding process.

Please find more details on the API Exchange discovery in CPAS 15 Operator implementation Guidelines document [1].

5 Identity Gateway Implementation

The ID GW is one of the key logical components of the Mobile Connect architecture and provides the following functionality:

- Act as the entry point for the Mobile Connect interactions.
- Expose OpenID Connect and act as the OpenID Connect provider.
- Ability to implement asymmetric decryption to decrypt the Mobile Station International Subscriber Directory Number (MSISDN), if present in the request.
- Manage the interaction with the authenticators.
- Manage the policy-based routing into authenticators.
- Manage the protocol mediation between the northbound OpenID Connect and the authenticator specific protocols using the adaptors.

- Manage the multi-variant throttling of the incoming requests based on declarative policies.
- Manage the logging and auditing of interactions and operations.
- Manage the identification, authentication and authorisation of the OpenID Connect clients (service providers).
- Allow to add features to access the userinfo/premiuminfo from the data gateway or other sources.
- Provide external support for credential management with API Exchange.

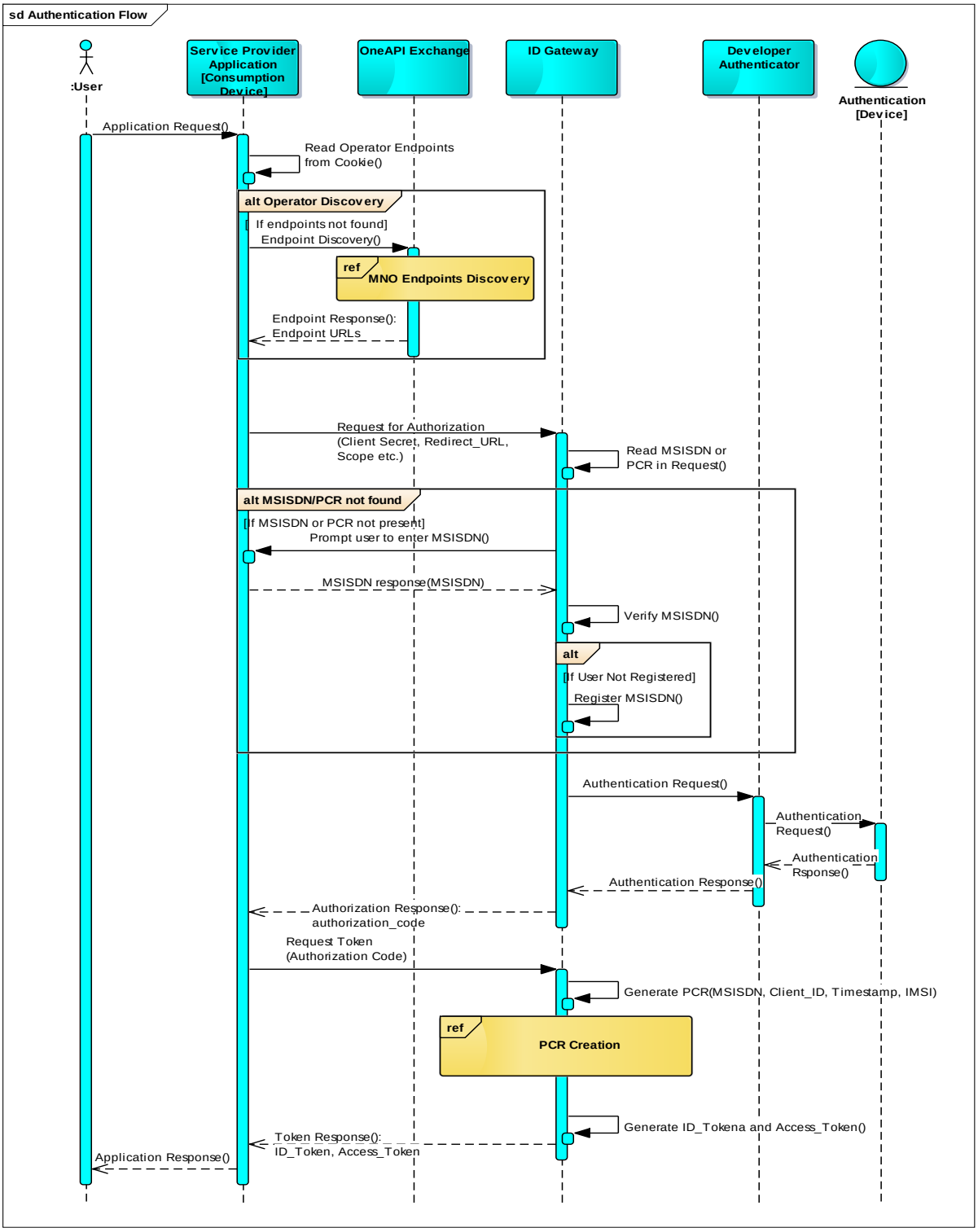

Figure 2: Authentication flow between the service provider and the ID GW

- The user selects the service provider application and clicks the “Login with Mobile Connect” button.
- The service provider connects to the API eEchange for discovery and finds out the serving operator for user.
- The service provider sends the authorisation request to the ID GW at the operator endpoint.
- The ID GW reads the MSISDN if available in the request, otherwise asks the user to present them.
- The ID GW verifies the MSISDN and registers the user in case they are not registered with Mobile Connect.
- The ID GW selects the authenticator based on LoA and prompts the user to authenticate them.
- The ID GW generates the authorisation code and sends a response back to the service provider.
- The service provider makes a token request passing authorisation code on the token endpoint.
- The ID GW validates the client credential and generates a PCR.
- The ID GW generates the `ID_Token` and `Access_token` and sends response back to the service provider
- The service provider validates the token and provides access to the request page.

5.1 Authenticator Selection

One of the key architecture principles of Mobile Connect is the Pluggable Authenticator principle. To achieve that, the authenticators need to be selected dynamically based on the configured policies using inputs such as the LoA indicated in the OIDC request by the SP, the available authenticator, the user/device eligibility, etc.

Some key points to implement the authenticator selection mechanism:

- The authenticators should be integrated with the ID GW based on an adaptor pattern.
- The specific information of the authenticator implementation should be confined within the authenticator adaptor as much as possible.
    - This provides a loose coupling between the authenticator implementation and the Mobile Connect system and drives the Pluggable Authenticator principle.
- The policy engine should route the authentication request to the adaptor.
    - The adaptor may then use authenticator specific interaction model to invoke and communicate.
    - The adaptor may use more than one call to invoke the authenticator (composition).
- The various factors (inputs) used for the authentication selection should be stored in a configuration database.

    - It should be possible to manage and administer the lifecycle of the factors.
    - Some of the inputs that will be used for the authenticator selection can be LoA, context request, SP `client_id`, user profiles, etc.

- The authenticator selection mechanism should validate and ensure that the request contains enough information to invoke the authenticator and all authenticator specific details are available.
- The authenticator selection mechanism should convert and add any specific information needed for the authenticator.

Following are the few examples to implement the authenticator selection policy:

- The SP identifier can be used to map one LoA with different types of authenticators: e.g. Client ID 1 + LoA3 can be mapped with Authenticator 1 and Client ID 2 + LoA3 can be mapped to Authenticator 2. These configuration policies will be stored in the operator's repository.
- The authenticator can be selected based on the service context request, SP preference, user profiles, etc.
- Fall-back mechanism should be configured and authenticators can be selected based on the LoA order. The service provider may pass multiple LoA in `acr_values` parameter in order of authenticator preference.
- The operator may authenticate with a lower LoA in case the requested LoA is not available. The authenticator used for authentication will be passed in the `amr` parameter in the OIDC response. The service provide will take the decision to provide the service or show an error message to the user based on their policy. For example, a service provider only needs SIM applet authentication and the user is authenticated with USSD. The operator will pass the USSD authenticator in the `amr` parameter. The service provider will decide whether to grant the full service, limited service or error to user.

5.1.1 Authenticator Selection Using OIDC Request Parameter- (amr Attribute)

The service provider may use one of the existing attributes defined in the OpenID Connect Specification document [4] to indicate the desired authentication needed to authenticate the end user. The operator can use the requested authenticator to authenticate the end user. In case the operator does not support the request authenticator, they can use any other supported authenticator against the requested LoA.

The service provider can use the `amr` attribute and can use it to pass the desired authentication, e.g. the service provider can pass `amr=SIM-OK` in the authentication request. The operator will read this attribute and will use the SIM-OK authenticator. In case the operator does not use the SIM-OK authenticator, they can use any other supported authenticator.

The operator will pass the actual authenticator used for authentication in `amr` attribute in the ID Token. The service provider will read this attribute and decide to allow or deny the request if the requested authenticator does not match with actual authenticator.

5.1.2 Authenticator Selection Using "Request Validator API" Response Metadata

The client can mention a desired authenticator for their application during the application onboarding in the API Exchange. The API Exchange will store all these application details in its repository. The API Exchange can distribute these additional application details along with other details to all participating operators in case of API Exchange light integration.

The operator can also use a request validation API to validate the client credential with the API Exchange. The API Exchange returns the service provider details in the response of the request validation API. The operator can use the desired authenticator to authenticate the end user. If the operator cannot support the requested authenticator, they can use any other authenticator configured against the requested LoA.

The service provider can read the `amr` attribute from the ID Token to get the authenticator used to authenticate the end user. The service provider can decide to allow or deny the access based on their policy in case the desired authenticator is not the actual authenticator used for the authentication.

5.1.3 Operator Supported Authenticator From Discovery Response

The operator can provide details about the supported authenticator against the LoA to the API Exchange during the onboarding process.The API Exchange can store these details along with other information, including for example endpoints, in its repository. The API Exchange returns these details to the service provider in the discovery response. The service provider can read the available authenticator from the discovery response and request the desired authenticator from the list in the `amr` request parameter during the authentication request. The service provider can use any other authenticator if the desired authenticator is not supported by the operator.

5.2 Mapping of Authenticators to LoA Indicated

One of the key architectural principles of Mobile Connect is the Pluggable Authenticator principle where the authenticators are dynamically selected by the ID GW based on the configured policy and primarily taking into account the LoA indicated by the SP in the OpenID Connect request.

The ID GW policy engine needs to have a configuration to map the LoA to the appropriate authenticator. This configuration should be stored in a repository and it should be possible to manage the CRUD (Create, Read Update, and Delete) lifecycle of the mapping configuration. For performance reasons, the configuration can be cached in memory to optimise the reading of the mapping by the policy execution engine.

Authenticator	LoA
Seamless authenticators	LoA 2
Header enrichment based Authenticators	LoA 2
OpenID Connect with no consent flow (1FA)	LoA 2
Application sending MO SMS to Application Backend	LoA 2
§Device agent/library based authenticator	LoA 2
SIM Applet based (using MSSP) [PIN based]	LoA 3 [PIN based]
GBA – Generic Bootstrapping Architecture 3GPP TS 33.220	LoA 2
FIDO Authenticator	LoA 3
FIDO Authenticator using SIM as the secure element	LoA 3
QR Code based authenticator	LoA 2
SIM alliance API based authenticator	LoA 3 [PIN based]
USSD based authenticator	LoA 2 [OK], LoA 3 [PIN based]
SQRL based authenticator	LoA 2
SIM applet – "Click OK" based Authenticator	LoA 2
SMS OTP based Authenticator	LoA 2
Username/password + SMS OTP (URL) for untrusted devices	LoA 3
OTP generated on the device (HOTP) based authenticator	LoA 2
Device App using device SDK for "Click OK"- based Authenticator	LoA 2
Device App using device SDK for "PIN"- based Authenticator	LoA 3

Figure 3: Mapping of authenticators to LoA indicated

5.3 Pseudonymous Customer Reference (PCR)

The ID GW will need to generate a system facing identifier to be included in the OIDC response. This is used to carry out critical cross-component interaction between the systems.

An Anonymous Customer Reference (ACR) will be used to create a Pseudonymous Customer Reference (PCR) between the operator and the SP. This will have two parts:

- Set of metadata providing context around identity (e.g. type of identifier, country code or network code).
- Encrypted user identity (e.g. MSISDN).

The PCR will identify an authenticated user at a service provider. The PCR will be unique for a user at a ervice provider. Following are the characteristics of the PCR.

- A PCR is not dependent on the end user's current operator (Operator 1). The creation can be specific to an operator (to support uniqueness). However, once created, the PCR must be able to be used and stored by any operator.
- The implementation guide will recommend the size of the PCR to ensure that a PCR can be imported into a Mobile Connect account.
- PCRs are purely used to link service providers with a Mobile Connect account; data cannot be extracted from a PCR.
- If an end user changes the MSISDN or the International Mobile Subscriber Identity (IMSI), the PCRs should not be regenerated.
- All of the PCRs associated with an end user's account must be migrated across operators when the end user switches to a new operator.
- A PCR created by the end user's original operator must be able to be used by the end user new operator.

5.4 ID_Token Format

The ID GW will generate an OIDC response once a user is successfully authenticated and has provided consent for the request. The OIDC response contains an `Access_Token` and an `ID_Token`.

The `ID_Token` is a security token that contains claims related to authenticated user set by authorisation server. The authorisation server within the ID GW digitally signs the `ID_Token` and passes the JSON Web Signature (JWS) header.

The JWS represents digitally signed or MACed content using JSON data structure and ebase64url encoding.The JWS represents these logical values.

- JWS Header: JSON object containing the parameters describing the cryptographic operation and the parameter employed.
- JWS Payload: Message content to be secured.
- JWS Signature: Digital signature over the JWS header and payload.

The JSON Header consists of algorithms used to sign the JSON object that may be selected by the operator based on the available infrastructure. The client needs to validate the `ID_Token` before granting access. The `alg` header parameter contains the algorithm used in the JWS. The operator may need to validate the ID Token based on their own implementation. This will not be part of the Software Development Kit (SDK) provided by Mobile Connect.

To indicate the details of the authenticator used for the SIM applet, the `amr` parameter can be used. The `amr` will also be used to provide other authenticator details as well.

5.5 User Registration

The user should be able to register for Mobile Connect. Mobile Connect should provide a registration portal where the user can register for Mobile Connect with minimal registration data involved, such as the PIN or MSISDN. Mobile Connect needs to store the user's details in its repository. It should have provisioning to store sensitive information in hashed format using a secure hash algorithm such as SHA256.

It should also provide support for full user lifecycle management where the user can update their personal details, including the PIN for example.

For the data storage, a standard repository. Some examples that can be used are Lightweight Directory Access Protocol (LDAP), Rational Database Management System (RDBMS), Name/Value DB or non-relational databases such as MongoDB.

The following diagram represents the flow during the MSISDN validation and the user registration.

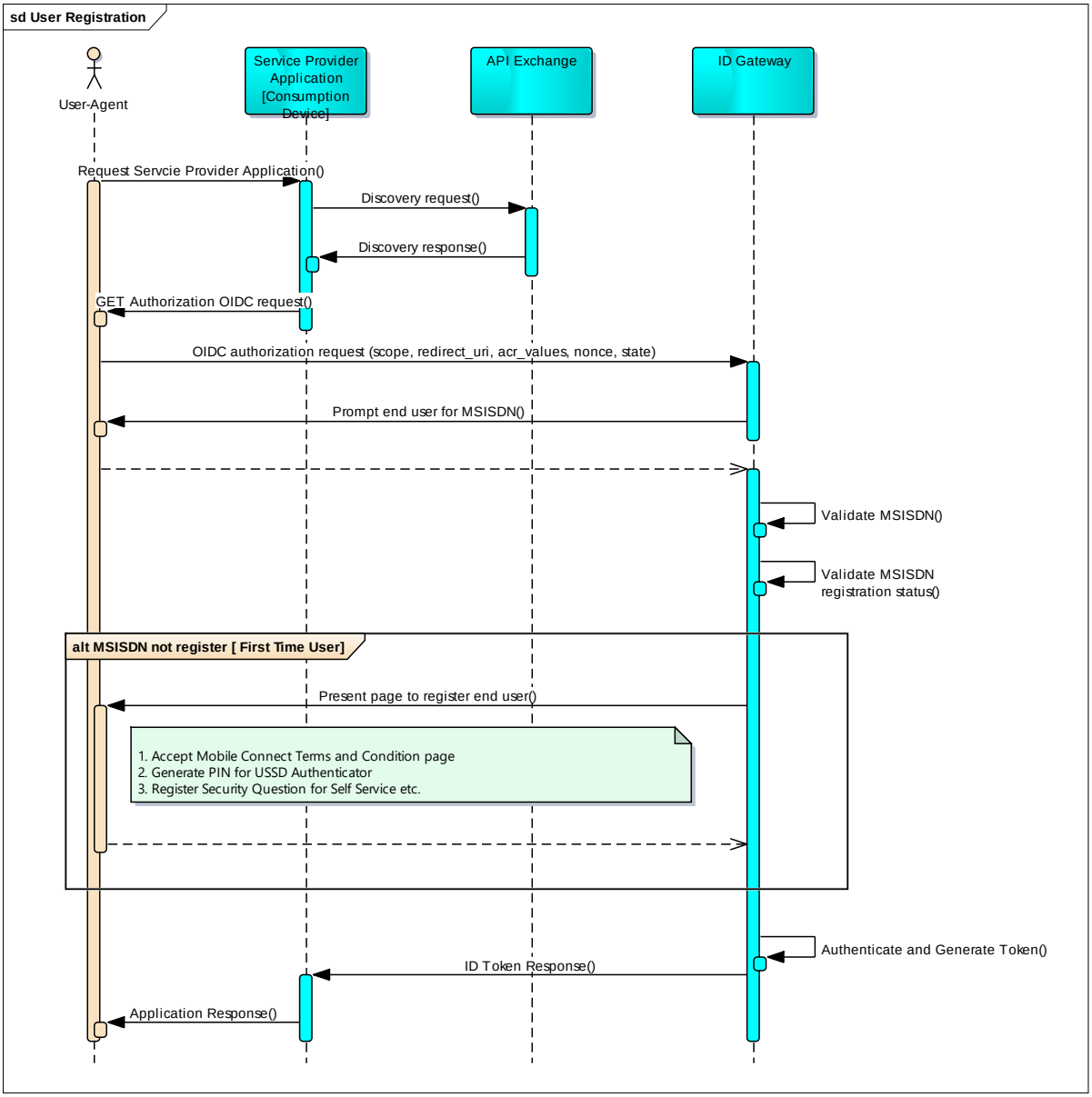

Figure 4: User registration

1. The end user accesses the service provider application and clicks on the “Login with Mobile Connect” button.
2. The service provider does the operator discovery and sends the OIDC authorisation request to the operator ID GW.
3. The ID GW validates the request parameter and retrieves the encrypted MSISDN from the `login_hint` attribute. The ID GW asks the user to enter the MSISDN if it is not present in the request.
4. The ID GW validates the MSISDN and checks for the valid token for the corresponding MSISDN or PCR.
5. The ID GW verifies the MSISDN and checks the registration status.
6. The ID GW asks the user to register the MSISDN by accepting Mobile Connect’s T&Cs.
7. The ID GW stores these details securely in its repository and completes the authentication process.

8. The ID GW generates an ID Token and send it back to the service provider to allow access.

5.6 MSISDN Decryption

The API Exchange can prompt the user to provide their MSISDN to perform discovery in case it does not find the MCC/MNC or IP address information in the request. The API Exchange will encrypt this MSISDN using the serving operator public key and pass the encrypted MSISDN to the service provider.

Please refer to the CPAS 15 Operator Implementation Guidelines document [1] for MSISDN encryption.

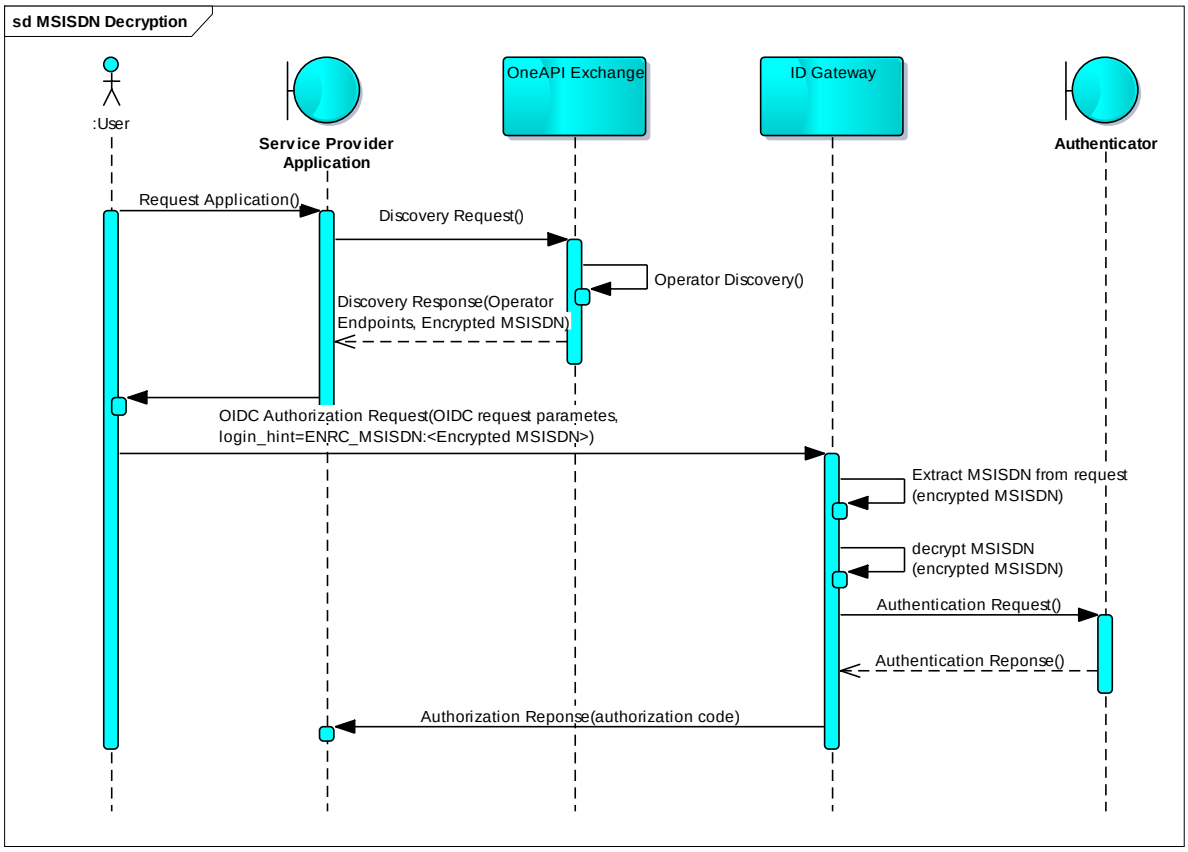

Figure 5: MSISDN decryption

- The service provider passes the encrypted MSISDN in the OIDC request under the `login_hint` attribute. The parameter can be a URL encoded.
- Following are the example of encrypted MSISDN passed:
    - **in the OIDC request**:
      login_hint=ENCR_MSISDN:5beb20e59de04f7e0840440036cb088dee393e4dfd269a5bcc29cb33369b7eb6c38b05c95fba6585d63d7c31bf3d7f331194b4caab3ef6d2e9c9341c36f09ad112a3847bde1dd049540eec799e81895960e27c6dda1dd04f2696a30473a38bbbf045e5a44a582d2fc9b25f7f22e0dc7cf5053deb73194588a1d3ae4fd2d7376a50bb20da1701722c0650ffb1cabc23ba885d3ae854516be2079c520ffcb56e7566f1fe94dc20196f856caa6d20411868d93bb70232b4f3e24c3e7838b9acd

a18e9b9d6787a67927a1f487ef8956c2ddc0539204174f830e508c89d600b81da796399dba1df566dc3a6b1d23d44e6a353775c06e7c5b8a2db39aee70e108aa11c

- **or with URL encoding**:
  login_hint=ENCR_MSISDN%3A0bb3020c7758f34e012da3f0bf13dc7674b3a95276e804388d5aae4a034fe442a65e03027d0651da3b0646df6c11d3c5d6f46879480b623bd5024d9e0879727f46fbd1e8f5383a115678ea638a4ba5399a2dd37138246edb06718bb44be98f5331a1331902d6333993642e2f25197961ee0b0a14ddf660834d49f7f385d82cad5a12003cd8aa235a92b71589110d76df382eab80b12a8dfa6d05b4ca548538ac4b09a2868448957604eb52b1ceecc89dfe836e7113e51645c2a14fff900228a8475983435647e88552a96eb692685b12abfc7ae0ad2bc23d30b3c8d828ca101e186455b4d618a8c9022662ee1c5b8ffea40defdb92a20dce39bdbedcbf785a2e

- The serving operator recognises the input of the encrypted MSISDN and decodes the base64 encoded data.
- The serving operator applies their private key to decode the RSA coded data.
- The decrypted string looks like 441234567890|dasd23231139dskdeirirewr0234043ekewrwe4034c.
- The serving operator then extracts the initial (numeric) portion of the decrypted data as the MSISDN separated by (|) pipe and uses this for any relevant purpose in API services/user sign-in.

5.7 Redirect_uri Validation

The service provider registered at operator provides details including the `redirect_uri` that will be used to send the authorisation code and token. The service provider passes the `redirect_uri` as a parameter in the authorisation request and token request. The operator needs to validate this `redirect_uri` before authenticating the end user or returning the token.

The API Exchange can pass all the service provider details including the `redirect_uri`, scopes, etc. via offline process to all participating operators or it can return the `redirect_uri` in the response for “Request Validator” API. The operator should reject any authentication or token request if the `redirect_uri` does not match with register one. The operator should also validate the `redirect_uri` in the token request and match it with the `redirect_uri` passed during the authorisation request.

The `redirect_uri` must exactly match with one of the `redirect_uri` pre-registered at the operator with the matching performed as simple string comparison. The `redirect_uri` should use the HTTPS flow (although it may use the HTTP url as well).

The redirection endpoint should require the use of TLS when the requested response type is code or token. If the TLS is not available, the authorisation server should warn the resource owner about the insecure redirection.

5.8 Device Interlock

Mobile Connect can be accessed in off channel when the consumption device is a PC and the authentication device is a mobile phone. To mitigate risks in case of a off-channel use case, Mobile Connect can display code on both devices. The user does not have to transfer

this code, it is just for comparison on both devices. This could be a number, text or image. As suggested previously in this document, in order to solve device interlock problem, the ID GW can show a text on the consumption device. The same text will be displayed on the authentication device. The end user can compare this text on both devices before the authentication.

In addition to that, the service provider can pass additional text in the query parameter to pass contextual information to the ID GW during authentication. The ID GW will pass this information to the authentication device that will be showed during the authentication.

The solution would consist of adding a query parameter in the authentication request with the message text. If it is present, it would be showed, otherwise the preconfigured text and the one-time-code would be showed.

The service provider can use the DTBS (data to be signed) attribute to pass the contextual information to the ID GW. The ID GW can display this text if it is present in the request along with the unique code on the authentication device.

5.9 Client Authentication

- The operator must read the client credential from the HTTP Header for client authentication.
- Behaviour will be different (undefined) from operator to operator if credential passed in HTTP Header and payload.
- The operator may reject or process the request based on their own implementation in case payload contains credentials or any other parameter out of specification
- The operator must reject the request if no client credential in HTTP Header

5.10 ID Token Signing

The operator must sign the ID token in the format of JSON Web Token (JWT). In most cases the ID token will be signed according to JSON Web Signing (JWS) specification. The id token consist of three components, a header, a payload and the digital signature. As per the JWT standards, these three sections are Base64url encoded and separated by periods (.).

6 Open ID Connect Implementation

Mobile Connect support authorisation code flow as defined in the OpenID Connect Specification document [4]. The authorisation code flow is a two-step process to obtain the access token and the ID token. The operator needs to implement the following endpoints to implement Mobile Connect. The operator needs to implement Mobile Connect as per specification defined in the CPAS 5 OpenID Connect Mobile Connect Profile [3].

6.1 Authorisation Endpoint

The service provider prepares the authorisation request as per Mobile Connect profiles and sends it to the operator authorisation endpoint for authentication. The service provider must follow the following requirements while sending the authorisation request:

- The communication between IDP and the service provider must use HTTPS connection.

- The request must use the HTTP GET or POST method.
- The request parameters are added using Query String Serialisation. If the POST method is used, request parameters are serialised using Form Serialisation.

The following table represents the parameters supported in the Mobile Connect authorisation request. The parameters defined as “mandatory” should always be present in the authorisation request. The IDP must respond with error in case any of the mandatory parameters are missing from the request. There should not be any error in case any of the optional attribute are missing in the request. The IDP must responds to request with or without optional attributes.

Following are the attribute list defined in Mobile Connect specification.

Parameter	Mandatory in Spec	Mandatory in Profile	Description
`response_type`	Mandatory	Mandatory	The value must be `code`, to indicate that the grant type flow to be used is authorisation code. It also indicates that the `access_token` (and `id_token`) be returned in exchange of `code`.
`client_id`	Mandatory	Mandatory	Needed for OAuth 2.0 authorisation request.
`scope`	Mandatory	Mandatory	Space delimited and case-sensitive list of ASCII strings for OAuth 2.0 scope values. OIDC authorisation request must contain the scope value `openid`. The other optional values for scope in OIDC are: `profile`, `email`, `address`, `phone` and `offline_access`.
`redirect_uri`	Mandatory	Mandatory	The Universal Resource Identifier (URI) where the response will be sent through redirection. The URI must match one of the pre-registered `redirect_uri` at client registration/provisioning.
`state`	Recommended	Mandatory	Value used by the client to maintain state between request and callback. A security mechanism as well, if a cryptographic binding is done with the browser cookie, to prevent Cross-Site Request Forgery.
`nonce`	Optional	Mandatory	String value used to associate a client session with the ID Token. It is passed unmodified from authorisation request to ID Token. The value should be unique per session to mitigate replay attacks.
`display`	Optional	Optional	ASCII String value to specify the user interface display for the authentication and consent flow. The values can be:

			`page`**:** Default value, if the display parameter is not added. The user interface (UI) should be consistent with a full page view of the User-Agent. `popup`**:** The popup window should be 450px X 500px (wide X tall). `touch`**:** The authorisation server should display the UI consistent with a "touch" based interface. `wap`**:** The UI should be consistent with a "feature-phone" device display.
`prompt`	Optional	Recommended	Space delimited, case-sensitive ASCII string values to specify to the authorisation server whether to prompt or not for reauthentication and consent. The values can be: `none`**:** must not display any UI for reauthentication or consent to the user. If the user is not authenticated already, or authentication or consent is needed to process the authorisation request, a `login_required` error is returned. This can be used as a mechanism to check existing authentication or consent. `login`**:** should prompt the user for reauthentication or consent. In case it cannot be done, an error must be returned. `consent`**:** should display a UI to get consent from the user. `select_account`**:** In the situations, where the user has multiple accounts with the IDP/authorisation server, this should prompt the user to select the account. If it cannot be done, an error must be returned.
`max_age`	Optional	Recommended	Specifies the maximum elapsed time in seconds since last authentication of the user. If the elapsed time is greater than this value, a reauthentication must be done. When this parameter is used in the request, the ID Token must contain the `auth_time` claim value.
`ui_locales`	Optional	Optional	Space separated list of user preferred languages and scripts for the UI as per RFC 5646 [6]. This parameter is for guidance only and in case the locales are not supported, error should not be returned.

claims_locales	Optional	Optional	Space separated list of user preferred languages and scripts for the Claims being returned as per RFC 5646 [6]. This parameter is for guidance only and in case the locales are not supported, error should not be returned.
id_token_hint	Optional	Optional	Generally used in conjunction with prompt=none to pass the previously issued ID Token as a hint for the current or past authentication session. If the ID Token is still valid and the user is logged in then the server returns a positive response, otherwise should return a login_error response. For the ID Token, the server need not be listed as audience, when included in the id_token_hint.
login_hint	Optional	Optional [Mandatory, when MSISDN or Encrypted MSISDN needs to be passed]	An indication to the IDP/Authorisation Server on what ID to use for login, e.g. emailed, MSISDN (phone_number) etc. It is recommended that the value matches the value used in discovery. The login_hint can contain the MSISDN or the Encrypted MSISDN and should be tagged as MSISDN:<Value> and ENCR_MSISDN:<Value> respectively – in case MSISDN or Encrypted MSISDN is passed in login_hint.
acr_values	Optional	Mandatory	Authentication context class reference. space separated string that specifies the authentication context reference to be used during authentication processing. The LoA required by the RP/Client for the use case can be used here. The values appear as order of preference. The acr satisfied during authentication is returned as acr claim value. The recommended values are the LoAs as specified in ISO/IEC 29115 Clause 6 – 1, 2, 3, 4 [7] – representing the LOAs of LOW, MEDIUM, HIGH and VERY HIGH. The acr_values are indication of what authentication method to used by the IDP. The authentication methods to be used are linked to the LoA value passed in the acr_values. The IDP configures the authentication method selection logic based on the acr_values.

Table 1: Mobile Connect attribute list

6.1.1 RP/Client Sends the Authorisation Request to the IDP/Authorisation Server

Following are the examples of authorisation requests sent using HTTPS/TLS to the IDP/authorisation server using GET or POST.

Sample request:

```
GET /authorize?
response_type=code&
client_id=s6BhdRkqt3
&redirect_uri=https%3A%2F%2Fclient.mid.org
&scope=openid
&state=af0ifjsldkj
&nonce=n-0S6_WzA2Mj
&acr_values=2


HTTP/1.1

Host: mid.example.com
Accept: application/json
```

6.1.2 IDP/Authorisation Server Authenticates User, Gets User Consent, Returns "Code" to the RP/Client

The code is returned to the URI value specified in the `redirect_uri`, response parameters are included as query parameters encoded using application/x-www-form-urlencoded.

Parameter	Mandatory in Spec	Mandatory in Profile	Description
`code`	Mandatory	Mandatory	Authorisation code as per OAuth 2.0.
`state`	Mandatory (if state was added in the request)	Mandatory	Must be same as the state value added in the request.

Table 2: Authorisation response

Sample response

```
HTTP/1.1 302 Found

Location: https://client.mid.org?code=SplxlOBeZQQYbYS6WxSbIA

&state=af0ifjsldkj
```

In case user authentication fails or user does not provide consent, operator should return authentication error in response as per OAuth 2.0.

6.2 Token Endpoint

The operator must implement token endpoint that will return the ID Token and Access Token. Communication with token endpoint must use HTTPS and the request encoding is used as application/x-www-form-urlencoded. The service provider must pass the authorisation code along with client credentials to token endpoint url.

The operator must validate the client credential before returning the Access Token and ID Token.

6.2.1 RP/Client Requests for Access Token and ID Token

Communication with the Token endpoint must use TLS. The request encoding used is application/x-www-form-urlencoded.

Sample request:

```
POST /token HTTP/1.1
Host: mid.example.com
Authorisation: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW
Content-Type: application/x-www-form-urlencoded

grant_type=authorisation_code&code=SplxlOBeZQQYbYS6WxSbIA
&redirect_uri=https%3A%2F%2Fclient%2Emid%2Ecom
```

Parameter	Mandatory in Spec	Mandatory in Profile	Description
`grant_type`	Mandatory	Mandatory	The value must be set to `authorisation_code`
`code`	Mandatory	Mandatory	The authorisation code received from the authorisation server, from the authorisation request.
`redirect_uri`	Mandatory	Mandatory	The `redirect_uri` value must match the one sent in the authorisation request.
`client credential`	Mandatory	Mandatory	The `client_secret` used in HTTP Basic Authentication using the OAuth 2.0 Client Password mechanism (RFC 6749 Section 2.3.1 [8]).

Table 3: Token request parameter

6.2.2 RP/Client Gets the Tokens (Access Token and ID Token)

The response for token endpoints is in accordance with OAuth 2.0 and should be encoded in UTF-8.

Parameter	Mandatory in Spec	Mandatory in Profile	Description
`access_token`	Mandatory	Mandatory	OAuth 2.0 `access_token`, used to get the UserInfo object from the UserInfo endpoint and can be reused for accessing other protected resources, if required.

token_type	Mandatory	Mandatory	Must be “bearer” unless another token_type value as agreed between the RP/Client and the IDP/Authorisation Server. For the Mobile Connect profile, token_type=bearer is the recommended value.
id_token	Mandatory	Mandatory	This is the additional token used in OIDC to provide the Identity token claim.
expires_in	Optional	Recommended	Expiration time in seconds from the time of generation of the response.
refresh_token	Optional	Optional	OAuth 2.0 referesh token to get the access_token when the access_token expires.

Table 4: Token response

Sample response:

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

{
"access_token":"SlAV32hkKG",
"token_type":"Bearer",
"expires_in":3600,
"refresh_token":"tGzv3JOkF0XG5Qx2TlKWIA",
"id_token":"eyJ0 ... NiJ9.eyJ1c ... I6IjIifX0.DeWt4Qu ... ZXso"
}
```

6.2.3 ID Token

ID Token is an extension to the OAuth 2.0 token (Access Token) to provide the claims for Authentication Context/Event, represented as a JWT (http://tools.ietf.org/html/draft-ietf-oauth-json-web-token-14).

The ID Token is created and returned by the IDP (e.g. operator).

Claims used in an ID Token:

Parameter	Mandatory in Spec	Mandatory in Profile	Description
iss	Mandatory	Mandatory	Issuer Identifier. It is a case-sensitive HTTPS based URL, with host. It may contain the port and path element (optional) but no query parameters.
sub	Mandatory	Mandatory	Subject Identifier. A unique (locally) identifier of the end user. It is a case-sensitive ASCII string with a maximum length of 255. The MSISDN should not be passed as the sub value. The ACR (Anonymous Customer Reference) - https://tools.ietf.org/html/draft-uri-acr-extension-04] can be used as the sub value, if possible. There can be other implementations of the sub parameter value.

aud	Mandatory	Mandatory	The intended audience for the ID Token. It is an array of case-sensitive strings. It must contain the client_id of the RP/Client and may contain identifiers of other optional audiences.
exp	Mandatory	Mandatory	The expiration time after which the ID Token must not be accepted for processing. It is represented as the number of seconds from 1970-01-01T0:0:0Z as measured in UTC until the date/time specified.
iat	Mandatory	Mandatory	The time of issue of the ID Token. It is represented as the number of seconds from 1970-01-01T0:0:0Z as measured in UTC until the date/time specified.
auth_time	Mandatory (if max_age was used in the Request), Optional otherwise.	Mandatory	Time of end user authentication. It is represented as the number of seconds from 1970-01-01T0:0:0Z as measured in UTC until the date/time specified.
nonce	Mandatory (If nonce was used in the Authorisation Request), Optional otherwise.	Mandatory	Opaque string value to associate the RP/Client session with the ID Token, to avoid the replay attacks. The nonce value must be same as the nonce used in the authorisation request. For the Mobile Connect profile it is a recommended parameter.
at_hash	Optional	Recommended (SHA-256 is the recommended hash algorithm)	A base64url encoded value of the hash of the access_token (the hash algorithm is negotiated during registration).
acr	Optional	Mandatory	Authentication Context Class Reference. It is a case sensitive string, representing the fact that the authentication process followed the acr (e.g. LOA) requested or not.
amr	Optional	Mandatory (The values are defined in the CPAS5_Values_Dictionary)	Authentication Methods References. An array of case-sensitive strings to indicate the authentication method used. The values need to be negotiated offline.
azp	Mandatory (if the audience to the ID Token is different to the Authorised Party), Optional otherwise.	Mandatory (if the audience to the ID Token is different to the Authorised Party), Optional otherwise.	Authorised Party – the party to which the ID Token is issued. Represented as the client_id of the party.

Table 5: ID token claims

7 SFRA Requirement

As defined in GSMA SFRA group, Mobile Connect may include various threats on operator side such as DDoS attack or data leak. The user may suffer social media attacks, OS incompatible bug, malware, spam, etc. A SP may also suffer from DDoS and phishing attack.

7.1 Operator Side Security and Fraud Mitigation

7.1.1 DDoS Attack

The operator should provide anti-DDoS solution by traditional way, such as a IPS system, to protect the ID GW and other components. The operator should also consider DDoS mitigation solution to ensure resilience of service. This should be applied on the all the in-scopes system and should be coupled with effective incident response.

7.1.2 Data Leak

The operator should provide data protection such as data encryption and access control mechanism to keep the user's personal information safe. Tools such as Intrusion Detection System and/or Intrusion Prevention should also be considered along with effective monitoring, detection and incident management.

7.1.3 Mass Spam and Target Spam

The operator should provide antispam solution by using second attribute such as location, or using alias input such as MSISDN. Meanwhile identity gateway should have the ability to detect abnormal patterns.

7.1.4 SIM Cloning

A SIM card may be cloned and despite of network countermeasures, this may in itself allow for fraudulent registration for service. The operator should implement SIM cloning process such as use of volume, value and velocity checking within fraud management system for detection.

7.1.5 MSISDN Recycling

An abuse of MSISDN recycle/purge process may create an opportunity for fraudulent registration service. The operator should implement internal audit process to tackle such issues.

7.1.6 OTA

The operator may use OTA campaign to distribute updated SIM application. A specific fraud risk includes the possibility to download the application to SIM profiles with known OTA vulnerabilities. OTA campaign must be constructed to identify and reject downloads to SIM with known vulnerable profile.

7.1.7 SMS Gateway and SMSC

An attacker sends spoofed SMS to customer using SMS Gateway or SMCS as part of fake authentication process to socially engineer the customer to believe they have authenticated to legitimate sites. The operator should take precaution to identify the SMS source.

Annex A Document Management

A.1 Document History

Version	Date	Brief Description of Change	Approval Authority	Editor / Company
0.1	29/05/2015	Created base line version	David Pollington	Afsar Hussain
0.2	15/06/2015	Incorporated review comments	Gautam Hazari	Afsar Hussain
0.3	26/06/2015	Converted to new template	David Pollington	Afsar Hussain