Configuration

Last updated by | Saldua, Roseller | Sep 30, 2025 at 8:42 AM GMT+5

⚙️ CAL Angular Configuration

📋 Table of Contents

🎯 Overview

> 💡 Before you begin, make sure you have an `Azure App Registration` This identifies your application to Azure AD and gives it permission to authenticate users. For more details, check these documents: 🔗 Authenticating with App Registrations 🔗 Configuring Your App Registration

> 💡 Ensure that you have `CAL Angular` installed. If not, run this command: `npm install @cvx/cal-angular`

Before you begin, please note that this guide demonstrates the Angular 19 standalone approach. If you are using the module-based approach, you may not see `app.config.ts` . However, both standalone and module approaches use the same method for configuring CAL Angular: `CalAngularModule.forRoot`

With `CalAngularModule.forRoot` , you can take advantage of the 🔗 ConfigService, which is designed to load and access configuration from external sources.

🗂️ Creating the CAL Configuration File

🔎 First, ensure the `assets` folder exists within your `src` directory. If it does not, please create it.
📝 Then, add a configuration file named `config.json` to this folder. Using `config.json` is strongly recommended for CAL config, as other names may lead to deployment issues. Copy and paste the CAL configuration properties below into your file.

> 💡 Note: In the sample configuration, the `clientId` field is left blank. Be sure to provide your own `clientId` which is typically available in your Key Vault.

```
{
    "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"
    ]
}
```

> 📖 To learn more about CAL configuration properties, please see: 🔗 Config.ts

> 📖 For details on **`cacheLocation`** options, visit: 🔗 Token Caching.

> 💡 Note: Items stored in cache include tokens and claims.

💡 If you want a dynamic configuration of CAL during deployment, use Azure App Service Ansible roles in your pipeline.

References:

- 👤 🔗 Role Variables — Read the `RoleVariables.md` and search for `azure_app_service_angular_app_settings`
- 📋 🔗 Angular Example Playbook - Read the `ExamplePlaybooks.md` and search for `Angular App Settings`

🧩 Registering CalAngularModule

✅ Once you have created your configuration file, the next step is to register `CalAngularModule` in your Angular application.

- 🏗 For the Angular 19 standalone approach, add the registration to `app.config.ts`
- 🏛 For the module-based approach, add it to `app.module.ts`

🔧 The `CalAngularModule` provides a method called `forRoot` which accepts the path to your CAL configuration file. For example: `CalAngularModule.forRoot('assets/config.json')`

📚 Below are examples for both approaches:

🅰 Angular 19 Standalone Approach:

```
import {
  ApplicationConfig,
  importProvidersFrom,
  provideZoneChangeDetection,
} from '@angular/core';
import { provideRouter } from '@angular/router';
import { RouterModule } from '@angular/router';

import { routes } from './app.routes';
import { BrowserModule } from '@angular/platform-browser';
import { CalAngularModule } from '@cvx/cal-angular';
import { provideHttpClient } from '@angular/common/http';

export const applicationConfig: ApplicationConfig = {
  providers: [
    provideZoneChangeDetection({ eventCoalescing: true }),
    provideRouter(routes),
    importProvidersFrom(
      BrowserModule,
      RouterModule,
      // Loads CAL configuration and provides CAL services (ConfigService, CalAngularService, RoleGuardService
      CalAngularModule.forRoot('assets/config.json')
    ),
    provideHttpClient(),
  ],
};
```

📦 Module-Based Approach:

```
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { ReactiveFormsModule } from '@angular/forms';
import { HttpClientModule } from '@angular/common/http';
// app routing is required if you would like to use CAL Guard
import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';

import { CalAngularModule } from 'cal-angular';

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    HttpClientModule,
    CalAngularModule.forRoot('assets/config.json'),
    AppRoutingModule,
    ReactiveFormsModule
  ],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Registering `CalAngularModule.forRoot('assets/config.json')` enables you to use the following CAL services throughout your Angular application:

- 🛠️ `ConfigService` – Loads and provides access to CAL configuration from external sources.
- 🔐 `CalAngularService` – Main service for authentication, user claims, and related functionality (🔗 cal-angular.service.ts).

- 🛡️ **`RoleGuardService`** – Protects routes based on user roles and permissions.
- 🚪 **`CalGuardService`** – Protects routes that require authentication

✨ Example Usage of CAL Angular

Once your configuration is complete, you can start using CAL Angular in your application. Below is a simple example demonstrating how to use both the `CalAngularService` and `ConfigService`

```
import { CommonModule } from '@angular/common';
import { Component, inject, OnInit } from '@angular/core';
import { CalAngularService, ConfigService } from '@cvx/cal-angular';
@Component({
  selector: 'app-simple',
  imports: [CommonModule],
  templateUrl: './simple.component.html',
  styleUrl: './simple.component.css',
})
export class SimpleComponent implements OnInit {
  private calService = inject(CalAngularService);
  protected configService = inject(ConfigService);
  userName: string = '';

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

```
<p>👋 Welcome, {{ userName }}!</p>
<p>🔄 Auto Sign-In: {{ configService.getSettings("autoSignIn") }}</p>
```

> 🎯 🚩 **`try the live example:`** 🔗 isUserSignedIn()

> 🎯 🚩 **`try the live example:`** 🔗 getSettings()