Azure AD Graph API Operations Overview

By JimacoMS

Archived: 2026-04-06 01:12:36 UTC

The Azure Active Directory (AD) Graph API is an OData 3.0 compliant service that you can use to read and modify objects such as users, groups, and contacts in a tenant. Azure AD Graph API exposes REST endpoints that you send HTTP requests to in order to perform operations using the service. The following sections provide general information about how to format requests and what to expect in responses when you use the Graph API to read and write directory resources, call directory functions or actions, or perform queries against the directory. For more detailed information about performing specific operations directory resources, see the appropriate operations topic in the Azure AD Graph API reference.

This article applies to Azure AD Graph API. For similar info related to Microsoft Graph API, see Microsoft Graph overview.

A successful request to the Graph API must be addressed to a valid endpoint and be well-formatted, that is, it must contain any required headers and, if necessary, a JSON payload in the request body. The app making the request must include a token received from Azure AD that proves that it is authorized to access the resources targeted by the request. The app must be able to handle any responses received from the Graph API.

The sections in this topic will help you understand the format of requests and responses used with the Graph API.

Every request to the Graph API must have a bearer token issued by Azure Active Directory attached. The token carries information about your app, the signed-in user (in the case of delegated permissions), authentication, and the operations on the directory that your app is authorized to perform. This token is carried in the **Authorization** header of the request. For example (the token has been shortened for brevity):

```
Authorization: Bearer eyJ0eX ... FWSXfwtQ
```

The Graph API performs authorization based on OAuth 2.0 permission scopes present in the token. For more information about the permission scopes that the Graph API exposes, see Graph API Permission Scopes.

In order for your app to authenticate with Azure AD and call the Graph API, you must add it to your tenant and configure it to require permissions (OAuth 2.0 permission scopes) for Windows Azure Active Directory. For information about adding and configuring an app, see Integrating Applications with Azure Active Directory.

Azure AD uses the OAuth 2.0 authentication protocol. You can learn more about OAuth 2.0 in Azure AD, including supported flows and access tokens in OAuth 2.0 in Azure AD.

To perform operations with the Graph API, you send HTTP requests with a supported method - typically GET, POST, PATCH, PUT, or DELETE -- to an endpoint that targets the service, a resource collection, an individual resource, a navigation property of a resource, or a function or action exposed by the service. Endpoints are expressed as URLs.

The following describes the basic format of a Graph API endpoint:

```
https://graph.windows.net/{tenant_id}/{resource_path}?{api_version}
```

The following components comprise the URL:

- **Service Root**: The service root for all Graph API requests is `https://graph.windows.net`.
- **Tenant Identifier {tenant_id}**: The identifier for the tenant that the request targets.
- **Resource path {resource_path}**: The path to the resource -- for example, a user or a group -- that the request targets.
- **Graph API Version {api_version}**: The version of the Graph API targeted by the request. This is expressed as a query parameter and is required.

Note: In some cases, when reading resource collections, OData query parameters can be added to the request to filter, order, and page the result set. For more information, see the OData query parameters section in this topic.

You can specify the target tenant of a request in one of four ways:

- **By tenant object ID**. The GUID that was assigned when the tenant was created. This can be found in the **objectId** property of the TenantDetail object. The following URL shows how to address the users resource collection by using the tenant object ID:
  `https://graph.windows.net/12345678-9abc-def0-1234-56789abcde/users?api-version=1.6`.

- **By verified (registered) domain name**. One of the domain names that are registered for the tenant. These can be found in the **verifiedDomains** property of the TenantDetail object. The following URL shows how to address the

users resource collection of a tenant that has the domain contoso.com: `https://graph.windows.net/contoso.com/users?api-version=1.6`.

- **By using the `myOrganization` alias**. This alias is only available when using OAuth Authorization Code Grant type (3-legged) authentication; that is, when using a delegated permission scope. The alias is not case sensitive. It replaces the object ID or tenant domain in the URL. When the alias is used, Graph API derives the tenant from the claims presented in the token attached to the request. The following URL shows how to address the users resource collection of a tenant using this alias: `https://graph.windows.net/myorganization/users?api-version=1.6`.
- **By using the `me` alias**. This alias is only available when using OAuth Authorization Code Grant type (3-legged) authentication; that is, when using a delegated permission scope. The alias is not case sensitive. It replaces the object ID or tenant domain in the URL. When the alias is used, Graph API derives the user from the claims presented in the token attached to the request. The following URL to address the signed-in user using this alias: `https://graph.windows.net/me?api-version=1.6`.

Note: You use `me` alias solely to target operations against the signed-in user. For more information, see Signed-in User Operations.

You specify the `{resource_path}` differently depending on whether you are targeting a resource collection, an individual resource, a navigation property of a resource, a function or action exposed on the tenant, or a function or action exposed on a resource.

Target	Path	Explanation
Service Metadata	`/$metadata`	Returns the service metadata document. The following example t using the contoso.com tenant: `https://graph.windows.net/contoso.com/$metadata?api-versi` **Note**: No authentication is necessary to read the service metadata
Resource collection	`/{resource_collection}`	Targets a resource collection, such as users or groups in the tenan read resources in the collection, and, depending on the resource t resource in the tenant. In many cases the result set returned by a r refined by the addition of query parameters to filter, order, or pag following example targets the user collection: `https://graph.windows.net/myorganization/users?api-versio`
Single resource	`/{resource_collection}/{resource_id}`	Targets a specific resource in a tenant, such as a user, organizatio most resources the `resource_id` is the object ID. Some resourc specifiers; for example, users can be also specified by user princi Depending on the resource, you can use this path to get the decla resource, to modify its declared properties, or to delete the resour example targets the user john@contoso.com: `https://graph.windows.net/contoso.com/users/john@contoso.`
Navigation property (objects)	`/{resource_collection}/{resource_id}/{property_name}`	Targets a navigation property of a specific resource, such as a use memberships, or a group's members. You can use this path to retu referenced by the target navigation property. The following exam of john@contoso.com: `https://graph.windows.net/contoso.com/users/john@contoso.` `version=1.6` **Note**: This form of addressing is only available for reads.
Navigation property (links)	`/{resource_collection}/{resource_id}/$links/{property_name}`	Targets a navigation property of a specific resource, such as a use memberships, or a group's members. You can use this form of ad modify a navigation property. On reads, the objects referenced by as one or more links in the response body. On writes, the objects more links in the request body. The following example targets the john@contoso.com: `https://graph.windows.net/contoso.com/users/john@contoso.` `version=1.6`
Functions or actions exposed	`/{function_name}`	Targets a function or action exposed at the tenant. Functions and invoked with a POST Request and may include a request body. T

Target	Path	Explanation
on the tenant		targets the isMemberOf function: `https://graph.windows.net/myorganization/isMemberOf?api-v`
Functions or actions exposed on a resource.	`/{resource_collection}/{resource_id}/{function_name}`	Targets a function or action exposed on a resource. Functions and invoked with a POST Request and may include a request body. T targets the getMemberGroups function: `https://graph.windows.net/myorganization/users/john@conto` `api-version=1.6` .

You use the `api-version` query parameter to target a specific version of the Graph API for an operation. This parameter is required.

The value for the `api-version` parameter can be one of the following:

- "beta"
- "1.6"
- "1.5"
- "2013/11/08"
- "2013/04/05"

For example the following URL targets the user collection using Graph API version 1.6:

```
https://graph.windows.net/myorganization/users?api-version=1.6
```

For more information about versions and feature changes between versions, see Versioning.

In many cases when you read a collection of resources, you can filter, sort, and page the result set by attaching OData query parameters to your request.

The Graph API supports the following Odata query parameters:

- $filter
- $batch
- $expand
- $orderby
- $top
- $skiptoken and previous-page

See Supported Queries, Filters, and Paging Options for more information about supported OData query parameters and their limitations in the Graph API.

The following table shows common HTTP headers used in requests with the Graph API. It is not meant to be comprehensive.

Request Header	Description
Authorization	Required. A bearer token issued by Azure Active Directory. See Authentication and Authorization in this topic for more information.
Content-Type	Required if request body is present. The media type of the content in the request body. Use application/json. Parameters may be included with the media type. **Note**: application/atom+xml and application/xml (for links) are supported but are not recommended both for performance reasons and because support for XML will be deprecated in a future release.
Content-Length	Required if request body is present. The length of the request in bytes.

The following table shows common HTTP headers returned in responses by the Graph API. It is not meant to be comprehensive.

Response Header	Description
Content-Type	The media type of the content in the response body. The default is application/json. Requests for user thumbnail photos return image/jpeg by default.

Response Header	Description
Location	Returned in responses to POST requests that create a new resource (object) in the directory. Contains the URI of the newly created resource.
ocp-aad-diagnostics-server-name	The identifier for the server that performed the requested operation.
ocp-aad-session-key	The key that identifies the current session with the directory service.

At a minimum, we recommend you do the following for each request:

1. Log an accurate time stamp of the request submission.
2. Send and log the `client-request-id` .
3. Log the HTTP response code and `request-id` .

Providing information in such logs will help Microsoft troubleshoot issues when you ask for help or support.

Request bodies for POST, PATCH, and PUT requests can be sent in JSON or XML payloads. Server responses can be returned in JSON or XML payloads. You can specify the payload in request bodies by using the `Content-Type` request header and in responses by using the `Accept` request header.

We strongly recommend that you use JSON payloads in requests and responses with the Graph API. This is both for performance reasons and because XML will be deprecated in a future release.

The preceding sections discussed the formatting of basic requests and responses with the Graph API. More advanced features may add additional query string parameters or have significantly different request and response bodies than the basic operations discussed previously in this topic.

These features include:

- **Batch Processing**: The Graph API supports batching. Sending requests in batches reduces round trips to the server, which reduces network overhead and helps your operations complete more quickly. For more information about batch processing with the Graph API, see Batch Processing.
- **Differential Query**: The Graph API supports performing differential queries. Differential query allows you to return changes to specific entities in a tenant between requests issued at different times. This feature is often used to sync a local store with changes on the tenant. For more information about differential query with the Graph API, see Differential Query.

- Azure AD Graph API reference
