Use managed identities on a virtual machine to acquire access token - Managed identities for Azure resources

By kengaderdus

Archived: 2026-04-06 15:16:28 UTC

Managed identities for Azure resources provide Azure services with an automatically managed identity in Microsoft Entra ID. You can use this identity to authenticate to any service that supports Microsoft Entra authentication, without having credentials in your code.

This article provides various code and script examples for token acquisition. It also contains guidance about handling token expiration and HTTP errors.

- If you're not familiar with the managed identities for Azure resources feature, see this overview. If you don't have an Azure account, sign up for a free account before you continue.

If you plan to use the Azure PowerShell examples in this article, be sure to install the latest version of Azure PowerShell.

Important

- All sample code/script in this article assumes the client is running on a virtual machine with managed identities for Azure resources. Use the virtual machine "Connect" feature in the Azure portal, to remotely connect to your VM. For details on enabling managed identities for Azure resources on a VM, see Configure managed identities for Azure resources on a VM using the Azure portal, or one of the variant articles (using PowerShell, CLI, a template, or an Azure SDK).

Important

- The security boundary of managed identities for Azure resources is the resource where the identity is used. All code/scripts running on a virtual machine can request and retrieve tokens for any managed identities available on it.

A client application can request a managed identity app-only access token to access a given resource. The token is based on the managed identities for Azure resources service principal. As such, there's no need for the client to obtain an access token under its own service principal. The token is suitable for use as a bearer token in service-to-service calls requiring client credentials.

Link	Description
Get a token using HTTP	Protocol details for managed identities for Azure resources token endpoint

Link	Description
Get a token using Azure.Identity	Example of using managed identities for Azure resources REST endpoint from a C# client using Azure.Identity
Get a token using C#	Example of using managed identities for Azure resources REST endpoint from a C# client using HttpClient
Get a token using Java	Example of using managed identities for Azure resources REST endpoint from a Java client
Get a token using Go	Example of using managed identities for Azure resources REST endpoint from a Go client
Get a token using PowerShell	Example of using managed identities for Azure resources REST endpoint from a PowerShell client
Get a token using CURL	Example of using managed identities for Azure resources REST endpoint from a Bash/CURL client
Handling token caching	Guidance for handling expired access tokens
Error handling	Guidance for handling HTTP errors returned from the managed identities for Azure resources token endpoint
Resource IDs for Azure services	Where to get resource IDs for supported Azure services

The fundamental interface for acquiring an access token is based on REST, making it accessible to any client application running on the VM that can make HTTP REST calls. This approach is similar to the Microsoft Entra programming model, except the client uses an endpoint on the virtual machine (vs a Microsoft Entra endpoint).

Sample request using the Azure Instance Metadata Service (IMDS) endpoint *(recommended)*:

```
GET 'http://169.254.169.254/metadata/identity/oauth2/token?api-version=2018-02-01&resource=https://management.a
```

Element	Description
`GET`	The HTTP verb, indicating you want to retrieve data from the endpoint. In this case, an OAuth access token.
`http://169.254.169.254/metadata/identity/oauth2/token`	The managed identities for Azure resources endpoint for the Instance Metadata Service.
`api-version`	A query string parameter, indicating the API version for the IMDS endpoint. Use API

Element	Description
	version `2018-02-01` or greater.
`resource`	A query string parameter, indicating the App ID URI of the target resource. It also appears in the `aud` (audience) claim of the issued token. This example requests a token to access Azure Resource Manager, which has an App ID URI of `https://management.azure.com/`.
`Metadata`	An HTTP request header field required by managed identities. This information is used as a mitigation against server side request forgery (SSRF) attacks. This value must be set to "true", in all lower case.
`object_id`	(Optional) A query string parameter, indicating the object_id of the managed identity you would like the token for. Required, if your VM has multiple user-assigned managed identities.
`client_id`	(Optional) A query string parameter, indicating the client_id of the managed identity you would like the token for. Required, if your VM has multiple user-assigned managed identities.
`msi_res_id`	(Optional) A query string parameter, indicating the msi_res_id (Azure Resource ID) of the managed identity you would like the token for. Required, if your VM has multiple user-assigned managed identities.

Sample response:

```
 HTTP/1.1 200 OK
Content-Type: application/json
{
  "access_token": "eyJ0eXAi...",
  "refresh_token": "",
  "expires_in": "3599",
  "expires_on": "1506484173",
```

```
  "not_before": "1506480273",
  "resource": "https://management.azure.com/",
  "token_type": "Bearer"
}
```

Element	Description
`access_token`	The requested access token. When you call a secured REST API, the token is embedded in the `Authorization` request header field as a "bearer" token, allowing the API to authenticate the caller.
`refresh_token`	Not used by managed identities for Azure resources.
`expires_in`	The number of seconds the access token continues to be valid, before expiring, from time of issuance. Time of issuance can be found in the token's `iat` claim.
`expires_on`	The timespan when the access token expires. The date is represented as the number of seconds from "1970-01-01T0:0:0Z UTC" (corresponds to the token's `exp` claim).
`not_before`	The timespan when the access token takes effect, and can be accepted. The date is represented as the number of seconds from "1970-01-01T0:0:0Z UTC" (corresponds to the token's `nbf` claim).
`resource`	The resource the access token was requested for, which matches the `resource` query string parameter of the request.
`token_type`	The type of token, which is a "Bearer" access token, which means the resource can give access to the bearer of this token.

Using the Azure Identity client library is the recommended way to use managed identities. Complete the following steps:

1. Install the Azure.Identity package and other required Azure SDK library packages, such as Azure.Security.KeyVault.Secrets.

2. Use the sample code below. You don't need to worry about getting tokens. You can directly use the Azure SDK clients. The code is for demonstrating how to get the token, if you need to.

```
 using Azure.Core;
using Azure.Identity;
using Azure.Security.KeyVault.Secrets;

ManagedIdentityCredential credential = new(
    ManagedIdentityId.FromUserAssignedClientId("<managed_identity_client_ID>"));

// Option 1: Explicit token acquisition. Manually fetch the token and convert to a string, if necessary.
AccessToken accessToken = await credential.GetTokenAsync(
```

```
    new TokenRequestContext(["https://vault.azure.net"]));
string accessTokenString = accessToken.Token;

// Option 2: Implicit token acquisition. Pass the credential object to the Azure service client construct
// Token acquisition is triggered on the GetSecretAsync method call.
SecretClient client = new(new Uri("https://myvault.vault.azure.net/"), credential);
KeyVaultSecret secret = await client.GetSecretAsync("MySecret");
```

For more information, see Use a user-assigned managed identity and Use a system-assigned managed identity.

```
 using System;
using System.Net.Http;
using Newtonsoft.Json.Linq;

// Construct HttpClient
var httpClient = new HttpClient
{
    DefaultRequestHeaders =
    {
        { "Metadata", Boolean.TrueString }
    }
};

// Construct URI to call
var resource = "https://management.azure.com/";
var uri = $"http://169.254.169.254/metadata/identity/oauth2/token?api-version=2018-02-01&resource={resource}";

// Make call
var response = await httpClient.GetAsync(uri);
try
{
    response.EnsureSuccessStatusCode();
}
catch (HttpRequestException)
{
    var error = await response.Content.ReadAsStringAsync();
    Console.WriteLine(error);
    throw;
}

// Parse response using Newtonsoft.Json
var content = await response.Content.ReadAsStringAsync();
var obj = JObject.Parse(content);
var accessToken = obj["access_token"];

Console.WriteLine(accessToken);
```

Use this JSON library to retrieve a token using Java.

```
 import java.io.*;
import java.net.*;
import com.fasterxml.jackson.core.*;

class GetMSIToken {
    public static void main(String[] args) throws Exception {

        URL msiEndpoint = new URL("http://169.254.169.254/metadata/identity/oauth2/token?api-version=2018-02-01
        HttpURLConnection con = (HttpURLConnection) msiEndpoint.openConnection();
        con.setRequestMethod("GET");
        con.setRequestProperty("Metadata", "true");

        if (con.getResponseCode()!=200) {
            throw new Exception("Error calling managed identity token endpoint.");
        }

        InputStream responseStream = con.getInputStream();

        JsonFactory factory = new JsonFactory();
        JsonParser parser = factory.createParser(responseStream);

        while(!parser.isClosed()){
            JsonToken jsonToken = parser.nextToken();

            if(JsonToken.FIELD_NAME.equals(jsonToken)){
                String fieldName = parser.getCurrentName();
                jsonToken = parser.nextToken();

                if("access_token".equals(fieldName)){
                    String accesstoken = parser.getValueAsString();
                    System.out.println("Access Token: " + accesstoken.substring(0,5)+ "..." + accesstoken.substr
                    return;
                }
            }
        }
    }
}
```

```
 package main

import (
  "fmt"
  "io/ioutil"
  "net/http"
```

```
  "net/url"
  "encoding/json"
)

type responseJson struct {
  AccessToken string `json:"access_token"`
  RefreshToken string `json:"refresh_token"`
  ExpiresIn string `json:"expires_in"`
  ExpiresOn string `json:"expires_on"`
  NotBefore string `json:"not_before"`
  Resource string `json:"resource"`
  TokenType string `json:"token_type"`
}

func main() {

    // Create HTTP request for a managed services for Azure resources token to access Azure Resource Manager
    var msi_endpoint *url.URL
    msi_endpoint, err := url.Parse("http://169.254.169.254/metadata/identity/oauth2/token?api-version=2018-02-01
    if err != nil {
      fmt.Println("Error creating URL: ", err)
      return
    }
    msi_parameters := msi_endpoint.Query()
    msi_parameters.Add("resource", "https://management.azure.com/")
    msi_endpoint.RawQuery = msi_parameters.Encode()
    req, err := http.NewRequest("GET", msi_endpoint.String(), nil)
    if err != nil {
      fmt.Println("Error creating HTTP request: ", err)
      return
    }
    req.Header.Add("Metadata", "true")

    // Call managed services for Azure resources token endpoint
    client := &http.Client{}
    resp, err := client.Do(req)
    if err != nil{
      fmt.Println("Error calling token endpoint: ", err)
      return
    }

    // Pull out response body
    responseBytes,err := ioutil.ReadAll(resp.Body)
    defer resp.Body.Close()
    if err != nil {
      fmt.Println("Error reading response body : ", err)
      return
```

```
    }

    // Unmarshall response body into struct
    var r responseJson
    err = json.Unmarshal(responseBytes, &r)
    if err != nil {
      fmt.Println("Error unmarshalling the response:", err)
      return
    }

    // Print HTTP response and marshalled response body elements to console
    fmt.Println("Response status:", resp.Status)
    fmt.Println("access_token: ", r.AccessToken)
    fmt.Println("refresh_token: ", r.RefreshToken)
    fmt.Println("expires_in: ", r.ExpiresIn)
    fmt.Println("expires_on: ", r.ExpiresOn)
    fmt.Println("not_before: ", r.NotBefore)
    fmt.Println("resource: ", r.Resource)
    fmt.Println("token_type: ", r.TokenType)
}
```

The following example demonstrates how to use the managed identities for Azure resources REST endpoint from a PowerShell client to:

1. Acquire an access token.
2. Use the access token to call an Azure Resource Manager REST API and get information about the VM. Be sure to substitute your subscription ID, resource group name, and virtual machine name for `<SUBSCRIPTION-ID>`, `<RESOURCE-GROUP>`, and `<VM-NAME>`, respectively.

```
Invoke-RestMethod -Method GET -Uri 'http://169.254.169.254/metadata/identity/oauth2/token?api-version=2018-02-
```

Example of how to parse the access token from the response:

```
 # Get an access token for managed identities for Azure resources
$resource = 'https://management.azure.com'
$response = Invoke-RestMethod -Method GET `
                              -Uri "http://169.254.169.254/metadata/identity/oauth2/token?api-version=2018-02-01&
                              -Headers @{ Metadata="true" }
$accessToken = $response.access_token
Write-Host "Access token using a User-Assigned Managed Identity is $accessToken"

# Use the access token to get resource information for the VM
$secureToken = $accessToken | ConvertTo-SecureString -AsPlainText
$vmInfoRest = Invoke-RestMethod -Method GET `
                                -Uri 'https://management.azure.com/subscriptions/<SUBSCRIPTION-ID>/resourceGroups/
```

```
                                -ContentType 'application/json' `
                                -Authentication Bearer `
                                -Token $secureToken
Write-Host "JSON returned from call to get VM info: $($vmInfoRest.content)"
```

```
curl 'http://169.254.169.254/metadata/identity/oauth2/token?api-version=2018-02-01&resource=https%3A%2F%2Fmanag
```

Example of how to parse the access token from the response:

```
response=$(curl 'http://169.254.169.254/metadata/identity/oauth2/token?api-version=2018-02-01&resource=https%3A
access_token=$(echo $response | python -c 'import sys, json; print (json.load(sys.stdin)["access_token"])')
echo Access token using a User-Assigned Managed Identity is $access_token
```

The managed identities subsystem caches tokens but we still recommend that you implement token caching in your code. You should prepare for scenarios where the resource indicates that the token is expired.

On-the-wire calls to Microsoft Entra ID result only when:

- Cache miss occurs due to no token in the managed identities for Azure resources subsystem cache.
- The cached token is expired.

The managed identities endpoint signals errors via the status code field of the HTTP response message header, as either 4xx or 5xx errors:

Status Code	Error Reason	How To Handle
404 Not found.	IMDS endpoint is updating.	Retry with Exponential Backoff. See guidance below.
410	IMDS is going through updates	IMDS will be available within 70 seconds
429 Too many requests.	IMDS Throttle limit reached.	Retry with Exponential Backoff. See guidance below.
4xx Error in request.	One or more of the request parameters was incorrect.	Don't retry. Examine the error details for more information. 4xx errors are design-time errors.
5xx Transient error from service.	The managed identities for Azure resources subsystem or Microsoft Entra ID returned a transient error.	It's safe to retry after waiting for at least 1 second. If you retry too quickly or too often, IMDS and/or Microsoft Entra ID may return a rate limit error (429).
time out	IMDS endpoint is updating.	Retry with Exponential Backoff. See guidance later.

If an error occurs, the corresponding HTTP response body contains JSON with the error details:

Element	Description
error	Error identifier.
error_description	Verbose description of error. **Error descriptions can change at any time. Do not write code that branches based on values in the error description.**

This section documents the possible error responses. A "200 OK" status is a successful response, and the access token is contained in the response body JSON, in the access_token element.

Status code	Error	Error Description	Solution
400 Bad Request	invalid_resource	AADSTS50001: The application named <*URI*> wasn't found in the tenant named <*TENANT-ID*>. This message shows if the tenant administrator hasn't installed the application or no tenant user consented to it. You might have sent your authentication request to the wrong tenant.\	(Linux only)
400 Bad Request	bad_request_102	Required metadata header not specified	Either the `Metadata` request header field is missing from your request, or is formatted incorrectly. The value must be specified as `true`, in all lower case. See the "Sample request" in the preceding REST section for an example.

Status code	Error	Error Description	Solution
401 Unauthorized	unknown_source	Unknown Source *<URI>*	Verify that your HTTP GET request URI is formatted correctly. The `scheme:host/resource-path` portion must be specified as `http://localhost:50342/oauth2/token`. See the "Sample request" in the preceding REST section for an example.
	invalid_request	The request is missing a required parameter, includes an invalid parameter value, includes a parameter more than once, or is otherwise malformed.	
	unauthorized_client	The client isn't authorized to request an access token using this method.	Caused by a request on a VM that doesn't have managed identities for Azure resources configured correctly. See Configure managed identities for Azure resources on a VM using the Azure portal if you need assistance with VM configuration.
	access_denied	The resource owner or authorization server denied the request.	
	unsupported_response_type	The authorization server doesn't support obtaining an access token	

Status code	Error	Error Description	Solution
		using this method.	
	invalid_scope	The requested scope is invalid, unknown, or malformed.	
500 Internal server error	unknown	Failed to retrieve token from the Active directory. For details see logs in *<file path>*	Verify that the VM has managed identities for Azure resources enabled. See Configure managed identities for Azure resources on a VM using the Azure portal if you need assistance with VM configuration. Also verify that your HTTP GET request URI is formatted correctly, particularly the resource URI specified in the query string. See the "Sample request" in the preceding REST section for an example, or Azure services that support Microsoft Entra authentication for a list of services and their respective resource IDs.

Important

- IMDS isn't intended to be used behind a proxy and doing so is unsupported. For examples of how to bypass proxies, refer to the Azure Instance Metadata Samples.

Retry if you receive a 404, 429, or 5xx error code (see Error handling). If you receive a 410 error, it indicates that IMDS is going through updates and will be available in a maximum of 70 seconds.

Throttling limits apply to the number of calls made to the IMDS endpoint. When the throttling threshold is exceeded, IMDS endpoint limits any further requests while the throttle is in effect. During this period, the IMDS endpoint returns the HTTP status code 429 ("Too many requests"), and the requests fail.

For retry, we recommend the following strategy:

Retry strategy	Settings	Values	How it works
ExponentialBackoff	Retry count Min back-off	5 0 sec	Attempt 1 - delay 0 sec Attempt 2 - delay ~2 sec

Retry strategy	Settings	Values	How it works
	Max back-off Delta back-off First fast retry	60 sec 2 sec false	Attempt 3 - delay ~6 sec Attempt 4 - delay ~14 sec Attempt 5 - delay ~30 sec

See Azure Services with managed identities support for a list of resources that support managed identities for Azure resources.

- To enable managed identities for Azure resources on an Azure VM, see Configure managed identities for Azure resources on a VM using the Azure portal.
