

Workvivo JWT SSO

Yosuke Sawamura
Workvivo Solution Engineer March, 2025

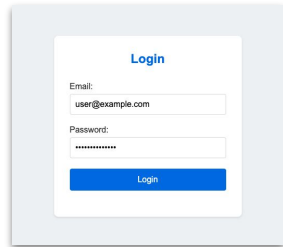
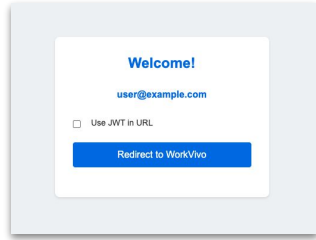


Here are some options for you, I hope they are helpful.

High-level Overview

Third party Web Infrastructure

Workvivo



2. Third party system will need to know the matching email address on Workvivo to generate user token.
Then user receives the token (JWT) for redirect.



3. Then using the token to
redirect into Workvivo.



1. User logs in through third party managed url.
Credential can be dedicated.
(e.g. non-workvivo id/password)

4. User now gets into Workvivo portal without
entering any additional credential.

Requirements and steps

1. Generate your key pair. (Private and Public key)
- ↓
2. Generate JWKS based on your Public key.
- ↓
3. Make JWKS accessible towards Workvivo.
- ↓
4. Configure your environment on Workvivo portal.
- ↓
5. Generate token in JWT format signed by your Private key.
- ↓
6. Forward the token to let the user redirect to Workvivo.

1. Generate your key pair. (Private and Public key)

**Server side example.*

This is an example to generate Private & Public key pairs.

This example saves generated files in the same directory.

Private key (jwtRS256.key) will be used to sign JWT at the end.

Public key (jwtRS256.key.pub) will be used when generating JWKS.

```
const crypto = require('crypto');
const fs = require('fs');

// Generate key pair
const { privateKey, publicKey } = crypto.generateKeyPairSync('rsa', {
  modulusLength: 2048,
  publicKeyEncoding: {
    type: 'spki',
    format: 'pem'
  },
  privateKeyEncoding: {
    type: 'pkcs8',
    format: 'pem'
  }
});


// Write the keys to files
fs.writeFileSync('jwtRS256.key', privateKey);
fs.writeFileSync('jwtRS256.key.pub', publicKey);

console.log('Key pair generated successfully!');
```

2. Generate JWKS based on your Public key.

**Server side example.*


This is an example to create a 'jwks.json' based on the previously generated public key 'jwtRS256.key.pub'.



'jwks.json' will include public key information along with 'kid'(Key Id) which is required when generating the final JWT token for the user.

**Recommended to host the Public Key in JWKS format.*

Make sure the format includes kid (Key Id).
"kid" will be required for JWT.



```
const fs = require('fs');
const crypto = require('crypto');
const jose = require('node-jose');

async function generateJWKS() {
  try {
    // Read the public key
    const publicKey = fs.readFileSync('jwtRS256.key.pub');

    // Create a keystore
    const keystore = jose.JWK.createKeyStore();

    // Import the public key into the keystore
    await keystore.add(publicKey, 'pem', {
      use: 'sig',
      alg: 'RS256',
      kid: crypto.randomBytes(16).toString('hex') // Generate a random key ID
    });

    // Export the keystore as JWKS
    const jwks = keystore.toJSON();

    // Write to jwks.json
    fs.writeFileSync('jwks.json', JSON.stringify(jwks, null, 4));

    console.log('Generated jwks.json successfully');
  } catch (error) {
    console.error('Error generating JWKS:', error);
  }
}

generateJWKS();
```

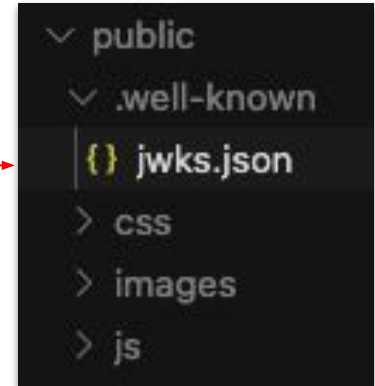
3. Make JWKS accessible from Workvivo.

Public keys are required to be published and to be discoverable via HTTP GET to a well known URL.

Make sure the format is in JWK format along with kid (Key Id).
(e.g. <https://workvivo.com/.well-known/jwks.json>)

Note) Zoom/Workvivo are only interested in the Public key via JWKS. (Do not expose your private key)

**Server side example.*



4. Configure your environment on Workvivo portal.

Note) From your Workvivo portal, go to "Admin > Administration > API Keys & JWT Settings". Make sure the user you are logging with has "It Administrator" role.

Will enable the JWTSSO mechanism.

Will force your authentication mechanism. Workvivo default login url will be disabled.

Will enable token to be included in the redirect url path.

Your login url. (e.g. third party authentication url.)

Url path to the jwks.json file.

Url of your path to prevent CORS failure during user's redirection.
Typically your third-party url.

Select the key type you are using.
Recommend 'JWKS' along with 'Public Key URL'(above) and leave 'JWKS Hosted on Workvivo'(below) empty.

☒ Enable login via JWT SSO

☐ Force JWT SSO authentication only

☒ Enable JWT in URL

☐ JWT SSO Test mode only

Test User's email address

JWT SSO login URL

https://your-domain/login

Public Key URL

https://your-domain/.well-known/jwks.json

Allowed Origins for CORS

https://your-domain

Key Type

JWKS

JWKS Hosted on Workvivo

5. Generate token in JWT format signed by your Private key

iss: Mandatory, Issuer (your domain)

sub: Mandatory, Subject (e.g. your target user's email address.)

aud: Mandatory, Audience (your Wokrvivo subdomain)

iat: Mandatory, Issued at (epoc format)

nbf: Mandatory, Not before (epoc format)

exp: Mandatory, Expiration (epoc format)

email: Mandatory, Your target user's email address.

state: Mandatory, Not applicable if "disableState" is set to true.
Random string. If set, JWT will only be active once.

disableState: Optional, If set to true, "state" will be ignored and can be used multiple times.

organisation_id: Mandatory, Your Workvivo Id.

mobile: Optional, If set to true, JWT will only be able to redirect to launch mobile app.

Make sure header includes matching "kid"(Key ID) from jwks.json.

Generate and sign the payload using your private key.

```
const jwt = require('jsonwebtoken');
const fs = require('fs');

// Read the private key (for signing)
const privateKey = fs.readFileSync('jwtRS256.key');

// Current time in seconds
const currentTime = Math.floor(Date.now() / 1000);

// Payload
const payload = {
  // Mandatory, Issuer (your domain)
  iss: '{your-domain}',
  // Mandatory, Subject (e.g. your target user's email address.)
  sub: 'user@example.com',
  // Audience
  aud: '{your-workvivo-subdomain}',
  // Mandatory, Issued at
  iat: currentTime,
  // Mandatory, Not before
  nbf: currentTime,
  // Mandatory, Example for a 1 hour expiration (current time + 1 hour)
  exp: currentTime + (60 * 60),
  // Mandatory, Your target user's email address.
  email: 'user@example.com',
  // Mandatory if "disableState" is not set to true.
  // Random string. If set, JWT will only be used once.
  state: '1234567890',
  // Optional. If set to true, "state" will be ignored and can be used multiple times.
  // disableState: true,
  // Mandatory. Your target user's Workvivo Id.
  organisation_id: 1234
  // Optional. If set to true, JWT will only be used to redirect to launch mobile app.
  // mobile: true
};

// Read JWKS file to get kid
const jwks = JSON.parse(fs.readFileSync('jwks.json'));
const kid = jwks.keys[0].kid; // Get the kid from the first key

// Sign the token with kid in the header
const token = jwt.sign(payload, privateKey, {
  algorithm: 'RS256',
  header: {
    kid: kid
  }
});

console.log('Generated JWT Token:');
console.log(token);
```

Note) Since private key is required to sign, we recommend generating JWT on server side.

6. Forward the token to let the user redirect to Workvivo.

**Client side example.*

The JWT will need to be sent to this dedicated endpoint
'https://{your-workvivo-subdomain}/proxy/redirect/sso'.
(e.g. https://acme.workvivo.com/proxy/redirect/sso)

Can add token along with the url if you have enabled "Enable JWT in URL".
(Workvivo admin portal configuration required)

Recommended to have JWT included via
Header Parameter: 'x-workvivo-jwt'.

Seek for the url from successful reply and redirect accordingly.

```
document.getElementById('actionButton').addEventListener('click', function() {
  modal.style.display = "block";
  const token = document.getElementById('jwt-token').value;
  const useUrlParam = document.getElementById('useUrlParam').checked;
  const baseUrl = 'https://{your-workvivo-subdomain}/proxy/redirect/sso/';

  if (useUrlParam) {
    window.location.href = `${baseUrl}${encodeURIComponent(token)}`;
  } else {
    fetch(baseUrl, {
      method: 'GET',
      headers: {
        'x-workvivo-jwt': token
      },
      credentials: "include"
    })
    .then(response => {
      console.log('response: ', response);
      if (response.redirected) {
        window.location.replace(response.url);
      } else {
        modal.style.display = "none";
        throw new Error('No redirect URL received');
      }
    })
    .catch(error => {
      modal.style.display = "none";
      console.error('There was a problem with the operation:', error);
    });
  }
});
```

JWTSSO Sample demo

<https://github.com/Yosuke-wokrvivo/Workvivo-JWTSSO> (2025-03-27 private: please contact if required)

Workvivo Support Documents

JWT SSO (guide)

https://docs.zoom.us/doc/pAU65ZG0R4qW9_fHizY5Jg

JWT SSO (guides and requirements)

<https://developer.workvivo.com/#07d356f1-3995-4f9b-82d9-9fb4a9b8f159>

User Contextual API

<https://developer.workvivo.com/#ee314687-5d11-4217-bb1f-5eace5135438>

Thank you!



workvivo
by zoom