Authentication with OAuth

7/29/2021 • 2 minutes to read • Edit Online

IMPORTANT

Starting March 1st, 2022 we will requiremulti-factor authenticationforall users who sign in through a third-party application that uses the Bing Ads API, Content API, and Hotel APIs.

You must update your application to get user consent using the new `msads.manage` scope. All application developers must take action to use the new scope.

For more information see the Multi-factor authentication requirement guide.

Consider the user that you want to sign in e.g., example@contoso.com. The Bing Ads API will not accept that email address and password. Instead you need to set the AuthenticationToken header element that contains a user access token. You can think of an access token as representing a user name and password.

How can you get an access token for a user? As an application developer you'll use a Microsoft authorization URL to prompt the Microsoft Advertising user for consent. Once a user provides consent, you can get an access token and act on behalf of the user.

Microsoft Advertising leverages the Microsoft identity platform endpoint for developers and the OAuth 2.0 protocol to authenticate work or school accounts from Azure Active Directory (AAD) and personal Microsoft accounts (MSA), such as hotmail.com, outlook.com, and msn.com.

1. Register an application
2. Request user consent for your application to manage their Microsoft Advertising accounts
3. Get access and refresh tokens
4. Make your first API call

TIP

For details about how to get access and refresh tokens using the Bing Ads SDKs, see Authentication With the SDKs.

Next steps

Register an application

See Also

Get started

Register an application

7/29/2021 • 2 minutes to read • Edit Online

IMPORTANT

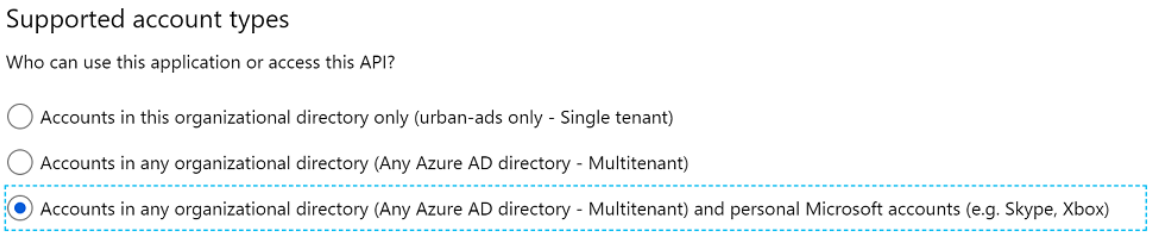
Starting March 1st, 2022 we will requiremulti-factor authenticationforall users who sign in through a third-party application that uses the Bing Ads API, Content API, and Hotel APIs.

You must update your application to get user consent using the new `msads.manage` scope. All application developers must take action to use the new scope.

For more information see the Multi-factor authentication requirement guide.

Before your application can authenticate Microsoft Advertising users, you must register your application and get the corresponding client ID and client secret.

1. Navigate to the Microsoft identity platform for developers in the Azure portal - App registrations page. You can login using either a personal Microsoft Account or a Work or School Account.
2. Select **New registration**.
3. When the **Register an application page** appears, enter your application's registration information:
   - In the **Name** section, enter a meaningful application name that will be displayed to users of the app, for example `My browserless client`.
   - In the **Supported account types** section, select **Accounts in any organizational directory and personal Microsoft accounts**.



IMPORTANT

You must select **Accounts in any organizational directory and personal Microsoft accounts** during the initial app registration. This setting cannot be updated later in the portal unless you modify the application manifest e.g., `"signInAudience": "AzureADandPersonalMicrosoftAccount"`.

4. Select **Register** to create the application.
5. On the app **Overview** page, find the **Application (client) ID** value and record it for later. You will use it as the `client_id` when you request user consent and get an access token.
6. Select the **Add a Redirect URI** link and then you should see the **Redirect URIs** page.
   - For web applications, provide the base URL of your application. For example, http://localhost:31544 might be the URL for a web application running on your local machine. Users would use this URL to sign into a web client application.
   - For public applications, locate the **Suggested Redirect URIs for public clients (mobile, desktop)**

section. Select the **https://login.microsoftonline.com/common/oauth2/nativeclient** URI.

> **IMPORTANT**
>
> Clients running apps on services that span regions and devices such as Microsoft Azure should register a web application with client secret. You can get a refresh token on one device and refresh it on another so long as you have the same client ID and client secret. If you register a public application without a client secret, then you cannot use a refresh token across devices. A confidential token is bound to the client secret.

7. For web applications, select **Certificates & secrets** under **Manage**. Select the **New client secret** button. Enter a value in **Description**, select any option for **Expires** and choose **Add**. Copy the client secret value before leaving the page. You will use it later as the `client_secret` to get an access token.

Next steps

Request user consent

See Also

Get started

Request user consent

8/4/2021 • 5 minutes to read • Edit Online

IMPORTANT

Starting March 1st, 2022 we will requiremulti-factor authenticationforall users who sign in through a third-party application that uses the Bing Ads API, Content API, and Hotel APIs.

You must update your application to get user consent using the new `msads.manage` scope. All application developers must take action to use the new scope.

For more information see the Multi-factor authentication requirement guide.

Once you have registered an application you need to get user consent for you to manage their Microsoft Advertising account.

TIP

For troubleshooting help, see the Common OAuth errors guide.

Each user must be prompted and provide consent through a web browser control at least once for your application to manage their Microsoft Advertising accounts. This is a standard OAuth 2.0 flow and is defined in detail in the Authorization Code Grant section of the OAuth 2.0 spec.

The authorization code flow begins with the client directing the user to the `/authorize` endpoint. In this request, the client indicates the permissions it needs to acquire from the user:

```
// Line breaks for legibility only

https://login.microsoftonline.com/{tenant}/oauth2/v2.0/authorize?
client_id=6731de76-14a6-49ae-97bc-6eba6914391e
&response_type=code
&redirect_uri=http%3A%2F%2Flocalhost%2Fmyapp%2F
&response_mode=query
&scope=openid%20offline_access%20https%3A%2F%2Fads.microsoft.com%2Fmsads.manage
&state=12345
```

1. Click the link below to execute this example request for user consent.

   https://login.microsoftonline.com/common/oauth2/v2.0/authorize?client_id=6731de76-14a6-49ae-97bc-6eba6914391e&response_type=code&redirect_uri=http%3A%2F%2Flocalhost%2Fmyapp%2F&response_mode=query&scope=openid%20offline_access%20https%3A%2F%2Fads.microsoft.com%2Fmsads.manage&state=12345

2. Sign in with your Microsoft account credentials and grant the **Tutorial Sample App** consent to manage your Microsoft Advertising accounts.

3. After signing in, your browser should be redirected to `https://localhost/myapp/` with a `code` in the address bar. You can ignore the error message on the page.

4. Next you'll use the code to get access and refresh tokens.

The Microsoft identity platform endpoint will also ensure that the user has consented to the permissions indicated in the `scope` query parameter. If the user has not consented to any of those permissions, it will ask the user to consent to the required permissions. Although this guide is primarily focused on managing Microsoft Advertising accounts via `scope=https://ads.microsoft.com/msads.manage`, more details about permissions and consent in the Microsoft identity platform are provided here.

Once the user authenticates and grants consent, the Microsoft identity platform endpoint will return a response to your app at the indicated `redirect_uri`, using the method specified in the `response_mode` parameter. For

example the callback URI includes an authorization code as follows if the user granted permissions for your application to manage their Microsoft Advertising accounts: *http://localhost/myapp/?code=CodeGoesHere&state=12345*.

If the user granted your application permissions to manage their Microsoft Advertising accounts, you should use the code right away in the next step. The short duration of the authorization code, approximately 5 minutes, is subject to change. If the user denied your application permissions to manage their Microsoft Advertising accounts, the callback URI includes an error and error description field as follows: *http://localhost/myapp/?error=access_denied&error_description=ERROR_DESCRIPTION&state=ClientStateGoesHere*. For more details about OAuth errors, please see Common OAuth Errors and Authentication and authorization error codes.

The following table includes parameters that Bing Ads API clients can include in the consent request. For additional details about optional parameters see the Microsoft identity platform OAuth 2.0 authorization code flow documentation.

PARAMETER	REQUIRED/OPTIONAL	DESCRIPTION
`client_id`	required	The application (client) ID that the Azure portal - App registrations portal assigned your app.
`code_challenge`	recommended	Used to secure authorization code grants via Proof Key for Code Exchange (PKCE). Required if `code_challenge_method` is included. For more information, see the PKCE RFC. This is now recommended for all application types - native apps, SPAs, and confidential clients like web apps.
`code_challenge_method`	recommended	The method used to encode the `code_verifier` for the `code_challenge` parameter. Can be one of the following values: - `plain` - `S256` If excluded, `code_challenge` is assumed to be plaintext if `code_challenge` is included. Microsoft identity platform supports both `plain` and `S256`. For more information, see the PKCE RFC.

PARAMETER	REQUIRED/OPTIONAL	DESCRIPTION
`prompt`	optional	Indicates the type of user interaction that your application requires. Supported values include the following: - `prompt=login` will force the user to enter their credentials on that request, negating single-sign on. - `prompt=none` is the opposite of "login" i.e., it will ensure that the user is not presented with any interactive prompt whatsoever. If the request can't be completed silently via single-sign on, the Microsoft identity platform endpoint will return an interaction_required error. - `prompt=consent` will trigger the OAuth consent dialog after the user signs in, asking the user to grant permissions to the app. - `prompt=select_account` will interrupt single sign-on and provide the account selection experience, listing all the accounts in session or any remembered account or an option to choose to use a different account altogether.
`redirect_uri`	required	The redirect_uri of your app, where authentication responses can be sent and received by your app. It must exactly match one of the redirect_uris you registered in the portal, except it must be url encoded. For native & mobile apps, you should use the default value of `https://login.microsoftonline.com/common/oauth2/nati`. .
`response_mode`	recommended	Specifies the method that should be used to send the resulting token back to your app. Can be one of the following: - `query` - `fragment` - `form_post` `query` provides the code as a query string parameter on your redirect URI. If you're requesting an ID token using the implicit flow, you cannot use `query` as specified in the OpenID spec. If you're requesting just the code, you can use `query`, `fragment`, or `form_post`. `form_post` executes a POST containing the code to your redirect URI. For more info, see OpenID Connect protocol.
`response_type`	required	Must include `code` for the authorization code flow.

PARAMETER	REQUIRED/OPTIONAL	DESCRIPTION
`scope`	required	A space-separated list of scopes that you want the user to consent to. Be sure to include `https://ads.microsoft.com/msads.manage` to prompt the user for Microsoft Advertising access. Include `offline_access` to ensure that a refresh token is included in the response.
`state`	recommended	A value included in the request that will also be returned in the token response. It can be a string of any content that you wish. A randomly generated unique value is typically used for preventing cross-site request forgery attacks. The value can also encode information about the user's state in the app before the authentication request occurred, such as the page or view they were on.
`tenant`	required	The `{tenant}` value in the path of the request can be used to control who can sign into the application. To ensure that your application supports both MSA personal accounts and Azure AD work or school accounts, we suggest that you use `common` as the tenant for Bing Ads API authentication. In case your application requires another tenant, see Microsoft identity platform endpoints for more information.

Next steps

Get access and refresh tokens

See Also

Get started

Get access and refresh tokens

8/4/2021 • 8 minutes to read • Edit Online

IMPORTANT

Starting March 1st, 2022 we will requiremulti-factor authenticationforall users who sign in through a third-party application that uses the Bing Ads API, Content API, and Hotel APIs.

You must update your application to get user consent using the new `msads.manage` scope. All application developers must take action to use the new scope.

For more information see the Multi-factor authentication requirement guide.

Once a user has granted consent for you to manage their Microsoft Advertising account, you can redeem the authorization `code` for an access token.

1. Request an access token by redeeming the `code` returned after the user granted consent. Get the *access_token*, *refresh_token*, and *expires_in* values from the JSON response stream.
2. When you received an access token, the value of *expires_in* represents the maximum time in seconds, until the access token will expire. Before the access token expires or before you will need API access again, you should refresh the access token.
3. Once you've successfully acquired an `access_token`, you can use the token in requests to Bing Ads APIs. See the Make your first API call guide for an example.

Here's an example of steps 1 and 2 above.

```
# Replace the Tutorial Sample App ID with your registered application ID.
$clientId = "6731de76-14a6-49ae-97bc-6eba6914391e"

Start-Process "https://login.microsoftonline.com/common/oauth2/v2.0/authorize?
client_id=$clientId&scope=openid%20profile%20https://ads.microsoft.com/msads.manage%20offline_access&respons
e_type=code&redirect_uri=https://login.microsoftonline.com/common/oauth2/nativeclient&state=ClientStateGoesH
ere&prompt=login"

$code = Read-Host "Grant consent in the browser, and then enter the response URI here:"
$code = $code -match 'code=(.*)\&'
$code = $Matches[1]

# Get the initial access and refresh tokens.

$response = Invoke-WebRequest https://login.microsoftonline.com/common/oauth2/v2.0/token -ContentType
application/x-www-form-urlencoded -Method POST -Body
"client_id=$clientId&scope=https://ads.microsoft.com/msads.manage%20offline_access&code=$code&grant_type=aut
horization_code&redirect_uri=https%3A%2F%2Flogin.microsoftonline.com%2Fcommon%2Foauth2%2Fnativeclient"

$oauthTokens = ($response.Content | ConvertFrom-Json)
Write-Output "Access token: " $oauthTokens.access_token
Write-Output "Access token expires in: " $oauthTokens.expires_in
Write-Output "Refresh token: " $oauthTokens.refresh_token

# The access token will expire e.g., after one hour.
# Use the refresh token to get new access and refresh tokens.

$response = Invoke-WebRequest https://login.microsoftonline.com/common/oauth2/v2.0/token -ContentType
application/x-www-form-urlencoded -Method POST -Body
"client_id=$clientId&scope=https://ads.microsoft.com/msads.manage%20offline_access&code=$code&grant_type=ref
resh_token&refresh_token=$($oauthTokens.refresh_token)"

$oauthTokens = ($response.Content | ConvertFrom-Json)
Write-Output "Access token: " $oauthTokens.access_token
Write-Output "Access token expires in: " $oauthTokens.expires_in
Write-Output "Refresh token: " $oauthTokens.refresh_token
```

TIP

For troubleshooting help, see the Common OAuth errors guide.

Access token request details

You can redeem the `code` for an `access_token` to the desired resource. Do this by sending a `POST` request to the `/token` endpoint:

```
// Line breaks for legibility only

POST /{tenant}/oauth2/v2.0/token HTTP/1.1
Host: https://login.microsoftonline.com
Content-Type: application/x-www-form-urlencoded

client_id=6731de76-14a6-49ae-97bc-6eba6914391e
&scope=https%3A%2F%2Fads.microsoft.com%2Fmsads.manage
&code=OAAABAAAAiL9Kn2Z27UubvWFPbm0gLWQJVzCTE9UkP3pSx1aXxUjq3n8b2JRLk4OxVXr...
&redirect_uri=http%3A%2F%2Flocalhost%2Fmyapp%2F
&grant_type=authorization_code
&client_secret=JqQX2PNo9bpM0uEihUPzyrh    // NOTE: Only applicable for web apps
```

The body of the request must include the request parameters and the Content-Type header must be set to *application/x-www-form-urlencoded*. Set the code parameter to the value of the authorization code retrieved in

the previous step, and the grant type set to *authorization_code*. The *redirect_uri* must exactly match the redirect URI used to obtain the authorization code. Be sure to encode the redirect URL. If you registered a web application, include the *client_secret* parameter and set it to the value provisioned in Register an application.

The following table includes parameters that Bing Ads API clients can include in the request for an initial access token. For additional details about optional parameters see the Microsoft identity platform OAuth 2.0 authorization code flow documentation.

PARAMETER	REQUIRED/OPTIONAL	DESCRIPTION
`client_id`	required	The application (client) ID that the Azure portal - App registrations portal assigned your app.
`client_secret`	required for web apps	The application secret that you created in the app registration portal for your app. It should not be used in a native app, because client_secrets cannot be reliably stored on devices. It is required for web apps and web APIs, which have the ability to store the client_secret securely on the server side. The client secret must be URL-encoded before being sent.
`code`	required	The authorization_code that you acquired as a result of requesting user consent.
`code_verifier`	recommended	The same code_verifier that was used to obtain the authorization_code. Required if PKCE was used in the authorization code grant request. For more information, see the PKCE RFC.
`grant_type`	required	Must be `authorization_code` for the authorization code flow.
`redirect_uri`	required	The same redirect_uri value that was used to acquire the authorization_code.
`scope`	required	A space-separated list of scopes. The scopes requested in this leg must be equivalent to or a subset of the scopes included when you requested user consent. If the scopes specified in this request span multiple resource servers, then the Microsoft identity platform endpoint will return a token for the resource specified in the first scope. For a more detailed explanation of scopes, refer to permissions, consent, and scopes.

PARAMETER	REQUIRED/OPTIONAL	DESCRIPTION
`tenant`	required	The `{tenant}` value in the path of the request can be used to control who can sign into the application. To ensure that your application supports both MSA personal accounts and Azure AD work or school accounts, we suggest that you use `common` as the tenant for Bing Ads API authentication. In case your application requires another tenant, see Microsoft identity platform endpoints for more information.

Refresh token request details

Access tokens are short lived, and you must refresh them after they expire to continue accessing resources. You can do so by submitting another `POST` request to the `/token` endpoint, this time providing the `refresh_token` instead of the `code`. Refresh tokens are valid for all permissions that your client has already received consent.

Refresh tokens do not have specified lifetimes. Typically, the lifetimes of refresh tokens are relatively long. However, in some cases, refresh tokens expire, are revoked, or lack sufficient privileges for the desired action. Your application needs to expect and handle errors returned by the token issuance endpoint correctly. For more details about OAuth errors, please see Common OAuth Errors and Authentication and authorization error codes.

```
// Line breaks for legibility only

POST /{tenant}/oauth2/v2.0/token HTTP/1.1
Host: https://login.microsoftonline.com
Content-Type: application/x-www-form-urlencoded

client_id=6731de76-14a6-49ae-97bc-6eba6914391e
&scope=https%3A%2F%2Fads.microsoft.com%2Fmsads.manage
&refresh_token=OAAABAAAAiL9Kn2Z27UubvWFPbm0gLWQJVzCTE9UkP3pSx1aXxUjq...
&grant_type=refresh_token
&client_secret=JqQX2PNo9bpM0uEihUPzyrh      // NOTE: Only applicable for web apps
```

The body of the request must include the request parameters and the Content-Type header must be set to *application/x-www-form-urlencoded*. Set the refresh token parameter to the value of the refresh token retrieved in the previous step, and the grant type set to *refresh_token*. If you registered a web application, include the *client_secret* parameter and set it to the value provisioned in Register an application.

The following table includes parameters that Bing Ads API clients can include in the request to refresh an access token. For additional details about optional parameters see the Microsoft identity platform OAuth 2.0 authorization code flow documentation.

PARAMETER	REQUIRED/OPTIONAL	DESCRIPTION
`client_id`	required	The application (client) ID that the Azure portal - App registrations portal assigned your app.

PARAMETER	REQUIRED/OPTIONAL	DESCRIPTION
`client_secret`	required for web apps	The application secret that you created in the app registration portal for your app. It should not be used in a native app, because client_secrets cannot be reliably stored on devices. It is required for web apps and web APIs, which have the ability to store the client_secret securely on the server side.
`grant_type`	required	Must be set to `refresh_token` for this leg of the authorization code flow.
`refresh_token`	required	The refresh_token that you acquired when you requested an access token.
`scope`	required	A space-separated list of scopes. The scopes requested in this leg must be equivalent to or a subset of the scopes included when you requested user consent. If the scopes specified in this request span multiple resource servers, then the Microsoft identity platform endpoint will return a token for the resource specified in the first scope. For a more detailed explanation of scopes, refer to permissions, consent, and scopes.
`tenant`	required	The `{tenant}` value in the path of the request can be used to control who can sign into the application. To ensure that your application supports both MSA personal accounts and Azure AD work or school accounts, we suggest that you use `common` as the tenant for Bing Ads API authentication. In case your application requires another tenant, see Microsoft identity platform endpoints for more information.

Although refresh tokens are not revoked when used to acquire new access tokens, you are expected to discard the old refresh token. The OAuth 2.0 spec says: "The authorization server MAY issue a new refresh token, in which case the client MUST discard the old refresh token and replace it with the new refresh token. The authorization server MAY revoke the old refresh token after issuing a new refresh token to the client."

Refresh tokens are, and always will be, completely opaque to your application. They are long-lived e.g., 90 days for public clients, but the app should not be written to expect that a refresh token will last for any period of time. Refresh tokens can be invalidated at any moment, and the only way for an app to know if a refresh token is valid is to attempt to redeem it by making a token request. Even if you continuously refresh the token on the same device with the most recent refresh token, you should expect to start again and request user consent if for example, the Microsoft Advertising user changed their password, removed a device from their list of trusted devices, or removed permissions for your application to authenticate on their behalf. At any time without prior warning Microsoft may determine that user consent should again be granted. In that case, the authorization service would return an invalid grant error as shown in the following example.

```
{"error":"invalid_grant","error_description":"The user could not be authenticated or the grant is expired.
The user must first sign in and if needed grant the client application access to the requested scope."}
```

Please keep in mind that public refresh tokens are only bound to the granted device. For example if you registered a Native app and use `https://login.microsoftonline.com/common/oauth2/nativeclient` as the redirect URI, we only guarantee that it can be refreshed on the same device. Clients running apps on services that span regions and devices such as Microsoft Azure should register a Web app with client secret. The redirect URI can be localhost but cannot be `https://login.microsoftonline.com/common/oauth2/nativeclient`. If you use `https://login.microsoftonline.com/common/oauth2/nativeclient` with a client secret the following error would be returned.

```
{"error":"invalid_request","error_description":"Public clients can't send a client secret."} Likewise for
Web apps please note that refresh tokens can be invalidated at any moment.
```

You will encounter the same error if you try to request new access and refresh tokens using a refresh token that was provisioned without a client secret.

For more details about OAuth errors, please see Common OAuth Errors and Authentication and authorization error codes.

Next steps

Make your first API call

See Also

Get started

Make your first API call

7/29/2021 • 6 minutes to read • Edit Online

IMPORTANT

Starting March 1st, 2022 we will requiremulti-factor authenticationforall users who sign in through a third-party application that uses the Bing Ads API, Content API, and Hotel APIs.

You must update your application to get user consent using the new `msads.manage` scope. All application developers must take action to use the new scope.

For more information see the Multi-factor authentication requirement guide.

If you just want to get something working right away, follow these steps to get your Microsoft Advertising user information.

Production Quick Start

To authenticate in the production environment you should first register an application. Otherwise for testing you can use the public "Tutorial Sample App" client ID i.e., **6731de76-14a6-49ae-97bc-6eba6914391e**.

1. Create a new file and paste into it the following script. Set `$clientId` to the Application Id of your registered app. If you registered a web application with client secret, then you'll also need to include `$client_secret=YourWebAppClientSecret` when requesting the access tokens.

```
# Replace the Tutorial Sample App ID with your registered application ID.
$clientId = "6731de76-14a6-49ae-97bc-6eba6914391e"

Start-Process "https://login.microsoftonline.com/common/oauth2/v2.0/authorize?
client_id=$clientId&scope=openid%20profile%20https://ads.microsoft.com/msads.manage%20offline_access&
response_type=code&redirect_uri=https://login.microsoftonline.com/common/oauth2/nativeclient&state=Cl
ientStateGoesHere&prompt=login"

$code = Read-Host "Grant consent in the browser, and then enter the response URI here:"
$code = $code -match 'code=(.*)\&'
$code = $Matches[1]

# Get the initial access and refresh tokens.

$response = Invoke-WebRequest https://login.microsoftonline.com/common/oauth2/v2.0/token -ContentType
application/x-www-form-urlencoded -Method POST -Body
"client_id=$clientId&scope=https://ads.microsoft.com/msads.manage%20offline_access&code=$code&grant_t
ype=authorization_code&redirect_uri=https%3A%2F%2Flogin.microsoftonline.com%2Fcommon%2Foauth2%2Fnativ
eclient"

$oauthTokens = ($response.Content | ConvertFrom-Json)
Write-Output "Access token: " $oauthTokens.access_token
Write-Output "Access token expires in: " $oauthTokens.expires_in
Write-Output "Refresh token: " $oauthTokens.refresh_token

# The access token will expire e.g., after one hour.
# Use the refresh token to get new access and refresh tokens.

$response = Invoke-WebRequest https://login.microsoftonline.com/common/oauth2/v2.0/token -ContentType
application/x-www-form-urlencoded -Method POST -Body
"client_id=$clientId&scope=https://ads.microsoft.com/msads.manage%20offline_access&code=$code&grant_t
ype=refresh_token&refresh_token=$($oauthTokens.refresh_token)"

$oauthTokens = ($response.Content | ConvertFrom-Json)
Write-Output "Access token: " $oauthTokens.access_token
Write-Output "Access token expires in: " $oauthTokens.expires_in
Write-Output "Refresh token: " $oauthTokens.refresh_token
```

Save the file and name it `Get-Tokens-Production.ps1` (you can name it anything you want but the extension must be .ps1).

To programmatically manage a Microsoft Advertising account, you must provide consent at least once through the web application consent flow. From then on you can use the latest refresh token to request new access and refresh tokens without any further user interaction.

2. Now to run `Get-Tokens-Production.ps1` open a console window. At the command prompt, navigate to the folder where you saved `Get-Tokens-Production.ps1` and enter the following command:

```
powershell.exe -File .\Get-Tokens-Production.ps1
```

When the PowerShell script successfully runs, it starts a browser session where you enter your Microsoft Advertising credentials. After consenting, the browser's address bar contains the grant code (see ? code=UseThisCode&...).

```
https://login.microsoftonline.com/common/oauth2/nativeclient?code=M.R4_BAY.f202904c-2269-4daf-1e21-
862ed4d49143
```

Copy the grant code (your own code, not the example M.R4_BAY.f202904c-2269-4daf-1e21-862ed4d49143) and enter it in the console window at the prompt. The PowerShell script then returns the access and refresh tokens. (The script makes a second call to Invoke-WebRequest as an example of how

to refresh the tokens.) You should treat the refresh token like you would a password; if someone gets hold of it, they have access to your resources. The refresh token is long lived but it can become invalid. If you ever receive an invalid_grant error, your refresh token is no longer valid and you'll need to run the `Get-Tokens-Production.ps1` PowerShell script again to get user consent and a new refresh token.

3. Create a new file and paste into it the following script. Set the `accessToken` to the value you received from `Get-Tokens-Production.ps1` and set `$developerToken` to the developer token you received from Step 1 above.

```
$accessToken = "AccessTokenGoesHere";
$developerToken = "DeveloperTokenGoesHere";

[xml]$getUserRequest =
'<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:v13="https://bingads.microsoft.com/Customer/v13">
<soapenv:Header>
   <v13:DeveloperToken>{0}</v13:DeveloperToken>
   <v13:AuthenticationToken>{1}</v13:AuthenticationToken>
</soapenv:Header>
<soapenv:Body>
   <v13:GetUserRequest>
      <v13:UserId xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/>
   </v13:GetUserRequest>
</soapenv:Body>
</soapenv:Envelope>' -f $developerToken, $accessToken

$headers = @{"SOAPAction" = "GetUser"}

$uri =
"https://clientcenter.api.bingads.microsoft.com/Api/CustomerManagement/v13/CustomerManagementService.
svc"
$response = Invoke-WebRequest $uri -Method post -ContentType 'text/xml' -Body $getUserRequest -
Headers $headers
Write-Output $response.Content
```

   Save the file and name it `Get-User.ps1` (you can name it anything you want but the extension must be .ps1).

4. Now to run `Get-User.ps1` open a console window. At the command prompt, navigate to the folder where you saved `Get-User.ps1` and enter the following command:

```
powershell.exe -File .\Get-User.ps1
```

   When the PowerShell script successfully runs it should print out the details of your Microsoft Advertising user, including customer roles. For details see GetUser.

Sandbox Quick Start

To authenticate in the sandbox environment you don't need to register an application. Just use the public "Tutorial Sample App" client ID i.e., **4c0b021c-00c3-4508-838f-d3127e8167ff**.

1. Sign up for a Microsoft Advertising sandbox account. The Microsoft account (MSA) email address must be outlook-**int**.com (for example, someone@outlook-int.com). For more details see Sandbox.

2. Create a new file and paste into it the following script.

```
# Replace the Tutorial Sample App ID with your registered application ID.
$clientId = "4c0b021c-00c3-4508-838f-d3127e8167ff"

Start-Process "https://login.windows-ppe.net/consumers/oauth2/v2.0/authorize?
client_id=$clientId&scope=openid%20profile%20https://api.ads.microsoft.com/msads.manage%20offline_acc
ess&response_type=code&redirect_uri=https://login.windows-
ppe.net/common/oauth2/nativeclient&state=ClientStateGoesHere&prompt=login"

$code = Read-Host "Grant consent in the browser, and then enter the response URI here:"
$code = $code -match 'code=(.*)\&'
$code = $Matches[1]

# Get the initial access and refresh tokens.

$response = Invoke-WebRequest https://login.windows-ppe.net/consumers/oauth2/v2.0/token -ContentType
application/x-www-form-urlencoded -Method POST -Body
"client_id=$clientId&scope=https://api.ads.microsoft.com/msads.manage%20offline_access&code=$code&gra
nt_type=authorization_code&redirect_uri=https://login.windows-ppe.net/common/oauth2/nativeclient"

$oauthTokens = ($response.Content | ConvertFrom-Json)
Write-Output "Access token: " $oauthTokens.access_token
Write-Output "Access token expires in: " $oauthTokens.expires_in
Write-Output "Refresh token: " $oauthTokens.refresh_token

# The access token will expire e.g., after one hour.
# Use the refresh token to get new access and refresh tokens.

$response = Invoke-WebRequest https://login.windows-ppe.net/consumers/oauth2/v2.0/token -ContentType
application/x-www-form-urlencoded -Method POST -Body
"client_id=$clientId&scope=https://api.ads.microsoft.com/msads.manage%20offline_access&code=$code&gra
nt_type=refresh_token&refresh_token=$($oauthTokens.refresh_token)"

$oauthTokens = ($response.Content | ConvertFrom-Json)
Write-Output "Access token: " $oauthTokens.access_token
Write-Output "Access token expires in: " $oauthTokens.expires_in
Write-Output "Refresh token: " $oauthTokens.refresh_token
```

Save the file and name it `Get-Tokens-Sandbox.ps1` (you can name it anything you want but the extension must be .ps1).

A user must provide consent at least once through the web application consent flow. From then on you can use the latest refresh token to request new access and refresh tokens without any further user interaction.

3. Now to run `Get-Tokens-Sandbox.ps1` open a console window. At the command prompt, navigate to the folder where you saved `Get-Tokens-Sandbox.ps1` and enter the following command:

```
powershell.exe -File .\Get-Tokens-Sandbox.ps1
```

When the PowerShell script successfully runs, it starts a browser session where you enter your Microsoft Advertising credentials. After consenting, the browser's address bar contains the grant code (see ? code=UseThisCode&...).

```
https://login.windows-ppe.net/common/oauth2/nativeclient?code=M.R0_CD1.132de532-5105-7550-b1fd-
d37f9af2f009
```

Copy the grant code (your own code, not the example M.R0_CD1.132de532-5105-7550-b1fd-d37f9af2f009) and enter it in the console window at the prompt. The PowerShell script then returns the access and refresh tokens. (The script makes a second call to Invoke-WebRequest as an example of how to refresh the tokens.) You should treat the refresh token like you would a password; if someone gets

hold of it, they have access to your resources. The refresh token is long lived but it can become invalid. If you ever receive an invalid_grant error, your refresh token is no longer valid and you'll need to run the `Get-Tokens-Sandbox.ps1` PowerShell script again to get user consent and a new refresh token.

4. Create a new file and paste into it the following script. Set the `accessToken` to the value you received from `Get-Tokens-Sandbox.ps1`.

```
$accessToken = "AccessTokenGoesHere";
$developerToken = "BBD37VB98";

[xml]$getUserRequest =
'<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:v13="https://bingads.microsoft.com/Customer/v13">
<soapenv:Header>
   <v13:DeveloperToken>{0}</v13:DeveloperToken>
   <v13:AuthenticationToken>{1}</v13:AuthenticationToken>
</soapenv:Header>
<soapenv:Body>
   <v13:GetUserRequest>
      <v13:UserId xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:nil="true"/>
   </v13:GetUserRequest>
</soapenv:Body>
</soapenv:Envelope>' -f $developerToken, $accessToken

$headers = @{"SOAPAction" = "GetUser"}

$uri =
"https://clientcenter.api.sandbox.bingads.microsoft.com/Api/CustomerManagement/v13/CustomerManagement
Service.svc"
$response = Invoke-WebRequest $uri -Method post -ContentType 'text/xml' -Body $getUserRequest -
Headers $headers
Write-Output $response.Content
```

   Save the file and name it `Get-User.ps1` (you can name it anything you want but the extension must be .ps1).

5. Now to run `Get-User.ps1` open a console window. At the command prompt, navigate to the folder where you saved `Get-User.ps1` and enter the following command:

```
powershell.exe -File .\Get-User.ps1
```

   When the PowerShell script successfully runs it should print out the details of your Microsoft Advertising user, including customer roles. For details see GetUser.

See Also

Get started

Multi-factor authentication requirement

8/4/2021 • 4 minutes to read • Edit Online

We already require multi-factor authentication in Microsoft Advertising online. Multi-factor authentication is a security process that requires you to verify your identity in two different ways.

> **IMPORTANT**
>
> Starting March 1st, 2022 we will requiremulti-factor authenticationforall users who sign in through a third-party application that uses the Bing Ads API, Content API, and Hotel APIs.
>
> You must update your application to get user consent using the new `msads.manage` scope. All application developers must take action to use the new scope.

The new `msads.manage` scope **requires renewed consent from all users of your application**. You must prompt users for consent using the new `msads.manage` scope whether or not they have turned on multi-factor authentication. This ensures that they provide a second form of identification or proof when granting consent to your application.

Action required

You must update your application and prompt users for consent using `msads.manage` scope via the Microsoft identity platform endpoint. All Microsoft Advertising developers must take action to use the new scope.

With each API request we will check the access token to ensure the user granted consent via the new `msads.manage` scope. Upon enforcement of multi-factor authentication, any access token provisioned otherwise won't be accepted.

We recommend that you make the necessary changes as soon as possible.

We also recommend that you inform and guide users of your application to set up MFA so that a second proof will be required when they grant permissions for any application. For the Microsoft Advertising requirement, it isn't enough for them to turn on MFA. Either way you must get user consent by prompting with the `msads.manage` scope.

After enforcement

Upon enforcement of MFA, we will only authenticate access tokens on behalf of a user who granted consent to your application via the `msads.manage` scope on the **Microsoft identity platform endpoint**.

- Prior to MFA enforcement the **Microsoft identity platform endpoint** supports the `ads.manage` scope. Access tokens that you acquire for users via the ads.manage scope will no longer be accepted.
- Prior to MFA enforcement the Live Connect endpoint supports the `bingads.manage` scope. The **Live Connect** endpoint is already deprecated and will no longer be supported. Access tokens that you acquire for users via the `bingads.manage` scope will no longer be accepted.

SDK support

Support for the new `msads.manage scope` is available starting with version 13.0.10 of Bing Ads SDKs (.NET, Java, Python, and PHP).

The new `msads.manage` scope is used by default. For backwards compatibility, until the enforcement date you can use the `ads.manage` or `bingads.manage` scopes with a short workaround.

```
var oAuthDesktopMobileAuthCodeGrant = new OAuthDesktopMobileAuthCodeGrant(
    Settings.Default["ClientId"].ToString(),
    apiEnvironment,
    OAuthScope.ADS_MANAGE // temporary workaround; remove or use MSADS_MANAGE instead
);
```

```
OAuthDesktopMobileAuthCodeGrant oAuthDesktopMobileAuthCodeGrant = new OAuthDesktopMobileAuthCodeGrant(
    ClientId,
    ApiEnvironment,
    OAuthScope.ADS_MANAGE // temporary workaround; remove or use MSADS_MANAGE instead
);
```

```
$authentication = (new OAuthDesktopMobileAuthCodeGrant())
    ->withClientId(ClientId)
    ->withEnvironment(ApiEnvironment)
    ->withOAuthScope(OAuthScope::ADS_MANAGE); // temporary workaround; remove or use MSADS_MANAGE instead
```

```
oauth_web_auth_code_grant = OAuthDesktopMobileAuthCodeGrant(
    client_id=CLIENT_ID,
    env=ENVIRONMENT,
    oauth_scope="ads.manage" # temporary workaround; remove or use "msads.manage" instead
)
```

Example scenarios

Here are example scenarios that might apply to your business.

Example: Get a new access token by refreshing with a different scope

An access token represents permissions by a user to act on their behalf with limited permissions based on scopes. When you request consent to manage their accounts you set the scope parameter to either **ads.manage** and **msads.manage**. You are really asking for a user access token that has permissions for whatever is defined by the scope.

> **IMPORTANT**
> Upon enforcement of multi-factor authentication, an access token will only be accepted if it was provisioned or refreshed via the **msads.manage** scope. You can continue refreshing the tokens via **ads.manage**, but the Bing Ads API will not accept them.

To confirm that an access token will be accepted upon enforcement of multi-factor authentication, you can check the response scope. If the scope includes **msads.manage** then it will be accepted.

Let's say for example, that currently a user consents for your application to manage their accounts via both **ads.manage** and **msads.manage** scopes. They may have granted consent via **ads.manage** last month and then granted consent via **msads.manage** this month.

If you refresh the token with **ads.manage** the token refresh response will include the **ads.manage** scope. Upon enforcement of multi-factor authentication, "MyAccessToken-1" would not be accepted.

```
{
    "token_type":"Bearer",
    "scope":"https://ads.microsoft.com/ads.manage",
    "expires_in":3600,
    "ext_expires_in":3600,
    "access_token":"MyAccessToken-1",
    "refresh_token":"MyRefreshToken-1"
}
```

If you refresh the token with **msads.manage** the token refresh response will include both **ads.manage** and **msads.manage** scopes. Upon enforcement of multi-factor authentication, "MyAccessToken-2" would be accepted.

```
{
    "token_type":"Bearer",
    "scope":"https://ads.microsoft.com/msads.manage https://ads.microsoft.com/ads.manage",
    "expires_in":3600,
    "ext_expires_in":3600,
    "access_token":"MyAccessToken-2",
    "refresh_token":"MyRefreshToken-2"
}
```

An **invalid_grant** error will be returned if you attempt to refresh the token using any scope where the user does not currently provide consent.

```
{
    "error":"invalid_grant",
    "error_description":"AADSTS70000: The request was denied because one or more scopes requested are
unauthorized or expired. The user must first sign in and grant the client application access to the
requested scope."
}
```

See Also

OAuth FAQ Request user consent