Plataforma Autonómica de Interoperabilidad

Desarrollo y consumo de servicios web. Buenas prácticas

GENERALITAT VALENCIANA
Conselleria d'Hisenda i Model Econòmic

DIRECCIÓN GENERAL DE TECNOLOGÍAS
DE LA INFORMACIÓN Y LAS COMUNICACIONES

Versión 034
Marzo de 2022

Unió Europea
Fons Europeu de Desenvolupament Regional
Una manera de fer Europa

Unión Europea
Fondo Europeo de Desarrollo Regional
Una manera de hacer Europa

1 Control del documento 4
1.1 Información general 4
1.2 Histórico de revisiones 4
1.3 Estado del documento 5
1.4 Objetivos 6
1.5 Audiencia 6
1.6 Glosario 6
1.7 Referencias 6

2 ¿Por qué usar servicios web? 7
2.1 Servicios web en el ámbito de la Administración electrónica 8

3 Criterios para crear un Servicios web. 10
3.1 Granularidad de los servicios 10
3.2 Definiendo el servicio (*Contract-First*) 11
3.3 Norma Técnica 11
3.3.1 Estilo de construcción del servicio (*binding style – operation use*) 11
3.3.2 Atributo de cabecera SOAP: MustUnderstand 12
3.3.3 Namespace de Soap envelope 13
3.3.4 Definición de atributos y métodos legibles 13
3.3.5 Recomendación respecto a definición de esquemas 16
3.3.5.1 Patrón de diseño de XML Schema 16
3.3.5.2 Representaciones del null en XML Schema 16
3.3.5.3 Uso de los tags elementFormDefault y attributeFormDefault 16
3.3.5.4 Espacio de nombres 19
3.4 Estrategias de versionado de servicios y XSD 19
3.5 Estrategia de comunicación de adjuntos 20
3.6 Definición de la Taxonomía de Servicio 21
3.7 Servicios REST 22
3.7.1 Modalidades de Publicación de servicios REST 24
3.7.1.1 Servicios SOAP a REST 24
3.7.1.2 Servicios REST 24

4 Entorno tecnológico de la PAI 26
4.1 Política de seguridad 26
4.2 Proveedores de servicios 26
4.3 Trazabilidad en el consumo de servicios de la PAI 27
4.3.1 Trazabilidad en los servicios instrumentales. 27
4.3.1.1 Trazabilidad en los servicios instrumentales en el Bus de Innovación. 28
4.3.1.2 Ejemplo de petición incluyendo trazabilidad en Servicios Instrumentales 28
4.3.1.3 Ejemplo de petición incluyendo trazabilidad en Servicios Innovación 30

4.3.2 Trazabilidad en Servicios de Verificación 31
4.4 Definición de Errores de negocio y SoapFaults 32
4.4.1 Especificación Errores de negocio 32
4.4.2 Especificación Soap Fault 33
4.4.2.1 faultcode 34
4.4.2.2 faultstring 34
4.4.2.3 detail 34
4.4.2.4 Ejemplo de SoapFault 37
4.4.2.5 Gestión de errores en aplicaciones clientes 40

5 Apoyo a los desarrolladores 41
5.1 Generación de servicio web 41
5.2 Clientes de servicios 41

1 Control del documento

1.1 Información general

Título	Desarrollo y consumo de servicios web. Buenas prácticas
Creado por	DGTI
Revisado por	
Lista de distribución	
Nombre del fichero	Desarrollo_y_consumo_de_servicios_web_Buenas_prácticas_v34.docx

1.2 Histórico de revisiones

Versión	Fecha	Autor	Observaciones
001	13/02/2012	DGM	Versión Inicial
002	08/04/2014	DGTI	Inclusión particularidades PAI
003	24/04/2014	DGTI	SoapFaults y gestión de errores
004	1/05/2014	DGTI	Eliminar texto/puntos generales y ambiguos. Añadir criterios técnicos que deben implementar los desarrolladores
005	16/05/2014	DGTI	Modificación e inclusión de nuevos tags y revisión general del documento.
006	10/09/2014	DGTI	Revisión del documento. Inclusión faultcode.
007	23/09/2014	DGTI	Reorganización del contenido. Inclusión de punto para proveedores de servicios y espacios de nombres.
008	26/09/2014	DGTI	Revisión del contenido. Reorganización de los puntos. Inclusión de Ayuda a los desarrolladores, patrones XML Schema, ¿Por qué usar servicios web?
009	31/10/2014	DGTI	Revisión de contenido. Sustituidas referencias a PIE (Plataforma de Intermediación del Estado) por PID (Plataforma de Intermediación de Datos). Referencia al documento de generación de servicios web.
010	21/11/2014	DGTI	Revisión del contenido. Añadido MustUnderstand. Cambio en Granularidad.
011	28/11/2014	DGTI	Modificaciones en faultcode, soapfault, definición de atributos y métodos, granularidad de servicios, parámetros de los xsd.
012	01/12/2014	DGTI	Revisión del contenido. Cambio en ejemplo soapFault, reasignación de puntos para la norma técnica.
013	015/01/2015	DGTI	Incluido ejemplo del Id_trazabilidad en el punto 4,3,1.
014	27/01/2015	DGTI	
015	26/02/2015	DGTI	Añadida referencia al documento de consumo de servicios en la sección mustUnderstand

016	27/02/2015	DGTI	Corrección de la sección mustUnderstand
017	02/11/2015	DGTI	Mustunderstand en servicios web
018	26/05/2016	DGTI	Corrección de la sección mustUnderstand
019	11/02/2016	DGTI	Ampliación Servicios REST, Id_trazabilidad, servicios innovación Cambio de Logos
020	09/06/2016	DGTI	Revisión de Contenido
021	10/06/2016	DGTI	Modificaciones relativas a cambios de redacción y aclaración
022	22/06/2016	DGTI	Inclusión de Ejemplo y Recomendaciones en servicios REST
023	12/07/2016	DGTI	Revisión del Contenido y Corrección de errores
024	18/07/2016	DGTI	Revisión del faultcode, cabecera aplicación en REST
025	03/10/2016	DGTI	Modificación formato Timestamp Id_trazabilidad y tabla de errores soapFault. Se añaden ejemplos de soapFault con descripción de tags.
026	03/04/2017	DGTI	Corrección de errores
027	24/08/2017	DGTI	Se modifican las instrucciones para la creación del id_trazabilidad, para que se genere automáticamente
028	15/09/2017	DGTI	Se modifica código de creación del id_trazabilidad
029	09/03/2018	DGTI	Eliminadas referencias a gvNix, modificadas referencias a documentos inexistentes, actualizadas versiones de CXF
030	27/09/2019	DGTI	Se corrige la url del protocolo SCSP del Ministerio (punto 2.1).
031	26/11/2019	DGTI	Se actualizan las versiones de CXF
032	27/11/2019	DGTI	Gestión de errores de aplicaciones consumidoras en el punto 4.4.2.5
033	23/08/2021	DGTI	Eliminación referencias a e-Sirca
034	14/03/2022	DGTI	Incorporación de información relativa a los servicios REST

1.3 Estado del documento

Responsable aprobación	Fecha

1.4 Objetivos

El presente documento pretende guiar en las buenas prácticas en el desarrollo de servicios web para permitir orientar la plataforma de servicios SOA de la GVA hacia un entorno de interoperabilidad real y eficiente.

1.5 Audiencia

Nombre y Apellidos	Rol
	CONSUMIDORES Y DESARROLADORES DE SERVICIOS

Tabla 1: Audiencia

1.6 Glosario

Término	Definición

Tabla 2: Glosario

1.7 Referencias

Referencia	Título

Tabla 3: Referencias

2 ¿Por qué usar servicios web?

SOA (Service Oriented Architecture) es una arquitectura de Software orientada a servicios. Para implementar con éxito una SOA se consideran unos principios importantes sobre los que debemos trabajar. Sobre ellos hemos destacado los que nos parecen más importantes y por tanto el "core" SOA:

- **Bajo acoplamiento**: Es uno de los factores críticos para el éxito de la estrategia SOA. El objetivo del bajo acoplamiento es reducir las dependencias entre los sistemas implicados. En el caso de los Web services la comunicación entre consumidores y servicios se realizará mediante la interfaz del servicio (WSDL), logrando así la independencia entre el servicio a ejecutar y el consumidor.
- **Reusabilidad**: Un servicio debe ser diseñado y construido pensando en su reutilización dentro de la misma aplicación, empresa o dominio público para su uso masivo. En el caso de los Servicios Web, tienen que ver en este apartado la relación entre WSDL y XML Schemas, como veremos más adelante el hecho de separar los XSD del WSDL es clave para su reutilización.
- **Descubrimiento**: Todo servicio debe poder ser descubierto de alguna forma para que pueda ser utilizado, consiguiendo así evitar la creación accidental de servicios que proporcionen las mismas funcionalidades.
- **Interoperabilidad**: Permite que los consumidores y los servicios desarrollados en tecnologías y plataformas diferentes puedan información y colaborar.
- **Gobernabilidad**: En un entorno SOA, la gobernanza guía el desarrollo de servicios reutilizables, lo que establece cómo se diseñarán y se desarrollarán los servicios y cómo cambiarán con el tiempo. Establece acuerdos entre los proveedores de servicios y los consumidores de esos servicios, indicando a los consumidores lo que deben esperar y lo que los proveedores están obligados a proporcionar.

Si bien SOA no está estrictamente vinculado necesariamente con Servicios Web, estos suman cada día más importancia ya que las características que poseen los convierten en el mecanismo ideal para cubrir los principios de la orientación a servicios (SOA).

Básicamente los Servicios Web son mecanismos de comunicación que permiten interoperabilidad entre aplicaciones a través del uso de estándares abiertos. De esta manera aplicaciones

desarrolladas en diferentes lenguajes de programación y ejecutadas en diferentes plataformas pueden intercambiar datos y presentar información dinámica al usuario de forma homogénea y coherente.

2.1 Servicios web en el ámbito de la Administración electrónica

En el entorno del desarrollo de servicios web en el ámbito de Administración electrónica, hay que tener muy en cuenta la Resolución de 28 de Junio de 2012, de la Secretaría de Estado de Administraciones Públicas, por la que se aprueba la Norma Técnica de Interoperabilidad de Protocolos de intermediación de datos. En dicha resolución se especifica que de forma general, en servicios de intercambio e intermediación, se debe seguir la estructura del protocolo SCSP (Sustitución de Certificados en Soporte Papel) cuya especificación está disponible en el Portal de Administración electrónica PAE/CTT en la dirección https://administracionelectronica.gob.es/ctt/scsp .

La recomendación desde la PAI es seguir esta estructura en lo que a servicios de *verificación* e *intermediación* compete, para facilitar la homogeneidad en el consumo de servicios ya sea como cedentes o requirentes de información en el ámbito de la norma técnica de interoperabilidad. En el caso de servicios denominados como ***instrumentales,* se han de seguir en la medida de lo posible las especificaciones y normas expuestas en este documento**, sin utilizar expresamente los esquemas SCSP si no aplican y pudiendo **en caso de duda, consultar a los responsables de la PAI** en el uso de los estándares en base a los servicios a publicar.

La especificación SCSP define un conjunto de mensajes, basados en XML, definidos por sus correspondientes esquemas XSD.

El sistema permite que la petición-respuesta sea síncrona o asíncrona y los mensajes van firmados mediante ***WS-Security***.

- Mensajes generales: Son comunes a todos los servicios y están compuestos por los siguientes mensajes.
  - Petición (peticion.xsd)
  - Confirmación (confirmacionPeticion.xsd)
  - Solicitud de Respuesta (solicitudRespuesta.xsd)
  - Respuesta (respuesta.xsd)
  - Soap Fault Atributos (soapfaultatributos.xsd)

- Mensajes Específicos: Depende de cada certificado administrativo o transmisión de datos. Lo define el organismo emisor.
    - Datos Específicos

La PAI dispone de todos los esquemas definidos en el ENI.

Al tratar en la mayoría de ocasiones, de información sensible a LOPD, se deberán tener muy presentes las medidas en cuanto la confidencialidad e integridad de la información intercambiada, que será protegida conforme al grupo de «medidas de protección», capítulos «Protección de las comunicaciones» (mp.com) y «Protección de la información» (mp.info) definidas en el Real Decreto 3/2010, de 8 de enero, y con las medidas de seguridad dispuestas en la Ley Orgánica 15/1999, de 13 de diciembre, y normativa de desarrollo, asegurando que no se almacena información personal de ningún ciudadano.

En el apartado III.6 de la Resolución de 28 de Junio, por el que se aprueba la Norma Técnica de Interoperabilidad de Protocolos de intermediación de datos, se hace referencia a los requisitos de trazabilidad y auditoría de los intercambios.

3 Criterios para crear un Servicios web.

Un Web Service es una parte del software que puede comunicarse con otra aplicación a través de una red usando un juego específico de protocolos estandarizados- SOAP, UDDI, WSDL.

Entre los principales beneficios que se exponen al hablar de los servicios Web, generalmente se encuentran aquellos que tienen que ver con granularidad e interoperabilidad, es decir, con la posibilidad de desarrollar componentes de software totalmente independientes que tienen funcionalidad propia, pero que son capaces de exponer tal funcionalidad y compartirla con otros servicios y aplicaciones para lograr crear sistemas más complejos.

Por otro lado, la visión planteada por este paradigma computacional, donde “todo es un servicio”, permite manejar un esquema de integración universal en el cual se pueden aprovechar todos los beneficios de cada componente con un nuevo nivel de complejidad y dinamismo.

Los Servicios Web permiten que las organizaciones integren sus diferentes aplicaciones de una manera eficiente, sin preocuparse por cómo fueron construidas, donde residen, sobre qué sistema operativo se ejecutan o cómo acceder a ellas.

3.1 Granularidad de los servicios

La granularidad de un servicio en un contexto SOA viene determinada por la cantidad global de funcionalidad encapsulada en dicho servicio. La granularidad de los servicios es fundamental para la reusabilidad y versatilidad de los servicios que se diseñen.

Los servicios web son definidos en WSDL (WebService Description Language), que de forma resumida, aglutinan en distintos métodos u operaciones, la funcionalidad implementada en los mismos. La recomendación desde la PAI es el desarrollo de un mayor número de servicios con menor número de métodos (servicios más simples), más que la encapsulación en un solo servicio de toda la funcionalidad aplicable a un determinado modelo de negocio complejo. Esto permite la clara identificación de los servicios de negocio y la simplificación en el consumo de los mismos debido a la minimización en las interacciones del método de consumo. Además permiten un mayor detalle en el registro de estadísticas de consumo de servicios.

Para entender mejor este concepto de granularidad, imaginemos un determinado organismo que por ejemplo, desarrolla un servicio que permite verificar los datos catastrales, pero al que se le puede preguntar bien por bienes inmuebles asociados a un titular, bien por todos los datos catastrales de

los inmuebles, que figuran en la base de datos del Catastro, asociados a un titular . La recomendación sería exponer ambas funcionalidades como servicios distintos.

Como resumen de este punto, se deben diseñar servicios web con las responsabilidades bien repartidas, que sean cohesivos, extensibles, escalables y reutilizables.

3.2 Definiendo el servicio (*Contract-First*)

Existen dos formas tradicionales de acometer el desarrollo de un servicio web. En la jerga tradicionalmente se les conoce como *ContractFirst* (contrato al principio) y *ContractLast* (contrato al final). La recomendación en este caso es claramente utilizar *ContractFirst,* es decir, definir las operaciones, métodos y datos del negocio del servicio como fase inicial del análisis, para implementar posteriormente el código. Los diseños Contract-First permiten dar mayor robustez al servicio frente a variaciones. También mejora los aspectos de reusabilidad, rendimiento y versionado, haciendo que sea una de las características más habituales en el diseño de webservices.

Se debe tener en cuenta que el desarrollo de servicios con esta filosofía requiere un planteamiento y definición iniciales, tanto de los servicios, como de los datos que van a formar parte del mismo. Este esfuerzo en la definición redundará como hemos dicho en la mejora del servicio a proporcionar, una mayor claridad en el planteamiento y una independencia en cuanto a la lógica de negocio interna de la aplicación.

3.3 Norma Técnica

En este capítulo se tratan puntos relativos a la aplicación de buenas prácticas en la definición técnica de los servicios web.

La definición de los mismo se basa en un conjunto de especificaciones y buenas prácticas definidas por la industria conocido como WS-I Basic Profile, para la obtención de servicio web interoperables.

3.3.1 Estilo de construcción del servicio (*binding style – operation use*)

En el capítulo de definición de los contratos de integración de los servicios, el documento en el que se especifican las operaciones y métodos está basado en el conocido WSDL. Respecto al WSDL, la particularidad más importante a tener en cuenta hace referencia a dos parámetros que enlazan cómo un determinado protocolo de mensajería es asociado a las operaciones y métodos del servicio (*binding style*) así como se conforman formalmente las peticiones (*operation use*). Las opciones posibles para el parámetro '*binding'* son 'rpc/document' mientras que para el parámetro '*use'* son: 'encoded/literal'. De las cuatro combinaciones posibles de estos parámetros, la opción adecuada para publicar un servicio en el bus de la PAI exige que esta definición sea:

- binding=***document*** use=***literal***

La razón fundamental para el uso de esta opción es la ventaja que supone para los desarrolladores frente a cambios del interfaz del servicio. La simple adición de un parámetro a la operación de un servicio suele redundar en cambios importantes en el código desarrollado en base a un interfaz *rpc/encoded*, mientras que dichos cambios pueden ser menores en la evolución de un servicio codificado en base a un interfaz *document/literal*. Esta razón, entre otras, ha concluido finalmente que el style/operation *document/literal* se haya convertido en el enfoque de diseño recomendado en el diseño de webservices.

Mostramos a continuación un detalle que muestra sobre un determinado wsdl los parámetros sobre los que hacemos referencia anteriormente.

```
<wsdl:binding name="AutenticacionArangiPortTypeSoap11" type="tns:AutenticacionArangiPortTy
  <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http" xmlns:so
  <wsdl:operation name="autenticaUsuarioWS" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
    <soap:operation soapAction="" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"/>
    <wsdl:input name="autenticaUsuarioWSRequest" xmlns:wsdl="http://schemas.xmlsoap.org/ws
      <soap:body use="literal" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"/>
    </wsdl:input>
```

Finalmente, el WSDL debe ser compatible con el WS-I Basic Profile.

3.3.2 Atributo de cabecera SOAP: MustUnderstand

Las cabeceras en el ámbito SOAP son utilizadas para extender capacidades como puedan ser seguridad, enrutamiento, autenticación, transaccionabilidad, etc.

Los contratos de integración en el mundo SOAP (los wsdl) nos dicen claramente cómo debe ser un el cuerpo (*body*), qué operaciones debe tener un servicio, como se deben conformar las peticiones, etc. Pero nunca dicen nada acerca de cómo deben ser los headers, la propia especificación SOAP tampoco lo define, sino que las deja a expensas de las definiciones del usuario.

Cuando alguien manda una petición con el parámetro de cabecera *'mustUnderstand=1'* lo que está queriendo decir es que el nodo SOAP al que está mandando la petición debe obligatoriamente entender la cabecera marcada por ese parámetro. Si alguno no lo entiende, no puede ignorarlo, y el servicio web debe devolver un SoapFault informando de ello.

Pongamos un ejemplo para entender su utilidad; Imaginemos que una determinada petición exigiera que fuera transaccional, y que sólo se hiciera el commit si todas y cada una de las 'solicitudes' que viajan en la petición fueran exitosas. Si un servidor omitiera (no hiciera caso) el hecho de que el

cliente le está informando que debe entender que esa petición exige transaccionalidad, podría darse el caso que la aplicación se quedara en un estado inconsistente. Por eso en la gran mayoría de las ocasiones se omite informar este parámetro, ya que exige que el nodo al que le están mandando la petición entienda y procese toda la cabecera adecuadamente.

En el escenario de la PAI, el *mustUnderstand* no tiene sentido ya que los consumidores de los servicios web no deben exigir el procesado de las cabeceras a los servicios finales.

En el documento “Manual de usuario para el consumo de servicios instrumentales y de verificación” publicado en el portal de la PAI junto al presente documento, aparece detallado como crear un cliente de ejemplo mediante CXF con el parámetro mustUnderstand desactivado.

Aun así, la aplicación más habitual de este parámetro se realiza la cabecera “Security” de WSS, obligando de esta forma a que todos los sistemas por lo que pasa la petición deban entender este protocolo de seguridad SOAP. Por ello, ya que la PAI no altera el mensaje de petición los servicios finales se ven obligados a comprender estas cabeceras. En el documento “Manual Usuario Generación de Servicios” de detalla cómo realizar las acciones necesarias sobre un servicio web CXF para procesar una cabecera de este tipo.

3.3.3 Namespace de Soap envelope

Es recomendable, con la finalidad de obtener una validación correcta de la firma realizada sobre el cuerpo Soap, tal y como marca el estándar, definir correctamente los namespaces de envoltorio Soap.

Por ello, a nivel de nodo “*Envelope*” solo debe aparecer una única definición de namespaces para Soap, evitando cabeceras como puede ser la siguiente, donde observamos dos definiciones de namespace iguales, junto a prefijos distintos:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
```

Esto crea inconsistencia en la validación de la firma que se hay realizado, provocando un error en la misma. La forma correcta de los namespaces es:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
```

3.3.4 Definición de atributos y métodos legibles

No se permiten servicios que expongan métodos en los que un archivo XML lleve la lógica asociada como un solo parámetro de llamada al método, ya que en esta práctica encontramos ciertas desventajas.

Petición:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:urn="urn:es:gva:comunicaciones:ComunicacionesWS">
      <soapenv:Header/>
      <soapenv:Body>
            <urn:getComunicaciones>
                  <!--Optional:-->
                  <data><![CDATA[<?xml version="1.0" encoding="UTF-8"
standalone="yes"?>
                                    <REQUEST_CONSULTA_COMUNICACIONES
xmlns="urn:gva:es:comunicaciones:requestConsultaComunicaciones">
<CODIGO_ENTIDAD>L046111-00000000</CODIGO_ENTIDAD>
<NUMERO_IDENTIFICACION>00265889D</NUMERO_IDENTIFICACION>
<ACUSE>true</ACUSE>
                                    <PAGINACION>
<TAMANYO_PAGINA>1000</TAMANYO_PAGINA>
<NUMERO_PAGINA>0</NUMERO_PAGINA>
                                    </PAGINACION>
                                    </REQUEST_CONSULTA_COMUNICACIONES> ]]>
                  </data>
            </urn:getComunicaciones>
      </soapenv:Body>
</soapenv:Envelope>
```

Esquemas:

```
<xsd:element name="getComunicaciones" type="getComunicaciones"/>
<xsd:complexType name="getComunicaciones">
                <xsd:sequence>
                    <xsd:element minOccurs="0" name="data" type="xs:string"/>
                </xsd:sequence>
</xsd:complexType>
```

Como vemos en el ejemplo anterior, la inclusión de un XML sin estandarizar, en un parámetro definido en los esquemas como "*string*" en la petición, tiende a aumentar la complejidad de la capa de negocio al tener que interpretar dicho XML incrustado.

En cuanto a la parte del consumidor, dificulta la comprensión del servicio al no estar basado en un estándar público, siendo más complicado el testeo y posterior tratamiento del mensaje en la capa de negocio de la aplicación.

Además a nivel de orquestación en la PAI, este tipo de prácticas hace imposible el tratamiento del mensaje transportado, así como la validación del mismo contra esquemas.

Para evitar esta problemática se debe eludir este tipo de prácticas en los servicios web y realizar la definición de esquemas precisos para cada uno de los elementos que van a viajar en la petición. A

continuación vemos una aproximación de cómo deberían ser las peticiones y los esquemas bien estructurados.

Petición:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:urn="urn:es:gva:comunicaciones:v3:ComunicacionesWS">
   <soapenv:Header/>
   <soapenv:Body>
      <urn:getComunicaciones>
         <urn:peticion>
            <urn:codigoEntidad>GV</urn:codigoEntidad>
            <urn:fechaFin>2016-02-16T17:30:04.347+01:00</urn:fechaFin>
            <urn:paginacion>
               <urn:tamanyoPagina>2</urn:tamanyoPagina>
               <urn:numeroPagina>1</urn:numeroPagina>
            </urn:paginacion>
         </urn:peticion>
      </urn:getComunicaciones>
   </soapenv:Body>
</soapenv:Envelope>
```

Extracto de esquema:

```
<xs:complexType name="consultaComunicacionesRequest">
              <xs:sequence>
                     <xs:element name="codigoEntidad" type="types:typeCodigoEntidad" />
                     <xs:element name="numeroComunicacion" type="types:typeNumeroComunicacion"
                            minOccurs="0" />
                     <xs:element name="numeroIdentificacion" type="types:typeNumeroIdentificacion"
                            minOccurs="0" />
                     <xs:element name="fechaInicio" type="xs:dateTime"
                            minOccurs="0" nillable="true" />
                     <xs:element name="fechaFin" type="xs:dateTime" minOccurs="0"
                            nillable="true" />
                     <xs:element name="acuse" type="xs:boolean" minOccurs="0"
                            nillable="true" />
                     <xs:element name="estados" minOccurs="0" nillable="true">
                            <xs:complexType>
                                   <xs:sequence>
                                          <xs:element name="estado"
type="types:typeEstadoComunicacion"
                                                 minOccurs="0" nillable="true"
maxOccurs="unbounded" />
                                   </xs:sequence>
                            </xs:complexType>
                     </xs:element>
                     <xs:element name="paginacion" type="tns:paginacion"
                            minOccurs="0" nillable="true" />
              </xs:sequence>
       </xs:complexType>
```

3.3.5 Recomendación respecto a definición de esquemas

Una recomendación habitual en la definición de servicios web es la de **separar el modelo de datos de las distintas peticiones (*XSD's*) de las correspondientes definiciones asociadas a los métodos y operaciones del servicio (WSDL).** Aunque es muy habitual el uso autocontenido en los wsdl de los modelos de datos, la separación de ambos, ayuda en la organización y legibilidad del servicio, reduce el tamaño/complejidad del WSDL, permite utilizar editores especializados para el diseño del schema XSD y permite reutilizar schemas y namespaces.

A la hora de diseñar el schema XSD, conviene crear tipos y elementos globales (a nivel raíz) para poder reutilizarlos, tanto a nivel de elementos XML como clases del lenguaje de implementación del servicio web.

3.3.5.1 Patrón de diseño de XML Schema

Existen muchas maneras de diseñar XML Schemas válidos, pero si los queremos utilizar junto a los descriptores de contratos de servicios web (WSDL), tendrán que cumplir además con las especificaciones WS-I Basic Profile con el propósito de cumplir unos requisitos de interoperabilidad y asegurar la compatibilidad en las invocaciones entre los mismos.

Las buenas prácticas en el diseño de XML Schema, giran en gran parte, en torno a la existencia de:

- Declaración de elementos (*element*) globales (elemento global es cualquier hijo de un nodo <*schema*> o una referencia directa a uno de ellos) o locales (Se anidan dentro de la estructura del <*schema*> y no son hijos directos de la raíz).
- Ídem para con la declaración de tipos definidos (*complexType* ó *simpleType*)

Existen varios patrones de diseño aceptados, entre los cuales se recomienda el uso del conocido como "*Salami Slice*" que consiste declarar los elementos a nivel global del documento y utilizar las referencias a los mismos en los tipos complejos, facilitando la reutilización de componentes.

3.3.5.2 Representaciones del null en XML Schema

Hay dos maneras cuya utilización recomienda el WS-I Basic Profile para representar un valor nulo para elementos: ya sea con el atributo *nillable=true*, o con *minOccurs=0*.

Para matizar sobre su uso, se debe utilizar *minOccurs=0* para definir un elemento como opcional y *nillable=true* para indicar que un elemento puede ser vacío pero siempre estará presente en el mensaje XML.

3.3.5.3 Uso de los tags elementFormDefault y attributeFormDefault

En ocasiones cuando procesamos mensajes en nuestras aplicaciones de integración encontramos que algunas acciones especificadas como mapas de transformación o asignaciones *xpath()* no son

ejecutadas como se esperan, después de revisar los esquemas de definición y los mensajes definidos no encontramos aparente lógica en la causa del error ya que los nombres de nodos (*xs:element*), atributos (*xs:attribute*) y nombres de espacios (xs:xmlns) utilizados están correctos.

Los parámetros *elementFormDefault* y *attributeFormDefault* especifican cómo deben ser formados los correspondientes XML, a nivel de elementos/nodos y a nivel de atributos de los elementos/nodos. Estas definiciones están muy relacionadas con los conceptos de elementos *globales* y *locales* en el XSD. En resumen un elemento global es cualquier hijo de un nodo *schema* o una referencia directa a uno de ellos. Un elemento local es cualquiera que no sea global.

En el siguiente ejemplo mostramos detalle de ambos:

```
 1: <?xml version="1.0" encoding="UTF-8"?>
 2: <schema xmlns="http://www.w3.org/2001/XMLSchema"
 3:         targetNamespace="http://www.intertech.com/band"
 4:         xmlns:tns="http://www.intertech.com/band"
 5:         elementFormDefault="qualified"
 6:         attributeFormDefault="unqualified">
 7:
 8:     <complexType name="Band_Member_Type">
 9:         <sequence>
10:             <!-- LOCAL: not an immediate child of "schema" -->
11:             <element name="fullName" type="string" />
12:
13:             <!-- GLOBAL: a ref to an immediate child of "schema" -->
14:             <element ref="tns:instrument" maxOccurs="unbounded" />
15:         </sequence>
16:     </complexType>
17:
18:     <!-- GLOBAL: an immediate child of "schema" -->
19:     <element name="instrument" type="string" />
20:
21:     <!-- GLOBAL: an immediate child of "schema" -->
22:     <element name="bandName" type="string" />
23:
24:     <!-- GLOBAL: an immediate child of "schema" -->
25:     <element name="band">
26:         <complexType>
27:             <sequence>
28:                 <!-- GLOBAL: a ref to an immediate child of "schema" -->
29:                 <element ref="tns:bandName" />
30:
31:                 <!-- LOCAL: not an immediate child of "schema" -->
32:                 <element name="member" type="tns:Band_Member_Type" minOccurs="1"
33:                         maxOccurs="unbounded" />
34:                 </sequence>
35:             <attribute name="id" type="string" />
36:         </complexType>
37:     </element>
38: </schema>
```

Así pues, *elementFormDefault=qualified,* significa que todos los elementos (globales y locales) deben ser 'qualified', esto es, deben ser asociados a un determinado namespace. Por otra parte, *elementFormDefault=unqualified* significa que sólo los elementos globales deben ser asociados con namespaces, no así los elementos llamados 'locales'. Esto mismo se puede aplicar al parámetro *attributeFormDefault* el cual aplicará la norma a los nodos *<xs:attribute>.*

La recomendación es emplear *elementFormDefault* '***qualified' y*** *attributeFormDefault* "***unqualified***", para evitar colisiones y conflictos en caso de indefiniciones adecuadas de *namespaces*.

Un ejemplo de esta definición lo mostramos seguidamente:

```
<xs:schema targetNamespace="http://intermediacion.redsara.es/scsp/esquemas/V3/peticion"
elementFormDefault="qualified" attributeFormDefault="unqualified"
xmlns="http://intermediacion.redsara.es/scsp/esquemas/V3/peticion"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:ns1="http://intermediacion.redsara.es/scsp/esquemas/datosespecificos">
        <xs:element name="Apellido1">
                <xs:simpleType>
                        <xs:restriction base="xs:string">
                                <xs:maxLength value="40"/>
                        </xs:restriction>
                </xs:simpleType>
        </xs:element>
        <xs:element name="Apellido2">
                <xs:simpleType>
                        <xs:restriction base="xs:string">
                                <xs:maxLength value="40"/>
                        </xs:restriction>
                </xs:simpleType>
        </xs:element>

.....

</xs:schema>
```

Por tanto la petición a este servicio seria correcta:

```
<soapenv:Body>
        <pet:Peticion>
                <pet:Atributos>
              ...
                </pet:Atributos>
                <pet:Solicitudes Id="">
                        <pet:SolicitudTransmision>
                                <pet:DatosGenericos>
                                        <pet:Emisor>
                     ...
                                        </pet:Emisor>
                                        <pet:Solicitante>
                                                ...
                                        </pet:Solicitante>
                                        <pet:Titular>
                                                <pet:TipoDocumentacion>NIF</pet:TipoDocumentacion>
                                                <pet:Documentacion>99999999R</pet:Documentacion>
                                                <pet:Nombre>JUAN</pet:Nombre>
                                                <pet:Apellido1>ESPAÑOL</pet:Apellido1>
                                                <pet:Apellido2>ESPAÑOL</pet:Apellido2>


                                        </pet:Titular>
```

```
                                        <pet:Transmision>
                                            ...
                                        </pet:Transmision>
                                    </pet:DatosGenericos>
                                </pet:SolicitudTransmision>
                            </pet:Solicitudes>
                        </pet:Peticion>
</soapenv:Body>
```

Como podemos ver en la petición, con el *elementFormDefault=qualified* todos los elementos globales deben estar asociados a un determinado namespace. Seria además posible obviar las referencias en este caso a **pet** simplemente utilizando lo siguiente en el nodo Petición:

```
<Peticion xmlns="http://intermediacion.redsara.es/scsp/esquemas/V3/peticion">
```

3.3.5.4 Espacio de nombres

En XML, como en muchos lenguajes de programación, frecuentemente se debe especificar un "espacio de nombres" para varios elementos y atributos. Esto permite distinguir entre elementos que tienen el mismo nombre, pero diferentes propósitos y tal vez diferentes orígenes. XML hace referencia a un espacio de nombres mediante un identificador URI que no tiene por qué ser accesible. Por ejemplo, el espacio de nombres del esquema XML es http://www.w3.org/2001/XMLSchema. Para facilitar la tarea, sin embargo, también se asigna un alias o un prefijo. Por ejemplo, justo en el ejemplo anterior el espacio de nombres de esquema tiene un prefijo "xs:". Recuerde que el alias es tan solo eso: un alias. El identificador URI es que realmente importa.

En el caso de los servicios de verificación se debe seguir la definición de espacio de nombres del protocolo SCSP (Sustitución de Certificados en Soporte Papel) cuya especificación está disponible en el Portal de Administración electrónica PAE/CTT en la dirección http://administracionelectronica.gob.es/es/ctt/scsp.

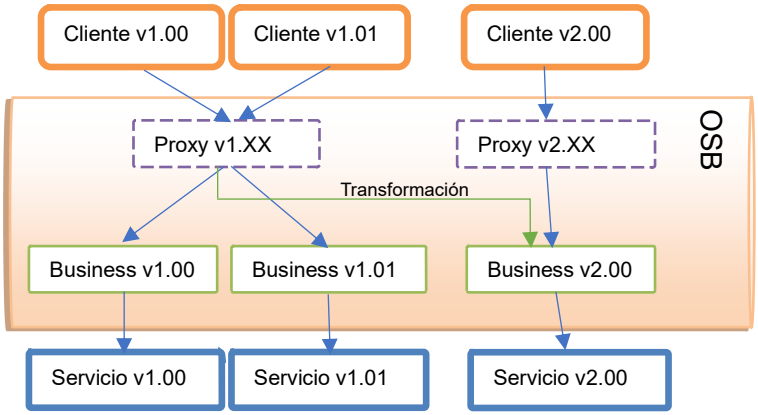
3.4 Estrategias de versionado de servicios y XSD

Por necesidades de los proyectos, los servicios web se encuentran en constante cambio y evolución. Las bases y principios de una arquitectura orientada a servicios (SOA) deben asegurar que los proveedores de servicios puedan:

- Liberar una nueva versión de un servicio, después de que los consumidores del servicio hayan probado y certificado la nueva versión del servicio.

- Liberar una nueva versión de un servicio, sin esperar a que los consumidores adapten la invocación al nuevo servicio.

En la PAI no se aceptarán escenarios de *transformación* como el que se muestra en la imagen, puesto que implica tener un conocimiento del negocio final, que en la mayoría de los casos se salen de las funciones de intermediación que se atribuye al bus de la PAI.



La recomendación, es que las distintas revisiones de los servicios guarden compatibilidad con versiones anteriores, para producir el menor número de incidencias en el consumo de las nuevas versiones.

3.5 Estrategia de comunicación de adjuntos

En el caso de que el servicio tenga que transmitir ficheros asociados al servicio, es interesante conocer y gestionar las diferentes modalidades de hacerlo. Entre las más importantes y utilizadas encontramos MTOM y Base64.

La transmisión de ficheros en Base64 sobre el body es una práctica común históricamente y el método más utilizado y extendido de transmisión de ficheros, pero principalmente una desventaja muy grande, ya que la codificación en Base64 añade al contenido original aproximadamente un 33% más de “peso”.

MTOM proporciona una forma más efectiva y normalizada por W3C de transmitir ficheros, donde en lugar de proporcionar el contenido en el body codificado en Base64, envía los datos binarios como adjunto.

A efectos prácticos es recomendable la utilización del método de transmisión MTOM siempre que se deban transmitir adjuntos, salvo cuando el tamaño máximo de los ficheros objeto de la transmisión no superen en ningún caso el tamaño de 1MB.

3.6 Definición de la Taxonomía de Servicio

Con el fin de tener una correcta estructuración de los servicios incorporados a la infraestructura SOA de la PAI es de vital importancia mantener una correcta taxonomía de servicios, que permita categorizar dichos servicios a publicar conforme a criterios diferentes que sirvan como base en la definición de alertas, monitorización, gestión de recursos asociados, etc.

La información que normalmente se requiere para categorizar estos servicios hace referencia a características relativas:

- **Semántica e información del servicio**. En la mayoría de los casos esta información viene auto contenida en los *wsdl* y *xsd* del servicio, pero en ocasiones, esta información se debe completar con mayor detalle. Por ello se exige un contrato de integración en el que se facilite toda la información relevante tanto de entrada como de salida para consumir el servicio y tratar adecuadamente las respuestas que éste proporcione, tanto por el sistema mediador (bus de interoperabilidad) como por el usuario/consumidor final.
- **Dimensionamiento del Sistema**. Los recursos a destinar a la hora de exponer los diferentes servicios, son dependientes en ocasiones, de las características como los perfiles de tráfico, tamaño de las peticiones/respuestas, etc. Es importante detallar el perfil de tráfico esperado en número de peticiones por minuto/segundo, nivel de concurrencia máximo, así como si el tráfico es estacionario, periódico, etc. Todas estas características, más las que se puedan considerar de interés por parte de los desarrolladores deben ser incorporadas en el formulario de alta de proveedor. Con ellas se definen los acuerdos de nivel de servicio a cumplir entre los diferentes componentes. Puede consultar los formularios de alta de proveedor en este enlace.
- **Comunicaciones**: En ocasiones, se producen restricciones de acceso administrativas, que obligan a considerar temas como direcciones IP origen y destino, protocolos a nivel de VPN,

etc. La exposición de los servicios de la PAI se hace mediante protocolo https securizando las peticiones mediante WSSecurity en cabecera SOAP. No existe de momento exposición de servicios REST, JSON-RPC, etc.

- **Requisitos de Integración**: Ciertos componentes y servicios desplegados, con frecuencia requieren de la interacción con otros elementos o servicios que deban ser configurados y definidos en sincronía. Todas las interacciones y dependencias que sean necesarias para el correcto funcionamiento del servicio deben ser establecidas en la solicitud a la PAI mediante el formulario del proveedor, haciendo constar en la petición todos los requerimientos de integración accesorios que deban ser contemplados.

3.7 Servicios REST

La arquitectura REST se define como un conjunto de principios arquitectónicos por los cuales se diseñan servicios web haciendo foco en los recursos del sistema, incluyendo cómo se accede al estado de dichos recursos y cómo se transfieren por HTTP hacia clientes escritos en diversos lenguajes. Se caracteriza por la siguiente serie de fundamentos clave:

- Un **protocolo cliente/servidor** sin estado: cada mensaje HTTP contiene toda la información necesaria para comprender la petición. Como resultado, ni el cliente ni el servidor necesitan recordar ningún estado de las comunicaciones entre mensajes.
- Un **conjunto de operaciones** bien definidas que se aplican a todos los recursos de información: HTTP en sí define un conjunto pequeño de operaciones, las más importantes son **POST** (genera un nuevo elemento)**, GET** (consulta)**, PUT** (modificación del estado del recurso) **y DELETE** (eliminación del recurso).
- Los parámetros que se le pueden pasar tienen que ser representados en Strings (en http lo que viaja es siempre string).
- Una **sintaxis universal** para identificar los recursos. En un sistema REST, cada recurso es direccionable únicamente a través de su URI.
- El **uso de estándares**.

  El resultado que va a devolver el servicio tendrá que ser String, en un formato estándar, originalmente ese formato era XML, actualmente se suele usar JSON.

Para su inclusión siguiendo unos mínimos de securización en los servicios publicados en la plataforma PAI, se ha determinado una norma de consumo de servicios REST publicados en la PAI. Consta de dos partes:

API-KEY

Inclusión en la cabecera HTTP del mensaje realizado a la PAI un campo destinado a un código o token proporcionado por la Plataforma en el momento de la solicitud de alta al servicio.

Este código se compone de 32 caracteres y es único por cada aplicación.

x-api-key: [VALOR]

Como ejemplo:

x-api-key: 1345c25df49d43c8153d2e91c8346ba4

Aplicación

Inclusión en la cabecera HTTP de un campo de usuario con nombre de Aplicación, que debe contener la aplicación de la consulta con el que se ha realizado el alta al servicio.

aplicacion: [VALOR]

Como ejemplo:

aplicacion: TEST_00001

Se han de incluir ambas partes para un consumo correcto del servicio. De esta forma, como ejemplo del mensaje final:

```
POST https://innovacion-pre.gva.es/pai_bus_inno/REST/REST_operacion HTTP/1.1
Accept-Encoding: gzip,deflate
Content-Type: application/json
Content-Length: 60
Host:  innovacion-pre.gva.es
x-api-key: 1345c25df49d43c8153d2e91c8346ba4
aplicacion: TEST_00001
Accept: application/xml
Connection: Keep-Alive
User-Agent: Apache-HttpClient/4.1.1 (java 1.5)
{
  "usuario": "prueba",
```

```
  "password": "test001"
}
```

NOTA: Los servicios en modalidad REST utilizan el protocolo HTTP y sus operaciones, esto no significa que la conexión tenga que ser en http sin cifrar, es posible securizar la conexión utilizando https sin afectar al funcionamiento del mismo y con el mismo funcionamiento que normalmente. Este protocolo (https) es el utilizado por la PAI para la publicación de servicio REST.

Un ejemplo de inclusión de estas cabeceras en una petición Post a un servicio Rest desde una aplicación Android (java) con API level 21 sería el siguiente:

```
HttpPost httpPost = new HttpPost(serverURL);

httpPost.addHeader("x-api-key","1345c25df49d43c8153d2e91c8346ba4");
httpPost.addHeader("aplicacion","TEST_00001");
```

3.7.1 Modalidades de Publicación de servicios REST

Existen varias opciones para la publicación de un determinado servicio en la PAI cuando se desea un consumo del servicio en estándar REST. Por un lado es posible implementar la conversión de un servicio SOAP a REST y por otro lado, es posible la publicación de un servicio de naturaleza REST como tal.

3.7.1.1 Servicios SOAP a REST

Los servicios que proporciona el proveedor en estándar SOAP a los cuales se les puede anexar una fachada de entrada REST mediante un recurso de bus. De esta forma, el consumidor de dicho servicio lo utiliza como si de un REST puro se tratase y el bus de la PAI realiza la conversión de la llamada al servicio final en SOAP.

Esta modalidad requiere realizar un estudio de los path, las operaciones http a asignar a cada uno de los métodos, así como el formato de los menajes. Por tanto, se recomienda consultar con la Plataforma en caso de querer acogerse a esta posibilidad.

3.7.1.2 Servicios REST

La publicación de un servicio proporcionado en REST requiere las mismas características que la publicación de un servicio SOAP, en cuanto a documentación propia del servicio (juego de pruebas, contrato de integración, ...).

Además se recomienda una serie de buenas prácticas para el correcto funcionamiento del servicio y la integración del mismo en las distintas aplicaciones que puedan consumirlo:

- Usa los códigos de operación HTTP correctamente, evitando por ejemplo el uso del código GET para la inserción de un registro en lugar de utilizar POST para este fin.
- Las llamadas GET y HEAD no deben alterar el estado de los registros ni utilizarse para el transporte de datos sensibles.
- Las llamadas erróneas deben devolver un error suficientemente claro y bien formado siguiendo la especificación lo más cercana al punto 5.2 Especificación Soap Fault.
- No utilizar verbos en las URI, utilizar nombres y en plural.
- Respetar el patrón par nombre/valor en Json en todos los escenarios:

```
{"Error": "El usuario no existe"}
```

4 Entorno tecnológico de la PAI

4.1 Política de seguridad

Las peticiones que se realizan a la PAI para los servicios verificación e instrumentales se securizan y autentican utilizando mecanismos WS-Security. Concretamente, se permite seguridad WS-Security con identificador de clave “Binary Security Token”.

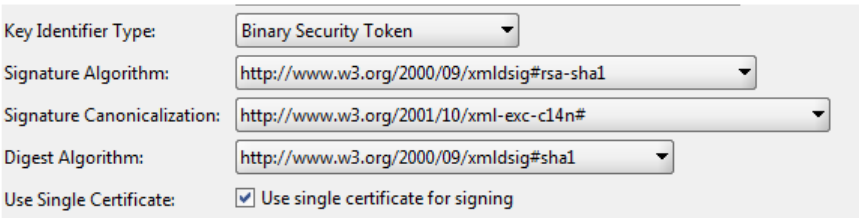
La PAI cuenta con una política de autorización de firma de mensaje en cabecera SOAP, siguiendo las especificaciones de WS-Security. A los servicios se les aplica una política que exige que las peticiones vengan firmadas mediante un certificado que pueda validar la ACCV.

Los consumidores de servicios de verificación de datos de la GVA o servicios instrumentales, pueden utilizar tanto certificado de Sello de Órgano como certificados de aplicación de la ACCV.

Los consumidores de servicios de verificación de datos ofrecidos por la Plataforma de Intermediación de Datos (PID) a través de la PAI, están obligados a utilizar certificados de Sello de Órgano de la ACCV para firmar sus peticiones.

La particularización de los parámetros de configuración de seguridad se puede ver en la siguiente figura:

Ilustración



Especificación disponible en:
https://www.oasis-open.org/committees/download.php/16785/wss-v1.1-spec-os-x509TokenProfile.pdf

4.2 Proveedores de servicios

Tal y como se describe en la parte de clientes de servicio, hay diversas tecnologías que hacen posible la generación de servicios web. En el ‘Manual de Usuario para Generación de servicios’ se muestran los pasos para la generación de un servicio web con el framework CXF siguiendo los patrones descritos en este documento. Si la aplicación sigue el estándar de la oficina java, se deberá consultar la documentación expuesta en su espacio de confluence.

En el desarrollo en sí de los servicios, existen en la actualidad multitud de entornos de desarrollo (frameworks) que permiten agilizar los procesos de integración, así como la publicación rápida de

webservices centrándose únicamente en la lógica de negocio a implementar, obviando las labores accesorias. La facilidad e integración con los IDEs de uso habitual (Eclipse, Netbeans, JDeveloper, IntelliJ, BlueJ, etc) permite un ágil y rápido desarrollo teniendo curvas de aprendizaje con tiempos muy cortos. En el punto 3 se esbozan directrices generales de orientación para el desarrollo de servicios en entornos SOA.

4.3 Trazabilidad en el consumo de servicios de la PAI

Las aplicaciones que consumen los servicios de la PAI, deben guardar información suficiente que permita hacer un seguimiento de la trazabilidad de las transacciones realizadas e intercambiadas con la plataforma.

Es importante destacar que las aplicaciones que hacen uso de la PAI, deben almacenar los correspondientes identificadores que permitirán hacer un seguimiento de dichas transacciones, ante futuras auditorias del proveedor del servicio.

Por parte de la PAI y de manera general a todos los servicios desplegados en la misma, se registran los siguientes parámetros de las peticiones realizadas;

- Cabecera de la petición (Header).
- Timestamp en el que se recibió la petición
- Servicio y operación destino de la petición
- Tipo de mensaje; *request* | *response* | *fault*
- Certificado del consumidor.
- Procedimiento de la petición.

Además específicamente para los ***servicios instrumentales*** se registra el “Id_trazabilidad” y para los ***servicios de verificación*** se registra el “IdPeticion” y el/los “Idsolicitud”. Seguidamente procedemos a exponer estas peculiaridades y describimos las características que deben implementar las peticiones a realizar en las llamadas a los servicios expuestos en la PAI.

NOTA: Para los servicios con política aplicada se registra adicionalmente a todo lo descrito anteriormente el “thread-id” como identificador inequívoco del hilo.

4.3.1 Trazabilidad en los servicios instrumentales.

Con el fin de minimizar los cambios a realizar en los correspondientes desarrollos, e igualmente, impactar de la forma más liviana posible en el rendimiento de la plataforma, se ha optado por

incorporar una cabecera de mensaje SOAP, en la petición al servicio web, que incluya el tag 'Id_trazabilidad' que, permitirá hacer el correspondiente seguimiento posterior.

Ejemplo:

```
<Id_trazabilidad xmlns="http://dgti.gva.es/interoperabilidad">68254ed222d65eeb-CODIGO_CATI-20140130184955000</Id_trazabilidad>
```

Este tag se compone del número de serie del certificado utilizado para firmar la petición, CODIGO_CATI que toma el valor del código CATI de la aplicación y el instante de la petición codificado como año,mes,dia,hora,minuto,segundo,milisegundos (patrón YYYYMMddHHmmssSSS y formato 24 horas). Estos tres componentes del tag de trazabilidad se han de separar por un carácter '-' como se aprecia en el formato de ejemplo mostrado anteriormente, se define además el namespace correspondiente "**http://dgti.gva.es/interoperabilidad**".

4.3.1.1 Trazabilidad en los servicios instrumentales en el Bus de Innovación.

Para el consumo correcto de servicios instrumentales en el entorno de innovación, se deben realizar algunos ajustes sobre el Id_trazabilidad. Dado que en este entorno no se realiza firma WS-Security con certificado, no hay que incluir el número de serie como primer campo del Id_trazabilidad, por tanto, este campo no se completará, quedando el tag de la siguiente forma:

Ejemplo:

```
<Id_trazabilidad xmlns="http://dgti.gva.es/interoperabilidad">-CODIGO_CATI-20140130184955000</Id_trazabilidad>
```

En sustitución de la validación del certificado que se realiza en el bus instrumental, en este caso, en el bus de innovación, la validación se realiza sobre la IP o rango de IPs consumidora/s del servicio.

Este tag se compone del CODIGO_CATI que toma el valor del código CATI de la aplicación y el instante de la petición codificado como año,mes,dia,hora,minuto,segundo,milisegundos (patrón YYYYMMddHHmmssSSS y formato 24 horas). Estos componentes del tag de trazabilidad se han de separar por un carácter '-' como se aprecia en el formato de ejemplo mostrado anteriormente, se define además el namespace correspondiente "**http://dgti.gva.es/interoperabilidad**".

NOTA: En caso de existir por parte del proveedor del servicio alguna necesidad de utilizar la primera parte de este Id_trazabilidad con el fin de trasladar algún código que queda vacío en este ámbito, pónganse en contacto con la PAI y se estudiará el caso.

4.3.1.2 Ejemplo de petición incluyendo trazabilidad en Servicios Instrumentales

Un ejemplo de código en Java - CXF para incluir esta cabecera sería del estilo:

```
java.util.Date date = Calendar.getInstance().getTime();
SimpleDateFormat dt1 = new SimpleDateFormat("yyyyMMddHHmmssSSS");
```

```
List<Header> headersList = new ArrayList<Header>();
Header testSoapHeader1 = new Header(new QName("http://dgti.gva.es/interoperabilidad", "Id_trazabilidad"), getSerial()+"-
  "+ getCodigoCati() + "-"+ dt1.format(date), new JAXBDataBinding(String.class));
headersList.add(testSoapHeader1);
client.getRequestContext().put(Header.HEADER_LIST, headersList);
```

NOTA: Las funciones *getSerial()* y *getCodigoCati()*, recuperarán el número de serie del certificado que se esté utilizando y el código CATI de la aplicación utilizada. La definición de estas funciones se puede ver en detalle en el documento: *CONSTRUCCION-2-Manual_Usuario_Consumo_Instrumentales_y_de_Verificacion (anexos I y II)* de la web de la DGTIC.

Con el código anterior, la cabecera se generaría automáticamente y quedaría de la siguiente manera:

```
<Id_trazabilidad xmlns="http://dgti.gva.es/interoperabilidad">68254ED222D65EEB-CODIGO_CATI-20140130184955000</Id_trazabilidad>
```

Existen documentación de inclusión de esta misma cabecera referenciada arriba, en tecnología CXF (documento Manual de usuario para el consumo de servicios instrumentales y de verificación) y soapUI (documento GUIA_CONFIGURACION_SOAP_UI).

A continuación se adjunta el ejemplo de una petición completa incluyendo la etiqueta destinada a la trazabilidad.

```
<soapenv:Envelope xmlns:sch="http://www.telefonica.es/MI/InterfazSimplificado/schemas"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
        <soapenv:Header>
            <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
                <wsse:BinarySecurityToken EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-
message-security-1.0#Base64Binary" ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3"
wsu:Id="X509-
DCB9D5EE6640DE7420142131045237816">MIII4DCCBsigAwIBAgIIK9mt3c2fCZgwDQYJKoZIhvcNAQEFBQAwQzETMBEGA1UEAwwKQUNDVkNBLTEyMD
EQMA4GA1UECwwHUEtJQUNDVjENMAsGA1UECgwEQUNDVjELMAkGA1UEBhMCRVMwHhcNMTMxMDA0MTAzNTQxWhcNMTYxMDAzMTAzNTQxWjC
BuTE7MDkGA1UEAwyRElSRUNDSU9OIEdFTkVSQUwgREUgVEVDTk9MT0dJQVMgREUgTEEgSU5GT1JNQUNJT04xEjAQBgNVBAUTCVM0NjExMDAxQTE
bMBkGA1UECwSc2VsbG8gZWxlY3Ryyw7N.....++mD5nYvY4BriImaW+bRJeaR+5xS20uvwM6bo4y929jqeABB9v5FgBwrSh98C6MtEQ/J9Z/iJZwlGRQyo
lgt6vtocQat3YOKE03/Gd7rjmrSYsECow4STBKvd4QRqcp5b5fs8jNt97nrgsdaM+nFSFkHy8pIBecgoDbuG9JLzxJ+XVO2RSvijNb1HX7qUVrkARkxVXdDHm7x
MNGSPKl9a5ea0ShWTCDU7ctquc2E5oDK6Tv+bDf7Z1Z4JBM0T0Vnh75+PDnizmFhSzAvjIGujZigGm2ouzch34MfG0/KpflYaUnldGPbMAP4SGar77COaVw
VJ8mrfOeMunpU=</wsse:BinarySecurityToken>
                <ds:Signature Id="SIG-DCB9D5EE6640DE7420142131045238020"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
                    <ds:SignedInfo>
                        <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                            <ec:InclusiveNamespaces PrefixList="sch soapenv"
xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#"/>
                        </ds:CanonicalizationMethod>
                        <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
                        <ds:Reference URI="#id-DCB9D5EE6640DE7420142131045237819">
                            <ds:Transforms>
                                <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
                                    <ec:InclusiveNamespaces PrefixList="sch"
xmlns:ec="http://www.w3.org/2001/10/xml-exc-c14n#"/>
                                </ds:Transform>
                            </ds:Transforms>
```

```
                                                <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
                                                <ds:DigestValue>nXA9tKHn/MaC1V1P/BzI6rY1xAY=</ds:DigestValue>
                                        </ds:Reference>
                                </ds:SignedInfo>

        <ds:SignatureValue>LlnonOkwUP/N6EmDub9J+jPMW/Q03p042Dge6+B8cQ/4S67VtzHbJGab0e4BAMj2nx3fz6wLMP3P
ACiDEG5HlIJjbZvpxlEehWzN8LGZE8wwh/0thtiHzAD5LwCJND5JGXsK7ps10s0EIntCPEr9GpoQ
ZXhoJzzWBdTThI718uQ=</ds:SignatureValue>
                                <ds:KeyInfo Id="KI-DCB9D5EE6640DE7420142131045237817">
                                        <wsse:SecurityTokenReference wsu:Id="STR-DCB9D5EE6640DE7420142131045237818">
                                                <wsse:Reference URI="#X509-DCB9D5EE6640DE7420142131045237816"
ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3"/>
                                        </wsse:SecurityTokenReference>
                                </ds:KeyInfo>
                        </ds:Signature>
                </wsse:Security>
                <Id_trazabilidad xmlns="http://dgti.gva.es/interoperabilidad">68254ED222D65EEB-CODIGO_CATI-
20140130184955000</Id_trazabilidad>
        </soapenv:Header>
        <soapenv:Body wsu:Id="id-DCB9D5EE6640DE7420142131045237819" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-wssecurity-utility-1.0.xsd">
                <sch:EnviarReq>
                        <Remitente>PRUEBASGVA</Remitente>
                        <Destinatarios>
                                <Para>N_TELF</Para>
                        </Destinatarios>
                        <TextoSMS>Hola Mundo</TextoSMS>
                        <NotificacionEntrega>No</NotificacionEntrega>
                </sch:EnviarReq>
        </soapenv:Body>
</soapenv:Envelope>
```

4.3.1.3 Ejemplo de petición incluyendo trazabilidad en Servicios Innovación

En cuanto al código necesario en Java - CXF para incluir esta cabecera no dista demasiado del punto anterior y la aproximación seria la siguiente:

```
java.util.Date date = Calendar.getInstance().getTime();
SimpleDateFormat dt1 = new SimpleDateFormat("yyyyMMddHHmmssSSS");


List<Header> headersList = new ArrayList<Header>();
Header testSoapHeader1 = new Header(new QName("http://dgti.gva.es/interoperabilidad", "Id_trazabilidad"),
  getCodigoCati() + "-"+ dt1.format(date), new JAXBDataBinding(String.class));
headersList.add(testSoapHeader1);
client.getRequestContext().put(Header.HEADER_LIST, headersList);
```

NOTA: La función *getCodigoCati()*, recuperará el código CATI de la aplicación utilizada. La definición de esta función se puede ver en detalle en el documento: *CONSTRUCCION-2-Manual_Usuario_Consumo_Instrumentales_y_de_Verificacion (anexo II)* de la web de la DGTIC.

Con el código anterior, la cabecera se generaría automáticamente y quedaría de la siguiente manera:

```
<Id_trazabilidad xmlns="http://dgti.gva.es/interoperabilidad">-CODIGO_CATI-20140130184955000</Id_trazabilidad>
```

A continuación se adjunta el ejemplo de una petición completa incluyendo la etiqueta destinada a la trazabilidad en innovación

```
<soapenv:Envelope xmlns:sch="http://www.telefonica.es/MI/InterfazSimplificado/schemas"
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
        <soapenv:Header>
                <Id_trazabilidad xmlns="http://dgti.gva.es/interoperabilidad">-CODIGO_CATI-20140130184955000</Id_trazabilidad>
        </soapenv:Header>
        <soapenv:Body >
                <sch:EnviarReq>
                        <Remitente>PRUEBASGVA</Remitente>
                        <Destinatarios>
                                <Para>N_TELF</Para>
                        </Destinatarios>
                        <TextoSMS>Hola Mundo</TextoSMS>
                        <NotificacionEntrega>No</NotificacionEntrega>
                </sch:EnviarReq>
        </soapenv:Body>
</soapenv:Envelope>
```

4.3.2 Trazabilidad en Servicios de Verificación

En el caso de los servicios de verificación, la trazabilidad ha de ser igualmente registrada por la aplicación consumidora del servicio, pero los valores a guardar, vienen directamente informados como parámetros/tags del mensaje. Para ello se registran siempre los campos de '*IdSolicitud*' (para peticiones con más de un "*IdSolicitud*" se registran todos los incluidos en el mensaje) y '*IdPeticion*' del mensaje (ambos son tags obligatorios String 16/26 caracteres).

Adicionalmente a estos dos campos, se ha de registrar el código del procedimiento administrativo que faculta y habilita la petición que se realiza por parte del organismo que cede los datos. Este procedimiento puede ser informado de dos formas. Por una parte, a través del tag '*CodProcedimiento*' que es informado directamente en el mensaje en gran parte de los servicios de verificación. Se debe poblar con el código GUC adecuado.

En los casos en los que este campo no sea informado, bien porque no exista, bien porque sea opcional, se deberá incluir el código de procedimiento, insertándolo en las primeras 20 posiciones del tag Finalidad, conforme al criterio que exponemos a continuación (el Id_expediente puede ir vacío).

Finalidad: **CodProcedimiento#::#Id_Expediente#::#Texto Finalidad**

4.4 Definición de Errores de negocio y SoapFaults

Es imprescindible contar con un tratamiento de errores definido y detallado para aportar la información necesaria por una parte al usuario o aplicación final consumidora, pero también para el entorno de interoperabilidad con el fin de que lo medie o derive al cauce adecuado.

Por ello, y siguiendo el mismo funcionamiento establecido en la Plataforma de Intermediación de Datos, se devolverá un mensaje SOAP Fault cuando el error detectado pertenezca a alguno de los siguientes tipos:

- Error de conexión a la BD.
- Error de conexión a los sistemas externos (SAFE, Servidores Externos, etc).
- Error en la validación de esquemas (o petición recibida sin firma).
- Error por Validación de la Firma digital
    - No estar firmada la petición
    - Certificado caducado, revocado o no válido
- Error del Sistema Interno en el tratamiento de la petición.

Los mensajes SoapFault no se firmarán.

4.4.1 Especificación Errores de negocio

En el resto de casos, no contemplados en la lista anterior, se entenderá que la petición se ha podido tramitar y se devolverá un mensaje de Respuesta especificando en las etiquetas correspondientes el código y el texto del error o estado correspondiente (una vez mapeado) al considerarse una respuesta contemplada por el negocio.

Esos códigos y texto deberán estar recogidos en el contrato de integración del servicio.

Un ejemplo de estos tipos de errores controlados por el servicio final y que deben ser tratados atendiendo a lo especificado en el contrato de servicio pueden ser a modo de ejemplo algunos como los que se expone en la siguiente tabla, en la que se recogen errores de negocio típicos de diferentes servicios estatales. El siguiente listado de códigos y descripciones de errores se devuelve en el nodo DatosEspecificos del esquema de respuesta, concretamente, en los campos CodigoEstado y LiteralError:

Código de respuesta	Descripción de la respuesta obtenida

0	El contribuyente se encuentra al corriente en sus obligaciones tributarias
1	El contribuyente no se encuentra al corriente en sus obligaciones tributarias.
2	Se ha encontrado más de un registro para los datos del titular indicados
3	Para la persona consultada no hay información en la Comunidad Autónoma.

Tabla 2.-Errores de negocio devueltos por servicio de estar al corriente de las obligaciones tributarias GVA

Para las peticiones que devuelven respuestas con errores de negocio, en el nodo “Estado” del nodo “Retorno" en “DatosEspecificos” del mensaje de Respuesta se devolverá información que indica que la petición es errónea.

4.4.2 Especificación Soap Fault

Según el esquema de SOAP, la forma del mensaje de la parte SOAP Fault es la siguiente:

```
<xs:element name="Fault" type="tns:Fault" />
<xs:complexType name="Fault" final="extension">
<xs:annotation>
        <xs:documentation>Fault reporting structure</xs:documentation>
</xs:annotation>
    <xs:sequence>
          <xs:element name="faultcode" type="xs:QName" />
          <xs:element name="faultstring" type="xs:string" />
          <xs:element name="faultactor" type="xs:anyURI" minOccurs="0" />
          <xs:element name="detail" type="tns:detail" minOccurs="0" />
    </xs:sequence>
</xs:complexType>
<xs:complexType name="detail">
    <xs:sequence>
        <xs:any namespace="##any" minOccurs="0" maxOccurs="unbounded" processContents="lax" />
    </xs:sequence>
    <xs:anyAttribute namespace="##any" processContents="lax" />
</xs:complexType>
```

Según acuerdo y acercándonos a los estándares los valores que tomarán cada uno de estos elementos será el siguiente:

4.4.2.1 faultcode

Se debería dejar abierta la especificación de este elemento, siempre y cuando sea un Xml Qualified Name correcto, según especificación de la W3C. Esto permitirá usar los faultcode que se rellenan automáticamente por los motores SOAP. Opcionalmente, si se desea se podría indicar un faultcode del tipo:

Faultcode = **Sender | Client | Receiver | Server [.Subcode]***

SOAP 1.1	SOAP 1.2	Descripcion
Client	Sender	El *SOAPMessage* no se formó correctamente o no contiene la información necesaria para tener éxito.
Server	Receiver	El *SOAPMessage* no pudo ser procesado debido a un error de procesamiento.

Donde Sender o Client, se utilizarían para indicar que el problema proviene del mensaje enviado por el requirente, y Receiver o Server, para indicar que el problema ha surgido por los procesos del receptor del mensaje.

Si el mensaje de error se localizase en la PAI este campo vendría informado como ”**soap-env:PAI**”. En cuanto a los proveedores de servicio, se recomienda, la información de este campo de forma que sea detectable el elemento que origina el error.

4.4.2.2 faultstring

Puede tomar cualquier valor de tipo string. El proveedor debe tratar este campo para devolver errores legibles. Nunca se debe devolver información sobre las IP’s, URL u otros elementos sensibles para la seguridad del servicio.

Como sugerencia podría tomar el valor:

faultstring = [**Cod_Error**] **Literal Error**

Donde el Cod_Error serían 4 dígitos.

Por ejemplo: “[0301] Organismo no autorizado”

4.4.2.3 detail

Se recomienda que este elemento contenga una estructura/nodo Atributos, en la que se indicará toda la información necesaria para el error, con el fin de mantener una coherencia sintáctica con los

errores proporcionados por los servicios expuestos en la Plataforma de Intermediación de Datos (PID).

Esquema de nodo Atributos:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>

<xs:schema xmlns="http://intermediacion.redsara.es/scsp/esquemas/V3/soapfaultatributos"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://intermediacion.redsara.es/scsp/esquemas/V3/soapfaultatributos"
elementFormDefault="qualified" attributeFormDefault="unqualified">
	<xs:element name="Atributos">
		<xs:complexType>
			<xs:all>
				<xs:element ref="IdPeticion"/>
				<xs:element ref="NumElementos"/>
				<xs:element ref="TimeStamp"/>
				<xs:element ref="Estado" minOccurs="0"/>
				<xs:element ref="CodigoCertificado"/>
			</xs:all>
		</xs:complexType>
	</xs:element>
	<xs:element name="CodigoCertificado">
		<xs:simpleType>
			<xs:restriction base="xs:string">
				<xs:maxLength value="64"/>
			</xs:restriction>
		</xs:simpleType>
	</xs:element>
	<xs:element name="CodigoEstado">
		<xs:simpleType>
			<xs:restriction base="xs:string">
				<xs:maxLength value="4"/>
			</xs:restriction>
		</xs:simpleType>
	</xs:element>
	<xs:element name="CodigoEstadoSecundario">
		<xs:simpleType>
			<xs:restriction base="xs:string">
				<xs:maxLength value="16"/>
			</xs:restriction>
		</xs:simpleType>
	</xs:element>
	<xs:element name="IdPeticion">
		<xs:simpleType>
			<xs:restriction base="xs:string">
				<xs:maxLength value="26"/>
			</xs:restriction>
		</xs:simpleType>
```

```
</xs:element>
<xs:element name="LiteralError">
        <xs:simpleType>
                <xs:restriction base="xs:string">
                        <xs:maxLength value="255"/>
                </xs:restriction>
        </xs:simpleType>
</xs:element>
<xs:element name="LiteralErrorSec">
        <xs:simpleType>
                <xs:restriction base="xs:string">
                        <xs:maxLength value="255"/>
                </xs:restriction>
        </xs:simpleType>
</xs:element>
<xs:element name="NumElementos">
        <xs:simpleType>
                <xs:restriction base="xs:int">
                        <xs:totalDigits value="7"/>
                        <xs:minInclusive value="0"/>
                </xs:restriction>
        </xs:simpleType>
</xs:element>
<xs:element name="TiempoEstimadoRespuesta">
        <xs:simpleType>
                <xs:restriction base="xs:int">
                        <xs:totalDigits value="4"/>
                        <xs:minInclusive value="0"/>
                </xs:restriction>
        </xs:simpleType>
</xs:element>
<xs:element name="TimeStamp">
        <xs:simpleType>
                <xs:restriction base="xs:string">
                        <xs:maxLength value="29"/>
                </xs:restriction>
        </xs:simpleType>
</xs:element>
<xs:element name="Estado">
        <xs:complexType>
                <xs:all>
                        <xs:element ref="CodigoEstado" minOccurs="0"/>
                        <xs:element ref="CodigoEstadoSecundario" minOccurs="0"/>
                        <xs:element ref="LiteralError" minOccurs="0"/>
                        <xs:element ref="LiteralErrorSec" minOccurs="0"/>
                        <xs:element ref="TiempoEstimadoRespuesta" minOccurs="0"/>
                </xs:all>
        </xs:complexType>
```

```
        </xs:element>
</xs:schema>
```

4.4.2.4 Ejemplo de SoapFault

Un ejemplo de estos tipos de errores lo podemos ver en la siguiente tabla que es extraída de un contrato de integración.

Código de estado SCSP	Descripción del error	Cuando se devuelve este error
0101	Imposible ejecutar el servicio	El servicio esta caído y no se pudo cursar la petición del cliente.
0204	La petición no existe en el sistema	Este error se produce cuando se solicita la respuesta asíncrona de un idPeticion para el cual no se ha registrado ninguna respuesta correcta
0230	El timestamp de la petición debe ser válido y de hoy o de ayer	Se recibe una petición con un timestamp con formato incorrecto o que no es ni de ayer ni de hoy.
0301	Organismo no autorizado	El certificado o procedimiento utilizados son incorrectos y no se pudo autorizar el consumo al servicio
0302	Certificado caducado	El certificado esta caducado
0303	Certificado revocado	El certificado esta revocado
0305	La firma de la petición no es válida	Se ha recibido una petición en la que la firma no es válida
0307	La petición no tiene cabecera de seguridad válida o La petición no tiene cabecera de seguridad válida	La petición llega sin cabecera WS-Security válida
0309	Error general al verificar el certificado	Se produce un error al validar el certificado
0310	No se ha podido verificar la CA del certificado	Se produce un error al verificar la Autoridad de Certificación del certificado
0401	La estructura del XML introducido no corresponde con el esquema	La petición enviada no cumple la estructura del esquema.

0502	Error de sistema e identificación del sistema	Se dará cuando se produzca algún error interno que impida el funcionamiento correcto (error de configuración, inconsistencia de datos, etc.)
0807	Falta la cabecera Id_trazabilidad	la petición recibida no contenía el tag Id_trazabilidad en la cabecera SOAP Este error solo se producirá en los servicios instrumentales
0808	El usuario en el Id_Trazabilidad no corresponde con el usuario en la cabecera de seguridad o El número de serie del Id_Trazabilidad no corresponde con el certificado de firma	El usuario o núm. De serie que se han incluido en el tag Id_trazabilidad no se corresponden con los que vienen incluidos en la cabecera WS-Security Este error solo se producirá en los servicios instrumentales de firma/username (no en los instrumentales de innovación)

Tabla 3.- Ejemplo de Errores devueltos como SOAPFault

Con lo expuesto anteriormente un ejemplo de SoapFault de un servicio de verificación podría tener el siguiente aspecto:

```
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
    <soapenv:Body>
    <soapenv:Fault>
        <faultcode>env:Server</faultcode>
        <faultstring>[0403] Error al obtener el contenido XML del mensaje SOAP </faultstring>
        <faultactor>XSPRUW30 </faultactor>
        <detail>
        <Atributos xmlns="http://intermediacion.redsara.es/scsp/esquemas/V3/soapfaultatributos">
          <IdPeticion>CODCERT00001</IdPeticion>
          <NumElementos>1</NumElementos>
          <TimeStamp>2010-09-20T13:22:35.993+0200</TimeStamp>
          <Estado>
          <CodigoEstado>0403</CodigoEstado>
          <LiteralError>Imposible obtener el contenido XML del mensaje SOAP.</LiteralError>
          <CodigoEstadoSecundario>04031</CodigoEstadoSecundario>
          <LiteralErrorSec>Imposible obtener el campo atributos del SOAP</LiteralErrorSec>
          <TiempoEstimadoRespuesta>0</TiempoEstimadoRespuesta>
          </Estado>
          <CodigoCertificado>AEAT103I</CodigoCertificado>
```

```
        </Atributos>
        </detail>
    </soapenv:Fault>
    </soapenv:Body>
</soapenv:Envelope>
```

- ***faultcode***: Código que determina el tipo de error producido.
- ***faultstring***: Descripción del tipo de error producido.
- ***faultactor***: Código que determina donde se origina el error.
- ***IdPeticion***: Identificador con el que se realiza la petición.
- ***NumElementos***: Número de elementos que se incluyen en la petición.
- ***TimeStamp***: Fecha y hora del momento en que se realiza la petición.
- ***CodigoEstado***: Código del error producido.
- ***LiteralError***: Descripción del error producido.
- ***CodigoEstadoSecundario***: Código que especifica con más detalle el error producido.
- ***LiteralErrorSec***: Descripción más detallada del error producido.
- ***TiempoEstimadoRespuesta***: Tiempo en el que se estima que se obtenga la respuesta.
- ***CodigoCertificado***: Código del certificado del servicio que se pretende consumir.

Ejemplo de SoapFault de un servicio instrumental podría tener el siguiente aspecto:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <soapenv:Fault>
      <faultcode xmlns:soap-env="http://schemas.xmlsoap.org/soap/envelope/">soap-env:PAI</faultcode>
      <faultstring>[0305]La firma de la petición no es válida</faultstring>
      <detail>
        <Atributos xmlns="http://intermediacion.redsara.es/scsp/esquemas/V3/soapfaultatributos">
          <IdPeticion/>
          <NumElementos>0</NumElementos>
          <TimeStamp>2016-10-03T13:01:24.582+02:00</TimeStamp>
          <Estado>
            <CodigoEstado>0305</CodigoEstado>
            <CodigoEstadoSecundario/>
            <LiteralError>La firma de la petición no es válida</LiteralError>
```

```
                <LiteralErrorSec>OSB-382500La acción de llamada de servicio de OSB ha recibido una respuesta de fallo de SOAPsoapenv:ServerOSB-386200: Error de seguridad de servicio web generalAutenticacionrequest-7f000101.3956d90b.0.1511ee842f3.N72d1Validacion_Firmarequest-pipeline</LiteralErrorSec>
                <TiempoEstimadoRespuesta>0</TiempoEstimadoRespuesta>
              </Estado>
              <CodigoCertificado/>
            </Atributos>
          </detail>
        </soapenv:Fault>
      </soapenv:Body>
    </soapenv:Envelope>
```

- ***faultcode***: Código que determina el tipo de error producido.
- ***faultstring***: Descripción del tipo de error producido.
- ***IdPeticion***: Identificador con el que se realiza la petición. En esta modalidad no se utiliza.
- ***NumElementos***: Número de elementos que se incluyen en la petición.
- ***TimeStamp***: Fecha y hora del momento en que se realiza la petición.
- ***CodigoEstado***: Código del error producido.
- ***CodigoEstadoSecundario***: Código que especifica con más detalle el error producido. En esta modalidad no se utiliza.
- ***LiteralError***: Descripción del error producido.
- ***LiteralErrorSec***: Descripción más detallada del error producido.
- ***TiempoEstimadoRespuesta***: Tiempo en el que se estima que se obtenga la respuesta.
- ***CodigoCertificado***: Código del certificado del servicio que se pretende consumir. En esta modalidad no se utiliza.

4.4.2.5 Gestión de errores en aplicaciones clientes

Tal y como se indica en los puntos anteriores, los SoapFault poseen información del error que se ha producido durante una invocación. Por ello, para llevar a cabo el diagnóstico y resolución de los errores recibidos, es importante que la aplicación cliente disponga de toda información posible, es decir, del mensaje SoapFault de respuesta completo o sus atributos más descriptivos como son; ***CodigoEstado, CodigoEstadoSecundario, LiteralError, LiteralErrorSec.***

Al poseer esta información, tanto los desarrolladores como los usuarios de la aplicación podrán saber con más exactitud el origen del error y realizar las acciones pertinentes, ya sea mediante la solución indicada en la tabla de Soapfaults o reportando la incidencia al ente correspondiente.

5 Apoyo a los desarrolladores

La PAI pretende ofrecer herramientas e información al desarrollador, mediante la cual se pueda facilitar, reducir el tiempo y dedicación destinado a la integración con la Plataforma.

5.1 Generación de servicio web

Definimos un manual de creación de servicios web (Manual de usuario de generación de servicios web) según las directrices descritas en el documento que nos ocupa. Utilizando Apache CXF en su versión 3 y teniendo en cuenta ciertos aspectos de configuración deseada para el correcto cumplimiento de las buenas prácticas.

NOTA: Si la aplicación sigue el estándar de la oficina java, es de interés consultar la documentación expuesta en su espacio de confluence, en concreto la relativa a la capa de servicios (190-analisis-capa-servicios-soap).

5.2 Clientes de servicios

Existen diversas tecnologías en la actualidad para consumir los servicios web. No es capítulo de este documento servir de guía tutorial de desarrollo, sino dar unas pautas de ejemplo. En el 'Manual de usuario para el consumo de servicios instrumentales y de verificación' se muestra un cliente de servicio realizado con Apache CXF 3.3.4. Concretando las recomendaciones que desde la PAI se han realizado, se optó por clientes basado en CXF por su comodidad en la confección de los Stubs y clases asociadas al consumo de servicios web.

NOTA: Si la aplicación cliente sigue el estándar de la oficina java, es de interés consultar la documentación expuesta en su espacio de confluence, en concreto la relativa a la capa de servicios (190-analisis-capa-servicios-soap).