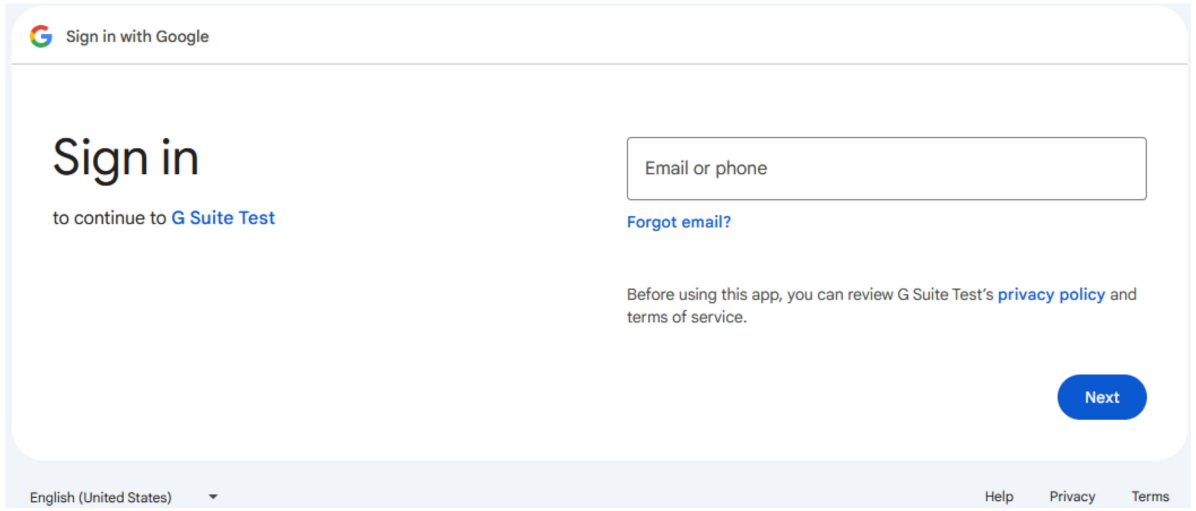
OAUTH 2.0 SECURE WEB SERVICES

OAUTH 2.0 SECURE WEB SERVICES

- **OAuth 2.0 is the modern standard for securing access to Web services**
- **Allows you to get authorized with Web services, such as Google, Salesforce or any Web service that uses OAuth 2.0**
- **Example: An application wants to be able to upload and download a file to a Google Drive**
  - Oauth 2.0 is used by the application to get the user to sign in to their Google account and allow their application access to their Google drive



Sign in with Google

Sign in

to continue to G Suite Test

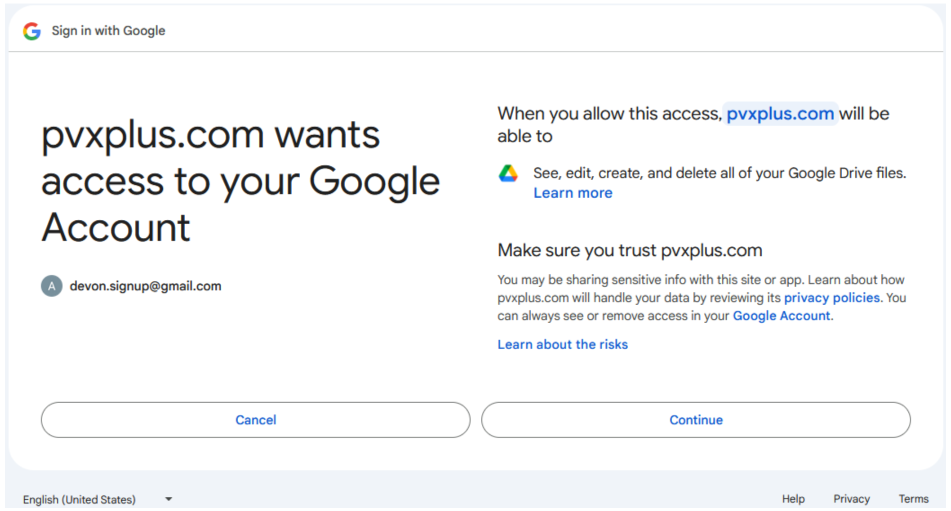
Email or phone

Forgot email?

Before using this app, you can review G Suite Test's privacy policy and terms of service.

Next

English (United States) Help Privacy Terms



Sign in with Google

pvxplus.com wants access to your Google Account

devon.signup@gmail.com

When you allow this access, pvxplus.com will be able to

See, edit, create, and delete all of your Google Drive files. Learn more

Make sure you trust pvxplus.com

You may be sharing sensitive info with this site or app. Learn about how pvxplus.com will handle your data by reviewing its privacy policies. You can always see or remove access in your Google Account.

Learn about the risks

Cancel Continue

English (United States) Help Privacy Terms

OAUTH 2.0 SECURE WEB SERVICES

- **Two types of OAuth 2.0 to consider**
  - Grant type
    - **client_credentials** - simpler and can be handled with a simple Web request to the token endpoint URL
      - Adds extra layer on top of username and password that is needed for web service access
      - This layer can be modified/revoked at any time separate to username and password
    - **authorization_code** - requires user to allow access via Web browser
      - Adds same extra layer as above with same benefits
      - Adds another extra layer where a user has to manually allow the application access
      - The application can ask for specific access and the user can pick and choose which they grant
      - Supports refresh tokens to avoid asking the user every time

OAUTH 2.0 SECURE WEB SERVICES

- **You set up a User ID/Client with the Web service provider and they will provide you with a Client ID and a secret code, as well as one or two URLs (these may be the same)**
  - Authorization endpoint URL (request URL for users to allow access)
  - Token endpoint URL (get access/refresh token)

- **To access an OAuth 2.0 restricted Web service, an access token must be acquired and then passed in with the header of the Web service request**
  - Access tokens expire, and once expired, a new access token must be requested to make a new Web service request
  - This token must be included in the HTML header for any subsequent requests

OAUTH 2.0 SECURE WEB SERVICES

How to get access token for grant type client_credentials

- **Make an HTTP POST request to the token endpoint of the OAuth 2.0 Web service you want access to**
  - The header of the request must include "Authorization: Basic " followed by the BASE64 encoded Client ID and Client secret separated by a : (colon)
  - The body of the request must be "grant_type=client_credentials"
  - The response from a successful request is a 200 status in the response header and the access token via a JSON response

```
{
  "access_token": "Access-Token",
  "token_type": "Bearer",
  "expires_in": 3600
}
```

OAUTH 2.0 SECURE WEB SERVICES

Example

```
clientId$="xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
clientSecret$="xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
call "*WEB/BASE64;ENCODE_STR",clientId$+":"+clientSecret$,usrpsw$
extrahdr$="Authorization: Basic "+usrpsw$
reqData$="grant_type=client_credentials"
call "*plus/web/request","https://www.exsrvr.com/oauth2/token",reqData$,resp$,resphdr$,"","",extrahdr$
dim load jsonAuth${all}=resp$
accessToken$=jsonAuth$["access_token"]
```

OAUTH 2.0 SECURE WEB SERVICES

How to get access token for grant type authorization_code

- **The First stage is to have the user grant your application access to an account of the provider (e.g. grant your application access to a Salesforce account)**
- **If you have done this step before and saved the Refresh token, you can skip this step**
- **First stage steps are:**

1. The application identifies itself to the provider, giving it the Client ID and secret code
2. The service provider returns PVX Plus a URL that the user must go to in order to authorize access
3. The user authorizes access to the provider via a Web browser

OAUTH 2.0 SECURE WEB SERVICES

How to get access token for grant type authorization_code

- **The Second stage is where you request an access and refresh using the token endpoint of the OAuth 2.0 Web service you want access to**
- **Second stage steps are:**

1. Request an Access token and a Refresh token from the web service provider using the Token_URL$
2. If this is the first time, save the Refresh token to avoid logging in the next time

OAUTH 2.0 SECURE WEB SERVICES

- **The "*obj/oauth2" object handles OAuth 2.0 grant type authorization_code for you**
- **The object has a few predefined services so you don't have to specify the authorization and token URLs**
  - Google
  - Salesforce
  - If accessing a non-predefined service, just specify URLs via the **Authorization_URL$ and Token_URL$ properties**
- **Properties used to set ClientID$ and client_secret$**
- **Methods used to perform first and second stage of authorization**
  - First stage:
    - **Enable_Certification(msg$)**
      - msg$ will be the message that appears on the authorization accepted screen
    - **Get_Authorization_URL$(scope$, prompt$)**
      - scope$ is used by some Oauth 2.0 servers when they define multiple scopes of access to specify what access they require
      - prompt$ specifies whether the user is prompted and for what when they request authorization **https://developers.google.com/identity/openid-connect/openid-connect#prompt**
  - Second stage: **Get_Access_token()**

OAUTH 2.0 SECURE WEB SERVICES

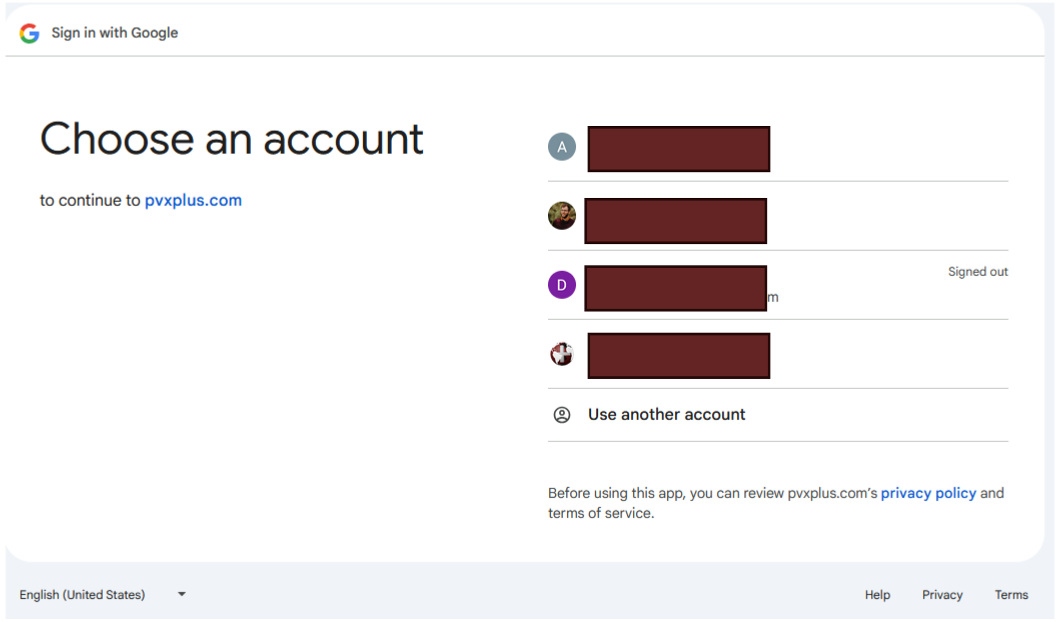
- **The First stage requires an OAuth 2.0 agent to process the user authorization**
  - This *obj/oauth2 object, by default, points to a PVX Plus Technologies hosted OAuth 2.0 agent
    - https://www.pvxplus.com/oauth.pvp
  - You may need to register the agent URL used with the Web service so that it knows it is safe to redirect to that URL
    - This is usually done from a Web browser via a site provided by the Web service
  - It is also possible to self-host the OAuth 2.0 agent
    - Self-hosting may be desirable if you want to avoid relying on the PVX Plus servers being up or if you want to keep it in house for security
    - To Self-host
      - Have a web server setup that can run PxPlus programs
      - Copy the files from the *web/services/oauth2/agent directory to the Web server docroot directory
      - Set the **Agent_URL$** property to my_server_url/oauthagent.pvp

OAUTH 2.0 SECURE WEB SERVICES

Example

```
clientId$="xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx"
clientSecret$="xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx“
oAuth2=new("*obj/oauth2")
oAuth2'Service$="google"
oAuth2'client_id$=clientID$; oAuth2'client_secret$=clientSecret$
if refreshToken$="" then {
oAuth2'Enable_Certification("Do you consent to allow access of your Google account to example app?")
wait 1
url$=oAuth2'Get_Authorization_URL$("https://www.googleapis.com/auth/drive","consent select_account")
system_help url$
input "Press any key to continue after logging into account and allowing PxPlus access:",*;print ""
} else { oAuth2'Refresh_token$=refreshToken$ }
oAuth2'Get_Access_token()
refreshToken$=oAuth2'Refresh_token$
accessToken$=oAuth2'Access_token$
drop object oAuth2
```

OAUTH 2.0 SECURE WEB SERVICES



Sign in with Google

Choose an account

to continue to pvxplus.com

A [redacted]

[redacted]

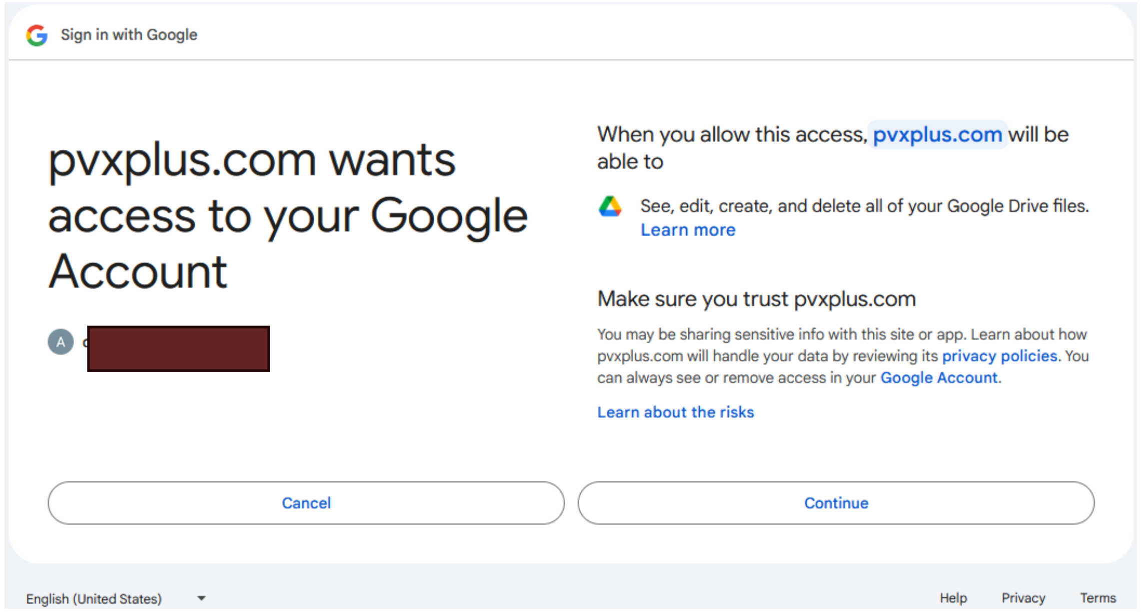
D [redacted]m Signed out

[redacted]

Use another account

Before using this app, you can review pvxplus.com's privacy policy and terms of service.

English (United States) Help Privacy Terms



Sign in with Google

pvxplus.com wants access to your Google Account

A [redacted]

When you allow this access, pvxplus.com will be able to

See, edit, create, and delete all of your Google Drive files. Learn more

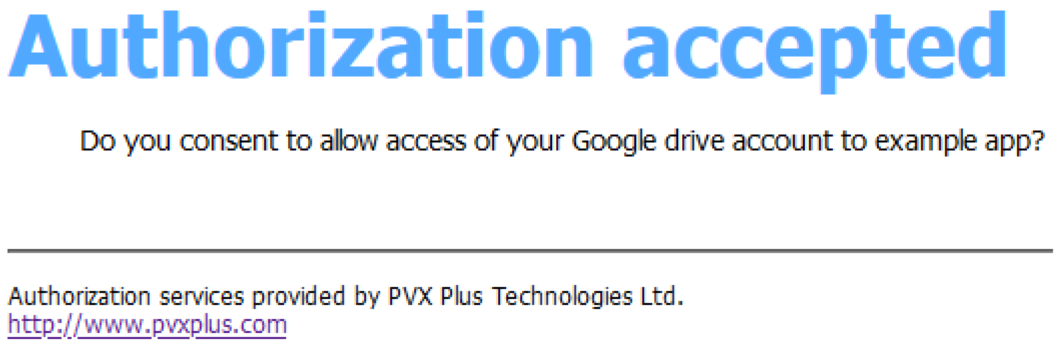
Make sure you trust pvxplus.com

You may be sharing sensitive info with this site or app. Learn about how pvxplus.com will handle your data by reviewing its privacy policies. You can always see or remove access in your Google Account.

Learn about the risks

Cancel Continue

English (United States) Help Privacy Terms



Authorization accepted

Do you consent to allow access of your Google drive account to example app?

Authorization services provided by PVX Plus Technologies Ltd.
http://www.pvxplus.com

OAUTH 2.0 SECURE WEB SERVICES

Make Request with Access Token

- **Either way you acquire an Access token making the Web request is the same**
- **Request the Web service with an "Authorization: Bearer " followed by the Access token in the header**
  - The Web service request is otherwise the same as a request to a Web service with no OAuth2 security

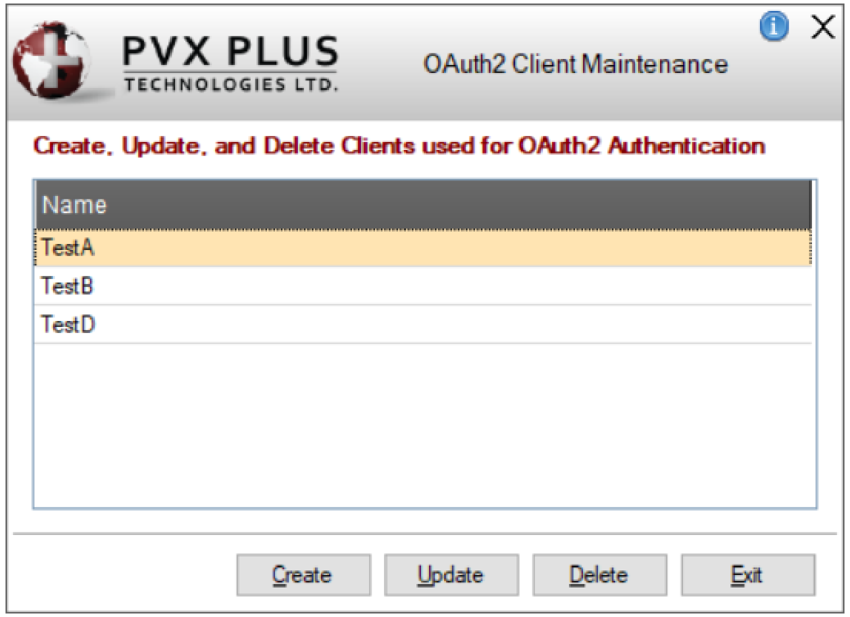
```
authHdr$="Authorization: Bearer "+accessToken$
call "*plus/web/request","https://www.exsrvr.com/exService","",resp$,resphdr$,"","",authHdr$
```

OAUTH 2.0 SECURE WEB SERVICES

- **PxPlus provides some built-in Web Services**
  - Query
  - Chart
  - Report
  - File Maintenance
  - File Access
- **OAuth2 security can be added to restrict access to PxPlus Web Services**
  - First, OAuth2 clients must be defined using either OAuth2 Client Maintenance or the OAuth2 Clients Object
  - Next, access is restricted either via NOMADS security on a query or report or by security enabled in Web Services Maintenance
- **The grant type is client_credentials and the token URL is pxplusServer/services/oauth2/token.pxp**

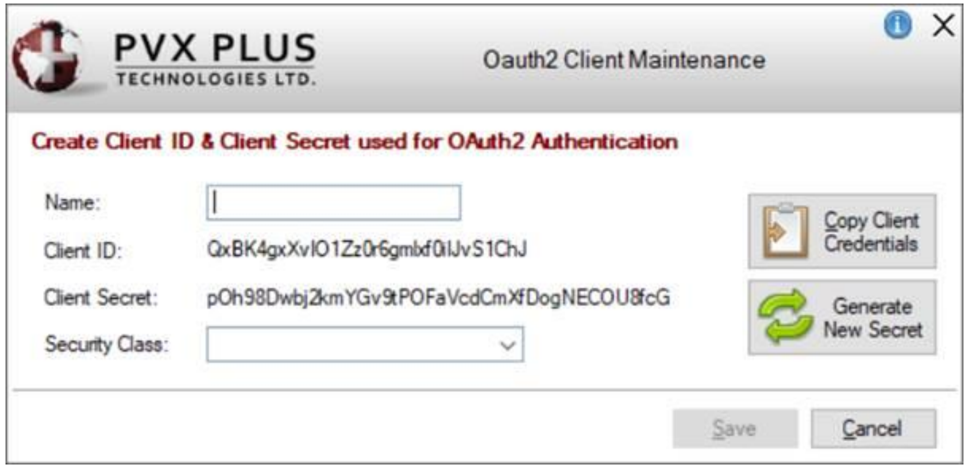
OAUTH 2.0 SECURE WEB SERVICES

- **You must first set up Security Classifications and at least an ADMIN User in User Maintenance prior to setting up OAuth2 clients**
- **OAuth2 Client Maintenance is used for adding and maintaining OAuth2 clients**
  - OAuth2 clients are required to access PxPlus Web Services that have access restricted either via NOMADS security on the query or report or by security enabled in Web Services Maintenance
- **OAuth2 allows for strong security by properly managing clients**
  - If a user's system has been compromised, you can change the Client secret, thus revoking the compromised credentials access
  - If a user no longer needs access or access needs to be revoked, the client can be deleted, thus revoking the user access
  - OAuth2 clients can be managed programmatically and/or without a graphical user interface using the **OAuth2 Clients Object**


PVX PLUS TECHNOLOGIES LTD. OAuth2 Client Maintenance

Create, Update, and Delete Clients used for OAuth2 Authentication

Name
TestA
TestB
TestD


PVX PLUS TECHNOLOGIES LTD. Oauth2 Client Maintenance

Create Client ID & Client Secret used for OAuth2 Authentication

Name:
Client ID: QxBK4gxXvIO1Zz0r6gmbf0iIJvS1ChJ
Client Secret: pOh98Dwbj2kmYGv9tPOFaVcdCmXfDogNECOU8fcG
Security Class:

OAUTH 2.0 SECURE WEB SERVICES

- OAuth 2.0 Clients Object **(*web/services/oauth2/clients)**
  - Maintain OAuth 2.0 clients programmatically and without the need for a user interface
  - Generate and validate Access tokens

```
! Create a new Oauth 2.0 client
oauth2_clients=new("*web/services/oauth2/clients",adminUsername$,adminPassword$)
read data from oauth2_clients'SaveNewClient$("ABC Shipping", "USER") to client_Id$,client_Secret$,access_Token_Key$
```

```
! Revoke Access to Compromised Client by Changing Client Secret
oauth2_clients=new("*web/services/oauth2/clients",adminUsername$,adminPassword$)
read data from oauth2_clients'GetClient$("ABC Shipping") to client_Id$,client_Secret$,access_Token_Key$,security_Class$
oauth2_clients'SaveClient("ABC Shipping", client_Id$, oauth2_clients'NewClientSecret$() ,access_Token_Key$,security_Class$)
```

OAUTH 2.0 SECURE WEB SERVICES

- **Add OAuth2 Security to PxPlus-built Web Service**
- **The grant type is client_credentials and the token URL is pxplusServer/services/oauth2/token.pxp**
  - Provide this to the consumers of your Web service
- **It is possible to implement your own OAuth2 Access token server using the OAuth2 Clients Object if the one provided with PxPlus does not meet an application's requirements**
- **Use the OAuth 2.0 Clients Object in the code for your Web service to validate Access token**

```
if len(%http_authorization$)>=7 and lcs(mid(%http_authorization$,1,7))="bearer " {
    base64AccessToken$=stp(mid(%http_authorization$,8,err=Return_auth_err),"B")
    accessToken$=cvs(base64AccessToken$,"BASE64URL:ASCII",0)
    oauth2clients=new("*web/services/oauth2/clients",err=return_auth_err)
    if oauth2clients'ValidateAccessToken$(accessToken$)="" then goto return_auth_err
    drop object oauth2clients,err=*next
 }
```