Meri Pehchaan API Specification

Version 1.0

June2022

A Digital India Initiative
National e-Governance Division.
Ministry of Electronics and Information Technology.

Revision History

Version	Date	Comments
0.1	16/06/2022	Released draft version.

Table of Contents

Revision History..1
Introduction..2
Authorization APIs (For Server Side Web Applications)..2
Get Authorization Code..2
Get Access Token..4
Get Access Token(openid connect protocol)...4

Authorized Partner API Specification

Introduction

Meri Pehchaanis a key initiative under Digital India program. It aims at eliminating the use of multiple usernames for each government application.It is a single sign-on platform. It will enable seamless use of multiple government applications.It is an extensive collaboration of the three mainstream SSO platforms Jan Parichay, e-Pramaan, and Meri Pehchaan.Meri Pehchaan enables standardized registration means users don't need to provide different information for different services. It uses two-factor authentication that provides a high level of security to users' data.

This document provides technical detailsof Meri Pehchaanintegration with applications of trusted partners. This document assumes that the reader is aware of NSSO application functionality.

Meri PehchaanAuthorization APIs (For Server Side Web Applications)

To access files in the user's Meri Pehchaan account from your application, you must first obtain the user's authorization. Meri Pehchaan APIs use the OAuth 2.0 protocol for authorization. Meri Pehchaan supports common OAuth 2.0 scenarios such as those for web server, mobile applications and limited input devices such as printers and scanners.Meri Pehchaan also supports the Proof Key for Code Exchange (PKCE) protocol for the higher security of mobile application clients.For more information on OAuth 2.0 please refer to Internet Engineering Task Force's (IETF) documentation on The OAuth 2.0 Authorization Framework(https://tools.ietf.org/html/rfc6749), Proof Key for Code Exchange by OAuth Public Clients (https://tools.ietf.org/html/rfc7636) and OAuth 2.0 for Native Apps(https://tools.ietf.org/html/rfc8252).

Get Authorization Code

A call to this API starts the authorization flow using the OAuth 2.0 protocol. This isn't an API call—it's a Meri Pehchaan web page that lets the user sign in to Meri Pehchaan and authorize your application to access the user's data. After the user decides whether or not to authorize your app, they will be redirected to the redirect link provided by your application.

URL STRUCTURE

```
Production Environment:
https://*.meripehchaan.gov.in/public/oauth2/1/authorize
```

HTTP METHOD GET

PARAMETERS

- **response_type** (*required*) Provide the grant type requested, either token or code
- **client_id** (*required*) Provide the app id/client id that was created during the application registration process.

- **redirect_uri** (*required*) The URI to redirect the user after authorization has completed. This must be the exact URI registered in the Meri Pehchaan Partner Portal. A redirect URI is required for the token flow, but optional for the code flow.
- **state** (*required*) This is your application specific data that will be passed back to your application through *redirect_uri*.
- **code_challenge**(*optional but recommended for server based client applications, required for mobile client applications*) A unique random string called code verifier (*code_verifier*) is created by the client application for every authorization request. A *code_verifier* is a high-entropy cryptographic random string created using the unreserved characters [A-Z] / [a-z] / [0-9] / "-" / "." / "_" / "~", with a minimum length of 43 characters and a maximum length of 128 characters. The *code_verifier* should have enough entropy to make it impractical to guess the value.
  The code_challenge sent as this parameter is the Base64URL (with no padding) encoded SHA256 hash of the code verifier.

  ```
  code_challenge = base64_url_encode_without_padding(sha256(code_verifier))
  ```

  Here is the pseudo code to implement a base64url-encoding function without padding, based upon the standard base64-encoding function that uses padding:

  ```
  string base64_url_encode_without_padding(stringarg)
  {
  strings= base64encode(arg); //Regular base64 encoder with padding
  s=s.replace('=',''); //Remove any trailing '='
  s=s.replace('+', '-'); //Replace '+' with '-'
  s=s.replace('/', '_'); //Replace '/' with '_'
  return s;
  }
  ```

- **code_challenge_method** (*required if code_challenge parameter is passed*)Specifies what method was used to encode a *code_verifier* to generate *code_challenge* parameter above. This parameter must be used with the *code_challenge* parameter. The only supported values for this parameter is *S256*.
- **Scope**(*optional*) If this parameter is provided its value will always be *openid*. This parameter indicates that client will receive "*id_token*" in response of "Get Access Token" API.
- **acr**(*optional*) If this parameter is provided its value will always be either *pan*,*aadhaar* or*driving_licence*. This parameter indicates that the user have to verify their authentic content recognition from Meri Pehchaan service, if user successfully verified acr then client will receive the verified acr data inside"*id_token*" in the response of "Get Access Token" API .

RETURNS

Since /oauth2/1/authorizeis a website, there is no direct return value. However, once a user successfully authorizes your app, the Meri Pehchaan application will forward the flow to your redirect URI. The type of response varies based on the response_type.

If theresponse_type parameter is passed as code then the following parameters are returned in the query string:

- **code** The authorization code, which can be used to attain a bearer token by calling the Get Access Token API.
- **state** This is application specific data, if any, originally passed to /oauth2/1/authorize

If the response_type parameter is passed as token then the following parameters are returned in the query string:

- **access_token** Theaccess token that can be used to call the Meri Pehchaan APIs.
- **expires_in** The duration in seconds for which the access token is valid.
- **token_type** The type of token which will always be Bearer.
- **scope** Scope of the token.

ERRORS

If the request fails due to a missing, invalid, or mismatching redirection URI, or if the client identifier is missing or invalid, the flow will result in error response.

If the resource owner denies the access request or if the request fails for reasons other than a missing or invalid redirection URI, the following parameters will be included in the redirect URI:

- **error** An error code as per the OAuth 2.0 spec.
- **error_description** A user-friendly description of the error that occurred.
- **state** The state content originally passed to authorization flow if any.

Get Access Token

This endpoint only applies to apps using the authorization code flow. An app calls this endpoint to acquire a bearer token once the user has authorized the app.Calls to /oauth2/1/token need to be authenticated using the app's key and secret. These can either be passed as application/x-www-form-urlencoded POST parameters (see parameters below) or via HTTP basic authentication. If basic authentication is used, the app key should be provided as the username, and the app secret should be provided as the password.

Get Access Token(openid connect protocol)

This endpoint only applies to apps using the authorization code flow. An app calls this endpoint to acquire a bearer token once the user has authorized the app.Calls to /oauth2/2/token need to be authenticated using the app's key and secret. These can either be passed as application/x-www-form-urlencoded POST parameters (see parameters below) or via HTTP basic authentication. If basic authentication is used, the app key should be provided as the username, and the app secret should be provided as the password.

URL STRUCTURE

```
Production Environment:
```

```
https://*.meripehchaan.gov.in/public/oauth2/2/token
```

HTTP METHOD **POST**

HTTP REQUEST HEADER

- *Content-Type: application/x-www-form-urlencoded*

PARAMETERS

- **code**(*required*) The code acquired by directing users to /oauth2/1/authorize?response_type=code.
- **grant_type**(*required*) The grant type, which must be authorization_code.
- **client_id** (*required*) If credentials are passed in POST parameters, this parameter should be present and should be the app key/client id.
- **client_secret** (*required*) If credentials are passed in POST parameters, this parameter should be present and should be the app's secret.
- **redirect_uri**(*required*) Only used to validate that it matches the original /oauth2/authorize, not used to redirect again.
- **code_verifier**(*required if code_challenge parameter is passed in authorization request*) The code_verifier created during authorization request. This parameter is mandatory for mobile client applications.

RETURNS

A JSON string containing following fields will be returned in response:

- **access_token** The access token that can be used to call the Meri Pehchaan APIs.
- **expires_in** The duration in seconds for which the access token is valid.
- **token_type** The type of token which will always be Bearer.
- **scope** Scope of the token.
- **id_token**This contain claims that carry information about the user. This response is JWT.

```
Sample Response:
{
    "access_token": "bc125c212a4b03a9a188a858be5a163f379e878a",
    "expires_in": 3600,
    "token_type": "Bearer",
    "scope":"openid",
    "id_token":
"eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsImtpZCI6IjVmOWRkNjI3ZDRlNDVlNmM5OGV
iZGJkOWU5YmIxMzI0In0.eyJpc3MiOiJodHRwczpcL1wvYXBpLmRpZ2l0YWxsb2NrZXIuZ29
2LmluIiwic3ViIjoiYWppdC5kbCIsImF1ZCI6IkFCQ0RFRkdIIiwiaWF0IjoxNjU0MTQ1OTc
1LCJleHAiOjE2NTQyMzIzNzUsImF1dGhfdGltZSI6MTY1NDE0NTk3NSwiZ2l2ZW5fbmFtZSI
6IkFqaXQgS3VtYXIiLCJwcmVmZXJyZWRfdXNlcm5hbWUiOiJhaml0LmRsIiwiZW1haWwiOiJ
haml0Lmt1bWFyQGRpZ2lsb2NrZXIuZ2l2LmluIiwiYmlydGhkYXRlIjoiMDFcLzAxXC8xOTk
wIiwicGhvbmVfbnVtYmVyIjoiOTg3NjU0MzIxMCIsImp0aSI6IjNmZmQ5OGQzLTAzMjEtNDQ
xYS1iMmVhLTE4YTY2ZDU1YWEwYiIsInVzZXJfc3NvX2lkIjoiREwtOTNmMzM5MGMtNmQ5Mi0
xMWU5LWE4NWUtOTQ1N2E1NjQ1MDY5IiwicGFuX251bWJlciI6IkFCQ0RLMTIzMkciLCJkcml
2aW5nX2xpY2VuY2UiOiJETDAxMjAyMjAwMDAwMDAxIiwibWFza2VkX2FhZGhhYXIiOiJ4eHh
4eHh4MTIzNCJ9.ZNfwZpf4ws7btEHxpRV9sOTRDR1g4CpnQEJi3VXLbdYrvDEwLyGpnQ8uQ9
g1cq_mTmv11K2scaRd16Cg9AKBl51FVuXW4_WJC7CmIOz4Ys9YJf_m4NU3v4mV-
aDDOjLV6RHX9G6uHtS9Llemek-8yIE4rjcjUabq0vlC5JkclAcYcRY7pTGm0BKRQU4O-
SktKFcR_X5b7dnwU08qJkpeCsL9B72gbCAdxLK8ZQp6npjX0BZU-
8ocieRaARS_5MjpAJVkNAwgUQ0rv_nwh15jG9P9bjGmVVn6djlBZ_PWJbLcxtfJEUFSeMupv
6QCg3lbGgKGOVPdS9CEKeP2t1G8-w"
}
```

ERRORS

The authorization server responds with an HTTP status code as follows:

Code	Description
400	Bad request.
401	If the access token is expired or has been revoked by Meri Pehchaan user.