Componente Java para 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

- Dirección General de Tecnologías de la Información y Comunicaciones
  - Áreas de Gobierno Tecnológico de SSII y del Dato
- 1. Introducción
- 2. Dependencias
- 3. Errores Conocidos
- 4. Historial de Cambios
- 5. Integración de la librería
  - Configuracion
  - Configuración detallada de los endpoints de consulta
  - HTTPS
- 6. Funcionalidades
  - Autenticación
  - Operadores
  - Profesionales
  - Validacion de Tickets de Maco
    - Factoría

Resumen

- **Versión:** v01r21
- **Fecha publicación:** 17 sept 2020
- **Entrada en vigor desde:** 26 feb 2018

blocked URL

Cumplimiento normativo

Las normas expuestas son de obligado cumplimiento. La STIC podrá estudiar los casos excepcionales los cuales serán gestionados a través de los responsables del proyecto correspondiente y autorizados por el Área de Gobernanza de la STIC. Asimismo cualquier aspecto no recogido en estas normas deberá regirse en primera instancia por las guías técnicas correspondientes al esquema nacional de seguridad y esquema nacional de interoperabilidad según correspondencia y en su defecto a los marcos normativos y de desarrollo software establecidos por la Junta de Andalucía, debiendo ser puesto de manifiesto ante la STIC.

La STIC se reserva el derecho a la modificación de la norma sin previo aviso, tras lo cual, notificará del cambio a los actores implicados para su adopción inmediata según la planificación de cada proyecto.

En el caso de que algún actor considere conveniente y/o necesario el incumplimiento de alguna de las normas y /o recomendaciones, deberá aportar previamente la correspondiente justificación fehaciente documentada de la solución alternativa propuesta, así como toda aquella documentación que le sea requerida por la STIC para proceder a su validación técnica.

Contacto Arquitectura: l-arquitectura.stic@juntadeandalucia.es

Histórico de cambios

Los cambios en la normativa vendrán acompañados de un registro de las modificaciones. De este modo se podrá realizar un seguimiento y consultar su evolución. Ordenándose de mas recientes a menos recientes, prestando especial cuidado a las cabezeras de la tablas dónde se indican las fechas de entrada en vigor y versión.

Versión	v01r20	**Fecha publicación**	16 mar 2021	**Fecha entrada en vigor**	26 feb 2018
Alcance					
• Actualización de urls de acceso al servicio y el repositorio de artefactos dónde se aloja. • Añadida sección de "Configuración detallada de los endpoints de consulta".					

Versión	v01r20	**Fecha publicación**	17 sept 2020	**Fecha entrada en vigor**	26 feb 2018
Alcance					
• Actualizaciones de la librería.					

1. Introducción

Esta librería permitirá de hacer uso de la Api proporcionada por Maco y tener acceso a utilidades relacionadas con la operativa con Maco.

Puede encontrarse un ejemplo de integración con maco mediante esta librería en el Arquetipo "javaee7-sample", en el branch "MacoIntegration".

Actualmente la librería tiene las siguiente funcionalidades:

- Autentificación
    - Permite realizar la autentificación con Maco.
    - Permite comprobar la autorización a un servicio.
- Operadores
    - Permite realizar búsquedas de Operadores
    - Permite cambiar la contraseña de un Operador
- Profesionales
    - Permite realizar búsquedas de Profesionales
- Ticket
    - Permite realizar validaciones de tickets

De cara a priorizar futuras funcionalidades, si algún proveedor tiene la necesidad de alguna funcionalidad adicional se deberá de comunicar a l-arquitectura.stic.sspa@juntadeandalucia.es

2. Dependencias

MacoApiClient	sasutils-test	sasutils-xml	sasutils-parse	sasutils-file
2.1.0	2.0.2	2.0.2	2.0.2	2.0.2
2.0.1	2.0.2	2.0.2	2.0.2	2.0.2
2.0.0	2.0.2	2.0.2	2.0.2	2.0.2
1.8.0.2	1.0.0.1	1.0.0.1	1.0.0.1	1.0.0.1
1.8.0.1	1.0.0.1	1.0.0.1	1.0.0.1	1.0.0.1
1.7.2.2	1.0.0.1	1.0.0.1	1.0.0.1	1.0.0.1
1.7.2.1	1.0.0.1	1.0.0.1	1.0.0.1	1.0.0.1
1.7.1.1	1.0.0.1	1.0.0.1	1.0.0.1	1.0.0.1
1.7.0.1	1.0.0.1	1.0.0.1	1.0.0.1	1.0.0.1
1.6.0.1	1.0.0.1	1.0.0.1	1.0.0.1	1.0.0.1
1.5.0.1	1.0.0.1	1.0.0.1	1.0.0.1	1.0.0.1
1.4.0.1	1.0.0.1	1.0.0.1	1.0.0.1	1.0.0.1
1.3.1.1	1.0.0.1	1.0.0.1	1.0.0.1	1.0.0.1
1.3.0.1	1.0.0.1	1.0.0.1	1.0.0.1	1.0.0.1
1.2.1.1	-	-	-	-
1.2.0.1	-	-	-	-
1.1.0.1	-	-	-	-
1.0.0.1	-	-	-	-

3. Errores Conocidos

- Bug al configurar por sasconfiguration los permisos del ticket.
    - Problema al transformar de string a enumerado
    - *Corregido desde de la version 1.8.0.2*
- Bug al validar tickets expirados, da como valido tickets caducados.
    - La comprobación de la fecha del ticket esta invertida por lo que realiza una comprobación similar a tickets a futuro
    - *Corregido desde de la versión 1.7.1.1*
- Error al realizar consulta de operadores: Unable to create JAXBContext
    - Es causado por una etiqueta introducida por el wsimport al crear los servicios para la librería, y solo sucede cuando se hacen los test unitarios, no fallando en el desplegable final en Weblogic.
    - *Corregido desde de la versión 1.2.1.1*

4. Historial de Cambios

v2.1.0

- `Forzado de endpoints HTTPS.`

v2.0.1

- `Actualización del método serviceAuthorization (Eliminar comprobación de los servicios asociados) en la clase BaseMacoAuthentication.`

v2.0.0

- `Actualización de dependencias.`

v1.8.0.2

- Corrección de la transformación de los permisos configurados.

v1.8.0.1

- Integracion con sasconfiguration, para poder configurar automaticamente la libreria

v1.7.2.2

- Se prioriza Ticket sobre Password en la mensajeria soap
- Se usara un Modulo por defecto en servicios de MACO si no se especifica uno.

v1.7.1.1

- Corregido bug en el TicketValidator, concretamente en la validación de ticket expirados

v1.7.0.1

- Modificacion para permitir llamadas a algunos servicios solo con ticket y firma

v1.6.0.1

- Extendido método en la api de validación de tickets, parámetro adicional a la llamada isTicketValid, para poder indicarle Permisos adicionales en esa validación.

v1.5.0.1

- Extendido método en la api de Autentificacion, ahora permite opcionalmente especificar directamente el código de modulo, para que use temporalmente ese modulo en vez del configurado en MacoApiConfig
- A partir de esta version el ModuleCode del usuario que se use para atentificar, se mantendra estatico. Anteriormente al ralizar el login, se establecia el ModuleCode especificado en el MacoApiConfig.

v1.4.0.1

- Version con compatibilidad directa con CDI

v1.3.1.1

- Perfectivo en el webService de PreValidar, reduciendo el tamaño del xml generado y enviado

v1.3.0.1

- Refactorización del código
    - Division en modulos
    - Nuevos test unitarios
    - Factoria nueva para crear todos los recursos encesarios

v1.2.1.1

- Corregido bug en la busqueda de operadores que impedia hacer uso de la busqueda en test unitarios

v1.2.0.1

- Añadido `MacoProfessionalManager` para la gestión de operaciones referentes a los profesionales
- Refactorizado el nombre de los métodos de `MacoOperatorManager`. Todas las referencias de `User' han sido sustituidas por `Operator'.
- Refactorizado los parámetros de entrada para los métodos `findOperator/s)' de la clase `MacoOperatorManager`, ahora aceptan un objeto dto con los parámetros que configuran la búsqueda.
- Modificada la forma en que el validador de expiración de ticket funcionaba (`ExpiredTicketValidation`), para que por defecto use la fecha actual como referencia a la hora de validar los tickets, si no se le especifica ninguna en la configuración.

5. Integración de la librería

Para incluir la librería en un proyecto Maven es necesario realizar los siguientes pasos:

1. Agregar el repositorio de la Junta de Andalucía. Para ello, seguir las indicaciones de la página Repositorio de artefactos.
2. Especificar la dependencia en el POM del proyecto que requiere la funcionalidad.

1.
```
          Con CDI:(A partir de la version 1.4.0.1
<dependency>
     <groupId>es.ja.csalud.sas.componentescomunes.macoapiclient</groupId>
     <artifactId>macoApiClient-jee</artifactId>
     <version>x.x.x.x</version>
</dependency>
```

```
          Sin CDI:
<dependency>
    <groupId>es.ja.csalud.sas.componentescomunes.macoapiclient</groupId>
    <artifactId>macoApiClient</artifactId>
    <version>x.x.x.x</version>
</dependency>
```

Puede ser usada mediante CDI o de forma tradicional creando los objetos requeridos por cada constructor. Con CDI es indispensable especificar como se producirán las inyecciones de los objetos de configuración.

Configuracion

Para que la libreria se autoconfigure mediante la libreria sasconfiguration, se tendran que realizar las siguientes acciones:

- Añadir la siguiente dependencia:

```
          <dependency>
    <groupId>es.ja.csalud.sas.componentescomunes.macoapiclient</groupId>
    <artifactId>macoApiClient-config</artifactId>
    <version>x.x.x.x</version>
</dependency>
```

- Agregar las dependencias necesarias de la librería sasconfiguration
- definir los siguientes parámetros:

es.ja.csalud.sas.componentescomunes.macoapiclient.config.url	Url de maco

es.ja.csalud.sas.componentescomunes.macoapiclient.config.modulecode	Codigo del modulo del aplicativo
es.ja.csalud.sas.componentescomunes.macoapiclient.config.macocharset	Codificacion de maco (opcional)
es.ja.csalud.sas.componentescomunes.macoapiclient.config.ticket.ttl	Tiempo de vida del ticket (en milisegundos)
es.ja.csalud.sas.componentescomunes.macoapiclient.config.ticket.futuretolerance	Tolerancia a fecha futura del ticket (en milisegundos)
es.ja.csalud.sas.componentescomunes.macoapiclient.config.ticket.permissions	Permisos que se validaran en todos los tickets
es.ja.csalud.sas.componentescomunes.macoapiclient.config.ticket.rsakey	Clave publica de Maco en formato XML
es.ja.csalud.sas.componentescomunes.macoapiclient.config.ticket.ticketValidations	Tipos de validaciones que se realizaran (FUTURE,EXPIRED,PERMISSION, SIGNATURE)

Estos parámetros se definirán usando la librería sasconfiguration.(https://ws001.sspa.juntadeandalucia.es/unifica/web/gobernanza/sasconfiguration)

De forma alternativa se podrán definir manualmente estas configuraciones.

Actualmente existen dos objetos de configuración:

- **`MacoApiConfig`** gestiona la configuración general de la librería. Ejemplo:

```
       // Esta implementación presupone que el endpoint a consultar se encuentran en el path por
defecto "/wsCentralizada" relativo a la url que se especifica en el constructor de MacoApiConfig.
// Dicha información puede no ser precisa en el momento de la configuración de su solución por lo que le
recomendamos que consulte con la Oficina Técnica de Interoperabilidad la configuración adecuada.
// Consulte la sección "Configuración detallada de los endpoints de consulta" en este documento para
customizar la configuración.
...
@Produces
   public MacoApiConfig getMacoApiConfig() {
      final MacoApiConfig macoApiConfig = new BaseMacoApiConfig("https://servicios.pre.sas.junta-
andalucia.es/");
      macoApiConfig.setModuleCode("CódigoMóduloEnMacoDelSistema");
      return macoApiConfig;
   }
...
```

- **`TicketValidatorConfig`** gestiona la configuración para la validación de tickets. Ejemplo:

```
        ...
    @Produces
    public TicketValidatorConfig getTicketValidatorConfig() {
        TicketValidatorConfig config = new TicketValidatorConfig();
        config.setRSAkey(xmlKey);
        config.setReferenceDate(new Date());
        config.setFutureTicketToleranceMillis(-1);
        config.setTicketTtlMillis(24 * 60 * 60 * 1000);
        config.setTicketValidations(EnumSet.of(TicketValidationType.SIGNATURE, TicketValidationType.
EXPIRED));
        return config;
    }
...
```

Estos objetos serán usados por defecto en todas las inyecciones de la librería.

Configuración detallada de los endpoints de consulta

La estrategia de servicios, puede presentar modificaciones en la configuración de los endpoints que su solución puede consumir en cada momento. MacoApiClient, provee una configuración rápida por defecto que puede no estar actualizada respecto al direccionamiento en cada momento.

Para articular el direccionamiento de dichos servicios correctamente puede configurar MacoApiClient aprovisionando un EnumMap con el direccionamiento detallado hacia en endpoint.

Snippet https

```
       @Produces public MacoApiConfig getMacoApiConfig() {
// URL BASE DEL ENDPOINT DE MACO
 String urlMaco="https://servicios.pre.sas.junta-andalucia.es/"

// SUBPATH DEL CONTEXTO DE MACO
 String pathUrl="/maco";

// DEFINICION DE LOS ENDPOINTS ESPECIFICOS PARA LOS SERVICIOS DE VALIDACIÓN
 String macoValidationEndpoint="/wsValidacion.asmx?wsdl";

// DEFINICIONES DE LOS ENDPOINTS ESPECIFICOS PARA LOS SERVICIOS DE OPERADOR
 String macoOperadoresEndpoint="/wsOperador.asmx?wsdl";

// DEFINICIONES DE LOS ENDPOINTS ESPECIFICOS PARA LOS SERVICIOS DE PROFESIONAL
 String macoProfesionalesEndpoint="/wsProfesional.asmx?wsdl";

// SOBRESCRITURA DEL ENUMAP
EnumMap<MacoEndPointType,String> oVal=new EnumMap<>(MacoEndPointType.class);

oVal.put(MacoEndPointType.VALIDACION, pathUrl+macoValidationEndpoint);

oVal.put(MacoEndPointType.OPERADOR, pathUrl+macoOperadoresEndpoint);

oVal.put(MacoEndPointType.PROFESIONAL, pathUrl+macoProfesionalesEndpoint);
......

// Instanciado de la configuración de Maco con el constructor adicional que permite el mapping
customizado

final MacoApiConfig macoApiConfig = new BaseMacoApiConfig(urlMaco,oVal);

macoApiConfig.setModuleCode("338");

return macoApiConfig;

}
```

HTTPS

Para reapuntar los servicios por completo a las urls HTTPS de Maco se debe montar un objeto EnumMap<MacoEndPointtype,String> que representaría el path a los servicios, donde MacoEndPointType es un enumerado público para cada dominio de operación del servicio web (validación,operadores,profesionales), y el string identifica el path adicional hasta el endpoint que se concatenará a la url definida en el constructor.

A modo de ejemplo, la conexión a los servicios de Maco en PRE quedaría así.

Snippet https

```
        private String urlMaco="https://servicios.pre.sas.junta-andalucia.es/"
private String pathUrl="/maco";

private String macoValidationEndpoint="/wsValidacion.asmx?wsdl";

private String macoOperadoresEndpoint="/wsOperador.asmx?wsdl";

private String macoProfesionalesEndpoint="/wsProfesional.asmx?wsdl";

EnumMap<MacoEndPointType,String> oVal=new EnumMap<>(MacoEndPointType.class);

oVal.put(MacoEndPointType.VALIDACION, pathUrl+macoValidationEndpoint);

oVal.put(MacoEndPointType.OPERADOR, pathUrl+macoOperadoresEndpoint);

oVal.put(MacoEndPointType.PROFESIONAL, pathUrl+macoProfesionalesEndpoint);
......
#Instanciado de la configuración de Maco con el constructor adicional que permite el mapping customizado

MacoApiConfig macoApiConfig = new BaseMacoApiConfig(urlMaco,oVal);
```

Los certificados https serán gestionados por la JVM, puede consultar más información al respecto en " Import the Certificate as a Trusted Certificate "

Básicamente es ejecutar:
`keytool -import -alias example -keystore "C:\Program Files (x86)\Java\jre1.6.0_22\lib\security\cacerts" -file example.cer`
sustituyendo el alias, ruta del cacerts de java y ruta del fichero del certificado .cer por tus datos.

6. Funcionalidades

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`, se podrá realizar la autenticación con Maco.

Si la autenticación no se ha podido realizar con existo se lanzara una excepción de tipo `MacoException`.

Ejemplo:

```
        ...
    @Inject
    MacoAuthentication macoAuthentication;

    public MacoUser login(String username, String password) {
        return macoAuthentication.loginAuthentication(username, password);
    }
...
```

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 existo se lanzara una excepción de tipo `MacoException`.

Ejemplo:

```
        ...
    @Inject
    MacoAuthentication macoAuthentication;

    public boolean autorizarServicio(MacoUser user, String requestedService) {
        return macoAuthentication.serviceAuthorization(user, requestedService);
    }
...
```

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:

```
        ...
    @Inject
    MacoOperatorManager macoOperatorManager;

    //busca un usuario por su login
    public MacoUser BuscarUsuarioPorLogin(String login) {
       QuerysUserSearchDto querysUserSearchDto= new QuerysUserSearchDto();
       querysUserSearchDto.setPageNumber(1);
       querysUserSearchDto.setUsersPerPage(1);
       querysUserSearchDto.setLogin(login);
       return macoOperatorManager.findOperator(usuarioQueRealizaLaPeticion, querysUserSearchDto);
    }
    //Permite la búsqueda por nombre, devuelve una lista de usuarios
    public List<MacoUser> BuscarUsuariosPorNombre(String name) {
       QuerysUserSearchDto querysUserSearchDto= new QuerysUserSearchDto();
        querysUserSearchDto.setPageNumber(1);
        querysUserSearchDto.setUsersPerPage(10);
        querysUserSearchDto.setName(name);
        return macoOperatorManager.findOperators(usuarioQueRealizaLaPeticion, querysUserSearchDto); //
se recomienda el uso de la version paginada de este método.
    }
...
```

Para realizar un cambio de contraseña de un usuario en maco se deberá de hacer uso de la función `changeUserPassword`

```
        ...
    @Inject
    MacoOperatorManager macoOperatorManager;

    //realiza un cambio de contraseña
    public MacoUser CambiarContraseña(MacoUser user, String password) {
        String hashpass= MacoUtil.MD5(password); //la manera de generar una contraseña que acepte MACO
puede variar y debe de ser consultada en la documentación de MACO.
        return macoOperatorManager.changeUserPassword(user, hashpass);
    }
...
```

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 que 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, solo acepta 0 o 1 (STATUS_ACTIVE(0L), STATUS_PASSIVE(1L))

Validacion de Tickets de Maco

La configuración especifica de los parametros la define la Oficina de Interoperabilidad por lo que sera necesario contactar con ellos para poder usar la validacion.

Si no se especificó en la configuracion una fecha de referencia se usara por defecto la fecha del sistema.

- **ExpiredTicketValidation**: Valida que el ticket no este caducado
    - Se configurara este parametro: ticketValidatorConfig.getTicketTtlMillis()
- **FutureTicketValidation**: Valida que el ticket no sea futuro
    - Se configurara este parametro: ticketValidatorConfig.getFutureTicketToleranceMillis()
- **PermissionTicketValidation**: Valida que el permiso requerido se encuentre en el ticket
    - Se configurara este parametro:ticketValidatorConfig.getPermissions()
- **SignatureTicketValidation**: Valida que la firma es correcta
    - Se configurara este parametro: ticketValidatorConfig.getRSAkey()

Para poder configurar unos parámetros de forma global para estas validaciones se tendrá que configurar un producer de `TicketValidatorConfig` para poder especificar como se realizara la validación. Todo lo que se configure en el producer serán validaciones globales y se aplicaran a todas las llamadas de isTicketValid, de esta manera se podrá inyectar la configuración del validador en cualquier servicio.

IMPORTANTE: La clase TicketValidator **no es ThreadSafe** por lo que debe ser instanciada en cada uso, con la configuración que se requiera.

```
new TicketValidator(ticketValidatorConfig)
```

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

```
	List<String> permissions = Arrays.asList("CUSTOMPERMISSION", "CUSTOMPERMISSION2");
	return new TicketValidator(ticketValidatorConfig).isTicketValid(ticket, permissions)
```

Si solo se quiere validar los valores por defecto(configurados en el Producer) no sera necesario pasarle la lista de permisos:

```
	 @Inject
TicketValidatorConfig ticketValidatorConfig ;
	private boolean validate(String ticket, String signature) throws InvalidTicket {
		return new TicketValidator(ticketValidatorConfig).isTicketValid(ticket, signature);
	}
```

Factoría

Si no se esta usando CDI, se puede usar la factoría proporcionada:

```
Ejemplo: import es.ja.csalud.sas.componentescomunes.macoapiclient.utils.*;
...
MacoFactory macoFactory = new MacoFactory(macoApiConfig, ticketConfig)
MacoAuthentication macoAuthentication = macoFactory.getMacoAutentication();

...
```

```
	Clases proporcionadas por la factoria:
```

- MacoAuthentication getMacoAutentication()
- MacoOperatorManager getMacoOperatorManager()
- MacoExceptionManager getMacoExceptionManager() {
- TicketValidator getTicketValidator()
- WSMacoManager getWSMacoManager()
- MacoProfessionalManager getMacoProfessionalManager()
- MacoUserManager getMacoUserManager()