**VPP Oracle Health Patient Portal Integration with Health Gateway**

VPP patient portal is to be launched from within the Health Gateway web site (https://www.healthgateway.gov.bc.ca/):

* Seamless User Experience: Ensuring a smooth and intuitive user experience is crucial.
* Clicking on a link in Health Gateway will launch the Portal in a new window or the same window
* User information (e.g., PHN, birthday) must be passed from Health Gateway to the Portal.
* BC Services Card Authentication: The BC Services Card app is the central identity provider in British Columbia, Canada.
* Health Gateway uses BC Services Card for user authentication and does not have its own auth server.

Steps to use the VPP Patient Portal:

1. The patient logs in to the Health Gateway website authenticated with BC Services Card authentication.
2. The patient clicks on a link in Health Gateway to launch CoreServe.
3. User information and access token from the BC Services Card Auth server are passed to CoreServe.
4. CoreServe launches, allowing the user to access their health record.

here's a comprehensive step-by-step guide to implement the integration between Health Gateway and the VPP Patient Portal, ensuring a seamless user experience and secure handling of access and refresh tokens using the BC Services Card authentication.

**Step 1: Configure Cross-Origin Resource Sharing (CORS) on Health Gateway**

1. **Enable Cross-Origin Access**: Configure Health Gateway to allow cross-origin requests from the VPP Patient Portal’s domain (https://yourvppportal.com).
2. **Set CORS Headers**:

Access-Control-Allow-Origin: https://yourvppportal.com

Access-Control-Allow-Credentials: true

Access-Control-Allow-Methods: GET, POST, OPTIONS

Access-Control-Allow-Headers: Content-Type, Authorization

1. **Security Restriction**: Limit Access-Control-Allow-Origin to the Portal’s specific domain to prevent unauthorized cross-origin access.

**Step 2: Authenticate User with BC Services Card on Health Gateway**

1. **BC Services Card Authentication**: Users log into Health Gateway using the BC Services Card, which authenticates their identity.
2. **Obtain Tokens and User Information**:
   * **Access Token**: Used to authorize API calls.
   * **Refresh Token**: Used to obtain new access tokens without re-authenticating.
   * **User Information**: Essential data like Personal Health Number (PHN) and birthdate.

**Step 3: Pass Tokens and User Information to the VPP Patient Portal**

1. **Launch Portal in a New Window**: When the user clicks the link, Health Gateway opens the VPP Patient Portal in a new window.
2. **Include Tokens and User Data in URL Parameters**:
   * **Example URL**:

https://yourvppportal.com?access\_token=ACCESS\_TOKEN&refresh\_token=REFRESH\_TOKEN&phn=USER\_PHN&birthdate=USER\_BIRTHDATE

1. **Security Considerations**:
   * **Encrypt or Encode Tokens**: To protect sensitive data, consider encrypting the tokens before including them in the URL.
   * **Use HTTPS**: Ensure the URL uses HTTPS to encrypt data in transit.

**Step 4: Retrieve and Verify Tokens in the VPP Patient Portal**

**Client-Side Retrieval**

1. **Extract Tokens and User Data from URL Parameters**:

// Extract tokens and user data from URL parameters

const urlParams = new URLSearchParams(window.location.search);

const accessToken = urlParams.get('access\_token');

const refreshToken = urlParams.get('refresh\_token');

const phn = urlParams.get('phn');

const birthdate = urlParams.get('birthdate');

if (accessToken && refreshToken && phn && birthdate) {

// Store refresh token securely

storeRefreshToken(refreshToken);

// Verify access token

verifyAccessToken(accessToken);

} else {

alert('Authentication data is missing. Please log in again through Health Gateway.');

}

**Secure Storage of Refresh Token**

1. **Store Refresh Token in HTTP-Only Cookie**:
   * **Client-Side Request to Store Refresh Token**:

function storeRefreshToken(refreshToken) {

// Send refresh token to backend for secure storage

axios.post('/api/store-refresh-token', { refresh\_token: refreshToken })

.then(response => {

console.log('Refresh token stored securely.');

})

.catch(error => {

console.error('Failed to store refresh token:', error);

});

}

* + **Backend Endpoint to Store Refresh Token (Flask Example)**:

from flask import Flask, request, jsonify, make\_response

app = Flask(\_\_name\_\_)

@app.route('/api/store-refresh-token', methods=['POST'])

def store\_refresh\_token():

refresh\_token = request.json.get('refresh\_token')

response = make\_response(jsonify({"status": "stored"}))

response.set\_cookie('refresh\_token', refresh\_token, httponly=True, secure=True)

return response

**Backend Verification of Access Token**

1. **Set Up Verification Endpoint**:
   * **Endpoint**: /api/verify-access-token
   * **Verification Code (Python Flask Example)**:

import jwt # PyJWT library for JWT decoding

from flask import request, jsonify

BC\_SERVICES\_PUBLIC\_KEY = "-----BEGIN PUBLIC KEY-----\n...\n-----END PUBLIC KEY-----"

@app.route('/api/verify-access-token', methods=['POST'])

def verify\_access\_token():

access\_token = request.json.get('access\_token')

try:

decoded\_token = jwt.decode(

access\_token,

BC\_SERVICES\_PUBLIC\_KEY,

algorithms=['RS256'],

audience="yourvppportal.com"

)

# Token is valid; proceed to create session

login\_user(decoded\_token)

return jsonify({"status": "success", "user\_data": decoded\_token}), 200

except jwt.ExpiredSignatureError:

return jsonify({"status": "error", "message": "Access token expired"}), 401

except jwt.InvalidTokenError:

return jsonify({"status": "error", "message": "Invalid access token"}), 403

1. **Client-Side Token Verification Request**:

function verifyAccessToken(accessToken) {

axios.post('/api/verify-access-token', { access\_token: accessToken })

.then(response => {

if (response.data.status === 'success') {

const userData = response.data.user\_data;

buildUI(userData);

}

})

.catch(error => {

if (error.response.status === 401) {

// Access token expired, attempt to refresh

refreshAccessToken();

} else {

console.error('Access token verification failed:', error);

alert('Authentication failed. Please log in again through Health Gateway.');

}

});

}

**Step 5: Implement Token Refresh Mechanism in the Portal**

1. **Backend Endpoint for Token Refresh**:
   * **Endpoint**: /api/refresh-access-token
   * **Refresh Token Flow (Flask Example)**:

import requests

BC\_AUTH\_SERVER\_TOKEN\_URL = "https://auth.bcservicescard.gov/token"

CLIENT\_ID = "your\_client\_id"

CLIENT\_SECRET = "your\_client\_secret"

@app.route('/api/refresh-access-token', methods=['POST'])

def refresh\_access\_token():

refresh\_token = request.cookies.get('refresh\_token')

if not refresh\_token:

return jsonify({"status": "error", "message": "No refresh token available"}), 401

response = requests.post(BC\_AUTH\_SERVER\_TOKEN\_URL, data={

'grant\_type': 'refresh\_token',

'refresh\_token': refresh\_token,

'client\_id': CLIENT\_ID,

'client\_secret': CLIENT\_SECRET

})

if response.status\_code == 200:

new\_tokens = response.json()

# Update access token in session or wherever it is stored

session['access\_token'] = new\_tokens['access\_token']

# Optionally update refresh token if it has changed

return jsonify({"status": "success", "access\_token": new\_tokens['access\_token']}), 200

else:

return jsonify({"status": "error", "message": "Failed to refresh access token"}), 401

1. **Client-Side Function to Refresh Access Token**:

function refreshAccessToken() {

axios.post('/api/refresh-access-token')

.then(response => {

if (response.data.status === 'success') {

const newAccessToken = response.data.access\_token;

sessionStorage.setItem('access\_token', newAccessToken);

verifyAccessToken(newAccessToken);

}

})

.catch(error => {

console.error('Failed to refresh access token:', error);

alert('Session expired. Please log in again through Health Gateway.');

});

}

**Step 6: Create Secure Session in the Portal**

1. **Session Creation (Backend)**:

from flask import session

def login\_user(user\_data):

session['phn'] = user\_data['phn']

session['birthdate'] = user\_data['birthdate']

session['user\_id'] = user\_data['sub'] # Unique identifier

# Set session expiration based on access token's expiry

session.permanent = True

app.permanent\_session\_lifetime = timedelta(seconds=user\_data['exp'] - int(time.time()))

1. **Use Session Data to Personalize User Experience**:
   * **Example**:

@app.route('/dashboard')

def dashboard():

if 'user\_id' in session:

# Fetch user-specific data

return render\_template('dashboard.html', phn=session['phn'], birthdate=session['birthdate'])

else:

return redirect('/login')

**Step 7: Ensure HTTPS and Secure Transmission**

1. **Use HTTPS Everywhere**:
   * Ensure that both Health Gateway and the VPP Patient Portal are served over HTTPS to encrypt data in transit.
2. **Secure Cookies**:
   * When setting cookies (e.g., for the refresh token), mark them as Secure and HTTPOnly.

response.set\_cookie('refresh\_token', refresh\_token, httponly=True, secure=True, samesite='Strict')

**Step 8: Implement Error Handling and Expiry Management**

1. **Access Token Expiry Handling**:
   * Before each protected API call, check if the access token is still valid.
   * If expired, initiate the **token refresh flow**.
2. **Refresh Token Expiry Handling**:
   * Refresh tokens usually have a longer lifespan but can expire or be revoked.
   * If the refresh token is invalid or expired, prompt the user to reauthenticate through Health Gateway.
3. **Client-Side Error Handling**:
   * Provide user-friendly error messages and guidance.
   * **Example**:

function handleError(error) {

if (error.response.status === 401) {

alert('Your session has expired. Please log in again through Health Gateway.');

window.location.href = 'https://www.healthgateway.gov.bc.ca/';

} else {

console.error('An error occurred:', error);

alert('An unexpected error occurred. Please try again later.');

}

}

**Step 9: Summary of Workflow**

1. **User logs into Health Gateway** using BC Services Card authentication.
2. **User clicks** to launch the VPP Patient Portal in a new window, with access and refresh tokens and user data passed as URL parameters.
3. **Portal retrieves and verifies** the access token, storing the refresh token securely.
4. **Portal creates a secure session** for the user and allows access to their health records.
5. **Automatic Token Refresh**:
   * Before access token expiry, the Portal uses the refresh token to obtain a new access token from the BC Services Card authorization server.
6. **Error Handling**:
   * If both access and refresh tokens are expired or invalid, the Portal prompts the user to reauthenticate through Health Gateway.

**Additional Security Considerations**

* **Encrypt Tokens in Transit**:
  + Although HTTPS encrypts data in transit, consider encrypting sensitive parameters (like tokens) in the URL to prevent exposure through browser history or logs.
* **Avoid Storing Tokens in Local Storage**:
  + Local storage is accessible via JavaScript and vulnerable to XSS attacks. Prefer HTTP-only cookies or server-side sessions.
* **Set Appropriate Token Lifetimes**:
  + Configure token lifetimes to balance security and user experience.
* **Regularly Update and Patch Systems**:
  + Keep all components (e.g., web servers, libraries) up-to-date to protect against known vulnerabilities.