MARCH 20, 2026

GSCCCA IDENTITY SERVER AUTHENTICATION GUIDE

GEORGIA SUPERIOR COURT CLERKS' COOPERATIVE AUTHORITY
https://www.gsccca.org

REVISION HISTORY

Revision	Description	Date
1	Initial Version	03/20/2026

CONTENTS

INFORMATION FOR IMPLEMENTERS

The Georgia Superior Court Clerks' Cooperative Authority (GSCCCA) provides the GSCCCA Identity Server, an OAuth 2.0–based authentication and authorization service. API clients obtain a bearer token through REST API calls and include this token in the header of subsequent requests to securely authenticate with other GSCCCA API services.

ASSUMPTIONS

This is a technical document intended for developers and service providers making use of one or more Clerks' Authority APIs. Anyone reading this document should be well-versed in the OAuth 2.0 protocol, HTTP Request methods, and reading and serializing JSON.

SUPPORT/CONTACT

For all Clerks' Authority Image API related questions, please contact our customer support at (800) 304-5174 or email help@gsccca.org. Customer service will be able to help or escalate any issues or questions to the relevant personnel.

WHERE DO I CONNECT?

The GSCCCA Identity Server provides access to a discovery document. This page returns a JSON response and lists various other pages, scopes, and data used by the API.

ACCOUNT CREATION

To access certain web services for testing or production, users must have a credentialed authorization grant created with the appropriate permissions and requirements. Access to the GSCCCA test website is restricted. Please email help@gsccca.org to request access. When contacting the help desk, please specify the API that you are interested in implementing. You will receive specific instructions related to the API that you wish to implement against.

TEST SERVICE ENDPOINT

Once you have an account, use the public discovery document located at this

address. https://identitystg.gsccca.org/.well-known/openid-configuration

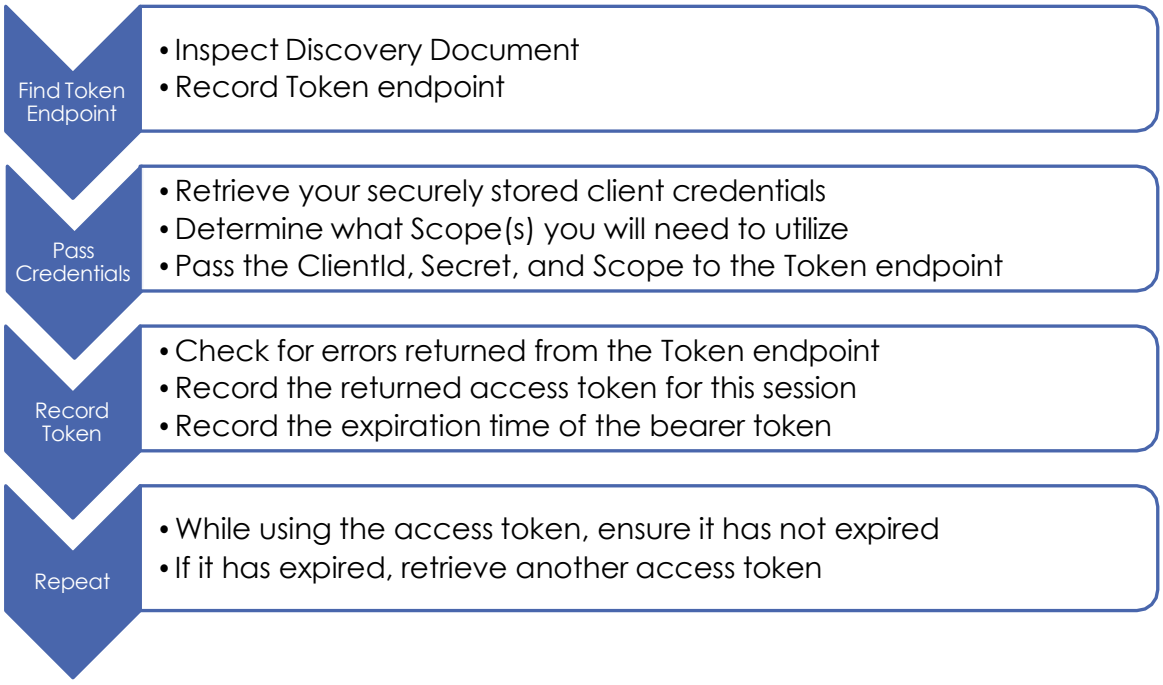
SERVICE ENDPOINT

Once you have an account, use the public discovery document located at this

address. https://identity.gsccca.org/.well-known/openid-configuration

AUTHENTICATION WORKFLOW

This document is not meant to be an instruction set on connecting with and utilizing the OAuth 2.0 protocol. Below is the basic workflow for retrieving and using client credentials grants.



WHAT ARE CLIENTS AND SECRETS?

The client identifier is a unique name representing the client application, defined and assigned by the GSCCCA. It is not a username for an individual to log into a system, but instead a public identifier used to retrieve an authorization grant. The client identifier should be transparent to any users of your system – they never need to know it exists.

The client secret is a cryptographically random string assigned to a client identifier when the account is created. This must be kept as securely as possible on the client's end. If it is ever made available to anyone inadvertently, others will be able to impersonate the client account and access any secured API to which the client has access. In that event, immediately contact the GSCCCA to revoke that secret and issue a new one.

WHAT IS AN ACCESS TOKEN?

The Clerks' Authority credentialed authorization grant generates a JSON Web Token (JWT) in response to authentication requests from the Token endpoint. The JWT is an OAuth 2.0 access token containing various information necessary to authorize access to other Clerks' Authority endpoints. It is signed by RS256 private encryption to allow trusted authorization when passed into another endpoint.

HOW DO I AUTHENTICATE?

Use the Token endpoint indicated in the discovery document to authenticate.

POST

To retrieve an access token, a POST call with the following form fields is required.

GRANT_TYPE

The grant type for your access is "client_credentials" to indicate what type of credentials are being passed in. In this case, authentication is a client.

CLIENT_ID

This is the client identifier assigned to the account on file.

CLIENT_SECRET

This is the client secret assigned to the account on file.

SCOPE

The scope of the access token.

RESPONSE

The response is JSON with the following items.

ACCESS_TOKEN

The JWT bearer token. This must be saved for the session to authorize access to secured Clerks' Authority endpoints.

EXPIRES_IN

Time in seconds when the bearer token will expire.

TOKEN_TYPE

The type of token you are receiving.

CODE EXAMPLES

Below are various code examples to help developers retrieve Token endpoint data. These do not constitute complete code and come with no guarantee that they work on their own. These examples define baseline functionality without proper sanitation, logging, error handling, or user interaction. Be sure to integrate using best practices.

All code examples are in the C# programming language. If you use a different language, look for equivalent calls and utilize the code examples as pseudo-code.

TOKEN RETRIEVAL

The application can call the POST endpoint directly. In this case, parse the returned content as needed for use by your application.

```
private async Task<string> GetTokenAsync()
{
    // Open Client
    using var client = new HttpClient();
    using var formData = new MultipartFormDataContent();

    // Add Form Data
    formData.Add(new StringContent("client_credentials"), "grant_type");
    formData.Add(new StringContent("example_client"), "client_id");
    formData.Add(new StringContent("example_secret"), "client_secret");
    formData.Add(new StringContent("example_scope"), "scope");

    // Retrieve Response
    using var response = await client.PostAsync(IdentityTokenUrl.Value,
                                                formData);

    return await response.Content.ReadAsStringAsync();
}
```

USING TOKEN

When creating an HTTP call to the relevant REST API endpoint, include the authentication token in the request header. The below example shows how to create an HTTP client with the proper header.

```
private static async Task<HttpClient> CreateAuthenticatedClientAsync()
{
   var client = new HttpClient();
   var token = await new GscccaAuth().GetTokenAsync();
   client.DefaultRequestHeaders.Authorization =
          new AuthenticationHeaderValue(TokenType, token);

   return client;
}
```