CAPÍTULO 9: SAML

Damos una breve descripción de los conceptos que son la base de SAML y de trozos de componentes definidos en el estándar y de aquellas reglas para su correcto uso. Para aportar más claridad, explicaremos de forma breve cada componente y la relación entre ellos para poder tener una idea general. Así hablaremos de las aserciones, de los protocolos, de los bindings y de los perfiles. Además entraremos a comentar aquellos componentes secundarios como los contextos de autentificación y metadatos.

9.1.- Conceptos Básicos

En primer lugar, y para aportar más claridad, detallamos unos conceptos que aparecerán a lo largo de toda la especificación:

- Aserción:

Una aserción es un paquete de información que suministra cero o más declaraciones realizadas por una autoridad SAML.

- Sujeto solicitado:

Es la entidad sobre la que se solicitan una o más aserciones.

- Proveedor de identidad (IP):

Es el sistema, o dominio administrativo, que afirma una determinada información acerca de un sujeto. Es una autoridad SAML que crea, mantiene y gestiona la información de identidad de los sujetos y provee de información de autentificación a otras entidades. Son las entidades que crean las aserciones SAML. Nos referimos a dichas entidades como partes que afirman (***Asserting parties***).

Por ejemplo, el proveedor de identidad afirma que este sujeto ha sido autentificado y se le han asociado atributos. Dicho de otra manera, este usuario es Alberto, tiene una dirección de mail de valor *Alberto@universidad.com* y fue autentificado en este sistema usando un mecanismo de contraseña.

- Proveedor de servicio (SP):

Es el sistema, o dominio administrativo, que proporcionan servicios a sujetos u otras entidades. Es una entidad que decide emprender acciones basadas en la información que le proporciona otra entidad. Esto es, confía en la información que le suministra un proveedor de identidad, es decir, aquellas entidades que confían en las aserciones que le llegan. SAML define mecanismos que permiten a estos proveedores confiar en las aserciones que se le proporcionan, siendo las políticas locales las que permitan o no al sujeto, el acceso al recurso que se solicita. Son autoridades SAML y también nos referimos a ellas como partes que confían (***Relying parties***).

Al hilo del ejemplo anterior, aunque el proveedor de servicio confíe en que soy Alberto, esto no significa que tenga libertad de acceso a todos los recursos.

- Presentador:

Es la entidad que presenta la petición al proveedor de identidad y bien se autentifica durante la transmisión del mensaje, bien confía en un contexto de seguridad existente para establecer su identidad. Si no es el solicitante, el presentador actúa como intermediario entre el solicitante y el proveedor de identidad que responde.

SAML consiste en un número de componentes constituidos en bloque que, cuando los unimos, soportan diferentes casos de uso. Principalmente los componentes permiten la transferencia de:

- **Identidad**.
- **Autentificación**.
- **Información de la autorización**.

Todo esto se hace con el fin de que sean intercambiadas entre organizaciones autónomas.

La especificación SAML define la estructura y el contenido de las **aserciones** – que llevan declaraciones sobre un principal según lo afirmado por una Asserting Party. Éstas son definidas mediante esquemas de XML. Las aserciones se solicitan al proveedor de servicio. Cómo y qué aserción se solicita está definido por los **protocolos de SAML**, que tienen su propio esquema de XML. Los protocolos de nivel inferior de la comunicación o de mensajes (tales como HTTP o SOAP) que pueden ser transportados por protocolos SAML son definidos por **Bindings**.

Los protocolos y bindings de SAML, junto con la estructura de aserciones, se pueden combinar juntos para crear un perfil (**Profile**). Hay también los perfiles de atributos, que definen qué la información de la identidad y de atributos lleva dentro una aserción.

Otros dos componentes de SAML pueden ser utilizados en la construcción de un sistema:

- Metadata: indica cómo se define y se transmite la información compartida de configuración entre dos entidades al comunicarse. Por ejemplo, pueden ser definidos el soporte de una entidad a un binding dado de SAML, la información de identificador, y la información de PKI. Metadata está definido por un esquema de XML y su localización se define usando registros DNS.

- Contexto de Autentificación: en determinadas situaciones, el proveedor de servicio puede desear tener información adicional para determinar la autenticidad y la confidencialidad que tiene la información dentro de una aserción. El contexto de autentificación permite el aumento de aserciones con la información adicional que pertenece a la autentificación del director en el proveedor de identidad. Por ejemplo, los detalles de la autentificación de varios factores pueden ser incluidos.

Estos componentes se relacionan de la manera mostrada en la figura de abajo.

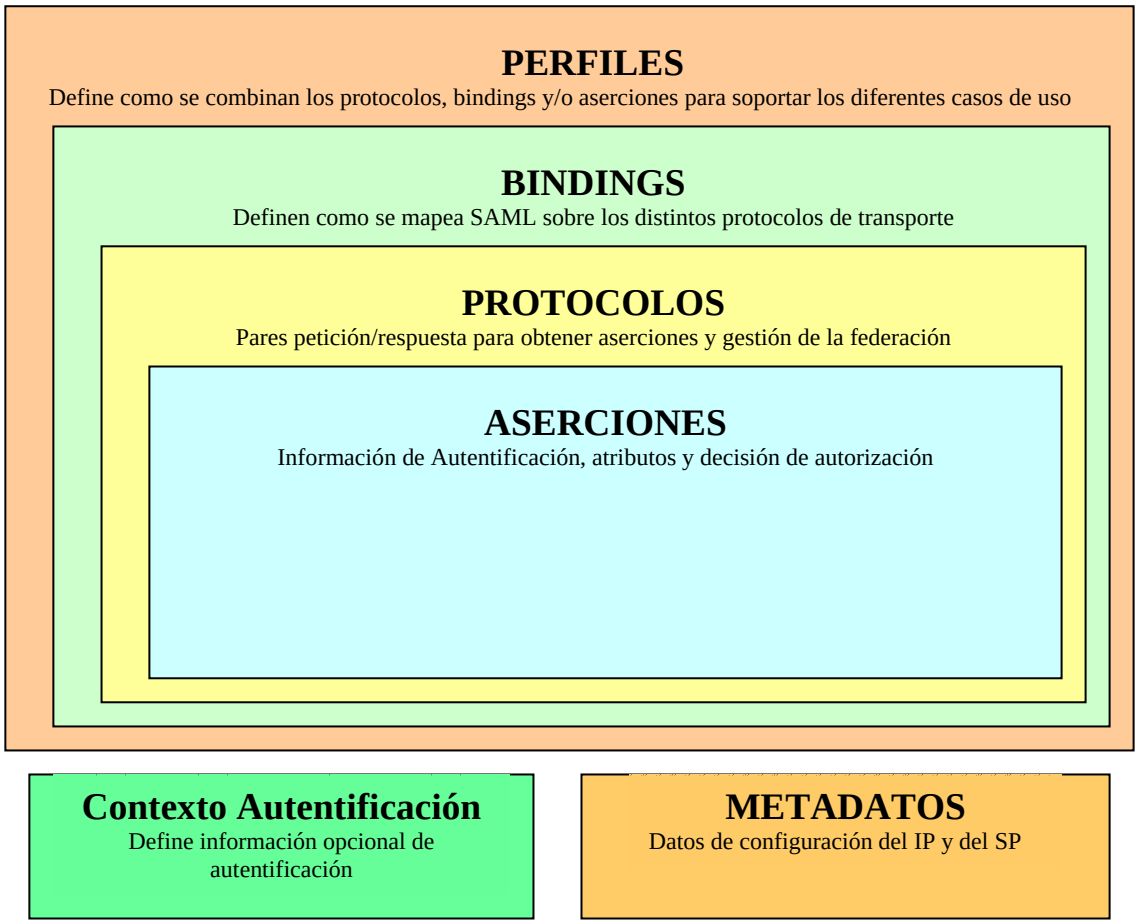

Figura 9. 1: Relación entre los componentes de SAML

Con este dibujo ya tenemos una visión de los distintos componentes y de cómo se relacionan. Ahora pasamos a detallar cada uno de ellos en las secciones posteriores.

9.2.- Aserciones SAML

Como hemos dicho, una aserción es un paquete de información que suministra cero o más declaraciones realizadas por una **autoridad SAML.** Para tener una perspectiva más amplia del apartado hay que ver las partes que intervienen. De este modo podemos decir que un proveedor de identidad (IP) proporciona una aserción a un proveedor de servicio (SP) que hace referencia a un determinado sujeto. Así, mediante la aserción, el proveedor de servicio podrá decidir que privilegios le asocia al sujeto anterior.

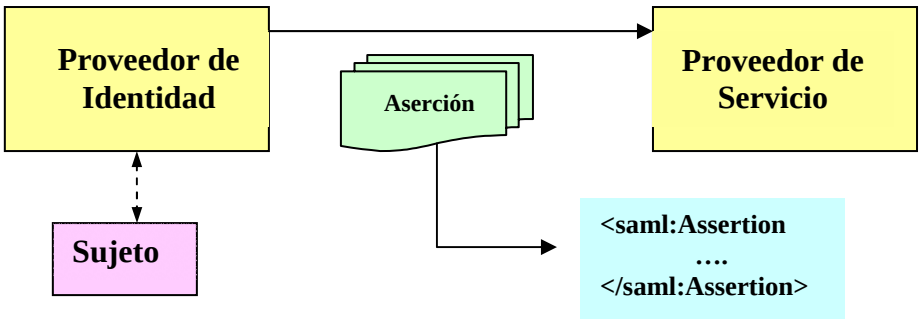

Figura 9. 2: Relación entre las distintas entidades

A veces nos referimos a las autoridades SAML como las **partes que afirman** en discusiones de generación e intercambio de aserciones, y las entidades de sistema que utilizan las aserciones recibidas se conocen como **partes que confían** (observamos que estos términos son diferentes desde el solicitante y el respondedor, que se reservan para las discusiones de intercambio de mensajes de protocolo SAML). Típicamente hay un número de proveedores de servicio que pueden hacer uso de aserciones sobre un sujeto para controlar el acceso y proporcionar servicios personalizados, y por consiguiente se convierten en las partes que confían de una parte que afirma llamada proveedor de identidad.

Esta especificación SAML define tres clases diferentes de declaraciones de la aserción que puedan ser creadas por una autoridad SAML. Todas las declaraciones definidas en SAML se asocian a un sujeto.

Las tres clases de declaraciones definidas en esta especificación son:
- **Autentificación:** El sujeto de la aserción fue autentificado por medios particulares en un tiempo concreto.
- **Atributo:** El sujeto de la aserción se asocia a los atributos proporcionados.
- **Decisión de Autorización:** Una petición para permitir al sujeto de la aserción acceder al recurso solicitado ha sido concedida o denegada.

La estructura externa de una aserción es genérica, proporcionando la información que es común a todas las declaraciones dentro de ella. Dentro de una aserción, una serie de elementos internos describen la autentificación, los atributos, la decisión de autorización, o las declaraciones definidas por el usuario que contienen los detalles.

Las extensiones son permitidas por el esquema de la aserción SAML, permitiendo extensiones definidas por el usuario en aserciones y declaraciones, posibilitando la definición de nuevas clases de aserciones y declaraciones. De esta manera, SAML permite a una de las partes afirmar características y atributos de una entidad. Como ya se ha explicado, se puede extender SAML para particularizar cada tipo de aserción y declaración de los diferentes usuarios. Hay que recordar que las aserciones están codificadas en esquemas XML y pueden ser firmadas digitalmente.

Ya hemos comentado los posibles tipos de declaraciones de aserciones. A continuación pasamos a detallarlas. Para ello pondremos un esquema de su posible contenido y un ejemplo para ver cómo puede ser su definición

Declaraciones de autenticación

El sujeto especificado fue autentificado con un significado particular y en un tiempo concreto. Son emitidas por la parte que autentificó con éxito al usuario. Definen quién emitió la aserción, el sujeto autenticado, el periodo de validez, además de otras informaciones relacionadas con la autenticación.

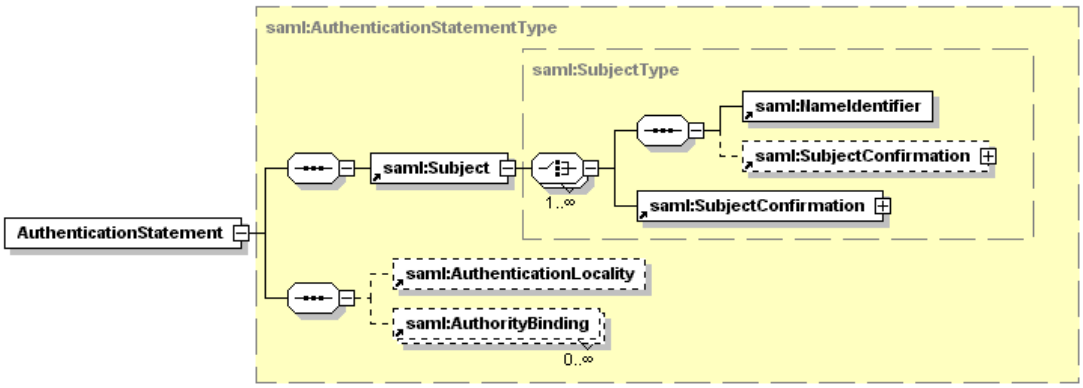

Figura 9. 3: Esquema de declaración de autentificación

Como ejemplo de esquema XML podemos dar el siguiente. En el podemos ver como el sujeto de nombre “joeuser” se autentificó en un tiempo concreto y mediante un mecanismo de “password”. Todo esto viene reflejado en el esquema de abajo:

```
<saml:Assertion ...>
  <saml:AuthenticationStatement
    AuthenticationMethod="password"
    AuthenticationInstant="2001-12-03T10:02:00Z">
    <saml:Subject>
      <saml:NameIdentifier
        SecurityDomain="smithco.com"
        Name="joeuser" />
      <saml:ConfirmationMethod>
        http://...core-25/sender-vouches
      </saml:ConfirmationMethod>
    </saml:Subject>
  </saml:AuthenticationStatement>
</saml:Assertion>
```

Figura 9. 4: Ejemplo de esquema XML para la declaración de autentificación

Declaraciones de atributos

Se utiliza para relacionar ciertas propiedades o atributos con el sujeto indicado en la aserción. De este modo, una autoridad SAML puede emitir aserciones en las que indique que cierto sujeto tiene ciertos atributos. De esta forma, toda entidad que tenga establecida una relación de confianza con la autoridad SAML, tiene la certeza de que cierto sujeto tiene unas determinadas características. Normalmente, estas características serán una de las variables evaluadas por otra entidad a la hora de tomar una decisión de autorización.

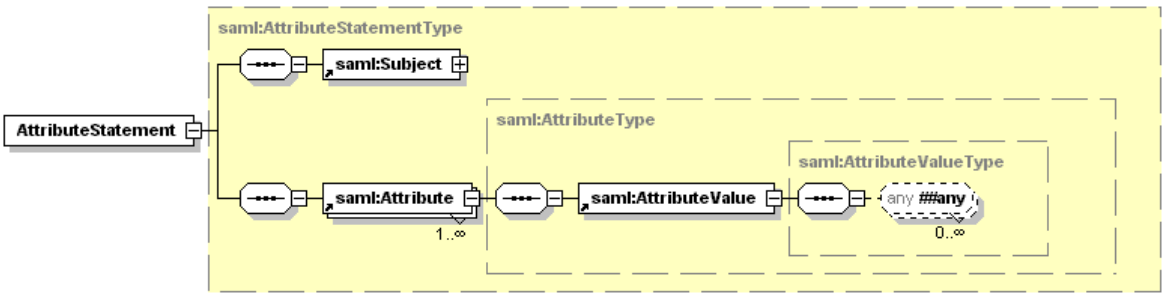

Figura 9. 5: Esquema de declaración de atributos

Como hemos dicho, contiene detalles específicos respecto al usuario (por ejemplo que posee un estatus privilegiado). En ella una entidad afirma que un determinado sujeto S está asociado a unos atributos A, B, C,… de valor a, b, c,…Todo esto se puede ver en el ejemplo de esquema XML de declaración de atributos:

```
<saml:Assertion ...>
  <saml:AttributeStatement>
    <saml:Subject>...</saml:Subject>
    <saml:Attribute
      AttributeName="PaidStatus"
      AttributeNamespace="http://smithco.com">
      <saml:AttributeValue>
        PaidUp
      </saml:AttributeValue>
    </saml:Attribute>
 <saml:Attribute
      AttributeName="CreditLimit"
      AttributeNamespace="http://smithco.com">
      <saml:AttributeValue>
        <my:amount currency="USD">500.00
        </my:amount>
      </saml:AttributeValue>
    </saml:Attribute>
  </saml:AttributeStatement>
</saml:Assertion>
```

Figura 9. 6: Ejemplo de esquema XML para la declaración de atributos

Declaraciones de decisión de autorización

Este tipo de declaración contiene la información generada por una autoridad SAML que confirma que una petición hecha por el sujeto de la declaración para acceder a cierto recurso ha sido aceptada o denegada (decisión de autorización) basándose en algunas evidencias especificadas opcionalmente.

El recurso es identificado mediante una URI. Con el fin de que todas las autoridades y servicios SAML puedan tomar decisiones coherentes deberán ponerse de acuerdo en la forma de interpretar las URIs. La especificación SAML señala que la interpretación de las URIs debe basarse en las reglas definidas en el IETF RFC 2396.

Dicho de otra manera, identifica lo que le está permitido hacer al usuario (por ejemplo si el usuario puede comprar un determinado producto). Se entiende por usuario

tanto a la persona como el programa que necesite hacer la citada acción. Es muy útil para transacciones distribuidas y para autorización de servicios.

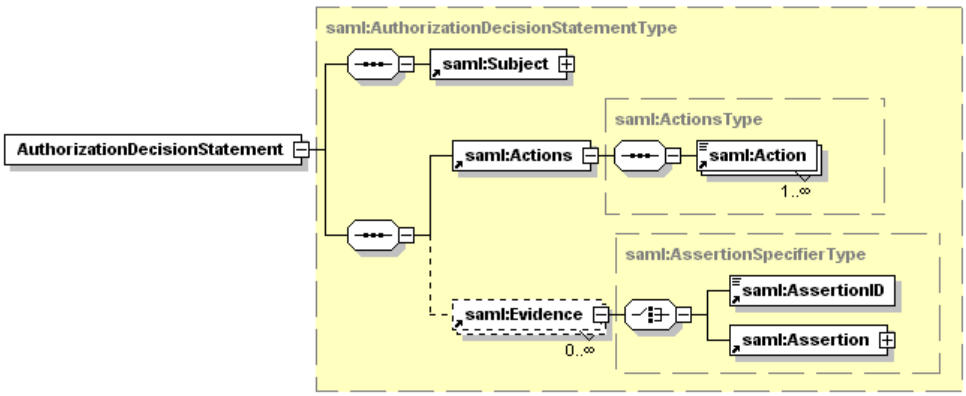

Figura 9. 7: Esquema XML para la declaración de decisión de autorización

Damos el ejemplo de un esquema XML de declaración de decisión de autorización. En el se pueden observar los distintos elementos dentro del esquema:

```
<saml:Assertion ...>
  <saml:AuthorizationStatement
    Decision="Permit"
    Resource="http://jonesco.com/rpt_12345.htm">
    <saml:Subject>...</saml:Subject>
    <saml:Actions
      ActionNamespace="http://...core-25/rwedc">
      <saml:Action>Read</saml:Action>
    </saml:Actions>
  </saml:AuthorizationStatement>
</saml:Assertion>
```

Figura 9. 8: Ejemplo de esquema XML para la declaración de decisión de autorización

9.2.1.- Elemento <Assertion>

En esta sección detallamos el elemento <Assertion>, explicando los elementos y atributos que contiene. El elemento <Assertion> es un tipo complejo de **AssertionType**. Este tipo especifica la información básica que es común a todas las aserciones. Para tener una visión general del elemento damos el esquema y a continuación explicamos los atributos y elementos que lo forman.

```
<element name="Assertion" type="saml:AssertionType"/>
<complexType name="AssertionType">
        <sequence>
                <element ref="saml:Issuer"/>
                <element ref="ds:Signature" minOccurs="0"/>
                <element ref="saml:Subject" minOccurs="0"/>
                <element ref="saml:Conditions" minOccurs="0"/>
                <element ref="saml:Advice" minOccurs="0"/>
                <choice minOccurs="0" maxOccurs="unbounded">
                        <element ref="saml:Statement"/>
                        <element ref="saml:AuthnStatement"/>
                        <element ref="saml:AuthzDecisionStatement"/>
                        <element ref="saml:AttributeStatement"/>
                </choice>
        </sequence>
        <attribute name="Version" type="string" use="required"/>
        <attribute name="ID" type="ID" use="required"/>
        <attribute name="IssueInstant" type="dateTime" use="required"/>
</complexType>
```

Figura 9. 9: Definición del elemento <Assertion> y su tipo complejo **AssertionType**

- Version (Obligatorio)

La versión de esta aserción. El identificador de la versión definido en esta especificación es “2.0”.

- ID (Obligatorio)

Es el identificador de la aserción. Es del tipo **xs:ID**, y debe seguir los requerimientos para unicidad de identificadores.

- IssueInstant (Obligatorio)

Es el instante de tiempo de emisión en formato UTC.

- <Issuer> (Obligatorio)

Es la autoridad SAML que esta haciendo las afirmaciones en la aserción. El emisor no debe ser ambiguo con respecto a las partes que confían relacionadas.

Esta especificación no define relaciones particulares entre la entidad representada por este elemento y la que firma la aserción (si la hubiera). Cualquier requerimiento impuesto por una parte que confía que consume la aserción o de perfiles determinados, son específicos de la aplicación.

- <ds:Signature> (Opcional)

Es una firma XML que protege la integridad y autentifica al emisor de la aserción.

- <Subject> (Opcional)

Es el sujeto de las declaraciones en la aserción.

- <Conditions> (Opcional)

Son condiciones que debemos evaluar cuando juzguemos la validez y/o cuando usemos la aserción.

- <Advice> (Opcional)

Información adicional relacionada con la aserción que colabora con el proceso en determinadas situaciones pero que deben ser ignoradas en aplicaciones en las que no se entienda o no se desee usarla.

Puede haber cero o más de los siguientes elementos de declaración que comentamos en una sección anterior:

- <Statement>

Una declaración de un tipo definido en una extensión de esquema. Se debe usar un atributo xsi:type para indicar el tipo actual de declaración.

- <AuthnStatement>

Una declaración de autentificación.

- <AuthzDecisionStatement>

Una declaración de decisión de autorización.

- <AttributeStatement>

Una declaración de atributo.

A modo de resumen, y para añadir más claridad, incluimos una tabla donde pondremos de manera esquemática, los distintos atributos y su descripción.

Atributo	*Descripción*	*Uso*
Version	La versión de la aserción	Obligatorio
ID	Identificador de la aserción	Obligatorio
IssueInstant	Instante de tiempo de emisión	Obligatorio

Tabla 9. 1: Atributos de <Assertion>

De igual manera, ponemos los elementos que intervienen en <Assertion>.

Elemento	*Descripción*	*Uso*
<Issuer>	La autoridad que afirma en la aserción	Obligatorio
<ds:Signature>	Detalla la firma XML	Opcional
<Subject>	Sujeto al que hace referencia la aserción.	Opcional
<Conditions>	Condiciones para validez de aserción	Opcional
<Advice>	Información adicional de la aserción	Opcional
<Statement>	Declaración de un tipo de una extensión	Opcional
<AuthnStatement>	Declaración de autentificación	Opcional
<AuthzDecisionStatement>	Declaración decisión de autorización	Opcional
<AttributeStatement>	Declaración de atributo	Opcional

Tabla 9. 2: Elementos de **<Assertion>**

Restricciones al uso del elemento <Assertion>

En este apartado damos unas reglas generales para el correcto uso del elemento <Assertion>. Daremos restricciones de aquellos elementos que tienen un

funcionamiento especial, es decir, aquellos que hay que tener en cuenta para un correcto uso.

Una aserción sin declaraciones debe contener un elemento **<Subject>**. Una aserción identifica a un principal de modo que pueda ser referenciado o confirmado usando métodos SAML, pero no afirmando posteriores informaciones asociadas con el principal. Si está presente <Subject>, identifica al sujeto de todas las declaraciones de la aserción. Si por el contrario se omite, entonces dichas declaraciones se aplican al sujetos o sujetos identificados de manera específica del perfil o de la aplicación. SAML no define por sí mismo tales declaraciones, y una aserción sin sujeto no tiene significado en esta especificación.

Dependiendo de los requisitos de los perfiles o protocolos particulares, el emisor de una aserción SAML frecuentemente puede necesitar ser autentificado y puede ser requerida también protección de integridad. La autentificación y la integridad de mensaje pueden ser proporcionadas por los perfiles de protocolo que se usen durante la entrega de la aserción.

La aserción SAML puede ser firmada, lo cual proporciona tanto autentificación del emisor como protección de integridad. Si se usa, entonces se debe presentar el elemento **<ds:Signature>**. Y la parte que confía debe verificar que la firma es válida. Si no es válida, entonces la parte que confía no deberá fiarse de los contenidos de la aserción. Si es válida, la parte que confía deberá evaluar la firma para determinar la identidad y la no apropiación del emisor y poder continuar con el procesado de la aserción de acuerdo con la especificación y como se estime oportuno (por ejemplo evaluando las condiciones, los consejos, etc.).

Advertimos que si se firma o no, la inclusión de múltiples declaraciones dentro de una aserción es semánticamente equivalente a establecer aserciones que contengan esas declaraciones individualmente (siendo el sujeto, las condiciones, etc. siempre los mismos).

9.2.2.- Identificadores de nombres

En la siguiente sección definimos las construcciones SAML que contienen los identificadores que describen a los sujetos y emisores de aserciones y mensajes de protocolo.

Hay muchas circunstancias en la que es útil para dos entidades de sistema comunicarse refiriéndose a una tercera parte, es decir, cuando se comunican dos entidades es importante poder hacer referencia a una tercera. De esta manera, es útil establecer medios por los que podamos asociar los identificadores con cada una de las partes. En algunos casos será necesario limitar el alcance dentro del cual se hace uso de los identificadores a un pequeño conjunto de entidades de sistema (para preservar la privacidad del sujeto, por ejemplo).

Identificadores similares pueden ser también usados para referirnos al emisor de un mensaje de protocolo SAML o aserción. Es posible que dos o más entidades de sistema usen el mismo valor de identificador cuando se refieran a diferentes identidades.

De esta manera, cada entidad puede tener diferentes significados para el mismo nombre. SAML proporciona calificadores de nombres (**namequalifiers**) para eliminar la ambigüedad de los identificadores de nombres, colocándolo efectivamente en un espacio de nombres federados que se relaciona con dicho calificador. SAML V2.0 permite que un identificador sea calificado en términos tanto de la parte que asiente como la que confía. Los identificadores de nombres pueden se encriptados con el fin de mejorar las características de conservación de la privacidad, particularmente en los casos donde el identificador pueda ser transmitido por medio de un intermediario.

Tipo Complejo NameIDType

El tipo complejo **NameIDType** se usa cuando un elemento sirve para representar una entidad por medio de un nombre dado por una cadena, siendo el tipo que subyace en los elementos <NameID> y <Issuer>. Damos su esquema a continuación.

```
<complexType name="NameIDType">
        <simpleContent>
                <extension base="string">
                        <attributeGroup ref="saml:IDNameQualifiers"/>
                        <attribute name="Format" type="anyURI" use="optional"/>
                        <attribute name="SPProvidedID" type="string" use="optional"/>
                </extension>
        </simpleContent>
</complexType>
```

Figura 9. 10: Definición del tipo complejo **NameIDType**

Además de la cadena que contiene el identificador, proporciona los siguientes atributos:

- NameQualifier (Opcional)

Es el dominio administrativo o de seguridad que califica el nombre. Este atributo proporciona un medio para federar nombres de distintos almacenes de usuarios sin colisión.

- SPNameQualifier (Opcional)

Más allá de calificar a un nombre con el de un proveedor de servicio o afiliación de proveedores, proporciona un medio adicional de federación de nombres a base de parte o partes que confían.

- Format (Opcional)

Es unas referencias URI que representa a la clasificación de información de identificadores basadas en cadenas.

- SPProvidedID (Opcional)

Es un identificador de nombre establecido por el proveedor de servicio o por la afiliación de proveedores para la entidad, si es diferente del primer identificador de nombre dado en el contenido del elemento. Este atributo proporciona un medio de

integración para los identificadores ya existentes en uso por los proveedores de servicio. Por ejemplo, un identificador existente puede ser unido a la entidad usando el protocolo de gestión de identificadores de nombres definido en SAML.

Normas adicionales para el contenido (o la omisión) de estos atributos pueden ser definidos por los elementos que hagan uso de este tipo, y por definiciones específicas de Format. Los atributos NameQualifier y SPNameQualifier deberán ser omitidos a menos que el elemento o formato defina explícitamente sus usos y semánticas.

A modo de resumen incluimos una tabla con los atributos definidos.

Atributo	*Descripción*	*Uso*
NameQualifier	Entidad que califica el nombre	Opcional
SPNameQualifier	Medio adicional de federación de nombres	Opcional
Format	Clasificación de los calificadores de nombres	Opcional
SPProviderID	Nombre dado por el SP para el IP	Opcional

Tabla 9. 3: Atributos del tipo **NameIDType**

Elemento <NameID>:

El elemento <NameID> es del tipo **NameIDType**. Se usa en varias construcciones de aserciones SAML como en los elementos <Subject> y <SubjectConfirmation> y también en varios mensajes de protocolo.

El siguiente fragmento de esquema lo define:

```
<element name="NameID" type="saml:NameIDType"/>
```

Figura 9. 11: Definición del elemento <**NameID**>

Elemento <Issuer>:

El elemento <Issuer>, con el tipo complejo **NameIDType**, proporciona información acerca del emisor de la aserción SAML o de los mensajes del protocolo. El elemento requiere el uso de una cadena que lleve el nombre del emisor, pero permite varios trozos de datos descriptivos.

Contradiciendo las reglas usuales de este tipo de elementos, si el valor de Format no se da, se elige el de urn:oasis:names:tc:SAML:2.0:nameid-format:entity.

```
<element name="Issuer" type="saml:NameIDType"/>
```

Figura 9. 12: Definición del elemento <**Issuer**>

Valores posibles del atributo Format

Los siguientes identificadores pueden ser usados en el atributo Format de los elementos <NameID>, <NameIDPolicy>, o <Issuer> para referirnos a formatos

comunes para el contenido de los elementos y las reglas asociadas de procesado, si las hubiese. A modo de resumen los listamos en una tabla.

Format	*URI*
No especificado	urn:oasis:names:tc:SAML:1.1:nameid-format:**unspecified**
Dirección Email	urn:oasis:names:tc:SAML:1.1:nameid-format:**emailAddress**
Nombre X.509	urn:oasis:names:tc:SAML:1.1:nameid-format:**X509SubjectName**
Dominio Windows	urn:oasis:names:tc:SAML:1.1:nameid-format:**WindowsDomainQualifiedName**
Principal Kerberos	urn:oasis:names:tc:SAML:2.0:**nameid-format:kerberos**
Entidad	urn:oasis:names:tc:SAML:2.0:nameid-format:**entity**
Id. Persistentes	urn:oasis:names:tc:SAML:2.0:nameid-format:**persistent**
Id. Transitorios	urn:oasis:names:tc:SAML:2.0:nameid-format:**transient**

Tabla 9. 4: Valores para el atributo **Format**

Algunos identificadores que fueron desaprobados en SAML V1.1 han sido borrados por SAML V2.0. Pasamos a detallar cada uno de los posibles identificadores para el atributo Format.

1.- No especificado
URI: urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified

La interpretación del contenido de este elemento se deja para implementaciones particulares.

2.- Dirección Email
URI: urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress

Indica que el contenido del elemento está en la forma de una dirección de email, específicamente “addr-spec.” según lo definido en la sección 3.4.1 de la RFC 2822 [RFC 2822] del IETF. Una addr-spec tiene la forma: local-part@dominio.

Observamos que addr-spec no tiene ninguna frase (tal como nombre común) antes de ella, no tiene ningún comentario (texto rodeado por paréntesis) después de ella, y no está rodeada por “<” y “>”.

3.- Nombre de Sujeto X.509
URI: urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName

Indica que el contenido del elemento está en la forma especificada para el contenido del elemento <ds:X509SubjectName> en la Recomendación de la firma XML [XMLSig].

Los implementadores deben observar que la especificación de la firma XML marca las reglas de codificación para los nombres de Sujetos X.509 que se diferencian de las reglas dadas en RFC 2253 [RFC 2253] del IETF.

4.- Nombre Cualificado Dominio de Windows
URI: urn:oasis:names:tc:SAML:1.1:nameid-format:WindowsDomainQualifiedName

Indica que el contenido del elemento es un nombre cualificado dominio de Windows. Un nombre cualificado dominio del usuario de Windows es una secuencia de la forma “NombreDominio\NombreUsuario”. El Nombre de Dominio y el separador “\” pueden ser omitidos.

5.- Nombre del Principal de Kerberos

URI: urn:oasis:names:tc:SAML:2.0:nameid-format:kerberos

Indica que el contenido del elemento está en la forma de un nombre principal de Kerberos usando el formato nombre[/instancia]@REINO. La sintaxis, el formato y los caracteres permitidos para el nombre, la instancia, y el reino se describen en la RFC 1510 [RFC 1510] del IETF.

6.- Identificador de Entidad:

URI: urn:oasis:names:tc:SAML:2.0:nameid-format:entity

Indica que el contenido del elemento es un identificador de una entidad que proporciona servicios SAML (como el solicitante, el respondedor, o una autoridad SAML) o que participa en un perfil SAML. Como identificador puede ser usado en el elemento <Issuer> para identificar al emisor de una petición, respuesta, o aserción SAML, o dentro del elemento <NameID>, para crear aserciones sobre entidades de sistema que puedan enviar peticiones, respuestas o aserciones SAML. También puede ser usado en otros elementos y atributos cuyo propósito sea identificar a una entidad de sistema dentro de varios protocolos de intercambios.

La sintaxis de este identificador es una URI de no más de 1024 caracteres de longitud. Se recomienda que las entidades de sistema usen una URL que contenga su propio nombre de dominio para identificarse.

Los atributos NameQualifier, SPNameQualifier y SPProvidedID deben ser omitidos.

7.- Identificadores persistentes

URI: urn:oasis:names:tc:SAML:2.0:nameid-format:persistent

Indica que el contenido del elemento es un identificador opaco persistente para un principal que esta determinado por un proveedor de identidad y un proveedor de servicio o afiliación de proveedores. Los identificadores de nombres persistentes generados por los proveedores de identidad se deben construir usando valores pseudo-aleatorios que no tienen ninguna correspondencia perceptible con el identificador actual del sujeto (por ejemplo, nombre de usuario). La intención es crear un seudónimo no público para prevenir el descubrimiento de la identidad o de las actividades del sujeto. Los identificadores de nombres persistentes no deben exceder de una longitud de 256 caracteres.

8.- Identificadores Transitorios

URI: urn:oasis:names:tc:SAML:2.0:nameid-format:transient

Indica que el contenido del elemento es un identificador con la semántica transitoria y se debe tratar como un valor opaco y temporal por la parte que confía. Los valores de los identificadores transitorios se deben generar de acuerdo con las reglas para los identificadores SAML), y no deben exceder una longitud de 256 caracteres. Los

atributos NameQualifier y SPNameQualifier se pueden utilizar para significar que el identificador representa a parejas de identificadores transitorios y temporales. En tal caso, pueden ser omitidos de acuerdo con las reglas especificadas en la sección de identificadores persistentes.

9.2.3.- Elemento <Subject>

Repasando el elemento <Assertion>, vemos que uno de los elementos que lo forma es el elemento <Subject>. En esta sección pasamos a definirlo. Para ello nos basamos en la definición de su esquema.

```
<element name="Subject" type="saml:SubjectType"/>
<complexType name="SubjectType">
        <choice>
                <sequence>
                        <choice>
                                <element ref="saml:BaseID"/>
                                <element ref="saml:NameID"/>
                                <element ref="saml:EncryptedID"/>
                        </choice>
                        <element ref="saml:SubjectConfirmation" minOccurs="0"
                                                        maxOccurs="unbounded"/>
                </sequence>
                <element ref="saml:SubjectConfirmation" maxOccurs="unbounded"/>
        </choice>
</complexType>
```

Figura 9. 13: Definición del elemento <**Subject**> y su tipo complejo **SubjectType**

El elemento opcional <Subject> determina al principal que es el sujeto de todas (cero o más) las declaraciones de la aserción. Contiene un identificador, una serie de uno o más confirmaciones de sujeto o ambos:

- <BaseID>, <NameID>, o <EncryptedID> (Opcional)
  Identifica al sujeto.

- <SubjectConfirmation> [Cero o más]
  Información que permite al sujeto ser confirmado. Si se proporciona más de una confirmación del sujeto, entonces es suficiente con satisfacer una de ellas para confirmar al sujeto en la aplicación de la aserción.

Un elemento <Subject> puede contener un identificador y cero o más confirmaciones de sujeto que las partes que confían pueden verificar cuando se procese la aserción. Si se verifican cualquiera de las confirmaciones de sujeto incluidas, la parte que confía puede manejar la entidad como una a la que la parte que da testimonio la asoció con el principal identificado en el identificador de nombre y asociado con las declaraciones en la aserción. Esta entidad que da testimonio y el sujeto actual puede o no ser la misma entidad. Si no se incluyen confirmaciones de sujeto, entonces no se especifica ninguna relación entre el presentador de la aserción y el sujeto actual.

Un elemento <Subject> no deberá identificar más de un principal. A modo de resumen se ponen los elementos en una tabla.

Elemento	*Descripción*	*Uso*
<BaseID> **<NameID>** **<EncryptedID>**	Identifican al sujeto	Opcional
<SubjetcConfirmation>	Permite al sujeto ser confirmado	Cero o más

Tabla 9. 5: Elementos del elemento **<Subject>**

Elemento <SubjectConfirmation>

El elemento <SubjectConfirmation> está incluido dentro del elemento <Subject> que es el que identifica al sujeto dentro de una aserción. Proporciona los medios por los que la parte que confía puede verificar la correspondencia del sujeto de la aserción con la parte a la que se está comunicando dicha parte que confía. Sigue el siguiente esquema que lo define:

```
<element name="SubjectConfirmation" type="saml:SubjectConfirmationType"/>
<complexType name="SubjectConfirmationType">
        <sequence>
                <choice minOccurs="0">
                        <element ref="saml:BaseID"/>
                        <element ref="saml:NameID"/>
                        <element ref="saml:EncryptedID"/>
                </choice>
                <element ref="saml:SubjectConfirmationData" minOccurs="0"/>
        </sequence>
        <attribute name="Method" type="anyURI" use="required"/>
</complexType>
```

Figura 9. 14: Definición del elemento **<SubjectConfirmation>** y su tipo complejo **SubjectConfirmationType**

El elemento <SubjectConfirmation>, a su vez, contiene los siguientes atributos y elementos:

- Method (Obligatorio)

Es una referencia URI que identifica un protocolo o mecanismo que debe usarse para confirmar al sujeto. Las referencias URI que identifican métodos de confirmación definidos en SAML son actualmente definidos en la especificación de perfiles SAML. Se pueden añadir métodos adicionales definiendo nuevas URIs y perfiles o mediante acuerdos privados.

- <BaseID>, <NameID>, o <EncryptedID> (Opcional)

Identifica la entidad que se espera que satisfaga los requerimientos de confirmación de sujeto incluidos.

- <SubjectConfirmationData> (Opcional)

Información de confirmación adicional que es usada por el método de confirmación. Por ejemplo, el contenido típico de este elemento podría ser el elemento <ds:KeyInfo> como se define en la especificación de procesado y sintaxis de firma XML, el cual identifica una clave criptográfica. Métodos de confirmación particulares pueden definir un tipo de esquema para describir los elementos, atributos, o contenidos que pueden aparecer en el elemento <SubjectConfirmationData>.

Elemento <SubjectConfirmationData>

Para ubicarnos un poquito, hemos visto el elemento <Assertion> que poseía un elemento <Subject>. Este, a su vez, tenía un elemento <SubjectConfirmation> y es dentro de este elemento donde colocamos al que viene ahora. El elemento <SubjectConfirmationData> tiene el tipo complejo **SubjectConfirmationDataType** y sigue el esquema siguiente:

```
<element name="SubjectConfirmationData" type="saml:SubjectConfirmationDataType"/>
<complexType name="SubjectConfirmationDataType" mixed="true">
        <complexContent>
            <restriction base="anyType">
                <sequence>
                        <any namespace="##any" processContents="lax" minOccurs="0"
                                maxOccurs="unbounded"/>
                </sequence>
                <attribute name="NotBefore" type="dateTime" use="optional"/>
                <attribute name="NotOnOrAfter" type="dateTime" use="optional"/>
                <attribute name="Recipient" type="anyURI" use="optional"/>
                <attribute name="InResponseTo" type="NCName" use="optional"/>
                <attribute name="Address" type="string" use="optional"/>
                <anyAttribute namespace="##other" processContents="lax"/>
            </restriction>
        </complexContent>
</complexType>
```

Figura 9. 15: Definición del elemento <**SubjectConfirmationData**> y su tipo complejo **SubjectConfirmationDataType**

Especifica datos adicionales que permiten confirmar al sujeto o limita las circunstancias bajo las cuales se puede llevar a cabo la confirmación de sujeto. La confirmación de sujeto tiene lugar cuando una parte que confía trata de verificar la relación entre una entidad que presenta la aserción (es decir, la parte que da testimonio) y el sujeto de las afirmaciones de la aserción. Contiene los siguientes atributos opcionales que pueden aplicarse a cualquier método:

- NotBefore (Opcional)

Es un instante de tiempo antes del cual el sujeto no puede ser confirmado. El valor de tiempo se codifica en UTC.

- NotOnOrAfter (Opcional)

Es un instante de tiempo después del cual no se puede confirmar al sujeto. El valor de tiempo se codifica en UTC.

- Recipient (Opcional)

Es una URI que especifica la entidad o localización a la que la parte que da testimonio puede presentar la aserción. Por ejemplo, este atributo podría indicar que la aserción debe ser entregada a un punto final de una red particular con el fin de prevenir un intermediario que la redireccione a cualquier otro lugar.

- InResponseTo (Opcional)

La ID de un mensaje de protocolo SAML en respuesta del cual la entidad que atestigua puede presentar la aserción. Por ejemplo, este atributo podría ser usado para correlacionar la aserción con una petición SAML que dio lugar a su presentación.

- Address (Opcional)

La dirección/localización de la red de la cual una entidad que atestigua puede presentar la aserción. Por ejemplo, este atributo se pudo utilizar para unir la aserción a las direcciones particulares del cliente para evitar que un atacante robe y presente fácilmente la aserción desde otra localización.

Las direcciones IPv4 deberán ser presentadas en el formato decimal con puntos (ejemplo, “1.2.3.4”). Las direcciones IPv6 deberán ser presentadas de acuerdo con IETF RFC 3513 (ejemplo, “FEDC:BA98:7654:3210:FEDC:BA98:7654:3210”).

A modo de resumen listamos los atributos que contiene este elemento en la siguiente tabla:

Atributo	*Descripción*	*Uso*
NotBefore	Marca el tiempo más pequeño de validez	Opcional
NotOnOrAfter	Marca el tiempo más grande de validez	Opcional
Recipient	Entidad a la que se presenta la aserción	Opcional
InResponseTo	ID de mensaje al que se responde con la aserción	Opcional
Address	La dirección a la que se presenta la aserción	Opcional
AnyAttribute	Atributos arbitrarios	

Tabla 9. 6: Atributos del elemento <**SubjectConfirmationData**>

Ahora definimos aquellos atributos y elementos arbitrarios, es decir, aquellos que se pueden añadir como extensión.

- Atributos Arbitrarios

Este tipo complejo usa un punto de extensión <xs:anyAttribute> para permitir que atributos con espacio de nombres cualificados XML, sean añadidos a construcciones <SubjectConfirmationData> sin la necesidad de una extensión de esquema adicional. Esto permite que los campos adicionales sean agregados como necesarios para proporcionar la información relacionada con la confirmación adicional.

Las extensiones SAML no deben añadir atributos XML locales (sin espacio de nombres cualificados) o atributos XML cualificados por espacios de nombres definidos por SAML para el tipo complejo **SubjectConfirmationDataType** o una derivación de él. Tales atributos son reservados para el mantenimiento y realce del propio SAML.

- Elementos Arbitrarios

Este tipo complejo usa un punto de extensión <xs:any> para permitir que elementos XML arbitrarios sean añadidos en construcciones <SubjectConfirmationData> sin la necesidad de una extensión de esquema explícita. Esto permite añadir elementos adicionales como necesarios para proporcionar información relacionada con la confirmación adicional.

Los métodos y los perfiles particulares de confirmación que hacen uso de esos métodos pueden requerir el uso de uno o más de los atributos definidos dentro de este tipo complejo.

Observamos que el período especificado por los atributos opcionales NotBefore y NotOnOrAfter, si están presentes, debe caer dentro del período total de validez de la aserción según lo especificado por los atributos NotBefore y NotOnOrAfter del elemento <Conditions>. Si están presentes ambos atributos, el valor para NotBefore debe ser menor que (anterior que) el valor para NotOnOrAfter.

Tipo Complejo Type KeyInfoConfirmationDataType

El tipo complejo **KeyInfoConfirmationDataType** limita el elemento <SubjectConfirmationData> que contiene uno o más elementos <ds:KeyInfo>. Estos son elementos que identifican las claves criptográficas que se utilizan para autentificar una entidad que atestigua. Ponemos el esquema que define a este tipo complejo a continuación.

```
<complexType name="KeyInfoConfirmationDataType" mixed="false">
        <complexContent>
                <restriction base="saml:SubjectConfirmationDataType">
                        <sequence>
                                <element ref="ds:KeyInfo" maxOccurs="unbounded"/>
                        </sequence>
                </restriction>
        </complexContent>
</complexType>
```

Figura 9. 16: Definición del tipo complejo **KeyInfoConfirmationDataType**

El método particular de confirmación debe definir el mecanismo exacto por el cual se pueden utilizar los datos de confirmación. También pueden aparecer los atributos opcionales definidos por el tipo complejo **SubjectConfirmationDataType**. Este tipo complejo, o un tipo derivado de él, debe ser utilizado por cualquier método de confirmación que defina sus datos de confirmación en los términos del elemento <ds:KeyInfo>.

Observamos que de acuerdo con [XMLSig], cada elemento <ds:KeyInfo> debe identificar una única clave criptográfica. Múltiples claves pueden ser identificadas mediante elementos <ds:KeyInfo>, tales como cuando un principal utiliza claves diferentes para confirmarse a diversas partes que confían.

Ejemplo de <Subject> confirmado mediante clave

Para ilustrar la manera mediante la cual varios elementos y tipos se adecuan juntos, más abajo aparece un ejemplo de un elemento <Subject> que contiene un identificador de nombre y una confirmación de sujeto que se basa en la prueba de la posesión de la clave.

Hacemos notar el uso de **KeyInfoConfirmationDataType** para identificar la sintaxis de datos de confirmación como si fuese un elemento <ds:KeyInfo>:

```
<Subject>
        <NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">
                scott@example.org
        </NameID>
        <SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:holder-of-key">
                <SubjectConfirmationData xsi:type="saml:KeyInfoConfirmationDataType">
                        <ds:KeyInfo>
                                <ds:KeyName>Scott's Key</ds:KeyName>
                        </ds:KeyInfo>
                </SubjectConfirmationData>
        </SubjectConfirmation>
</Subject>
```

Figura 9. 17: Ejemplo de <**Subject**> confirmado mediante clave

9.2.4.- Elemento <Conditions>

Conviene recordar que el elemento <Assertion> tenía varios elementos. Ya hablamos del elemento <Subject> en una sección anterior. Ahora pasamos a describir otro de los elementos de <Assertion>, el elemento <Conditions>. Este elemento define las construcciones SAML que ponen limitaciones al uso de las aserciones. A continuación aparece el esquema de este elemento y su tipo complejo:

```
<element name="Conditions" type="saml:ConditionsType"/>
<complexType name="ConditionsType">
        <choice minOccurs="0" maxOccurs="unbounded">
                <element ref="saml:Condition"/>
                <element ref="saml:AudienceRestriction"/>
                <element ref="saml:OneTimeUse"/>
                <element ref="saml:ProxyRestriction"/>
        </choice>
        <attribute name="NotBefore" type="dateTime" use="optional"/>
        <attribute name="NotOnOrAfter" type="dateTime" use="optional"/>
</complexType>
```

Figura 9. 18: Definición del elemento <**Conditions**>

Dentro del elemento <Conditions> nos encontramos los siguientes elementos y atributos:

- NotBefore (Opcional)

Especifica el instante de tiempo más temprano en el que la aserción es válida. Este tiempo se codifica en formato UTC.

- NotOnOrAfter (Opcional)

Especifica el instante de tiempo en el que la aserción expira. Este tiempo se codifica en formato UTC.

- <Condition> (Cualquier número)

Es una condición de un tipo definido en un esquema extensión. Un atributo xsi:type debe ser usado para indicar el tipo actual de condición.

- <AudienceRestriction> (Cualquier número)

Determina que la aserción es direccionada hacia un público determinado.

- <OneTimeUse> (Opcional)

La aserción debe ser usada inmediatamente y no debe ser retenida para un uso futuro. Aunque el esquema permita que pueda haber multiples ocurrencias, debe haber al menos una instancia de este elemento

- <ProxyRestriction> (Opcional)

Limitaciones que impone la parte que afirma a la que confía cuando esta última actúa como la primera y manda aserciones en base a la información de la aserción original. Aunque el esquema permita que pueda haber múltiples ocurrencias, debe haber al menos una instancia de este elemento

En las tablas siguientes pasamos a listar los atributos y elementos que contiene el elemento <Conditions>:

Atributo	*Descripción*	*Uso*
NotBefore	Marca el tiempo más pequeño de validez	Opcional
NotOnOrAfter	Marca el tiempo más grande de validez	Opcional

Tabla 9. 7: Atributos del elemento <**Conditions**>

Elemento	*Descripción*	*Uso*
<Condition>	Condición impuesta a una extensión del esquema	Opcional
<AudienceRestriction>	Audiencia a la que va dirigida la aserción	Opcional
<OneTimeUse>	Se debe usar la aserción una sola vez	Opcional
<ProxyRestriction>	Limitaciones a las entidades	Opcional

Tabla 9. 8: Elementos del elemento <**Conditions**>

El uso del atributo xsi:type permitiría a una aserción contener más de una instancia de un subtipo de **ConditionsType** (como **OneTimeUseType**). El esquema no limita explícitamente el número de condiciones de tiempo. Un determinado tipo de condición puede definir límites en ese uso.

Ya hemos explicado este elemento. El apartado sigue definiendo las normas generales para el procesamiento de este elemento, unas normas para ver si son válidos

los atributos NotBefore y NotOnOrAfter, terminando con la definición de los elementos que contiene.

Normas generales de procesado del elemento Conditions:

Una vez definido el elemento, tenemos que ver cuándo podemos decir que una aserción es válida en base a las condiciones que dicho elemento impone. De esta función se encarga esta sección. Para ello se darán unas condiciones de uso que si son cumplidas se puede dar por válida la aserción.

Si una aserción contiene un elemento <Conditions>, entonces la validación de ella depende de los atributos y sub-elementos proporcionados, usando las siguientes normas en el orden mostrado abajo.

Advertimos que una aserción que tenga un estado de validación de una condición ***Valid*** puede ser, aún así, no digna de confianza o no válida por razones cómo no estar bien formada, no mandarse a través de una autoridad SAML de confianza o no ser autentificado por medios de confianza.

También advertimos que algunas condiciones pueden no influenciar en la validación del contenido de la aserción (se evalúan siempre cómo ***Valid***), pero pueden restringir el comportamiento de las partes que confían en el uso de ésta.

1. Si no hay ningún sub-elemento o atributo dentro del elemento <Condition>, entonces la aserción se considera como ***Valid*** respecto al procesado de la condición.
2. Si cualquier sub-elemento o atributo dentro del elemento <Condition>, se evalúa como no válida, entonces la aserción se considera ***Invalid***.
3. Si cualquier sub-elemento o atributo dentro del elemento <Condition> no puede ser evaluado, o si no se entiende un elemento, entonces la validación de la aserción se considera como ***Indeterminate***.
4. Si todos los sub-elementos o atributos dentro del elemento <Condition> son evaluados como ***Valid***, entonces la aserción es considerada como válida respecto a las normas de procesado.

La primera regla termina el procesado de las condiciones. Como determinación si una aserción es ***Invalid***, toma preferencia sobre una ***Indeterminate***.

Una aserción que se evalúa como ***Invalid*** o ***Indeterminate*** debe ser rechazada por la parte que confía (dentro de cualquier contexto o perfil que esté siendo procesado), como si una aserción no estuviese bien formada o por el contrario no se pudiera usar.

Atributo NotBefore y NotOnOrAfter

Este atributo especifica la limitación de tiempo de la validez de la aserción dentro del contexto del perfil de uso. No se garantiza que las declaraciones de la aserción sean correctas o exactas a lo largo del periodo de validez. El principio del intervalo de validez empieza donde marca NotBefore, marcadándose el final mediante NotOnOrAfter. Si se omiten alguno de los valores, entonces se considera no especificado este atributo, considerándose la aserción como ***Valid*** con respecto a las condiciones de tiempo (siempre y cuando las demás condiciones la validen también).

Si aparecen ambos atributos, el valor de NotBefore debe ser menor que NotOnOrAfter.

A partir de ahora pasamos a definir los elementos que contiene el elemento <Conditions>. Estos son cuatro:
- <Condition>
- <AudienceRestriction> y <Audience>
- <OneTimeUse>
- <ProxyRestriction>

Elemento <Condition>

El elemento <Condition> sirve como punto de extensión para nuevas condiciones. Su tipo complejo **ConditionAbstractType** es abstracto y por lo tanto se usa sólo como base para tipos derivados. Ponemos a continuación el esquema del elemento:

```
<element name="Condition" type="saml:ConditionAbstractType"/>
<complexType name="ConditionAbstractType" abstract="true"/>
```

Figura 9. 19: Definición del elemento <**Condition**>

Elementos <AudienceRestriction> y <Audience>

El elemento <AudienceRestriction> indica que la aserción se redirecciona hacia una o más audiencias particulares identificadas por los elementos <Audience>. Contiene el siguiente elemento:

- <Audience>

Es una referencia URI que identifica la futura audiencia. Dicha referencia puede identificar a un documento que describe los términos y condiciones de los miembros de la audiencia. Puede también contener el identificador URI único de un identificador de nombre SAML que describe una entidad de sistema.

```
<element name="AudienceRestriction"
type="saml:AudienceRestrictionType"/>
<complexType name="AudienceRestrictionType">
        <complexContent>
                <extension base="saml:ConditionAbstractType">
                        <sequence>
                            <element ref="saml:Audience" maxOccurs="unbounded" / >
                        </sequence>
                </extension>
        </complexContent>
</complexType>
<element name="Audience" type="anyURI"/>
```

Figura 9. 20: Definición del elemento <**AudienceRestriction**> y su tipo complejo **AudienceRestrictionType**

La condición de restricción de la audiencia se evalúa como ***Valid*** si y sólo si la parte que confía es miembro de una o más de las audiencias especificadas. La parte que afirma no puede impedir que una parte a la que se le descubra la aserción actúe en base a la información proporcionada. Sin embargo, el elemento <AudienceRestriction> le permite declarar que no se proporciona garantía a cada parte de forma legible tanto por los humanos y por las máquinas.

Advertimos que muchos elementos <AudienceRestriction> pueden ser incluidos en una aserción, y cada una debe ser evaluada independientemente. El efecto de este requisito y la definición anterior es que dentro de una condición dada, las audiencias forman una disyunción (un "OR") mientras las múltiples condiciones forman una conjunción (un "AND").

Elemento <OneTimeUse>

En general, las partes que confían pueden elegir retener las aserciones, o la información que contienen para volver a usarlas de cualquier otra forma. El elemento condición <OneTimeUse> permite a una autoridad indicar que la información de la aserción es probable que cambie muy pronto y debe obtenerse información nueva para cada uso. Un ejemplo podría ser una aserción que tenga un <AuthzDecisionStatement> (es decir, una declaración de decisión de autorización) la cual podría ser el resultado de una política en la que el control de acceso se determina según el tiempo del día. Si relojes del sistema en un medio distribuido pueden ser sincronizados con precisión, entonces este requisito puede ser usado cuidadosamente en el intervalo de validación. Sin embargo, no es muy conveniente limitar la vida de las aserciones ya que esta podría llegar al destinatario una vez finalizada debido a saltos entre los relojes combinados con posibles retrasos en la transmisión. El esquema de definición de este elemento se pone a continuación.

```
<element name="OneTimeUse" type="saml:OneTimeUseType"/>
<complexType name="OneTimeUseType">
        <complexContent>
                <extension base="saml:ConditionAbstractType"/>
</complexContent>
</complexType>
```

Figura 9. 21: Definición del elemento **<OneTime>** y su tipo complejo **OneTimeUseType**

Este elemento indica que la aserción debe usarse inmediatamente por la parte que confía y no ser retenida para un uso futuro. Dicha parte puede siempre pedir una petición actual para cada uso. Sin embargo, las implementaciones que eligen retener aserciones deben observar el elemento <OneTimeUse>. Esta condición es independiente de la información de las condiciones NotBefore y NotOnOrAfter.

Para respaldar la limitación de un solo uso, la parte que confía debe tener memoria de las aserciones que ha procesado conteniendo esta condición. Cuando se procesa una aserción con esta condición, la memoria debe ser chequeada para asegurarnos que la misma aserción no ha sido previamente recibida y procesada por dicha parte.

Una autoridad SAML no debe incluir más de un elemento <OneTimeUse> dentro del elemento <Conditions> de la aserción. Con el fin de determinar la validez del elemento <Conditions>, el elemento <OneTimeUse> se considera siempre ***Valid***. Es decir, esta condición no afecta a la validez pero si es una condición de uso.

Elemento <ProxyRestriction>

Es el último de los elementos que contiene el elemento <Conditions> que vamos a explicar. Especifica las limitaciones que impone la parte que afirma a la que confía que desea actuar como parte afirmante y mandar aserciones que contienen la información de la aserción original. Una parte que confía actuando como parte afirmante no debe mandar aserciones que infrinjan las restricciones especificadas en la aserción original.

```
<element name="ProxyRestriction" type="saml:ProxyRestrictionType"/>
<complexType name="ProxyRestrictionType">
    <complexContent>
      <extension base="saml:ConditionAbstractType">
        <sequence>
          <element ref="saml:Audience" minOccurs="0" maxOccurs="unbounded"/>
        </sequence>
          <attribute name="Count" type="nonNegativeInteger" use="optional" />
      </extension>
    </complexContent>
</complexType>
```

Figura 9. 22: Definición del elemento **<ProxyRestriction>**

Contiene los siguientes elementos y atributos:

- Count (Opcional)

Especifica el número máximo de redirecciones que la parte que afirma permite que haya entre esta aserción y la última aserción que se mandó en base a la primera.

-<Audience> (Cero o más)

Indica un conjunto de audiencias a las cuales la parte que afirma permite que manden aserciones en base a la actual.

En los párrafos siguientes pasaremos a detallar aquellas reglas que se deben seguir para un correcto uso de este elemento.

Un valor de **Count** de cero indica que la parte que confía no debe mandar una aserción que se base en la actual hacia ninguna otra parte que confía. Si es mayor que cero, cualquier aserción así mandada debe tener un elemento **<ProxyRestriction>** con un valor de Count de al menos uno menos que el del valor anteriormente citado.

Si no se especifican elementos **<Audience>**, entonces no se imponen limitaciones en las partes que confían que se les pueda mandar la aserción. De lo contrario, cualquier aserción así mandada debe contener un elemento <AudienceRestriction> con al menos uno de los elementos <Audience> presente en el

anterior elemento <AudienceRestriction> y no presentar elementos <Audience> que no estuvieran en el citado elemento anterior. Una autoridad SAML no debe incluir más que un elemento <ProxyRestriction> dentro del elemento <Conditions> de la aserción.

Con el fin de decidir la validez del elemento <Conditions>, este elemento se considera siempre ***Valid***. Es decir, esta condición no afecta a la validez pero si es una condición de uso.

9.2.5.- Declaraciones (Statements)

Hacemos un repaso general de lo que llevamos explicado del elemento <Assertion>, indicando que llevamos explicado el elemento <Subject> (aquel que sirve para identificar al sujeto al que hace referencia la aserción) y el elemento <Conditions> (impone condiciones que nos servirán para saber si la aserción es válida o no). Ahora pasamos a definir el elemento <Statements>, es decir, las construcciones SAML que contienen información de declaraciones.

Elemento <Statement>

Este elemento es un punto de extensión que permite a otras aplicaciones basadas en aserciones volver a usar el marco de aserciones SAML. El propio núcleo de declaraciones de SAML deriva de este punto de extensión. Su tipo complejo **StatementAbstractType** es abstracto por lo que sólo podemos usarlo como base de tipos derivados. A continuación ponemos un trozo de esquema explicativo:

```
<element name="Statement" type="saml:StatementAbstractType"/>
<complexType name="StatementAbstractType" abstract="true"/>
```

Figura 9. 23: Definición del elemento **<Statement>**

Elemento <AuthnStatement>

Describe una declaración realizada por la parte que afirma, diciendo que el sujeto fue autentificado por unos medios particulares en un tiempo determinado. Las aserciones que contienen elementos <AuthnStatement> deben contener un elemento <Subject>. Su tipo **AuthnStatementType**, extiende de **StatementAbstractType**. En él se añaden los siguientes elementos y atributos:

- AuthnInstant (Obligatorio)

Es el tiempo en el que se realizó la autentificación. Dicho valor de tiempo se codifica en UTC.

- SessionIndex (Opcional)

Especifica el índice de una sesión particular entre el principal y la autoridad que autentifica.

- SessionNotOnOrAfter (Opcional)

Especifica el tiempo en el que se tiene que dar por terminada la sesión existente entre el principal y la autoridad SAML. El valor de tiempo se codifica en UTC. No hay relación entre este atributo y el atributo de condición NotOnOrAfter que debe estar presente en la aserción.

- <SubjectLocality> (Opcional)
Indica el nombre de dominio DNS y la dirección IP para el sistema del cual el sujeto fue aparentemente autentificado.

- <AuthnContext> (Obligatorio)
Es el contexto usado por la autoridad de autentificación e incluso los eventos que la propiciaron. Contiene una referencia a una clase de contexto de autentificación, una declaración de contexto o ambos.

El esquema del elemento se añade a continuación.

```
<element name="AuthnStatement" type="saml:AuthnStatementType"/>
<complexType name="AuthnStatementType">
   <complexContent>
      <extension base="saml:StatementAbstractType">
          <sequence>
                <element ref="saml:SubjectLocality" minOccurs="0"/>
                <element ref="saml:AuthnContext"/>
          </sequence>
          <attribute name="AuthnInstant" type="dateTime" use="required"/>
          <attribute name="SessionIndex" type="string" use="optional"/>
          <attribute name="SessionNotOnOrAfter" type="dateTime"use="optional"/ >
                </extension>
        </complexContent>
</complexType>
```

Figura 9. 24: Definición del **tipo <AuthnStatement>** y el tipo complejo **AuthnStatementType**

A continuación comentamos unas **reglas generales de uso** de los elementos y atributos que componen este elemento.

En general, cualquier cadena puede ser usada como valor **SessionIndex**. Sin embargo, cuando se considera la privacidad, hay que tomar cuidado en asegurarse que el valor de SessionIndex no invalida otros mecanismos de seguridad. En consecuencia, el valor no debería ser usado relacionar actividades de un principal a través de varias sesiones. Para solucionar esto se proporcionan dos soluciones:

- Usar para el valor de SessionIndex un valor entero positivo pequeño. La autoridad SAML debe escoger un rango de valores de tal modo que el valor de cualquiera de ellos debe ser lo suficientemente alto para evitar acciones de un principal determinado en un contexto de multiples participantes de sesión. Dicha autoridad debe tomar valores aleatorios para este atributo dentro de este rango (excepto cuando se necesite asegurar valores únicos para declaraciones siguientes dadas por el mismo participante de la sesión pero que forme parte de una sesión diferente).
- Usar el valor de ID incluido en la aserción en SessionIndex.

A modo de resumen y para aclarar mejor los elementos y atributos que posee, los listamos a continuación.

Atributo	*Descripción*	*Uso*
AuthnInstant	Tiempo en el que se realizó la autentificación	Obligatorio
SessionIndex	Índice de la sesión establecida	Opcional
SessionNotOnOrAfter	Tiempo en el que se tiene que terminar la sesión	Opcional

Tabla 9. 9: Atributos del elemento <**AuthnStatement**>

Elemento	*Descripción*	*Uso*
<SubjectLocality>	Nombre y dirección del sistema del sujeto	Opcional
<AuthnContext>	Contexto usado por autoridad autentificación	Obligatorio

Tabla 9. 10: Elementos del elemento <**AuthnStatement**>

Elemento <SubjectLocality>

Este elemento pertenece al elemento <AuthnStatement> que a su vez pertenece a <Statement>. Especifica el nombre de dominio DNS y la dirección IP para el sistema del cual el sujeto de la aserción fue autentificado. Ponemos el fragmento de esquema y su tipo complejo **SubjectLocalityType**:

```
<element name="SubjectLocality" type="saml:SubjectLocalityType"/>
<complexType name="SubjectLocalityType">
        <attribute name="Address" type="string" use="optional"/>
        <attribute name="DNSName" type="string" use="optional"/>
</complexType>
```

Figura 9. 25: Definición del elemento <**SubjectLocality**>

Posee los siguientes atributos. Primero los explicaremos y pondremos al final una tabla en donde podremos verlos gráficamente:

- Address (Opcional)

Es la dirección de red del sistema que autentificó al principal identificado por el sujeto. Las direcciones IPv4 deben ser representadas en decimal (por ejemplo "*1.2.3.4*"). Las direcciones IPv6 deben ser representadas de la forma definida en la RFC 3513 de IETF (por ejemplo "*FEDC:BA98:7654:3210:FEDC:BA98:7654:3210*").

- DNSName (Opcional)

El nombre DNS del sistema que autentificó al principal identificado por el sujeto.

Atributo	*Descripción*	*Uso*
Address	Dir. de red del sistema que autentificó al sujeto	Obligatorio
DNSName	Nombre DNS sistema que autentificó al sujeto	Opcional

Tabla 9. 11: Atributos del elemento <**SubjectLocality**>

Este elemento es por completo **consultivo** puesto que ambos campos son fácilmente imitados, pero puede ser información útil en algunas aplicaciones.

Elemento <AuthnContext>

Junto con el elemento anterior, <SubjectLocality>, forman el elemento <AuthnStatement> que pertenece a <Statement>. Especifica el contexto de un evento de autentificación. El elemento puede contener una declaración o una referencia a una clase de contexto de autentificación o ambos. Su tipo complejo **AuthnContextType** tiene el siguiente esquema:

```
<element name="AuthnContext" type="saml:AuthnContextType"/>
<complexType name="AuthnContextType">
        <sequence>
                <choice>
                        <sequence>
                                <element ref="saml:AuthnContextClassRef"/>
                                <choice minOccurs="0">
                                        <element ref="saml:AuthnContextDecl"/>
                                        <element ref="saml:AuthnContextDeclRef"/>
                                </choice>
                        </sequence>
                        <choice>
                                <element ref="saml:AuthnContextDecl"/>
                                <element ref="saml:AuthnContextDeclRef"/>
                        </choice>
                </choice>
                <element ref="saml:AuthenticatingAuthority" minOccurs="0"
                                                        maxOccurs="unbounded"/>
        </sequence>
</complexType>
<element name="AuthnContextClassRef" type="anyURI"/>
<element name="AuthnContextDeclRef" type="anyURI"/>
<element name="AuthnContextDecl" type="anyType"/>
<element name="AuthenticatingAuthority" type="anyURI"/>
```

Figura 9. 26: Definición del elemento <**AuthnContext**>

Tiene los siguientes elementos. Como en anteriores elementos, primero los explicaremos y después los listaremos en una tabla.

- <AuthnContextClassRef> (Opcional)

Es una referencia URI que identifica una clase de contexto de autentificación que describe el contexto que le sigue.

- <AuthnContextDecl> o <AuthnContextDeclRef> (Opcional)

Bien una declaración de contexto proporcionada por valor, bien una referencia URI que identifica tal declaración. La referencia URI puede resolverse en un documento XML que contiene la declaración referenciada.

- <AuthenticatingAuthority> (Cero o más)

Cero o más identificadores únicos de autoridades de autentificación que estaban implicadas en la autentificación del principal (no incluyendo el emisor de la aserción, que se presume que está implicado sin estar nombrado explícitamente aquí).

Elemento	*Descripción*	*Uso*
<AuthnContextClassRef>	Clase de contexto de autentificación	Opcional
<AuthnContextDecl> **<AuthnContextDeclRef>**	Declaración a contexto autentificación o una referencia al mismo	Opcional
<AuthenticatingAuthority>	ID de autoridades de autentificación	Cero o más

Tabla 9. 12: Elementos del elemento **<AuthnContext>**

9.2.6.- Advice

Recordemos que este es el último elemento de <Assertion> que vamos a explicar. Ya explicamos con anterioridad <Subject>, <Conditions> y <Statement>. En este punto, donde se explica el elemento <Advice>, se definen las construcciones SAML que contienen información adicional sobre una aserción que la parte que afirma desea proporcionar a una parte que confía.

Este elemento, <Advice>, contiene cualquier información que la autoridad SAML quiera proporcionar. Esta información puede ser ignorada por las aplicaciones que no le afecten ni la semántica, ni la validez de la aserción. Ponemos su esquema y su tipo complejo **AdviceType**:

```
<element name="Advice" type="saml:AdviceType"/>
<complexType name="AdviceType">
        <choice minOccurs="0" maxOccurs="unbounded">
                <element ref="saml:AssertionIDRef"/>
                <element ref="saml:AssertionURIRef"/>
                <element ref="saml:Assertion"/>
                <element ref="saml:EncryptedAssertion"/>
                <any namespace="##other" processContents="lax"/>
        </choice>
</complexType>
```

Figura 9. 27: Definición del elemento **<Advice>**

Este elemento contiene una mezcla de cero o más elementos <Assertion>, <EncryptedAssertion>, <AssertionIDRef>, y <AssertionURIRef> y elementos calificadores de espacio de nombres en otros espacios de nombres que no son SAML.

Algunos de los **posibles usos de <Advice>**:

- Incluir una prueba, para que sea citada, apoyando la veracidad de la aserción, bien directamente (por medio de incorporar la afirmación) bien indirectamente (referenciando la aserción apoyada).
- Declarar una prueba de la veracidad de la aserción.
- Especificar los puntos de distribución y tiempo para actualizar la aserción.

9.3.- Protocolos SAML

Los mensajes de protocolos SAML pueden ser generados e intercambiados usando diferentes protocolos. La especificación **binding** de SAML describe los medios específicos de transporte de dichos mensajes usando protocolos de transporte ampliamente desarrollados. La especificación **Profile** de SAML describe un número de aplicaciones de los protocolos SAML junto con reglas de procesado adicionales, restricciones, y requerimientos que facilitan la interoperabilidad.

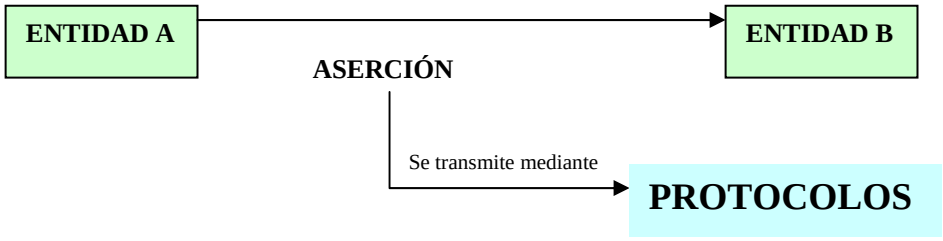

Figura 9. 28: Relación entre protocolo y aserción

De esta manera podemos decir que los protocolos definen cómo se intercambian los mensajes que contienen las aserciones. Existen muchos **tipos de protocolos**. SAML define un número de protocolos de petición/respuesta diferentes:

- Petición de unas o más aserciones (incluye una petición directa de las aserciones deseadas, así como preguntar para las aserciones que resuelven criterios particulares).
- Petición por la cual un director es autentificado con la correspondiente aserción de vuelta.
- Petición por la cual un identificador de nombre es registrado.
- Petición por la cual se avisa de la finalización de una federación.
- Recuperación un mensaje de protocolo que se ha solicitado por medio de un artefacto.
- Petición de un registro cercano-simultáneo de logout de un registro de las sesiones relacionadas ("single logout").
- Petición de identificador de nombre mapeado.

Vemos que los protocolos son **petición/respuesta**. Para estructurar el capítulo lo que haremos será explicar primero el espacio de nombres que define SAML. Después veremos las particularidades de los protocolos petición/respuesta generales de SAML y terminaremos explicando el protocolo que más nos interesa. Este es el protocolo de petición de autentificación que nos servirá para pedir una aserción.

En ciertos casos, cuando el perfil lo permita, una respuesta SAML puede generarse y enviarse sin que el respondedor haya recibido la correspondiente solicitud. Los protocolos definidos en SAML llevan a cabo las siguientes **acciones**. Mediante ellas podremos saber qué es lo que podemos hacer al aplicar un protocolo en concreto:

- Devolver una o más aserciones solicitadas. Esto puede ocurrir en respuesta bien de una petición directa para una determinada aserción, bien de pregunta sobre aserciones que cumplan un criterio particular.
- Realizar la autentificación sobre una petición y devolver la aserción correspondiente.
- Registrar o terminar el registro de un identificador de nombre.
- Encontrar un mensaje de protocolo que ha sido solicitado por medio de un artefacto.
- Realizar la desconexión de una colección de sesiones relacionadas (“single logout”).
- Proporcionar mapeo de identificadores de nombre.

9.3.1.- Declaración del espacio de nombres

El siguiente fragmento de esquema define el espacio de nombres XML y otra información de cabecera para el esquema protocolo:

```
<schema
        targetNamespace="urn:oasis:names:tc:SAML:2.0:protocol"
        xmlns="http://www.w3.org/2001/XMLSchema"
        xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
        :saml="urn:oasis:names:tc:SAML:2.0:assertion"
        xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
        elementFormDefault="unqualified"
        attributeFormDefault="unqualified"
        blockDefault="substitution"
        version="2.0">
        <import namespace="urn:oasis:names:tc:SAML:2.0:assertion"
                schemaLocation="saml-schema-assertion-2.0.xsd"/>
        <import namespace="http://www.w3.org/2000/09/xmldsig#"
                schemaLocation="http://www.w3.org/TR/2002/REC-xmldsig-core-
                                        20020212/xmldsig-core-schema.xsd"/>
        <annotation>
                <documentation>
                        Document identifier: saml-schema-protocol-2.0
                        Location: http://docs.oasis-open.org/security/saml/v2.0/
                                Revision histórica:
                        V1.0 (November, 2002):
                                Esquema Estandar inicial.
                        V1.1 (September, 2003):
                                Actualizaciones dentro del mismo espacio de nombres V1.0.
                        V2.0 (March, 2005):
                                 Nuevos esquemas prot. basados en esp. de nombres  V2.0.
                </documentation>
        </annotation>
...
</schema>
```

Figura 9. 29: Declaración del **espacio de nombres**

9.3.2.- Protocolo peticiones y respuestas

En la siguiente sección definimos las construcciones SAML y requerimientos básicos que subyacen en todos los mensajes de petición y respuesta usados en los protocolos SAML. Es decir, siempre que hagamos una petición o una respuesta SAML debemos seguir las pautas que marca este protocolo. Los mensajes petición y respuesta de SAML derivan del tipo común. El solicitante manda un elemento derivado de **RequestAbstractType** a un respondedor SAML y éste genera un elemento derivando desde **StatusResponseType**.

Figura 9. 30: Relación entre **tipos comunes**

Una vez definidos los tipos comunes que intervienen en los intercambios de mensajes de petición/respuesta, pasamos a explicar los elementos y atributos que los componen.

Tipo Complejo RequestAbstractType

Todas las peticiones SAML son de un tipo que deriva del tipo abstracto complejo **RequestAbstractType**. El esquema de definición de este tipo se pone a continuación.

```
<complexType name="RequestAbstractType" abstract="true">
        <sequence>
                <element ref="saml:Issuer" minOccurs="0"/>
                <element ref="ds:Signature" minOccurs="0"/>
                <element ref="samlp:Extensions" minOccurs="0"/>
        </sequence>
        <attribute name="ID" type="ID" use="required"/>
        <attribute name="Version" type="string" use="required"/>
        <attribute name="IssueInstant" type="dateTime" use="required"/>
        <attribute name="Destination" type="anyURI" use="optional"/>
        <attribute name="Consent" type="anyURI" use="optional"/>
</complexType>
<element name="Extensions" type="samlp:ExtensionsType"/>
<complexType name="ExtensionsType">
        <sequence>
                <any namespace="##other" processContents="lax"
maxOccurs="unbounded"/>
        </sequence>
</complexType>
```

Figura 9. 31: Definición del tipo **RequestAbstractType**

Dicho tipo define los atributos y elementos comunes que están asociados a todas las peticiones SAML:

- ID (Obligatorio)

Es un identificador para la petición. Es del tipo **xs:ID** y debe seguir las normas de los tipo de datos comunes. El valor de este atributo debe coincidir con el atributo InResponseTo de la correspondiente respuesta.

- Version (Obligatorio)

Es la versión de esta petición. El valor de la versión definida por la especificación es de “2.0”.

- IssueInstant (Obligatorio)

Es el instante en el que se manda la petición. El valor de tiempo se codifica en UTC.

- Destination (Opcional)

Referencia URI que indica la dirección a la que se debe enviar la petición. Es útil para prevenir envíos de peticiones maliciosas a destinatarios involuntarios, protección que es requerida por algunos bindings de protocolos. Si está presente, el destinatario actual debe comprobar que la referencia URI identifica a la localización en la que fue recibido el mensaje. Si no es así, la petición debe ser deshechada.

- Consent (Opcional)

Indica si se ha obtenido o no consentimiento (y bajo qué condiciones) al mandar la petición. Si no se da ningún valor, se toma el del identificador urn:oasis:names:tc:SAML:2.0:consent:unspecified

<saml:Issuer> (Opcional)

Identifica la entidad que generó el mensaje de petición.

<ds:Signature> (Opcional)

Una firma XML que autentifica al solicitante y proporciona integridad de mensaje.

<Extensions> (Opcional)

Este punto contiene elementos de extensión opcionales que son convenidos entre las partes que se comunican. No se requiere ningún esquema para hacer uso de este punto de extensión.

A modo de resumen incluimos una tabla con los atributos que contiene este tipo complejo:

Atributo	*Descripción*	*Uso*
ID	ID de la petición	Obligatorio
Version	Versión de la petición	Obligatorio
IssueInstant	Instante de envío de la petición	Obligatorio
Destination	Dirección a la que debemos enviar la petición	Opcional
Consent	Indica si ha habido consentimiento o no	Opcional

Tabla 9. 13: Atributos del tipo complejo **RequestAbstractType**

Podemos hacer lo mismo con los elementos que contiene.

Elemento	*Descripción*	*Uso*
<Issuer>	Entidad que generó la petición	Opcional
<ds:Signature>	Firma que autentifica al solicitante	Opcional
<Extensions>	Elementos de extensión del esquema	Opcional

Tabla 9. 14: Elementos del tipo complejo **RequestAbstractType**

Dependiendo de los requerimientos particulares de protocolos y perfiles, un solicitante SAML puede necesitar autentificarse, y puede requerirse integridad de mensajes. Autentificación e integridad de mensajes la proporciona los **binding** de protocolos. La petición SAML puede ser firmada, proporcionando tanto autentificación del solicitante como integridad del mensaje.

Para un correcto uso de este tipo complejo se deben seguir las siguientes **normas de uso**:
- Si se usa una firma, entonces debe aparecer el elemento **<ds:Signature>** y el respondedor SAML debe verificar la firma (esto es, que el mensaje no ha sido alterado). Si no es válida la firma, entonces el respondedor no debe fiarse de los contenidos de la petición y debe responder con un error. Si es válida, el respondedor debe evaluarla para determinar la identidad y la decencia del que firma y puede continuar procesando la petición o responder con un error (si la petición no es válida por alguna otra razón).

- Si se incluye un atributo **Consent** y su valor indica que se ha obtenido alguna forma del principal, entonces la petición debe ser firmada.

- Si un respondedor SAML estima que la petición no es válida de acuerdo con la sintaxis SAML o las reglas de proceso, entonces, si responde, debe devolver un mensaje de respuesta SAML con un elemento **<StatusCode>** de valor *urn:oasis:names:tc:SAML:2.0:status:Requester*. En algunos casos, por ejemplo durante un supuesto ataque de "negación de servicio", no responder nada esta justificado.

Tipo Complejo StatusResponseType

Todas las respuestas SAML son de un tipo que derivan del tipo complejo **StatusResponseType**. Este tipo define los atributos comunes asociados a todas las respuestas SAML:

- ID (Obligatorio):

Es un identificador para la respuesta. Es del tipo **xs:ID** y debe seguir las normas de los tipo de datos comunes.

- InResponseTo (Opcional):

Es una referencia al identificador de la petición que generó la respuesta, si la hubiera. No debe aparecer en las respuestas que no fueran generadas por una petición o cuando el valor del atributo ID de la petición no puede ser determinado (por ejemplo, si la petición no está bien formada. De lo contrario, debe aparecer y su valor debe coincidir con el atributo ID de la petición correspondiente.

- Version (Obligatorio):
Es la versión de la respuesta. El identificador para la versión SAML definida en esta especificación es “2.0”.

- IssueInstant (Obligatorio):
Es el instante de tiempo de la respuesta. Se codifica en UTC.

- Destination (Opcional):
Referencia URI que indica la dirección a la que se debe enviar la respuesta. Es útil para prevenir envíos de respuestas maliciosas a destinatarios involuntarios, protección que es requerida por algunos bindings de protocolos. Si está presente, el destinatario actual debe comprobar que la referencia URI identifica a la localización en la que fue recibido el mensaje. Si no es así, la respuesta debe ser deshechada.

- Consent (Opcional):
Indica si se ha obtenido o no consentimiento (y bajo qué condiciones) de un principal al mandar la respuesta. Si no se da ningún valor, se toma el del identificador urn:oasis:names:tc:SAML:2.0:consent:unspecified

- <saml:Issuer> (Opcional):
Identifica la entidad que generó el mensaje respuesta.

Atributo	*Descripción*	*Uso*
ID	ID de la respuesta	Obligatorio
InResponseTo	ID de la petición que generó la respuesta	Opcional
Version	Versión de la respuesta	Obligatorio
IssueInstant	Instante de envío de la respuesta	Obligatorio
Destination	Dirección a la que debemos enviar la respuesta	Opcional
Consent	Indica si ha habido consentimiento o no	Opcional

Tabla 9. 15: Atributos del tipo complejo **StatusResponseType**

Este elemento contiene también los siguientes elementos:

- <ds:Signature> (Opcional):
Es un XML Signature que autentifica al respondedor y proporciona integridad en el mensaje.

- <Extensions> (Opcional):
Este punto contiene elementos de extensión opcionales que son convenidos entre las partes que se comunican. No se requiere ningún esquema para hacer uso de este punto de extensión.

- <Status> (Obligatorio):
Es un código que representa el estado de la petición correspondiente.

También podemos listar todos los elementos de este tipo complejo:

Elemento	*Descripción*	*Uso*
<Issuer>	Entidad que generó la respuesta	Opcional
<ds:Signature>	Firma que autentifica al respondedor	Opcional
<Extensions>	Elementos de extensión del esquema	Opcional
<Status>	Estado de la petición correspondiente	Obligatorio

Tabla 9. 16: Elementos del tipo complejo **StatusResponseType**

Ponemos el esquema de definición de este tipo para tener una idea general de qué elementos consta.

```
<complexType name="StatusResponseType">
        <sequence>
                <element ref="saml:Issuer" minOccurs="0"/>
                <element ref="ds:Signature" minOccurs="0"/>
                <element ref="samlp:Extensions" minOccurs="0"/>
                <element ref="samlp:Status"/>
        </sequence>
        <attribute name="ID" type="ID" use="required"/>
        <attribute name="InResponseTo" type="NCName" use="optional"/>
        <attribute name="Version" type="string" use="required"/>
        <attribute name="IssueInstant" type="dateTime" use="required"/>
        <attribute name="Destination" type="anyURI" use="optional"/>
        <attribute name="Consent" type="anyURI" use="optional"/>
</complexType>
```

Figura 9. 32: Definición del tipo **StatusResponseType**

Al igual que explicamos en la sección anterior, si queremos utilizar correctamente este tipo, debemos seguir unas **normas de uso**:

- Dependiendo de los requerimientos particulares de protocolos y perfiles, un respondedor SAML puede necesitar autentificarse, y puede requerirse integridad de mensajes. Autentificación e integridad de mensajes la proporciona los binding de protocolos. La respuesta SAML puede ser firmada, proporcionando tanto autentificación del respondedor como integridad del mensaje.

- Si se usa una firma, entonces debe aparecer el elemento **<ds:Signature>** y el solicitante SAML debe verificar la firma (esto es, que el mensaje no ha sido alterado). Si no es válida la firma, entonces el solicitante no debe fiarse de los contenidos de la respuesta y debe responder con un error. Si es válida, el solicitante debe evaluarla para determinar la identidad y la decencia del que firma y puede continuar procesando la respuesta como considere oportuno.

- Si se incluye un atributo **Consent** y su valor indica que se ha obtenido de alguna forma el consentimiento del principal, entonces se debe firmar la respuesta.

Elemento <Status>

El elemento <Status> pertenece al tipo **StatusResponseType**. Es decir, aparecerá en todas las respuestas que se generen. En el siguiente esquema se define este elemento y su correspondiente tipo complejo **StatusType**:

```
<element name="Status" type="samlp:StatusType"/>
<complexType name="StatusType">
        <sequence>
                <element ref="samlp:StatusCode"/>
                <element ref="samlp:StatusMessage" minOccurs="0"/>
                <element ref="samlp:StatusDetail" minOccurs="0"/>
        </sequence>
</complexType>
```

Figura 9. 33: Definición del tipo **StatusType**

El elemento <Status> contiene los siguientes elementos:

- <StatusCode> (Obligatorio):

Es un código que representa el estado de la actividad llevada a cabo en la respuesta de la correspondiente petición.

- <StatusMessage> (Opcional):

Es un mensaje que debe ser devuelto al operador.

- <StatusDetail> (Opcional):

Información adicional concerniente al estado de la solicitud.

De modo explicativo, ponemos los elementos que contiene el elemento <Status> en una tabla:

Elemento	*Descripción*	*Uso*
<StatusCode>	Código de la respuesta	Obligatorio
<StatusMessage>	Mensaje devuelto al operador	Opcional
<StatusDetail>	Información adicional al estado	Opcional

Tabla 9. 17: Elementos del elemento <**Status**>

Pasamos a describir cada uno de estos elementos.

Elemento <StatusCode>

Para ganar en claridad haremos un repaso de dónde estamos. El elemento que detallamos a continuación pertenece a otro elemento llamado <Status>. Éste, a su vez, pertenece al tipo complejo **StatusResponseType**. Dicho de otra manera y viendo que este elemento es obligatorio, podemos decir que todas las respuestas que creemos que se ajusten a la especificación SAML deberán contener un elemento de tipo <StatusCode>. Este elemento especifica un código o un conjunto de códigos anidados que representa el

estado de la correspondiente petición. Definimos este elemento y su tipo complejo **StatusCodeType** a continuación:

```
<element name="StatusCode" type="samlp:StatusCodeType"/>
<complexType name="StatusCodeType">
        <sequence>
                <element ref="samlp:StatusCode" minOccurs="0"/>
        </sequence>
        <attribute name="Value" type="anyURI" use="required"/>
</complexType>
```

Figura 9. 34: Definición del elemento **StatusCode**

Tiene los siguientes elementos y atributos:

- Value (Obligatorio):

Es el valor del código de estado. Este atributo contiene una referencia URI. El valor más alto del elemento <StatusCode> debe estar en el lugar más alto de la lista proporcionada en esta sección.

- <StatusCode> (Opcional):

Es un código de estado secundario que proporciona información adicional específica sobre una condición de error. Los respondedores pueden omitir los códigos de estado secundario para prevenir ataques que intenten investigar para encontrar información con la intención de mandar peticiones erróneas de forma intencionada.

Se permiten los siguientes **valores más altos para <StatusCode>**. Estos valores se tienen que poner dejando de manera opcional aquellos de segundo nivel que ponemos un poco más abajo.

- urn:oasis:names:tc:SAML:2.0:status:**Success**

  La petición tuvo éxito. Se puede devolver información adicional en el elemento <StatusMessage> y/o <StatusDetail>.

- urn:oasis:names:tc:SAML:2.0:status:**Requester**

  La petición no se pudo llevar a cabo debido a un error en la parte del solicitante.

- urn:oasis:names:tc:SAML:2.0:status:**Responder**

  La petición no se pudo llevar a cabo debido a un error en la parte del respondedor o de la autoridad SAML.

- urn:oasis:names:tc:SAML:2.0:status:**VersionMismatch**

  El respondedor no pudo procesar la petición porque la versión del mensaje de solicitud era incorrecta.

En la especificación se referencian los siguientes **códigos de estado de segundo nivel**. Se pueden definir más de ellos en futuras versiones de SAML. Las entidades de sistema son libres para definir más códigos de estado específicos definiendo las apropiadas referencias URI:

- urn:oasis:names:tc:SAML:2.0:status:AuthnFailed

  El proveedor que responde no fue capaz de autentificar al principal con éxito.

- urn:oasis:names:tc:SAML:2.0:status:InvalidAttrNameOrValue

Contenido no esperado o no válido encontrado dentro del elemento <saml:Attribute> o <saml:AttributeValue>.

- urn:oasis:names:tc:SAML:2.0:status:InvalidNameIDPolicy

El proveedor que responde no puede o no podrá apoyar la política de identificador de nombre solicitada.

- urn:oasis:names:tc:SAML:2.0:status:NoAuthnContext

Los requerimientos del contexto de autentificación específicos no pueden ser cumplidos por el respondedor.

- urn:oasis:names:tc:SAML:2.0:status:NoAvailableIDP

Usado por un intermediario para indicar que ninguno de los proveedores de identidad proporcionados por los elementos <Loc> de la <IDPList> pueden ser resueltos o que ninguno de dichos proveedores están disponibles.

- urn:oasis:names:tc:SAML:2.0:status:NoPassive

Indica que el proveedor de identidad que responde no puede autentificar al principal de manera pasiva, como se solicitó.

- urn:oasis:names:tc:SAML:2.0:status:NoSupportedIDP

Usado por un intermediario para indicar que ninguno de los proveedores de identidad de la <IDPList> son aprobados por el intermediario.

- urn:oasis:names:tc:SAML:2.0:status:PartialLogout

Usado por una autoridad de sesión para indicar a los participantes de ésta que no es posible propagar la desconexión a todos los demás participantes.

- urn:oasis:names:tc:SAML:2.0:status:ProxyCountExceeded

Indica que un proveedor que responde no puede autentificar al principal de manera directa y que no se le permite hacer más de proxy con la petición.

- urn:oasis:names:tc:SAML:2.0:status:RequestDenied

El respondedor o la autoridad SAML puede procesar la petición pero ha elegido no responderla. Este código de estado puede ser usado cuando hay preocupación acerca del contexto de seguridad del mensaje o secuencia de mensajes de petición recibidos de un solicitante particular.

- urn:oasis:names:tc:SAML:2.0:status:RequestUnsupported

El respondedor o la autoridad SAML no puede apoyar la petición.

- urn:oasis:names:tc:SAML:2.0:status:RequestVersionDeprecated

El respondedor no puede procesar ninguna petición con la versión de protocolo especificado en ella.

- urn:oasis:names:tc:SAML:2.0:status:RequestVersionTooHigh

El respondedor no puede procesar la petición porque la versión del protocolo especificada en ella es mayor que la soportada por él.

- urn:oasis:names:tc:SAML:2.0:status:RequestVersionTooLow

El respondedor no puede procesar la petición porque la versión del protocolo especificada en ella es demasiado pequeña.

- urn:oasis:names:tc:SAML:2.0:status:ResourceNotRecognized

El valor del recurso proporcionado en la petición es irreconocible o no válido.

- urn:oasis:names:tc:SAML:2.0:status:TooManyResponses

El mensaje respuesta contiene más elementos de los que es capaz de devolver el respondedor.

- urn:oasis:names:tc:SAML:2.0:status:UnknownAttrProfile

Un atributo de un perfil de atributo particular ha sido presentado a una entidad que no tiene conocimiento de ese perfil.

- urn:oasis:names:tc:SAML:2.0:status:UnknownPrincipal

El proveedor que responde no reconoce al principal determinado o implicado por la petición.

- urn:oasis:names:tc:SAML:2.0:status:UnsupportedBinding

El respondedor no puede rellenar la respuesta adecuadamente usando el binding de protocolo indicado en la petición.

Elemento <StatusMessage>

Este elemento, junto con <StatusCode> y <StatusDetail>, pertenece al elemento <Status>. Es de carácter opcional por lo que puede que no aparezca en algunas respuestas. Este elemento determina el mensaje que puede ser devuelto a un operador. Su esquema se define a continuación:

```
<element name="StatusMessage" type="string"/>
```

Figura 9. 35: Definición del elemento **StatusMessage**

Elemento <StatusDetail>

Puede ser usado para determinar información adicional relativa al estado de la petición. Dicha información consiste en cero o más elementos de cualquier espacio de nombres, sin requerimientos para presentar esquemas o por validación de esquemas de los contenidos de este elemento.

El siguiente fragmento define al elemento y a su tipo complejo **StatusDetailType**:

```
<element name="StatusDetail" type="samlp:StatusDetailType"/>
<complexType name="StatusDetailType">
        <sequence>
                <any namespace="##any" processContents="lax" minOccurs="0"
maxOccurs="unbounded"/>
        </sequence>
</complexType>
```

Figura 9. 36: Definición del elemento **StatusDetail**

9.3.3.- Protocolo de petición de Autentificación

Para entender mejor la forma de este capitulo, hemos preferido explicar su estructura por adelantado con el fin de hacernos una idea global. De este modo primero daremos unas indicaciones sobre la funcionalidad de este protocolo y definiremos algunos conceptos de utilidad. Seguidamente, pasaremos a explicar el elemento <AuthnRequest> con todos los subelementos que posee. Así ya sabremos los elementos que intervienen en los mensajes de este protocolo. Después tendremos que definir qué hacemos cuando nos llegue uno de estos mensajes y cuando tengamos que crearlos y mandarlos. Se hará en los apartados reglas de procesado y uso de los mensajes <AuthnRequest> y <Response>. Para terminar incluimos un ejemplo en el que podremos comprobar cómo están constituidos cada uno de estos mensajes.

Se usa este protocolo cuando un principal (o una agente actuando en el nombre del mismo) desea obtener aserciones que contengan declaraciones de autentificación para establecer un contexto de seguridad en una o más entidades que deseen tenerlo. Por medio de este protocolo es posible enviar un elemento de mensaje <AuthnRequest> hacia una autoridad SAML y pedirle que devuelva un mensaje <Response> que contenga una o más aserciones. Éstas pueden contener declaraciones adicionales de cualquier tipo pero al menos una de las aserciones debe contener una declaración de autentificación. A las autoridades SAML que soportan este protocolo también se les llama proveedor de identidad (IP).

Aparte de este requerimiento, el contenido específico de las aserciones devueltas depende del perfil o contexto de uso. También no se especifican los medios exactos por los que un proveedor de identidad autentifica a un principal o a un agente, aunque la forma de hacerlo pudiera cambiar el contenido de la respuesta. Quedan fuera del alcance de este protocolo otras emisiones relacionadas con la validación de credenciales de autentificación o cualquier comunicación entre el IP y alguna otra entidad implicada en los procesos de autentificación.

Pasamos a describir las diferentes partes que intervienen en este protocolo:

- **Solicitante**:

Es la entidad que crea la petición de autentificación y a la que se le devuelve la respuesta.

- **Presentador**:

Es la entidad que presenta la petición al proveedor de identidad y bien se autentifica durante la transmisión del mensaje, bien confía en un contexto de seguridad existente para establecer su identidad. Si no es el solicitante, el presentador actúa como intermediario entre el solicitante y el proveedor de identidad que responde.

- **Sujeto solicitado**:

Es la entidad sobre la que se solicitan una o más aserciones.

- **Partes que dan testimonio**:

Son aquella o aquellas que se espera que sean capaces de satisfacer uno de los elementos <SubjectConfirmation> de las aserciones resultantes.

- **Partes que confían**:

Es la entidad o entidades que se suponen que van a consumir la aserción para cumplir un propósito determinado por el perfil o el contexto de uso, generalmente para establecer un contexto de seguridad.

- **Proveedor de Identidad (IP)**:

Es aquella a la que el presentador entrega la petición y de la que el mismo recibe la respuesta.

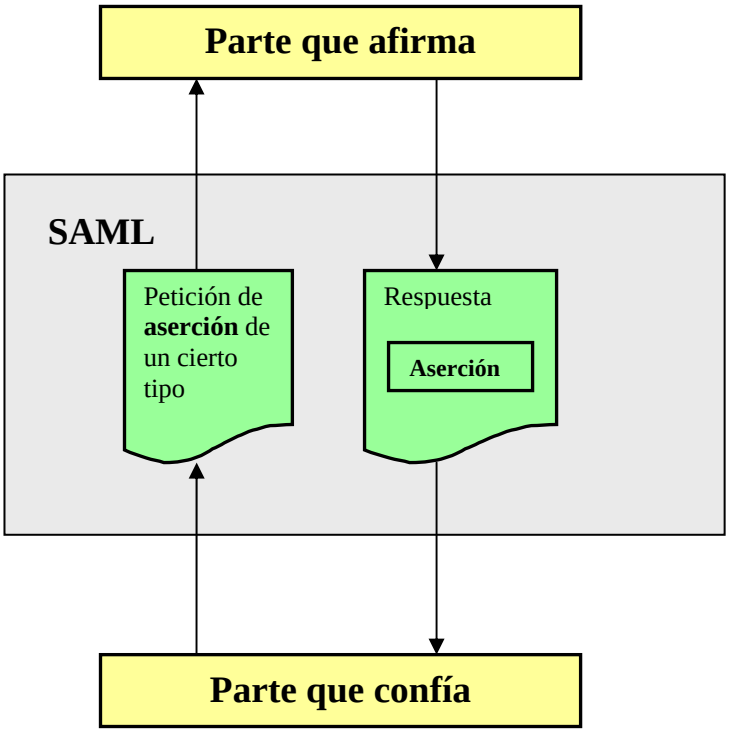

Figura 9. 37: Relación entre las partes que afirman y confían

a.- Elemento <AuthnRequest>

Para pedir que un proveedor de identidad mande una aserción con una declaración de autentificación, un presentador tiene que autentificarse en ese IP (o confiar en un contexto de seguridad existente) y enviar un mensaje <AuthnRequest> que describa las propiedades que la aserción resultante debe tener para satisfacer su propósito. Entre estas propiedades puede estar información relacionada con el contenido de la aserción y/o información relacionada con cómo el mensaje <Response> resultante debiera ser entregado al solicitante. El proceso de autentificación del presentador puede tener lugar antes, durante o después de la entrega del mensaje <AuthnRequest>.

El solicitante podría no ser el mismo que el presentador de la petición si, por ejemplo, el solicitante es una parte que confía que intenta usar una aserción resultante para autentificar o autorizar al sujeto requerido con el fin de poder decidir si proporciona un servicio.

El mensaje <AuthnRequest> debería ser firmado o de lo contrario autentificado y protegido en su integridad por el protocolo/binding usado para entregar el mensaje.

Este mensaje contiene el tipo complejo **AuthnRequestType**, el cual extiende de RequestAbstractType. El siguiente fragmento de esquema define el elemento <AuthnRequest> y su tipo complejo **AuthnRequestType**.

```
<element name="AuthnRequest" type="samlp:AuthnRequestType"/>
<complexType name="AuthnRequestType">
        <complexContent>
                <extension base="samlp:RequestAbstractType">
                        <sequence>
                                <element ref="saml:Subject" minOccurs="0"/>
                                <element ref="samlp:NameIDPolicy" minOccurs="0"/>
                                <element ref="saml:Conditions" minOccurs="0"/>
                                <element ref="samlp:RequestedAuthnContext" minOccurs="0"/>
                                <element ref="samlp:Scoping" minOccurs="0"/>
                        </sequence>
                        <attribute name="ForceAuthn" type="boolean" use="optional"/>
                        <attribute name="IsPassive" type="boolean" use="optional"/>
                        <attribute name="ProtocolBinding" type="anyURI" use="optional"/>
                        <attribute name="AssertionConsumerServiceIndex" type="unsignedShort"
                        use="optional"/>
                        <attribute name="AssertionConsumerServiceURL" type="anyURI"
                        use="optional"/>
                        <attribute name="AttributeConsumingServiceIndex"
                        type="unsignedShort" use="optional"/>
                        <attribute name="ProviderName" type="string" use="optional"/>
                </extension>
        </complexContent>
</complexType>
```

Figura 9. 38: Descripción del elemento <**AuthnRequest**>

Este tipo añade los siguientes elementos y atributos (en general son opcionales, pero pueden ser requeridos por algún perfil específico):

- <saml:Subject> (Opcional)

Especifica el sujeto requerido en la aserción resultante. Puede incluir uno o más elementos <saml:SubjectConfirmation> para indicar cómo y/o por quién fue confirmada la aserción.

Si es omitido por completo o si no se incluye ningún identificador, se da por supuesto que el presentador del mensaje es el sujeto solicitado. Si no se incluyen los elementos <saml:SubjectConfirmation>, se da por supuesto que el presentador es la única entidad que da testimonio y el método está implícito en el perfil de uso y/o en las políticas del proveedor de identidad.

- <NameIDPolicy> (Opcional)

Especifica limitaciones en el identificador de nombre que se usa para el sujeto solicitado. Si se omite, entonces se puede usar cualquier identificador proporcionado por el proveedor de identidad para el sujeto solicitado, restringido por cualquier política específica como por ejemplo las que se refieren a privacidad.

- <saml:Conditions> (Opcional)

Especifica las condiciones SAML que el solicitante espera para limitar la validez y/o el uso de la aserción resultante. El respondedor puede modificarlas o complementarlas como estime oportuno. La información de este elemento se usa como entrada en el proceso de construcción de la aserción, antes que como condición en el uso de la propia petición.

- <RequestedAuthnContext> (Opcional)

Especifica los requisitos, si los hay, que el solicitante pone sobre el contexto de autentificación que se aplica a la autentificación del presentador de la entidad que responde.

- <Scoping> (Opcional)

Especifica un conjunto de proveedores de identidad en los que confía el solicitante para autentificar al presentador, además de limitaciones y contexto relacionados por el respondedor, con proxying del mensaje <AuthnRequest> a los proveedores de identidad.

A modo de resumen ponemos en una tabla los elementos que contiene.

Elemento	*Descripción*	*Uso*
<Subject>	Sujeto de la aserción resultante	Opcional
<NameIDPolicy>	Limitaciones en id de nombre del sujeto	Opcional
<Conditions>	Condiciones que limitan la aserción	Opcional
<RequestedAuthnContext>	Requisitos sobre contexto autentificación	Opcional
<Scoping>	IPs para autentificar al presentador	Opcional

Tabla 9. 18: Elementos del elemento **<AuthnRequest>**

- ForceAuthn (Opcional)

Es un valor booleano. Si vale “true”, el proveedor de identidad debe autentificar al presentador directamente antes que confiar en un contexto de seguridad previo. Si no se proporciona un valor, por defecto sería “false”. No obstante, si tanto ForceAuthn como IsPassive son “true”, el proveedor de identidad no debe autentificar al presentador a menos que las limitaciones de IsPassive lo permitan.

- IsPassive (Opcional)

Es un valor booleano. Si es “true”, el proveedor de identidad y el propio agente de usuario no deben tomar el control del interfaz de usuario desde el solicitante ni interactuar con el presentador de manera notable. Su valor por defecto es “false”.

- AssertionConsumerServiceIndex (Opcional)

Identifica, indirectamente, la localización a la que hay que devolver el mensaje <Response> al solicitante. Se aplica sólo en perfiles en los que el solicitante es distinto que el presentador. El proveedor de identidad debe tener un medio de confianza para mapear el valor del índice en el atributo en una localización asociada con el solicitante. Si se omite, entonces el proveedor de identidad debe devolver el mensaje <Response> a la localización por defecto asociada con el solicitante por el perfil de uso. Si el índice especificado no es válido, el proveedor de identidad puede devolver un <Response> error o puede usar la localización por defecto. Este atributo, AssertionConsumerServiceURL y el atributo ProtocolBinding son mutuamente exclusivos.

- AssertionConsumerServiceURL (Opcional)

Especifica el valor de la localización a la que se debe devolver el mensaje <Response> al solicitante. El respondedor debe garantizar de algún modo que el valor

especificado está asociado al solicitante. Uno de estos métodos puede ser firmar el mensaje <AuthnRequest>. Este atributo es mutuamente exclusivo con AssertionConsumerServiceIndex y suele venir acompañado del atributo ProtocolBinding.

- ProtocolBinding (Opcional)

Es una referencia URI que identifica el perfil de protocolo SAML usado cuando devolvamos el mensaje <Resonse>. Este atributo es mutuamente exclusivo con AssertionConsumerServiceIndex y viene típicamente acompañado del atributo AssertionConsumerServiceURL.

- AttributeConsumingServiceIndex (Opcional)

Indirectamente identifica información asociada con el solicitante describiendo los atributos SAML que el solicitante requiere o desea que le sea proporcionado por el proveedor de identidad en el mensaje <Response>. El proveedor de identidad debe tener un medio de confianza para mapear el valor del índice en el atributo en información asociada con el solicitante. Dicho proveedor puede usar esta información para rellenar uno o más de los elementos <saml:AttributeStatement> en la aserción que devuelve.

- ProviderName (Opcional)

Especifica un nombre del solicitante legible por los humanos para usarlo por el agente de usuario o el proveedor de identidad.

Ponemos en una tabla todos los atributos.

Atributo	*Descripción*	*Uso*
ForceAuthn	Fuerza una nueva autentificación	Opcional
IsPassive	Toma control del interfaz de usuario	Opcional
AssertionConsumerServiceIndex	Indica lugar al que mandar la aserción	Opcional
AssertionConsumerServiceURL	Indica lugar al que mandar la aserción	Opcional
ProtocolBinding	Identifica el perfil de protocolo usado	Opcional
AttributeConsumingServiceIndex	Información sobre atributos solicitante	Opcional
ProviderName	Nombre solicitante	Opcional

Tabla 9. 19: Atributos del elemento **<AuthnRequest>**

Dentro de este elemento aparecen unos cuentos elementos. Algunos de ellos ya lo hemos explicado en anteriores apartados. Por eso explicaremos a continuación los elementos <NameIDPolicy> y <Scoping>.

Elemento <NameIDPolicy>

El elemento <NameIDPolicy> adapta el identificador de nombre de sujetos en las aserciones que resultan de un <AuthnRequest>. El siguiente fragmento de esquema define el elemento <NameIDPolicy> y su tipo complejo **NameIDPolicyType**:

```
<element name="NameIDPolicy" type="samlp:NameIDPolicyType"/>
<complexType name="NameIDPolicyType">
        <attribute name="Format" type="anyURI" use="optional"/>
        <attribute name="SPNameQualifier" type="string" use="optional"/>
        <attribute name="AllowCreate" type="boolean" use="optional"/>
</complexType>
```

Figura 9. 39: Descripción del elemento <**NameIdPolicy**>

Su tipo complejo **NameIDPolicyType** define los siguientes atributos:

- Format (Opcional)

Especifica la referencia URI correspondiente al formato del identificador de nombre definido en ésta u otra especificación. Se define un valor adicional que se usa dentro de este atributo para indicar la encriptación del identificador resultante.

```
urn:oasis:names:tc:SAML:2.0:nameid-format:encrypted
```

Figura 9. 40: Valor del atributo **Format**

- SPNameQualifier (Opcional)

Opcionalmente especifica que se devuelve (o se crea) el identificador de sujeto de la aserción en el espacio de nombres de un proveedor de servicio excepto el solicitante, o en el espacio de nombres de una afiliación de proveedores de servicio.

- AllowCreate (Opcional)

Es un valor booleano usado para indicar si se le permite al proveedor de identidad, en el curso de rellenar la petición, crear un nuevo identificador que represente al principal. Su valor por defecto es “false”, que significa que el solicitante limita al IP que le mande una sola aserción si ya se ha establecido un identificador aceptable para el principal.

Asociado al **uso de estos atributos** aparecen unas determinadas reglas que hay que seguir:

- Hay que hacer notar que el usar **AllowCreate**, no impide que el IP cree identificadores fuera del contexto de esta petición específica (por ejemplo, con antelación para un gran número de identificadores).

- Cuando se usa este atributo (**AllowCreate**), si no se entiende el contenido o no es aceptable, entonces se debe devolver, dentro de un mensaje <Response>, el elemento con un error <Status> pudiendo contener un segundo nivel de error <StatusCode> de valor urn:oasis:names:tc:SAML:2.0:status:InvalidNameIDPolicy.

- Si el valor de **Format** es omitido o establecido a urn:oasis:names:tc:SAML:2.0:nameidformat:unspecified, entonces el proveedor de identidad puede devolver cualquier tipo de identificador, sujeto a limitaciones adicionales debidas al contenido de este elemento o a las políticas del IP o del principal.

- El valor de **Format** especial urn:oasis:names:tc:SAML:2.0:nameid-format:encrypted indica que la aserción resultante debe contener elementos <EncryptedID> en lugar de texto plano. La forma del identificador de nombres sin encriptar puede ser de cualquier tipo soportado por el proveedor de identidad para el sujeto solicitado.

- Sin tener el cuenta el atributo **Format de <NameIDPolicy>**, el proveedor de identidad puede devolver un <EncryptedID> en el sujeto de la aserción resultante si las políticas del IP (posiblemente especificadas por el SP) requieren el uso de un identificador encriptado.

- Notamos que si el solicitante desea permitir que el IP establezca un nuevo identificador para el principal si no existía, debe incluir este elemento con el atributo **AllowCreate** establecido a "true". De lo contrario, sólo un principal del que previamente tenía establecido un identificador usable por el solicitante puede ser autentificado con éxito. Esto es principalmente útil en unión con el valor de Format urn:oasis:names:tc:SAML:2.0:nameid-format:persistent.

Atributo	*Descripción*	*Uso*
Format	Formato del identificador nombres	Opcional
SPNameQualifier	Id. sujeto en espacio nombre SP	Opcional
AllowCreate	Indica IP puede crear un identificador	Opcional

Tabla 9. 20: Atributos del elemento <**NameIDPolicy**>

Elemento <Scoping>

Este elemento pertenece al elemento <AuthnRequest> y junto con el anterior son los que no habíamos explicado con anterioridad.

El elemento <Scoping> especifica los proveedores de identidad en los que confía el solicitante para que se autentifique al presentador, además de contexto y limitaciones relacionados por el respondedor, con proxying del mensaje <AuthnRequest> a los proveedores de identidad. El siguiente fragmento de esquema define el elemento <Scoping> y su tipo complejo:

```
<element name="Scoping" type="samlp:ScopingType"/>
<complexType name="ScopingType">
        <sequence>
                <element ref="samlp:IDPList" minOccurs="0"/>
                <element ref="samlp:RequesterID" minOccurs="0"
maxOccurs="unbounded"/>
        </sequence>
        <attribute name="ProxyCount" type="nonNegativeInteger" use="optional"/>
</complexType>
<element name="RequesterID" type="anyURI"/>
```

Figura 9. 41: Descripción del elemento <**Scoping**>

El tipo complejo **ScopingType** define los siguientes elementos y atributos:

- ProxyCount (Opcional)

Especifica el número de saltos proxying permitidos entre el proveedor de identidad que recibe este <AuthnRequest> y el proveedor de identidad que autentifica en última instancia al principal. Una cuenta de cero no permite ningún proxying, mientras que omitir este atributo no expresa ninguna restricción de éstas.

<IDPList> (Opcional)

Es una lista consultiva de proveedores de identidad e información asociada que el solicitante estima aceptable para responder la solicitud.

<RequesterID> (Cero o más)

Identifica un conjunto de entidades solicitantes en cuyo nombre actúa el solicitante. Se usa para comunicar la cadena de solicitantes cuando se da el proxying.

Atributo / Elemento	*Descripción*	*Uso*
ProxyCount	Saltos proxying permitidos	Opcional
<IDPList>	Lista consultiva de IP	Opcional
<RequesterID>	Entidades en cuyo nombre actúa el solicitante	Cero o más

Tabla 9. 21: Atributos y elementos del elemento <**Scoping**>

Elemento <IDPList>

Este elemento esta contenido dentro del elemento <Scoping> que a su vez pertenece al elemento <AuthnRequest>. El elemento <IDPList> especifica los proveedores de identidad en los que confía el solicitante para autentificar al presentador. El próximo fragmento de esquema define el elemento <IDPList> y su tipo complejo **IDPListType**:

```
<element name="IDPList" type="samlp:IDPListType"/>
<complexType name="IDPListType">
        <sequence>
                <element ref="samlp:IDPEntry" maxOccurs="unbounded"/>
                <element ref="samlp:GetComplete" minOccurs="0"/>
        </sequence>
</complexType>
<element name="GetComplete" type="anyURI"/>
```

Figura 9. 42: Descripción del elemento <**IDPList**>

Su tipo complejo **IDPListType** define los siguientes elementos:

<IDPEntry> (Uno o más)

Información sobre un solo proveedor de identidad.

<GetComplete> (Opcional)

Si no está completo el <IDPList>, usando este elemento especificamos la referencia URI que puede ser usada para recuperar la lista completa. Este recurso asociado a la URI debe resultar ser una instancia XML cuyo elemento raíz es <IDPList> que no contiene el elemento <GetComplete>.

Elemento	*Descripción*	*Uso*
<IDPEntry>	Información sobre un IP	Uno o más
<GetComplete>	URI donde se puede recuperar la lista de IP	Opcional

Tabla 9. 22: Atributos del elemento **<IDPList>**

Elemento <IDPEntry>

Para entender dónde se coloca este elemento hay que decir que es un subelemento del elemento <IDPList> y este a su vez de <Scoping>. Como ya se comentó este último pertenece al elemento <AuthnRequest>. El elemento <IDPEntry> especifica un proveedor de identidad en el que confía el solicitante para autenticar al presentador.

El siguiente fragmento de esquema define el elemento <IDPEntry> y su tipo complejo **IDPEntryType**:

```
<element name="IDPEntry" type="samlp:IDPEntryType"/>
<complexType name="IDPEntryType">
        <attribute name="ProviderID" type="anyURI" use="required"/>
        <attribute name="Name" type="string" use="optional"/>
        <attribute name="Loc" type="anyURI" use="optional"/>
</complexType>
```

Figura 9. 43: Descripción del elemento **<IDPEntry>**

Su tipo complejo **IDPEntryType** define los siguientes atributos:

- ProviderID (Requerido)

Es el identificador único del proveedor de identidad.

- Name (Opcional)

Es un nombre legible para los humanos del proveedor de identidad.

- Loc (Opcional)

Es una referencia URI que representa la localización de un punto final específico del perfil que soporta el protocolo de petición de autentificación. El binding que se use deberá ser entendido desde el perfil de uso.

Los atributos los colocamos en la tabla siguiente.

Atributo	*Descripción*	*Uso*
ProviderID	Identificador del IP	Obligatorio
Name	Nombre del IP	Opcional
Loc	Localización del punto final	Opcional

Tabla 9. 23: Atributos del elemento **<IDPList>**

b.- Reglas de procesado de los elementos <AuthnRequest> y <Response>

En este apartado vamos a describir una serie de reglas generales que debemos cumplir cuando usemos algunos de estos elementos. Primero daremos las reglas relativas al elemento <AuthnRequest> y en otro apartado las del elemento <Response>.Se ha elegido esta estructura para diferenciar, dentro de todo lo posible, el funcionamiento que se debe seguir cuando recibimos alguno de estos mensajes. Es decir, cuando nos llega un mensaje que contiene uno de estos elementos.

Reglas de procesado del mensaje <AuthnRequest>

El intercambio de **<AuthnRequest>** y **<Response>** soporta una variedad de escenarios de uso y por ello debe ser perfilado mediante contextos específicos en los que se limita esta variedad mediante el uso o prohibición de los distintos input y output. Las siguientes reglas de procesado se aplican como invariantes aún a pesar del perfil elegido. Se deben cumplir también todas las demás reglas asociadas a la petición y respuesta subyacentes.

El respondedor deben finalmente contestar a un <AuthnRequest> con un mensaje <Response> que contenga una o más aserciones que cumplan con las especificaciones definidas por la petición, o con un mensaje <Response> que contenga un **<Status>** que describa el error ocurrido. El respondedor puede dirigir intercambios de mensajes adicionales con el presentador como estime necesario para iniciar o completar el proceso de autentificación, sujeto a la naturaleza del binding del protocolo y al mecanismo de autentificación.

Si el respondedor no es capaz de autentificar al presentador o no reconoce al sujeto solicitado, o si por políticas en curso del IP se le impide proporcionar una aserción, entonces debe devolver un <Response> con un <Status> error y puede devolver un segundo nivel de error <StatusCode> de valor urn:oasis:names:tc:SAML:2.0:status:AuthnFailed o urn:oasis:names:tc:SAML:2.0:status:UnknownPrincipal.

Si se presenta el elemento **<saml:Subject>** de la petición, entonces debe coincidir con el <saml:Subject> de la aserción resultante, excepto si el identificador puede venir en un formato diferente como especifica <NameIDPolicy>. En tal caso, el contenido físico del identificador puede ser diferente pero debe referirse al mismo principal. Todos los contenidos definidos específicamente dentro de <AuthnRequest> son opcionales, aunque algunos pueden ser requeridos por ciertos perfiles. **En ausencia de cualquier tipo de contenido específico de <AuthnRequest>**, se aplica el siguiente comportamiento:

- La aserción (o aserciones) devuelta debe contener un elemento <saml:Subject> que representa al presentador. El tipo de identificador y el formato son determinados por el proveedor de identidad. Al menos una de las aserciones debe contener al menos un **<saml:AuthnStatement>** que describa la autentificación efectuada por el respondedor o el servicio de autentificación asociado.

- La petición del presentador debe, en la extensión posible, ser la única parte que da testimonio capaz de satisfacer **<saml:SubjectConfirmation>** de la aserción. En el caso de métodos de confirmación más débiles, se deben usar mecanismos específicos del perfil u otros para satisfacer este requerimiento.

- La aserción (o aserciones) resultante debe contener un elemento **<saml:AudienceRestriction>** referenciando al solicitante como una parte aceptable que confía. Se pueden incluir otras audiencias como lo estime apropiado el proveedor de identidad.

Reglas de procesado del mensaje <Response>

Al igual que se hizo con el mensaje <AuthnRequest>, damos unas reglas generales que debemos seguir cuando tengamos que procesar un mensaje <Response>. A pesar del binding SAML usado, el proveedor de servicio debe seguir los siguientes pasos:

- Verificar cualquier **firma** presente en la aserción o en la respuesta.

- Debe, dentro de cualquier portador **<SubjectConfirmationData>**, verificar los siguientes atributos:
  - Recipient: debe emparejar la aserción al servicio de consumo URL al que se entregó el artefacto o <Response>.
  - NotOnOrAfter: comprobar que no se ha cumplido. Depende de un reloj permitido entre proveedores.
  - InResponseTo: comprobar que es igual a ID del mensaje original <AuthnRequest>, a menos que la respuesta sea sin solicitar, en cuyo caso no debe aparecer este atributo.
  - Address: el SP puede comparar este campo con la dirección del cliente del agente de usuario.

- Comprobar que cualquier aserción en la que se confía, es válida en otros campos.

- Cualquier aserción no válida o que no se conozcan los requerimientos para la confirmación del sujeto, debe ser eliminada, no siendo usada para establecer un contexto de seguridad para el director.

- Si se usa <AuthnStatement> con un atributo **SessionNotOnOrAfter** para establecer un contexto de seguridad para el principal, dicho contexto debe ser eliminado cuando se alcance el tiempo a menos que el SP reestablezca la identidad del director repitiendo el uso del perfil.

c.- Uso de los mensajes <AuthnRequest> y <Response>

En esta sección damos unas directivas de lo que tendremos que incluir en los mensajes <AuthnRequest> y <Response> cuando tengamos que usarlos.

Uso del mensaje <AuthnRequest>

Un proveedor de servicio puede incluir cualquier contenido de mensaje descrito en la especificación, siguiendo todas las reglas de proceso definidas.

- El elemento **<Issuer>** deber estar presente y contener un identificador único del proveedor de servicio solicitante; el atributo Format se debe omitir o tener un valor de *urn:oasis:names:tc:SAML:2.0:nameid-format:entity*.

- Si el proveedor de identidad no puede satisfacer la petición, debe responder con un mensaje <Response> que contenga un **código de estado de error** apropiado.

- Si el proveedor de servicio desea permitir al IP que establezca un nuevo identificador para el principal si no existiese, debe incluir un elemento **<NameIDPolicy>** con el atributo AllowCreate establecido a “true”. Si no, sólo un principal al que el proveedor de identidad haya establecido previamente un identificador válido usable por el SP, puede ser autentificado con éxito.

- Hacemos notar que el SP puede incluir un elemento <Subject> en la petición que nombre a la identidad actual sobre la que se desea obtener la aserción. Este elemento no debe contener elementos **<SubjectConfirmation>**. Si el IP no reconoce al principal como esa identidad, entonces debe responder con un mensaje <Response> que contenga un estado de error y ninguna aserción.

- El mensaje <AuthnRequest> debe ser **firmado** de la forma que lo indique el perfil de uso. Si dicho mensaje no se ha autentificado o protegido su identidad, no se debe confiar en la información que transporte. Tanto si se firma la petición como si no, el proveedor de identidad debe asegurarse que el elemento <AssertionConsumerServiceURL> o <AssertionConsumerServiceIndex> de la petición se validan como pertenecientes al proveedor de servicio al cual se le enviará la respuesta. Si no se hace así, pueden darse ataques de hombre-en-el-medio.

Uso del mensaje <Response>

Si el proveedor de identidad desea devolver un error, no debe incluir ninguna aserción en el mensaje <Response>. De lo contrario, si la petición tiene éxito (o la respuesta no está asociada con la petición), **el elemento <Response> debe ajustarse a lo siguiente**:

- El elemento **<Issuer>** puede ser omitido, pero si está presente debe contener el identificador único del proveedor de identidad que lo envía. El atributo *Format* debe ser omitido o tener un valor de *urn:oasis:names:tc:SAML:2.0:nameid-format:entity*.

- Debe contener al menos una <**Assertion>**. El elemento <Issuer> de cada aserción debe seguir las pautas establecidas en el punto precedente.

- El conjunto de una o más aserciones debe contener al menos un **<AuthnStatement>** que refleje la autentificación del principal por el proveedor de identidad.

- Al menos una aserción que contenga un <AuthnStatement> debe contener un elemento **<Subject>**. Este debe contener, al menos, un elemento <SubjectConfirmation> que contiene un *Method* de *urn:oasis:names:tc:SAML:2.0:cm:bearer*.

- El elemento **<SubjectConfirmation>** descrito arriba debe contener un elemento <SubjectConfirmationData> con un atributo Recipient. En dicho atributo está el servicio URL consumidor de la aserción del proveedor de servicio. También está el atributo NotOnOrAfter que limita la ventana de tiempo durante la cual puede ser entregada la aserción. Puede contener el atributo *Address* limitando las direcciones de clientes desde las cuales se puede entregar la aserción. Si el contenido del mensaje se debe a la respuesta de un mensaje <AuthnRequest>, entonces el atributo *InResponseTo* debe coincidir con el ID de la petición.

- Otras declaraciones y métodos de confirmación pueden ser incluidos en la aserción según le parezca al proveedor de identidad. En particular, los elementos **<AttributeStatement>** pueden ser incluidos. El mensaje <AuthnRequest> puede tener un atributo XML *AttributeConsumingServiceIndex* que referencia información sobre atributos deseados o requeridos. El proveedor de identidad puede ignorar esto, o enviar otros atributos como le parezca.

- La aserción que tenga confirmación de sujeto debe contener un **<AudienceRestriction>** incluyendo el identificador único del SP como <Audience>.

- **Otras condiciones** (y otros elementos <Audience>) pueden ser solicitados por el SP o según le parezca al IP. (Por supuesto, todas estas condiciones deben ser entendidas y aceptadas por el SP con el fin de considerar válida la aserción). El IP no esta obligado a cumplir las peticiones de <Conditions> de la <AuthnRequest>, si las hubiera.

d.- Ejemplo de <AuthnRequest> y <Response>

A modo de ejemplo y para ilustrar toda la teoría que hemos incluido, ponemos un mensaje <AuthnRequest>, un mensaje <Response> y un ejemplo de aserción contenida dentro del mensaje respuesta.

```
<samlp:AuthnRequest
        xmlns:samlp="urn:oasis:names:tc:SAML:2.0:protocol"
        ForceAuthn="true"
        AssertionConsumerServiceURL="http://www.ejemplo.com/"
        ProviderName="nombre_legible_por_humanos"
        ID="abe567de6"
        Version="2.0"
        IssueInstant="2005-01-31T12:00:00Z">

        <saml:Subject
                xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion">
                <saml:NameID
                        Format= “urn:oasis:names:tc:SAML:2.0:nameid-format:entity” >
                                URLdelsujeto
                </saml:NameID>
        </saml:Subject>
        <saml:Issuer
                xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
                Format="urn:oasis:names:tc:SAML:2.0:nameid-format:entity" >
                        URLdelSP
        </saml:Issuer>
        <RequestedAuthnContext Comparison=”exact”>
                  <AuthenContextClassRef>
                        urn:oasis:names:tc:SAML:2.0:ac:classes:Password
                </AuthnContextClassRef>
        </RequestedAuthnContext>
</samlp:AuthnRequest>
```

Figura 9. 44: Ejemplo de <**AuthnRequest**>

```
<samlp:Response
                        Version=”2.0”
                        ID=”abI4gxLGuc”
                        IssueInstant="2005-01-31T12:02:00Z"
                        InResponseTo="ID del mensaje que lleva al
                                        elemento <AuthnRequest">
                        <Status>
                                <StatusCode
                                Value=”urn:oasis:names:tc:SAML:2.0:status:Success”>
                        </Status>
                        <saml:Assertion>
                                --mirar abajo--
                        </saml:Assertion>
</samlp:Response>
```

Figura 9. 45: Ejemplo de <**Response**>

```
<saml:Assertion
        Version=”2.0”
        ID=”buGxcG4gIL”
        IssueInstant=”2005-01-31T12:02:00Z”>
        <saml:Issuer>proveedor_de_identidad.com/identidad</saml:Issuer>
        <saml:Conitions
                NotBefore=”2005-01-31T12:02:00Z”
                NotOnOrAfter=”2005-01-31T12:05:00Z ”>
                <AudienceRestriction>
                        <Audience>
                                URLdelSP
                        </Audience>
                </AudienceRestriction>
        </saml:Conditions>
        <saml:Subject>
                <saml:NameID
                        Format= “urn:oasis:names:tc:SAML:2.0:nameid-format:entity” >
                                URLdelsujeto
                </saml:NameID>

                <SubjectConfirmation
                        Method=”urn:oasis:names:tc:SAML:2.0:cm:bearer”>
                        <SubjectConfirmationData NotOnOrAfter=
                        ”2005-01-31T12:05:00Z” InResponseTo=”abe567de6”
                         Recipient=”URLdelSP ”>
                </SubjectConfirmation>
        </saml:Subject>
        <saml:AuthnStatement AuthnInstant=”2005-01-31T12:02:00Z”>
                <AuthnContext>
                        <AuthenContextClassRef>
                                urn:oasis:names:tc:SAML:2.0:ac:classes:Password
                        </AuthnContextClassRef>
                </AuthnContext>
        </saml:AuthStatement>
</saml:Assertion>
```

Figura 9. 46: Ejemplo de <**Assertion**> que lleva el mensaje <Response>

9.4.- Bindings

9.4.1.- Introducción a los bindings SAML

Se llaman **bindings** del protocolo SAML al mapeado del intercambio de mensajes de petición/respuesta de SAML sobre protocolos estándares de mensajes o comunicación.

Por ejemplo Por ejemplo, la especificación de SAML proporciona un binding sobre como las peticiones/respuesta son transportadas mediante el intercambio de mensajes SOAP. mientras que el SAML URI Binding define cómo los mensajes de gestión de protocolo de SAML se pueden comunicar con la resolución de URI. Para aclarar lo que son los binding incluimos un dibujo donde aparece el binding SOAP.

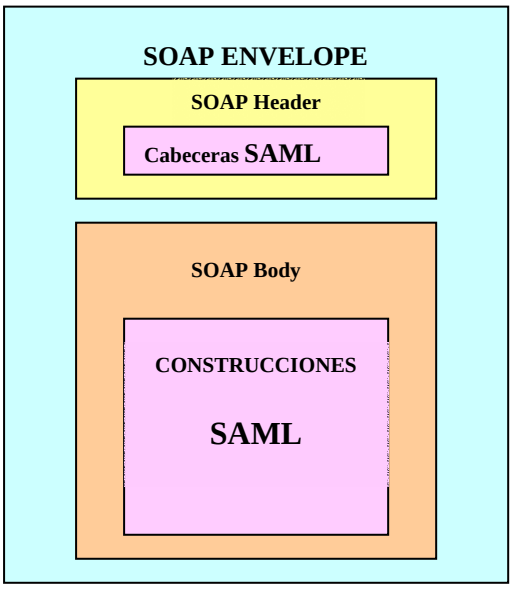

Figura 9. 47: Ejemplo de **Binding SOAP de SAML**

Los bindings definidos dentro de la especificación SAML son:

- SAML SOAP Binding:

Define cómo los mensajes de protocolo de SAML son transportados dentro de los mensajes SOAP 1.1. Además también define cómo los mensajes SOAP son transportados sobre HTTP. Se detallará en secciones siguientes porque es uno de los que se utilizan en la aplicación realizada.

- Reverse SOAP (PAOS) Binding:

Define un intercambio de mensajes multi-escenario SOAP/HTTP que permite que un cliente HTTP sea el que responda en SOAP. Utilizado en el perfil ECP y diseñado particularmente para soportar las entradas WAP. Al igual que el binding SAML SOAP, se detallará en una sección posterior.

- HTTP Redirect Binding:

Define cómo los mensajes de protocolo de SAML pueden ser transportados usando mensajes HTTP redireccionados.

- HTTP POST Binding:

Define cómo los mensajes de protocolo SAML se pueden transportar dentro del contenido de codificación base64 de un control de forma HTML.

- HTTP Artifact Binding:

Define cómo una referencia a una petición o a una respuesta (es decir un artefacto) de SAML es transportada por HTTP. Define dos mecanismos: un control de forma HTML, o una secuencia de preguntas en el URL.

- SAML URI Binding:

No se puede definir fácilmente.

Para aclararnos los **conceptos que vamos a explicar** es conveniente que detallemos la estructura del apartado para ganar facilidad en el entendimiento. Una vez que hemos explicado algunos conceptos relacionados y los diferentes tipos de binding, pasamos a explicar los que nos interesan ya que serán los que usemos en nuestra implementación. Éstos son los llamados binding SOAP de SAML y binding PAOS. Dentro de la parte relacionada con el binding SOAP, detallaremos características que son independientes del protocolo. Empezamos con unas operaciones básicas que debemos realizar para ajustarnos al citado binding; definimos aquellas cabeceras SOAP necesarias; vemos el uso de SOAP sobre HTTP y damos un ejemplo que ilustra esta función; definimos las cabeceras HTTP necesarias para la utilización del binding SOAP de SAML; vemos los requerimientos y limitaciones en el almacenaje de los mensajes que intervienen, terminando, por último, con un apartado en dónde explicamos los errores que se deben devolver según qué caso.

Lo explicado hace referencia al binding SOAP de SAML pero recordemos que también interviene el binding PAOS. Para tratar esta parte empezamos explicando el intercambio de mensajes que intervienen en este binding y detallamos los dos posibles tipos de mensajes del binding. Al igual que en el binding anterior, ponemos las limitaciones que se producen en el almacenaje de los mensajes. Terminamos dando los posibles tipos de errores que se pueden producir y cómo devolverlos.

9.4.2.- Binding SOAP de SAML

SOAP es un protocolo ligero que se concibió para el intercambio de información estructurada en un medio descentralizado y distribuido. Usa tecnología XML para definir un marco extensible de mensajería proporcionando un constructor de mensajes que pueden ser intercambiados sobre muchos protocolos subyacentes. Dicho marco se ha sido diseñado para ser independiente tanto del modelo de programación como de la semántica de la implementación específica. Sus dos principales objetivos de diseño son la **simplicidad** y la **extensibilidad**.

SOAP trata de alcanzar esos objetivos por omisión, desde el marco de mensajes, características que son frecuentemente encontradas en los sistemas distribuidos. Se incluyen tales características pero no se limitan a “honradez”, “coherencia”, “routing” y “patrones de intercambio de mensajes (MEPs).

Un mensaje SOAP es fundamentalmente una transmisión de un solo sentido entre nodos SOAP, desde un transmisor hasta un receptor y atravesando, posiblemente, varios intermediarios. Éstos son unidos por aplicaciones para implementar modelos de interacción más complejos.

SOAP define un sobre (envelope) de mensaje XML. Incluye secciones de cabeceras y cuerpo, permitiendo que los datos y la información de control se puedan transmitir. También define reglas de proceso asociadas a este sobre y un binding para la transmisión de mensajes SOAP. Dicho binding define cómo usar SOAP para mandar y recibir peticiones y respuestas SAML.

En los apartados siguientes definimos aspectos del binding SOAP de SAML que son independientes de los protocolos subyacentes sobre los que se transportan los mensajes SOAP. **Este binding solo soporta el uso de la versión 1.1 de SOAP.**

a.- Operaciones básicas

Un mensaje SOAP 1.1 tiene tres elementos: un sobre, datos de cabecera y un cuerpo de mensaje. Los elementos del protocolo SAML petición-respuesta deben ir dentro de dicho cuerpo de mensaje.

SOAP 1.1 también define un sistema opcional de codificación de datos. Este sistema no se usa dentro del binding SOAP de SAML. Esto significa que los mensajes SAML pueden ser transportados usando SOAP pero sin recodificación desde el "estándar" SAML hasta otro esquema basado en una codificación SOAP.

El sistema usado para las conversaciones SAML sobre SOAP es un simple modelo de **petición-respuesta**.

1.- Una entidad de sistema actuando como un solicitante SAML puede transmitir un elemento de petición SAML hacia otra entidad, que actúa como respondedor SAML, dentro del cuerpo de un mensaje SOAP. No se debe incluir más de una petición SAML por mensaje SOAP ni incluir ningún elemento XML adicional dentro del cuerpo del mensaje.

2.- El respondedor SAML debe retornar bien un elemento respuesta SAML dentro del cuerpo de otro mensaje SOAP ó generar un error SOAP. No se debe incluir más de una respuesta SAML por mensaje SOAP ni incluir ningún elemento XML adicional dentro del cuerpo del mensaje. Si dicho respondedor no puede, por alguna razón, procesar la petición SAML, entonces debe generar un error SOAP. Los códigos de error SOAP no deben ser enviados por errores dentro del dominio de problemas SAML, por ejemplo, la incapacidad de encontrar la extensión de un esquema ó que el sujeto no está autorizado para acceder a un recurso en una pregunta de autorización.

Al recibir una respuesta SAML en un mensaje SOAP, el solicitante SAML no debe enviar un código de falta ni ningún mensaje de error al respondedor SAML. Puesto que se trata de un modelo simple de intercambio de mensajes, añadirle elementos adicionales como condiciones de error, se complicaría innecesariamente el protocolo.

Los solicitantes SAML deben generar documentos SOAP referenciando sólo el esquema de espacio de nombres final de XML. Los respondedores SAML deben ser capaces de procesar tanto el esquema de espacio de nombres XML usado en SOAP 1.1 como el esquema de espacio de nombres final XML.

b.- Cabeceras SOAP

Un solicitante SAML en una conversación sobre SOAP puede añadir arbitrariamente cabeceras al mensaje SOAP. Este binding no define ninguna cabecera SOAP adicional.

La razón de la necesidad de permitir cabeceras es que algún software y librerías SOAP podrían añadirlas al mensaje SOAP, estando éstas fuera del control SAML. También algunos protocolos subyacentes podrían añadirlas al requerir enrutado de mensajes ó por mecanismos de seguridad de mensajes.

Un respondedor SAML no debe requerir ninguna cabecera en el mensaje SOAP a fin de procesarlo correctamente por si mismo, pero puede requerirlas en requerimientos de seguridad de mensaje ó direccionamiento de enrutado subyacente.

El motivo es que requerir cabeceras extra podría causar fragmentación en el estándar SAML y dañar la interoperabilidad.

c.- Uso del binding SOAP sobre HTTP

Un procesador SAML que reclama conformidad con el binding SOAP de SAML debe implementar SAML sobre SOAP sobre HTTP. El perfil HTTP para SOAP describe el uso de una cabecera **SOAPAction** como parte de la petición HTTP SOAP.

Un respondedor SAML no debe contar con el valor de este campo. Dicho valor lo establece un solicitante SAML con el valor:

```
SOAPAction= http://www.oasis-open.org/committees/security
```

Figura 9. 48: Valor de la cabecera **SOAPAction**

Ponemos un ejemplo del uso del binding SOAP sobre HTTP en el que se pregunta por una aserción que contiene una declaración de atributos desde una autoridad de atributos SAML. Para entender mejor el dibujo observamos que la parte SAML se ha remarcado en un color rosa, la parte SOAP en uno naranja mientras que la parte HTTP se ha puesto de color amarillo.

Ponemos también la correspondiente respuesta, la cual suministra una aserción que contiene la declaración de atributos que se pidió. Para mayor claridad seguiremos el mismo código de colores

```
POST /SamlService HTTP/1.1
Host: www.example.com
Content-Type: text/xml
Content-Length: nnn
SOAPAction: http://www.oasis-open.org/committees/security

<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Body>

  <samlp:AttributeQuery xmlns:samlp:="..."
  xmlns:saml="..." xmlns:ds="..." ID="_6c3a4f8b9c2d"
  Version="2.0" IssueInstant="2004-03-27T08:41:00Z"
  <ds:Signature> ... </ds:Signature>
  <saml:Subject> ... </saml:Subject>
  </samlp:AttributeQuery>

</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figura 9. 49: Ejemplo de SOAP sobre HTTP

```
HTTP/1.1 200 OK
Content-Type: text/xml
Content-Length: nnnn

<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Body>

  <samlp:Response xmlns:samlp="..." xmlns:saml="..."
  xmlns:ds="..."
  ID="_6c3a4f8b9c2d" Version="2.0" IssueInstant="2004-03-
  27T08:42:00Z">
  <saml:Issuer>https://www.example.com/SAML</saml:Issuer>
  <ds:Signature> ... </ds:Signature>
  <Status>
  <StatusCode Value="..."/>
  </Status>
  <saml:Assertion>
  <saml:Subject> ... </saml:Subject>
  <saml:AttributeStatement> ... </saml:AttributeStatement>
  </saml:Assertion>
  </samlp:Response>

</SOAP-Env:Body>
</SOAP-ENV:Envelope>
```

Figura 9. 50: Respuesta al ejemplo SOAP sobre HTTP

d.- Cabeceras HTTP

Un solicitante SAML en una conversación SAML sobre SOAP sobre HTTP puede añadir cabeceras arbitrariamente a la petición HTTP. Este binding no define ninguna cabecera HTTP adicional.

La razón de la necesidad de permitir cabeceras es que algún software y librerías HTTP podrían añadirlas al mensaje HTTP, estando éstas fuera del control SAML. También algunos protocolos subyacentes podrían añadirlas al requerir enrutado de mensajes ó por mecanismos de seguridad de mensajes.

Un respondedor SAML no debe requerir ninguna cabecera en el mensaje HTTP a fin de procesarlo correctamente por si mismo, pero puede requerirlas en requerimientos de seguridad de mensaje ó direccionamiento de enrutado subyacente.

El motivo es que requerir cabeceras extra podría causar fragmentación en el estándar SAML y dañar la interoperabilidad.

e.- Almacenaje

Los proxies HTTP no deben almacenar los mensajes SAML. Para asegurarnos de esto, debemos seguir las siguientes reglas para los **solicitantes HTTP**:

- Incluir un campo de cabecera Cache-Control establecido a “no-cache, no-store”.
- Incluir un campo de cabecera Pragma con el valor de “no-cache”.

En cambio para los que **responden**:

- Incluir campo de cabecera Cache-Control con el valor “no-cache, no-store, must-revalidate, private”.
- Incluir campo de cabecera Pragma establecido a “no-cache”.
- No incluir un Validator, tal como las cabeceras Last-Modified ó ETag.

f.- Informe de errores

Un respondedor SAML que rechaza efectuar un intercambio de mensajes con el solicitante SAML debería retornar una respuesta “**403 Forbidden**”. En este caso, el contenido del cuerpo del mensaje HTTP no es significante.

En el caso de una error SOAP mientras se procesa una petición SOAP, el servidor HTTP SOAP debe devolver una respuesta “**500 Internal Server Error**” e incluir un mensaje SOAP con un elemento SOAP <SOAP-ENV:fault>. Este tipo de error debería ser devuelto por errores relacionados con SOAP antes que el control sea transferido al procesador SAML, ó cuando el procesador SOAP informa de un error interno (por ejemplo espacio de nombres SOAP XML sea incorrecto, el esquema SAML no puede ser localizado, el procesador SAML lanza una excepción, etc.).

En el caso de un error en el proceso SAML, el servidor SOAP HTTP debe responder con “**200 OK**” e incluir un elemento específico <samlp:Status> en la respuesta SAML dentro del cuerpo del mensaje SOAP. Recalcamos que el elemento

<samlp:Status> no aparece por sí mismo en el cuerpo SOAP, pero solo dentro de una respuesta SAML de cualquier tipo.

9.4.3.- Binding PAOS (Reverse SOAP)

Los implementadores deben cumplir con las normas de proceso general especificadas en [PAOS] así como con las dadas a continuación. En caso de conflicto debe seguirse las normas [PAOS].

El **binding PAOS** es un mecanismo mediante el cual un solicitante HTTP puede anunciar la habilidad para actuar como respondedor SOAP ó como un intermediario de un solicitante SAML. El solicitante HTTP puede sostener un modelo donde se le mande un petición SAML dentro de un sobre SOAP y en una petición HTTP desde el solicitante SAML. Y donde el solicitante HTTP responda con una respuesta SAML dentro de un sobre SOAP en la siguiente petición HTTP. Este modelo de intercambio de mensajes soporta el caso de uso del perfil ECP SSO, en el cual el solicitante HTTP es un intermediario en un intercambio de autenticación.

a.- Intercambio de mensajes

El binding PAOS incluye **dos modelos de intercambio de mensajes**:

- EL solicitante HTTP manda una petición HTTP a un solicitante SAML. Este último responde con una respuesta HTTP que contiene un sobre SOAP y dentro un mensaje de solicitud SAML.

- Después, el solicitante HTTP manda una solicitud HTTP al solicitante SAML original que contiene un sobre SOAP y dentro de él, un mensaje respuesta SAML. El solicitante SAML responde con una respuesta HTTP, posiblemente en respuesta a una petición de servicio realizada en un paso anterior.

El **perfil ECP usa el binding PAOS** para proporcionar autentificación a un cliente de un proveedor de servicio antes de que proporcionar un determinado servicio. Esto ocurre en los siguientes pasos:

1. El cliente solicita un servicio mediante una petición HTTP.
2. El proveedor de servicio responde con una petición de autentificación SAML. Se envía mediante una petición SOAP transportada en una respuesta HTTP.
3. El cliente devuelve una respuesta SOAP que lleva una respuesta de autentificación SAML. Es enviada mediante una nueva petición HTTP.
4. Si la autentificación es correcta y el cliente tiene las autorizaciones pertinentes, el proveedor de servicio puede responder a la original petición de servicio en una respuesta HTTP.

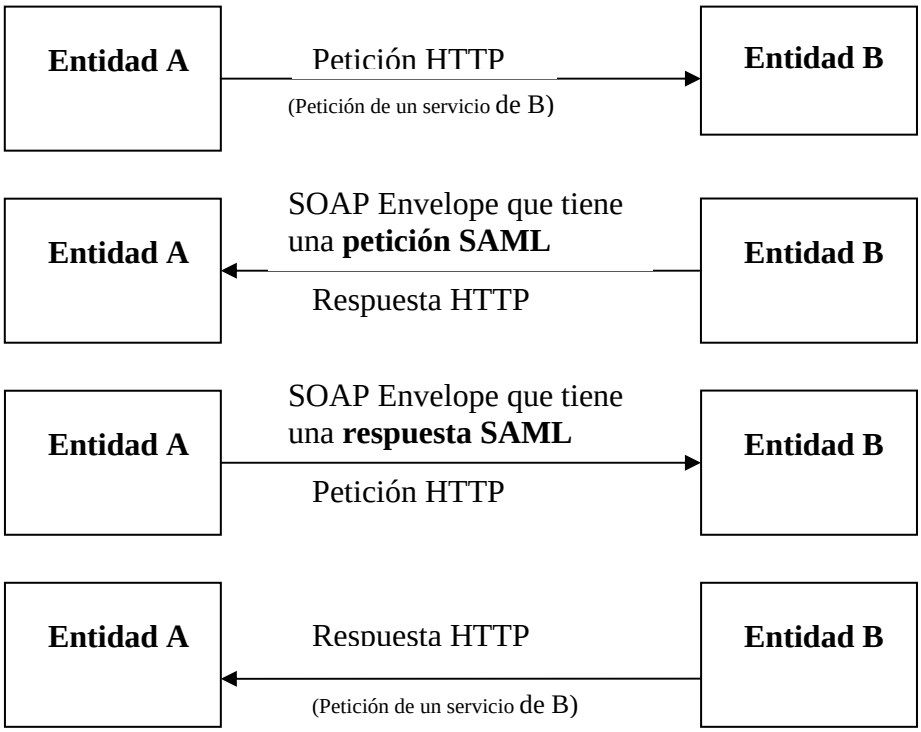

Figura 9. 51: Intercambio de mensajes del binding PAOS

El solicitante HTTP anuncia la capacidad de manejar este binding en su petición HTTP, usando las cabeceras HTTP definidas en la especificación PAOS:

- Campo de cabecera HTTP Accept:
  Debe indicar la capacidad de aceptar el tipo de contenido:

```
Accept= “application/vnd.paos+xml”
```

Figura 9. 52: Valor del campo de cabecera **Accept**

- Campo de cabecera PAOS:
  Debe presentar y especificar la versión PAOS con un valor como mínimo de:

```
PAOS=“urn:liberty:paos:2003-08”
```

Figura 3. 1: Valor del campo de **cabecera PAOS**

Pueden especificarse cabeceras adicionales PAOS por perfiles que usan dicho binding. El solicitante HTTP puede añadir cabeceras adicionales en la petición HTTP. Este binding no define ningún mecanismo “RelayState”. Se sugiere el uso de cabeceras SOAP cuando se tenga que implementar dicho mecanismo.

En la siguiente sección pasamos a profundizar en los dos pasos del intercambio de mensajes.

b.-Petición HTTP, Petición SAML en respuesta SOAP

En respuesta de una petición HTTP arbitraria, el respondedor HTTP puede devolver un mensaje petición SAML usando este perfil. Esto se lleva a cabo devolviendo un sobre SOAP 1.1 que contiene un simple mensaje de petición SAML en el cuerpo SOAP, sin que haya contenido adicional en dicho cuerpo. El sobre SOAP puede contener cabeceras SOAP adicionales definidas por PAOS, perfil SAML ó demás especificaciones.

Hay que hacer notar que mientras el mensaje de petición SAML es entregado al solicitante HTTP, el actual futuro destinatario puede ser otra entidad de sistema, con el solicitante HTTP actuando como un intermediario.

c.- Respuesta HTTP, Respuesta SAML dentro de petición SOAP

Cuando el solicitante HTTP entrega el mensaje de respuesta SAML al futuro destinatario usando el binding PAOS, lo toma como el único elemento del cuerpo SOAP, dentro de un sobre SOAP y una petición HTTP. El solicitante HTTP puede ó no ser el que originó la respuesta SAML. El sobre SOAP puede contener cabeceras SOAP arbitrarias definidas por PAOS, perfiles de SAML ó demás especificaciones

La respuesta HTTP no se especifica en este perfil. Los perfiles pueden definir limitaciones adicionales para el contenido HTTP de las respuestas que no son SOAP durante los intercambios de mensajes cubiertos por este binding.

d.- Almacenaje

Los proxies HTTP no deberían almacenar los mensajes del protocolo SAML. Para asegurarnos de que se cumple este requisito, se tiene que dar los pasos explicados a continuación (suponemos que se usa HTTP 1.1), incluyendo los campos:

- Campo de cabecera **Cache-Control**.
  Para los solicitantes que manden mensajes del protocolo SAML se deberá establecer a “no-cache, no-store”.
  En cambio para los que responden dichos mensajes se deberá poner a “no-cache, no-store, must-revalidate”.

- Campo de cabecera **Pragma**.
  Establecido a “no-cache”.

- No debe incluir un “Validador” como los campos “Last-Modified” ó “ETag” cuando se responde a un mensaje.

e.- Informe de Error:

Se deben seguir los convenios dados en el estándar HTTP y SOAP. Los errores que ocurran mediante el proceso SAML no deben ser señalados en las capas SOAP ni

HTTP, debiendo ser tratados usando los mensajes de respuesta SAML al que se le incluye un elemento de error **<samlp:Status>**.

9.5.- Perfiles (Profiles)

Generalmente, un perfil de SAML define limitaciones y/o extensiones en el uso de SAML para una aplicación particular. Para realzar la interoperabilidad hay que quitar algo de flexibilidad en un estándar general de uso. Un perfil introducirá limitaciones en el uso de la especificación con el fin de aplicarse de manera más óptima a un determinado escenario de aplicación.

Como ejemplo, dos diversas combinaciones se demuestran en el diagrama de abajo. En el diagrama superior, tanto la petición de autorización como su respuesta se envían usando el HTTP POST Binding. En el diagrama inferior, se envía la petición de autorización usando el HTTP POST Binding, la respuesta sin embargo utiliza una combinación de HTTP Artifact y de SOAP Bindings.

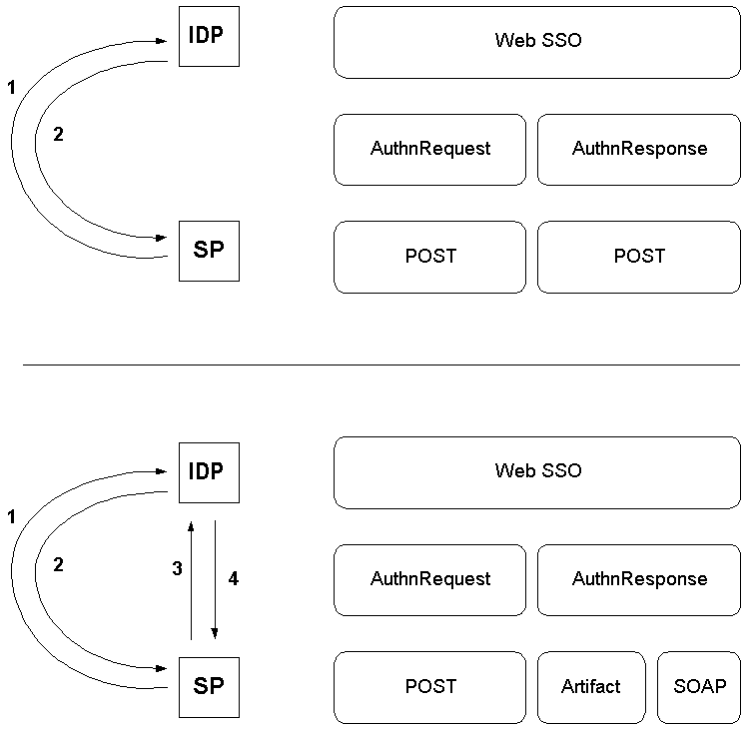

Figura 9. 53: Ejemplo de perfiles utilizados en SAML

SAML define una serie de perfiles que corresponden a unos posibles escenarios de uso. No nos adentraremos en ellos ya que la implementación realizada corresponde con uno, con lo que se explicará en otra sección.

Hace una pequeña diferenciación entre perfiles que implementan Single Sign On (SSO) y los que no. SSO no es más que aquel escenario en el que un usuario se

identifica en una entidad y todas las entidades tiene acceso a esa identidad de forma transparente al usuario. Los distintos perfiles SSO definidos son:

- **Web Browser SSO**
  En este perfil se define el SSO para un usuario Web.

- **Enhanced Client or Proxy** (ECP)
  Es el perfil que vamos a implementar. Se deja su descripción para un apartado posterior.

- **Identity Provider Discovery**
  Mediante este perfil podemos descubrir, de manera dinámica, qué proveedor de identidad autentificó al usuario.

- **Single Logout** (SLO)
  Este perfil indica cómo podemos indicar a todas las entidades que intervienen en un escenario, que un determinado usuario ya no posee una determinada identidad o que dicha identidad ya no se relaciona con él.

- **Name Identifier Management**
  Haciendo uso de este perfil podremos propagar el cambio de identidad que ha tenido un determinado usuario en un determinado proveedor de identidad.

Por otro lado SAML también define perfiles que no son SSO. Éstos no nos interesan tanto desde el punto de vista de nuestra aplicación, así que sólo se les nombrará:
- **Artifact Resolution**
- **Assertion Query/Request**
- **Name Identifier Mapping**
- **SAML Attribute**
  Dentro de este atributo aparecen los siguientes:
  - **Basic Attribute**
  - **X.500/LDAP Attribute**
  - **UUID Attribute**
  - **DCE PAC Attribute**
  - **XACML Attribute**

9.6.- Contexto de autentificación

9.6.1.- Introducción

En este apartado especificamos una sintaxis para la definición de declaraciones de contexto de autentificación y una lista inicial de clases de contextos. Para ello lo primero que hacemos es definir algunos conceptos importantes; indicamos la declaración de un contexto y, por último, explicamos lo que son las clases de contextos

(resaltando las ventajas que tienen y cómo se pueden extender esos esquemas) y damos los que están ya definidos.

Conceptos de Contextos de Autentificación

Si una parte que confía se fía de una autentificación de un principal realizada por una autoridad, puede querer información adicional a la propia aserción para estimar el nivel de confianza que puede tener de la aserción. La especificación define esquemas XML para la creación de declaraciones de contextos de autentificación que permiten proporcionar la citada información adicional. Además, la especificación, define un número de clases de contexto; categorías en las que podemos meter las declaraciones de contexto, simplificando de ese modo su interpretación.

Las elecciones que una autoridad de autentificación hace aquí, serán manejadas en gran parte por las partes que confían con las que dicha autoridad interactúa. Estos requisitos serán determinados por la naturaleza del servicio (eso es, la sensibilidad de cualquier información intercambiada, el valor financiero asociado, la tolerancia del riesgo de partes que confían, etc.). Consecuentemente, para cualquier otro servicio que no sea el trivial, si la parte que confía debe tener la confianza suficiente en las aserciones que recibe de la autoridad de autentificación, debe saber qué tecnologías, protocolos y procesos se usaron ó siguieron en los mecanismos de autentificación en los que se basa la aserción. Provisto de esta información y confiando en el origen de la aserción actual, la parte que confía podrá decidir mejor a qué servicios da acceso la aserción de autentificación.

El **contexto de Autentificación** se define como la información, adicional a la propia aserción de autentificación, que la parte que confía puede requerir antes de decidir los derechos respecto a una aserción de autentificación. Tal contexto puede incluir, pero no está limitado, el método de autentificación utilizado.

9.6.2.- Declaración de contexto de Autentificación

Si una parte que confía se fía de una autentificación realizada por una autoridad a otra entidad, puede requerir **información adicional** de la propia autentificación para ponerla en un marco de gestión de riesgo. Esta información puede incluir:

- Los mecanismos de identificación del usuario inicial (por ejemplo cara-a-cara, en línea, secreto compartido).

- Los mecanismos para minimizar el medio de las credenciales (por ejemplo frecuencia de renovación de credenciales, generación de clave del lado del cliente).

- Los mecanismos para almacenar y proteger credenciales (por ejemplo, smartcard, reglas de password).

- El mecanismo ó método de autentificación (por ejemplo password, SSL basado en certificado).

Las variaciones y permutaciones de las características listadas arriba garantizan que no todas las aserciones tendrán el mismo nivel de confianza con respecto a la parte que confía. Una aserción de autentificación determinada se caracterizará por los valores de cada uno de esas (y otras) variables.

Una autoridad de autentificación puede entregar, a la parte que confía, la información de contexto de autentificación en forma de **declaración de contexto**, un documento XML bien insertado directamente, bien referenciado dentro de una aserción.

Los solicitantes SAML son capaces de pedir que una autentificación cumpla un determinado contexto, identificándolo en la petición. Un solicitante puede también especificar que una autentificación debe ser conducida con un contexto que exceda algún valor de declaración (para alguna definición acordada de "**exceeds**").

a.- Modelo de Datos

Una declaración de contexto de autentificación captura características de los procesos, procedimientos, y mecanismos por los cuales la autoridad de autentificación identificó al principal, protege los secretos en los que se basan las siguientes autentificaciones y los mecanismos con los que las realiza.

Estas características se categorizan en el esquema Contexto de Autentificación como sigue:

- **Identificación**:

Características que describen los procesos y mecanismos que la autoridad de autentificación usa para crear una asociación entre el sujeto y la identidad (ó nombre) por el que se conoce al sujeto.

- **Protección Técnica**:

Características que describen cómo se guarda de manera segura el "secreto" (el conocimiento ó posesión de lo que permite autentificar la autoridad al sujeto).

- **Protección Operacional**:

Características que describen los controles de seguridad de procedimiento empleadas por la autoridad de autentificación (por ejemplo chequeos de seguridad, registros de archivos).

- **Métodos de Autentificación**:

Características que definen los mecanismos por los cuales el sujeto de la aserción mandada se autentifica en la autoridad (por ejemplo un password contra una smartcard).

- **Acuerdos Directivos**:

Características que describen el marco legal (por ejemplo limitaciones de obligación y obligaciones contractuales) subyacentes al evento de autentificación y/ó su infraestructura técnica de autentificación asociada.

b.- Extensibilidad

El esquema tiene puntos bien definidos de extensión a través del elemento <Extension>. Las autoridades de autentificación pueden usar este elemento para insertar detalles adicionales del contexto de autentificación que ellos mandan en la aserción (siempre y cuando dichos puntos de extensión sean entendibles por la parte que confía que va a consumir la aserción). Estos elementos adicionales deben de estar en un espacio de nombres XML separado de la declaración del contexto base ó del esquema de clase que se aplica a la propia declaración.

9.6.3.- Clases de Contexto de Autentificación

El número de permutaciones de diferentes características asegura que, teóricamente, hay un número infinito de contextos de autentificación. La implicación es que, en teoría, cualquier parte que confía sería capaz de parsear las declaraciones del contexto y, lo que es más importante, analizar dichas declaraciones con el fin de afirmar la "calidad" de la aserción de autentificación asociada.

Afortunadamente, es posible hacer una **optimización**. En la práctica la mayoría de los contextos pueden introducirse dentro de categorías determinadas por las actividades industriales y tecnológicas. Como ejemplo, en el mundo empresarial, es común la autentificación basada en certificados. Por supuesto, el contexto completo no está limitado a las especificaciones de cómo se autentificó al principal. A pesar de eso, el método de autentificación es con frecuencia la característica más visible y como tal, puede servir como un clasificador útil para una clase de contextos de autentificación relacionados.

En la especificación se expresa este concepto como la **definición** de una serie de **clases de contextos de autentificación**. Cada clase define un subconjunto apropiado del conjunto completo de contextos de autenticación. Las clases se han escogido como representativas de las prácticas y tecnologías actuales de autentificación, y proporcionan, a las partes que confían y afirman, una forma convenientemente abreviada cuando nos referimos a emisiones de contexto de autentificación.

Por ejemplo, una autoridad de autentificación puede incluir, junto con la declaración completa del contexto de autentificación que le proporciona a una parte que confía, una aserción que indica que el contexto pertenece también a una clase del contexto de autentificación. Para algunas partes que confían, esta aserción es suficiente para poder asignar un nivel de confianza determinado. Pero otras podrían preferir examinar por completo la propia declaración del contexto.

En la siguiente tabla indicamos las definiciones de las clases de contextos de autentificación definidos. En una sección posterior pasaremos a describir brevemente cada una de ellas para poder conocerlas mejor.

Internet Protocol	Internet Protocol Password
Kerberos	Mobile One Factor Unregistered
Mobile Two Factor Unregistered	Mobile One Factor Contract
Mobile Two Factor Contract	Password
Password Protected Transport	Previous Session
Public Key – X.509	Public Key – PGP
Public Key – SPKI	Public Key – XML Signature
Smartcard	Smartcard PKI
Software PKI	Telephony
Nomadic Telephony	Personalized Telephony
Authenticated Telephony	Secure Remote Password
SSL/TLS Cert-Based Client Authn	Time Sync Token
Unspecified	

Tabla 9. 24: Clases de **Contexto de autentificación**

a.- Ventajas de las Clases de Contextos de Autentificación

La introducción de capas adicionales de clases y la definición de una lista inicial de clases representativas y flexibles, hace posible que:

- Sea más fácil llegar un acuerdo entre la autoridad de autentificación y la parte que confía respecto a los **contextos aceptables**, proporcionando un marco de discusión.

- Sea más fácil indicar, a la parte que confía, sus **preferencias al solicitar una aserción** de autentificación a una autoridad de autentificación.

- Las partes que confían **simplifiquen la carga de proceso** de las declaraciones de contexto dándole la opción de quedar satisfechas mediante la clase asociada.

- Las partes que confían se puedan **aislar** del impacto de las **nuevas tecnologías** de autentificación.

- Sea más fácil, para las autoridades de autentificación, publicar sus **capacidades de autentificación**, por ejemplo a través de WSDL.

b.- Extensibilidad

Como se hace en el núcleo del esquema de declaración de contexto, los esquemas separados permiten el elemento <Extension> en ciertos lugares del árbol de estructura. En general, donde aparezca dicho elemento como hijo de un elemento <xs:choice>, se puede quitar esta opción al crear la apropiada definición del esquema de clase como una restricción del tipo base. Cuando el elemento <Extension> aparezca como un hijo opcional de un elemento <xs:sequence>, se le permite quedarse además de cualquier elemento requerido.

Consecuentemente, las declaraciones de contexto de autentificación pueden incluir el elemento **<Extension>** (con elementos adicionales en diferentes espacios de

nombres) y ajustarse a los esquemas de clases de contexto (si ellos satisfacen los otros requisitos del esquema, por supuesto).

Los esquemas de clases de contexto restringen las definiciones de tipos del esquema de contexto base. Como punto de la extensión, los esquemas de clases de contexto en sí mismos, pueden ser adicionalmente restringidos – sus definiciones de tipo sirven como tipos base en algún otro esquema (potencialmente delimitados por alguna comunidad que desea una clase de contexto de autentificación definida más ajustadamente). Para prevenir lógicas contradicciones, cualquier extensión del esquema sólo puede limitar las definiciones de tipo del esquema de clase. Para cumplir esta restricción, se define el esquema de clase de contexto con el atributo finalDefault="extension" del elemento <schema> para prevenir este tipo de derivación.

Clases de contexto de autentificación pueden ser desarrolladas por otros grupos diferentes al *Security Services Technical Committee* (SSTC). Los miembros de OASIS pueden desear documentar y someterlos para consideración por el SSTC en una versión futura de la especificación, y otros grupos pueden desear simplemente informar el comité de su trabajo.

Las pautas para la especificación de clases nuevas de contexto son así:
- Especificar una URI que identifique a la clase del contexto de manera única.
- Proporcionar información de contacto para el autor de la clase.
- Proporcionar una descripción textual de las circunstancias bajo las cuales de debe usar esta clase.
- Proporcionar un documento esquema XML válido que implemente la clase.

c.- Clases de contextos de autenticación

A continuación introducimos una breve reseña de las diferentes clases de contextos de autentificación definidas en la implementación. Se realizará en orden alfabético, no estando implicada ninguna otra característica en el citado orden. Las clases se identifican por una URL proveniente de:

```
urn:oasis:names:tc:SAML:2.0:ac:classes
```

Figura 9. 54: URL de donde provienen las identificaciones de clases

Los esquemas de clases son definidos como restricciones de partes del esquema de contexto de autentificación tipo.

- **Internet Protocol**:

Esta clase es aplicable cuando se autentica a un principal a través del uso de una dirección IP suministrada.

- **Internet Protocol Password**:

Esta clase es aplicable cuando se autentifica a un principal a través del uso de una dirección IP suministrada, además de un nombre de usuario y una clave.

- Kerberos:

Aplicable cuando el principal se ha autentificado usando una contraseña en una autoridad local de autorización y con el fin de obtener un ticket Kerberos. Dicho ticket es luego usado en posteriores autenticaciones de red.

- Mobile One Factor Unregistered:

No representa ningún procedimiento de registro de clientes sino autentificación del dispositivo móvil sin requerimientos explícitos de interacción con el usuario final. Esta clase de contextos autentica únicamente al dispositivo y nunca al usuario. Es útil cuando el operador móvil quiere añadir un dispositivo extra de seguridad a su proceso de autentificación.

- Mobile Two Factor Unregistered:

No representa ningún procedimiento de registro de clientes sino una autentificación basada en dos factores: dispositivo seguro y PIN de usuario. Esta de clase de contextos es útil en servicios en los que el operador móvil quiere unir la identificación del cliente con el dispositivo capturando los datos del teléfono móvil en la inscripción. De esta manera se proporciona un servicio de autentificación basado en los citados dos factores.

- Mobile One Factor Contract:

Representa un registro de cliente de contrato y un simple factor de autentificación. Por ejemplo, un dispositivo de firmado digital con memoria para almacenamiento de clave, tal como MSISDN pero sin requerir PIN para autenticación de usuario en tiempo real.

- Mobile Two Factor Contract:

Representa un registro de cliente de contrato y una autenticación basada en dos factores. Por ejemplo, un dispositivo de firmado digital con memoria para almacenamiento de clave, tal como GSM SIM, que requiere un certificado de identidad de usuario explicito, como el PIN.

- Password:

Esta clase es aplicable cuando un principal se autentifica a una autoridad de autentificación a través de una contraseña sobre una sesión HTTP no protegida.

- Password Protected Transport:

Esta clase es aplicable cuando un principal se autentifica a una autoridad de autentificación a través de una contraseña sobre una sesión HTTP protegida.

- Previous Session:

Aplicable cuando el principal se ha autentificado a una autoridad de autentificación en cualquier momento del pasado usando cualquier contexto de autentificación proporcionado por dicha autoridad. Por consiguiente, un posterior evento de autentificación debe ser significativamente separado en el tiempo de la actual petición de acceso al recurso.

El contexto para la sesión previa no se incluye en esta clase ya que el usuario no se autentifica durante esta sesión y esos mecanismos utilizados para la autentificación en una sesión previa no deberían ser usados para decidir si permite el acceso o no al recurso.

- **Public Key – X.509**:
La clase de contextos X509 indica que el principal se autentificó por medio de una firma digital donde la clave estaba validada como parte de una infraestructura de clave pública X.509.

- **Public Key – PGP**:
La clase de contextos PGP indica que el principal se autentificó por medio de una firma digital donde la clave estaba validada como parte de una infraestructura de clave pública PGP.

- **Public Key – SPKI**:
La clase de contextos SPKI indica que el principal se autentificó por medio de una firma digital donde la clave estaba validada como parte de una infraestructura SPKI.

- **Public Key - XML Digital Signature**:
Esta clase indica que el principal se autentificó por medio de una firma digital conforme a las reglas específicas de proceso de la especificación XML Digital Signature.

- **Smartcard**:
La clase Smartcard se identifica cuando un principal se autentifica en una autoridad de autentificación usando una Smartcard.

- **Smartcard PKI**:
Esta clase es aplicable cuando un principal se autentifica a una autoridad de autentificación a través de un mecanismo que usa una Smartcard con el añadido de clave privada y PIN.

- **Software PKI**:
Esta clase es aplicable cuando el principal usa un certificado X.509 almacenado en software con el fin de autentificarse a la correspondiente autoridad.

- **Telephony**:
Esta clase se usa para indicar que el principal se autentificó por medio de un número de teléfono de línea fija, transportado por un protocolo telefónico como ADSL.

- **Telephony (“Nomadic”)**:
Indica que el principal esta errante (quizás usando una tarjeta de teléfono) y se autentifica por medio de un número de línea, un sufijo de usuario y un elemento de contraseña.

- **Telephony (Personalized)**:
Esta clase se usa para indicar que el principal se autentificó por medio de un número de teléfono de línea fija y un sufijo de usuario, transportado por un protocolo telefónico como ADSL.

- **Telephony (Authenticated)**:
Indica que el usuario se autentificó por medio de un número de línea, un sufijo de usuario y un elemento de contraseña.

- **Secure Remote Password**:

Esta clase es aplicable cuando la autentificación se ha efectuado por medio de Secure Remote Password como se especifica en [RFC 2945].

- **SSL/TLS Certificate-Based Client Authentication**:

Esta clase indica que el principal fue autentificado por medio de un certificado de cliente, asegurado por un transporte SSL/TLS.

- **TimeSyncToken**:

Esta clase se aplica cuando el principal se autentificó a través de un testigo de sincronización de tiempo.

- **Unspecified**:

Esta clase indica que la autentificación fue realizada por medios que no se especifican.

9.6.4.- Esquema MobileTwoFactorRegistered (Contract)

En el siguiente esquema se muestran los elementos del esquema de la declaración de contexto de autentificación específico para dispositivos móviles de dos factores registrados, pudiendo ser por ejemplo, un dispositivo de firmado digital con memoria para almacenamiento de clave (GSM SIM), que necesita que haya un certificado de identidad de usuario explicito (PIN). Se ha elegido explicar este contexto debido a sus similitudes con el escenario que vamos a implementar.

De este tipo destacamos los siguientes elementos. Se explicaran un poco más abajo.

Elemento	*Descripción*	*Uso*
<Identification>	Características de la autentificación	Opcional
<TechnicalProtection>	Manera de guardar el “secreto”	Opcional
<OperationalPRotection>	Controles en proceso de seguridad	Opcional
<AuthnMethod>	Mecanismos de autentificación	Obligatorio
<GoverningAgreements>	Linkado de documentos externos	Opcional

Tabla 9. 25: Elementos resaltados del esquema **MobileTtwo Factor Registered**

- <Identification> (Opcional):

Referido a todas aquellas características que describen los procesos y mecanismos que la Autoridad de autentificación usa para crear inicialmente una asociación entre el principal y la identidad (el nombre) por el que será conocido el sujeto.

- <TechnicalProtection> (Opcional):

Son aquellas características que describen cómo se mantiene a salvo el “secreto” (el conocimiento ó la posesión de lo que permite al principal autentificarse en la Autoridad).

- <OperationalProtection> (Opcional):

Las características que describen los controles en el proceso de seguridad llevados a cabo por la citada Autoridad.

- <AuthnMethod> (Obligatorio):

Aquellas características que definen los mecanismos por los cuales el principal se autentificó a la Autoridad de autentificación.

- <GoverningAgreements> (Opcional):

Proporciona un mecanismo para el linkado de documentos externos (pudiendo ser documentos legibles por los humanos) en los que puede ser colocado los acuerdos adicionales del negocio (por ejemplo las obligaciones y las limitaciones a dichas obligaciones, etc.).

A continuación ponemos de manera informativa, las declaraciones de los diferentes tipos complejos que intervienen en la definición de clase de contexto *Mobile Two Factor Register*:

```
<xs:complexType name="AuthnContextDeclarationBaseType">
    <xs:complexContent>
      <xs:restriction base="AuthnContextDeclarationBaseType">
        <xs:sequence>
          <xs:element ref="Identification" minOccurs="0"/>
          <xs:element ref="TechnicalProtection" minOccurs="0"/>
          <xs:element ref="OperationalProtection" minOccurs="0"/>
          <xs:element ref="AuthnMethod"/>
          <xs:element ref="GoverningAgreements" minOccurs="0"/>
          <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="ID" type="xs:ID" use="optional"/>
      </xs:restriction>
    </xs:complexContent>
  </xs:complexType>
```

Figura 9. 55: Definición del tipo complejo **AuthnContextDeclarationBaseType**

```
<xs:complexType name="AuthnMethodBaseType">
    <xs:complexContent>
      <xs:restriction base="AuthnMethodBaseType">
        <xs:sequence>
          <xs:element ref="PrincipalAuthenticationMechanism" minOccurs="0"/>
          <xs:element ref="Authenticator"/>
          <xs:element ref="AuthenticatorTransportProtocol" minOccurs="0"/>
          <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:restriction>
    </xs:complexContent>
  </xs:complexType>
```

Figura 9. 56: Definición del tipo complejo **AuthnMethodBaseType**

```
<xs:complexType name="AuthenticatorBaseType">
    <xs:complexContent>
     <xs:restriction base="AuthenticatorBaseType">
      <xs:sequence>
       <xs:choice>
        <xs:element ref="DigSig"/>
        <xs:element ref="ZeroKnowledge"/>
        <xs:element ref="SharedSecretChallengeResponse"/>
        <xs:element ref="SharedSecretDynamicPlaintext"/>
        <xs:element ref="AsymmetricDecryption"/>
        <xs:element ref="AsymmetricKeyAgreement"/>
        <xs:element ref="ComplexAuthenticator"/>
       </xs:choice>
       <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
     </xs:restriction>
    </xs:complexContent>
  </xs:complexType>
```

Figura 9. 57: Definición del tipo complejo **AuthenticatorBaseType**

```
<xs:complexType name="ComplexAuthenticatorType">
    <xs:complexContent>
     <xs:restriction base="ComplexAuthenticatorType">
      <xs:sequence>
       <xs:choice>
        <xs:element ref="SharedSecretChallengeResponse"/>
        <xs:element ref="SharedSecretDynamicPlaintext"/>
       </xs:choice>
       <xs:element ref="Password"/>
      </xs:sequence>
     </xs:restriction>
    </xs:complexContent>
  </xs:complexType>
```

Figura 9. 58: Definición del tipo complejo **ComplexAuthenticatorType**

```
<xs:complexType name="AuthenticatorTransportProtocolType">
    <xs:complexContent>
     <xs:restriction base="AuthenticatorTransportProtocolType">
      <xs:sequence>
       <xs:choice>
        <xs:element ref="SSL"/>
        <xs:element ref="MobileNetworkNoEncryption"/>
        <xs:element ref="MobileNetworkRadioEncryption"/>
        <xs:element ref="MobileNetworkEndToEndEncryption"/>
        <xs:element ref="WTLS"/>
       </xs:choice>
       <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
     </xs:restriction>
    </xs:complexContent>
  </xs:complexType>
```

Figura 9. 59: Definición del tipo complejo **AuthenticatorTransportProtocolType**

```
<xs:complexType name="OperationalProtectionType">
    <xs:complexContent>
      <xs:restriction base="OperationalProtectionType">
        <xs:sequence>
          <xs:element ref="SecurityAudit"/>
          <xs:element ref="DeactivationCallCenter"/>
          <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:restriction>
    </xs:complexContent>
  </xs:complexType>
```

Figura 9. 60: Definición del tipo complejo **OperationalProtectionType**

```
<xs:complexType name="TechnicalProtectionBaseType">
    <xs:complexContent>
      <xs:restriction base="TechnicalProtectionBaseType">
        <xs:sequence>
          <xs:choice>
            <xs:element ref="PrivateKeyProtection"/>
            <xs:element ref="SecretKeyProtection"/>
          </xs:choice>
          <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:restriction>
    </xs:complexContent>
  </xs:complexType>
```

Figura 9. 61: Definición del tipo complejo **TechnicalProtectionBaseType**

```
<xs:complexType name="PrivateKeyProtectionType">
  <xs:complexContent>
    <xs:restriction base="PrivateKeyProtectionType">
      <xs:sequence>
        <xs:element ref="KeyActivation"/>
        <xs:element ref="KeyStorage"/>
        <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>
```

Figura 9. 62: Definición del tipo complejo **PrivateKeyProtectionType**

```
<xs:complexType name="SecretKeyProtectionType">
  <xs:complexContent>
    <xs:restriction base="SecretKeyProtectionType">
      <xs:sequence>
        <xs:element ref="KeyActivation"/>
        <xs:element ref="KeyStorage"/>
        <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>
```

Figura 9. 63: Definición del tipo complejo **SecretKeyProtectionType**

```
<xs:complexType name="KeyStorageType">
  <xs:complexContent>
    <xs:restriction base="KeyStorageType">
      <xs:attribute name="medium" use="required">
        <xs:simpleType>
          <xs:restriction base="mediumType">
            <xs:enumeration value="MobileDevice"/>
            <xs:enumeration value="MobileAuthCard"/>
            <xs:enumeration value="smartcard"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>
```

Figura 9. 64: Definición del tipo complejo **KeyStorageType**

```
<xs:complexType name="SecurityAuditType">
  <xs:complexContent>
    <xs:restriction base="SecurityAuditType">
      <xs:sequence>
        <xs:element ref="SwitchAudit"/>
        <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>
```

Figura 9. 65: Definición del tipo complejo **SecurityAuditType**

```
<xs:complexType name="IdentificationType">
  <xs:complexContent>
    <xs:restriction base="IdentificationType">
      <xs:sequence>
        <xs:element ref="PhysicalVerification"/>
        <xs:element ref="WrittenConsent"/>
        <xs:element ref="GoverningAgreements"/>
        <xs:element ref="Extension" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="nym">
        <xs:simpleType>
          <xs:restriction base="nymType">
            <xs:enumeration value="anonymity"/>
            <xs:enumeration value="verinymity"/>
            <xs:enumeration value="pseudonymity"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>
```

Figura 9. 66: Definición del tipo complejo **IdentificationType**