Componente .NET de integración con MACO

Dirección General de Tecnologías de la Información y Comunicaciones

Áreas de Gobierno Tecnológico de SSII y del Dato

Servicio Andaluz de Salud
Consejería de Sanidad, Presidencia y Emergencias

Contenido

Resumen

- **Versión:** v01r06
- **Fecha publicación: 19 may 2021**
- **Entrada en vigor desde:** 2 jun 2017

blocked URL

Cumplimiento normativo

Este documento es una guia de desarrollo en continua evolución. Para cualquier duda o sugerencia por favor contante con nosotros.
Contacto Arquitectura: l-arquitectura.stic@juntadeandalucia.es

Histórico de cambios

Versión	2.5.0 Upgrade	**Fecha publicación**	10 may 2020	**Fecha entrada en vigor**	10 may 2021
Alcance					

- Se realiza una reestructuración del proyecto para realizar una división en subcomponentes. De este modo se permite la instalación separada de las implementaciones WebService y Oracle del componente para el acceso a datos de Maco según sea la necesidad.

  Los componentes generados han sido los siguientes:

  - SAS.MacoApiClient: Es el proyecto core de la API. Contiene: diferentes tipos proporcionados por la API, como MacoUser, Person, Professionale etc, e interfaces para el acceso de datos de Profesionales, Operadores, Módulos y Autenticación.
  - SAS.MacoApiClient.Exceptions: Manejo de excepciones de negocio de Maco.
  - SAS.MacoApiClient.Ticket: Gestión de ticket
  - SAS.MacoApiClient.WebService: Implementación de acceso a los datos de Maco vía servicio web.

- SAS.MacoApiClient.ManagedDataAccess.Oracle: Implementación vía Réplica de BBDD para acceder en Maco a datos de módulo y perfil. En caso de que no se requieran datos de módulos y perfil no hay que instalar este componente.

Se ha cambiado el nombre a la interfaz **IMacoUserManager** y a la clase **MacoUserManager** por **IMacoUserTransformer y MacoUserTransformer** respectivamente.

Se ha cambiado el nombre de las clases MacoOperatorManager, MacoProfessionalManager, MacoAuthentication por WSMacoOperatorManager, WSMacoProfessionalManager y WSMacoAuthentication respectivamente.

El método RSA(string password, string rsaKey) , que se encontraba en la versión 2.4.6 en la clase MacoUtil, se ha trasladado a la clase TicketUtil del componente SAS.MacoApiClient.Ticket, dentro del espacio de nombre SAS.MacoApiClient.Ticket.Utils.

El tipo EndPointType, en el que se define los tipos de endpoints de Maco permitidos por la API, se ha llevado al espacio de nombre SAS.MacoApiClient.WebService.Api.Entity.EndPointType.

Versión	2.4.6 Bug fix	**Fecha publicación**	16 may 2021	**Fecha entrada en vigor**	16 may 2021
Alcance					

- Se añade método RSA(string password, string rsaKey) en la clase MacoUtil el cual encripta una cadena en format RSA, usando la clave pública de Maco. Esta cadena encriptada en formato RSA es el password que se pasa en el método ChangeUserPassword, para cambiar el password del operador. También se corrige error por mapeo de excepciones en el médodo ThrowException de la clase MacoExceptionManager.

Versión	2.4.5 Bug fix	**Fecha publicación**	11 mar 2021	**Fecha entrada en vigor**	11 mar 2021
Alcance					

- Se corrige error del mapping entre el maco mensaje CambioClaveOperadorMessage y su endpoint

Versión	2.4.4 Bug fix	**Fecha publicación**	7 ene 2021	**Fecha entrada en vigor**	7 ene 2021
Alcance					

- Se añade información de la versión de la API MacoApiClient que realiza la llamada a los servicios Web de Maco en el campo User-Agent de la petición.

Versión	2.4.3 Bug fix	**Fecha publicación**	8 jun 2020	**Fecha entrada en vigor**	8 jun 2020
Alcance					

- Se añade modo de Seguridad Básica para la comunicación HTTPS de los servicios web de Maco
  - Debido a que el protocolo de comunicación HTTP de los servicios web de maco han cambiado a HTTPS, se ha añadido la posibilidad de comunicación de transporte con cliente anónimo.

Versión	2.4.2 Bug fix	**Fecha publicación**	5 mar 2020	**Fecha entrada en vigor**	5 mar 2020
Alcance					

- Se actualiza la versión del componente SAS.Common.Kernel a v1.1.0

Versión	2.4.1 Bug fix	**Fecha publicación**	19 feb 2020	**Fecha entrada en vigor**	19 feb 2019
Alcance					

- Se actualiza la versión del componente SAS.Common.Oracle a v1.3.3, en el que se actualiza los componentes Oracle.ManagedDataAccess y Oracle.ManagedDataAccess.EntityFramework a la versión 19.6.0

Versión	2.4.0 Upgrade	Fecha publicación	21 nov 2019	Fecha entrada en vigor	21 nov 2019

Alcance

- Se añade la versión réplica BBDD del método FindOperator
  - **Proyecto SAS.MacoApiClient** :
    - src/Api/Boundary/
      - **MacoOperatorReplicaManager**: Implementa el método FindOperator de la interfaz IMacoOperatorManager en su versión réplica de BBDD.
  - **Proyecto SAS.MacoApiClient.ManagedDataAccess.Oracle** :
    - /Boundary
      - **MacoRepository**: Se añade método GetOperadorByDirayaCode en el que se obtiene los datos del profesional u operador a traves de su código Diraya. Si devuelve el CNP indica que el usuario que se obtiene es un profesional. En el caso de que el número de profesional devuelto sea null, se está recuperando una persona operador (no profesional).

        public IList GetOperadorByDirayaCode(string dirayaCode)

Versión	2.3.0 Upgrade	Fecha publicación	7 oct 2019	Fecha entrada en vigor	7 oct 2019

Alcance

- Se añade posibilidad de configurar tipo de contexto utilizado (manejado o no manejado)
- **Proyecto SAS.MacoApiClient.ManagedDataAccess.Oracle** :
  - Boundary/
    - **MacoContextConnection**: Se implementa el nuevo método GetIsManagedContext de la interfaz IConfigContextApiConnection, el cual devuelve si para la configuración del tipo de esquema que se pasa por parámetro se utiliza contexto manejado o no.

```
                                        public bool GetIsManagedContext
(SchemaMacoType tipo)
{
    return this.configuracion.FirstOrDefault(d => d.ShemaType == tipo).
IsManagedContext;
}
```

- **MacoRepository**: Se modifica el método GetConexion() para que, dependiendo del parámetro IsManagedContext, de la configuración, se devuelva una conexión Maco manejada o no.

Versión	2.2.0 Upgrade	Fecha publicación	1 oct 2019	Fecha entrada en vigor	1 oct 2019
Alcance					

- Se añade posibilidad de contexto no manejado
  - **Proyecto SAS.MacoApiClient.ManagedDataAccess.Oracle** :
    - Entity
      - **MacoConexion**: Se modifica para que no utilice configuración por código en la definición del contexto, es decir, utiliza contexto no manejado: su configuración hay que definirla en fichero de configuración.
      - **MacoManagedConexion**: Conexión manejado del contexto Maco. En la declaración de la clase se define su tipo de configuración. En este caso: [DbConfigurationType(typeof (ConfiguracionOracleProvider))]

Versión	2.1.0 Upgrade	Fecha publicación	18 sept 2019	Fecha entrada en vigor	19 sept 2019
Alcance					

- Se añade implementación por réplica de BBDD para la consulta de módulos y perfil.
- Todo sistema que utilice el modo réplica de BBDD tiene que tener acceso a las tablas de Maco siguientes: ModuloEntity y PerfilModuloEntity .

Versión	2.0.2 Bug fix	Fecha publicación	28 ago 2019	Fecha entrada en vigor	28 ago 2019
Alcance					

- Se sustituye en la definición del formato de fecha 'dd/MM/yyyy HH:mm:ss' , el HH por H, para que se contemple la hora tanto para uno como dos dígitos.

Versión	2.0.1 Bug fix	Fecha publicación	31 jul 2019	Fecha entrada en vigor	31 jul 2019
Alcance					

- Actualizar la versión de las librerías utilizadas del namespaces System.ServiceModel a la 4.5.3

Versión	2.0.0 Update	Fecha publicación	25 jun 2019	Fecha entrada en vigor	25 jun 2019
Alcance					

- Versión inicial .NET Standard 2.0. Se refactoriza el proyecto a la versión .NET Standard del .NET Framework.

Versión	1.7.6 Bug fix	Fecha publicación	16 abr 2021	Fecha entrada en vigor	16 abr 2021
Alcance					

Se añade método RSA(string password, string rsaKey) en la clase MacoUtil el cual encripta una cadena en format RSA, usando la clave pública de Maco. Esta cadena encriptada en formato RSAes el password que se pasa en el método ChangeUserPassword, para cambiar el password del operador. También se corrige error por mapeo de excepciones en el médodo ThrowException de la clase MacoExceptionManager.

Versión	1.7.5 Bug fix	Fecha publicación	11 mar 2021	Fecha entrada en vigor	11 mar 2021
Alcance					

- Se corrige error del mapping entre el maco mensaje CambioClaveOperadorMessage y su endpoint

Versión	1.7.4 Bug fix	Fecha publicación	21 oct 2020	Fecha entrada en vigor	21 oct 2020
Alcance					

- Corregido bug en el loging con Maco en el que el campo ticket era nulo.

Versión	1.7.3 Upgrade	Fecha publicación	9 nov 2020	Fecha entrada en vigor	9 nov 2020
Alcance					

- Versión de la librería de tipo .NET Framework 4.5.2. Permite el acceso a los datos de Maco utilizando el servicio web de Maco.

  Los cambios realizados en esta versión han sido los siguientes

  - Añadido posibilidad de configuración del servicio web de Maco vía HTTPS.
  - Se sustituye en la definición del formato de fecha 'dd/MM/yyyy HH:mm:ss' , el HH por H, para que se contemple la hora tanto para uno como dos dígitos.
  - Se añade información de la versión de la API que realiza la llamada a los servicios Web de Maco en el campo User-Agent de la petición.

SAS.MacoApiClient

Introducción

En el presente documento se presenta la librería **SAS.MacoApiClient**, que permite el acceso a datos de Maco.

Dispone de dos versiones, .NET Framework 4.5.2 y .NET Standard 2.0:

- **.NET Framework 4.5.2**: Permite el acceso a los datos de MACO utilizando el servicio web. **Versión de la API 1.7.6**
- **.NET Standard 2.0**: Permite el acceso a los datos de Maco utilizando dos implementaciones diferentes: una que utiliza internamente el servicio web de Maco y otra que utiliza réplica de BBDD. **Versión de la API >= 2.0.0**

La versión actual de la librería .NET Standard , en su versión acceso a réplica de BBDD, solo dispone de métodos para acceso a la información de módulos y perfiles.

Se recomienda la instalación de la versión de la librería .NET Standard 2.0, en su versión Servicio Web, compatible con .NET Framework desde la versión 4.6.1 de .NET Framework. por su actualización más periódica.

Dependencias

Se define las siguientes dependencia de la librería:

Versión API 1.7.6 y Versión API >= 2.0.0

Nombre del componente	Versión	Descripción
System.ServiceModel.Duplex	4.5.3	Proporciona clases para consumir y comunicarse con servicios en modo Dúplex
System.ServiceModel.Http	4.5.3	Gestiona solicitudes realizadas a través del protocolo HTTP
System.ServiceModel.NetTcp	4.5.3	Genera de forma predeterminada una pila de comunicación en tiempo de ejecución, que utiliza la seguridad de transporte, TCP para la entrega de mensajes
System.ServiceModel.Security	4.5.3	Se encarga de temas generales relacionados con la seguridad

Versión API >= 2.0.0

Nombre del componente	Versión	Descripción
SASUtils.Xml	1.0.1	Componente .NET común para tratamiento de XML
SAS.Common.Core	1.1.0	Componente .NET común con clases e interfaces core de aplicaciones
SAS.Common.Kernel	1.1.0	Componente .NET común kernel de aplicaciones
SAS.Common.OracleProvider	1.3.3	Componente .NET común de configuración de proveedor de datos Oracle
EntityFramework	6.2.0	ORM de Microsoft para .NET
Oracle.ManagedDataAccess	19.6.0	Proveedor de acceso a datos de Oracle para .NET
Oracle.ManagedDataAccess.EntityFramework	19.6.0	ORM de Oracle para .NET

Errores conocidos

- Bug al parsear fecha en formato cadena a un formato DateTime determinado.

  Problema al parsear una fecha en formato cadena, en la que la hora solo contiene un dígito. *Corregido en la versión 1.7.3 y desde de la version 2.0.2*
- *Bug al hacer loging con Maco: ticket devuelto nulo. Corregido en la versión 1.7.4.*

Integración de la librería

Configurar en Visual Studio un nuevo origen de paquetes Nuget (clic en “Administrar Paquetes Nuget …” del proyecto al que se desea añadir la referencia el componente MacoApiClient) indicando en el campo Origen la dirección de repositorios de componentes .NET del SAS: http://net.lib.repository.alm.sas.junta-andalucia.es/.

Versión API < 2.5.0

- Instalar el componente SAS.MacoApiClient.

Versión API >= 2.5.0

- Instalar el componente **SAS.MacoApiClient.WebService** para el acceso a datos de Maco utilizando los servicios web de Maco y validación de tickets,firma, operaciones y fecha del mensaje.
- Solo en el caso de que se necesite consultar datos de módulos y perfil, instalar el componente **SAS. MacoApiClient.ManagedDataAccess.Oracle**.
- En el caso de que solo se quiera utilizar la funcionalidad de validar ticket, firma, operaciones y fecha del mensaje, instalar **SAS.MacoApiClient.Ticket**.

Configuración

Prerequisitos

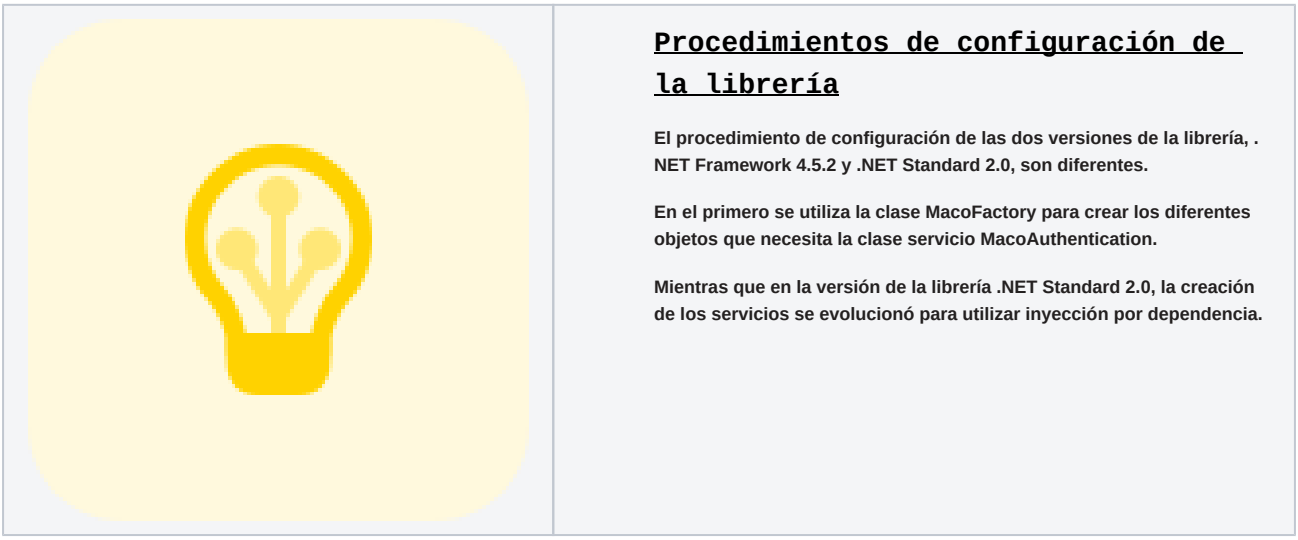
Previo a la configuración de la librería se debe de disponer de los siguientes datos proporcionados por la Oficina Técnica de Interoperabilidad:

- **Clave pública de Maco,** tanto de PRE como de PRO. Se corresponde con una clave de encriptación en formato RSA: <RSAKeyValue><Modulus>...</RSAKeyValue>
- **Código del módulo del sistema en Maco**. Se corresponde con el código en el que está identificado el sistema en Maco.
- **Endpoints de los servicios web de Maco**, tanto de PRE como de PRO. En el caso de PRE, son los siguientes:
  - https://servicios.pre.sas.junta-andalucia.es/maco/wsValidacion.asmx Servicio web de validación de Maco.
  - https://servicios.pre.sas.junta-andalucia.es/maco /wsOperador.asmx Servicio web para la gestión de datos de Operadores de Maco.
  - https://servicios.pre.sas.junta-andalucia.es/maco /wsProfesional.asmx Servicio web para la gestión de datos de Profesionales de Maco.

Otros parámetros de configuración

El validador de ticket requiere la configuración de los siguientes parámetros:

- RsaKey: Se corresponde con la Clave pública de Maco.
- ReferenceDate: Fecha de referencia utilizada para conocer si un ticket ha expirado
- FutureTicketToleranceMilli: Se utiliza para la validación de tolerancia de ticket futuro. Si su valor es inferior a 0, no se realiza validación de ticket futuro. Si su valor es mayor que 0, indica el número de milisegundos que se tolera en la validación de la fecha del ticket futuro. Si no se desea ticket a futuro, su valor se establecerá a 0.
- TicketTtlMillis: Se utiliza para validar la expiración de un ticket. Si ReferenceDate - TicketTtlMillis es superior a la fecha del ticket, se considera que el ticket ha expirado.
- TicketValidations. Listado de tipos de validación que se le aplica al ticket. Pueden ser los siguientes:
    - Expiración (TicketValidationType.Expired). Valida si el ticket ha expirado
    - Futuro (TicketValidationType.Future). Valida si se ha recibido un ticket con fecha futura y no cumple la tolerancia definida en el campo FutureTicketToleranceMilli.
    - Permisos (TicketValidationType.Permission). Valida que el ticket tiene los permisos adecuados para poder hacer la operación.
    - Firma (TicketValidationType.Signature). Valida que la firma del ticket sea correcta.



Procedimiento de configuración a partir de la versión 1.7.3

Cómo ejemplo de configuración, la clase UserService provee un servicio de autenticación y autorización contra el servicio de MACO.

En el ejemplo siguiente se utiliza el fichero de configuración app.config para definir los parámetros de configuración de Maco: clave pública de Maco, ruta principal servicio web de Maco y endpoints de Validación, Operador y Profesional de Maco.

1 Configurar parámetros en fichero de configuración

```
        <appSettings>
<add key="MacoXmlKey" value="CLAVE_PUBLICA_DE_MACO" />
<add key="MacoModule" value="CODIGO_EN_MACO_DEL_SISTEMA" />
<add key="ServiceAddressMaco" value="https://servicios.pre.sas.junta-andalucia.es/" />
<add key="EndPointAddressMacoValidacion" value="/maco/wsValidacion.asmx" />
<add key="EndPointAddressMacoOperador" value="/maco/wsOperador.asmx" />
<add key="EndPointAddressMacoProfesional" value="/maco/wsProfesional.asmx" />
</appSettings>
```

Se consulta la clave pública de Maco del fichero de configuración

2 Configurar clave pública de MACO

```
        string _xmlMacoKey = ConfigurationManager.AppSettings.Get("MacoXmlKey");
```

Se crea el objeto de configuración de Maco, MacoApiConfig. La url que recibe como parámetro se consultará de fichero de configuración o BBDD. En el ejemplo se ha utilizado la ruta inicial de PRE.

3 Configurar la api de Maco

```
        string serviceAddressMaco= ConfigurationManager.AppSettings.Get("ServiceAddressMaco");
MacoApiConfig macoApiConfig = new MacoApiConfig(serviceAddressMaco);
```

Se consulta el código de módulo asignado al sistema en Maco.

4 Configurar código Maco del sistema

```
        macoApiConfig.ModuleCode = ConfigurationManager.AppSettings.Get("MacoModule");
```

Se define los enpoints de los diferentes servicios web de Maco en el objeto macoApiConfig: Validación, Operador y Profesionales.

5 Configurar de los endpoints de acceso a los servicios de Maco

```
        string endPointAddressMacoValidacion = ConfigurationManager.AppSettings.Get
("EndPointAddressMacoValidacion");
string endPointAddressMacoOperador = ConfigurationManager.AppSettings.Get("EndPointAddressMacoOperador");
string endPointAddressMacoProfesional = ConfigurationManager.AppSettings.Get
("EndPointAddressMacoProfesional");


macoApiConfig.EndPoints = new Dictionary<MacoApiClient.Api.Entity.EndPointType, string>
```

```
{
         { MacoApiClient.Api.Entity.EndPointType.Validacion,  endPointAddressMacoValidacion  },
         { MacoApiClient.Api.Entity.EndPointType.Operador,  endPointAddressMacoOperador },
         { MacoApiClient.Api.Entity.EndPointType.Profesional,  endPointAddressMacoProfesional  }
};
```

Se define la configuración del validador de ticket. En el ejemplo, se comprueba que la fecha del ticket sea de 24 horas máximo y no se permite ticket a futuro (FutureTicketToleranceMillis =0).

6 Configurar validador de ticket

```
        TicketValidatorConfig ticketConfig = new TicketValidatorConfig()
            {
                RsaKey = _xmlMacoKey,
                ReferenceDate = DateTime.Now,
                FutureTicketToleranceMillis = 0,
                TicketTtlMillis = 24 * 60 * 60 * 1000,
                TicketValidations = new HashSet<TicketValidationType>()
            };
```

Se añaden los diferentes tipos de validaciones. En el ejemplo, validación de firma, expiración de ticket, ticket futuro y permisos.

7 Configurar los tipos de validación

```
        ticketConfig.TicketValidations.Add(TicketValidationType.Signature); //SignatureTicketValidation:
Valida que la firma es correcta
ticketConfig.TicketValidations.Add(TicketValidationType.Expired); //ExpiredTicketValidation: Valida que
el ticket no esté caducado
ticketConfig.TicketValidations.Add(TicketValidationType.Future); //FutureTicketValidation: Valida que el
ticket no sea futuro
ticketConfig.TicketValidations.Add(TicketValidationType.Permission); //PermissionTicketValidation:
Valida que el permiso requerido se encuentre en el ticket
```

Se inicializa un objeto de tipo MacoFactory, el cual recibe un objeto configuración de la api y del ticket. Permite crear el resto de objetos necesarios para la utilización de la API.

8 Creación del objeto MacoFactory

```
        factory = new MacoFactory(macoApiConfig, ticketConfig);
```

Se crea el objeto de autenticación de Maco, el cual permite realizar la autenticación o comprobar la autorización de un módulo concreto de Maco.

9 Creación del objeto de autenticación de Maco

```
        macoAuthentication = new MacoAuthentication(factory.MacoUserManager, macoApiConfig, factory.
TicketValidator, factory.WsMacoManager, factory.MacoExceptionManager);
```

A continuación se presenta el código completo del ejemplo:

10 Ejemplo completo configuración de la versión 1.7.6

```
         public class UserService
{
        MacoFactory factory;
        MacoAuthentication macoAuthentication;
        public UserService()
        {
            string _xmlMacoKey = "CLAVE_PUBLICA_MACO";

            MacoApiConfig macoApiConfig = new MacoApiConfig("https://servicios.pre.sas.junta-andalucia.es
/");
            macoApiConfig.ModuleCode = "MODULO_MACO_DEL_SISTEMA";
            macoApiConfig.EndPoints = new Dictionary<MacoApiClient.Api.Entity.EndPointType, string>
                   {
                       { MacoApiClient.Api.Entity.EndPointType.Validacion, @"/maco/wsValidacion.asmx" },
                       { MacoApiClient.Api.Entity.EndPointType.Operador, @"/maco/wsOperador.asmx"},
                       { MacoApiClient.Api.Entity.EndPointType.Profesional, @"/maco/wsProfesional.asmx" }
                   };

            TicketValidatorConfig ticketConfig = new TicketValidatorConfig()
            {
                RsaKey = _xmlMacoKey,
                ReferenceDate = DateTime.Now,
                FutureTicketToleranceMillis = -1,
                TicketTtlMillis = 24 * 60 * 60 * 1000,
                TicketValidations = new HashSet<TicketValidationType>()
            };
            ticketConfig.TicketValidations.Add(TicketValidationType.Signature);
//SignatureTicketValidation: Valida que la firma es correcta
            ticketConfig.TicketValidations.Add(TicketValidationType.Expired); //ExpiredTicketValidation:
Valida que el ticket no este caducado
            ticketConfig.TicketValidations.Add(TicketValidationType.Future); //FutureTicketValidation:
Valida que el ticket no sea futuro
            ticketConfig.TicketValidations.Add(TicketValidationType.Permission);
//PermissionTicketValidation: Valida que el permiso requerido se encuentre en el ticket

            factory = new MacoFactory(macoApiConfig, ticketConfig);
            macoAuthentication = new MacoAuthentication(factory.MacoUserManager, macoApiConfig, factory.
TicketValidator, factory.WsMacoManager, factory.MacoExceptionManager);
        }
}
```

Procedimiento de configuración a partir de la versión 2.0.0

La configuración de la API depende del tipo de implementación que se vaya a utilizar, según se acceda a los datos mediante el Servicio Web de Maco o mediante réplicas de base de datos. Veamos cada una de ellas:

En ambos ejemplos se utiliza como base un proyecto .NET MVC.

Primero, en el fichero Startup.cs, método ConfigureServices, se inicializa el contenedor de inyección de dependencia:

11 Inicializar contenedor de inyección de dependencia ContainerDI

```
         public void ConfigureServices(IServiceCollection services)
        {
```

```
          ...
          ContainerDI.SetAsyncScopedLifestyle(); // establecer estilo de vida por defecto del contenedor
de inyección. Permite procesar tanto operaciones asíncronas como síncronas
          ContainerDI.RegisterServices(ref services);
        ...
      }
```

ContainerDI, clase del componente común SAS.Util.Injection, el cual permite el registro por inyección de dependencia de los diferentes servicios y tipos utilizados en el sistema.

Configuración de la API versión Servicio Web

En el fichero appsettings se definen secciones en las que se establece configuración de Maco: clave pública de maco y módulo del sistema en Maco, y servicio web de Maco: dirección del servicio web de Maco y los endpoints de Validación, Operador y Profesional

12 Configurar parámetros Maco en fichero de configuración appsettings.json

```
        "MacoApiClient": {
        "XmlKey": "CLAVEPUBLICAMACO",
        "Module": "MODULOMACO"
},
"WebServiceLocator": {
        "ServiceAddressMaco": "https://servicios.pre.sas.junta-andalucia.es/",
        "EndPointAddressMacoValidacion": "/maco/wsValidacion.asmx",
        "EndPointAddressMacoOperador": "/maco/wsOperador.asmx",
        "EndPointAddressMacoProfesional": "/maco/wsProfesional.asmx"
},
```

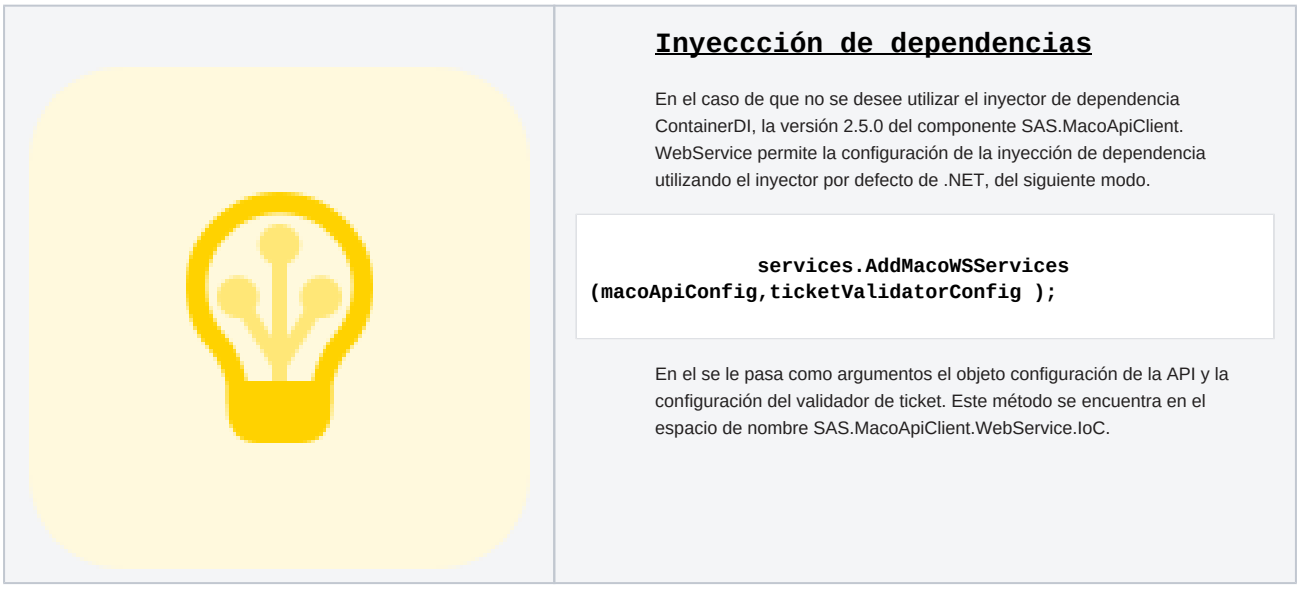
En el fichero Startup.cs, método Configure(IApplicationBuilder app, IHostingEnvironment env), añadir la siguiente configuración de los controladores , viewcomponents de MVC y la llamada al nuevo método InitializeContainer :

13 Registrar controladores y ViewComponents

```
         public void Configure(IApplicationBuilder app, IHostingEnvironment env)
        {
            ContainerDI.RegisterMvcControllers(ref app);
            ContainerDI.RegisterMvcViewComponents(ref app);

            this.InitializeContainer();
           ...
        }
```

A continuación, en el fichero Startup.cs, crear el método void InitializeContainer() y añadir la siguiente configuración para definir la configuración del validador de ticket, configuración con los endpoints de los servicios web de Maco y registrar por inyección de dependencia los diferentes servicios utilizados de la API:



Inyeccción de dependencias

En el caso de que no se desee utilizar el inyector de dependencia ContainerDI, la versión 2.5.0 del componente SAS.MacoApiClient.WebService permite la configuración de la inyección de dependencia utilizando el inyector por defecto de .NET, del siguiente modo.

```
services.AddMacoWSServices
(macoApiConfig,ticketValidatorConfig );
```

En el se le pasa como argumentos el objeto configuración de la API y la configuración del validador de ticket. Este método se encuentra en el espacio de nombre SAS.MacoApiClient.WebService.IoC.

Configuración de la validación del ticket de Maco. Se define validación de firma, fecha del ticket y permisos

14 Configurar validador de Ticket

```
        /// <summary>
/// Inicializador de contenedor utilizando el inyector de dependencia ContainerDI.
/// </summary>
private void InitializeContainer()
{
        var ticketValidatorConfig = new TicketValidatorConfig
        {
                RsaKey = ConfigurationManager.AppSettings["MacoApiClient:XmlKey"],
                ReferenceDate = DateTime.Now,
                FutureTicketToleranceMillis = -1,
                TicketTtlMillis = 24 * 60 * 60 * 1000,
                TicketValidations = new HashSet<TicketValidationType>()
        };

        ticketValidatorConfig.TicketValidations.Add(TicketValidationType.Signature); //
SignatureTicketValidation: Valida que la firma es correcta
        ticketValidatorConfig.TicketValidations.Add(TicketValidationType.Expired); //
ExpiredTicketValidation: Valida que el ticket no este caducado
        ticketValidatorConfig.TicketValidations.Add(TicketValidationType.Future); //
FutureTicketValidation: Valida que el ticket no sea futuro
        ticketValidatorConfig.TicketValidations.Add(TicketValidationType.Permission); //
PermissionTicketValidation: Valida que el permiso requerido se encuentre en el ticket
```

Configuración de la api de Maco, en la que se le pasa la dirección del servicio web de Maco y el código del módulo. También se define los endpoints de los servicios web de Maco.

15 Configurar api de Maco

```
        var macoApiConfig = new MacoApiConfig(this.Configuration.GetSection("WebServiceLocator:
ServiceAddressMaco").Value)
{
        ModuleCode = this.Configuration.GetSection("MacoApiClient:Module").Value
};

   macoApiConfig.EndPoints = new Dictionary<MacoApiClient.WebService.Api.Entity.EndPointType, string>
   {
            { MacoApiClient.WebService.Api.Entity.EndPointType.Validacion, Configuration.GetSection
("WebServiceLocator:EndPointAddressMacoValidacion").Value },
            { MacoApiClient.WebService.Api.Entity.EndPointType.Operador, Configuration.GetSection
("WebServiceLocator:EndPointAddressMacoOperador").Value},
            { MacoApiClient.WebService.Api.Entity.EndPointType.Profesional, Configuration.GetSection
("WebServiceLocator:EndPointAddressMacoProfesional").Value }
   };
```


Breaking changes

En la versión previa a la 2.5.0, el enumerado EndPointType tenía el espacio de nombre SAS.MacoApiClient.Api.Entity.

Registro del tipo TicketValidator que implementa la interfaz ITicketValidator. Se encarga de gestionar la validación de tickets.

16 Registrar tipo TicketValidator

```
        ContainerDI.Register<ITicketValidator>(() => new TicketValidator(ticketValidatorConfig),
ContainerDI.LifeStyleContainer.Transient);
```

Registro del tipo MacoExceptionManager que implementa la interfaz IMacoExceptionManager. Se encarga de gestionar las excepciones y errores informados del servicio web de Maco.

17 Registrar tipo MacoExceptionManager

```
        ContainerDI.Register<IMacoExceptionManager, MacoExceptionManager>(ContainerDI.LifeStyleContainer.
Transient);
```

Registro del tipo MacoUserTransformer que implementa la interfaz IMacoUserTransformer. Se encarga de transformar los mensajes XML recibidos del servicio web de MACO a objetos.

18 Registrar tipo MacoUserTransformer

```
        ContainerDI.Register<IMacoUserTransformer, WSMacoUserTransformer>(ContainerDI.LifeStyleContainer.
Transient);
```


Breaking changes

En la versión 2.5.0, se ha cambiado el nombre a la interfaz IMacoUserManager y a la clase MacoUserManager por IMacoUserTransformer y WSMacoUserTransformer respectivamente.

Registro del tipo MacoApiConfig que implementa la interfaz IMacoApiConfig. Contiene la información de configuración de la API.

19 Registrar tipo MacoApiConfig

```
         ContainerDI.Register<IMacoApiConfig>(() => macoApiConfig, ContainerDI.LifeStyleContainer.
Transient);
```

Registro del tipo WsMacoManager que implementa la interfaz IWsMacoActions. Es la clase wrapper encargada de llamar a los servicios web de Maco.

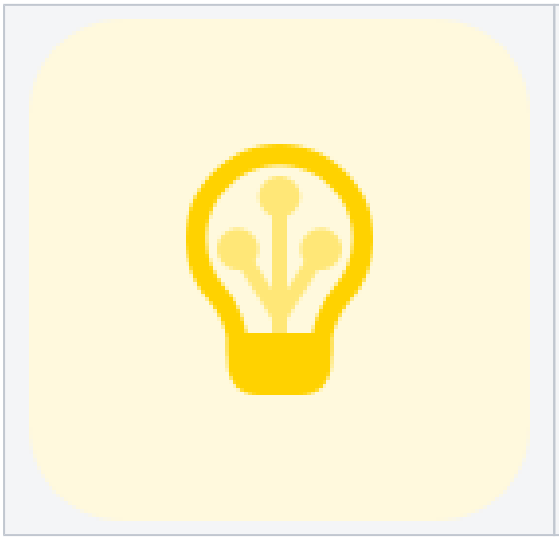
20 Registrar el servicio WsMacoManager

```
        ContainerDI.Register<IWsMacoActions>(() => new WsMacoManager(macoApiConfig), ContainerDI.
LifeStyleContainer.Transient);
```

Registro del servicio WSMacoAuthentication que implementa la interfaz IMacoAuthentication. Encargada de gestionar las consultas al servicio de Autenticación de Maco

21 Configurar parámetros en fichero de configuración

```
        ContainerDI.Register<IMacoAuthentication, WSMacoAuthentication>(ContainerDI.LifeStyleContainer.
Transient);
```



Breaking changes

En la versión 2.5.0, se ha cambiado el nombre de la clase MacoAuthentication por WSMacoAuthentication.

Registro del servicio WSMacoProfessionalManager que implementa la interfaz IMacoProfessionalManager. Encargada de gestionar las consultas al servicio Profesional de Maco.

22 Registrar el servicio WSMacoProfesionalManager

```
         ContainerDI.Register<IMacoProfessionalManager, WSMacoProfessionalManager>(ContainerDI.
LifeStyleContainer.Transient);
```

```
       Registro del servicio WSMacoOperatorManager que implementa la interfaz IMacoOperatorManager.
      Encargada de gestionar las consultas al servicio Operador de Maco
```

23 Registrar el servicio WSMacoOperatorManager

```
        ContainerDI.Register<IMacoOperatorManager, WSMacoOperatorManager>(ContainerDI.LifeStyleContainer.
Transient)
```

Configuración de la API versión Réplica de BBDD

Registro del servicio IConfigContextApiConnection, configuración del contexto de BBDD con el tipo concreto MacoContextConnection. Recibe como parámetros la cadena de conexión de la réplica, en este caso del fichero de configuración app.config, tipo de esquema Maco y nombre del esquema de la BBDD.

24 Registro del contexto de la BBDD de Maco

```
        var macoContextConnection = new MacoContextConnection( new
ManagedConfigConnectionBase<SchemaMacoType>[] { new ManagedConfigConnectionBase<SchemaMacoType>
(ConfigurationManager.ConnectionStrings["MACOConnection"].ConnectionString, SchemaMacoType.Maco,
"NOMBRE_ESQUEMA_REPLICA_MACO") });


ContainerDI.Register<IConfigContextApiConnection<SchemaMacoType>>(() =>macoContextConnection,
ContainerDI.LifeStyleContainer.Transient);
```

Registro del tipo MacoOperatorReplicaManager que implementa la interfaz IMacoOperatorManager. Servicio encargado de consultar datos de operadores vía réplica de BBDD.

25 Registro del tipo MacoOperatorReplicaManager

```
         ContainerDI.Register<IMacoOperatorManager, MacoOperatorReplicaManager>(ContainerDI.
LifeStyleContainer.Transient);
```

Registro del tipo MacoRepository que implementa la interfaz IMacoRepository. Repositorio encargado de consultar datos de Maco vía réplica de BBDD.

26 Registro del tipo MacoRepository

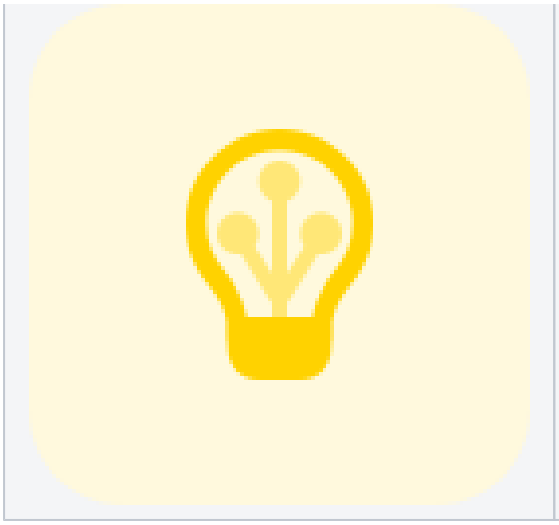
```
         ContainerDI.Register<IMacoRepository, MacoRepository>(ContainerDI.LifeStyleContainer.
Transient);
```

Registro del tipo MacoModuleManager que implementa la interfaz IMacoModuleManager . Servicio engargado de consultar datos de módulos en Maco vía réplica de BBDD.

27 Registrar el servicio MacoModuleManager

```
        ContainerDI.Register<IMacoModuleManager<object>, MacoModuleManager>(ContainerDI.
LifeStyleContainer.Transient);
}
```

	Inyeccción de dependencias



En el caso de que no se desee utilizar el inyector de dependencia ContainerDI, la versión 2.5.0 del componente SAS.MacoApiClient.ManagedDataAccess.Oracle permite la configuración de la inyección de dependencia utilizando el inyector por defecto de .NET, del siguiente modo.

```
services.AddMacoOracleServices
(macoContextConnection );
```

En el se le pasa como argumento la configuración del contexto de Maco. Este método se encuentra en el espacio de nombre SAS. MacoApiClient.ManagedDataAccess.Oracle.IoC.

A continuación se presenta el código completo del ejemplo de configuración de la versión >= 2.0.0:

28 Ejemplo de configuración versión librería MacoApiClient >= 2.0.0 utilizando el inyector de dependencia ContainerDI

```
         public class Startup
{
        public Startup(IConfiguration configuration)
        {
                Configuration = configuration;
        }

        public IConfiguration Configuration { get; }
        private static string _xmlMacoKey;
        private static TicketValidatorConfig _ticketValidatorConfig;

        // This method gets called by the runtime. Use this method to add services to the container.
        public void ConfigureServices(IServiceCollection services)
        {
                ...
                ContainerDI.SetAsyncScopedLifestyle();
                ContainerDI.RegisterServices(ref services);
        }

        // This method gets called by the runtime. Use this method to configure the HTTP request
pipeline.
        public void Configure(IApplicationBuilder app, IHostingEnvironment env)
        {
                ContainerDI.RegisterMvcControllers(ref app);
                ContainerDI.RegisterMvcViewComponents(ref app);

                app.UseDeveloperExceptionPage();

                this.InitializeContainer();
                ...
        }

        /// <summary>
        /// Inicializador de contenedor utilizando el inyector de dependencia ContainerDI.
        /// </summary>
        private void InitializeContainer()
        {
                #region registrar maco
                        _xmlMacoKey = Configuration.GetSection("MacoApiClient:MacoXmlKey").Value;

                        _ticketValidatorConfig = new TicketValidatorConfig
```

```
                        {
                                RsaKey = _xmlMacoKey,
                                ReferenceDate = DateTime.Now,
                                FutureTicketToleranceMillis = 0,
                                TicketTtlMillis = 24 * 60 * 60 * 1000,

                                TicketValidations = new HashSet<TicketValidationType>()
                        };
                        _ticketValidatorConfig.TicketValidations.Add(TicketValidationType.Signature);
//SignatureTicketValidation: Valida que la firma es correcta
                        _ticketValidatorConfig.TicketValidations.Add(TicketValidationType.Expired);
//ExpiredTicketValidation: Valida que el ticket no este caducado
                        _ticketValidatorConfig.TicketValidations.Add(TicketValidationType.Future);
//FutureTicketValidation: Valida que el ticket no sea futuro
                        _ticketValidatorConfig.TicketValidations.Add(TicketValidationType.Permission);
//PermissionTicketValidation: Valida que el permiso requerido se encuentre en el ticket

                        ContainerDI.Register<IMacoExceptionManager, MacoExceptionManager>(ContainerDI.
LifeStyleContainer.Transient);
                        ContainerDI.Register<IMacoUserTransformer, WSMacoUserTransformer>(ContainerDI.
LifeStyleContainer.Transient);


                        var macoApiConfig = new MacoApiConfig(Configuration.GetSection
("WebServiceLocator:ServiceAddressMaco").Value)
                        {
                                ModuleCode = Configuration.GetSection("MacoApiClient:MacoModule").Value
                        };
                        macoApiConfig.EndPoints = new Dictionary<MacoApiClient.WebService.Api.Entity.
EndPointType, string>
                        {
                                { MacoApiClient.WebService.Api.Entity.EndPointType.Validacion,
Configuration.GetSection("WebServiceLocator:EndPointAddressMacoValidacion").Value },
                                { MacoApiClient.WebService.Api.Entity.EndPointType.Operador,
Configuration.GetSection("WebServiceLocator:EndPointAddressMacoOperador").Value},
                                { MacoApiClient.WebService.Api.Entity.EndPointType.Profesional,
Configuration.GetSection("WebServiceLocator:EndPointAddressMacoProfesional").Value }
                        };

                        ContainerDI.Register<IMacoApiConfig>(() => macoApiConfig, ContainerDI.
LifeStyleContainer.Transient);
                        ContainerDI.Register<ITicketValidator>(() => new TicketValidator
(_ticketValidatorConfig), ContainerDI.LifeStyleContainer.Transient);
                        ContainerDI.Register<IWsMacoActions>(() => new WSMacoManager(macoApiConfig),
ContainerDI.LifeStyleContainer.Transient);
                        ContainerDI.Register<IMacoAuthentication, WSMacoAuthentication>(ContainerDI.
LifeStyleContainer.Transient);
                        ContainerDI.Register<IMacoOperatorManager, WSMacoOperatorManager>(ContainerDI.
LifeStyleContainer.Transient);


            //descomentar en caso de consultar datos de módulo y perfil vía réplica de BBDD
                         //var macoContextConnection = new MacoContextConnection( new
ManagedConfigConnectionBase<SchemaMacoType>[] {
                        //new ManagedConfigConnectionBase<SchemaMacoType>(ConfigurationManager.
ConnectionStrings["MACOConnection"].ConnectionString, SchemaMacoType.Maco,
"NOMBRE_ESQUEMA_REPLICA_MACO") });
                        //ContainerDI.Register<IConfigContextApiConnection<SchemaMacoType>>(()
=>macoContextConnection, ContainerDI.LifeStyleContainer.Transient);
                        //ContainerDI.Register<IMacoOperatorManager, MacoOperatorReplicaManager>
(ContainerDI.LifeStyleContainer.Transient);
                        //ContainerDI.Register<IMacoRepository, MacoRepository>(ContainerDI.
LifeStyleContainer.Transient);
                        //ContainerDI.Register<IMacoModuleManager<object>, MacoModuleManager>
(ContainerDI.LifeStyleContainer.Transient);

                #endregion registrar maco
        }
}
```

29 Ejemplo de configuración versión librería MacoApiClient >= 2.0.0 utilizando el inyector de dependencia por defecto de .NET

```
        using SAS.MacoApiClient.WebService.IoC;
//using SAS.MacoApiClient.ManagedDataAccess.Oracle.IoC;
public class Startup
{
        public Startup(IConfiguration configuration)
        {
                Configuration = configuration;
        }

        public IConfiguration Configuration { get; }
        private static string _xmlMacoKey;
        private static TicketValidatorConfig _ticketValidatorConfig;

        // This method gets called by the runtime. Use this method to add services to the container.
        public void ConfigureServices(IServiceCollection services)
        {

                #region Inyección de dependencia utilizando el inyector por defecto de .NET

                        #region registrar Maco
                        _xmlMacoKey = Configuration.GetSection("MacoApiClient:MacoXmlKey").Value;

                        _ticketValidatorConfig = new TicketValidatorConfig
                        {
                                RsaKey = _xmlMacoKey,
                                ReferenceDate = DateTime.Now,
                                FutureTicketToleranceMillis = 0,
                                TicketTtlMillis = 24 * 60 * 60 * 1000,

                                TicketValidations = new HashSet<TicketValidationType>()
                        };
                        _ticketValidatorConfig.TicketValidations.Add(TicketValidationType.Signature);
//SignatureTicketValidation: Valida que la firma es correcta
                        _ticketValidatorConfig.TicketValidations.Add(TicketValidationType.Expired);
//ExpiredTicketValidation: Valida que el ticket no este caducado
                        _ticketValidatorConfig.TicketValidations.Add(TicketValidationType.Future);
//FutureTicketValidation: Valida que el ticket no sea futuro
                        _ticketValidatorConfig.TicketValidations.Add(TicketValidationType.Permission);
//PermissionTicketValidation: Valida que el permiso requerido se encuentre en el ticket

                        var macoApiConfig = new MacoApiConfig(Configuration.GetSection
("WebServiceLocator:ServiceAddressMaco").Value)
                        {
                                ModuleCode = Configuration.GetSection("MacoApiClient:MacoModule").Value
                        };
                        macoApiConfig.EndPoints = new Dictionary<SAS.MacoApiClient.WebService.Api.Entity.
EndPointType, string>
                        {
                                { SAS.MacoApiClient.WebService.Api.Entity.EndPointType.Validacion,
Configuration.GetSection("WebServiceLocator:EndPointAddressMacoValidacion").Value },
                                { SAS.MacoApiClient.WebService.Api.Entity.EndPointType.Operador,
Configuration.GetSection("WebServiceLocator:EndPointAddressMacoOperador").Value},
                                { SAS.MacoApiClient.WebService.Api.Entity.EndPointType.Profesional,
Configuration.GetSection("WebServiceLocator:EndPointAddressMacoProfesional").Value }
                        };

                        //registro de servicios de la API MacoApiClient implementación webservices
utilizando el inyector de dependencia por defecto de .NET
                        services.AddMacoWSServices(macoApiConfig, _ticketValidatorConfig);

                        #endregion
                        #region registrar servicios Maco que acceden a datos vía réplica BBDD
                                //descomentar en caso de consultar datos de módulo y perfil vía réplica
de BBDD
                                //var macoContextConnection = new MacoContextConnection( new
```

```
ManagedConfigConnectionBase<SchemaMacoType>[] {
                                        //new ManagedConfigConnectionBase<SchemaMacoType>
(ConfigurationManager.ConnectionStrings["MACOConnection"].ConnectionString, SchemaMacoType.Maco,
"NOMBRE_ESQUEMA_REPLICA_MACO") });
                           //services.AddMacoOracleServices(macoContextConnection);
                        #endregion

                #endregion Inyección de dependencia utilizando el inyector por defecto de .NET
        }
}
```

Uso

A continuación se describe el uso de la librería.

Funcionalidades utilizando Servicio Web de Maco *Autenticación*

Para poder realizar la autenticación y autorización con Maco se deberá de hacer uso de la clase MacoAuthentication, mediante el uso del método LoginAuthentication() que devuelve un objeto MacoUser, y con el que se podrá realizar la autenticación con Maco. Si la autenticación no se ha podido realizar con éxito se lanzará una excepción de tipo MacoException. .

Ejemplo:

30 Ejemplo de autenticación

```
        public MacoUser login(string username, string password) {
        return _macoAuthentication.LoginAuthentication(username, password);
}
```


Autenticación de usuarios

El parámetro password del operador no se debe pasar encriptado. La librería MacoApiClient lo transforma internamente en MD5, antes de pasarlo como parámetro en la llamada al servicio web

Para realizar la comprobación de autorización de un módulo concreto con Maco se deberá de hacer uso del método ServiceAuthorization(). Si la autorización no se ha podido realizar con éxito se lanzará una excepción de tipo MacoException.

Ejemplo:

31 Ejemplo autorización de un módulo concreto

```
        public boolean autorizarServicio(MacoUser user, string requestedService) {
        return _macoAuthentication.ServiceAuthorization(user, requestedService);
}
```

Para realizar la comprobación de autenticación y autorización de un módulo concreto con Maco se deberá de hacer uso del método LoginAuthentication, pasando como argumento username, password y código del módulo en Maco. Si la autorización no se ha podido realizar con éxito se lanzará una excepción de tipo MacoException.

Ejemplo:

32 Ejemplo de autenticación pasando username/password y módulo

```
        public boolean LoginAuthentication(string username, string password,string moduleCode) {
        return _macoAuthentication.LoginAuthentication(username, password,moduleCode);
}
```

Operadores

Para poder buscar operadores en Maco se deberá de hacer uso de la función FindOperator, para un solo resultado, o FindOperators, para múltiples resultados. Los campos que pueden ser usados en el dto para la búsqueda son:

UsersPerPage: Número de usuarios por página PageNumber: Numero de la página a recibir Name: Nombre a buscar (puede ser parcial) FirstSurname: Apellido 1 (puede ser parcial) SecondSurname: Apellido 2 (Puede ser parcial) Login: login del usuario DirayaCode: código de diraya ModuleCode: código del módulo UnitCode: código de la unidad organizativa UnitTypeCode: tipo de unidad organizativa

Ejemplo:

33 Búsqueda de usuarios

```
        public MacoUser BuscarUsuarioPorLogin(string login) {
        QuerysUserSearchDto querysUserSearchDto= new QuerysUserSearchDto();
        querysUserSearchDto.PageNumber= 1;
        querysUserSearchDto.UsersPerPage = 1;
        querysUserSearchDto.Login = login;
        return _macoOperatorManager.FindOperator(usuarioMacoQueRealizaLaPeticion, querysUserSearchDto);
}

//Permite la búsqueda por nombre, devuelve una lista de usuarios
public PaginationList<MacoUser> BuscarUsuariosPorNombre(string name) {
        QuerysUserSearchDto querysUserSearchDto= new QuerysUserSearchDto();
        querysUserSearchDto.PageNumber = 1;
        querysUserSearchDto.UsersPerPage = 10;
        querysUserSearchDto.Name = name;
        return _macoOperatorManager.FindOperators(usuarioMacoQueRealizaLaPeticion, querysUserSearchDto);
}
```

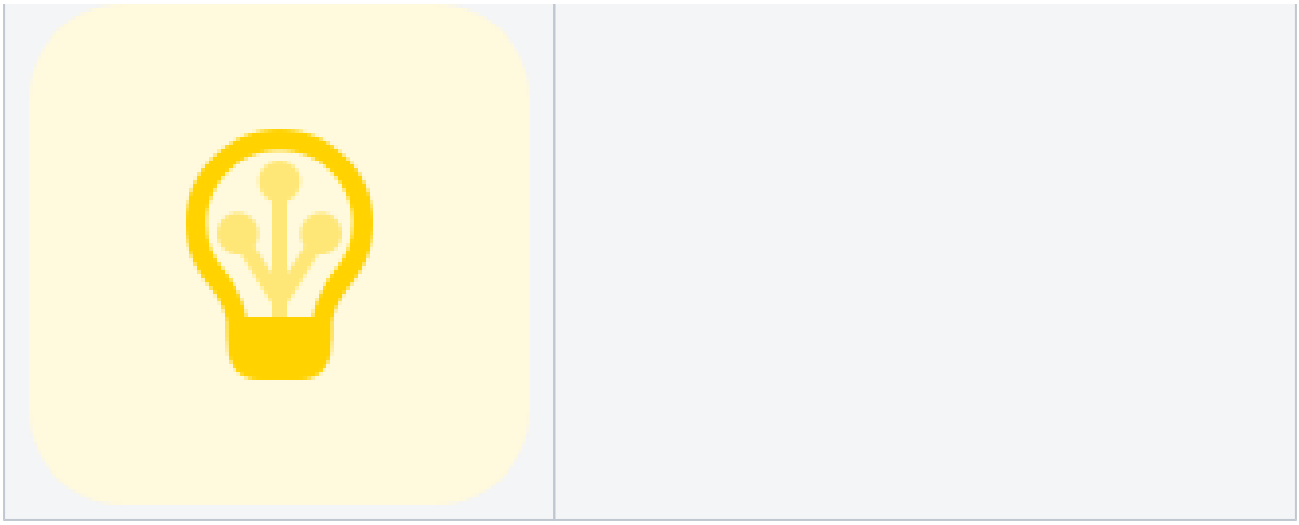
Para realizar un cambio de contraseña de un usuario en Maco, se deberá de hacer uso de la función ChangeUserPassword:

34 Cambio de contraseña de operador

```
        //realiza un cambio de contraseña
public void CambiarContraseña(MacoUser user, string password) {
        //la manera de generar una contraseña que acepte MACO puede variar y debe de ser consultada en
la documentación de MACO.
        var passwordRSA = MacoUtil.RSA(password,rsaKey);
        // rsaKey es la clave pública de Maco: <RSAKeyValue><Modulus>...</RSAKeyValue> proporcionada por
la Oficina Técnica de Interoperabilidad
        _macoOperatorManager.ChangeUserPassword(user, passwordRSA);
}
```

siendo _macoOperatorManager un objeto de tipo IMacoOperatorManager.

	Breaking Change ***En la versión 2.5.0, el método RSA(password, rsaKey) se encuentra en la clase TicketUtil del componente SAS. MacoApiClient.Ticket, dentro del espacio de nombre SAS. MacoApiClient.Ticket.Utils.***



Profesionales

Para poder buscar profesionales en Maco se deberá de hacer uso de la función FindProfessional, para un solo resultado, o FindProfessionals ,para múltiples resultados, de la misma manera que se realiza con los Operadores. Los campos que, adicionalmente a los descritos en Operador, podrán ser usados son los siguientes:

- ProfessionalCode: Código de profesional.
- LicenseNumber: Código de Colegiado.
- ProfessionalType : Tipo de profesional.
- ProfessionalSpecialtyCode : Código de la especialidad del profesional.
- StatusCode: Estado del usuario, cuyos posibles valores son:StatusAny, StatusActive, StatusPassive, StatusLocked, StatusChangePassword.

Validación de tickets Maco

- Si se quiere validar un ticket + permisos concretos, se realizará pasándole la lista de permisos a validar en la llamada IsTicketValid. Por ejemplo:

35 Validación de ticket y permisos concretos

```
       List<string> permissions = new List<string>() { "CUSTOMPERMISSION", "CUSTOMPERMISSION2" };
HashSet<string> hSetPermissions = new HashSet<string>(permissions);
return new _ticketValidator.IsTicketValid(ticket,null, hSetPermissions)
```

- Si solo se quiere validar los valores por defecto, registrados por inyección de dependencia en la configuración TicketValidatorConfig, no será necesario pasarle la lista de permisos:

36 Validación de ticket y firma con los valores por defecto de TicketValidatorConfig

```
        private bool Validate(string ticket, string signature) {
        return new _ticketValidator.IsTicketValid(ticket, signature);
}
```

Siendo _ticketValidator un objeto de tipo ITicketValidator.

- En el siguiente ejemplo se muestra la validación de ticket, firma, operación y fecha de un mensaje:

37 Validación de ticket , firma, permiso y fecha de ticket

```
        private bool ValidarTicket(string sTicketEntrada, string sFirmaEntrada, string
sOperacionEntrada, DateTime dtFechaMensaje)
        {
            var _xmlMacoKey = "CLAVE_PUBLICA_MACO";
            var result;
            var _ticketValidatorConfig = new TicketValidatorConfig
            {
                RsaKey = _xmlMacoKey,
                ReferenceDate = dtFechaMensaje,
                FutureTicketToleranceMillis = 0,
                TicketTtlMillis = 24 * 60 * 60 * 1000,

                TicketValidations = new HashSet<TicketValidationType>()
            };
            _ticketValidatorConfig.TicketValidations.Add(TicketValidationType.Signature);
//SignatureTicketValidation: Valida que la firma es correcta
            _ticketValidatorConfig.TicketValidations.Add(TicketValidationType.Expired);
//ExpiredTicketValidation: Valida que el ticket no este caducado
            _ticketValidatorConfig.TicketValidations.Add(TicketValidationType.Future);
//FutureTicketValidation: Valida que el ticket no sea futuro
            _ticketValidatorConfig.TicketValidations.Add(TicketValidationType.Permission); //Valida que
los permisos son los indicados en el ticket

            List<string> permissions = new List<string>() { sOperacionEntrada }; //si sOperacionEntrada
está formado por diferentes permisos separados por un caracter separador, realizar una operación Split
por dicho caracter separador
            HashSet<string> hSetPermissions = new HashSet<string>(permissions);

            var ticketValidator = new TicketValidator(_ticketValidatorConfig);
            try
            {
                result = ticketValidator.IsTicketValid(sTicketEntrada, sFirmaEntrada, hSetPermissions);
            }
            catch (MacoException me)
            {
                result = false;
            }

            return result;
        }
```

En el ejemplo anterior se crea un objeto de tipo TicketValidator con el operador "new". En el caso de haber usado inyección por dependencia para la configuración de los servicios, sólo habría que obtener una instancia de ITicketValidator y asignarle en el atributo Config, ticketValidator.Config, la configuración _ticketValidatorConfig adecuada.

En el siguiente ejemplo se muestra métdoos de autenticación de un usuario y validación de un ticket utilizando interfaces:

38 Validación de ticket , firma, y autenticación de usuario utilizando las interfaces de los servicios

```
        private IMacoAuthentication _macoAuthentication;
private ITicketValidator _ticketValidator;


public UserService( IMacoAuthentication macoAuthentication, ITicketValidator ticketValidator)
{
        _macoAuthentication = macoAuthentication;
        _ticketValidator = ticketValidator;
}

public MacoUser LoginAuthentication(string loginName, string password, string moduleCode)
{
        try
        {
                return _macoAuthentication.LoginAuthentication(loginName, password, moduleCode);
        }
        catch (Exception ex)
        {
                throw ex;
        }
}

public bool IsTicketValid(string ticket, string ticketSignature)
{
        return _ticketValidator.IsTicketValid( ticket, ticketSignature);
}
```