OPUS

Executive Playbook

OAuth 2.0 for Financial Industry

Securing Your Digitized Financial World



Table of Contents

	Page
Summary	03
Introduction	03
Overview of OAuth 2.0	04
Benefits of Using OAuth 2.0 in the Financial Industry	07
Detailed Case Study: Practical Implementation of OAuth 2.0 for a Bank	08
Validating the Solution Using Postman	14
Practical Examples of OAuth 2.0 in the Financial Industry	17
Major Challenges and Considerations	17
Conclusion	18

© 2026, Opus Technologies

Summary

Digital transformation has brought a renaissance into the financial Industry introducing a whole new world of opportunities both for financial institutions and their customers. It is a catalyst for a lot of new trends in the industry. Rise in the adoption of digital payments, use of digital channels such as wallets, watches and dashboard panels in cars for making payments, Open Banking, Real time money movement can be directly attributed to this fact.

However, the growth in digitization and APIs has also opened new avenues for malicious attacks. This means securing financial data while using APIs has become imperative. Organizations are seeking solutions to protect and secure data while using APIs.

OAuth 2.0 is a powerful and robust solution that integrates with your system and allows you to feel safe about your data by providing API security using tokens.

This white paper explores how OAuth 2.0 can enhance the security of your financial systems while allowing its resources to be accessed for digitization.

Introduction

As stated in the Summary, financial institutions are rapidly digitizing to offer enhanced convenience and efficiency to customers. This shift is increasing the threat landscape on the financial systems. Safeguarding information from unauthorized access and breaches has become challenging. OAuth 2.0 emerges as a critical technology to address these challenges by providing secure, token-based authorization mechanisms that limit exposure of user credentials and offer granular access control.

> Cybercrime is the greatest threat to every company in the world.
>
> **— Ginni Rometty, Former CEO of IBM**

© 2026, Opus Technologies

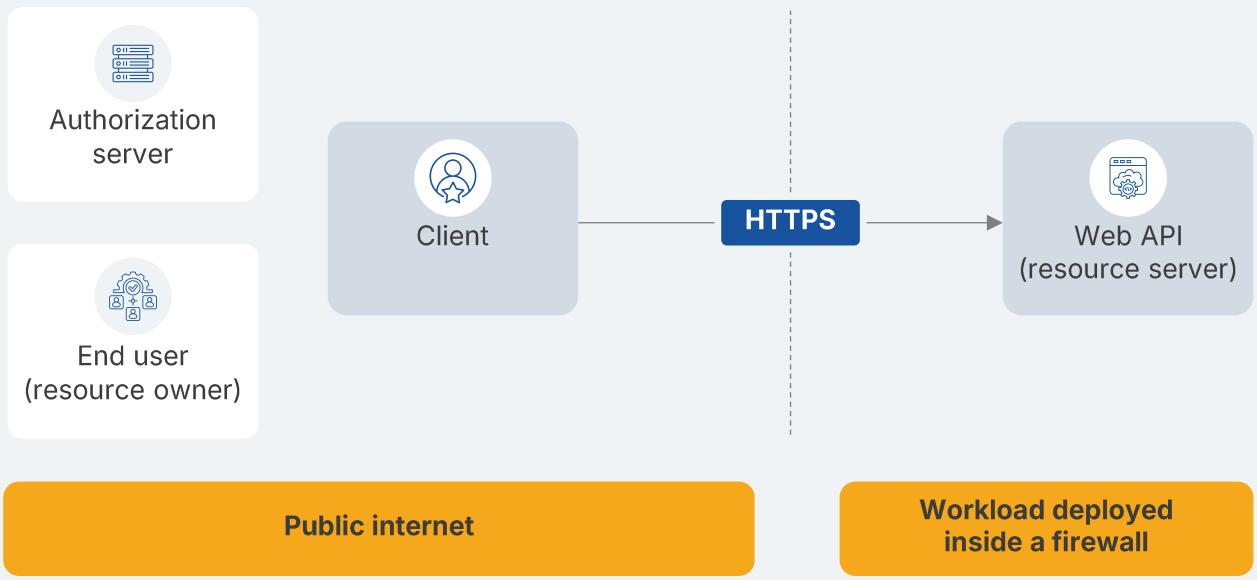
Overview of OAuth 2.0

OAuth 2.0 is an industry-standard authorization framework that enables third-party applications to access user resources without exposing user credentials. It operates on the principle of token-based authentication, where access tokens are issued to applications, providing limited access to user resources as defined by the user.

Key Components of OAuth 2.0

- **Resource Owner:** The user who authorizes an application to access their resources. It can be a financial institution authorizing a client to pull user account details for verification.
- **Client:** The application or client requesting access to the user's resources. It can be the party requesting access to User Account details to verify some information at their end.
- **Authorization Server:** The server that issues access tokens to the client after successfully authenticating the user and obtaining authorization.
- **Resource Server:** The server hosting the user's resources, accepting access tokens to grant access to the client.

Exhibit 1: OAuth 2.0 Security Architecture and Delegated Trust Model



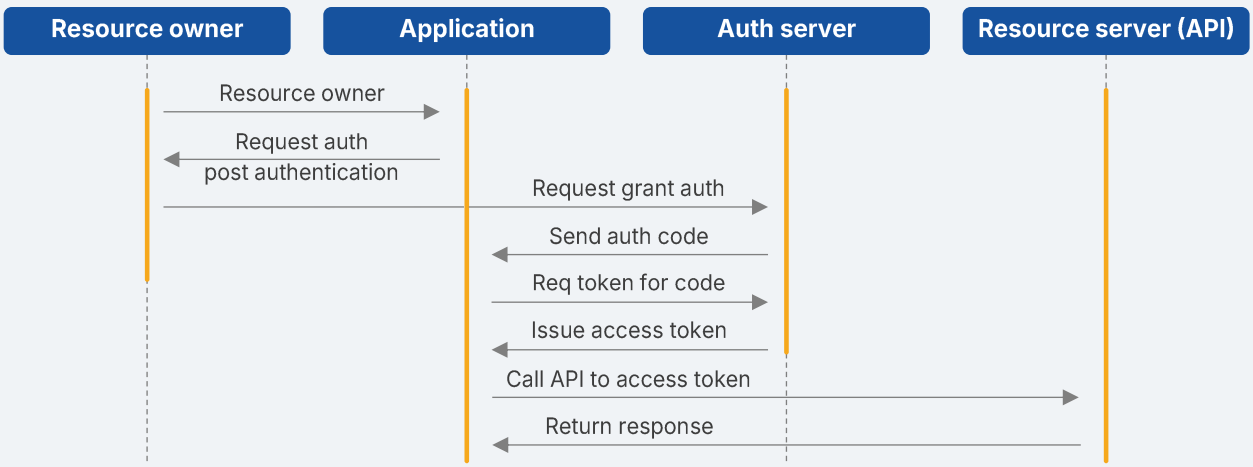
© 2026, Opus Technologies

Authorization Grant Types

OAuth 2.0 allows access via the following Grant Types.

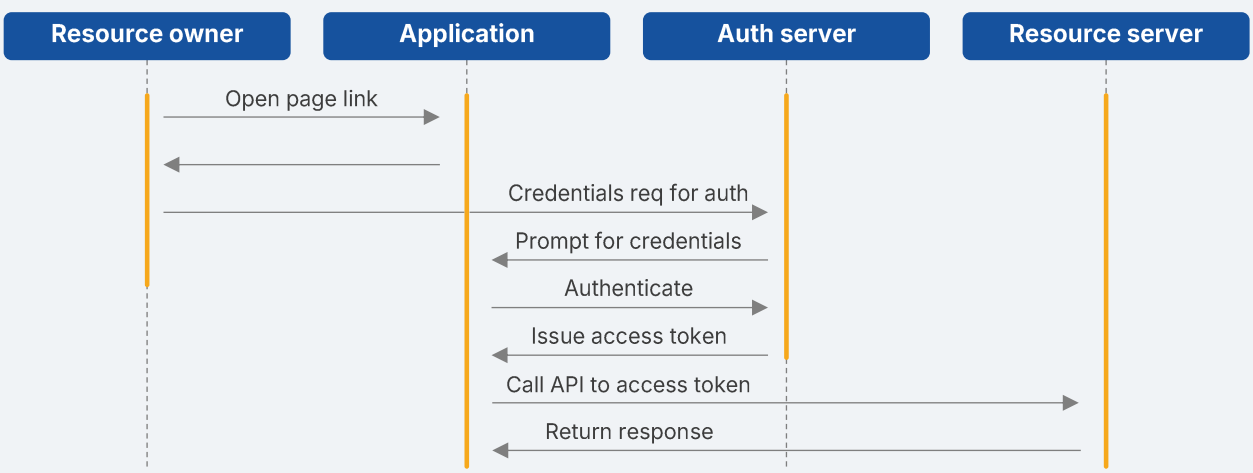
1. **Authorization Code Grant:** Suitable for web and mobile applications, this involves an intermediate code exchange to enhance security. In this case, the Authorization Server sends an Auth code to the client. The client uses this code to exchange for an access token that allows the client to access the resources. It is recommended that all clients use the PKCE extension with this flow as well to provide better security.

Exhibit 2: Authorization Code Grant Interaction Sequence



2. **Implicit Grant:** Simplified flow for public clients, primarily used in client-side web applications running on the browsers with a focus on reduced latency. This one does not allow refresh tokens, so a new token is requested every time it expires. It is not recommended these days as the token is returned without the Auth code exchange as the previous grant type.

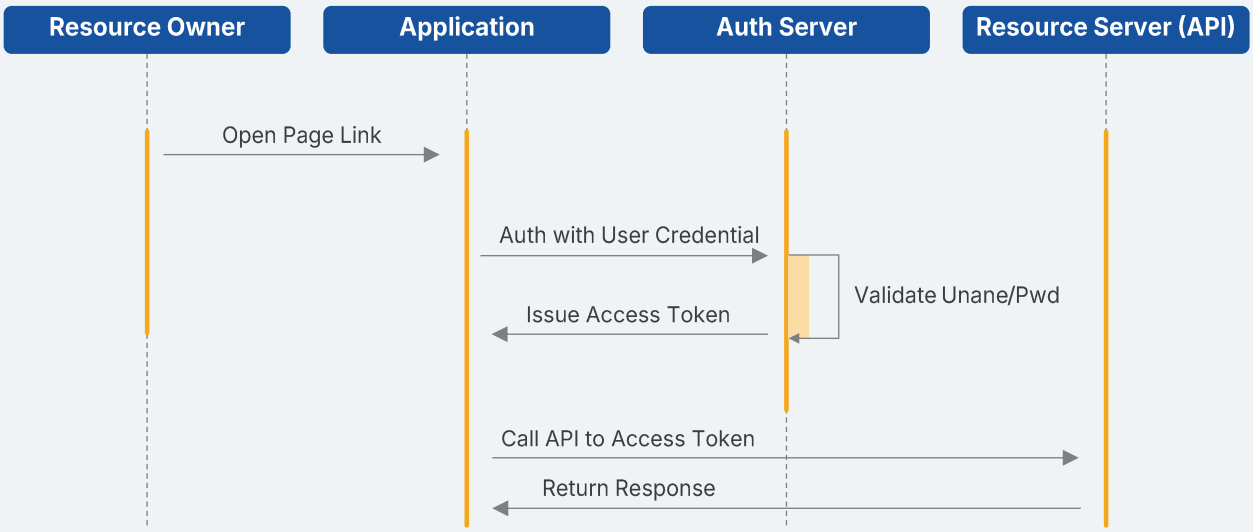
Exhibit 3: Implicit Grant Interaction Sequence



© 2026, Opus Technologies

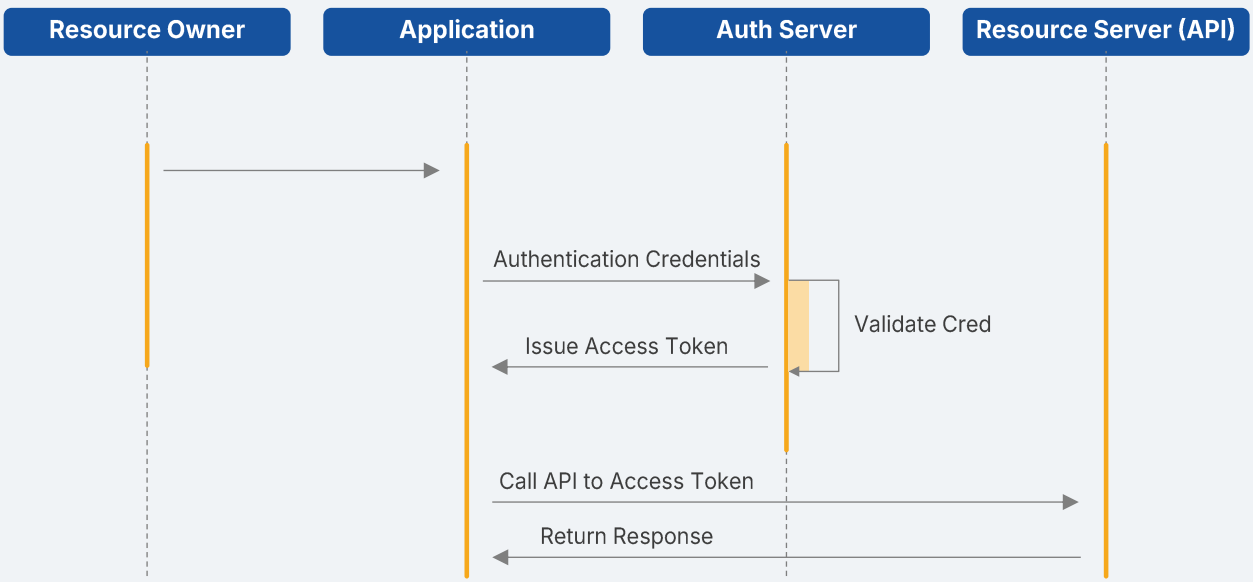
3. **Resource Owner Password Credentials Grant:** Allows the client to exchange user credentials for access tokens, typically used in highly trusted applications.

Exhibit 4: Resource Owner Password Credentials Interaction Sequence



4. **Client Credentials Grant:** Used for application-level access, where the client accesses resources on its own behalf. This is used when a client which is an API needs to access resources stored in a database.

Exhibit 5: Client Credentials Grant Interaction Sequence



© 2026, Opus Technologies

Benefits of using OAuth 2.0 in the Financial Industry

> As we've come to realize, the idea that security starts and ends with the purchase of a prepackaged firewall is simply misguided.
>
> **— Art Wittmann, Editor of Brainyard**

- **Centralized Authorization:** Financial institutions can centralize authorization management with OAuth 2.0, applying consistent security policies compliant with regulatory standards across all applications and services in their organization.
- **Controlled Access:** It allows users to grant limited access to third-party applications without sharing their credentials ensuring security in an event wherein third-party application is compromised.
- **Token-Based Authentication:** Access tokens are short-lived and scope-limited. They reduce the risk of credential theft and misuse. They can be easily revoked or refreshed, providing dynamic control over access permissions to financial data.
- **Principle of Least Privilege:** Supports granular permission scopes, enabling users to specify exact access levels for third-parties minimizing potential exposure and loss from compromised systems by curbing access to only necessary resources.
- **Secure Exchange**: It mandates and enforces the use of secure channels (e.g., HTTPS) for token exchange, ensuring they are transmitted securely and protected from interception.
- **Transparency:** Enhances trust by mandating explicit consent for access, clearly informing users about the data and the purpose of access.
- **Integration with MFA:** Fosters integration with multi-factor authentication (MFA), adding an additional layer of security. MFA reduces risk of a breach even if the token is compromised.
- **Continuous Improvement:** It is an industry standard and is continuously updated to address emerging security threats. Continuous improvement and community driven best practices elevate the benefits organizations have with OAuth 2.0.
- **On-Demand Revocation:** Tokens can be easily revoked, mitigating the impacts during breaches by allowing immediate termination of access in case of suspicious activities.
- **Interoperability:** It is highly adopted in the industry and is compatible with various applications and services. It facilitates secure integration across platforms. It is vital for the financial industry's complex ecosystem.

© 2026, Opus Technologies

Detailed Case Study: Practical Implementation of OAuth 2.0 for a Bank

Let us discuss a case study describing OAuth 2.0 implementation for a financial organization to understand the concept. This is a Bank that needed to protect their APIs ensuring no unauthorized personnel or system can access their resources through their APIs. Their API workloads are running in Kubernetes on Azure Cloud. Hence Azure AD was chosen for OAuth 2.0 integration. The API application code is written in Java Spring boot and hence Spring Security module was used to implement OAuth 2.0 in the application. This section describes the entire process to set-up OAuth 2.0 for APIs using Azure AD to facilitate secure, programmatic access to Application data.

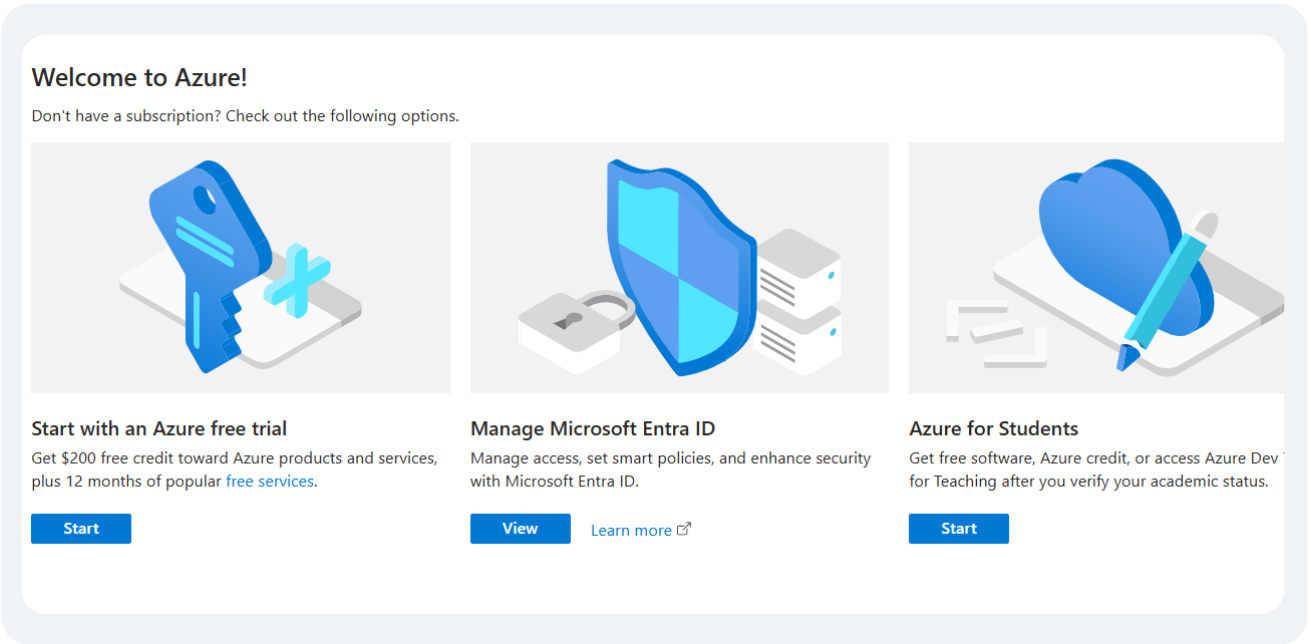
The process is divided into three major steps

1. Setting up OAuth 2.0 resource in Azure AD
2. Setting up the Application Code to implement OAuth 2.0
3. Validating the Work Using Postman

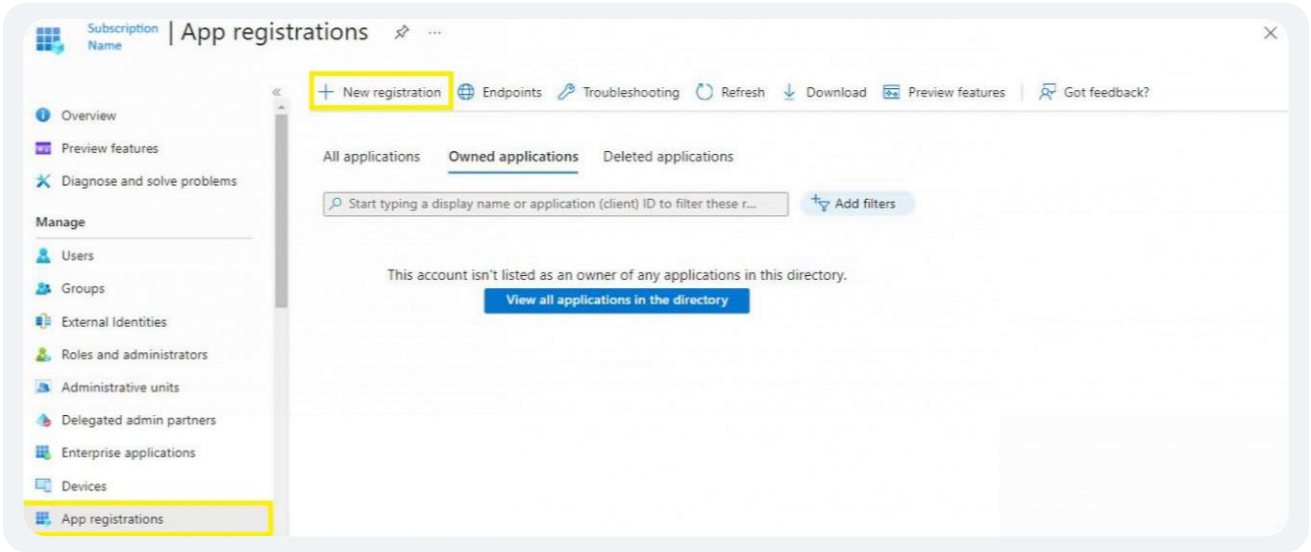
Configure OAuth 2.0 Resource in Azure AD

This section describes the step-by-step process to configure Azure AD to support security via OAuth 2.0. The steps are also documented in Azure portal. This paper simplifies the steps for a spring boot application.

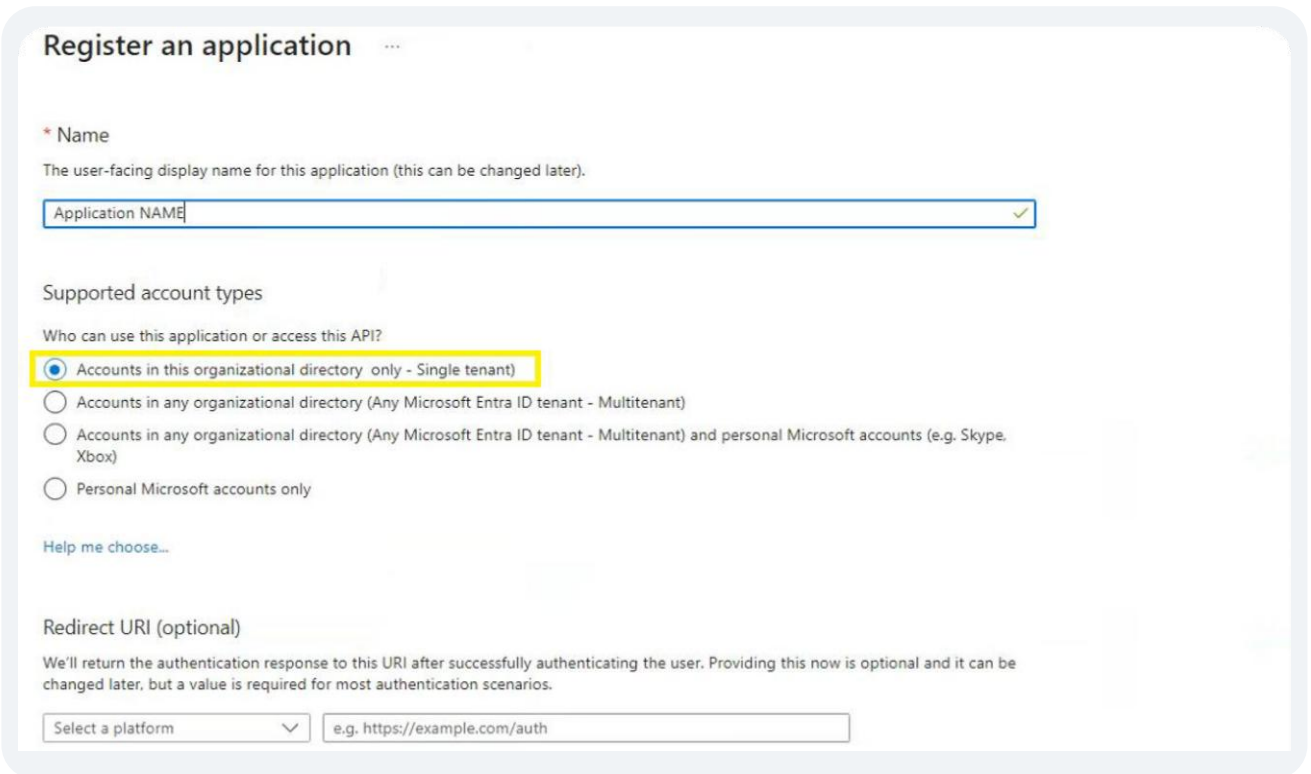
1. Navigate to the Microsoft Azure Portal and authenticate.



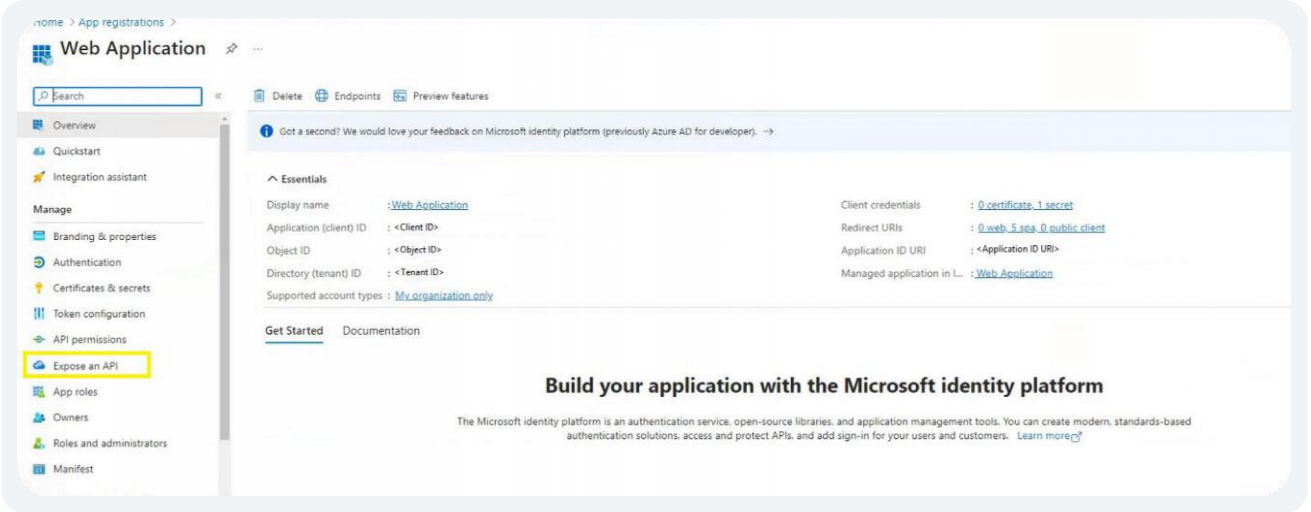
2. Navigate to **Azure Active Directory/ Microsoft Entra ID.**
3. Click on **App Registrations.**
4. Click on **New Registration.**



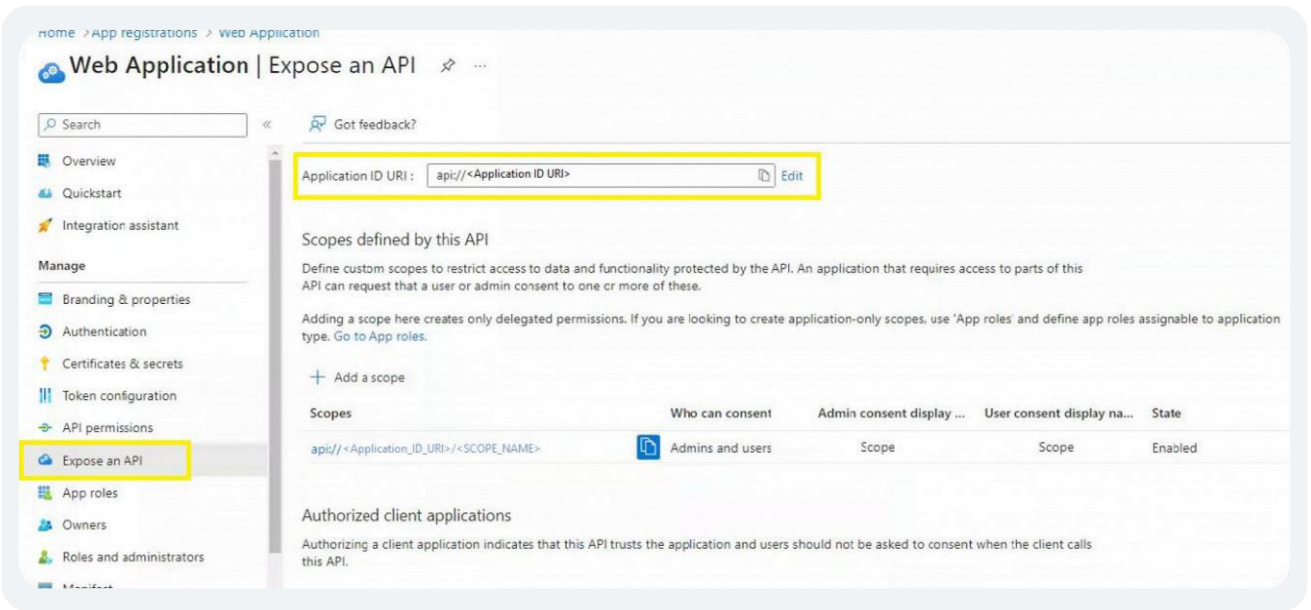
5. Enter **<Application Name>.**
6. Verify the **Supported account types** are set to **Single Tenant.**
7. Click **Register.**



8. Click on **Expose an API.**



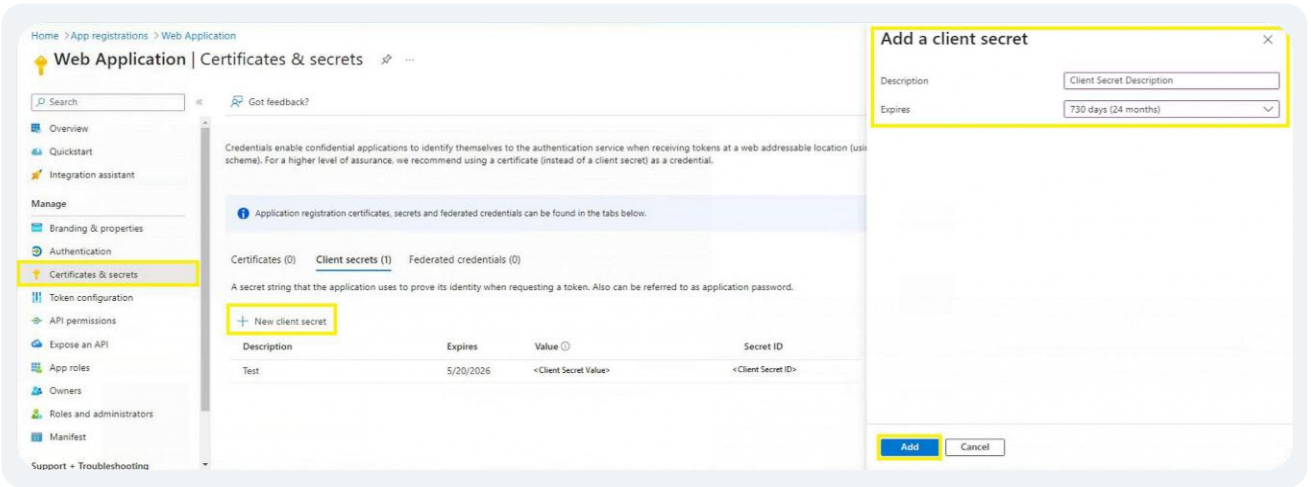
9. Click on the Set link next to **Application ID URI** to set the Application ID URI



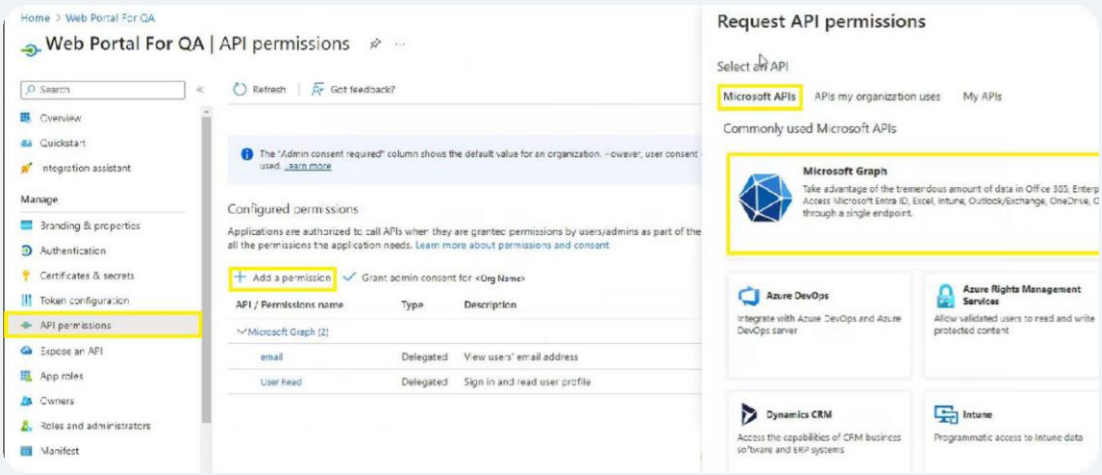
The Application ID URI must be unique within your organization's directory, such as https://your.company.com/4d2a8c2b-a5f4-4b86-93ca-294185f45f2e. This value will be referred to as the<APPLICATION_APPLICATION_ID_URI> in the subsequent configuration steps.

© 2026, Opus Technologies

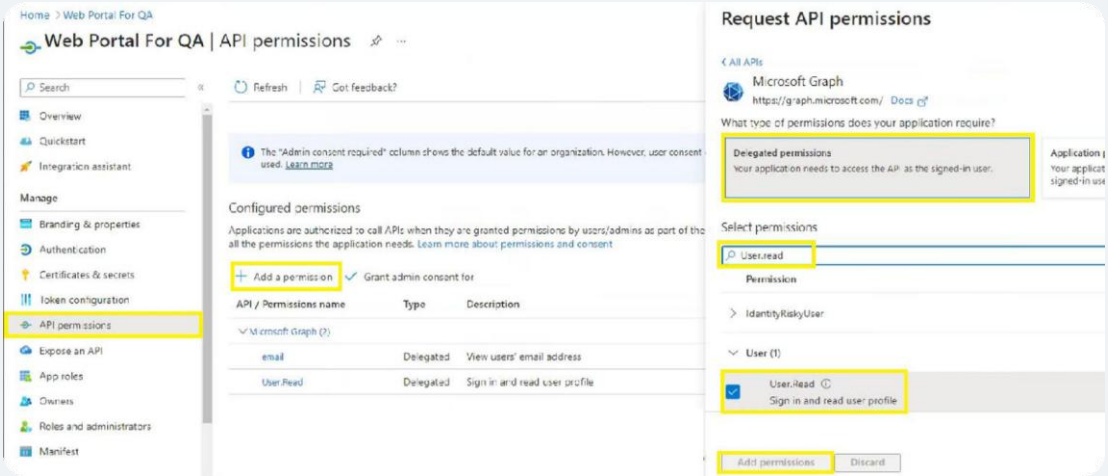
10. To add an Application Role as an OAuth scope for OAuth flows where the programmatic client acts on behalf of a user, click on Add a scope to add a scope representing the Application role.
    - Enter the scope by having the name of the Application role with the session:scope:prefix. For example, for the Application Analyst role, enter session:scope:roleName.
    - Select who can consent
    - Enter a display name for the scope (e.g.: Account Admin)
    - Enter a description for the scope (e.g.: Can administer the Application account)
    - Click Add Scope
11. Click **Save.**
12. Click on **Certificates & secrets and then new client secret.**
13. Add a **description** of the secret.
14. Select **Expiry time duration for client secret.**
15. Click **Add.** Copy the secret. This will be known as the <OAUTH_CLIENT _SECRET> in the following steps.



16. For programmatic clients that will request an Access Token on behalf of a user, configure Delegated permissions for Applications as follows.
    - Click on **API Permissions.**
    - Click on **Add Permission.**
    - Click on **Microsoft Graph.**



- Click on the **Delegated** Permissions box.
- Search **"User.Read"** in Select Permissions section.
- Select **"User.Read"** under **User Permissions.**
- Click **Add Permissions.**
- Click on the **Grant Admin** Consent button to grant the permissions to the client if Admin Consent Required.
- Click **Yes.**
- Add other permissions in the same way according to the requirement.



Set Up OAuth 2.0 in a Spring Boot Application

This section describes the step-by-step process to set up OAuth 2.0 using Azure AD in a Spring boot application using dependency packages.

1. Add the following to your Spring boot application to setup connection with Azure Cloud

```
<properties>
<java.version>17</java.version>
<spring-cloud-azure.version>5.12.0</spring-cloud-azure.version>
</properties>
```

2. Add the following dependencies in the Pom.xml file of your Spring boot application. The starter active directory package allows you to connect your web application to Azure AD.

The resource-server package is added to enable the server/API to validate the access token before serving a request.

With the addition of these two dependencies, your application is ready for integration with OAuth 2.0.

```
<dependency>
<groupId>com.azure.spring</groupId>
<artifactId>spring-cloud-azure-starter-active-directory</artifactId>
</dependency>
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-oauth2-resource-server</artifactId>
</dependency>
```

3. Add the following in the Application.properties file.

The spring-cloud-azure-starter-active-directory provides the most optimal way to connect your web application to a Microsoft Entra ID (Microsoft Entra ID for short) tenant and protect your resource server with Microsoft Entra ID. It uses the OAuth 2.0 protocol to protect web applications and resource servers. This property enables the azure.active-directory in Spring boot application.

spring.cloud.azure.active-directory.enabled=true

Spring boot application uses <Client_Id> provided in this property to make a connection with our Registered Application.

spring.cloud.azure.active-directory.credential.client-id=<CLIENT_ID>

Spring boot application uses <Application_Id_URI> provided in this property to look for the scopes under our Registered Application.

spring.cloud.azure.active-directory.app-id-uri= <APPLICATION_ID_URI>

4. Add the following annotation in the code for respective endpoints.

@PreAuthorize ("hasAuthority('SCOPE_<ScopeName>')")

Spring boot application checks Scope mentioned in above annotation before entering end point function. If the user is not authorized in defined scope of Application, the application will restrict the request from entering the end point and respond back with 401 Unauthorized message.

This completes the steps to configure OAuth 2.0, now the application is ready to be deployed and tested.

© 2026, Opus Technologies

Validating the Solution Using Postman

This section describes how Postman can be configured to test the APIs to ensure they areprotected using OAuth 2.0 preventing unauthorized accessesc

1. Open a new Postman page
2. Add Authentication Header
3. Authorization → Auth Type → OAuth 2.0
4. Add Authorization Data To → Request Headers
5. Configure New Token→

Header Prefix → Bearer

Grant Type: Password Credentials

Access Token Url: https://login.microsoftonline.com/<Tenant_id>/oauth2/v2.0/token

Client id: <Client_id>

Client Secret: <Client_Secret>

Username: user@domain_name.com

password: <Password>

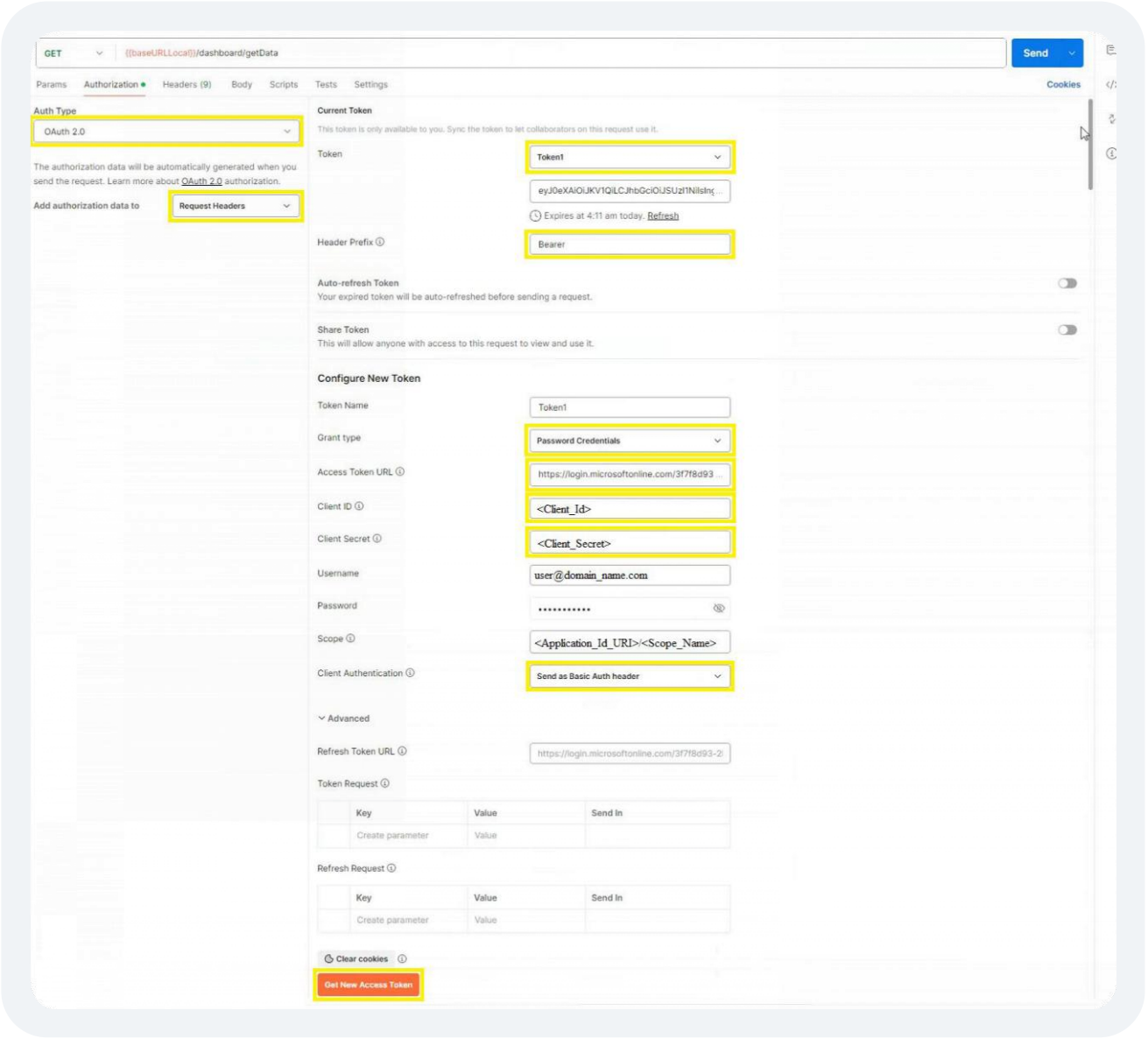
scope: <Application_Id_URI>/<Scope_Name>

Client authentication: send as a basic auth header

6. Click on "get new access token".
7. Select access token from available tokens.
8. Click on send.

© 2026, Opus Technologies

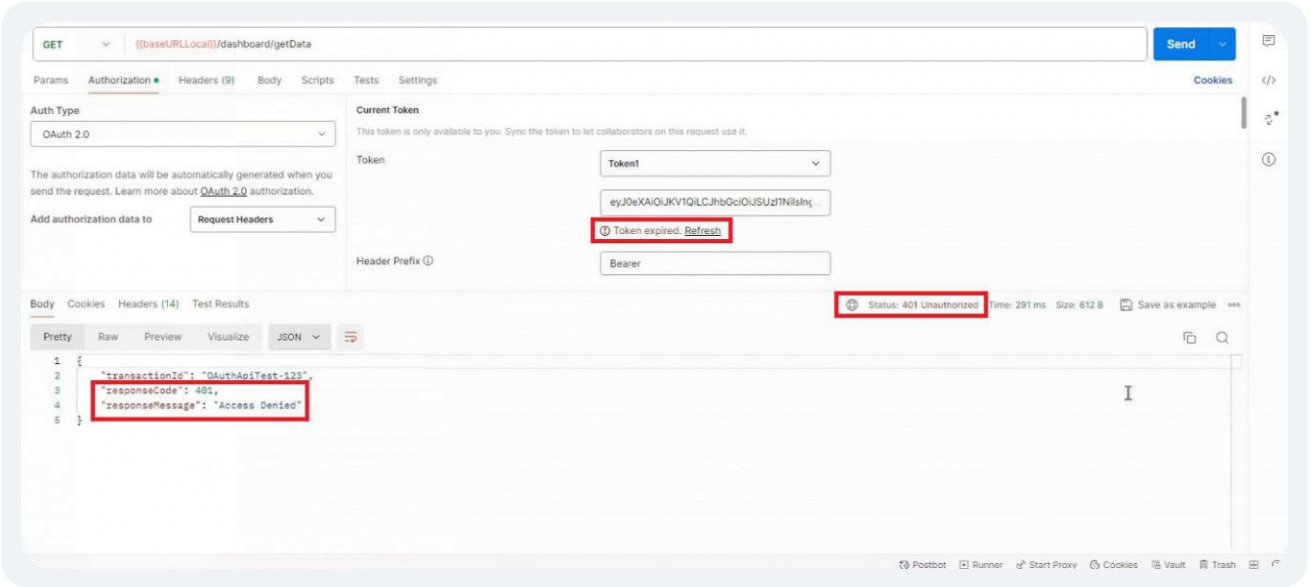
The Access Token generated by Azure AD/ Microsoft Entra gets expired in 75 minutes by default and this can be configured based on the needs.



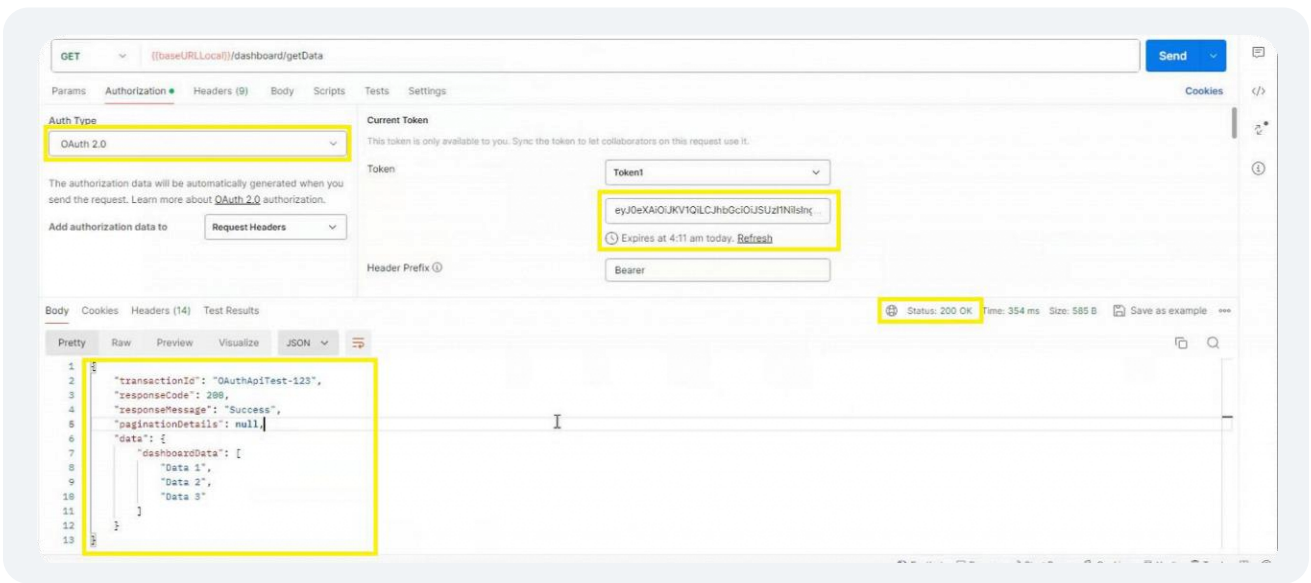
This is how you can get a secure access token from Azure AD. This token is further used by a client which can be a client application running on a browser to authenticate and get resource data from the backend API running on the Spring boot application.

When a client tries to access a resource/API in the application using an Invalid/Expired Token, then the request is rejected as you can see below. The response does not have any data and the request to access the API is rejected with a 401 Unauthorized error code stating "access denied" to the resource.

© 2026, Opus Technologies



When the client uses a valid token procured from the above API, access is granted, and data is returned from the resource API in the response. In the snapshot below, the response contains resource data and the server allowed access after validating the token sent in the request.



Practical Examples of OAuth 2.0in the Financial Industry

1. Securing API Access ensures only Authorized apps and systems can access information or perform financial transactions. A lot of FI's are using OAuth 2.0 to protect their Financial APIs today.

2. Open Banking Initiatives is boosted by leveraging OAuth 2.0 by enabling innovative financial services for third parties while maintaining security.

3. Financial mobile and web applications leverage OAuth 2.0 to authenticate users and authorize access to sensitive data.

4. Integrations with Processors, Financial Management and Investment tools are facilitated using OAuth 2.0.

Major Challenges and Considerations

1. Implementing OAuth 2.0 requires careful planning and expertise. Financial institutions must ensure proper configuration of authorization servers, secure storage of tokens, and robust user authentication mechanisms.

2. While OAuth 2.0 enhances security, the management and storage of access tokens are critical. Tokens must be stored securely to prevent unauthorized access and potential misuse. Organizations have been breached as they had stored the tokens in Session Storage in browsers, storing the tokens is an essential part of this process.

3. Ensuring that OAuth 2.0 implementations comply with relevant regulations, such as GDPR, PSD2, and other data protection laws are very important. This involves transparent data handling practices and secure user consent mechanisms.

4. OAuth 2.0 implementations require continuous monitoring and auditing to detect and respond to potential security threats. Regular security assessments and updates are necessary to maintain a robust security posture. Many organizations use tools like No Name Security to continuously monitor their APIs for anomalies.

© 2026, Opus Technologies

Conclusion

Those are my favorite lines from the movie Dark Knight, and I consider OAuth 2.0 as the Dark Knight protecting our systems. It offers a powerful framework for enhancing security in the financial industry. By providing secure, token-based authorization mechanisms, OAuth 2.0 protects sensitive financial data, supports regulatory compliance, and fosters user trust. As Fis continue to embrace digitization, security threats will grow. OAuth 2 will play a crucial role in ensuring pristine security for organizations in this regime.

> Because he can take it. Because he is not a hero. He is a silent guardian, a watchful protector. A dark knight.
>
> **— Commissioner Gordon**

Opus: Your Trusted Partner in Building Secure Solutions

At Opus we consider security to be a top priority item while building and transforming your Financial Software. Our teams are well equipped in building OAuth 2 Integrations using different platforms for your financial systems. They are also trained to write secure code, consult, and recommend software tools for SAST, Container and SCA scans and perform threat modeling while designing software. Opus can help you in your Continuous Security journey by building API security using OAuth 2.0 and educate your teams with Secure SDLC processes.

© 2026, Opus Technologies

About the Authors



Debashis Bhattacharyya
Technology Consulting Leader

Debashis Bhattacharyya has over 18 years of experience planning and delivering technology solutions, specializing in Cloud, API, DevSecOps and Application Modernization. He has been instrumental in Opus' transformational DevOps initiative with FIS that won the prestigious DevOps Dozen Award for 'Best DevOps Industry Implementation' in 2022.



Chetan Shinde
Senior Tech Lead

Chetan Shinde has over 12 years of experience in designing and developing Java based applications in Payment Domain, specializing in API-Microservices design and development using Spring Boot and their integration of applications with Cloud Technologies (Azure/AWS).

References

1. OAuth 2.0 Authorization Framework: RFC 6749, The Internet Engineering Task Force (IETF).
2. Financial Industry Regulatory Authority (FINRA): Guidelines on cybersecurity practices.
3. European Banking Authority (EBA): Regulatory Technical Standards (RTS) on strong customer authentication and secure communication.
4. Open Banking Implementation Entity (OBIE): Standards and guidelines for open banking APIs and security.
5. Microsoft identity platform (v2.0) overview
6. Authentication protocol (and related topics)
7. Application configuration (and related topics)
8. How-to guides for Authentication & Application Configuration (and related topics)

© 2026, Opus Technologies



Opus is a trusted engineering partner to payment providers, banks and fintechs navigating change in a real-time, digital-first world. We solve for business value—accelerating go-to-market, unlocking future readiness, and powering transformation at scale.

Our promise is simple: Business Value Acceleration. Realization. Maximization.

We deliver on this promise through the Opus trifecta—a domain-native engineering skillset, a value-creation mindset, and a toolset that drives business agility. From MVP (Minimum Viable Product to MPT (Maximum Possible Transformation), Opus turns engineering into business advantage.

Contact us

info@opustechglobal.com

www.opustechglobal.com

