Guía del usuario de BPA Crear un MFE personalizado v4.1.2

Contenido

Creación personalizada de MFE

MicroFrontEnd (MFE) es un modelo arquitectónico en el que se crean diferentes fragmentos de páginas de interfaz de usuario (IU) en tiempo de compilación o en tiempo de ejecución. Cada MFE se desarrolla de forma independiente.

Creación de MFE

Las aplicaciones desarrolladas con cualquier estructura se pueden cargar como un MFE. La sección siguiente es un ejemplo de cómo agregar una aplicación Angular como MFE.

Creación de aplicaciones angulares

1. Ejecute el siguiente comando para crear una aplicación Angular denominada test-mfe.

```
ng new test-mfe –routing –style=scss
```

```
[root@bpa-77 MFE]# ng new test-mfe --routing --style=scss
CREATE test-mfe/README.md (1061 bytes)
CREATE test-mfe/.editorconfig (274 bytes)
CREATE test-mfe/.gitignore (548 bytes)
CREATE test-mfe/angular.json (3106 bytes)
CREATE test-mfe/package.json (1039 bytes)
CREATE test-mfe/tsconfig.json (863 bytes)
CREATE test-mfe/.browserslistrc (600 bytes)
CREATE test-mfe/karma.conf.js (1425 bytes)
CREATE test-mfe/tsconfig.app.json (287 bytes)
```

test-mfe

2. Ejecute el comando `cd` para cambiar a test-mfe.
3. Ejecute el siguiente comando para agregar un único spa en test-mfe:

```
ng add single-spa-angular@7 –project test-mfe
```

✎ Nota: Se utiliza una versión compatible de Angular 14 y single-spa-angular@7.

```
[root@bpa-77 test-mfe]# ng add single-spa-angular@7 --project test-mfe
Skipping installation: Package already installed
? Does your application use Angular routing? Yes
? What port should your project run on? 4200
    Added 'single-spa' as a dependency
    Added 'single-spa-angular' as a dependency
    Added 'style-loader' as a dependency
    Added '@angular-builders/custom-webpack' as a dependency
    Generated 'main.single-spa.ts
    Generated 'single-spa-props.ts
    Generated asset-url.ts
    Generated extra-webpack.config.js
    Using @angular-builders/custom-webpack builder.
    Updated angular.json configuration
    @angular-builders/custom-webpack:browser
    Warning: Since routing is enabled, an additional manual
    configuration will be required, see https://single-spa.js.org/docs/ecosystem-angular/#configure-routes
CREATE extra-webpack.config.js (303 bytes)
CREATE src/main.single-spa.ts (932 bytes)
CREATE src/app/empty-route/empty-route.component.ts (143 bytes)
CREATE src/single-spa/asset-url.ts (502 bytes)
```

Spa individual en test-mfe

4. Especifique el puerto para el desarrollo local durante la invocación del comando anterior. Se agrega la misma configuración como deployUrl en "angular.json".

✎ Nota: La configuración deployUrl es sólo para desarrollo local.

```
"deployUrl": "http://localhost:4201/"
```

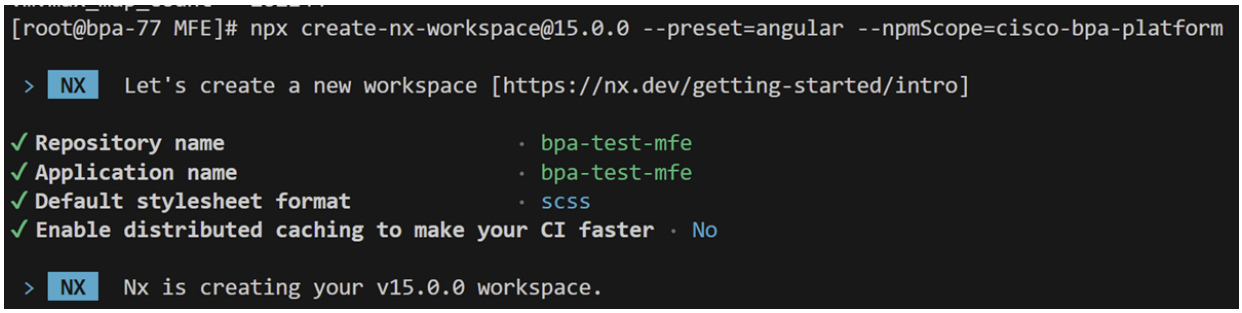
Creación de Aplicaciones Angulares en Nx Monorepo

1. Ejecute el siguiente comando para crear una aplicación Angular dentro del área de trabajo

Extensiones Nrwl (Nx):

```
npx create-nx-workspace@15.0.0 –preset=angular –npmScope=cisco-bpa-platform
```

2. Proporcione los datos proporcionados por el usuario como se muestra en el ejemplo siguiente:



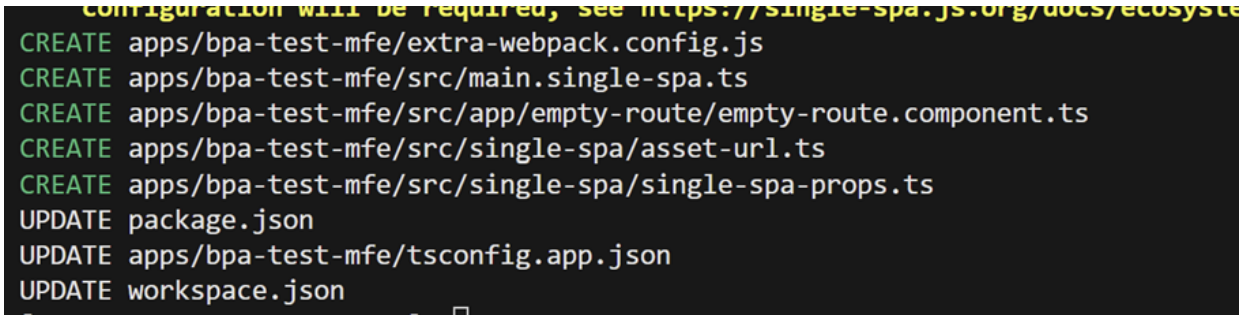
Entradas de usuario

El espacio de trabajo bpa-test-mfe Nx se crea con la aplicación angular bpa-test-mfe.

3. Aplique esquemas "single-spa-angular" para configurar la aplicación angular recién creada. La versión cambia en función de la versión angular.
4. Ejecute el comando `cd` para cambiar a test-mfe.
5. Ejecute el siguiente comando para agregar un único spa en bpa-test-mfe:

```
npm install single-spa-angular@7 \--legacy-peer-deps && npx nx g
single-spa-angular:ng-add \--project bpa-test-mfe
```

Se crean y actualizan los siguientes archivos:



spa único en bpa-test-mfe

6. Especifique el puerto para el desarrollo local durante la invocación del comando anterior. Se agrega la misma configuración como deployUrl en "project.json".

Nota: La configuración deployUrl es sólo para desarrollo local.

```
"deployUrl": "http://localhost:4201",
```

Cambios comunes para aplicaciones angulares y nx angulares

Comprobación de la versión de Webpack

Actualice la versión "@angular-builders/custom-webpack" correspondiente a la versión Angular. La versión del siguiente ejemplo se utiliza para aplicaciones Angular 14.

```
"@angular-builders/custom-webpack": "14.1.0",
```

Actualización del selector

Actualice la plantilla con un selector único "<app-test-mfe-root>" en ./src/main.single-spa.ts.

```
const lifecycles = singleSpaAngular({
  bootstrapFunction: singleSpaProps => {
    singleSpaPropsSubject.next(singleSpaProps);
    return platformBrowserDynamic(getSingleSpaExtraProviders()).bootstrapModule(AppModule);
  },
  template: '<app-test-mfe-root />',
  Router,
  NavigationStart,
  NgZone,
});
```

Actualizando selector

Actualización del Selector en el Componente Raíz

Actualice el mismo nombre del selector "<app-test-mfe-root>" en ./src/app/app.component.ts.

```
{} angular.json M | {} package.json M | TS main.single-spa.ts U | TS app.component.ts ×
MFE > test-mfe > src > app > TS app.component.ts > ...
You, 1 second ago | 1 author (You)
1  import { Component } from '@angular/core';
2
You, 1 second ago | 1 author (You)
3  @Component({
4    selector: 'app-test-mfe-root',
5    templateUrl: './app.component.html',
6    styleUrls: ['./app.component.scss']
7  })
8  export class AppComponent {
9    title = 'test-mfe';
10 }
```

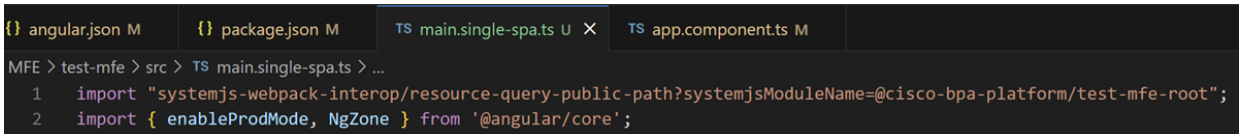
Actualización del Selector en el Componente Raíz

Instalación de systemsjs-web-interop

El paquete NPM "systemsjs-web-interop" se utiliza para establecer dinámicamente la ruta pública para los recursos.

```
systemjs-webpack-interop": "^2.3.7",
```

Instale "systemsjs-web-interop" y agregue la línea siguiente como la primera declaración de importación en el archivo "main.single-spa.ts".



Línea en main.single-spa.ts

systemjsModuleName debe ser único y utilizarse como el nombre que identifica al MFE, como se muestra anteriormente. El cambio anterior ayuda a resolver los activos cargados lentamente desde el host apropiado del MFE.

Actualizando ruta

Incluya las rutas específicas de la aplicación en el archivo "app.routes.ts" de la nueva aplicación angular.

Nota: Se debe incluir la siguiente ruta a "app.routes.ts".

```
{ path: '**',  component: EmptyRouteComponent }
```

Cambios de microservicio para la incorporación de aplicaciones MFE

La mayoría de las aplicaciones de interfaz de usuario están asociadas a un microservicio por diseño. Se deben realizar los siguientes pasos en las siguientes secciones para que el microservicio asociado cargue la aplicación de IU asociada como un MFE.

Definición de detalles de MFE

1. Cree un archivo "mfe.json" en microservice/src/config/mfe.json.

```
[
    {
        "layoutenabled" : true,
        "name" : "@cisco-bpa-platform/test-mfe-root",
        "type": "application",
```

```
        "path": "ex/test-mfe",
        "display_name": "Test MFE",
        "menu_path": "Application",
        "custom": true,
        "default": false,
        "distribution_path": "/assets/cisco-bpa-platform/mw-device-manager/test-mfe/main.js",
        "version": "4.1.0-501"
    }
]
```

2. Añádanse los detalles del MFE que se describen a continuación:

- Local': Indica si esta aplicación es personalizada
- Layout_enabled: Habilitar este atributo si se requiere un encabezado
- Nombre: Nombre único para la aplicación MFE
- Nombre_para_mostrar: Nombre mostrado como elemento de menú en la barra de navegación lateral
- Tipo: Indica la aplicación MFE
- Ruta_menú: Indica el menú de navegación en el que se agrega la aplicación MFE como elemento de menú; funciona en combinación con un atributo personalizado
- Ruta_distribución: Indica el archivo estático que carga la aplicación MFE
- Ruta: Indica la ruta URL a la que está asociada la aplicación MFE
- Predeterminado: Si se establece en true, la ruta de acceso no debe estar presente; si se establece en false, la ruta de acceso está presente

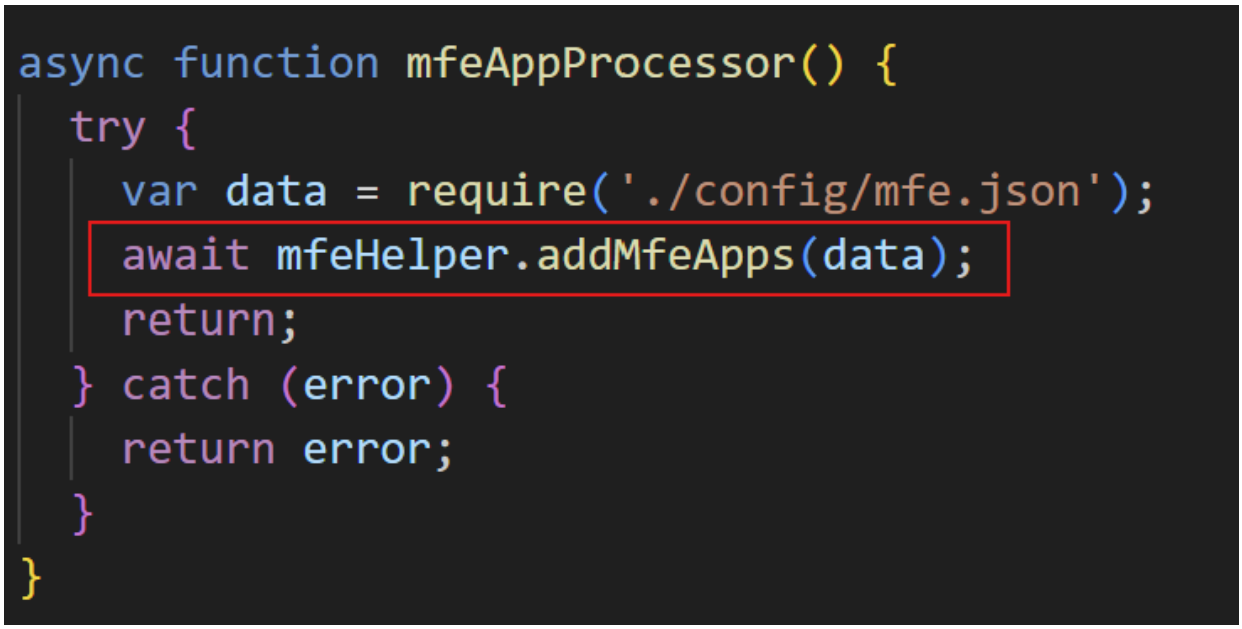
Registro de MFE

Invoque un terminal API principal (por ejemplo, POST mfe/apps) para incorporar los MFE respectivos. Para simplificar el proceso, se crea un método de la utilidad addMfeApps para aceptar la información de MFE. Para registrar los detalles de MFE definidos en el archivo "mfe.json" de BPA.

1. Importe mfeHelper desde la aplicación común en el microservicio "app.js".

```
const { mfeHelper } = require('@cisco-bpa-platform/mw-util-common-app');
```

2. Registre los detalles de MFE con el método addMfeApps(). Esto agrega una entrada en la colección "mongo db core_db -> mfe_apps".



Registrar detalles de MFE

El MFE está listo para la prueba.

Prueba del MFE personalizado

Configuración local

1. Genere la aplicación MFE.
2. Montar la distribución angular de la aplicación en la carpeta assets del contenedor que sirve al microservicio correspondiente (por ejemplo, /root/MFE-app/dist/:/home/node/app/assets).
3. Reinicie el contenedor de microservicios.
4. Actualice el explorador.

Cambios de compilación

1. Durante el proceso de creación de la imagen, actualice el archivo Dockerfile para empaquetar la distribución de la aplicación Angular en la carpeta assets dentro del contenedor. La navegación lateral muestra el menú Test MFE.
2. Haga clic en el menú Test MFE para cargar el MFE (URL https://<host>/ex/test-mfe).

Nota: Al cargar el nuevo MFE, no se muestra la navegación lateral porque no está integrada en el nuevo MFE.

Activación de la navegación lateral

1. Cree un nuevo archivo en la ruta de acceso test-mfe/src/types/decl.d.ts.
2. Añádase la siguiente línea de declaración:

```
declare module '@cisco-bpa-platform/utility-state';
```

3. Abra "test-mfe/extra-webpack.config.js" y agregue `config.externals = ['@cisco-bpa-platform/utility-state'].`

Los pasos anteriores definen @cisco-bpa-platform/utility-state y saltan el error de importación.

4. Verifique e instale las siguientes dependencias:
    - `@angular/cdk: "~14,2,7"`
    - `@angular/material: "~14,2,5"`
    - `"@angular/localize": "^14.2.6"`
    - `"@cisco-bpa-platform/cxui-components": "4.0.1-1003 "`
    - `@cisco-bpa-platform/data-access-services: "4.1.1-1003 "`
    - `"@cisco-bpa-platform/login": "4.0.3-1002 "`
    - `"@cisco-bpa-platform/shared-library": "4.0.1-1001 "`
    - `"@cisco-bpa-platform/spogui-components": "4.1.1-1005 "`
    - `"@cisco-bpa-platform/ui-shared-components": "4.1.1-1005 "`
    - `"@cisco-bpa-platform/util": "4.0.1-1002 "`
    - `"@ngrx/almacén de componentes": "~14.0.0"`
    - `@ngrx/efectos: "~14.0.0"`
    - `@ngrx/entidad: "~14.0.0"`
    - `@ngrx/router-store: "~14.0.0"`
    - `@ngrx/tienda: "~14.0.0"`
    - `"protector de archivos": "^2.0.5"`
    - `Dependencias de desarrollo`
    - `"@types/salvador de archivos": "^2.0.7"`

Nota: Se deben utilizar las versiones correctas para todas las bibliotecas de la plataforma cisco-bpa.

---

5. Actualice los siguientes archivos de nivel raíz como se indica a continuación:
    - “app.component.html”

- "app.component.ts"

```
import { Component, HostListener } from @angular/core';
import { Router } from @angular/router';
import { Observable } from 'rxjs';
import { Loaderservice } from '@cisco-bpa-platform/data-access-services/auth';
@Component({
    selector: 'app-test-mfe-root',
    templateUrl : '. /app. component. html',
    styleUrls: ['./app.component.scss']
})
export class AppComponent {
    title = 'test-mfe';
    showLoader = false;
    loadingSpinner$: Observable
```

```
= this.loaderService.spinnerObsv$; leftNavWidth = ['50px', '250px']; isExpanded = false; i
```

- "app.module.ts"

```
import { HTTP_INTERCEPTORS, HttpClientModule } from '@angular/common/http';
import { NgModule } from '@angular/core' ;
import { BrowserModule } from '@angular/platform-browser';
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
import { LoaderModule } from '@cisco-bpa-platform/cxui-components/loader';
import { AuthEffects, AuthInterceptor, authReducer } from '@cisco-bpa-platform/login/feature-logir
import { SideNavModule } from '@cisco-bpa-platform/spogui-components/side-nav';
import { EffectsModule } from '@ngrx/effects';
import { StoreModule } from '@ngrx/store';
import { environment } from '../environments/environment';
import { AppComponent } from './app.component';
import { AppRoutingModule } from './app-routing.module';

@NgModule({
    declarations : [AppComponent],
    imports: [BrowserModule, AppRoutingModule, HttpClientModule,
        BrowserAnimationsModule,
        HttpClientModule,
        StoreModule. forRoot({ ['auth']: authReducer }, {
            runtimechecks: {
                strictStatelmmutability: false,
                strictActionlmmutability: false
            }
        }),
        EffectsModule.forRoot([AuthEffects]), LoaderModule, SideNavModule],
    providers: [
        { provide: ‘config’, useValue: { apiUrl: environment .apiUrl, mfa: environment .mfa } },
        {
            provide: HTTP_INTERCEPTORS,
            useClass: AuthInterceptor,
            multi: true,
        },
    ],
    bootstrap: [AppComponent]
})
export class AppModule { }
```

- "environment.prod.ts"

```
const URL = `${i window.location.href.substring(0, window.location.href.lastIndexOf('ex'))}bpa` ;
export const environment = {
    production: true,
    appTitle: "BPA | ",
    appName: "Business Process Automation”,
    apiurl: `${URL}`,
    assetsUrl: "./assets”,
    packageUr1: "./assets/packages",
```

```
    mfa: {
        enabled: false,
        o t p_u r 1 : 'https://ssologon-prd.sm.bankofamerica.com/ssoconnect/sendOTP.htm1'
    },
    sync: {
        groups: false,
        users: false
    },
    autoRef reshlnterval : 60000,
    showFormBuilderListltems: false,
    deploymentType: "not_on_prem"
};
```

- "environment.ts"

```
const HOSTNAME = window.location.hostname;
const URL = `http ://${HOSTNAME}: 8000`;
export const environment = {
    production: false,
    appTitle: "BPA | ",
    appName: "Business Process Automation",
    apiUrl: `${URL}`,
    packageUrl: `http://${HOSTNAME}:9090/assets/packages`,
    mfa : {
        enabled: false,
        otp_url: 'https://ssologon-prd.sm.bankofamerica.com/ssoconnect/sendOTP.html'
    },
    sync: {
        groups: false,
        users: false
    },
    autoRefreshinterval: 60000,
    showFormBuilderListltems: false,
    deploymentType: "not_on_prem"
};
```

La navegación lateral está lista para la prueba.

Adición de una nueva ruta en MFE

1. Generar componentes comp-uno y comp-dos.
2. Importe estos componentes en el fichero de routing.

```
import { CompOneComponent } from './comp-one/comp-one.component';
import { CompTwoComponent } from './comp-two/comp-two. component';
```

3. Defina la ruta del prefijo.

```
const includePath = window.location.pathname.substring(1, window.location.pathname.lastIndexOf('/ex'));
```

```
let prefixPath = 'ex';
if (includePath.length > 1) {
    prefixPath = `${includePath}/${prefixPath}`
}
```

4. Defina las rutas. Coloque la ruta predeterminada como last (última).

```
const routes: Routes = [
    { path: `$iprefixPath}/test-mfe/compone`, component: CompOneComponent },
    { path: `${prefixPath}/test-mfe/comptwo`, component: CompTwoComponent },
    { path: '**', component: EmptyRouteComponent }
];
```

5. Revise el código completo.

```
import { APP_BASE_HREF } from '@angular/common';
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { EmptyRouteComponent } from './empty-route/empty-route.component';
import { CompOneComponent } from './comp-one/comp-one.component';
import { CompTwoComponent } from './comp-two/comp-two.component';

const includePath = window.location.pathname.substring(1, window.location.pathname.lastIndexOf('/ex'));
let prefixPath = 'ex';
if (includePath.length > 1) {
  prefixPath = `${includePath}/${prefixPath}`
}
const routes: Routes = [
  { path: `${prefixPath}/test-mfe/compone`, component: CompOneComponent },
  { path: `${prefixPath}/test-mfe/comptwo`, component: CompTwoComponent },
  { path: '**', component: EmptyRouteComponent }
];

You, 5 minutes ago | 1 author (You)
@NgModule({
  imports: [RouterModule.forRoot(routes, { enableTracing: true })],
  exports: [RouterModule],
  providers: [
    { provide: APP_BASE_HREF, useValue: '/' },
  ],
})
export class AppRoutingModule { }
```

Revisar código

Representación de un menú anidado para aplicaciones personalizadas

Un menú anidado se puede representar desde una aplicación personalizada. Esta sección describe los pasos necesarios para preparar el documento de registro de MFE. Para ello, se han introducido las siguientes propiedades:

- Nivel: Acepta números, empezando por uno; el número aumenta según el número de menús y el nivel; los niveles más bajos deben registrarse primero en BPA
- Padres: Almacena el valor de la propiedad display_name principal bajo la que debe

mostrarse el menú actual

Ejemplo de un menú anidado creado para la aplicación Test MFE custom MFE:

Menú Anidado

Para procesar un menú anidado:

1. Agregue valores correctos para las propiedades level, path y parent.

```
"layout_enabled": true,
"name": "@cisco-bpa-platform/test-mfe-root",
"type": "application",
"display_name": "Test MFE",
"menu_path": "Application",
"custom": true,
"default": false,
"level": 1,
"distribution_path": "/assets/cisco-bpa-platform/mw-device-manager/test-mfe/main.js",
"version": "4.1.0-501"
```

La propiedad path no debería estar presente si un menú tiene menús secundarios. El nivel 1 indica que este menú debe registrarse en primer lugar.

```
"layout_enabled": true,
"name": "@cisco-bpa-platform/test-mfe-root",
"type": "application",
"display_name": "Test MFE",
"menu_path": "Application",
"custom": true,
"default": false,
"level": 1,
"distribution_path": "/assets/cisco-bpa-platform/mw-device-manager/test-mfe/main.js",
"version": "4.1.0-501"
```

En la figura anterior, el valor primario es Test MFE; por lo tanto, el menú Test MFE Level One se anida en el menú Test MFE.

Nivel 2

```
"layout_enabled": true,
"name": "@cisco-bpa-platform/test-mfe-root",
"type": "application",
"path": "ex/test-mfe/compone",
"display_name": "Test MFE One",
"menu_path": "Application",
"parents": "Test MFE",
"custom": true,
"default": false,
"level": 2,
"distribution_path": "/assets/cisco-bpa-platform/mw-device-manager/test-mfe/main.js",
"version": "4.1.0-501"
```


Nivel 4


```
"layout_enabled": true,
"name": "@cisco-bpa-platform/test-mfe-root",
"type": "application",
"path": "ex/test-mfe/comptwo",
"display_name": "Test MFE Two",
"menu_path": "Application",
"parents": "Test MFE",
"custom": true,
"default": false,
"level": 4,
"distribution_path": "/assets/cisco-bpa-platform/mw-device-manager/test-mfe/main.js",
"version": "4.1.0-501"
```


Nivel 5


```
"layout_enabled": true,
"name": "@cisco-bpa-platform/test-mfe-root",
"type": "application",
"path": "ex/test-mfe/compone",
"display_name": "Test MFE First one",
"menu_path": "Application",
"parents": "Test MFE Level One",
"custom": true,
"default": false,
"level": 5,
"distribution_path": "/assets/cisco-bpa-platform/mw-device-manager/test-mfe/main.js",
"version": "4.1.0-501"
```


El menú Test MFE First One se anida en el menú Test MFE Level One.

Nivel 6


```
"layout_enabled": true,
```

```
"name": "@cisco-bpa-platform/test-mfe-root",
"type": "application",
"path": "ex/test-mfe/comptwo",
"display_name": "Test MFE First Two",
"menu_path": "Application",
"parents": "Test MFE Level One",
"custom": true,
"default": false,
"level": 6,
"distribution_path" : "/assets/cisco-bpa-platform/mw-device-manager/test-mfe/main.js",
"version": "4.1.0-501"
```


El menú Test MFE First Two se anida en el menú Test MFE Level One.

Después de preparar el archivo "mfe.json" en el paso 1, reinicie el contenedor de microservicios para registrar las entradas.

Acerca de esta traducción
Cisco ha traducido este documento combinando la traducción automática y los recursos humanos a fin de ofrecer a nuestros usuarios en todo el mundo contenido en su propio idioma.

Tenga en cuenta que incluso la mejor traducción automática podría no ser tan precisa como la proporcionada por un traductor profesional.

Cisco Systems, Inc. no asume ninguna responsabilidad por la precisión de estas traducciones y recomienda remitirse siempre al documento original escrito en inglés (insertar vínculo URL).