1. Generate public and private key:

openssl genpkey -algorithm RSA -out private\_key.pem -pkeyopt rsa\_keygen\_bits:2048

openssl rsa -pubout -in private\_key.pem -out public\_key.pem

1. Turn on custom identity option and paste the public key in the section provided to input custom identity
2. Generate a json web token with following code

Token.js

const jwt = require('jsonwebtoken');

const fs = require('fs');

const axios = require('axios');

// Load your private key

const privateKey = fs.readFileSync('private\_key.pem', 'utf-8'); // Replace with the actual path to your private key file

//const clientId = 'd098e898-494c-4b6d-b0d5-68664732d54a';

//const clientSecret = 'NDJiYjc5OTAtMDVkMi00YzRmLTk2YjktY2EwYzU3Y2ZkZWVj';

// Define JWT claims

const payload = {

  iss: 'https://eu-gb.appid.cloud.ibm.com/management/v4/21aa5aae-c640-4524-98cd-24bc5b49dae4', // Replace <region> and <tenant\_id> with your App ID region and tenant ID

  sub: '1234567890', // Replace with the subject for the authentication

  aud: 'https://eu-gb.appid.cloud.ibm.com/oauth/v4/21aa5aae-c640-4524-98cd-24bc5b49dae4', // Audience (App ID region URL)

  exp: Math.floor(Date.now() / 1000) + (60 \* 5), // Token expiration time (5 minutes from now)

  iat: Math.floor(Date.now() / 1000) // Issued at time

};

// Generate the signed JWT

const token = jwt.sign(payload, privateKey, { algorithm: 'RS256' });

console.log('Generated JWT:', token);

/\*

// Define the authentication URL for App ID

const authUrl = 'https://eu-gb.appid.cloud.ibm.com/oauth/v4/21aa5aae-c640-4524-98cd-24bc5b49dae4/token'; // Replace <region> and <tenant\_id> with your App ID region and tenant ID

// Function to authenticate with App ID

async function authenticateWithAppID() {

  try {

    // Make a request to exchange the JWT for an access token

    const response = await axios.post(authUrl, `grant\_type=urn:ietf:params:oauth:grant-type:jwt-bearer&assertion=${encodeURIComponent(token)}`, {

      headers: {

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

        'Authorization': 'Basic ' + Buffer.from(`${clientId}:${clientSecret}`).toString('base64')

      }

    });

    // Display the access token

    console.log('Access Token:', response.data);

  } catch (error) {

    // Handle authentication error

    console.error('Authentication error:', error.response ? error.response.data : error.message);

  }

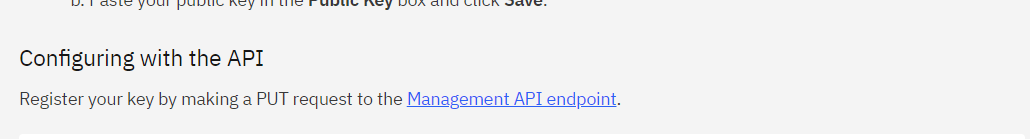
}

// Execute the authentication function

authenticateWithAppID();

\*/

**Explore APP ID API’s from:**

****

**https://us-south.appid.cloud.ibm.com/swagger-ui/#/Management%20API%20-%20Identity%20Providers/mgmt.set\_custom\_idp**

1. Generate an API key in IAM section in order to create an IAM access token for fetching custom identity public keys in order to verify token

Run the following code to generate IAM access toke(This code is to be run only once)

const axios = require('axios');

async function getIamToken(apiKey) {

    const url = 'https://iam.cloud.ibm.com/identity/token';

    try {

        const response = await axios.post(url, null, {

            headers: {

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

            },

            params: {

                apikey: apiKey,

                grant\_type: 'urn:ibm:params:oauth:grant-type:apikey',

            },

        });

        return response.data.access\_token; // Return the access token

    } catch (error) {

        console.error('Error obtaining IAM token:', error.response.data);

        throw error;

    }

}

// Use your API key

const apiKey = '<replace it with your API key from IAM section>';

getIamToken(apiKey)

    .then(token => {

        console.log('IAM Access Token:', token);

        // Now you can use this token in your API requests

    })

    .catch(err => {

        console.error('Failed to get IAM token:', err);

    });

1. Now integrate it with you main application

const express = require('express');

const jwt = require('jsonwebtoken');

const axios = require('axios');

const app = express();

const port = 3000;

// Your App ID public key URL for verifying the JWT signature

const APP\_ID\_PUBLIC\_KEY\_URL = 'https://eu-gb.appid.cloud.ibm.com/management/v4/21aa5aae-c640-4524-98cd-24bc5b49dae4/config/idps/custom';

// Middleware to verify the JWT

async function verifyToken(req, res, next) {

  const authHeader = req.headers['authorization'];

  if (!authHeader || !authHeader.startsWith('Bearer ')) {

    return res.status(401).json({ error: 'Authorization header missing or malformed' });

  }

  const token = authHeader.split(' ')[1];

  try {

    // Fetch the App ID public keys

    const response = await axios.get(APP\_ID\_PUBLIC\_KEY\_URL, {

      headers: {

          Authorization: `Bearer <your IAM Access token generated from previous code>` // Pass the Bearer token here

      },

  });

  console.log(response)

    const publicKeys = response.data.config.publicKey;

console.log(publicKeys);

    if (!publicKeys || publicKeys.length === 0) {

      return res.status(500).json({ error: 'Failed to retrieve public keys' });

    }

    // Decode the JWT header to find the key ID (kid)

    const decodedHeader = jwt.decode(token, { complete: true });

    if (!decodedHeader || !decodedHeader.header) {

      return res.status(400).json({ error: 'Invalid token format' });

    }

    // Find the corresponding public key based on the kid

    //const key = publicKeys.find(k => k.kid === decodedHeader.header.kid);

    if (!publicKeys) {

      return res.status(400).json({ error: 'Public key not found for token' });

    }

    // Construct the PEM formatted public key

   // const publicKey = `-----BEGIN PUBLIC KEY-----\n${publicKeys}\n-----END PUBLIC KEY-----`;

//console.log(publicKey);

    // Verify the token using the public key

    jwt.verify(token, publicKeys, { algorithms: ['RS256'] }, (err, decoded) => {

      if (err) {

        return res.status(401).json({ error: 'Token verification failed', message: err.message });

      }

      // Token is valid, attach user info to request and proceed

      req.user = decoded;

      next();

    });

  } catch (error) {

    res.status(500).json({ error: 'Failed to verify token', message: error.message });

  }

}

// Sample protected route

app.get('/protected', verifyToken, (req, res) => {

  res.json({ message: 'Access granted to protected resource', user: req.user });

});

// Start the server

app.listen(port, () => {

  console.log(`Server running at http://localhost:${port}`);

});