



Digital Seva Connect

2016

The document can be used as a guide for Merchants ready to integrate with the CSC platform and should be used as a reference document for understanding the usage and process for integration of the Digital Seva Connect service for user authentication and authorization.

www.csc.gov.in

Copyright

Copyright © 2016 CSC e-Governance Services India Limited. All rights reserved.

The material contained in this guide is copyrighted and owned by CSC e-Governance Services India Limited together with any other intellectual property in such material. Except for personal and non-commercial use, no part of this guide may be copied, republished, performed in public, broadcast, uploaded, transmitted, distributed, modified or dealt with in any manner at all, without the prior written permission of CSC e-Governance Services India Limited, and then only in such a way that the source and intellectual property rights are acknowledged.

To the maximum extent permitted by law, CSC e-Governance Services India Limited shall not be liable to any person or organisation, in any manner whatsoever from the use, construction or interpretation of, or the reliance upon, all or any of the information or materials contained in this guide.

The information in these materials is subject to change without notice and CSC e-Governance Services India Limited assumes no responsibility for any errors.

CSC E-Governance Services India Limited.

Registered office: CSC E Governance Services India Limited

3rd Floor Electronics Niketan, Lodhi Road, New Delhi

Preface

CSC e-Governance Services India Limited is a Special Purpose Vehicle (CSC SPV) incorporated under the Companies Act, 1956 by the Department of Electronics and Information Technology (DeitY), Ministry of Electronics and Information Technology (MEiTY), Government of India to monitor the implementation of the Common Service Center Scheme.

CSC SPV is connecting local population with the Government departments, banks, and insurance companies and with various service providers in private sector using IT-Enabled network of citizen service points.

Digital Seva Connect is an identity solution that helps the common service centre users to sign in to the Service Access Provider (SAP) portals quickly and securely using their CSC login credentials and do business transactions for the services. This document is designed as a guide to enable its users to understand the procedure to connect their website to the Digital Seva Connect Portal. The primary user of the document would be the application developer of the merchant who is working or interested to work with CSC.

The document details out an introduction and steps to integrate to the Digital Seva Connect portal along with all other information that is required to understand the functionality of Digital Seva Connect.

Table of Contents

1. Introduction	5
1.1 About this guide	6
1.1.1 Objectives and target audience	6
1.1.2 Convention	6
1.1.3 Abbreviation	6
1.1.4 Definition	7
1.1.5 Who to contact for queries	8
2. Overview	8
2.1 Connect Features	8
2.2 CSC User, Merchant & Developer Benefits	9
2.3 Digital Seva Connect Experience	10
3. Digital Seva Connect	12
3.1 The OAuth 2.0 Roles	12
3.2 The OAuth Flow	13
3.3 Redirect URIs	17
3.4 Token expiration	17
4. Getting Started with Integration	18
4.1 Merchant On-Boarding	18
4.2 Connect Config	23
4.3 Procedure at a Glance	24
5. Integration Language Support	32
5.1 PHP	33
5.2 JAVA	36
5.3 DOT NET	39

1. Introduction

About CSC

Common Services Centers (CSCs) are the access points for delivery of various e-governance and business services to citizens in rural and remote areas of the country. It is a pan-India network catering to regional, geographic, linguistic and cultural diversity of the country, thus enabling the Government's mandate of a socially, financially and digitally inclusive society.

CSCs offer assisted access of e-services to citizens with a focus on enhancing governance, delivering essential government and public utility services, social welfare schemes, financial services, education and skill development courses, health and agriculture services and digital literacy, apart from a host of B2C services.

CSCs are more than service delivery points in rural India; they are positioned as change agents, promoting rural entrepreneurship and building rural capacities and livelihoods.

Digital Seva Connect

Digital Seva Connect is an identity solution that helps the common service centre users to sign in to the Service Access Provider (SAP) portals quickly and securely using their CSC login credentials and do business transactions for the services.

Digital Seva Connect uses the OAuth 2.0 protocol for authentication & authorization and supports OAuth integration scenarios, such as those for web server.



Note: Given the security implications of getting the implementation correct, we strongly encourage you to use OAuth 2.0 libraries when interacting with Digital Seva Connect's OAuth 2.0 endpoints. It is a best practice to use well-debugged code, and it will help you protect yourself and your users.

1.1 About this guide

1.1.1 Objectives and target audience

The objective of this guide is to provide the complete understanding of Digital Seva Connect to our audience. This guide provides details on how to third party applications can integrate CSC DSC at their web applications to enable user logins to CSC users viz., Village Level Entrepreneur.

The document is intended for user such as application developer of the merchant who are working or interested to work with CSC.

1.1.2 Convention

The table below lists some of the conventions used in this guide.

CONVENTION	DESCRIPTION
Reference	Indicates a reference to another section in this guide.

Table 1

1.1.3 Abbreviation

The table below lists some of the abbreviations used in this guide.

ABBREVIATION	DESCRIPTION
MEiTY	Ministry of Electronics & Information Technology
DEiTY	Department of Electronics & Information Technology
CSC	Common Service Center
SPV	Special Purpose Vehicle
VLE	Village Level Entrepreneur also refried to as CSC user
DSC	Digital Seva Connect

URI	Uniform Resource Identifier
URL	Uniform Resource Locator
AES	Advanced Encryption Standard
SOA	Service-Oriented Architecture
API	Application Programming Interface
CSRF	Cross-Site Request Forgery

Table 2

1.1.4 Definition

The table below describes few keywords used in this guide.

TERM	DEFINITION
Merchant	Businesses ready to sell their product or services online
Protocol	Protocol is the special set of rules that end points in a telecommunication connection use when they communicate. Protocols specify interactions between the communicating entities.
Access token	A token used to access protected resources.
Authorization code	An intermediary token generated when a user authorizes a client to access protected resources on their behalf. The client receives this token and exchanges it for an access token.
Grant	A grant is a method of acquiring an access token.
Scope	A permission.
JWT	A JSON Web Token is a method for representing claims securely between two parties as defined in RFC 7519.

Table 3

1.1.5 Who to contact for queries

Please contact our support department at below mention email and toll free number for all integration related queries.

Email Id	partners@csc.gov.in
Toll Free Number	1800 3000 3468

Table 4

2. Overview

Digital Seva Connect (formerly CSC Connect) is an identity solution that enables the CSC users to sign in to your web site quickly and securely using their CSC login credentials and do business transactions for your services. The service is available for all who wish to enable access to the large community of the CSC users. Connect is a fast and convenient way for our network of users to login to your site or app built across multiple platforms.

2.1 Connect Features

Real & Validated Identity

The CSC users are prior verified using the Aadhaar e-KYC service for their real identity which becomes their CSC profile.

Cross Platform

Connect is built using open standards and is available for the most common application platforms.

Works along your User authentication system

Connect would always complement your existing user account system by providing an option to login using the connect service.

User Control Data Share

Being Transparent & open gives more freedom and builds users trust and connect ensures that the same is achieved by providing its users control over the application permissions where user allows access they allow to share.

Streamlined experience

Streamline experience with just only one user ID to remember and to login for CSC users.

Convenient

Convenience with fewer details to be refilled with new logins

Security

Provide peace of mind as CSC would ensure end to end encryption of all data shared with merchant.

Easy access

Enabling DSC also provide easy access to our pay services where users can check out merchant products and services to interact with the wallet to enable online payment processing.

2.2 CSC User, Merchant & Developer Benefits

1. Open Standards based solution

Digital Seva Connect uses open standards and is based on the popular OAuth 2.0 protocol. Digital Seva Connect securely enables you trust that users are securely logged-in. On your side, your system must manage the logged-in and log-out sessions. In addition, it must properly manage any of the user information provided through the Connect service.

User attributes are shared only after the user has provided consent to share the attributes with the merchant.

2. Streamlined Sign In services

CSC users use their Connect credentials to securely sign in to your website/application. This reduces payment failure, and can increase overall sales. Also, customers won't need to create a new user account to shop and pay on your site.

3. Access to large network of Customers

CSC has an active network of more than 200,000 users and is increasing each day. You benefit with our large network of users which results in more business opportunities.

4. Automatic Updates to User Information

Changes to user data are dynamically reflected in the user attributes that are delivered post authorization.

2.3 Digital Seva Connect Experience

Connect offers a secure authentication service that lets CSC users sign in to your web site or application using their Connect credentials which is quick. CSC ID frees you from the need to implement a customer-management system. The Connect Auth API manages and verifies the CSC user account data for you which lets you focus on the important details of your business website, your core domain specific challenges.

Using Digital Seva Connect

Sign In with Connect a secure sign-on system that utilizes cutting-edge security standards, to manage the user log in process; and as a merchant website operator you don't have to store user data on your system.

User experience

When a CSC user visits your website & clicks the Sign In with Connect button, the user is redirected to the Digital Seva Connect where user can directly sign in by providing their access credentials. Upon successful authentication the user is redirected back to requesting client application.

User sign's in for the first time using Connect

He/she is redirected to an authorization grants page where the user has to provide a confirmation feedback to allow you to access user information, which would be used to as a resource by the client application to allow the user to access their web site.

User is revisiting the Connect and Merchant website

He/ she would be directly redirected to the client web application as the permission granted is already available with the connect server. The user resource data is transferred to the client web application. The user information is always updated as the identity information is being managed at the connect servers.

3. Digital Seva Connect

DSC is based on OAuth 2.0 protocol that lets your application request authorization details of user's CSC account without getting their password.

You'll need to register your app before getting started. A registered app is assigned a unique Client ID and Client Secret which will be used in the OAuth flow.



Note: Client Secret or obtained *access_token* must not be shared with anyone and must be kept in safe custody. If these are shared or compromised, it may lead to serious security issue. You are solely responsible for safety of your client secret and hence utmost care must be taken by you.

Sign in with Digital Seva Connect is the best way to log individuals into your application.

3.1 The OAuth 2.0 Roles

OAuth defines four roles:

ROLES	DESCRIPTION
Resource Owner	The user who authorizes an application to access their account. The application's access to the user's account is limited to the "scope" of the authorization granted (e.g. read or write access).
Resource Server	A server which sits in front of protected resources (for example "tweets", users' photos, or personal data) and is capable of accepting and responding to protected resource requests using access tokens.
Client	An application which accesses protected resources on behalf of the resource owner (such as a user). The client could be hosted on a server, desktop, mobile or other device.

Authorization Server

A server which issues access tokens after successfully authenticating a client and resource owner, and authorizing the request.

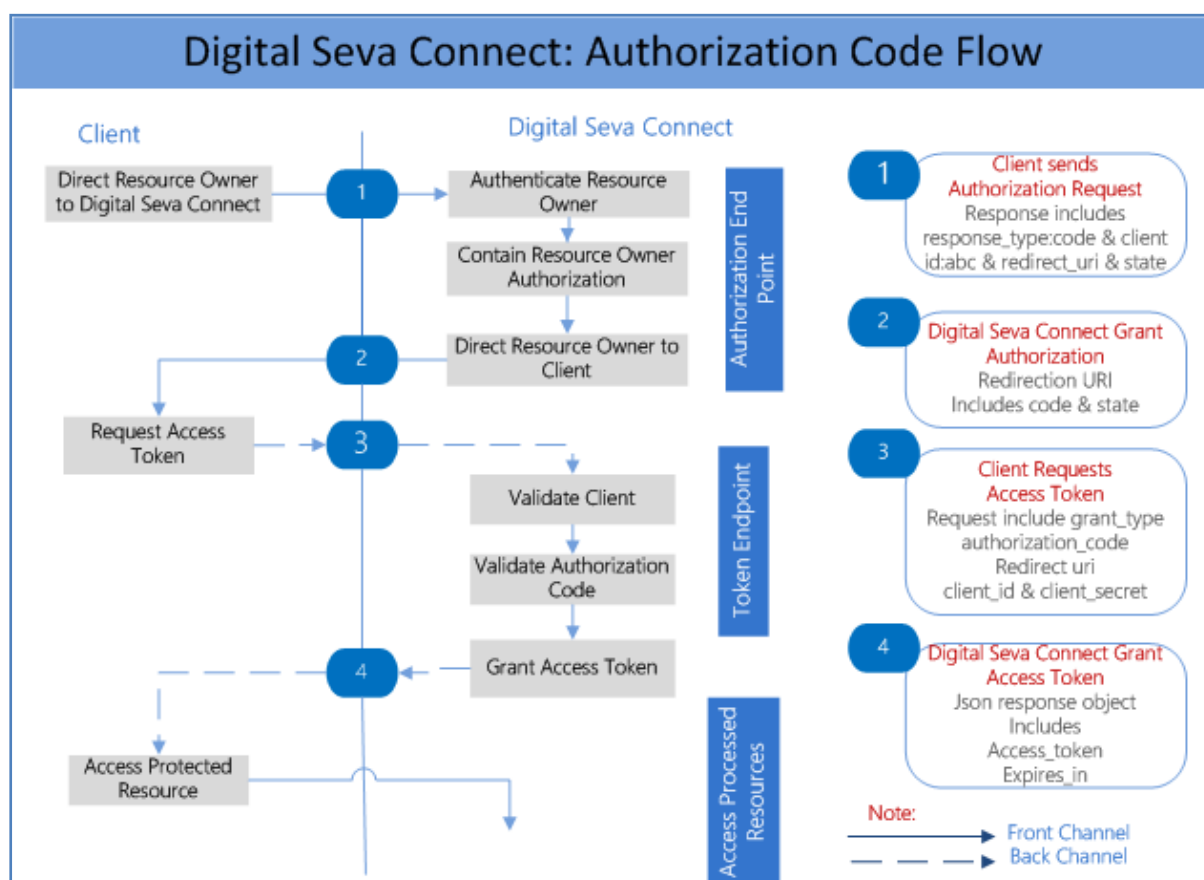
Table 5

Our authorization server generates access token for you that is readily accepted by our resource servers.

3.2 The OAuth Flow

Digital Seva Connect uses OAuth 2.0's authorization code grant flow to issue access tokens on behalf of users.

The abstract OAuth 2.0 flow describes the interaction between the four roles and includes the following steps:

**Figure 1**

The following flow diagram shows flow to authenticate a user to the Connect.

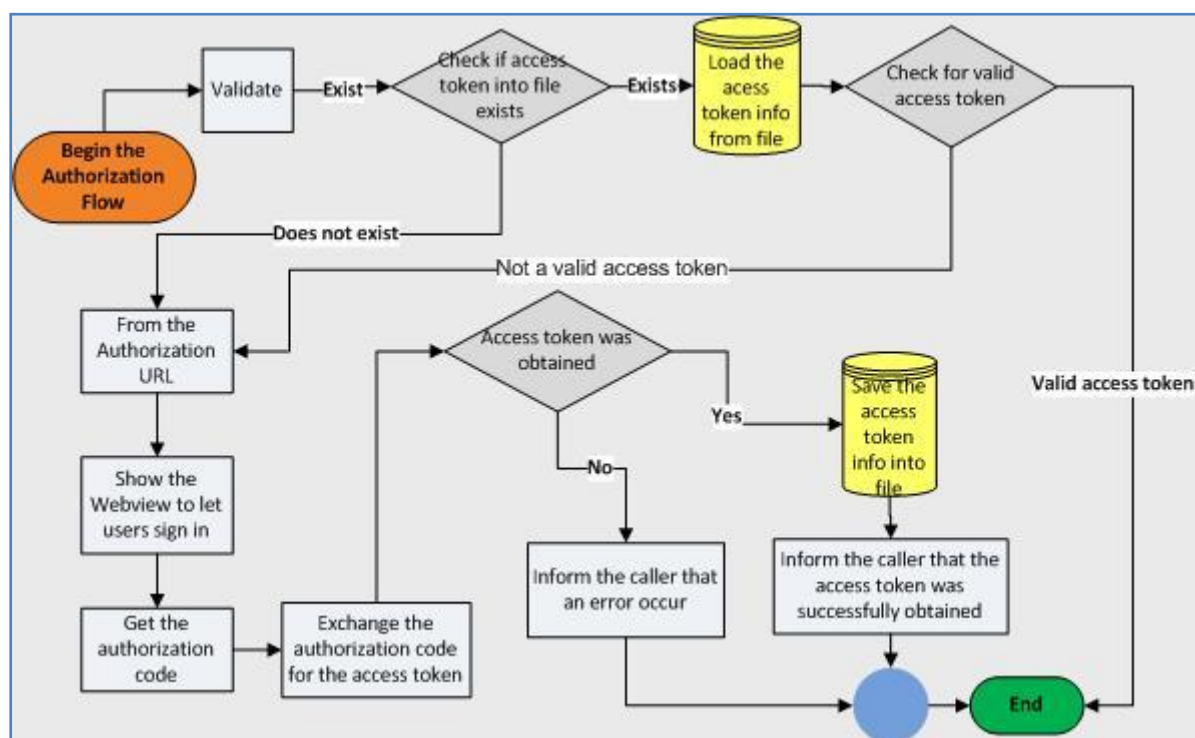


Figure 2

Step 1 - Authorization

The client requests authorization from the resource owner. The authorization request can be made directly to the resource owner, or preferably indirectly via the authorization server as an intermediary.

The client receives an authorization grant, which is a credential representing the resource owner's authorization, expressed using one of four grant types defined in this specification or using an extension grant type. The authorization grant type depends on the method used by the client to request authorization and the types supported by the authorization server.

The following values should be passed as GET parameters:

PARAMETER	DESCRIPTION
client_id	issued when you created your app
redirect_uri	URL to redirect back to
state	unique string to be passed back upon completion
response_type	response type is set (code)

Table 6

The **state** parameter should be used to avoid forgery attacks by passing in a value that's unique to the user you're authenticating and checking it when auth completes.

The **response_type**, authorization endpoint is used by the authorization code grant type and implicit grant type flows.

Step 2 - Token Issuing

The client can request an access token by authenticating with the authorization server and presenting the authorization grant.

The authorization server authenticates the client and validates the authorization grant, and if valid, issues an access token.

The client requests the protected resource from the resource server and authenticates by presenting the access token. The resource server validates the access token, and if valid, serves the request.



Note: Authorization codes may only be exchanged once and expire 30 seconds after issuance.

URL: Connect URLs should always be used as the one mentioned in the config file.

If all is well, exchange the authorization code for an access token using the **access_token** API method.

PARAMETER	DESCRIPTION
client_id	issued when you created your application
client_secret	issued when you created your application
code	a temporary authorization code
redirect_uri	must match the originally submitted URI

Table 7

You'll receive a JSON response containing an **access_token** (among other details):

```
{
  "access token": "xoxp-23984754863-2348975623103",
  "state": "generated-random-state-value"
}
```

You can then use this token to call API methods on behalf of the user.



Note: These access tokens are valid for 600 seconds.

Denied Requests

If the user (application) denies your request, Digital Seva Connect does not redirect back to your **redirect_uri** with an **error** parameter.

<https://connect.csccloud.in/account/errormsg>¹

Applications should handle this condition appropriately.

¹ URLs used are for example reference only. Actual URLs are part of the Config *file*.

3.3 Redirect URIs

The **redirect_uri** parameter is mandatory. Digital Seva Connect will redirect users to the callback URL configured in your app's settings. The redirect URL's host and port must exactly match the callback URL.

CALLBACK: `https://yourdomain/your-callback-url-context`

3.4 Token expiration

You should write your code to anticipate the possibility that a granted token might no longer work.

PARTICULAR	DURATION
Code	30 seconds
Token	300 seconds
Session of the Auth Server	10 minutes

Table 8

3.5 Error Codes

CODE	MESSAGE
201	Client Authorization Declined
202	Invalid Token Request
203	Invalid Authorization Code Request
204	User not Authorized
205	Unknown Error
206	Something Went Wrong
207	Invalid Request Method/Parameter
208	Malformed Request

Table 9

4. Getting Started with Integration

1.1 Merchant On-Boarding

Once the agreement is signed between CSC SPV and Merchant, the CSC Business Developer would request the CSC Support team to On-Board the respective Merchant.

As the CSC Support team adds the merchant in the system, Merchant will receive a welcome email followed by another email regarding Merchant centric portal link and its credentials.

Now, Merchant can use these credentials to get logged in to Merchant Center Portal and follow the following steps:

1. **Login to the merchant centre of bridge portal.**
2. **Generate Connect Config. File**
 - a. On left menu, click on “CSC Connect”, following screen will display:

Figure 3

The merchant would be required to fill the following details:

- **APP NAME-** Enter the required application name
- **CALL BACK URL-** Enter the call back URL
- **APP LOGO-** Upload your application logo here

URL Format

1. We strongly recommend using HTTPS whenever possible
2. URLs must be absolute (e.g. "https://example.com/auth/callback", not "/auth/callback")
3. URL arguments are ignored (i.e. https://example.com/?id=1 is the same as https://example.com/)
4. URLs cannot include #'s (i.e. "https://example.com/auth/callback#CSC Digital Seva Connect" is invalid)

Logo Format

Format: jpeg/png

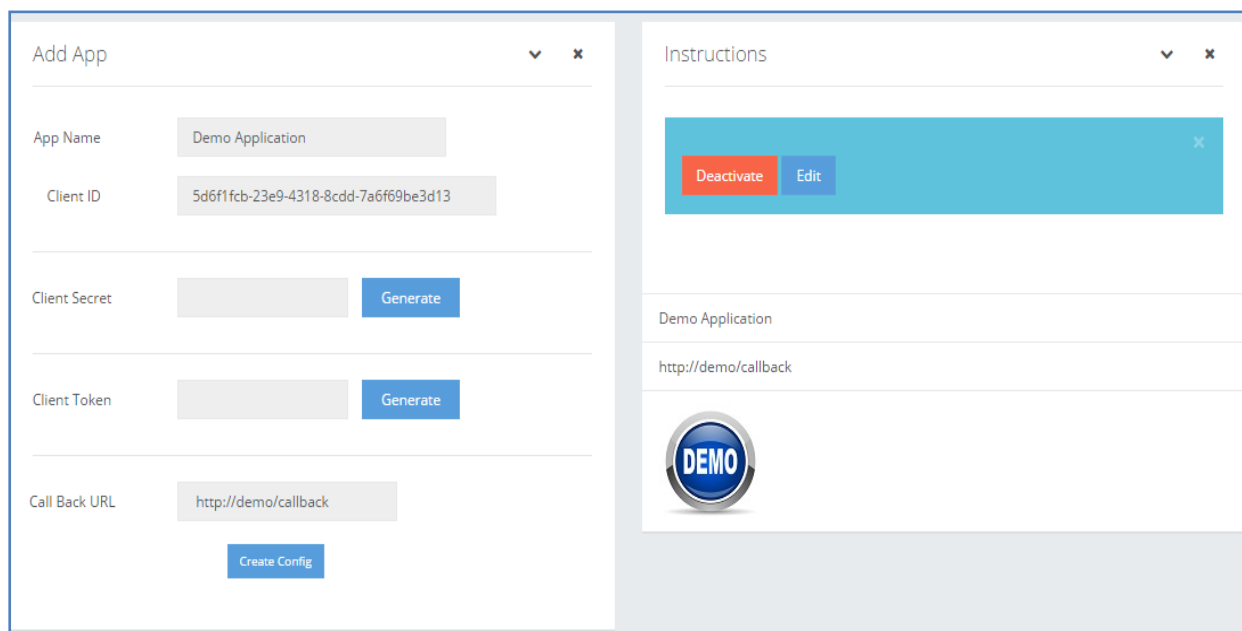
Size: Less than 25 kb

Click on **“Save”** button will add the application and displays following screen.

The screenshot shows two side-by-side panels. The left panel, titled 'Add App', has a green header bar with the text 'APP Name added successfully'. Below this, there are input fields for 'App Name' (containing 'Demo Application'), 'Client Secret', and 'Client Token'. A 'Generate Client ID' button is located below the 'App Name' field. At the bottom, there is a 'Call Back URL' field (containing 'http://demo/callback') and a 'Create Config' button. The right panel, titled 'Instructions', has a blue header bar. It contains a blue box with 'Deactivate' and 'Edit' buttons. Below this, it shows 'Demo Application' and 'http://demo/callback'. At the bottom of the right panel is a circular 'DEMO' logo.

Figure 4

- b. **Generate Client id-** Clicking on “**Generate Client ID**” button will generate the unique client id and the following screen will display.



The screenshot shows a web interface with two panels. The left panel, titled 'Add App', contains the following fields and buttons:

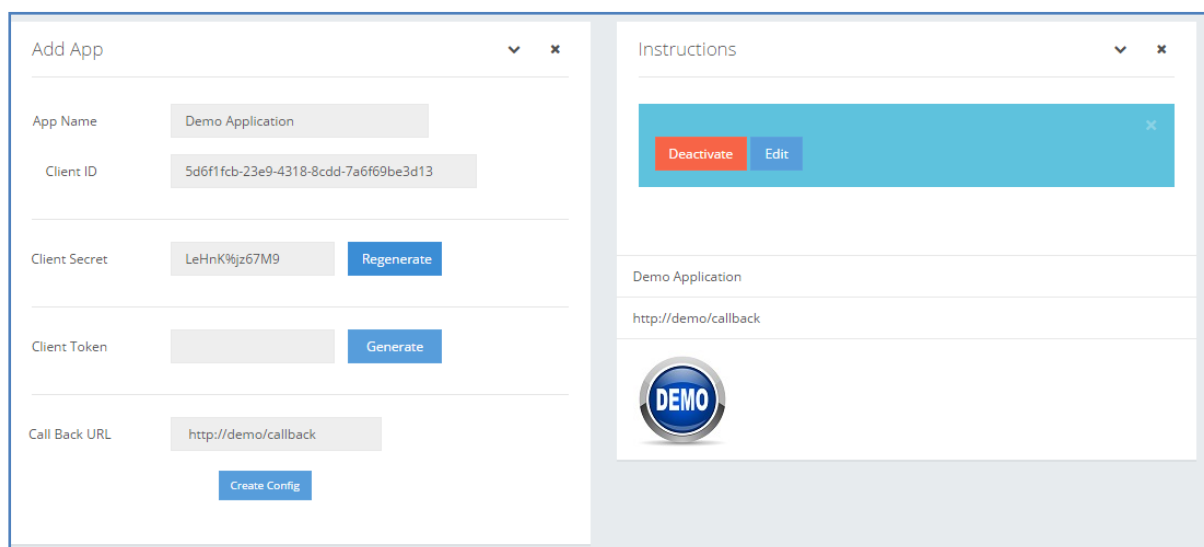
- App Name:** Demo Application
- Client ID:** 5d6f1fcb-23e9-4318-8cdd-7a6f69be3d13
- Client Secret:** [Empty field] with a **Generate** button.
- Client Token:** [Empty field] with a **Generate** button.
- Call Back URL:** http://demo/callback with a **Create Config** button.

The right panel, titled 'Instructions', contains a blue box with **Deactivate** and **Edit** buttons, and a section for the application details:

- Demo Application**
- http://demo/callback**
- A circular **DEMO** badge.

Figure 5

- c. **Generate Client Secret-** Click on “**Generate Client Secret**” button will generate the client secret and the following screen will display.



The screenshot shows the same web interface as Figure 5, but with the following changes:

- Client Secret:** LeHnK9jz67M9, with a **Regenerate** button.
- Client Token:** [Empty field] with a **Generate** button.
- Call Back URL:** http://demo/callback with a **Create Config** button.

The right panel remains the same as in Figure 5.

Figure 6

- d. **Generate Client Token-** Click on “**Generate Client Token**” button will generate the client token and following screen will display.

The screenshot shows two side-by-side panels. The left panel, titled 'Add App', contains the following fields and buttons:

- App Name: Demo Application
- Client ID: 5d6f1fcb-23e9-4318-8cdd-7a6f69be3d13
- Client Secret: LeHnK%jz67M9, with a 'Regenerate' button
- Client Token: WPeXrmukvWLj7qT8, with a 'Regenerate' button
- Call Back URL: http://demo/callback, with a 'Create Config' button

The right panel, titled 'Instructions', contains:

- A blue bar with a 'Deactivate' button (red) and an 'Edit' button (blue).
- A section for 'Demo Application' with the URL 'http://demo/callback'.
- A 'DEMO' badge.

Figure 7



Once the Config file is created, the option to edit or deactivate the application is enabled under the instructions shown in the navigation panel.

- e. **Create Config-** Click on “**Create Config**” button will generate the config file and following screen will display.

This screenshot is similar to Figure 7, but the 'Instructions' panel on the right now includes an additional button:

- The blue bar contains 'Deactivate' (red), 'Edit' (blue), and 'Download Config' (green) buttons.
- The 'Demo Application' section and 'DEMO' badge remain the same.

Figure 8

- f. Download the config file from the “**Download Config**” button.

Use the downloaded config file into your code.



DO NOT share configuration file with anyone!

1.2 Connect Config

Connect Config is the configuration parameters file required to implement the Connect. The configuration file can be generated from the Merchant Center portal; however a live Configuration would require an admin approval for changes to take place.

```
CLIENT_ID=648ac177-e78c-42d1-8e35-7501d7099d8c
CLIENT_APP_NAME=Application Name
CLIENT_SECRET=GznaAW2CcrYs
CLIENT_TOKEN=iLFiPaWXRbIlfF1I
CONNECT_SERVER_URI=https://connect.csccloud.in/account/
CLIENT_CALLBACK_URI=https://yourdomain/your-callback-url-context
VERSION=V1.0.7
```

The table below lists the Configuration parameters of Connect.

PARAMETER	DESCRIPTION
CLIENT_ID	Obtained at the time of Add Application
CLIENT_APP_NAME	Application Name
CLIENT_SECRET	Obtained at the time of Add Application
CLIENT_TOKEN	Obtained at the time of Add Application
CLIENT_SERVER_URI	Digital Seva Connect OAuth authorization URL
CLIENT_CALLBACK_URI	URL to which to return the user after authorization.
VERSION	Version of Connect platform

Table 10

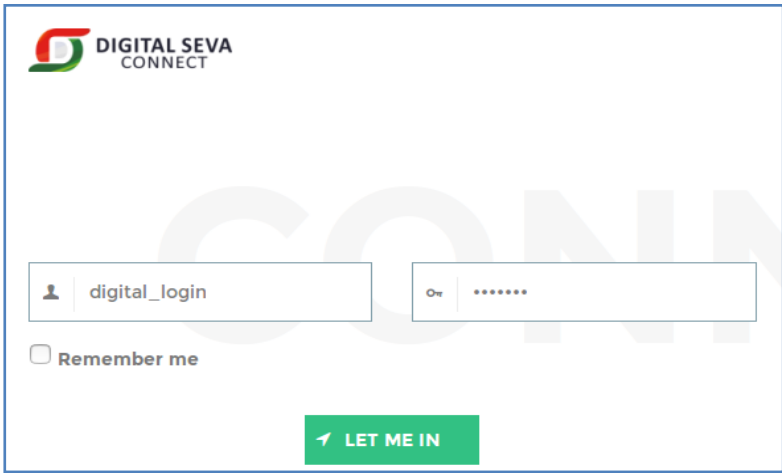
1.3 Procedure at a Glance

Once your application is configured as per specifications, it's time to request an authorization code. The authorization code is not the final token that you use to make calls to CSC Digital Seva Connect with. It is used in the next step of the OAuth 2.0 flow to exchange for an actual access token. This is an important step because it provides assurance directly from CSC Digital Seva Connect to the user that permission is being granted to the correct application, with the agreed-upon access to the member's CSC Digital Seva Connect profile.

Redirecting the User

To request an authorization code, you must direct the user's browser to CSC Digital Seva Connect's OAuth 2.0 authorization endpoint. Once the request is made, one of the following two situations will occur:

- If the user has not previously accepted your application's permission request, or the grant has expired or been manually revoked by the user, the browser will be redirected to CSC Digital Seva Connect's authorization screen. When the user completes the authorization process, the browser is redirected to CSC Digital Seva Connect's authentication dialog box.
- If there is a valid existing permission grant from the user, the authorization screen is by-passed and the user is immediately redirected to CSC Digital Seva Connect's authentication dialog box.



The image shows a login form for Digital Seva Connect. At the top left is the logo with the text 'DIGITAL SEVA CONNECT'. Below the logo, there are two input fields: one for the username labeled 'digital_login' and one for the password with masked characters. Below these fields is a checkbox labeled 'Remember me'. At the bottom center is a green button with a white arrow and the text 'LET ME IN'.

Figure 9

GET: <https://connect.csccloud.in/account/authorize>

Sample Response Redirect:

https://connect.csccloud.in/account/authorize?response_type=code&client_id=123456789&redirect_uri=https%3A%2F%2Fwww.example.com%2Fauth%2Fcallback&state=987654321

The table below lists the Configuration parameters of Connect.

PARAMETER	DESCRIPTION	REQUIRED
response_type	The values of this field should always be code	Yes
client_id	The “API Key” value generated when you registered your application	Yes
redirect_uri	The URI your users will be sent back to after authorization. This value must match with the defined OAuth 2.0 Redirect URLs in your application configuration. Example- https://yourdomain/your-callback-url-context	Yes
state	A unique string value of your choice that is hard to guess. Used to prevent CSRF.	Yes

Table 11

The User Experience- Authorize application prompt

Once redirected, the user will be presented with CSC Digital Seva Connect's authentication dialog box. This identifies your application as well as outlines the particular member permissions that your application has requested. If desired, the logo and application name can be changed in your application configuration.



Figure 10

Application is approved

By providing valid CSC Digital Seva Connect credentials and clicking on the "Allow Access" button, the user is approving your application's request to access their member data and interact with CSC Digital Seva Connect on their behalf. This approval instructs CSC Digital Seva Connect to redirect the user back to the callback URL that you defined in your **redirect_uri** parameter.

Response of Digital Seva Connect to the **redirect_uri** will have two important request parameters that need to be part by your application:

- code — The OAuth 2.0 authorization code.
- state — A value used to test for possible CSRF attacks.

The code is a value that you will exchange with CSC Digital Seva Connect for an actual OAuth 2.0 access token in the next step of the authentication process. For security reasons, the authorization code has a very short lifespan and must be used within

moments of receiving it - before it expires and you need to repeat all of the previous steps to request another.

Before you accept the authorization code, your application should ensure that the value returned in the state parameter matches the state value from your original authorization code request. This ensures that you are dealing with the real original user and not a malicious script that has somehow slipped into the middle of your authentication flow. If the state values do not match, you are likely the victim of a CSRF attack and you should throw an HTTP 401 error code in response.

Application is rejected

If the user chooses to cancel, or the request fails for any other reason, an error page is generated at Digital Seva Connect with the following additional parameters:

- error- A code indicating the type of error.
- error_description - A URL encoded textual description that summarizes error.

Exchange Authorization Code for an Access Token

The final step towards obtaining an Access Token is for your application to ask for one using the Authorization Code it just acquired. This is done by making the following "x-www-form-urlencoded" HTTP POST request:

POST: `https://connect.csccloud.in/account/token`

PARAMETER	DESCRIPTION	REQUIRED
grant_type	The value of this field should always be <code>authorization_code</code>	Yes
code	The authorization code you received at Merchant Center Portal	Yes
redirect_uri	The same 'redirect_uri' you have provided at Merchant Center Portal.	Yes
client_id	'API Key' value generated at Merchant Center Portal	Yes
client_secret	The 'Secret Key' value generated at Merchant Center Portal.	Yes

Table 12

Sample Call (Secure approach)

POST /account/token HTTP/1.1

Host: connect.csccloud.in

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

```
grant_type=authorization_code&code=987654321&redirect_uri=https://yourdomain/your-callback-url-context&client_id=123456789&client_secret=shhdonottell
```

Access Token Response

A successful Access Token request will return a JSON object containing the following fields:

- **access_token**- The access token for the user. This value must be kept secure, as per your agreement to the API Terms of Use.
- **expires_in**- The number of seconds remaining, from the time it was requested, before the token will expire. Currently, all access tokens are issued with 5 minutes lifespan.

Make authenticated requests

Once you've obtained an Access Token, you can start making authenticated API requests on behalf of the user. This is accomplished by including an "Authorization" header in your HTTP call to CSC Digital Seva Connect's API. Here is a sample HTTP request including the header value that includes the token:

Sample Call (Secure approach)

```
GET /account/resource HTTP/1.1
Host: connect.csccloud.in
Connection: Keep-Alive
Authorization: Bearer AQXdSP_W41_UPs5ioT_t8HESyODB4FqbkJ8LrV_5m
```

Invalid Tokens

If you make an API call using an invalid token, you will receive a "401 Unauthorized" response back from the server. A token could be invalid and in need of regeneration because:

- It has expired.
- The user has revoked the permission they initially granted to your application.
- A subsequent OAuth2 flow that generated a new access token. The previous token will be invalidated.

Since a predictable expiry time is not the only contributing factor to token invalidation, it is very important that you code your applications to properly handle an encounter with a 401 error by redirecting the user back to the start of the authorization workflow.

Response Parameter

PARAMETER	DESCRIPTION
csc_id	Unique 12 digit CSC ID E.g. 999999999999
email	Users registered/communication email id ex: test@gmail.com
user_type	Type of CSC user E.g. VLE, Merchant etc
state_code*	02 Character E.g. DL, JK etc
active_status	<i>Refer Table 14</i> Activate-1 Deactivate-0

Table 13

STATE_NAME	STATE_CODE
Jammu and Kashmir	JK
Himachal Pradesh	HP
Punjab	PB
Chandigarh	CH
Uttarakhand	UA
Haryana	HR
Delhi	DL
Rajasthan	RJ
Uttar Pradesh	UP
Bihar	BR
Sikkim	SK
Arunachal Pradesh	AR
Nagaland	NL
Manipur	MN
Mizoram	MZ
Tripura	TR
Meghalaya	ML
Assam	AS
West Bengal	WB
Jharkhand	JH
Orissa	OR
Chhattisgarh	CG
Madhya Pradesh	MP
Gujarat	GJ
Daman And Diu	DD
Dadra and Nagar Haveli	DN
Maharashtra	MH
Andhra Pradesh	AP
Karnataka	KA
Goa	GA
Lakshadweep	LD
Kerala	KL
Tamil Nadu	TN
Pondicherry	PY
Andaman and Nicobar	AD
Telangana	TS

Table 14

2. Integration Language Support

Connect is built on open standards and ensures seamless integration in any language or platform. While integration kits and sample merchant website applications have been built and are ready in **dot net, Java & PHP**, it is being worked out to put up similar examples in other languages.

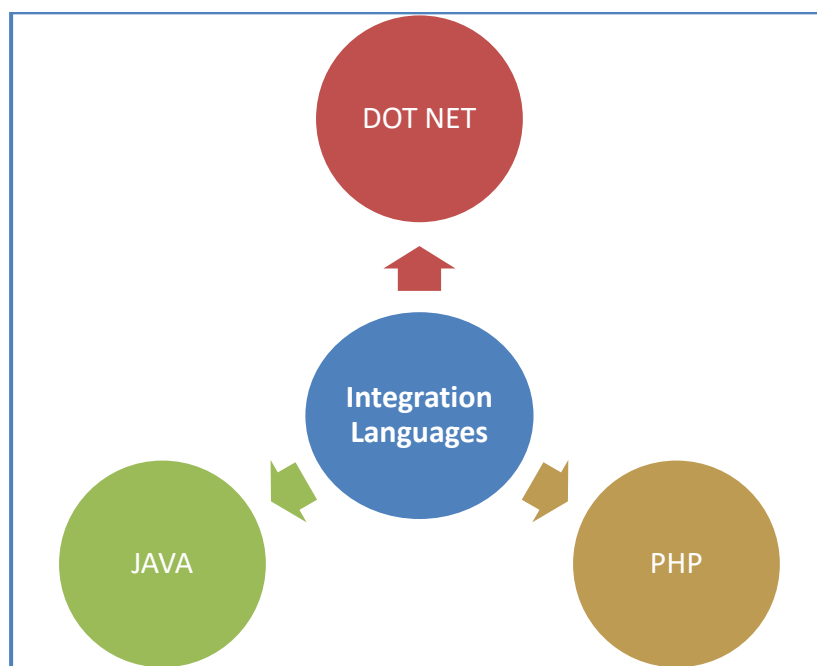


Figure 11

The Integration Kit comprises of below components-

- Config File
- Readme
- License



Download Link:

You can download Connect integration kit from the following link.

<https://github.com/csc-egov/Connect>

2.1 PHP

The illustrated code sample below provides the understanding of using the **php** DSC integration kit.

Step 1 Create a URL as in the sample **login.php**

```
<?php
session_start();
include("includes/Connect_client.php");
function login()
{
    $obj = new Connect_client();
    $client = $obj->get_client();
    $exparams = array(
        'state' => rand(10000, 99999)
    );
    $state_val = rand(10000, 99999);
    $_SESSION['state_val'] = $state_val;
    $exparams = array(
        'state' => $state_val
    );

    $auth_url = $client->getAuthenticationUrl(AUTHORIZATION_ENDPOINT,
    REDIRECT_URI, $exparams);
    $_SESSION[CSC_CURR_URL] = $_SERVER['REQUEST_URI'];
    header('Location: ' . $auth_url);
    die('Redirect');
}

login();
```

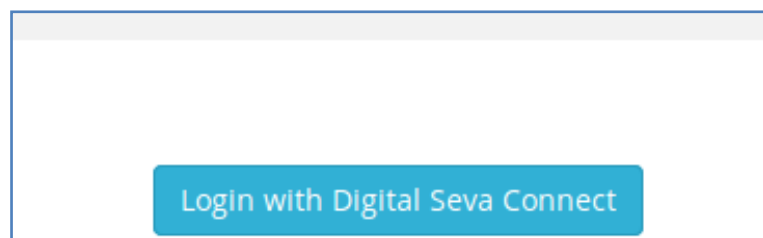
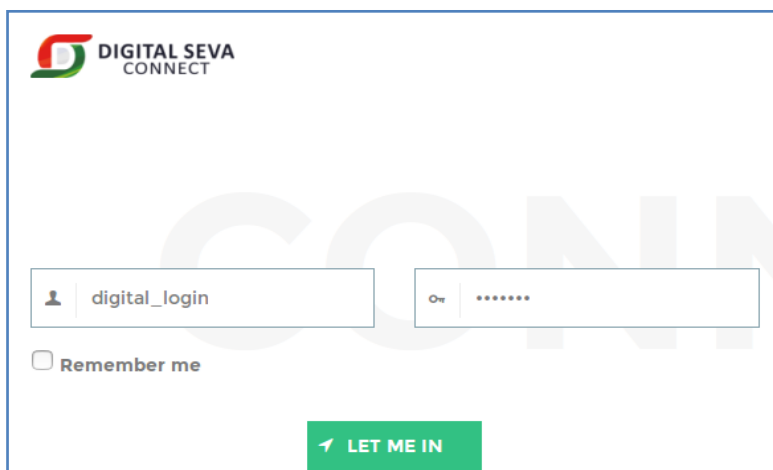


Figure 12



The image shows a login form for Digital Seva Connect. At the top left is the logo with the text "DIGITAL SEVA CONNECT". Below the logo, there are two input fields: the first is labeled "digital_login" and has a person icon; the second is labeled "Or" and has a password icon. Below these fields is a checkbox labeled "Remember me". At the bottom center is a green button with a white arrow and the text "LET ME IN".

Figure 13

Step 2 Handle response to get user data as in the sample **login_success.php**

```
<?php
    include("includes/Connect_client.php");

    function callback()
    {
        $obj = new Connect_client();
        $client = $obj->get_client();

        $params = array('code' => $_GET['code'], 'redirect_uri' =>
REDIRECT_URI);

        $response = $client->getAccessToken(TOKEN_ENDPOINT,
'authorization_code', $params);

        $res = $response['result'];

        $client->setAccessToken($res['access_token']);
        $response = $client->fetch(RESOURCE_URL);

        $csc_id=$response['result']['User']['csc_id'];
        $email=$response['result']['User']['email'];
        $user_type=$response['result']['User']['user_type'];
        $state_code=$response['result']['User']['state_code'];
        $active_status=$response['result']['User']['active_status'];

        session_start();
        $_SESSION['csc_id']=$csc_id;
        $_SESSION['email']=$email;
        $_SESSION['user_type']=$user_type;
        $_SESSION['state_code']=$state_code;
        $_SESSION['active_status']=$active_status;

        header("Location:login.php");
    }
```

```
callback();
```

Digital Seva Response

```
---User Details---  
csc_id : 88888888888888  
email : test@gmail.com  
user_type : VLE  
state_code : DL  
active status : Active
```

Figure 14

Step 3 Create a user session with data as mentioned in **Step 2: login_success.php**

```
session_start();  
$_SESSION['csc_id']=$csc_id;  
$_SESSION['email']=$email;  
$_SESSION['user_type']=$user_type;  
$_SESSION['state_code']=$state_code;  
$_SESSION['active_status']=$active_status;
```

Dependency

PHP Versions Supported: **5.3+, 7**

Libraries: **Mcrypt, Curl**

2.2 JAVA

The illustrated code sample below provides the understanding of using the **java** integration kit.

Step 1 Create a URL as in the Sample **login.jsp**

```
String state = "" + (int) (Math.random() * 1000000);
session.setAttribute("state", state);
String connect_url =
    authorizationAddress +
    "?state=" +
    state +
    "&response_type=code&client_id=" +
    clientId +
    "&redirect_uri=" +
    redirectUri;
%>
<a href="<%= connect_url %>">Login with Digital Seva Connect</a>
```

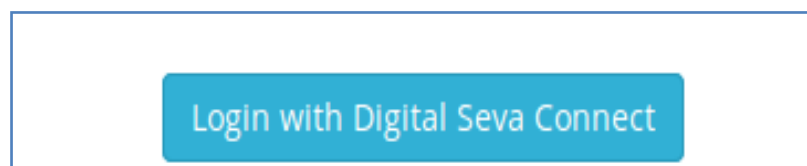


Figure 15

 A login form for Digital Seva Connect. At the top left is the logo, which consists of a stylized 'D' in red and green followed by the text "DIGITAL SEVA CONNECT". Below the logo are two input fields: the first is for the username, containing the text "digital_login", and the second is for the password, containing a series of dots. Below these fields is a checkbox labeled "Remember me". At the bottom right is a green button with a white arrow icon and the text "LET ME IN". A large, faint "CONNA" watermark is visible in the background.

Figure 16

Step 2 Handle response to get user data as in the sample **login_success.jsp**

```

<%
    String resp = request.getQueryString();
    String state_saved = (String)session.getAttribute("state");
    String state_req = request.getParameter("state");
    if(state_saved == null || state_req == null ||
!state_saved.equals(state_req) ){
        resp = "State mismatched! Please try to login again.";
    } else {
        String code = request.getParameter("code");
        if(code == null || code.length() <= 0){
            resp = "Error!! Code not received from server!";
        } else {
            //jsp post a request ..
            String url = tokenAddress;
            String parameters =
"code=" + code + "&" +
"redirect_uri=" + redirectUri + "&" +
"grant_type=authorization_code&" +
"client_id=" + clientId + "&" +
"client_secret=ee0f2b2fd3b57e1ddc0e71c24eafdd57";
            try{
                String ret = sendPost(url, parameters);
                JSONObject tResp = new JSONObject(ret);
                String token = tResp.getString("access_token");

                parameters = "access_token=" + token;
                resp = sendPost(resourceAddress, parameters);

                JSONObject obj = new JSONObject(resp);
                JSONObject user = obj.getJSONObject("User");

                Map<String, Object> mVals = user.toMap();
                resp = "\n<br>----User Details ----";
                for (Map.Entry<String, Object> entry :
mVals.entrySet()) {
                    System.out.println("Key = " + entry.getKey() +
", Value = " + entry.getValue());
                    resp += "\n" + entry.getKey() + " : " +
entry.getValue();
                }
                session.setAttribute("username",
mVals.get("username"));
            }catch(Exception ec){out.write(ec.toString());}
        }
    }
%>

```

Digital Seva Response

```
---User Details---  
csc_id : 88888888888888  
email : test@gmail.com  
user_type : VLE  
state_code : DL  
active status : Active
```

Figure 17

Step 3 Create a user session with data as mentioned in **Step 2 login_success.jsp**

```
session.setAttribute("csc_id", mVals.get("csc_id"));
```

Dependency

Java Cryptography Extension (JCE) unlimited strength jurisdiction policy files for using Bridge Utility platform.

2.3 DOT NET

The illustrated code sample below provides the understanding of using the **dot net** integration kit.

Step 1 Create a URL as in the sample **login.aspx.cs**

```
protected void Button1_Click(object sender, EventArgs e)
{
    try
    {
        if (Session["state"].ToString().Length > 0)
        {
            string url =
ConfigurationManager.AppSettings["CONNECT_SERVER_URI"] +
ConfigurationManager.AppSettings["AUTHORIZATION_ENDPOINT"] + "?state=" +
state + "&response_type=code&client_id=" +
ConfigurationManager.AppSettings["CLIENT_ID"] + "&redirect_uri=" +
ConfigurationManager.AppSettings["CLIENT_CALLBACK_URI"];

            Response.Write(url);
            Response.Redirect(url, true);
        }
        else
        {
            state = ConfigurationManager.AppSettings["merchant"] +
DateTime.Now.Year.ToString() + DateTime.Now.Month.ToString().PadLeft(2,
'0') + DateTime.Now.Day.ToString().PadLeft(2, '0') +
DateTime.Now.Hour.ToString().PadLeft(2, '0') +
DateTime.Now.Minute.ToString().PadLeft(2, '0') +
DateTime.Now.Second.ToString().PadLeft(2, '0') +
DateTime.Now.Millisecond.ToString().PadLeft(4, '0') + "login.aspx";
            string url =
ConfigurationManager.AppSettings["CONNECT_SERVER_URI"] +
ConfigurationManager.AppSettings["AUTHORIZATION_ENDPOINT"] + "?state=" +
state + "&response_type=code&client_id=" +
ConfigurationManager.AppSettings["CLIENT_ID"] + "&redirect_uri=" +
ConfigurationManager.AppSettings["CLIENT_CALLBACK_URI"];
            Response.Write(url);

            Response.Redirect(url, true);
        }
    }
    catch (Exception ex)
    {
        state = ConfigurationManager.AppSettings["merchant"] +
DateTime.Now.Year.ToString() + DateTime.Now.Month.ToString().PadLeft(2,
'0') + DateTime.Now.Day.ToString().PadLeft(2, '0') +
DateTime.Now.Hour.ToString().PadLeft(2, '0') +
DateTime.Now.Minute.ToString().PadLeft(2, '0') +
DateTime.Now.Second.ToString().PadLeft(2, '0') +
DateTime.Now.Millisecond.ToString().PadLeft(4, '0') + "login.aspx";
        string url =
```

```

ConfigurationManager.AppSettings["CONNECT_SERVER_URI"] +
ConfigurationManager.AppSettings["AUTHORIZATION_ENDPOINT"] + "?state=" +
state + "&response_type=code&client_id=" +
ConfigurationManager.AppSettings["CLIENT_ID"] + "&redirect_uri=" +
ConfigurationManager.AppSettings["CLIENT_CALLBACK_URI"];

        Response.Write(url);
        Response.Redirect(url, true);
    }
}

```

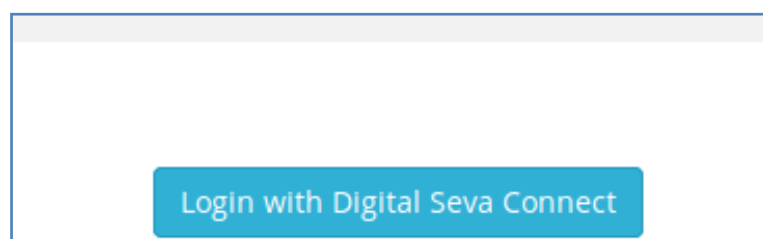


Figure 18

Figure 19

Step 2 Handle response to get user data as in the sample **login.aspx.cs**

```

try
{
    state = Session["state"].ToString();
    try
    {
        if (state == Session["state"].ToString())
        try
        {
            dcrData = Session["data"].ToString().Trim();
            JObject results =
JObject.Parse(Session["data"].ToString().Trim());
            string temp = (string)results["username"];
            if (temp == null)
            {
                dcrData = "<br />";
                JObject a =
JObject.Parse(Session["data"].ToString().Trim());

                foreach (JObject o in
a.First.Children<JObject>())
                {
                    foreach (JProperty p in o.Properties())
                    {
                        string name = p.Name;
                        string value = (string)p.Value;
                        dcrData = dcrData + name + " => " +
value + "<br />";
                    }
                }

                Session["username"] =
(string)results["User"]["username"];

                Session["loggedin"] = "true";
                LinkButton1.Text = "Logout";
                Button1.Visible = false;
            }
            else
            {
                dcrData = "<br />";
                JObject a =
JObject.Parse(Session["data"].ToString().Trim());

                foreach (JObject o in
a.Children<JObject>())
                {
                    foreach (JProperty p in o.Properties())
                    {
                        string name = p.Name;
                        string value = (string)p.Value;
                        dcrData = dcrData + name + " => " +
value + "<br />";
                    }
                }
            }
        }
    }
}

```

```

        Session["username"] =
(string)results["username"];

        Session["loggedin"] = "true";
        LinkButton1.Text = "Logout";
        Button1.Visible = false;
    }

    }
    catch (Exception)
    {
        Session["loggedin"] = "false";
        Button1.Visible = true;
        LinkButton1.Text = "Login";
    }
}
catch (Exception) { }
}
catch (Exception)
{
    state = ConfigurationManager.AppSettings["merchant_id"] +
DateTime.Now.Year.ToString() + DateTime.Now.Month.ToString().PadLeft(2,
'0') + DateTime.Now.Day.ToString().PadLeft(2, '0') +
DateTime.Now.Hour.ToString().PadLeft(2, '0') +
DateTime.Now.Minute.ToString().PadLeft(2, '0') +
DateTime.Now.Second.ToString().PadLeft(2, '0') +
DateTime.Now.Millisecond.ToString().PadLeft(4, '0') + "login.aspx";
    Session["loggedin"] = "false";
    Session["page"] = "login.aspx";
    Button1.Visible = true;
}
}

```