Mobile Connect server initiated OIDC profile

Version 2.0

03 June 2019

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

Mobile Connect is a portfolio of mobile-enabled services to provide Authentication, Authorisation, Identity Services and Network Attribute Services to be used in conjunction with services offered to a User by Service Providers (SPs).

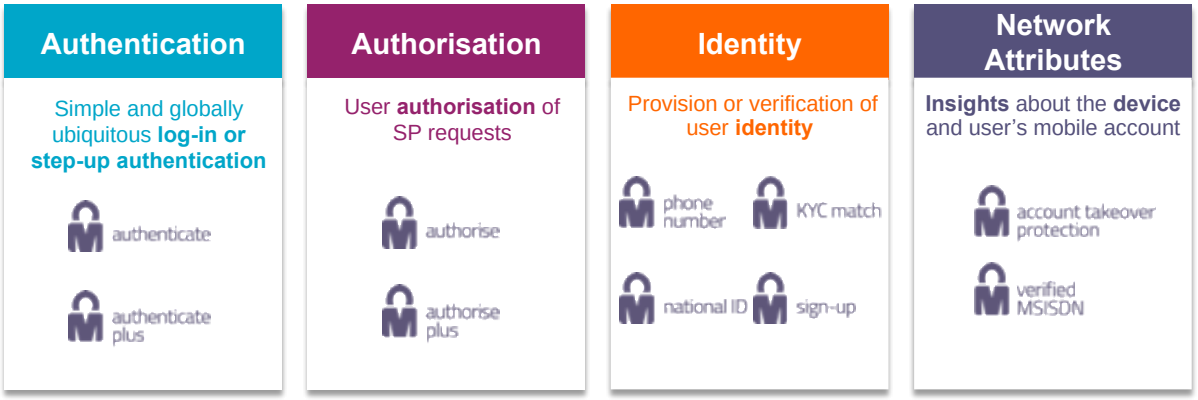

Figure 1: Mobile Connect Portfolio of Services

Mobile Connect is based upon the OpenID Connect (OIDC) protocol suite [1] and allows Users to be identified by their MSISDN (or a related Pseudonymous Customer Reference) and to provide Authentication, Authorisation and Consent via their mobile device.

The serving Mobile Operator supports and selects an appropriate Authenticator to present the Authentication, Authorisation and Consent request to the User on their mobile device to which the User responds. The Authenticator is selected based on Operator policy, device capability and the Level of Assurance required.

Mobile Connect also provides access to an enriched set of User attributes[1] provided by the Mobile Operator, that can be shared with a SP, subject to User consent.

The Mobile Connect architecture consists of a Core framework around which additional components can be added to support different Mobile Connect services that utilise the Core.

This specification defines a Mobile Connect Server-Initiated API (Application Programming Interface) offered by an Operator's Identity Gateway (IDGW) to a SP that enables a Mobile Connect service to be initiated when the User is not interacting online with the SP.

This specification is normative - it includes examples for illustration purposes that are non-normative.

[1] OpenID Connect specifies a set of attributes that can be obtained from the OIDC Provider's Resource Server (e.g., the serving Operator's IDGW) also referred to as 'Protected Resources'. Mobile Connect provides an enriched set of attributes that also includes information relating to a User's mobile account and status

1.2 Scope

In Scope	Out of Scope
• Mobile Connect Server-Initiated communication (Server-to-Server) • Mobile Connect Server-Initiated request through signed request object by value • Retrieval of tokens using Notification • Retrieval of tokens using Polling	• Mobile Connect Device-Initiated communication • Mobile Connect Server-Initiated signed request object through the reference [i.e. request_uri]. • TLS and mTLS implementation details

1.3 Audience

The target audience for this document are the Operator service/technical departments who are considering deploying Mobile Connect services in Server-Initiated mode.

1.4 Relationship to Other Mobile Connect Documentation

This document describes and specifies the Mobile Connect Server-Initiated mode and API. It includes details of the OIDC Authorization Request and Response and the mechanisms for then obtaining tokens upon successful Authorization. It also includes examples and generic error codes. This specification defines the server-initiated OIDC Authorization process and Token Retrieval that underpins Mobile Connect and forms part of the Core framework. All Mobile Connect Services published as server 2 server, make use Server-Initiated mode.

The Mobile Connect Technical Overview document [18] provides a high-level description of Mobile Connect and how it works. It also includes a master list of abbreviations and terminology used within the Mobile Connect Documentation set and a map of that documentation set. It serves as a starting point for understanding how Mobile Connect works and references the relevant documents for the reader to obtain further details.

The Mobile Connect Architecture and Core Technical Requirements document [19] describes the Mobile Connect Architecture in more detail and also includes the core technical requirements and specification of elements for Mobile Connect that are generic to all Mobile Connect services and modes of operation.

The Mobile Connect Resource Server Specification [21] provides details on how to handle a Resource request and the associated response for Mobile Connect Identity and Network Attribute services including error codes where this approach is used by a Mobile Connect service.

Each individual Mobile Connect service has its own definition document which includes service specific parameters, such as scope value and any service specific error codes. It also includes technical requirements that relate to that specific Mobile Connect service.

1.5 Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC2119 [6].

1.6 Terminology & Definitions

Mobile Connect technical specifications and related documentation make use of terms that are defined by the OpenID Connect Core Specification [2] and supporting specifications and extended in the OpenID Connect MODRNA Client Initiated Backchannel Authentication Flow specification [5].

The Mobile Connect Technical Overview document [18] provides a list of definitions and abbreviations that are used within the Mobile Connect Specifications. It includes terminology from source standards and interprets that terminology in Mobile Connect terms.

Due to potential confusion with OIDC (built on top of OAuth2.0) and OAuth2.0 terminology; the initial Mobile Connect Service Request (OIDC Authentication Request) which underpins Mobile Connect Authentication, Authorisation and User Consent associated with Identity Services and Network Attribute Services, is referred to as an OIDC Authorization Request following the OAuth2.0 terminology (spelled with a 'z') throughout this document.

1.7 References

Ref	Doc Number	Title
[1]	Open ID Connect	http://openid.net/connect/
[2]	OpenID Connect Core Specification	"An interoperable authentication protocol based on the OAuth 2.0 family of specifications" available at https://openid.net/specs/openid-connect-core-1_0.html
[3]	OIDC Basic Client Profile	OpenID Connect Basic Client Profile 1.0 http://openid.net/specs/openid-connect-basic-1_0-28.html
[4]	OIDC Basic Client Implementer's Guide	OpenID Connect Basic Client Implementer's Guide 1.0 https://openid.net/specs/openid-connect-basic-1_0.html
[5]	OIDF CIBA	OpenID Connect MODRNA Client initiated Backchannel Authentication Flow 1.0 https://openid.net/specs/openid-connect-modrna-client-initiated-backchannel-authentication-1_0.html
[6]	RFC 2119	"Keywords for use in RFCs to Indicate Requirement Levels", S. Bradner, March 1997. Available at https://tools.ietf.org/html/rfc2119
[7]	RFC 6749	"The OAuth 2.0 Authorization Framework", D. Hard5, Ed. October 2012 available at http://www.ietf.org/rfc/rfc6749.txt
[8]	RFC 6750	M. Jones and D. Hardt, "The OAuth 2.0 Authorization Framework: Bearer Token Usage," RFC 6750, October 2012 https://tools.ietf.org/html/rfc6750
[9]	RFC 5246	Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2," RFC 5246, August 2008 https://tools.ietf.org/html/rfc5322
[10]	RFC 3339	Klyne, G., Ed. and C. Newman, "Date and Time on the Internet: Timestamps," RFC 3339, July 2002 https://www.ietf.org/rfc/rfc3339.txt
[11]	RFC 3986	"Uniform Resource Identifier (URI): Generic Syntax"

Ref	Doc Number	Title
		http://www.ietf.org/rfc/rfc3986.txt
[12]	RFC 5646	Phillips, A., and M. Davis, "Tags for Identifying Languages" BCP 47, RFC 5646, September 2009 https://tools.ietf.org/html/rfc5646
[13]	RFC 7519	M. Jones, J Bradley, N. Sakimura "JSON Web Token (JWT)", RFC 7519, MAY 2015 https://tools.ietf.org/html/rfc7519
[14]	RFC 7518	JSON Web Algorithms (JWA) https://tools.ietf.org/html/rfc7518
[15]	RFC 7517	JSON Web Key (JWK) https://tools.ietf.org/html/rfc7517
[16]	RFC 7515	JSON Web Signature (JWS) https://tools.ietf.org/html/rfc7515
[17]	RFC 7516	Jones, M. and J. Hildebrand, "JSON Web Encryption (JWE)", RFC 7516, DOI 10.17487/RFC7516, May 2015, http://www.rfc-editor.org/info/rfc7516.
[18]	IDY.05	Mobile Connect Technical Overview
[19]	IDY.04	Mobile Connect Technical Architecture and Core Requirements
[20]	IDY.01	Mobile Connect Device-Initiated OIDC Profile
[21]	IDY.03	Mobile Connect Resource Server
[22]	IDY.33	API Exchange Functional Description v1.0
[23]		Mobile Connect Developer Portal: https://developer.mobileconnect.io

2 OpenID Connect

OpenID Connect (OIDC) is a simple Identity layer that sits on top of the OAuth 2.0 protocol. It enables Clients to verify the identity of a User based on the authentication performed by an Authorization Server as well as to obtain basic profile information about the User in an inter-operable and REST-like manner. Figure 2 outlines the Open ID Connect Protocol Suite reproduced from OpenID.net [1].

OpenID Connect provides an additional token (an ID Token) along with the OAuth 2.0 Access Token. The ID Token is represented as a JWT [13] and contains a claim set related to the Authentication Context of the subject. The JWT can be a plain text JWT or cryptographically protected JWT – represented as a signed JWT using JSON Web Signatures (JWS) [16] or as an encrypted JWT using JSON Web Encryption (JWE)[2] [17].

OpenID Connect does not specify how users should be authenticated - Mobile Connect is a specific implementation of Open ID Connect (OIDC) that uses the User's MSISDN (and an associated Pseudonymous Customer Reference) as an identifier and their mobile device as the Authentication Device. Additionally, it extends the range of information about the User that can be obtained by a Client, subject to the User's consent.

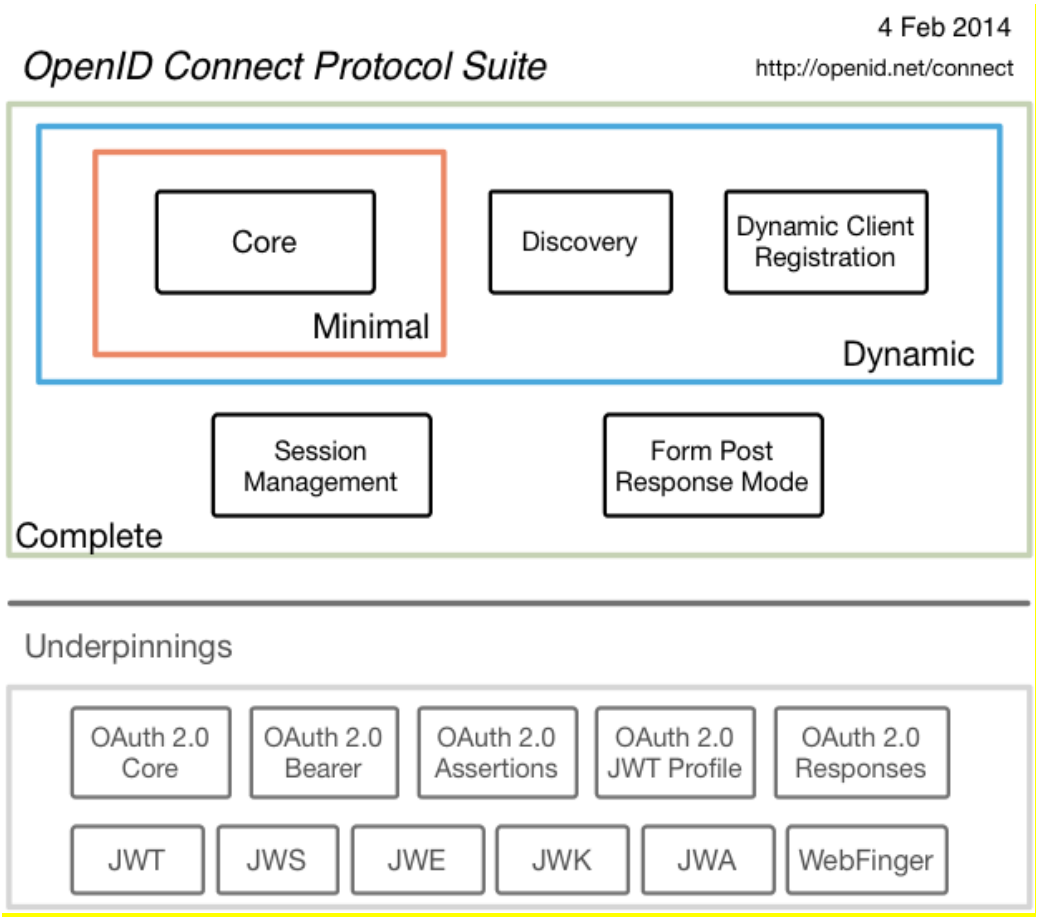

Figure 2: OpenID protocol suite

[2] Within Mobile Connect the ID Token is a signed JWT using JWS

A general introduction to OIDC is provided in [1]. The Core Specification [2] defines the core OIDC functionality and that underpins Mobile Connect. The Open ID Connect Basic Client Implementer's Guide [4] and the OpenID Connect Basic Client Profile 1.0 [3] contain a subset of the Core Specification that is designed to be easy to read and implement for basic web-based SPs using the OIDC Authorization Code Flow – referred to in Mobile Connect as Device-Initiated mode.

Extensions to OpenID Connect to support device-initiated / server-initiated requests (Mobile Connect Server-Initiated mode) are outlined in Reference [5].

2.1 Mobile Connect Server-Initiated Mode

There are some use cases where the SP server needs to be the initiator of the User authentication flow and where User interaction through a Consumption Device is not possible or not required for Mobile Connect calls.

The OIDF Client Initiated Backchannel Authentication Flow (CIBA)[5] specifies an authentication flow by means of which the SP knows the User identifier they want to authenticate (i.e. the MSISDN for Mobile Connect) and will be able to initiate an authorization flow to authenticate their Users. This is the basis for the Mobile Connect Server-Initiated mode. The flow is a direct communication from the SP server to the Operator IDGW. It introduces a new SI Authorization Endpoint[3] to which the OIDC Authorization request is posted and introduces multiple asynchronous methods for result delivery. It does not introduce new scope values, nor does it change the semantics of standard OpenID Connect Core parameters in the Authorization Request. Some OIDC core parameters which are required in a redirect-based protocol are meaningless in this server-to-server communication. A few new parameters are introduced e.g. those required for the asynchronous responses.

This specification describes Mobile Connect Server-Initiated mode including the structure of the OIDC Authorization Request and the various responses to that request including possible error codes. There are two variants to a Server-Initiated request described which relate to how Tokens are retrieved:

- **Token Retrieval through Notification (Asynchronous PUSH)** –The SP pre-registers Notification Endpoints with the IDGW and includes the Notification Endpoint to be used and the `response_type` to use notification in the OIDC Authorization Request.

  Once the OIDC Authorization Request has been validated and acknowledged, the request is processed and if successful (i.e. the User has been authenticated, has authorised a transaction or has provided consent to share personal information) then

[3] Mobile Connect recommends to use new SI Authorization endpoint for SI mode Mobile Connect services, however, based on few Operators infrastructure requirements, implementations can use the same endpoint for both DI and SI modes with the value of the response_type in the OIDC Authorization Request determining how the Authorization Server (in the IDGW responds). Implementation details of single authorization endpoint are out of scope this document.

a Token Response (including an ID Token and Access Token) is pushed to the Notification Endpoint.

For any processing failures, an appropriate error with a description is pushed to the Notification Endpoint.

The SP validates the Token Response and acknowledges the receipt if successful, otherwise returns an error for logging purposes.

- **Token Retrieval through Polling (Asynchronous PULL)** – The SP pre-registers with the IDGW for Server-Initiated mode using Polling and includes the appropriate response_type indicating use of polling in the OIDC Authorization Request.

  Once the OIDC Authorization Request has been validated and acknowledged, the request is processed and if successful (i.e. the User has been authenticated, has authorised a transaction or has provided consent to share personal information) then a Token Response (including an ID Token and Access Token) is prepared.

  In parallel, the SP starts to poll the IDGW's published Polling Endpoint based upon a minimum interval supplied by the IDGW in the Authorization Response/ Acknowledgement. Once the Mobile Connect service request has been processed, the IDGW responds to the next polling request with a Polling Response (i.e. with ID Token and Access Token).

  For any processing failures, the response includes the appropriate error and description.

Server-Initiated requests are specified by setting the response_type value in the OIDC Authorization Request to "`mc_si_async_code`" for Server-Initiated mode using Notification or "`mc_si_polling`" for Server-Initiated mode using Polling.

The relative advantages and disadvantages of each approach to Token Retrieval are summarised in Table 1.

	Advantages	Disadvantages
Using Notification	• Real time response • Minimal amount of network traffic and round trips	• Potential loss of critical information / packet loss if the communication channel is not configured correctly.
Using Polling	• SP does not need to open ports for ingress traffic.	• Multiple polling requests, if the IDGW is not ready with the data; thus, causing delays. • Greater network traffic and number of round trips compared to notification. • Potential loss of critical information / packet loss with the IDGW unable to verify that tokens have been received successfully by the SP and that they are error-free.

Table 1: Advantages and Disadvantages of Different Token Retrieval Mechanisms

Server-Initiated mode MUST only be used where its use has been pre-agreed and configured for both participating servers (i.e., the SP server application and the Operator IDGW). In other words, an SP must register for Server-Initiated mode using Notification or Server-Initiated mode using Polling with the Operator.

2.1.1 Endpoints for Server-Initiated Mode

The following Endpoints are specified to support Server-Initiated mode. IDGW Endpoints are published within the IDGW Provider Metadata, specified in [19].

For Mobile Connect Server-Initiated requests, the Operator IDGW MUST only support use of the HTTP POST method and the request MUST be a signed request object as described in Section 6 of the OIDC Core Specification [2].

2.1.1.1 Mobile Connect Server-Initiated Authorization Endpoint

The Server-Initiated Authorization Endpoint is provided by the serving Operator and is published as part of the Mobile Connect Provider Metadata which can be obtained by a SP via the Mobile Connect Discovery API.

Mobile Connect recommends maintaining a separate authorization endpoint value for SI mode services. However, Operators who have infrastructure dependencies can use single Authorization Endpoint for Device-Initiated and Server-Initiated Requests. Where a separate Server-Initiated Authorization Endpoint is in use, the RECOMMENDED value of the name for provider metadata is `"si-authorize"`.

e.g., https://operator.example.com/mc/si-authorize/

The SP MUST use TLS[4] (Transport Layer Security) [9] to access the Mobile Connect Server-Initiated Authorization Endpoint.

The SI Authorization Endpoint URI SHOULD follow the OAuth2.0 endpoint URI format as described in Section 3.1 of RFC 6749 [7].

2.1.1.2 Service Provider Notification Endpoint (when using Notification for Token Retrieval)

The SP's Notification Endpoint is registered at the Operator IDGW during SP registration and is provided by the SP in the OIDC Authorization Request. It is used by the Operator IDGW to return the Token Response to the SP. Once an SP has submitted an OIDC Authorization Request and has received an OIDC Authorization Response (acknowledgment) from the Operator IDGW, it monitors the notification endpoint for a response. It requires the request to be authenticated as described in [5]. For a given client_id, the SP can register multiple Notification Endpoints during SP registration.

2.1.1.3 Mobile Connect Server Initiated Polling Endpoint (when using Polling for Token Retrieval)

The Server-Initiated Polling Endpoint is provided by the serving Operator and is published as part of the Mobile Connect Provider Metadata which can be obtained by a SP via the Mobile Connect Discovery API.

The SP submits a Polling Request (Token Request) to the Polling Endpoint at regular intervals to request a Token Response.

The serving Operator returns the tokens in response to the Polling Requests with a response containing ID Token and Access Token.

It requires the Polling Request to be authenticated as described either using a private_key_jwt as specified in the OIDC Core Specification [2] or using mTLS[5]. mTLS implementation details are not in the scope of this specification.

2.1.2 Protocol Flow Using Notification

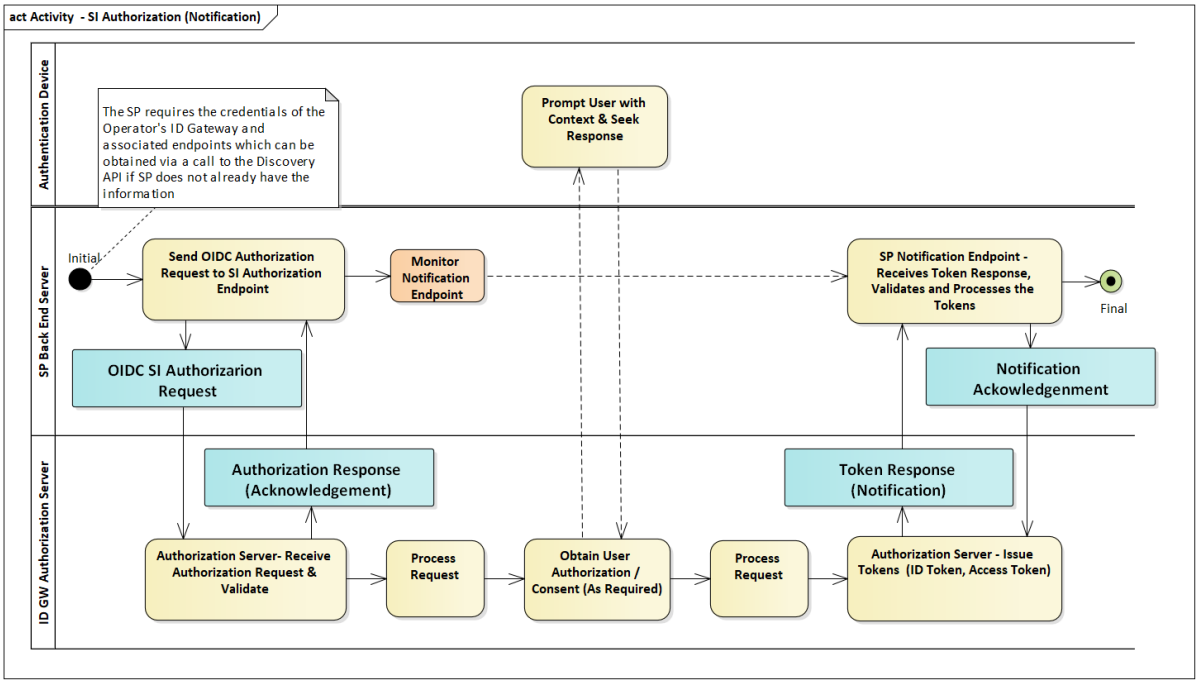

Figure 3: Mobile Connect Server-Initiated Flow Using Notification

Figure 3 illustrates the Server-Initiated asynchronous flow using Notification to obtain the ID Token, Access Token and Refresh Tokens directly and is optimised for trusted Clients. The SP MUST initiate the service requests from the SP Server (SP Client) using the OIDC

[5] Mutual Transport Layer Security (mTLS). There is no standard specified for mTLS. There might be several infrastructure setups to support mTLS. The implementation details are out of scope of this specifications. Operator and SP must negotiate offline to support mTLS.

signed request object. No re-direction via the user agent on a Consumption Device is involved.

This specification details the parameters involved in the OIDC Authorization Request and Response (Acknowledgement) and the Token Response and Acknowledgement for Server-Initiated mode using Notification. Note that it is assumed that the SP already has all the relevant details about the serving Operator's IDGW which have been cached from a previous service request or have been obtained via the Mobile Connect Discovery Service [22].

The high-level flow is as follows:

- The SP prepares the OIDC Authorization Request including a response type, client id, required LoA, the requested scope(s) and a "`request`". The request is in the form of a signed request[6] object which incorporates the details of the request. The SP needs to register the jwks_uri to enable the IDGW to validate the signature. The SP sends the OIDC Authorization Request to the Server-Initiated Authorization Endpoint at the Operator IDGW. Server-Initiated mode using Notification is requested by setting the `response_type` value to "`mc_si_async_code`".

- The Operator IDGW validates the signed request object using the SP's `jwks_uri` and public keys. It then immediately returns an OIDC SI Authorization Response (HTTP 200 OK response) to the SP and acknowledges that the request is in progress. The response includes an "`auth_req_id`" which allows the requests and responses to be correlated, an "`expires_in`" parameter which provides a timeout for the requests and provides an indication to the SP of the period for which it should monitor its Notification Endpoint. If the SP included a "`correlation_id`" in the OIDC Authorization Request then this is also returned. The SP starts listening to the SP's notification endpoint.

- The Operator IDGW selects the appropriate authenticator for the requested LoA and prompts the User to authenticate, authorize or optionally to give their consent to the sharing of User information (attributes) with the SP on their mobile device[7].

- The Operator IDGW returns the Token Response, (including an ID Token [JWT], Access Token, and optional Refresh Token), by pushing a notification to the registered Notification Endpoint. The SP Notification Endpoint authenticates the IDGW using the `client_notification_token` value as the bearer token.

- The SP validates the Token Response, extracts the PCR and issuer iss and stores in the User's profile. The SP acknowledges the response through either HTTP 204 or 200. It MAY return an error if validation fails.

[6] Refer to OIDC specs more information on request object.

[7] Note that there are use cases where consent capture from the end-user is managed by the SP and not explicitly requested by the Operator via their authentication device (i.e. mobile device). In those cases, the SP submits the consent proof to Operator using an offline business process (implementation details are out of the scope of this specification).

- Where requested (via the OIDC Authorization Request). the SP can then call the relevant resource endpoint (PremiumInfo or Mobile Connect Service-Specific Endpoint) by submitting the received Access Token to retrieve the requested attributes / claims (not shown in Figure 3).

2.1.3 Protocol Flow Using the Polling

Figure 4 illustrates the Server-Initiated asynchronous flow where the tokens are retrieved using a Polling mechanism. This specification details the parameters involved in the OIDC Authorization Request and Response and in the Polling Request (or Token Request) and Response (or Token Response) for Server-Initiated mode using Polling.

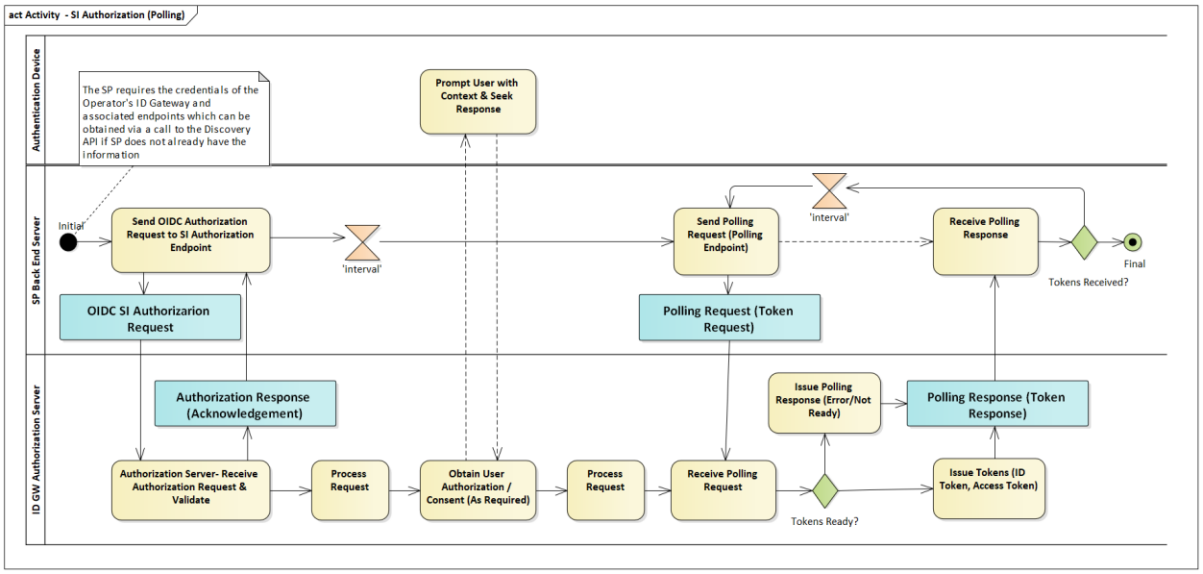

Figure 4: Mobile Connect Server-Initiated Flow Using Polling

The high-level flow is as follows:

- The SP prepares and submits the OIDC Authorization Request including a response type, client id, required LoA, the requested scope(s) and a "`request`". Server-Initiated mode using Polling is requested by setting the `response_type` value to "`mc_si_polling`".
- The Operator IDGW validates the request parameters and signed request object using the SP's `jwks_uri` and public keys. It then immediately returns an Authorization Response (HTTP 200 OK response) to the SP and acknowledges that the request is in progress. The response is the same as for SI mode using Notification except that an additional "`interval`" parameter is returned which provides an indication to the SP of the minimum interval before submitting a Polling Request.
- Where required, the Operator IDGW selects the appropriate authenticator for the requested LoA and prompts the User to authenticate, authorize or optionally to give their consent to the sharing of User information (attributes) with the SP on their mobile device.
- SP submits a signed Polling Request to the IDGW Polling Endpoint to retrieve tokens.
- If the response is ready, the Operator IDGW returns the response, i.e. the ID Token, Access Token, and optional Refresh Token, by responding to the polling request. If

tokens are not ready then a polling response is provided indicating the situation and the SP then waits for at least an "interval" of time before submitting a Polling Request again.

- The SP validates the Polling (Token) Response, extracts the PCR and issuer ID and stores them in the User's profile.
- Where requested (via the OIDC Authorization Request). the SP can then call the relevant resource endpoint by submitting the received Access Token to retrieve the requested attributes / claims (not shown in Figure 4).

3 Service Provider Client Registration – Required Information.

The SP MUST first register the appropriate credentials for their client applications with the Operator IDGW through the Mobile Connect Developer Portal [23] and API Exchange [22]. If an Operator is acting as an onboarding agent for SPs, the SPs register their services/applications with the Operator who then populates the API Exchange accordingly.

Table 2 defines the information that MUST be specified during SP registration to support Server-Initiated mode. Further details on SP registration can be found in [23].

Registration Parameter Name	Usage Category	Description
`sector_identifier_uri`	REQUIRED	The value of the `sector_identifier_uri`[8] MUST be a URL using the `https` scheme that references a file with a single JSON array of jwks_uri values for polling and / or SP notification_uri values for notification mode. It can also include `redirect_uri` for DI mode. It provides a way for a group of services under common administrative control to have consistent PCR values independent of the individual domain names. It also provides a way for SPs to change service domains without having to re-register all their Users. The values MUST be included in the elements of the array, or registration MUST fail. This MUST be validated at SP on-boarding time. If it is not registered then the registration process MUST throw an error. Mobile Connect Providers MUST utilize the `sector_identifier_uri`.
`jwks_uri`	REQUIRED	The URL for the SP's JSON Web Key Set [15] document that lists the signing keys that the IDGW uses to validate signatures from the SP. Only asymmetric signatures are supported[9].

[8] The previous version of Mobile Connect profile was considering 'host part of the redirect_uri should be same' for grouping SP applications to have consistent PCR value. This is deprecated since it has potential issues if SP hosts redirect URIs using multiple domain names, thus not allowing to group them to have same PCR value. The new sector_identifier_uri concept resolves them.

[9] The current version of Mobile Connect supports signing of the request object but does not specify encryption of the request object.

Registration Parameter Name	Usage Category	Description
		e.g. https://mc-sp.example.com/mc/jwks_uri.json All Server-Initiated Mobile Connect service requests MUST be submitted using an OIDC signed Request Object [2]. For the polling mechanism, the polling request will be signed using one of the keys from this JSON file using the mechanism as listed in OIDC Core Specification [2]
request_object_signing_alg	REQUIRED	The algorithm that MUST be used for signing the Request Objects sent to the IDGW. This algorithm MUST be used when the Request Object is passed by value (using the request parameter). All Request Objects from this SP MUST be rejected, if not signed with this algorithm. The possible algorithms are defined in the alg header parameter values (JOSE Header) for the JWS [16] (See also the JWA specification [14]). It is RECOMMENDED that the IDGW supports the RS256 signing algorithm, as a minimum.
notification_uris	OPTIONAL [REQUIRED for notification mode]	An array of Notification URI values. The notification_uri parameter value used in each OIDC Authorization Request MUST exactly match a uri element within the notification_uris array, with the matching performed as described in Section 6.2.1 of RFC 3986 (Simple String Comparison) [11].

Table 2: SP Metadata for SP Client Registration (Server-Initiated Mode)

4 OIDC Authorization Request

For Server-Initiated mode, an OIDC Authorization Request is submitted to the Operator's IDGW SI Authorization Endpoint.

- The communication with the IDGW MUST use HTTPS/TLS.
- The request MUST use POST as specified in Section 4 of the OIDC MODRNA Client Initiated Backchannel Authentication Flow specification [5]
- The SP MUST use a signed Request Object (JWT) as specified in Section 6 of the OIDC Core Specification [2].

4.1 OIDC Authorization Request Parameters

Table 3 lists the parameters to be included within a Server-Initiated OIDC Authorization Request.

Parameter	Usage Category	Description
response_type	REQUIRED	For Mobile Connect Server-Initiated requests using Notification, the value MUST be "mc_si_async_code" For Mobile Connect Server-Initiated requests using Polling, the value MUST be "mc_si_polling". The value MUST match with the value in the Request Object as defined in Table 4.
client_id	REQUIRED	The value MUST match with the value in the Request Object as defined in Table 4.
scope	REQUIRED	The value MUST match with the value in the Request Object as defined in Table 4.
request	REQUIRED	This parameter contains a Request Object which enables Mobile Connect service requests to be passed in a single, self-contained parameter and to be signed. It represents the request as a JWT whose Claims are the OIDC Authorization Request parameters. The Request Object MUST be signed. For more information see Section 6 of the OIDC Core Specification [2]. [10]

Table 3: OIDC Authorization Request parameters

4.1.1 The Signed Request Object

Server-Initiated OIDC Authorization Requests MUST use a signed Request Object to specify the parameters of the request. Only the `request` parameter is supported. OIDC Authorization Requests using the parameters detailed in Table 4 are represented as a JWT, which are respectively passed by value.

The `request` parameter, described in the Section 6.3.3 of the OIDC Core Specification [2], MUST be passed using this JWT. Support for the `request` parameter is REQUIRED in Mobile Connect.

- The `request` parameter MUST be signed, and it represents the request as a JWT, whose claims are the OIDC Authorization Request parameters specified in Section 6.3.3 of the OIDC Core Specification [2].
- Values for the `scope`, `response_type` and `client_id` parameters MUST be included using the OAuth2.0 request syntax. The values for these parameters MUST match those in the Request Object; i.e.: these parameters MUST exist both in the OIDC Authorization request and signed request object. Comparing both the values will yield more security.

[10] Mobile Connect does not adopt all the OIDC concepts listed for authorization requests as a JWT in the OIDC Core Specification. It adopts the OIDC signed request object concepts partially (i.e. only "request" is supported, but not request_uri etc.), and usage categories are changed from OPTIONAL to MANDATORY. Plain request objects and encrypted request objects are not supported.

- The Request Object MUST contain the `iss` [issuer of the Request Object, i.e. SP client ID] and `aud` [IDGW's issuer ID obtained through Mobile Connect Discovery].
- The `request` parameter MUST NOT be included in the Request Object as specified section 6.1 of OIDC Core [2].

4.1.1.1 Authorization Request Signatures

Depending on the communication channel through which the Mobile Connect service requests are sent, the integrity of the requests MUST be guaranteed by utilizing JSON Web Signature (JWS) [16] to sign the contents. The SP declares its required signing algorithms and its public keys during the SP onboarding process by registering a `jwks_uri` and `request_object_signing_alg` as specified in Table 2.

4.1.1.2 Signing

Mobile Connect only supports asymmetric signatures for Server-Initiated requests. The RECOMMENDED signing algorithm is RS256, but the signing algorithm can be negotiated offline between the SP and the IDGW.

When using Signatures, the `alg` Header Parameter value of the JOSE Header MUST be set to an appropriate algorithm as defined in JSON Web Algorithms (JWA) [14]. The private key used to sign the content MUST be associated with a public key used for signature verification published by the SP in its JWK set. If there are multiple keys in the referenced JWK set [15], a `kid` value MUST be provided in the JOSE Header. The key usage of the respective keys MUST support signing.

Further information including a RECOMMENDED approach for rotating asymmetric signing keys refer can be found in Section 10 of the OIDC Core Specification [2].

4.1.2 Request Object Parameters

Table 4 describes the Mobile Connect Server-Initiated request parameters that MUST be included within the signed Request Object (included in the SI OIDC Authorization Request as the request parameter). The IDGW MUST first process parameters listed in the Request Object. The parameters, MUST use the OAuth2.0 request syntax [7].

Note that two parameters relate to Server-Initiated Requests using Notification to retrieve the Token Response and are not required when Polling is used.

Parameter	Usage Category	Description
response_type	REQUIRED	• The OAuth 2.0 Response Type value that determines the authorization processing flow to use, including the parameters, returned from the endpoints used. • For Server-Initiated requests using Notification, the value MUST be "mc_si_async_code" to indicate that it is a Server-Initiated request with asynchronous flow. This value indicates to the IDGW to return both an Access Token and an ID Token by sending a notification. • For Server-Initiated requests using Polling, the value MUST be "mc_si_polling". This value indicates to the IDGW to return both an Access Token and an ID Token in response to a Polling Request (Token Request).
client_id	REQUIRED	OAuth 2.0 Client Identifier MUST be globally unique and needed for an OIDC Authorization Request which is valid at the IDGW Authorization Server.
scope	REQUIRED	A space delimited and a case-sensitive list of ASCII strings for OAuth 2.0 scope values. Mobile Connect OIDC Authentication Request MUST contain the scope value "openid" followed by other values depending on the specific Mobile Connect products/services being requested. For Mobile Connect "openid mc_authn" is the default scope value.
version	REQUIRED	It is a plain string and value used to identify the profile version. This is always used in conjunction with the scope parameter values. Note: Since, SI profile first version is Siv1.0. There is no backwards compatibility. See [19] for allowed values.
nonce	REQUIRED	A string value used to associate a client session with the ID Token. It is passed unmodified from Authorization Request to ID Token. The value SHOULD be unique per session to mitigate replay attacks.

Parameter	Usage Category	Description
`prompt`	OPTIONAL	Space delimited, case-sensitive ASCII string values to specify to the Authorization Server whether to prompt or not for re-authentication and consent. The value can only be: • **login**: Must prompt the user for re-authentication or consent. In case it is not possible, an error MUST be returned. • **none :** MUST NOT display any UI for re-authentication or consent to the user. If the user is not already authenticated or authentication or consent is needed to process the Authorization Request, a login_required error is returned. This can be used as a mechanism to check existing authentication or consent If scope is "`openid mc_authz`" the Authorization server MUST always prompt the User for Mobile Connect Authorisation using `context, binding_message` and `client_name` parameters which must override any setting in the prompt parameter.
`max_age`	OPTIONAL	Specifies the maximum elapsed time in seconds since the last authentication of the User. If the elapsed time is greater than this value, a re-authentication MUST be attempted. When this parameter exists in the request, the ID Token MUST contain the `auth_time` claim value.
`ui_locales`	OPTIONAL	Space separated list of User preferred languages and scripts for the UI as per RFC5646[12]. This parameter is for guidance only and in the case of unsupported locales, IDGW Authorization Server SHOULD NOT return an error. If scope value is "openid mc_authz" and a value is present the Mobile Connect Provider MUST consider this value for processing the context parameter. For instance, the value "fr-CA fr en" represents a preference for French as spoken in Canada, and then French (without a region designation), followed by English (without a region designation). An error SHOULD NOT result if some or all of the requested locales are not supported by the OpenID Provider. For more information see Reference [2].
`claims_locales`	OPTIONAL	Space separated list of User preferred languages and scripts to return the Claims as per RFC5646 [12]. This parameter is for guidance only and in the case of unsupported locales, IDGW Authorization Server SHOULD NOT return an error.

Parameter	Usage Category	Description
id_token_hint	OPTIONAL	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 User is logged in and the ID Token is still valid, 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 an audience, when included in the id_token_hint. However, the server SHOULD respond successfully when possible, even if it is not present. If scope is "openid mc_authz" this value MUST be ignored. IDGW Authorization Server MUST always display an authorization prompt to the User for approval.
login_hint	RECOMMENDED [REQUIRED if login_hint_token does not exist]	An indication to the IDGW Authorization Server on what ID to use for login. The login_hint can contain the MSISDN, encrypted MSISDN or PCR. The format MUST be as MSISDN:<Value,ENCR_MSISDN:<Value and PCR:<value The usage to transport the encrypted MSISDN will be deprecated in future releases, Instead login_hint_token will be used.
login_hint_token	OPTIONAL [REQUIRED if login_hint does not exist]	The "login_hint_token" is used to transport a User identifier (MSISDN for Mobile Connect) from the Discovery Servicer to the Operator IDGW without revealing the identifier to the SP Client [5]. The "login_hint_token" is an encrypted JSON Web Token (JWT) [13]. This token is typically created if a User has entered an MSISDN during the discovery process, and, if present, SHALL be used by the Client as login hint with the particular Operator.
acr_values	REQUIRED	Authentication Context Class Reference. Space separated string that specifies the Authentication Context Reference used during authentication processing. The acr_values are an indication of what level of assurance (LoA) is required and therefore which Authenticator should be selected by the IDGW. The SP Client can request Levels of Assurance in order of preference for a particular use case. Depending upon which Levels of Assurance are supported in the IDGW, the IDGW MUST consider only the first supported value in the list and ignore remaining values whilst processing the request. Possible values for acr_values are defined in the Mobile Connect Technical Architecture and Core Requirements document [19]. IDGW Authorization Server MUST return the achieved level of assurance in the acr claim in the ID Token.
binding_message	OPTIONAL [REQUIRED if scope = "openid mc_authz"]	Client provided plain text, "reference or ID" to interlock consumption device and authorization device for a better User experience and User assurance. The SP should provide the value of binding_message to the end-user wherever possible. Empty values are allowed. (zero length) binding_message MUST be provided by the SP Client if scope = "openid mc_authz".

Parameter	Usage Category	Description
client_name	OPTIONAL [REQUIRED if scope = "openid mc_authz"]	A short name to identify SP Client Application. It MUST be displayed on the Authentication Device. Client_name MUST be provided by the SP Client in the following scenarios. if scope = "openid mc_authz" i.e. all Mobile Connect Authorisation services.
context	OPTIONAL [REQUIRED if scope = "openid mc_authz"]	A transaction / action-based message displayed on the Authentication Device to provide context to the transaction. context MUST be provided by the SP Client if scope = "openid mc_authz".
claims	OPTIONAL	Within Mobile Connect this parameter is used to specify specific claims and associated values to be returned from the relevant Resource Endpoint in the context of a requested Mobile Connect service (via the scope parameter). The value is a JSON object listing the requested claims. Claim values can be in plain text or in a hashed form. This is only used for the Mobile Connect KYC Match services. New services will make use of the mc_claims parameter included within a Resource Request.
correlation_id	OPTIONAL	This parameter is generated by the SP only. It is used to correlate transactions across Mobile Connect components (Discovery, Mobile Connect Profile requests, IDGW internal components etc.,). It MUST be locally unique.
iss	REQUIRED	The issuer ID SHOULD be the Client ID of the SP, unless it was signed by a different party than the SP.
aud	REQUIRED	The aud value SHOULD be or include the IDGW's issuer identifier URL, obtained through the Mobile Connect Discovery API.
client_notification_token	REQUIRED (for Notification Mode only)	A unique token provided by the Client that will be used by the IDGW as a bearer token to authenticate the call-back request to send the tokens to the Client (Notification Mode) . See Section 4.1 of the OIDC MODRNA Client initiated Backchannel Authentication Flow [5]
notification_uri	REQUIRED (for Notification Mode only)	Notification URI to which the tokens will be sent. The URI MUST exactly match one of the notification URI value for the SP pre-registered at the time of onboarding. The matching MUST be performed as described in Section 6.2.1 of RFC 3986. When using the Server-Initiated flow the notification URI MUST use the HTTPS scheme.

Table 4: Server-Initiated OIDC Authorization Request – Request Object Parameters

4.2 The scope Parameter

OIDC scope values determine the specific Mobile Connect services being requested by the SP, subject to the SP being registered to use those services.

The Mobile Connect OIDC Authorization Request MUST contain the scope parameter which is a space delimited, case-sensitive list of ASCII strings (scope values). The scope values MUST include “openid”, to indicate that the request is an OpenID Connect request, followed by other values depending on the specific Mobile Connect services being requested. Multiple scope values can be requested simultaneously, subject to the SP being registered to use those “scopes”.

Scope values are defined for each Mobile Connect service in the relevant service “Definition and Technical Requirements” document.

4.3 IDGW Request Validation

The Operator IDGW SHOULD process the Server-Initiated OIDC Authorization Request as specified in the OIDC MODRNA Client Initiated Backchannel Authentication Flow [5] and Section 3.1.6 in the OIDC Core Specification [2]. When signed request object parameters are used, the following additional steps MUST be performed by the Operator IDGW to validate the Mobile Connect request:

- Confirm that the `response_type` value is either “`mc_si_async_code`” or “`mc_si_polling`”.
- Confirm that the notification endpoint (`notification_uri`) matches one of the values in the `notification_uris` that was registered by the SP requesting the Mobile Connect service. If not, then an error MUST be returned.
- Confirm that the SP is registered for Server-Initiated mode using Notification or Server-Initiated mode using Polling. If not then an error MUST be returned.
- Validate and process the signed request object[11], i.e. signature validation and assembly and validation of the parameters in the Request Object.

4.3.1 Signature Validation

The IDGW MUST perform signature validation: the `alg` Header Parameter in the JOSE Header MUST match the value of the `request_object_signing_alg` which was set during the SP registration process. The signature MUST be validated against the appropriate key for that `client_id` and algorithm. The IDGW MUST return an `invalid_request` error if signature validation fails. (See Section 6 of the OIDC Core Specification [2]).

4.3.2 Validation and Assembly of Request Parameters

The IDGW MUST assemble the set of Authorization Request parameters to be used from the Request Object value and the other SI OIDC Authorization Request parameters (minus the request parameter itself). If the same parameter exists both in the Request Object and the OIDC Authorization Request parameters, the parameter in the Request Object is used. Using the assembled set of Authorization Request parameters, the Authorization Server then validates the request. (See Section 6.3.3 of the OIDC Core Specification [2]).

[11] Encrypted request object is not required by Mobile Connect.

5 Authorization Response or Request Acknowledgement

The Operator IDGW MUST return Server-Initiated specific values as described in Table 5. In the case of the Mobile Connect service request failing, the Operator IDGW Authorization Server MUST return an error in the OIDC Authorization Response as described in Table 11.

After receiving and validating a valid and signed OIDC Authorization Request from the SP, the IDGW returns a successful response that includes HTTP 200 OK response to the SP to indicate that the Mobile Connect service request is validated and accepted. The response uses the application/json media type.

Parameter	Usage Category	Description
`auth_req_id`	REQUIRED	Unique per request-based ID, for more details refer to [5].
`expires_in`	REQUIRED	Expiration time of auth_req_id. For more details refer to [5]. The expires_in value MUST be equal to the timeout value for the SP Notification Endpoint (i.e. the maximum time an SP will wait to receive a notification from the IDGW). Any notification after this interval MUST be discarded.
`correlation_id`	OPTIONAL [REQUIRED if provided in the OIDC Authorization Request]	The correlation_id submitted through the OIDC Authorization Request, if present.
`interval`	OPTIONAL [REQUIRED if the SP is registered for Polling]	The minimum amount of time in seconds that the SP Client SHOULD wait between polling requests to the Polling Endpoint. This parameter exists, if the SP is registered for use of polling and the `respose_type` value in the OIDC Authorization Request is "`mc_si_polling`".

Table 5: Authorization Response Parameters

6 Token Retrieval

The use of Notification or Polling distinguishes how Tokens can be retrieved from the IDGW Authorization Server as a result of a successful Authorization in Server-Initiated mode.

6.1 Token Retrieval Using Notification

The Operator IDGW MUST generate the notification request that contains tokens which is sent to the SP's Notification Endpoint. The Token Response SHALL comply with that specified in the OIDF CIBA Notification Flow [5].

The SP must authenticate the IDGW using the `client_notification_token` value provided to the IDGW in the Mobile Connect OIDC Authorization Request.

Table 6 lists the parameters to be included within the Token Response after a successful OIDC Authorization:

Parameter	Required Category in Profile	Description
auth_req_id	REQUIRED	The auth_req_id received from the Operator IDGW Authorization Server, because of the successful OIDC Authorization Request. For more details refer [5].
access_token	REQUIRED	OAuth 2.0 access_token used to get the PremiumInfo/ User Info object from the PremiumInfo endpoint and can be reused for accessing other protected resources, if required. This parameter MUST be utilized using either the Authorization header field or a form-encoded POST body parameter.
token_type	REQUIRED	MUST be "bearer" as defined in RFC 6749 section 7.1 [7] and RFC6750 [8], unless another token_type value as agreed between the SP Client and IDGW Authorization Server. For the Mobile Connect Profile, token_type=bearer is the RECOMMENDED value.
id_token	REQUIRED	An additional security token used in OIDC to provide Claims about the Authentication of a User and potentially other requested Claims. The format of the ID Token is a JSON Web Token (JWT).
expires_in	REQUIRED	Expiration time of the Access Token in seconds since the response was generated.
refresh_token	OPTIONAL	OAuth 2.0 refresh token to get a new Access Token using the same authorization grant through a grant_type parameter.
correlation_id	REQUIRED [if provided in the OIDC Authorization Request]	The correlation_id submitted through the OIDC Authorization Request, if present.

Table 6: Token Response (Notification)

6.1.1 Notification Acknowledgement

On receipt of the Token Response from the Operator IDGW, the SP MUST send either an HTTP 204 or an HTTP 200 OK response as an acknowledgement. The IDGW MUST ignore the content if 200 OK is received.

6.2 Token Retrieval Using Polling

6.2.1 Polling Request

The SP MUST generate the polling request as specified in [5] using the private_key_jwt method and encoding scheme defined in the Section 9 of the OIDC Core Specification [2].

By default the SP should send the polling request using the private_key_jwt method but as an alternative mechanism, mTLS[12] can also be used. Infrastructure setup for mTLS and implementation of mTLS are out of scope of this specification. Table 7 specifies the Polling Request parameters.

Parameter	Usage Category	Description
`client_assertion_type`	REQUIRED	The value must be "`urn:ietf:params:oauth:client-assertion-type:jwt-bearer`".
`client_assertion`	REQUIRED	The JWT that contains relevant parameters as specified in Table 8.
`correlation_id`	OPTIONAL [REQUIRED if provided in the OIDC Authorization Request	The correlation_id submitted through the OIDC Authorization Request, if present.
`auth_req_id`	REQUIRED	The `auth_req_id` received from the Operator IDGW Authorization Server in the Authorization Response/Acknowledgement, because of the successful authorisation request. For more details refer to [5].
`client_id`	REQUIRED	OAuth 2.0 Client Identifier MUST be globally unique and needed for an OIDC Authorization Request which is valid at the IDGW Authorization Server.
`grant_type`	REQUIRED	For all Polling Requests the value of the grant_type parameter MUST have the value: "`urn:openid:params:mc:grant-type:server_initiated`".

Table 7: Polling Request Parameters

Table 8 lists all the parameters that MUST be included the client_assertion JWT. The assertion is only valid for a single Polling Request and a new client assertion MUST be generated for each polling request. Further details can be found in the OIDC Core Specification [2].

Parameter	Usage Category	Description
`iss`	REQUIRED	This MUST contain the client_id (issuer) of the SP that is registered with the IDGW.
`sub`	REQUIRED	The value MUST contain the client_id of the SP that is registered with the IDGW.
`aud`	REQUIRED	The value MUST be the polling endpoint of the IDGW. The value identifies the IDGW as an intended audience for this token.

[12] There is no standard specified for mTLS. There might be several infrastructure setups to support mTLS. The implementation details are out of scope of this specification. Operator and SP must negotiate offline to support mTLS.

Parameter	Usage Category	Description
jti	REQUIRED	The value must be a GUID (globally unique identifier), which is used to prevent reuse of the JWT.
exp	REQUIRED	The expiration time on or after which the ID Token and Access Tokens MUST NOT be accepted for processing. If the expiration time is elapsed then the IDGW must return an error, instead of processing the request.
iat	REQUIRED	The JWT Creation time. It should be used in conjunction with exp parameter to determine whether to process the request to return ID Token and Access Token.

Table 8:Client Assertion JWT Parameters

6.2.2 Polling Response

Table 9 shows the parameters that are reurned in the Token Response (or Polling Response) when using the Polling mechanism for Token retrieval.

Parameter	Required Category in Profile	Description
access_token	REQUIRED	OAuth 2.0 Access Token used to get the PremiumInfo/ User Info object from the PremiumInfo endpoint and can be reused for accessing other protected resources, if required. This parameter MUST be utilized using either the Authorization header field or a form-encoded POST body parameter.
token_type	REQUIRED	MUST be "bearer" as defined in RFC 6749 section 7.1 [7] and RFC6750 [8], unless another token_type value as agreed between the SP Client and IDGW Authorization Server. For the Mobile Connect Profile, token_type=bearer is the RECOMMENDED value.
id_token	REQUIRED	An additional security token used in OIDC to provide Claims about the Authentication of a User and potentially other requested Claims. The format of the ID Token is a JSON Web Token (JWT).
expires_in	REQUIRED	Expiration time of the Access Token in seconds since the response was generated.
refresh_token	OPTIONAL	OAuth 2.0 refresh token to get a new Access Token using the same authorization grant through a grant_type parameter.
correlation_id	OPTIONAL [REQUIRED if provided in the OIDC Authorization Request]	The correlation_id submitted through the OIDC Authorization Request, if present.

Table 9: Token Response (Polling)

6.3 ID Token

The primary extension that OpenID Connect makes to OAuth 2.0 in order to enable Users to be Authenticated is the ID Token data structure. The ID Token is a security token that contains Claims about the Authentication of a User by an Authorization Server when using a Client, and potentially other requested Claims. It is returned along with the Access Token.

The ID Token is represented as a JSON Web Token (JWT) [13] and is created and returned by the IDGW. The JWT is signed by the IDGW using JSON Web Signatures (JWS) [16]. See also the OIDC Core Specification [2].

Table 10 describes the contents of the ID Token for Mobile Connect Server-Initiated Mode.

Parameter	Required Category in Profile	Description
`iss`	REQUIRED	Issuer Identifier. It is a case-sensitive HTTPS based URL, with the host. It MAY contain the port and path element (OPTIONAL) but no query parameters.
`sub`	REQUIRED	Subject identifier. A globally unique identifier for the User to work in a federated environment. It is a case-sensitive ASCII string with a maximum length of 255. The sub value MUST not contain the MSISDN. Within Mobile Connect this is populated by a Pseudonymous Customer Reference (PCR) which is a system-facing unique identifier that links a User's MSISDN to a SP (SP). It is generated by the IDGW. This is defined in Mobile Connect Techncial Architecture and Core Requirements [19].
`aud`	REQUIRED	The intended audience for the ID Token. It is an array of case-sensitive strings. It MUST contain the client_id of the SP Client, and MAY contain identifiers of other OPTIONAL audiences. If there is one audience, the aud value MAY be a single case-sensitive string OR an array of case-sensitive strings with only one element. An implementation MUST support both scenarios.
`exp`	REQUIRED	The expiration time after which the ID Token MUST NOT be accepted for processing. The format is the number of seconds from 1970-01-01T0:0:0Z as measured in UTC until the date/time specified. If scope = "openid mc_authz", the implementor MUST give the lowest possible time but no more than a few minutes. The processing of this parameter requires that the current date/time MUST be before the expiration date/time listed in the value. See RFC 3339 [10] for details regarding date/times in general and UTC in particular.

Parameter	Required Category in Profile	Description
iat	REQUIRED	The time of issue of the ID Token. The format is the number of seconds from 1970-01-01T0:0:0Z as measured in UTC until the date/time specified.
auth_time	REQUIRED	Time of User authentication or authorization. The format is the number of seconds from 1970-01-01T0:0:0Z as measured in UTC until the date/time specified. See RFC 3339 [10] for details regarding date/times in general and UTC in particular.
nonce	REQUIRED	Opaque string value to associate the SP Client session with the ID Token, to avoid the replay attacks. The nonce value MUST be same as the nonce used in the Authorization request.
at_hash	REQUIRED	A base64url encoded value of the hash of the access_token The OIDC Core Specification, Section 3.1.3.6 [2] describes the process for generating the Access Token hash value (at_hash).
acr	REQUIRED	Authentication context class reference. This is a case-sensitive string, representing the achieved authentication by the IDGW Authorization Server. The value returned is one of the possible values for the acr_values parameter in the OIDC Authorization Request.
amr	REQUIRED	Authentication Methods References. An array of case-sensitive strings to indicate the authentication method used. The possible amr values are defined in Mobile Connect Technical Architecture and Core Requirements [19].
azp	REQUIRED	Authorised Party – the party to which the ID Token is issued. If present, it MUST contain the Client ID. This Claim MUST be present when the ID Token has a single audience value, and that audience is different than the authorized party. It MAY be included even when the authorized party is the same as the sole audience. The azp value is a case sensitive string containing a String or URI.
displayed_data	OPTIONAL [REQUIRED when scope is "openid mc_authz"]	Displayed data on the Authentication Device. Value is derived by combining client_name, context and binding_message.
dts	OPTIONAL [only applicable for future PKI Mobile Connect services]	Signature. The data signed includes displayed_data and dts_time. Optional extension that can be used in conjunction with a future LoA4 enabled Mobile Connect services that require a User-signed response to be returned to the SP
upk	OPTIONAL	User Public Key or User certificate / reference to the certificate.

Parameter	Required Category in Profile	Description
	[only applicable for future PKI Mobile Connect services]	Optional extension that can be used in conjunction with a future LoA4 enabled Mobile Connect services that require a User-signed response to be returned to the SP
dts_time	OPTIONAL [only applicable for future PKI Mobile Connect services]	The time of signing the text. The format is the number of seconds from 1970-01-01T0:0:0Z as measured in UTC until the date/time specified. Optional extension that can be used in conjunction with a future LoA4 enabled Mobile Connect services that require a User-signed response to be returned to the SP
hashed_login_hint	REQUIRED	Hashed login_hint or login_hint_token value to mitigate security threats. If the request is made using login_hint, then it MUST return the hashed login hint value and if the request is made using login_hint_token then it MUST return the hashed login_hint_token value. See Mobile Connect Technical Architecture and Core Requirements [19] for more information on hashing algorithms. The SHA256 algorithm SHOULD only be used for backwards compatibility only.
recipient	REQUIRED (for Notification Mode only)	The notification URI specified in the OIDC Authorization Request.

Table 10: ID Token Claims (Server-Initiated Mode)

6.4 Access Token

Access tokens are credentials used to access protected resources (User attributes). An access token is a string representing an authorization issued to the SP Client. The string is opaque to the SP Client. Tokens represent specific scopes and durations of access, granted by the User (through appropriate User consent), and enforced by the IDGW Resource Server and IDGW Authorization Server. Further information on the Access Token can be found in RFC6749 [7] and RFC6750 [8].

6.5 Refresh Token (OPTIONAL)

Refresh tokens are credentials used to obtain access tokens. Refresh tokens are issued to the client by the authorization server and are used to obtain a new access token when the current access token becomes invalid or expires.

A refresh token is a string representing the authorization granted to the client by the resource owner. The string is usually opaque to the client. The token denotes an identifier used to retrieve the authorization information. Unlike access tokens, refresh tokens are intended for use only with the Authorization Server and are never sent to Resource Servers.

Further information can be found in RFC6749 [7]

7 Security Considerations

The security considerations listed in the OIDF specifications SHALL be considered in the Mobile Connect implementation:

- Section 16, OIDC Core Specification [2].
- Section 7, OIDC MODRNA Client-Initiated Backchannel Authentication Flow [5].

Annex A Generic[13] Error Codes and Descriptions for Server-Initiated Mode

A.1 Authorization Response – Error Codes and Descriptions

After receiving a signed OIDC Authorization Request from the SP, the IDGW validates the request before processing the request. If there is any problem with that validation then an error response is returned. The error response uses the application/json media type with the format shown in Table 11.

The OIDC Authorization Request error response is an OIDC Authentication Error Response as specified in Section 3.1.2.6 of the OIDC Core Specification [2] but delivered in the form of a Token Error Response as specified in Section 3.1.3.4 of [2].

Parameter	Usage Category	Description
correlation_id	REQUIRED if provided in the OIDC Authorization Request	The correlation_id submitted through the OIDC Authorization Request, if present.
error	REQUIRED	Error as defined in [2]
error_description	REQUIRED	Error description as defined in [2].

Table 11: Authorization Response – Required Fields to Report an Error

Error codes and descriptions are described in the following tables:

Table 12 lists the possible errors generated as a result of validating the basic Server-Initiated OIDC Authorization Request including errors where the parameter passed on the request does not match the equivalent parameter in the signed Request Object in the request parameter.

Table 13 lists the possible errors that arise in validating the signed Request Object parameters in the OIDC Authorization Request.

[13] Generic means that they apply irrespective of the specific Mobile Connect service – Service-specific error codes and descriptions (if any) are defined in the relevant service "Definition and Technical Requirements" document

Error Scenario	HTTP Response	Error code	Error Description (Suggested Text)
SP credentials are invalid, and the IDGW has rejected the Request	401	access_denied (or) invalid_client	The client is not authorised to make Mobile Connect requests and the IDGW denies the request.
Invalid login hint or login_hint_token	Bad Request 400	invalid_request	Unable to find the corresponding Mobile Connect account.
response_type parameter is missing	Bad Request 400	Invalid_request	REQUIRED parameter response_type is missing.
response_type parameter exists but value is invalid (or) response_type parameter exists, and the value is valid as defined in the specs, but value is not supported by the IDGW (or) response_type parameter exists, and the value is valid, but not matching the request object response_type value.	Bad Request 400	invalid_request (or) unsupported_response_type	REQUIRED parameter response_type is missing (or) invalid (or) malformed request; response_type values do not match.
client_id parameter does not exist	Bad Request 400	access_denied (or) invalid_request	REQUIRED parameter client ID does not exist.
client_id parameter exists, but it has an invalid value	Bad Request 400	access_denied (or) invalid_client	Unknown client ID.
client_id parameter exists, but the value does not match the request object client_id value	Bad Request 400	invalid_request	Malformed request, ambiguous client ID values.
client_id is valid, but not allowed to make Mobile Connect service requests.	Bad Request 400	unauthorized_client(or) access_denied	The client is not allowed to make Mobile Connect service requests.
scope parameter is missing	Bad Request 400	Invalid_request	REQUIRED parameter scope is missing.

`scope` parameter exists, it does not contain "openid"	Bad Request 400	`invalid_scope`	REQUIRED parameter scope parameter is missing.
`scope` parameter exists, but it does not match the request object scope value	Bad Request 400	`invalid_request`	Malformed request, ambiguous scope values.
`scope` parameter exists, request scope is not supported by the Operator IDGW	Bad Request 400	`invalid_scope`	Service is not available.
`scope` parameter exists, IDGW has published the scope value in the provider metadata, but IDGW does not support the requested scope temporarily	Service Unavailable 503	`temporarily_unavailable`	Service is not available temporarily.
Request object parameter does not exist (or) request object parameter exists, but value is not valid	Bad Request 400	`invalid_request`	REQUIRED parameter request is missing.
Signature validation of request object failed	Bad Request 400	`invalid_request (or) invalid_request_object`	Malformed request, invalid signature.
System connection problem (or) Expiration in server	Service Unavailable 503	`server_error`	Service is not available,
Multiple requests for the same MSISDN sent at the same time	Internal Server Error 500	`server_error`	The User is busy with another transaction.
Unexpected error [Internal to IDGW]	Internal Server Error 500	`server_error`	Internal Server Error.
IDGW time-out due to internal error.	Internal Server Error 500	`server_error`	Timeout: Server internal error.

Table 12: Generic Error Responses for OIDC Authorization Request Validation

Error Scenario	HTTP Response	Error code	Error Description (Suggested Text)
MSISDN/ENCR MSISDN/ PCR provided does not belong to the Operator.	Bad Request 400	access_denied	User is not recognized. [OR] Unknown User.
MSISDN/ENCR_MSDISDN belongs to the Operator, but Mobile Connect services are not enabled, Note: This applies if IDGW policy does not allow “on-the-fly” User registration	Bad Request 400	access_denied	"User is not registered" [OR] "Unknown User".
response_type parameter is missing or invalid	Bad Request 400	invalid_request	REQUIRED parameter response_type is missing, or value is invalid.
client_id parameter is missing	Bad Request 400	invalid_request (or) access_denied	REQUIRED parameter client_id is missing.
client_id parameter value is invalid	Bad Request 400	invalid_request (or) unauthorized_client (or) access_denied	The client is not authorized to request an authorization code.
client_id is valid, but not allowed to make Mobile Connect service requests	Bad Request 400	access_denied (or) unauthorized_client	The Client is not allowed to make Mobile Connect service requests.
scope parameter is missing (or) scope value "openid" is missing (or) invalid scope values (ex: "abcd")	Bad Request 400	invalid_request (or) invalid_scope	REQUIRED parameter scope is missing (or) invalid scope value.
version parameter is missing (or) version parameter value is invalid	Bad Request 400	invalid_request	REQUIRED parameter version is missing (or) invalid.
nonce parameter is missing, or nonce parameter exists, but value is empty	Bad Request 400	invalid_request	REQUIRED parameter nonce is missing (or) invalid.
login_hint and login_hint_token parameters are missing	Bad Request 400	invalid_request	REQUIRED parameters login_hint_token (or) login_hint does not exist.
login_hint and login_hint_token both exist	Bad Request 400	invalid_request	Malformed request, duplicate parameter entries.

`login_hint` (or) `login_hint_token` value is invalid	Bad Request 400	`invalid_request`	Invalid value for `login_hint` (or) `login_hint_token`.
`acr_values` parameter is missing (or) `acr_values` exist but contains invalid value, other than supported values	Bad Request 400	`invalid_request`	REQUIRED parameter acr_values are missing (or) invalid values.
Same parameter exists multiple times	Bad Request 400	`invalid_request`	Multiple parameter names in the OIDC Authorization Request. Malformed request.
`claims` parameter exists but it does not have any value (or) invalid values.	Bad Request 400	`invalid_request`	`claims` value is invalid,
GET request is used, and the request parameters are NOT serialized using URI string serialization, IDGW able to validate the `redirect_uri`.	Bad Request 400	`invalid_request`	GET request invalid serialization.
POST request is used, the request parameters are NOT serialized using form serialization, IDGW can validate the redirect_uri	Bad Request 400	`invalid_request`	POST request Invalid serialization.
`max_age` parameter exists, and it has an invalid value	Bad Request 400	`invalid_request`	Invalid `max_age` value.
Multiple problems in OIDC Authorization Request [redirect URI is valid]	Bad Request 400	`invalid_request`	Malformed request multiple problems exist.
`correlation_id` exists, but it has empty value	Bad Request 400	`invalid_request`	Invalid correlation_id value.
`client_name` exists, but it has empty value (or) `client_name` parameter exists, but the provided value is not a registered `client_name` with Operator IDGW and is invalid	Bad Request 400	`invalid_request`	Invalid `client_name` value.
`client_notification_token` REQUIRED parameter is missing (or) `client_notification_token` exists, but it has an invalid value	Bad Request 400	`invalid_request`	REQUIRED parameter client_notification_token is missing (or) invalid.

`notification_uri` parameter is missing (or) `notification_uri` exists but it is not registered with IDGW (or) has an invalid value	Bad Request 400	`invalid_request`	REQUIRED parameter `notification_uri` is missing (or) invalid.
`iss` parameter is missing (or) `iss` parameter exists, but it has an invalid value	Bad Request 400	`invalid_request`	REQUIRED parameter SP's `iss` parameter is missing (or) invalid.
`aud` parameter is missing (or) `aud` parameter exists, but it has an invalid value	Bad Request 400	`invalid_request`	REQUIRED parameter `aud` parameter is missing (or) invalid.
`login_hint` parameter contains plain MSISDN and IDGW does not allow the SP (client_id) to send the plain MSISDN (Previously TSP level 1)	Bad Request 400	`access_denied`	SP is not allowed to send the plain MSISDN.
SP has sent `acr_value = 0`, indicating SP has captured the consent IDGW policy does not allow the SP to capture consent (IDGW does not explicitly capture the consent from the user for any reason (i.e. not available, policy does not allow) [Previously TSP level 2]	Bad Request 400	`access_denied`	SP is not allowed to capture the consent.
SP has sent request, by capturing the consent (through business process) and IDGW does not allow SP (client_id) to capture the consent. [Previously TSP level 2] Note : SP does not send acr_values = 0 (new feature).	Bad Request 400	`access_denied`	SP is not allowed to capture the consent.

Table 13: Generic Error Responses - Request Object Parameter Validation

A.2 Token Response (Using Notification) – Error Codes and Descriptions

In the event of an unsuccessful OIDC Authorization, where validation of the requests has been completed but where the processing of the request has failed, the Token Response to the SP's Notification Endpoint shall indicate an error and provide an error description. Table 14 shows the required fields in the Token Response in the event of an error when using Notification. Generic and all service-specific errors MUST be returned in this format.

Parameter	Usage Category	Description
auth_req_id	REQUIRED	A unique ID for this OIDC Authorization Request (See also [5]).
correlation_id	Only REQUIRED if provided in the OIDC Authorization Request	The correlation_id submitted through the OIDC Authorization Request, if present.
error	REQUIRED	Error as defined in [2]
error_description	REQUIRED	Error description as defined in [2]

Table 14: Token Response (Notification) – Required Fields to Report an Error

The following is an example of the token error response

```
POST /<token_notification> HTTP/1.1
Content-Type: application/json
Host: spserver.example.com
Authorisation: bearer <client_notification_token>
{
      "correlation_id": "42da5b19-457a-4d30-a5c4-038c62dccbb0",
      "auth_req_id":"162537",
    "error":" invalid_request,
    "error_description": "service is not available"
}
```

Table 15 describes the generic errors that can be returned in the Token Response.

Error Scenario	Error code	Error Description (Suggested Text)
Authenticator unreachable / expiration in server	server_error	Service is not available,
System connection problems (internal to IDGW)	server_error	Connection problem.
Unexpected error (internal to IDGW)	server_error	Internal server error.

Table 15: Generic Error Responses – Token Response Using Notification

A.3 Notification Acknowledgement (When Using Notification) – Error Codes and Descriptions

In the situation where the Operator IDGW has returned an OIDC Authorization Response but there is an error in the SP's ability to process that response, an error is returned through the server to server acknowledgement from the SP. The SP MUST return the errors through standard HTTP error mechanisms.

The IDGW must not respond to SP with another notification request; the error codes should be logged internally. IDGW must not do default redirection if it received 3xx errors in the acknowledgement for security reasons..

Table 16 describes the error responses that can be included within a Notification Acknowledgement.. The Notification Acknowledgement error response takes the following form:

```
HTTP/1.1 400 Bad Request
  Content-Type: application/json
  Cache-Control: no-store
  Pragma: no-cache

  {
   "correlation_id": "42da5b19-457a-4d30-a5c4-038c62dccbb0",
   "error": "invalid_request",
   "error_description": "invalid tokens"
  }
```

Error Scenario	HTTP Response	Error code	Error Description (Suggested Text)
Invalid JWT received from Operator IDGW [ID Token]	Bad Request 400	`invalid_request`	Invalid JWT [ID Token] received.
Invalid `access_token` is received	Bad Request 400	`invalid_request`	Invalid `access_token` received
Invalid auth_req_id	Bad Request 400	`invalid_request`	Malformed request, unable to identify the response.
Invalid correlation_id	Bad Request 400	`invalid_request`	Malformed request, unable to correlate the response.
`token_type` parameter is missing	Bad Request 400	`invalid_request`	REQUIRED parameter `token_type` is missing
`token_type` parameter exists, but the value is not Bearer	Bad Request 400	`invalid_request`	Invalid token type value
`expires_in` parameter does not exist	Bad Request 400	`invalid_request`	REQUIRED parameter `expires_in` does not exist.
`expires_in` parameter exists, but value is invalid	Bad request 400	`invalid_request`	Invalid `expires_in` value
ID Token parameter does not exist	Bad Request 400	`invalid_request`	REQUIRED parameter ID Token does not exist
Access Token parameter does not exist	Bad Request 400	`invalid_request`	REQUIRED parameter access token does not exist
System connection problem (or) Expiration in server	Service Unavailable 503	`server_error`	Service is not available,
Unexpected error [Internal to SP]	Internal Server Error 500	`server_error`	Internal Server Error
SP server time-out due to an internal error.	Internal Server Error 500	`server_error`	Timeout: Server internal error.

Table 16: Error Responses from SP - Notification Acknowledgement

A.4 Token (Polling) Response – Error Codes and Descriptions

In the event of an unsuccessful OIDC Authorization, where validation of the requests has been completed but where the processing of the request has failed, the Token Response in response to a Polling Request shall indicate an error and provide an error description. Table 17 shows the required fields in the Token Response in the event of an error when using Polling. Note that the auth_req_id is not required. Generic and all service-specific errors MUST be returned in this format.

Parameter	Usage Category	Description
correlation_id	Only REQUIRED if provided in the OIDC Authorization Request	The correlation_id submitted through the OIDC Authorization Request, if present.
error	Mandatory	Error as defined in [2]
error_description	Mandatory	Error description as defined in [2]

Table 17: Token Response (Polling) – Required Fields to Report an Error

Table 18 describes the generic errors that can be returned in the Token Response. Note that when using Polling, the Token Response uses a standard HTTP response code in response to the Polling Request whereas when using Notification, the Token Response is an HTTP POST to the SP's Notification Endpoint.

Error Scenario	HTTP Response	Error code	Error Description (Suggested Text)
The Authorization Server doesn't recognise the auth_req_id that the SP Client has submitted (Polling mode only).	Bad Request 400	`invalid_grant`	auth_req_id is not recognised.
The auth_req_id does not exist in the request (Polling mode only).	Bad Request 400	`invalid_request`	REQUIRED parameter auth_req_id is missing.
The authorization request is still pending as the User hasn't yet been authenticated (Polling mode only).	Bad Request 400	`authorization_pending`	Pending authorisation from the user.
The SP Client is polling too quickly and should back off at a reasonable rate (Polling mode only).	Bad Request 400	`slow_down`	Request is submitted too early.
The auth_req_id has expired. The SP Client will need to make a new OIDC Authorization Request.	Bad Request 400	`expired_token`	auth_req_id has expired.
The polling request is missing a required parameter	Bad Request 400	`invalid_request`	Required parameter is missing
includes an unsupported parameter value (other than grant type),	Bad Request 400	`invalid_request`	Unsupported parameter value.
Malformed request [other than above mentioned errors]	Bad Request 400	`Invalid_request`	Malformed request.
client_id does not correspond to auth_req_id	Bad Request 400	`Invalid_request`	Malformed auth_req_id.
Client_id is missing	Bad Request 400	`invalid_request`	Required parameter client_id is missing
Client authentication failed (e.g., unknown client, signature failure)	Bad Request 401	`invalid_client`	Client authentication failed
The grant_type is missing	Bad Request 400	`Invalid_request`	REQUIRED parameter grant_type is missing

The provided grant_type value is correct but it does not relate to auth_req_id.	Bad Request 400	`invalid_grant`	Required parameter grant_type is incorrect
The grant type parameter exists, but it is invalid	Bad Request	`unsupported_grant_type`	Grant type value is invalid.
Authenticator unreachable / expiration in server	Service Unavailable 503	`server_error`	Service is not available,
System connection problems (internal to IDGW)	Internal Server Error 500	`server_error`	Connection problem.
Unexpected error (internal to IDGW)	Internal Server Error 500	`server_error`	Internal server error.

Table 18: Generic Error Responses – Token Response (Using Polling)

Annex B Example Requests and Responses

B.1 OIDC Authorization Request - Signed Request Object (Using Notification)

The following is an example of a signed Request Object after base64url encoding and signing (with line wraps within values for display purposes only):

```
POST /authorize HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Host: operatoridgw.example.com
Authorization: Basic <Berdaz1234dzsdf>
   response_type=mc_si_async_code
   &client_id=s6BhdRkqt3
   &scope=openid%20Fmc_authn
&request=
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJzNkJoZFJrcXQzIiwiaWF0IjoxNT
I5NzEyMDY2LCJleHAiOjE1NDQ0NTY5MDAsImF1ZCI6Imh0dHBzOi8vbWMub3BlcmF0b3IuZXhhb
XBsZS5jb20iLCJzdWIiOiJzNkJoZFJrcXQzIiwicmVzcG9uc2VfdHlwZSI6Im1jX3NpX2FzeW5j
X2NvZGUiLCJjbGllbnRfaWQiOiJzNkJoZFJrcXQzIiwic2NvcGUiOiJvcGVuaWQgbWNfYXV0aG4
iLCJ2ZXJzaW9uIjoibWNfc2lfcjJfdjEuMobile
ConnectIsIm5vbmNlIjoiYTdkOGRhODQtYTkzNi00MWU3LWEyMGItN2UyYmZhZTkzOTdjIiwicH
JvbXB0IjoibG9naW4iLCJtYXhfYWdlIjoiODY0MDAiLCJ1aV9sb2NhbGVzIjoiZW4tdXMiLCJjb
GFpbXNfbG9jYWxlcyI6ImVuLXVzIiwibG9naW5faGludCI6IjQ0NzQxMTE4ODI1OCIsImFjcl92
YWx1ZXMiOiIyIiwiYmluZGluZ19tZXNzYWdlIjoiSW50ZXJsb2NrIG1zZyIsImNsaWVudF9uYW1
lIjoic3BfY2xpZW50X25hbWUiLCJjb250ZXh0IjoiY2FuIHlvdSBhdXRoZW50aWNhdGUgZm9yIG
15IFNQIiwiY29ycmVsYXRpb25faWQiOiJlYzNmNjVmNS00MzhkLTRjMzAtYTM1ZS1iYzhjYTUwZ
GU1MTQiLCJjbGllbnRfbm90aWZpY2F0aW9uX3Rva2VuIjoiNzhiYzZjOTgtYWEyNy00NzEwLWFk
MTAtMTJkYmM4ZmY4ZjIyIiwibm90aWZpY2F0aW9uX3VyaSI6Imh0dHBzOi8vd3d3LnNlcnZpY2V
wcm92aWRlci5leGFtcGxlLmNvbS9ub3RpZmljYXRpb25fZW5kcG9pbnQiLCJqdGkiOiI1NjYwNm
VmOC04OGY0LTQzNmItYTVmYy05ZmI3Yzg4YjdiODgifQ.N0c5A7mdKPTuJfqjRIEfBzO2Bx9Dm3
2Aw8aPbqhLkes

}
```

The following is an example of a Request Object before base64url encoding and signing:

```
Headers :
{
  "typ": "JWT",
  "alg": "RS256"
}

Claims :
{
  "iss": "s6BhdRkqt3",
  "iat": 1529712066,
  "exp": 1529713261,
  "aud": "https://mc.operator.example.com",
  "sub": "s6BhdRkqt3",
  "response_type": "mc_si_async_code",
```

```
  "client_id": "s6BhdRkqt3",
  "prompt": "login",
  "scope": "openid mc_authn",
  "version": "mc_si_r2_v1.0",
  "nonce": "a7d8da84-a936-41e7-a20b-7e2bfae9397c",
  "max_age": "86400",
  "ui_locales": "en-us",
  "claims_locales": "en-us",
  "login_hint": "447411188258",
  "acr_values": "2",
  "binding_message": "interlock id",
  "client_name": "sp_client_name",
  "context": "can you authenticate for my SP",
  "correlation_id": "ec3f65f5-438d-4c30-a35e-bc8ca50de514",
  "client_notification_token": "78bc6c98-aa27-4710-ad10-12dbc8ff8f22",
  "notification_uri":
"https://www.serviceprovider.example.com/notification_endpoint"
}
```

B.2 Authorization Response (Using Notification)

The following is an example of a successful Authorization Response from the IDGW:

```
HTTP/1.1 200 OK
  Content-Type: application/json
  Cache-Control: no-store
  Pragma: no-cache
  {
   "auth_req_id": "1234567",
   "correlation_id": "42da5b19-457a-4d30-a5c4-038c62dccbb0",
   "expires_in": 3600
  }
```

The following is an example for an unsuccessful Authorization Response from the IDGW:

```
HTTP/1.1 400 Bad Request
  Content-Type: application/json
  Cache-Control: no-store
  Pragma: no-cache

  {
   "correlation_id": "42da5b19-457a-4d30-a5c4-038c62dccbb0",
   "error": "invalid_request",
   "error_description": "acr values are missing"
  }
```

B.3 Token Response (Token Notification)

The following is an example of successful notification to SP.

```
POST /token_notif HTTP/1.1
Content -Type: application/json
Host: spserver.example.com
Authorization: bearer 78bc6c98-aa27-4710-ad10-12dbc8ff8f22
{
   “correlation_id”: "ec3f65f5-438d-4c30-a35e-bc8ca50de514",
   “auth_req_id”: “89bd81bd-6269-4a6c-9c55-73be670c4c1”,
   "access_token": "SlAV32hkKG",
   "token_type": "Bearer",
   "refresh_token": "8xLOxBtZp8",
   "expires_in": 3600,
   "id_token": "
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpc3MiOiJodHRwczovL3d3dy5tYy5vcGVyYX
Rvci5leGFtcGxlLmNvbSIsImlhdCI6MTUyOTczNzMzMywiZXhwIjoxNTI5NzM4NTM2LCJhdWQiO
iJzNkJoZFJrcXQzIiwic3ViIjoiOTM3OGQ3MzYtNjEzZS00ZjhjLWJlMDUtMGY1MDE4Y2ViODkx
IiwiYXV0aF90aW1lIjoiMTUyOTczNzMzMyIsIm5vbmNlIjoiYTdkOGRhODQtYTkzNi00MWU3LWE
yMGItN2UyYmZhZTkzOTdjIiwiYXRfaGFzaCI6ImFkNzFmYjQxNjU1MzY2NzVkODBhOGJiZmU5NT
c2OTdlNGI5ZTE3YjM1ZDU1MDA0MDZhYmMyMTk3MDUzMzcxMDciLCJhbXIiOiJTSU1fT0siLCJhe
nAiOiJzNkJoZFJrcXQzIiwiaGFzaGVkX2xvZ2luX2hpbnQiOiIyMDI0MGUzMjZjZTNhYTAxM2Iw
MGQzMDMyZThjMzc4N2Q1MjBmODdmZjFlOTNhMmQxYzdjMDQ0NzdmYTQ0YzliIiwicmVjaXBpZW5
0IjoiaHR0cHM6Ly93d3cuc2VydmljZXByb3ZpZGVyLmV4YW1wbGUuY29tL25vdGlmaWNhdGlvbl
9lbmRwb2ludCIsImRpc3BsYXllZF9kYXRhIjoic3BfY2xpZW50X25hbWUtSSBhbSBiaW5kaW5nI
G1lc3NhZ2UtY2FuIHlvdSBhdXRoZW50aWNhdGUgZm9yIG15IFNQIn0.nU7WlP1CIZTPzt_tsCRn
c6Q7lwnmej9heLU0m3hNwzWzbByNG3Cr19b3AGVxgQzTIjU485E71ur9A3pP4O8xLG7EKTjjuZF
jJjcSekATfeHsbeiKRk9a5phsk_ZXqH1tCaZ_ZkeIUkLI6Hn3z-
30QsmmHe7XXvDZ402isJih1oU"
}
```

The following is the decoded ID Token in plain text format.

```
Headers
{
  "typ": "JWT",
  "alg": "RS256"
}

Payload

{
  "iss": "https://www.mc.operator.example.com",
  "iat": 1529737333,
  "exp": 1529738536,
  "aud": "s6BhdRkqt3",
  "sub": "9378d736-613e-4f8c-be05-0f5018ceb891",
  "auth_time": "1529737333",
  "nonce": "a7d8da84-a936-41e7-a20b-7e2bfae9397c",
```

```
  "at_hash":
"ad71fb4165536675d80a8bbfe957697e4b9e17b35d5500406abc219705337107",
  "amr": "SIM_OK",
  "azp": "s6BhdRkqt3",
  "hashed_login_hint":
"20240e326ce3aa013b00d3032e8c3787d520f87ff1e93a2d1c7c04477fa44c9b",
  "recipient":
"https://www.serviceprovider.example.com/notification_endpoint",
  "displayed_data": "sp_client_name-I am binding message-can you
authenticate for my SP"
}
```

The following is an example of an unsuccessful Authorization sent as a notification to the SP:

```
POST /token_notif HTTP/1.1
Content -Type: application/json
Host: spserver.example.com
Authorization: bearer Bedsfe2134sd
{
   "correlation_id": "42da5b19-457a-4d30-a5c4-038c62dccbb0",
   "auth_req_id": "1234455",
   "error": "access_denied",
   "error_description": "authentication failure example"
}
```

B.4 Notification Acknowledgement

The following is an example of a successful acknowledgement:

```
HTTP/1.1 204 No Content
Content-Length:0
```

The following is an example of an error acknowledgement returned to the IDGW:

```
HTTP/1.1 400 Bad Request
  Content-Type: application/json
  Cache-Control: no-store
  Pragma: no-cache

  {
   "correlation_id: "42da5b19-457a-4d30-a5c4-038c62dccbb0",
   "error": "invalid_request",
   "error_description": "invalid tokens"
  }
```

B.5 Signed Request Object Authorization Request (Using Polling)

The following is an example of a signed Request Object for polling after base64url encoding and signing (with line wraps within values for display purposes only. The client authentication mechanism used is basic http header client authentication.

```
POST /authorize HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Host: operatoridgw.example.com
Authorization: Basic <Berdaz1234dzsdf>
   response_type=mc_si_polling
   &client_id=s6BhdRkqt3
   &scope=openid%20Fmc_authn
&request=
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJzNkJoZFJrcXQzIiwiaWF0IjoxNT
I5NzEyMDY2LCJleHAiOjE1NDQ0NTcwMDIsImF1ZCI6Imh0dHBzOi8vbWMub3BlcmF0b3IuZXhhb
XBsZS5jb20iLCJzdWIiOiJzNkJoZFJrcXQzIiwicmVzcG9uc2VfdHlwZSI6Im1jX3NpX3BvbGxp
bmciLCJjbGllbnRfaWQiOiJzNkJoZFJrcXQzIiwic2NvcGUiOiJvcGVuaWQgbWNfYXV0aG4iLCJ
2ZXJzaW9uIjoibWNfc2lfcjJfdjEuMobile
ConnectIsIm5vbmNlIjoiNWZmMDZmZDctY2MxMy00MDI2LTg2NGQtZjlmZTYzMWFiZDYxIiwicH
JvbXB0IjoibG9naW4iLCJtYXhfYWdlIjoiODY0MDAiLCJ1aV9sb2NhbGVzIjoiZW4tdXMiLCJjb
GFpbXNfbG9jYWxlcyI6ImVuLXVzIiwibG9naW5faGludCI6IjQ0NzQxMTE4ODI1OCIsImFjcl92
YWx1ZXMiOiIyIiwiYmluZGluZ19tZXNzYWdlIjoiSW50ZXJsb2NrIG1zZyIsImNsaWVudF9uYW1
lIjoic3BfY2xpZW50X25hbWUiLCJjb250ZXh0IjoiY2FuIHlvdSBhdXRoZW50aWNhdGUgZm9yIG
15IFNQIiwiY29ycmVsYXRpb25faWQiOiJmOTU2M2QyMi00YTZjLTRkYmEtYWUzZC0zMDI4OWY2Z
mQ0YWYiLCJqdGkiOiI2NjBmOWI0NS1jYjgxLTQyYzQtYTZhMobile
Connect1iZTI5Y2U5NGEzZDkifQ.bDVlPI7aVYiPTfhjXymNE9LQHXIDevnTG-01EDWGIBQ


}
```

The following is the example data before encoding and signing.

```
Headers :
{
  "typ": "JWT",
  "alg": "RS256"
}

Payload Data :
{
  "iss": "s6BhdRkqt3",
  "iat": 1529712066,
  "exp": 1529713261,
  "aud": "https://mc.operator.example.com",
  "sub": "s6BhdRkqt3",
  "response_type": "mc_si_polling",
  "client_id": "s6BhdRkqt3",
  "scope": "openid mc_authn",
  "version": "mc_si_r2_v1.0",
  "nonce": "5ff06fd7-cc13-4026-864d-f9fe631abd61",
"prompt": "login",
```

```
"max_age": "86400",
  "ui_locales": "en-us",
  "claims_locales": "en-us",
  "login_hint": "447411188258",
  "acr_values": "2",
  "binding_message": "Interlock msg"
  "client_name": "sp_client_name",
  "context": "can you authenticate for my SP",
  "correlation_id": "f9563d22-4a6c-4dba-ae3d-30289f6fd4af"
}
```

B.6 Authorization Response (Using Polling)

The following is an example for a authorization response where the request has been successfully validated:

```
HTTP/1.1 200 OK
  Content-Type: application/json
  Cache-Control: no-store
  Pragma: no-cache
  {
   "auth_req_id": "89bd81bd-6269-4a6c-9c55-73be670c4c17",
   "correlation_id": "f9563d22-4a6c-4dba-ae3d-30289f6fd4af",
   "expires_in": 3600,
   "interval" : 25
  }
```

The following is an example for authorization response where an error has occurred:

```
HTTP/1.1 400 Bad Request
  Content-Type: application/json
  Cache-Control: no-store
  Pragma: no-cache

  {
   "correlation_id": "f9563d22-4a6c-4dba-ae3d-30289f6fd4af",
   "error": "invalid_request",
   "error_description": "acr values are missing"
  }
```

B.7 Polling Request (Token Request) with private_key_jwt

The following is an example for polling request using basic http authorization flow:

```
POST /token_notif HTTP/1.1
Content -Type: application/x-www-form-urlencoded
Host: servce-provider.example.com
```

```
  client_id=s6BhdRkqt3&
  grant_type=urn%3Aopenid%3Aparams%3Amc%3Agrant+type%3Aserver_initiated&
  auth_req_id=89bd81bd-6269-4a6c-9c55-73be670c4c17&
  client_assertion_type=urn%3Aietf%3Aparams%3Aoauth%3Aclient-assertion-
type%3Ajwt-bearer&
  client_assertion=
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpc3MiOiJzNkJoZFJrcXQzIiwiaWF0IjoxNT
I5NzQxMjg1LCJleHAiOjE1Mjk3NDI0ODYsImF1ZCI6Imh0dHBzOi8vbWMub3BlcmF0b3IuZXhhb
XBsZS5jb20iLCJzdWIiOiJzNkJoZFJrcXQzIiwianRpIjoiOTVjY2YzODUtOTIxYy00ZTc4LTk3
ZTEtYWFjOWE5MDEyMmZiIn0.m8S9SrcwMXJQFYYH5VzcXUF_zYJUUPkNKMVyxMrcCwY2Q0lY-
oQUETan1pZTHxSPtF1QUmAqdYvVuAEV8MFNAzZnOSv9B7zbPXTVnh-
DIS_RhebBi6vjHqwxpvjKS2WksOwHNMo5K1TzBLdtFTyUpX3vb11n4fzIBULIc07pTz8
}
```

The following is the plain text data before encoding and signed.

```
Headers :

{
  "typ": "JWT",
  "alg": "RS256"
}

PAYLOAD

{
  "iss": "s6BhdRkqt3",
  "iat": 1529741285,
  "exp": 1529742486,
  "aud": "https://mc.operator.example.com",
  "sub": "s6BhdRkqt3",
  "jti": "95ccf385-921c-4e78-97e1-aac9a90122fb"
}
```

B.8 IDGW Polling Response (Token Response)

The following is an example of polling response.

```
HTTP/1.1 200 OK
  Content-Type: application/json
  Cache-Control: no-store
  Pragma: no-cache
  {
   "correlation_id": "ec3f65f5-438d-4c30-a35e-bc8ca50de514",
   "access_token": "SlAV32hkKG",
   "token_type": "Bearer",
"refresh_token": "8xLOxBtZp8"
```

```
"expires_in": 3600,
"id_token":"eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJpc3MiOiJodHRwczovL3d3dy
5tYy5vcGVyYXRvci5leGFtcGxlLmNvbSIsImlhdCI6MTUyOTczNzMzMywiZXhwIjoxNTI5NzM4N
TM2LCJhdWQiOiJzNkJoZFJrcXQzIiwic3ViIjoiOTM3OGQ3MzYtNjEzZS00ZjhjLWJlMDUtMGY1
MDE4Y2ViODkxIiwiYXV0aF90aW1lIjoiMTUyOTczNzMzMyIsIm5vbmNlIjoiYTdkOGRhODQtYTk
zNi00MWU3LWEyMGItN2UyYmZhZTkzOTdjIiwiYXRfaGFzaCI6ImFkNzFmYjQxNjU1MzY2NzVkOD
BhOGJiZmU5NTc2OTdlNGI5ZTE3YjM1ZDU1MDA0MDZhYmMyMTk3MDUzMzcxMDciLCJhbXIiOiJTS
U1fT0siLCJhenAiOiJzNkJoZFJrcXQzIiwiaGFzaGVkX2xvZ2luX2hpbnQiOiIyMDI0MGUzMjZj
ZTNhYTAxM2IwMGQzMDMyZThjMzc4N2Q1MjBmODdmZjFlOTNhMmQxYzdjMDQ0NzdmYTQ0YzliIiw
iZGlzcGxheWVkX2RhdGEiOiJzcF9jbGllbnRfbmFtZS1JIGFtIGJpbmRpbmcgbWVzc2FnZS1jYW
4geW91IGF1dGhlbnRpY2F0ZSBmb3IgbXkgU1AifQ.lmaFgpsIZQMgYyvOYvsknhHLgqVxrRp4g0
XWQ_cSlDkj2wCXoOncau7OTYwux7jmfKrDLL9RvgYD-
ndtrZzv_cnKHgbjYGYKVF__5OywuJId1j6QNk0mN8Chq8ioS2BRVFjRJ89jlsnW_Nq5a8jxfEDy
XtuyfvYRF9yVC5aqimE"
}
```


The following is the example of ID token before encoding and signing


```
Headers

{
  "typ": "JWT",
  "alg": "RS256"
}

PAYLOAD :

{
  "iss": "https://www.mc.operator.example.com",
  "iat": 1529737333,
  "exp": 1529738536,
  "aud": "s6BhdRkqt3",
  "sub": "9378d736-613e-4f8c-be05-0f5018ceb891",
  "auth_time": "1529737333",
  "nonce": "a7d8da84-a936-41e7-a20b-7e2bfae9397c",
  "at_hash":
"ad71fb4165536675d80a8bbfe957697e4b9e17b35d5500406abc219705337107",
  "amr": "SIM_OK",
  "azp": "s6BhdRkqt3",
  "hashed_login_hint":
"20240e326ce3aa013b00d3032e8c3787d520f87ff1e93a2d1c7c04477fa44c9b",
  "displayed_data": "sp_client_name-I am binding message-can you
authenticate for my SP"
}
```


The following is an example for a token response using Polling where an error has occurred:


```
HTTP/1.1 400 Bad Request
  Content-Type: application/json
```

```
  Cache-Control: no-store
  Pragma: no-cache

  {
   "correlation_id": "f9563d22-4a6c-4dba-ae3d-30289f6fd4af",
   "error": "unknown_auth_req_id",
   "error_description": "auth_req_id is not recognised"
  }
```

B.9 Sector Identifier URI example

The following is a non-normative example for sector identifier URI contents. The Sector Identifier URI is a JSON file containing a single array of notification Uris or JWKS URI for SI mode.

```
GET /sector_identifier_uri_file.json HTTP/1.1
Accept: application/json
Host: sp.example.com

HTTP/1.1 200 OK
Content-Type: application/json
Cache-Control: no-store
Pragma: no-cache
[ "https://sp.example.com/notification_uri1"
  "https://sp.example.com/notification_uri2" ,
  "https://sp.example.com/jwks_uri1",
  "https://sp.other_company.com/jwks_uri2"
]
```

Annex C Document Management

C.1 Document History

Version	Date	Brief Description of Change	Approval Authority	Editor / Company
1.0	14/08/2017	TG approved document	TG	Venkatasivakumar Boyalakuntla / GSMA
2.0	03/06/2019	Merged DQRT comments and David's inputs. A Major update document with Polling mechanism. In this document two asynchronous mechanisms for server initiated mode. A) Notification Mode B) Polling mechanism.	TG	Venkatasivakumar Boyalakuntla / GSMA

C.2 Other Information

Type	Description
Document Owner	IDG
Editor/Company	Yolanda Sanz/GSMA

It is our intention to provide a quality product for your use. If you find any errors or omissions, please contact us with your comments. You MAY notify us at prd@gsma.com.

Your comments or suggestions & questions are always welcome.