Get Started

This document provides all the basic information you need to start using the library. It covers the library concepts, shows examples for various use cases, and gives links to more information.

Setup

There are a few setup steps you need to complete before you can use this library:

1. If you don't already have a Google account, sign up (https://www.google.com/accounts).
2. If you have never created a Google API Console project, read the Managing Projects page (/console/help/managing-projects) and create a project in the Google API Console (https://console.developers.google.com/).
3. Install (http://www.nuget.org/packages?q=google.apis&prerelease=true&sortOrder=relevance) the NuGet package you want to work with.

Authentication and authorization

It is important to understand the basics of how API authentication and authorization are handled. All API calls must use either simple or authorized access (defined below). Many API methods require authorized access, but some can use either. Some API methods that can use either behave differently, depending on whether you use simple or authorized access. See the API's method documentation to determine the appropriate access type.

1. Simple API access (API keys)

These API calls do not access any private user data. Your application must authenticate itself as an application belonging to your Google API Console project. This is needed to measure project usage for accounting purposes.

API key: To authenticate your application, use an API key (/console/help/using-keys) for your API Console project. Every simple access call your application makes must include this key.

ıg: Keep your API key private. If someone obtains your key, they could use it to consume your quota or incur
:s against your API Console project.

2. Authorized API access (OAuth 2.0)

These API calls access private user data. Before you can call them, the user that has access to the private data must grant your application access. Therefore, your application must be authenticated, the user must grant access for your application, and the user must be authenticated in order to grant that access. All of this is accomplished with OAuth 2.0 (/accounts/docs/OAuth2) and libraries written for it.

Scope: Each API defines one or more scopes that declare a set of operations permitted. For example, an API might have read-only and read-write scopes. When your application requests access to user data, the request must include one or more scopes. The user needs to approve the scope of access your application is requesting.

Refresh and access tokens: When a user grants your application access, the OAuth 2.0 authorization server provides your application with refresh and access tokens. These tokens are only valid for the scope requested. Your application uses access tokens to authorize API calls. Access tokens expire, but refresh tokens do not. Your application can use a refresh token to acquire a new access token.

ıg: Keep refresh and access tokens private. If someone obtains your tokens, they could use them to access p
ata.

Client ID and client secret: These strings uniquely identify your application and are used to acquire tokens. They are created for your project on the API Console (https://console.developers.google.com/). There are three types of client IDs, so be sure to get the correct type for your application:

- Web application (/accounts/docs/OAuth2WebServer) client IDs
- Installed application (/accounts/docs/OAuth2InstalledApp) client IDs
- Service Account (/accounts/docs/OAuth2ServiceAccount) client IDs

ıg: Keep your client secret private. If someone obtains your client secret, they could use it to consume your qu
:harges against your Console project, and request access to user data.

Examples

In this section, there are examples of simple API usage without authorization. For more information about authorization calls, see the OAuth 2.0 page for .NET (/api-client-library/dotnet/guide/aaa_oauth).

Simple API example

This example uses simple API access for a command-line application. It calls the Google Discovery API (/discovery/) to list all Google APIs.

Setup for example

Get your Simple API key. To find your application's API key, do the following:

1. Open the Credentials page (https://console.developers.google.com/apis/credentials) in the API Console.
2. This API supports two types of credentials. Create whichever credentials are appropriate for your project:
    - **OAuth 2.0:** Whenever your application requests private user data, it must send an OAuth 2.0 token along with the request. Your application first sends a client ID and, possibly, a client secret to obtain a token. You can generate OAuth 2.0 credentials for web applications, service accounts, or installed applications.

      For more information, see the OAuth 2.0 documentation (https://developers.google.com/identity/protocols/OAuth2).
    - **API keys:** A request that does not provide an OAuth 2.0 token must send an API key. The key identifies your project and provides API access, quota, and reports.

      The API supports several types of restrictions on API keys. If the API key that you need doesn't already exist, then create an API key in the Console by clicking **Create**

credentials > API key. You can restrict the key before using it in production by clicking **Restrict key** and selecting one of the **Restrictions**.

To keep your API keys secure, follow the best practices for securely using API keys (//cloud.google.com/docs/authentication/api-keys).

Code for example

```
using System;
using System.Threading.Tasks;

using Google.Apis.Discovery.v1;
using Google.Apis.Discovery.v1.Data;
using Google.Apis.Services;

namespace Discovery.ListAPIs
{
    /// <summary>
    /// This example uses the discovery API to list all APIs in the discovery re
    /// https://developers.google.com/discovery/v1/using.
    /// <summary>
    class Program
    {
        [STAThread]
        static void Main(string[] args)
        {
            Console.WriteLine("Discovery API Sample");
            Console.WriteLine("====================");
            try
            {
                new Program().Run().Wait();
            }
            catch (AggregateException ex)
            {
                foreach (var e in ex.InnerExceptions)
                {
                    Console.WriteLine("ERROR: " + e.Message);
                }
            }
            Console.WriteLine("Press any key to continue...");
            Console.ReadKey();
        }
```

```
        private async Task Run()
        {
            // Create the service.
            var service = new DiscoveryService(new BaseClientService.Initializer
                {
                    ApplicationName = "Discovery Sample",
                    ApiKey="[YOUR_API_KEY_HERE]",
                });

            // Run the request.
            Console.WriteLine("Executing a list request...");
            var result = await service.Apis.List().ExecuteAsync();

            // Display the results.
            if (result.Items != null)
            {
                foreach (DirectoryList.ItemsData api in result.Items)
                {
                    Console.WriteLine(api.Id + " - " + api.Title);
                }
            }
        }
    }
}
```

Tips for using API keys:

- In order to use a specific service, you have to add a reference to it. For example if you want to use the Tasks API (https://developers.google.com/google-apps/tasks/), you should install its NuGet package Google.Apis.Tasks.v1 (http://www.nuget.org/packages/Google.Apis.Tasks.v1/).
- To create an instance of a service, just call its constructor. For example: `new TasksService(new BaseClientService.Initializer {...});`".
- All methods of a service are located on individual resources on the service object itself. The Discovery service has an `Apis` resource, which contains a `List` method. When you call `service.Apis.List(..)` a request object targeting this method is returned. To execute a request, call the `Execute()` or `ExecuteAsyc()` method on a request.
- Set the API key using the `ApiKey` property on the `BaseClientService.Initializer` instance.

Finding information about the APIs

The Supported APIs (/api-client-library/dotnet/apis) page lists all APIs that can be accessed using this library as well as links to documentation.

You can also use the APIs Explorer (/apis-explorer) to browse APIs, list available methods, and even try API calls from your browser.