User Sign In Strategies

Last updated by | Saldua, Roseller | Sep 10, 2025 at 12:47 PM GMT+5

👥 User Sign-In Strategies

📋 Table of Contents

🎯 Overview

CAL Angular provides a streamlined approach for authenticating users in Angular applications, leveraging 🔗 @azure/msal-browser to support Microsoft Single Sign-On (SSO).

When a user is signed in to any Microsoft product or account, they are automatically authenticated in your application. Signing out from one Microsoft service will sign the user out across all connected services.

User sign-in can be initiated in two ways:

1. ⚡ **Automatic Sign-In**: The application automatically starts the sign-in process when the page loads. If the user is already signed in to any Microsoft account (via SSO), they are signed in to the application without further action. If not, the browser prompts for sign-in. Set `autoSignIn: true` in your configuration.

2. 👤 **User-Initiated Sign-In**: The user actively starts the sign-in process, typically by clicking a "Sign In" button. This triggers the authentication flow, allowing users to log in at their convenience. Set `autoSignIn: false` in your configuration.

For authentication, you can choose between two methods:

1. ➡️ **Redirect**: The application redirects the user to the Azure AD login page. After successful authentication, the user is returned to the application. Set `popupForLogin: false`
2. 🪟 **Popup**: The application opens a popup window for sign-in, allowing the main application window to remain accessible. Set `popupForLogin: true`

> 💡 **Note:** Upon successful sign-in, CAL Angular creates a cache for your claims principal. Depending on your CAL configuration, this cache is stored in either sessionStorage or localStorage (key: `cachedCvxClaimsPrincipal`). You can inspect this cache in your browser's developer tools (e.g., Edge: F12 > Application tab > Storage). Signing out removes claims from the cache. For details on retrieving claims from cache, see getAccessTokenFromCache()

These options are defined in your CAL configuration. For setup instructions, refer to CAL Angular Configuration.

```
// config.json
{
    "autoSignIn": false,
    "popupForLogin": true,
    "cacheLocation": "localStorage",
    "instance": "https://login.microsoftonline.com/",
    "tenantId": "fd799da1-bfc1-4234-a91c-72b3a1cb9e26",
    "clientId": "",
    "redirectUri": "http://localhost:4200",
    "oidcScopes": [
        "openid",
        "profile",
        "offline_access",
        "User.Read"
    ],
    "graphScopes": [
        "User.Read",
        "User.Read.All"
    ]
}
```

Auto Sign-In

```
{
  "autoSignIn": true,
  "popupForLogin": true,
  // ... other configuration properties
}
```

```
import { Component, inject, OnInit } from '@angular/core';
import { CalAngularService, ICvxClaimsPrincipal } from '@cvx/cal-angular';

@Component({
  selector: 'app-simple',
  imports: [],
  templateUrl: './simple.component.html',
  styleUrl: './simple.component.css',
})
export class SimpleComponent implements OnInit {
  private calService = inject(CalAngularService);
  claimsPrincipal: ICvxClaimsPrincipal | null = null;
  name: string | null = null;
  username: string | null = null;

  ngOnInit() {
    this.calService.isUserSignedIn().subscribe((isSignedIn) => {
      if (isSignedIn) {
        this.claimsPrincipal = this.calService.cvxClaimsPrincipal;
        this.name = this.claimsPrincipal?.name;
        this.username = this.claimsPrincipal?.userName;
      }
    });
  }
}
```

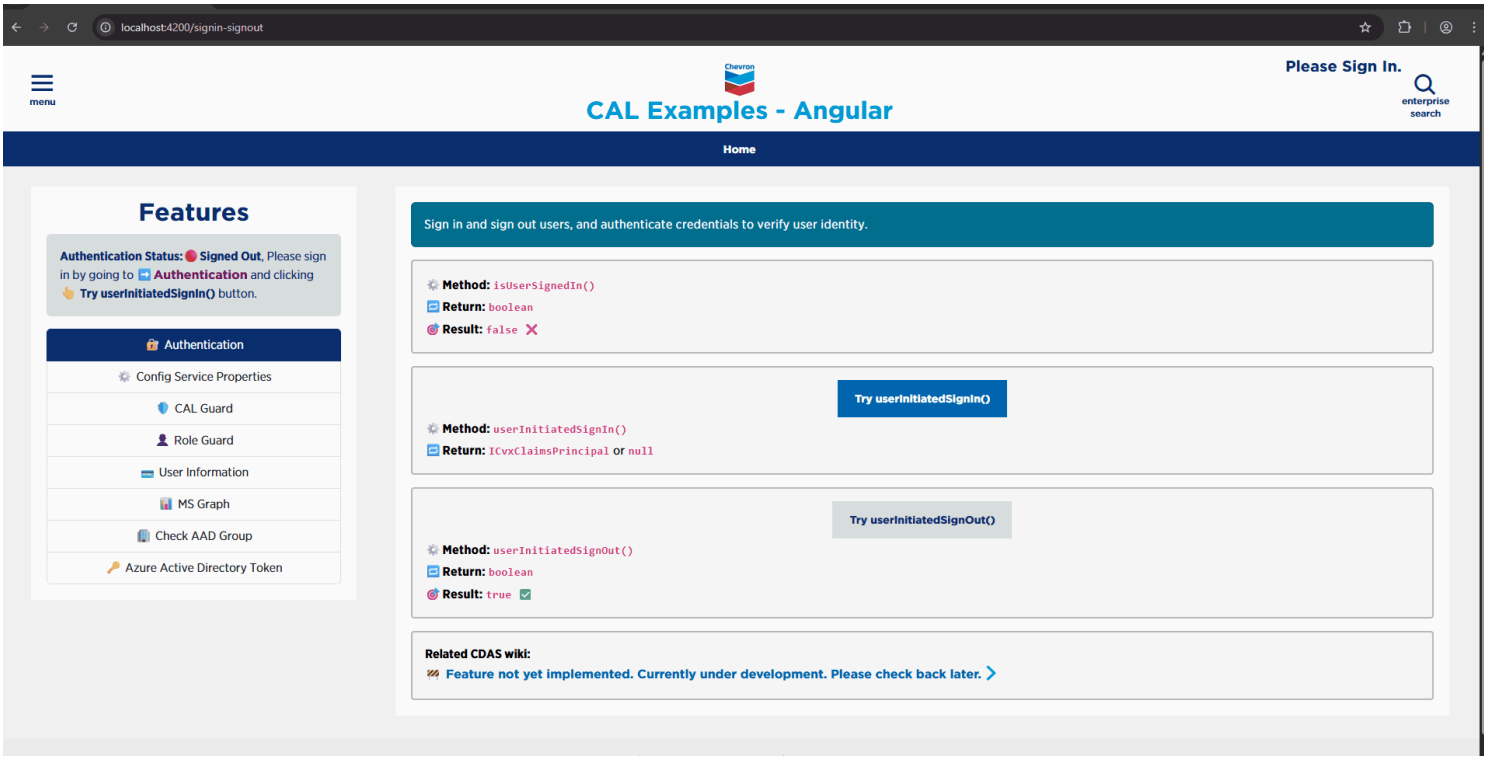
```
@if (claimsPrincipal) {
  <div>
    <h2>User Information</h2>
    <p><strong>Name:</strong> {{ name }}</p>
    <p><strong>Username:</strong> {{ username }}</p>
  </div>
}
```

If you set `autoSignIn: true` CAL will automatically handle user sign-in for you, no need to call any sign-in methods manually. This works on every page where CAL is used.

To check if a user is signed in, use the `isUserSignedIn()` method. It returns `true` if the user is signed in, or `false` if not.

> 🎯🚩 **Example: autoSignIn**

💡 In this GIF, notice that after signing out, we simply refresh the page. The sign-in popup appears automatically—without needing to click the sign-in button—because autoSignIn is enabled.



👤 User-Initiated Sign-In

```
{
  "autoSignIn": false,
  "popupForLogin": true
  // ... other configuration properties
}
```

```
import { Component, inject, OnInit } from '@angular/core';
import { CalAngularService, ICvxClaimsPrincipal } from '@cvx/cal-angular';

@Component({
  selector: 'app-simple',
  imports: [],
  templateUrl: './simple.component.html',
  styleUrl: './simple.component.css',
})
export class SimpleComponent implements OnInit {
  private calService = inject(CalAngularService);
  claimsPrincipal: ICvxClaimsPrincipal | null = null;
  loggedIn: boolean = false;
  loggedInName: string = '';

  async ngOnInit() {
    this.calService.isUserSignedIn().subscribe(async (isSignedIn) => {
      if (isSignedIn) {
        this.loggedIn = isSignedIn;
      }
    });
  }

  onSignIn() {
    this.calService.userInitiatedSignIn().subscribe((cvxClaimsPrincipal) => {
      this.loggedIn = true;
      this.claimsPrincipal = cvxClaimsPrincipal;
      this.loggedInName = cvxClaimsPrincipal?.name ?? '';
    });
  }

  onSignOut() {
    this.calService.userInitiatedSignOut().subscribe((value: boolean) => {
      this.loggedIn = value;
      if (!value) {
        this.loggedInName = 'No User Signed In';
      }
    });
  }
}
```

```
<button (click)="onSignIn()">Sign In</button>
<button (click)="onSignOut()">Sign Out</button>

@if (loggedInName) {
  <p>Welcome, {{ loggedInName }}!</p>
}
```

If you set `autoSignIn` to `false`, you need to call two methods in CAL Angular:

- 🔗 userInitiatedSignIn – This initiates user sign-in and returns ICvxClaimsPrincipal if successful.
- 🔗 userInitiatedSignOut – This initiates user sign-out and returns either `true` or `false`

> 🎯🚩 **try the live example:** 🔗 userInitiated

🔒 Azure AD Group Checking for Custom Authorization

If you want to check if a user is in an Azure AD group in order to enforce authorization on a resource. You can call the following method below. This allows you to build your own authorization based on this, stay tuned for future updates.

```ts
import { Component, inject, OnInit } from '@angular/core';
import { CommonModule } from '@angular/common';
import { FormsModule } from '@angular/forms';
import { CalAngularService } from '@cvx/cal-angular';

@Component({
  selector: 'app-simple',
  imports: [CommonModule, FormsModule],
  templateUrl: './simple.component.html',
  styleUrl: './simple.component.css',
})
export class SimpleComponent implements OnInit {
  private calService = inject(CalAngularService);
  loggedIn: boolean = false;
  isInGroupResult: boolean | null = null;
  groupName: string = '';
  groupInfo: string = '';
  async ngOnInit() {
    this.calService.isUserSignedIn().subscribe((isSignedIn) => {
      if (isSignedIn) {
        this.loggedIn = isSignedIn;
      }
    });
  }

  checkGroupInput(groupName: string) {
    if (groupName) {
      this.calService.isInGroup(groupName).subscribe((result) => {
        this.isInGroupResult = result;
        if (result) {
          this.groupInfo = 'You have access to confidential information!';
        } else {
          this.groupInfo = 'Unauthorized: You do not have access to this information.';
        }
      });
    }
  }
}
```

```html
<div>
  <label for="groupName">Enter Group Name:</label>
  <input id="groupName" #groupInput placeholder="Group name" />
  <button (click)="checkGroupInput(groupInput.value)">Check Group</button>
</div>

@if (isInGroupResult !== null) {
  @if (isInGroupResult) {
    <span>✅ You are in the group!</span>
    <span>{{ groupInfo }}</span>
  } @else {
    <span>❌ You are NOT in the group.</span>
  }
}
<div>{{ groupInfo }}</div>
```

> 🎯🚩 **try the live example:** 🔗 isInGroup