Tecnologías avanzadas para el desarrollo de Web Services

Parte 2

Agenda

- WS-SecureConversation
- WS-Trust
- WS-Addressing
- MTOM
- WS-ReliableMessaging

WS-SecureConversation

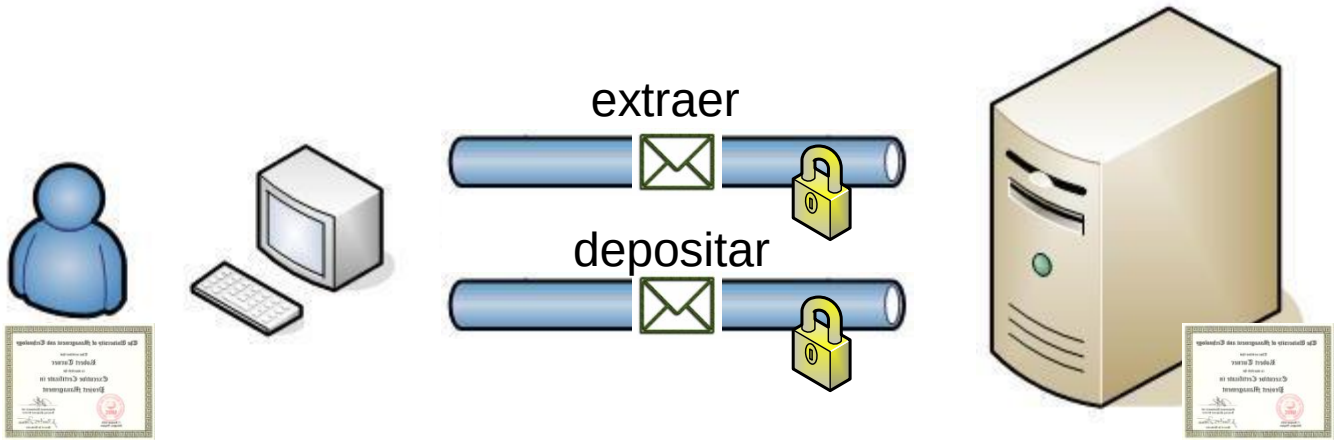
Sesiones seguras entre Web Services

Motivación

- WS-Security
  - Provee mecanismos para asegurar la comunicación de UN mensaje SOAP.
  - Generalemente se utiliza con mecanismos de cifrado asimétricos.
    - Infraestructura de clave pública (PKI).
  - Autenticación por mensaje.

- ¿Qué ocurre en escenarios con más de un mensaje y a un mismo servicio? ¿Sigue siendo WS-Security la mejor opción?

Escenario: Transferencia de fondos



- WS-Security
  - Doble autenticación
  - Cifrado asimétrico
- SSL
  - Una sola autenticación
  - Cifrado simétrico

SSL parece ser la mejor opción!

Sin embargo…

- Cifrado y firmado de mensajes sigue siendo punto a punto
- Autenticación de usuarios sigue siendo punto a punto
- El cifrado sigue siendo de todo el mensaje

- Sería deseable WS-Security pero a nivel de varios mensajes!
  - Una sesión segura tipo SSL/TLS, pero con WS
  - ¿WS-SecureConversation? ¿Qué es eso?

WS-SecureConversation

- Estándar de la OASIS
  - Actualmente en versión 1.4
- Extiende WS-Security definiendo un protocolo para el establecimiento de una sesión segura con el servicio
- Entre cliente y servicio se define una clave simétrica que se utiliza para autenticación, firma y/o cifrado de los mensajes.
- Es “SSL para Web Services”.

Derivación de claves

- Problema:
  - Misma clave (simétrica) utilizada varias veces en una comunicación segura
  - Existen varios textos cifrados con la misma clave.
  - Aumenta probabilidad de adivinarla ☹
- WS-SecureConversation define un algoritmo para la derivación de claves
  - Se deriva una nueva clave simétrica de la “original”.
  - Cada mensaje tiene su propia clave simétrica.
  - La clave original nunca se utiliza.
  - Disminuye probabilidad de adivinarla ☺

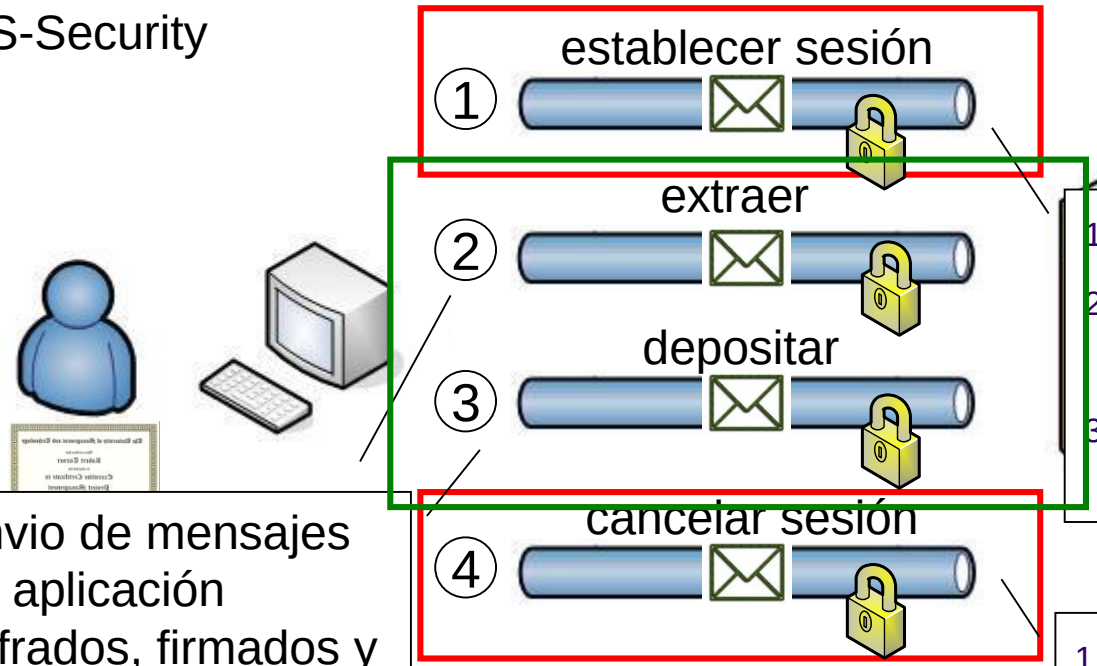
SSL vs WS-SecureConversation

Características	SSL	WS-SecureConversation
Estándar	Sí	Sí
Seguridad	A nivel de transporte	A nivel de applicación
Confidencialidad	Todo el mensaje.	Todo o partes del mensaje
Autenticación de clientes	X.509, Basic Authentication	Usuario/Passwd, X.509, Kerberos, SAML.
Protocolo	Potencialmente varios. HTTP el más popular.	Potencialmente varios. HTTP el más popular.
Algoritmos de cifrado	Simétricos	Simétricos
Sesiones seguras	Sí	Sí

Escenario: Transferencia de fondos

WS-SecureConversation

WS-Security



1. Envio de mensajes de aplicación (cifrados, firmados y autenticados).
2. Id de sesión adjunto en el mensaje

1. Autenticación cliente
2. Se define clave simétrica
3. Se identifica la sesión segura

1. Cancelar sesión
2. Id de sesión adjunto en el mensaje

Escer
Trans

WS-Secur

WS-Secur

1. Envio de r
   de aplicac
   (cifrados, f
   autenticad
2. Id de sesi
   en el mens

```
<envelope>
<header>
  …
  <Security>
    ...
    <SecurityContextToken>
      <Identifier>urn:uuid:1e4f01b…57624a</Identifier>
    </SecurityContextToken>
    <Signature>...</Signature>
  </Security>
</header>
<body>
  <depositar>
    <EncryptedData><CipherData>
      <CipherValue>A23B4…5C56</CipherValue>
    </CipherData></EncryptedData>
    <fecha>09-03-2009</fecha>
  </depositar>
</body>
</envelope>
```

Security Token

Id sesión

Firma

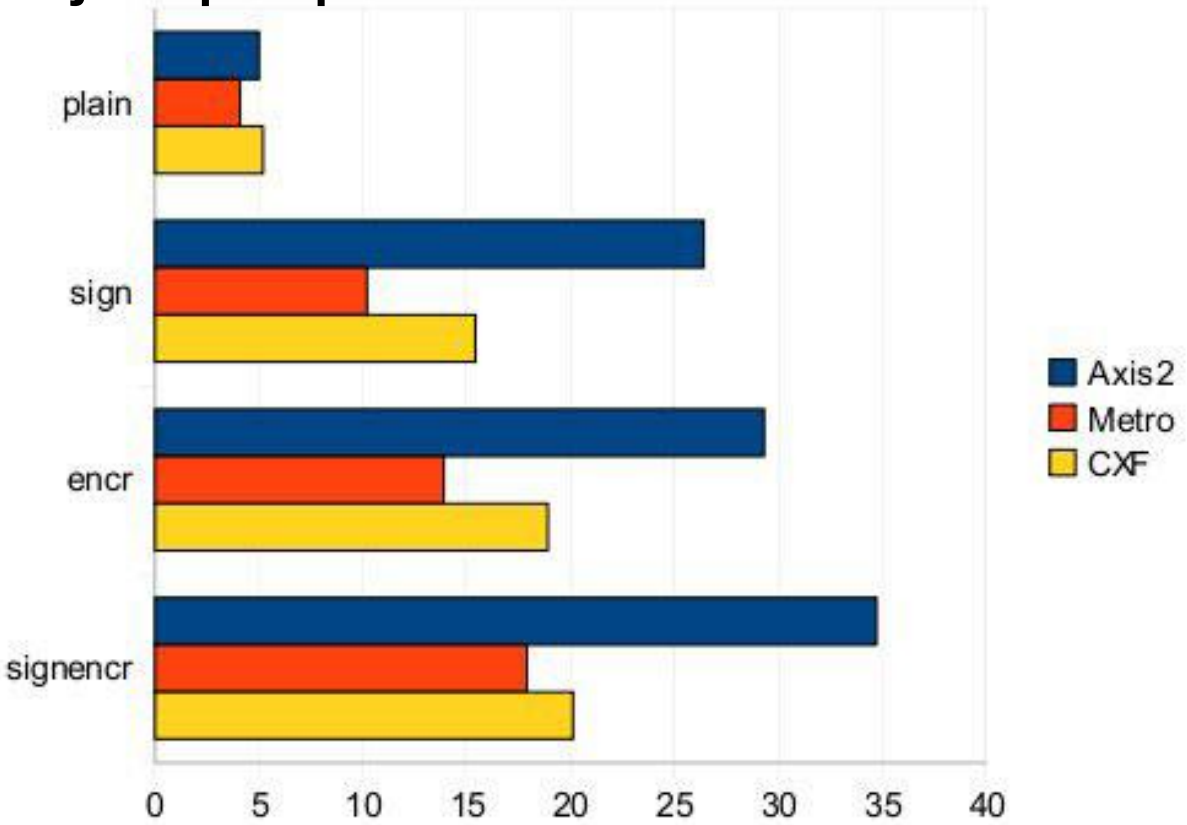
Cifrado

nticación cliente
efine clave
trica
lentifica la
ón segura

elar sesión
sesión adjunto
mensaje

Algunos números…

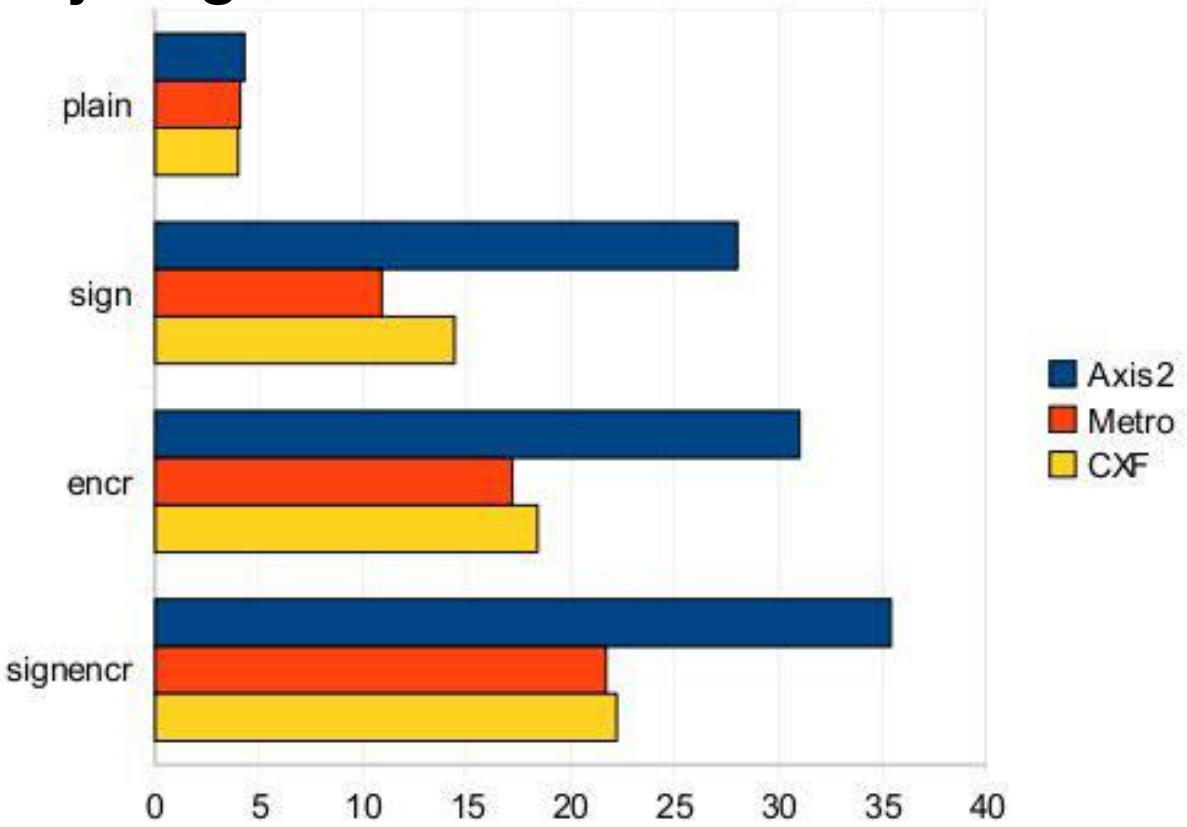
- Mensajes pequeños



http://www.ibm.com/developerworks/java/library/j-jws16/index.html

Algunos números…

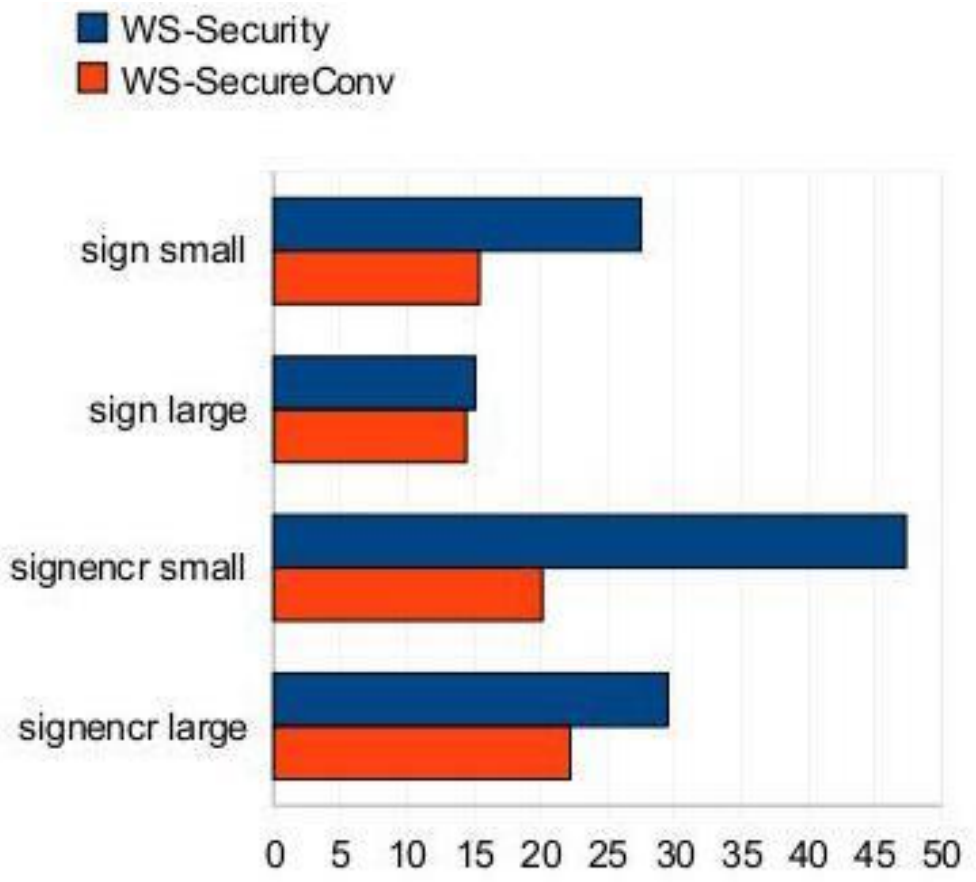
- Mensajes grandes



http://www.ibm.com/developerworks/java/library/j-jws16/index.html

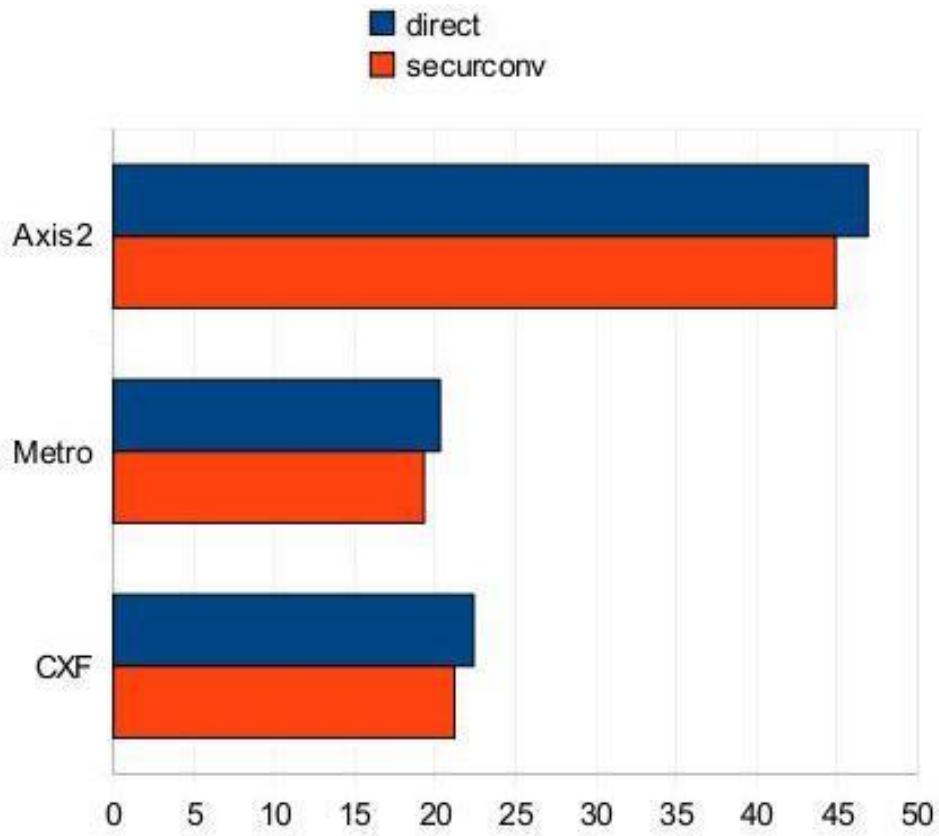
WS-Security vs WS-SecureConv.

- Implementación: CXF
- WS-Security
  - Cifrado asimétrico
- WS-SecureConv.
  - Cifrado simétrico



http://www.ibm.com/developerworks/java/library/j-jws16/index.html

WS-Sec. y cifrado simétrico



- Mensajes grandes
- WS-Security
  - Cifrado simétrico
  - Clave generada en cada llamada
  - Clave cifrada con algoritmos asimétricos
- WS-SecureConv.
  - Cifrado simétrico
  - Clave generada una única vez

http://www.ibm.com/developerworks/java/library/j-jws17/index.html

WS-Trust

Seguridad basada en confianza

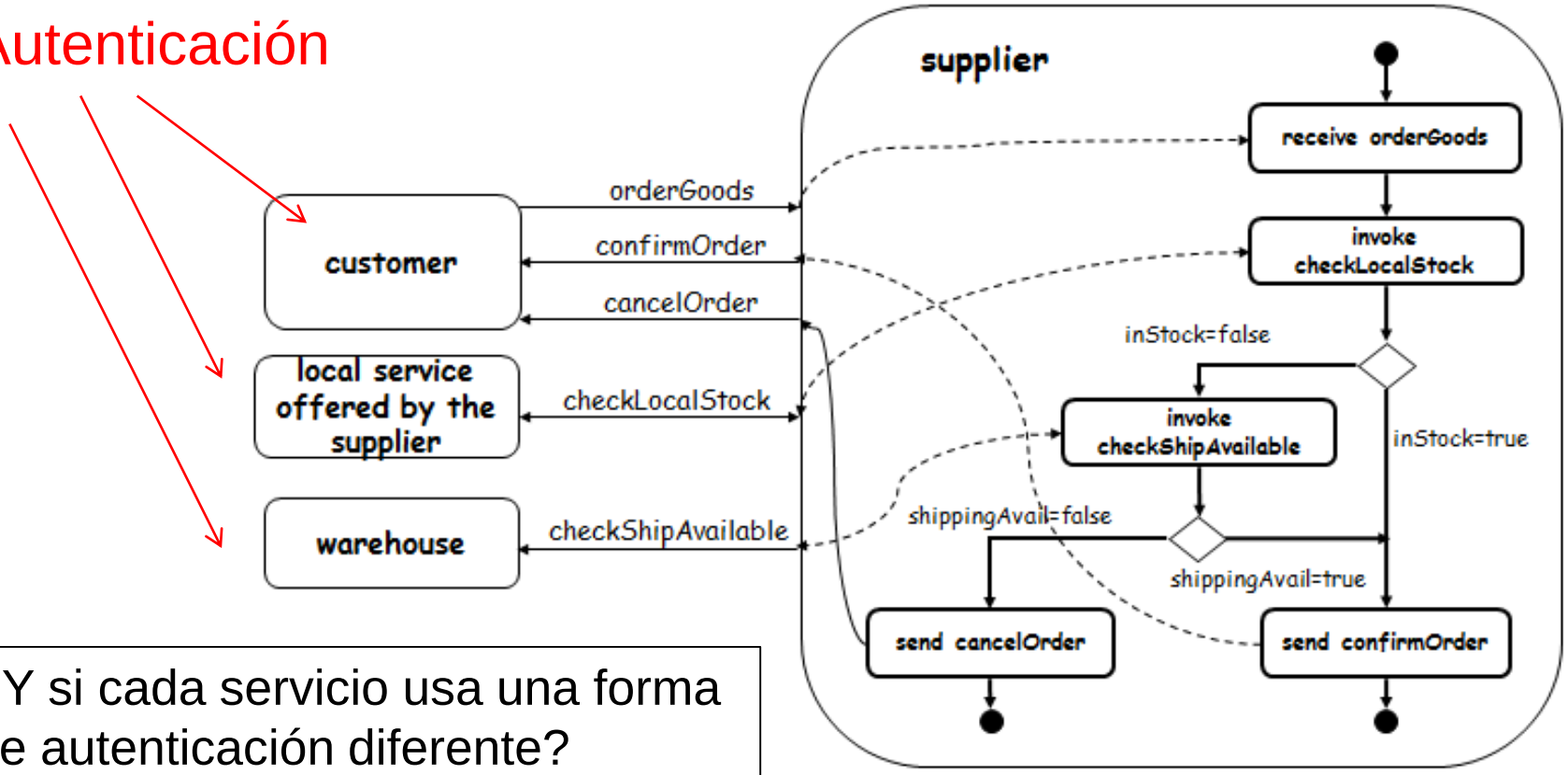
Escenario

- SOC propone el uso de composición de servicios para implementar procesos de negocio

- ¿Cómo podemos resolver la autenticación individual de los servicios?
- ¿Cómo podemos resolver la autenticación del servicio encargado de la composición?

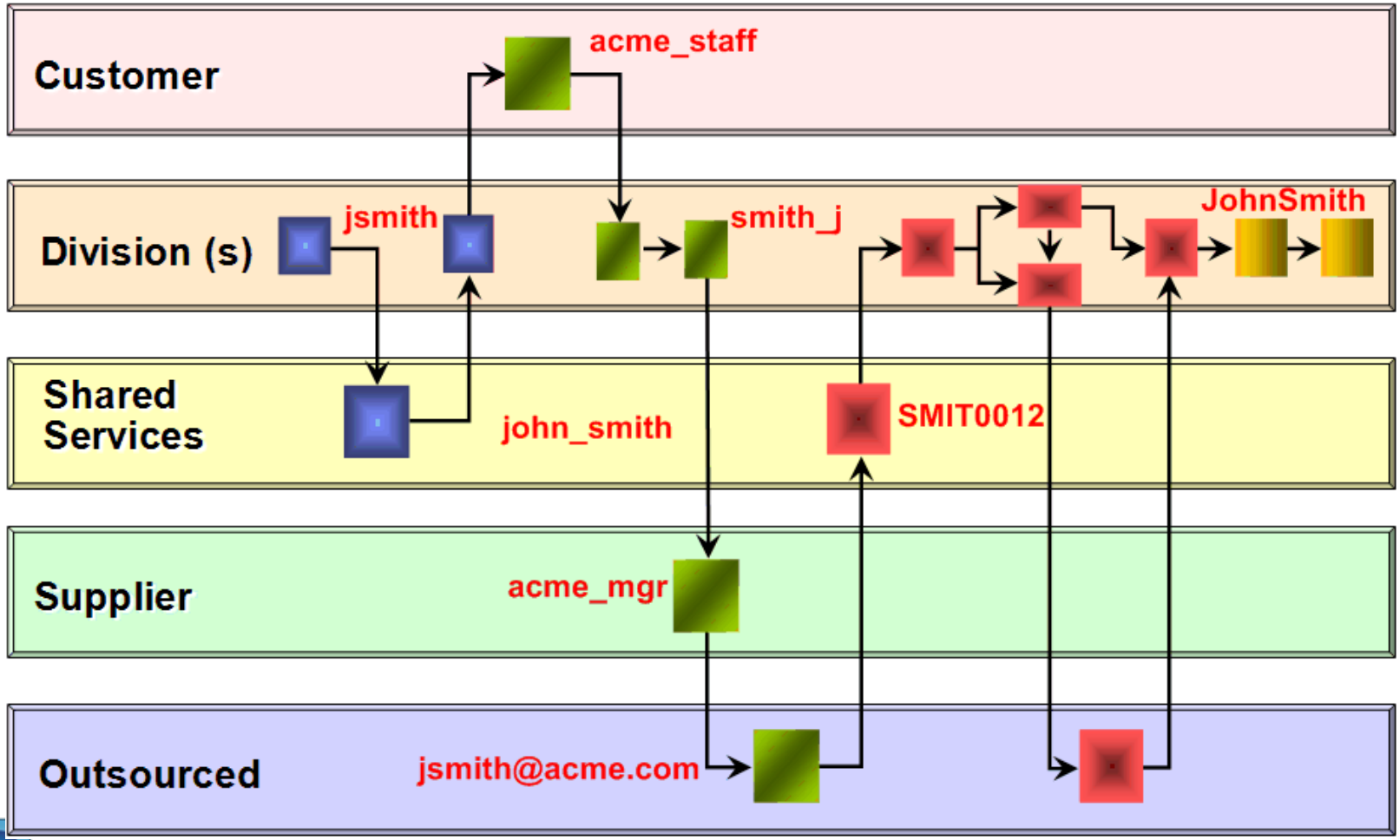
Composición de servicios

Autenticación



¿Y si cada servicio usa una forma de autenticación diferente?

Propagación de la identidad



Propagación de la identidad

- Composición de servicios de bajo acoplamiento para la construcción de nuevas aplicaciones
- Cada uno de estos servicios posee su propio directorio de usuarios que frecuentemente son administrados de forma aislada uno de otro
  - De acuerdo a un estudio realizado por Forrester Research, una organización de gran tamaño posee en promedio 181 repositorios de usuarios.
- Dentro de una misma organización, los usuarios suelen tener diferentes identidades dentro de los diferentes servicios que componen un proceso de negocios

Propagación de la identidad

- Establecer la identidad del consumidor del servicio es una tarea fundamental antes de poder implementar otros requerimientos como autorización, auditoría, etc.

- Son necesarios “servicios de identidades” que permitan fácilmente componer servicios propagando la correctamente las identidades

Desafíos

- Transformación de formatos
  - Capaz de comprender y operar diferentes formatos para representar una identidad
- Mapping de credenciales
  - Capaz de hacer la tradución entre diferentes identidades
  - P. ej: John.Smith a JSmith
- Diseño basado en SOA
  - Permitir desacoplar la lógica de negocio con la “lógica de seguridad”
- Interoperabilidad
  - Estar basado en el uso de estándares abiertos para fomentar la interoperabilidad entre plataformas

Security as a Service

- En el contexto de SOA, las aplicaciones ya no pueden ser responsables de autenticar y autorizar
  - Pueden no conocer el contexto de invocación

- Para esto, introducimos una entidad por fuera de las aplicaciones, un servicio nuevo, encargado de la seguridad

WS-Trust

- Estándar de la OASIS
  - Actualmente, versión 1.4

- Extensión a WS-Security que:
  - Define qué es un Security Token Service (STS)
    - Permite emitir, validar, cancelar tokens de seguridad
  - Define métodos para establecer, evaluar e intermediar relaciones de “confianza”:
    - **… characteristic that one entity is willing to rely upon a second entity to execute a set of actions and/or to make set of assertions about a set of subjects and/or scopes…**

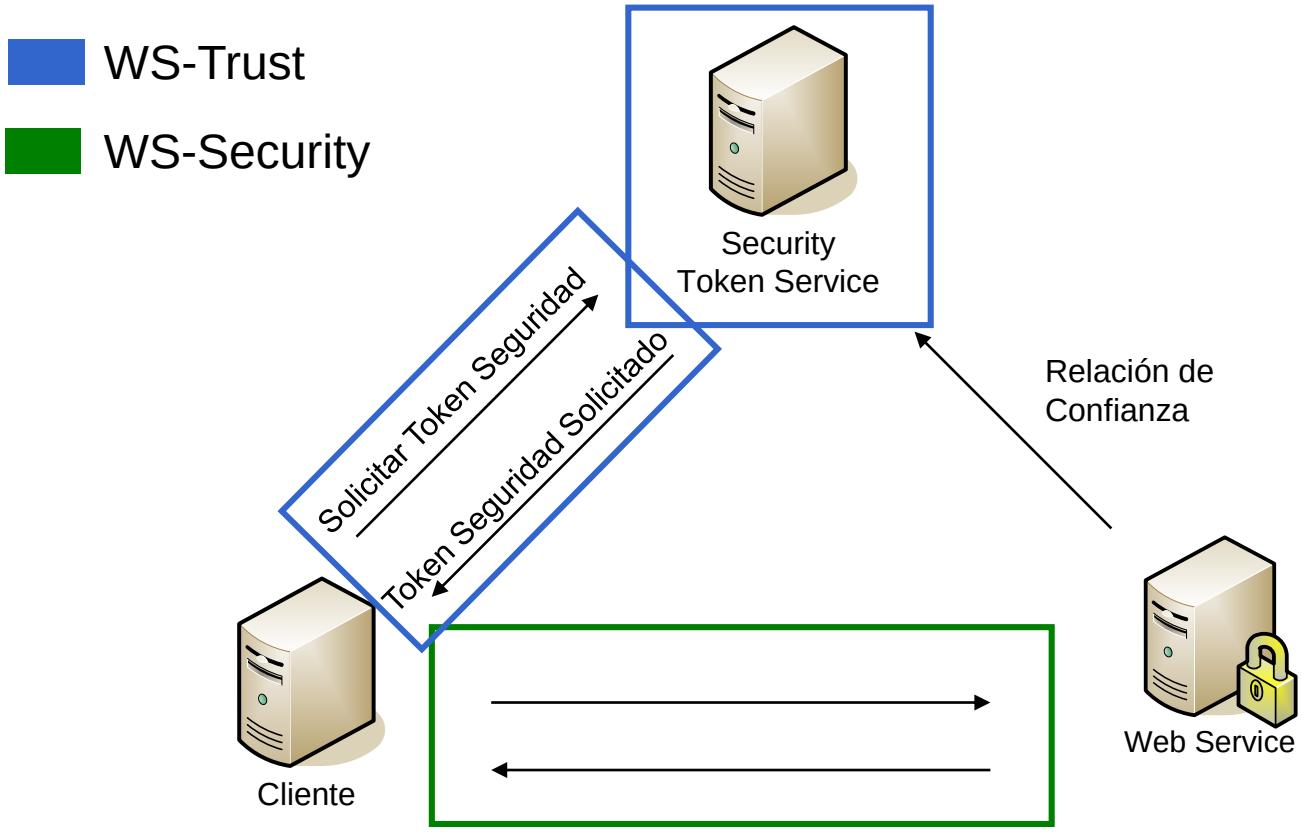
Security Token Service

- Un *security token service (STS)* es un Web Service que emite tokens de seguridad.
  - Equivale a hacer afirmaciones basadas en pruebas confiables para quienquiera que confíe en él (o destinatarios específicos).

- Los servicios que confían en el STS requieren pruebas que las afirmaciones son confiables
  - Tokens de seguridad deberán estar firmados (por el STS) para garantizar su procedencia.

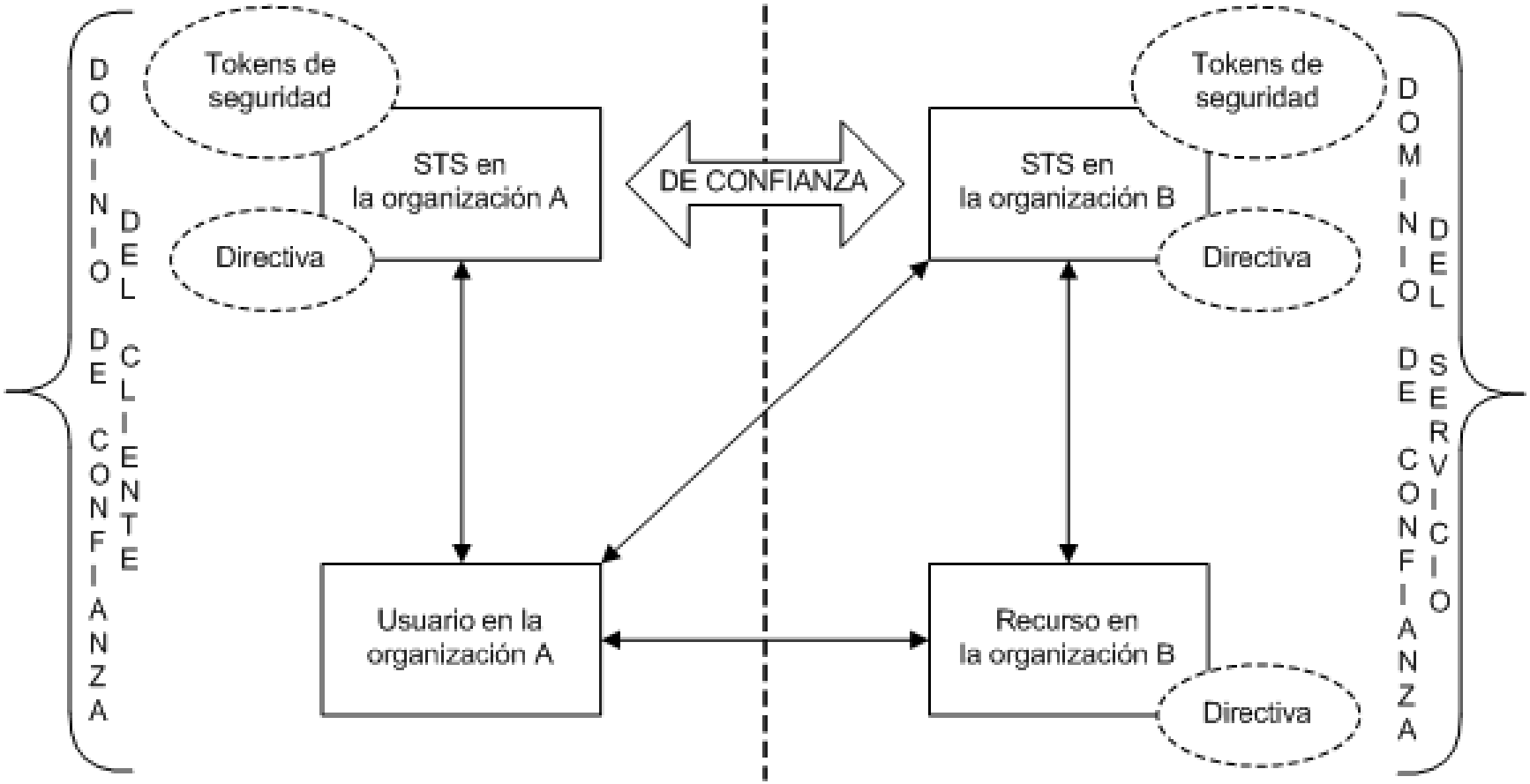
- Los STS cuentan con las siguientes operaciones:
  - Issue, renew, validate, cancel

Modelo de confianza



Misma fuente de confianza
Relación de confianza altamente acoplada
Útil dentro de una misma organización

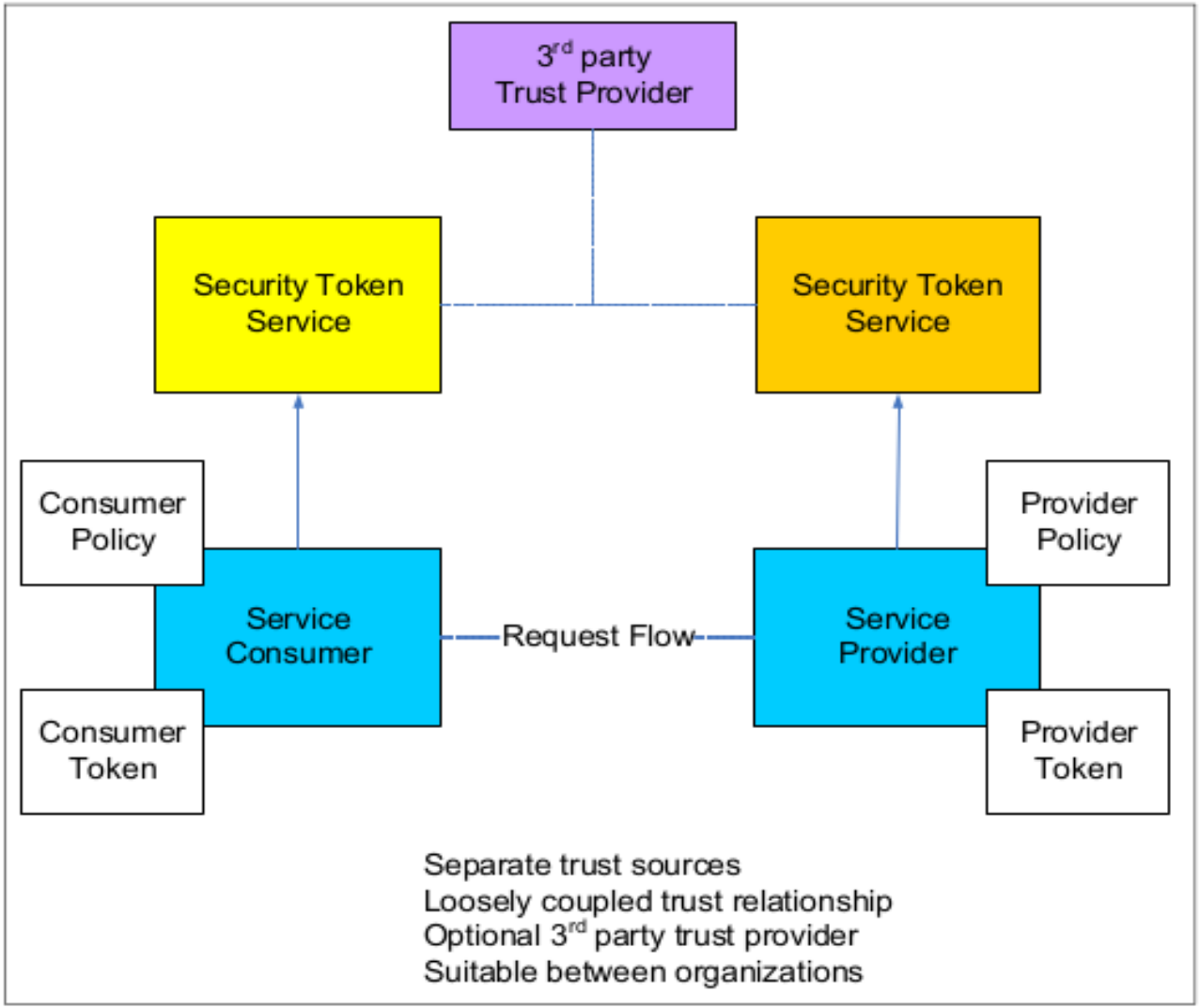
Seguridad federada



Seguridad federada

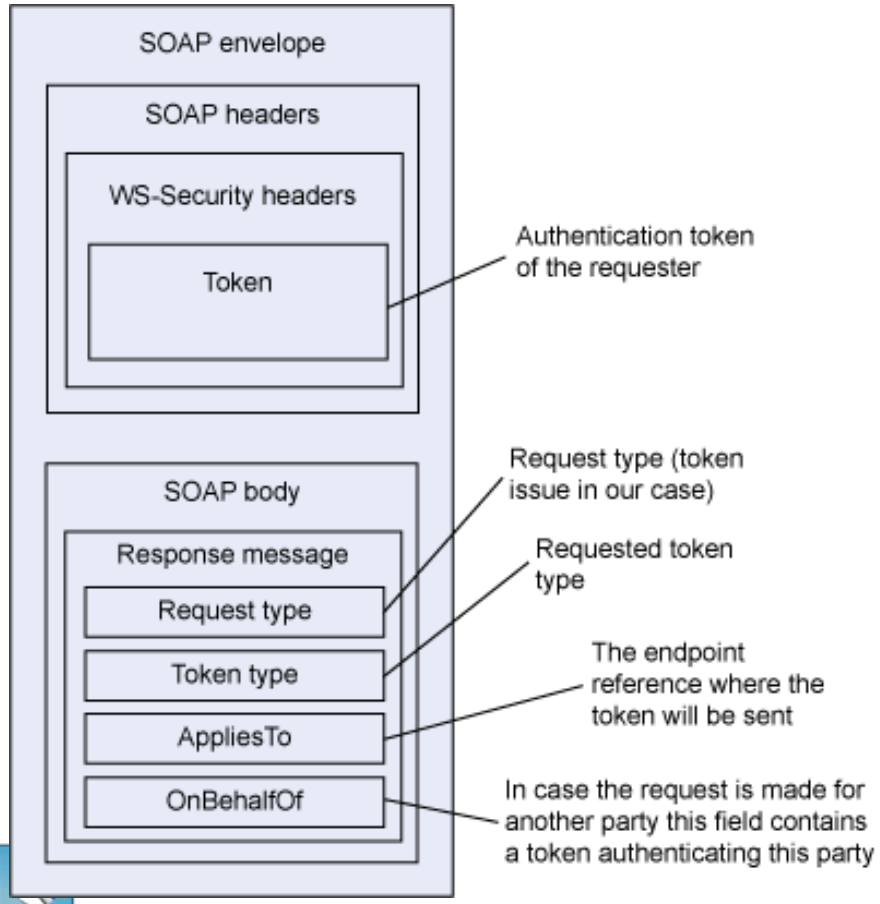
1. Los usuarios se ponen en contacto con el STS de la organización A y obtienen un token de seguridad presentando las credenciales de autenticación que utilizan normalmente para obtener acceso a cualquier otro recurso de la organización A. Esto reduce el problema de que los usuarios tengan que mantener varios conjuntos de credenciales o que usen el mismo conjunto de credenciales en varios sitios de servicios.
2. Una vez que los usuarios obtienen un token de seguridad del STS de A, presentan el token al STS de B. La organización B continúa con la autorización de las solicitudes de los usuarios y emite un token de seguridad a los usuarios desde su propio conjunto de tokens de seguridad.
3. Los usuarios pueden presentar a continuación el token emitido por el STS B al recurso de la organización B y obtener acceso al servicio.
4. El servicio confía en los token emitidos por el STS o consulta la validez del mismo al STS

Seguridad federada

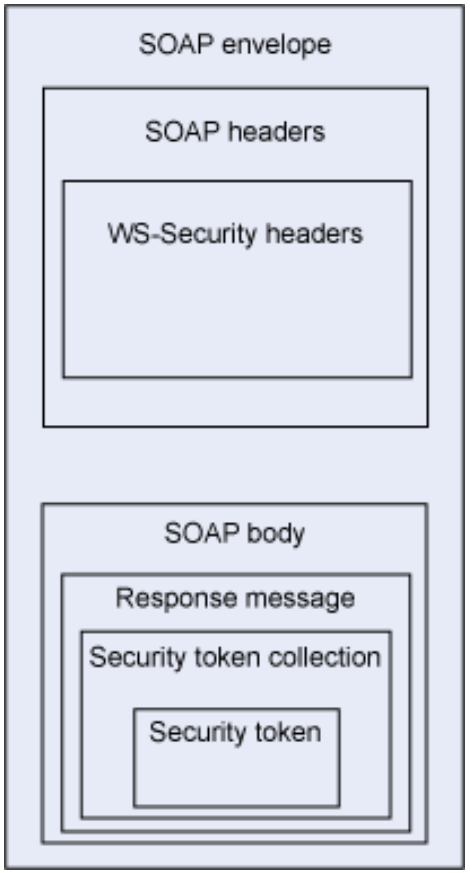


Ejemplos SOAP

WS-Trust Request



WS-Trust Response



```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
   xmlns:wsu="http://.../oasis-200401-wss-wssecurity-utility-1.0.xsd"
   xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:wsse="http://.../oasis-200401-wss-wssecurity-secext-1.0.xsd">
  <S:Header>
    <wsse:Security S:mustUnderstand="1">
      <wsse:BinarySecurityToken wsu:Id="SecurityToken-id"
    EncodingType="http://.../oasis-200401-wss-soap-message-security-1.0#Base64Binary"
    ValueType="http://.../oasis-200401-wss-x509-token-profile-1.0#X509v3">
       Gateway X509 Certificate
      </wsse:BinarySecurityToken>
      <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
        <SignedInfo>
          ...
        </SignedInfo>
        <SignatureValue>
          ...
        </SignatureValue>
        <KeyInfo>
          <wsse:SecurityTokenReference>
            <wsse:Reference URI="#SecurityToken-id"
          ValueType="http://.../oasis-200401-wss-x509-token-profile-1.0#X509v3"/>
          </wsse:SecurityTokenReference>
        </KeyInfo>
      </Signature>
      ...
    </wsse:Security>
  </S:Header>
  <S:Body>
    <RequestSecurityToken xmlns="http://docs.oasis-open.org/ws-sx/ws-trust/200512"
     xmlns:wsa="http://www.w3.org/2005/08/addressing"
     xmlns:pol="http://schemas.xmlsoap.org/ws/2004/09/policy">
      <RequestType>http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue</RequestType>
      <pol:AppliesTo>
        <wsa:EndpointReference>
          <wsa:Address>http://testHost/HelloService</wsa:Address>
        </wsa:EndpointReference>
      </pol:AppliesTo>
      <OnBehalfOf>
        <wsse:UsernameToken
        xmlns:wsse="http://.../oasis-200401-wss-wssecurity-secext-1.0.xsd">
          <wsse:Username>jdoe</wsse:Username>
        </wsse:UsernameToken>
      </OnBehalfOf>
      <TokenType>http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0</
    </RequestSecurityToken>
  </S:Body>
</S:Envelope>
```

Token de seguridad

Tipo de request

Propósito

Tipo de token solicitado

Problema…

- Credenciales tradicionales son poco descriptivas
  - Usuario/contraseña, certificados X.509, etc

- En determinados casos sería deseable contar con información extra que permita dar mayor contexto sobre el usuario
  - P. ej: grupos a los que pertenece el usuario

Solución ideal

- Un token avanzado de seguridad con:
  - Identificación del cliente
  - Grupos a los que pertenece el usuario
  - Correo electrónico

Security Assertion Markup Language (SAML)

- Estándar de la OASIS
  - Actualmente en versión 2.0

- SAML define un *framework* para intercambiar información de autenticación y autorización entre dominios de seguridad en la forma de *assertions*, en lugar de utilizar identidades o *tokens* de seguridad tradicionales. Está compuesto por tres secciones:
  - SAML Core
  - SAML Binding
  - SAML Protocol

- Nosotros nos centraremos en SAML Core. En particular, en las SAML Assertions

SAML Assertion

- Un SAML Assertion es un documento XML compuesto por información de seguridad (statements) que puede ser utilizado para realizar el control de acceso
  - Authentication statements:
    - El sujeto fue autenticado ante el emisor del token SAML a una hora determinada y utilizando un determinado mecanismo de autenticación
  - Attribute statements:
    - Información acerca del sujeto de la forma (atributo, valor)
  - Authorizations decision statements:
    - El sujeto se encuentra autorizado (o no autorizado) a realizar determinadas acciones

Ejemplo

Emisor

Validez y propósito

```
<saml:Assertion AssertionID="Assertion-uuid2765608b-0128-1ec9-aea1-8913af9af38d"
    IssueInstant="2010-04-22T21:21:14Z" Issuer="STS A" MajorVersion="1" MinorVersion="1"
    xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion">
  <saml:Conditions NotBefore="2010-04-22T21:06:14Z" NotOnOrAfter="2010-04-22T21:36:14Z">
    <saml:AudienceRestrictionCondition>
      <saml:Audience>http://www.dgi.gub.uy/validacionRut</saml:Audience>
    </saml:AudienceRestrictionCondition>
  </saml:Conditions>
  <saml:AuthenticationStatement AuthenticationInstant="2010-04-22T21:21:14Z"
    AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:password">
    <saml:Subject>
      <saml:NameIdentifier>gllambias</saml:NameIdentifier>
    </saml:Subject>
  </saml:AuthenticationStatement>
  <saml:AttributeStatement>
    <saml:Attribute AttributeName="emailAddress">
      <saml:AttributeValue>gllambias@fing.edu.uy</saml:AttributeVal
    </saml:Attribute>
    <saml:Attribute AttributeName="groups">
      <saml:AttributeValue>LINS</saml:AttributeValue>
      <saml:AttributeValue>Teacher</saml:AttributeValue>
    </saml:Attribute>
  </saml:AttributeStatement>
  <ds:Signature Id="uuid2765608c-0128-1428-9ce4-8913af9af38d"
    xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
    …
  </ds:Signature>
</saml:Assertion>
```

Certificado de autenticación

Certificado de info requerida

Firma digital del STS A

Autorización

- Una vez autenticado el usuario, debemos determinar si éste puede o no acceder a la funcionalidad solicitada
- Esto se denomina también “control de acceso”
- La decisión de si lo dejamos acceder o no, depende de muchos factores
  - Quién es, el rol que tiene, qué tipo de recurso quiere acceder

Autorización

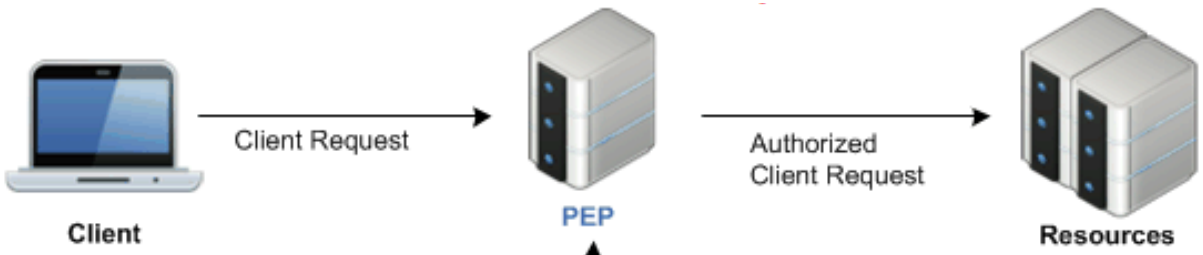
- Enfoques tradicionales
  - RBAC: Role Based Access Control
  - ACL: Access Control List
- En el caso de SOA, estas estrategias no funcionan, si la información de seguridad de cada servicio no se encuentra fuera del mismo
- Si ésta es opaca a la composición de servicios, entonces la reusabilidad del servicio se ve afectada

XACML

- Lenguaje basado en XML para autorización de Web Services
- Permite definir políticas de control de acceso
- Define un modelo de procesamiento que describe cómo evaluar peticiones de acceso de acuerdo a las políticas de control de acceso definidas
  - P.ej: un usuario/rol/grupo puede consumir determinado servicio

Autorización SOA

Servicio encargado del cumplimiento de las decisiones del PDP



Servicio encargado de autorizar los mensajes

Estándar para la especificación de políticas de control de acceso en xml

Autorización SOA

- Policy Definition Point (PDP)
  - Servicio encargado de definir las políticas XACML
- Policy Enforcement Point (PEP)
  - Servicio encargado de hacer cumplir las políticas definidas en el PDP

Mensajería

WS-Addressing, MTOM

SOAP

- Provee una forma estándar de estructurar mensajes utilizando XML
- Neutralidad en protocolo de transporte
- Especifica un modelo de procesamiento que indica cómo se deben procesar los mensajes
- Una forma de adjuntar datos no-XML a los mensajes.

SOAP

- Provee una forma estándar de estructurar mensajes utilizando XML
- Neutralidad en protocolo de transporte
- Especifica un modelo de procesamiento que indica cómo se deben procesar los mensajes
- Una forma de adjuntar datos no-XML a los mensajes.

Neutralidad en transporte

- SOAP no impone el uso de un determinado protocolo para el intercambio de mensajes

- A través del concepto de "*binding*" SOAP permite especificar:
  - cómo los mensajes SOAP se encapsulan en un protocolo de transporte
  - cómo los mensajes SOAP deben ser tratados con las primitivas del protocolo

Sin embargo…

- Existen dependencias en protocolos de transporte

Sin embargo…

- Existen dependencias en los protocolos de transporte
  - HTTP URI
    - Determinar dirección del Web Service
  - Cabezales http: SoapAction
    - Determinar el método del Web Service
    - Creador de problemas de interoperabilidad
    - Obsoleto a partir de SOAP 1.2!

Debilidades

- Necesidad de colocar información en el protocolo de forma estándar
  - Dirección del servicio
  - Operación a consumir
  - Otra información
- Dificultad al enviar mensajes a través de intermediarios utilizando diferentes protocolos
- Mecanismos de seguridad “atados” al protocolo de transporte
- Solución:
  - colocar información en cabezales SOAP
  - Mantener neutralidad y eliminar dependencias en protocolo de transporte

WS-Addressing

WS-Addressing

- Estándar de la W3C
  - Actualmente en versión 1.0

- Propone un mecanismo estándar e independiente del transporte para:
  - Referenciar Web Services
  - Direccionar mensajes

Endpoint Reference

- Es un “puntero” a un Web Service.
- Especifica:
  - Address
  - Reference Parameters
  - Metadata.

```
<wsa:EndpointReference xmlns:wsa="..." xmlns:example="...">
    <wsa:Address>http://example.com/weather</wsa:Address>
    <wsa:ReferenceProperties>
        <example:ServiceLevel>Basic</example:ServiceLevel>
    </wsa:ReferenceProperties>
    <wsa:ReferenceParameters>
        <example:CityCode>NYC</example:CityCode>
    </wsa:ReferenceParameters>
</wsa:EndpointReference>
```

Message Addressing Props

- Definen el conjunto de cabezales SOAP con información para el direccionamiento de mensajes.

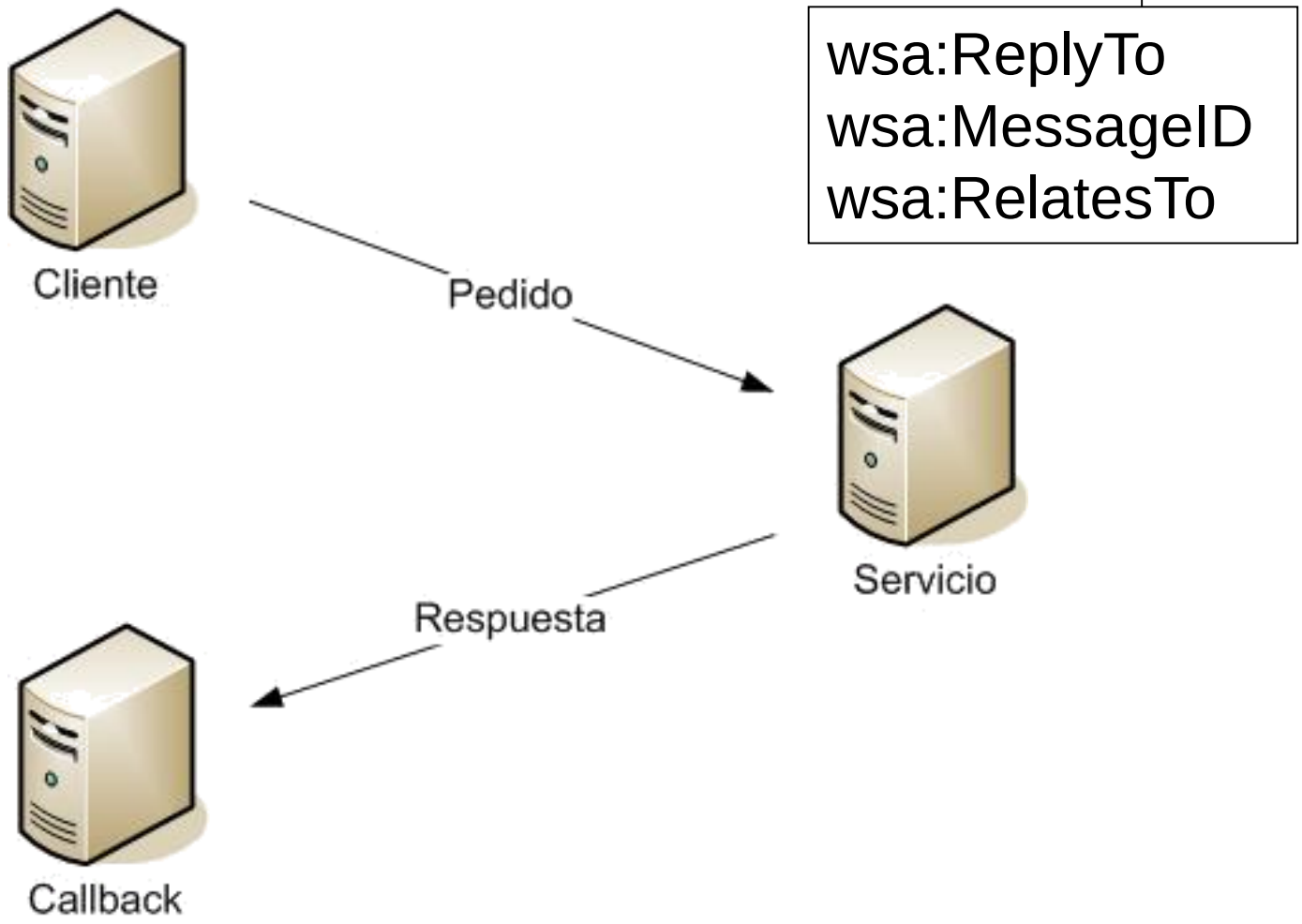
- Elementos requeridos: *To* y *Action*
  - **To**: Indica a donde se envía el pedido.
  - **Action**: Indica la acción a realizar por el receptor del mensaje.

```
<!-- Message 1 -->
    <wsa:To>mailto:ws0example.com</wsa:To>
    <wsa:Action>http://example.com/aservice</wsa:Action>
// ...
```

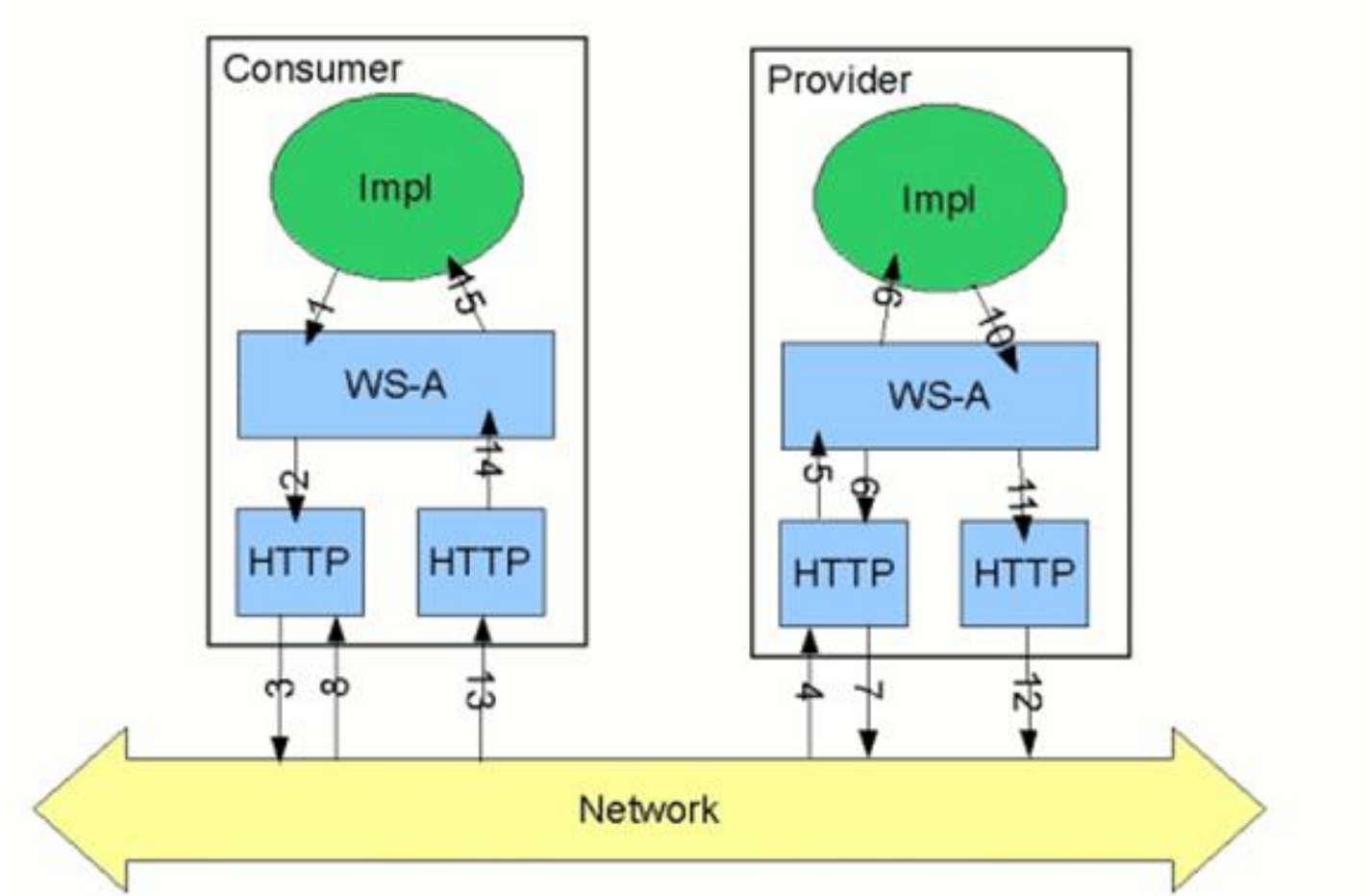
Message Addressing Props

- **ReplyTo**
  - Especifica la dirección de un EPR al que se debe enviar la respuesta a una solicitud.
- **FaultTo**
  - Especifica la dirección de un EPR al que se debe invocar cuando ocurra una situación anómala y no se pueda terminar de procesar la solicitud
- **MessageID**
  - String que identifica de forma univoca el mensaje.
- **RelatesTo**
  - String que permite correlacionar mensajes
  - Complementa ReplyTo para dar solución al uso de respuestas desacopladas entre WS.

Respuestas desacopladas



Respuestas desacopladas



Ejemplo de solicitud

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
    xmlns:wsa="http://www.w3.org/2005/08/addressing">
  <S:Header>
    <wsa:To>http://example.com/fabrikam</wsa:To>
    <wsa:Action>http://example.com/fabrikam/Delete</wsa:Action>
    <wsa:MessageID>http://example.com/someuniquestring</wsa:MessageID>
    <wsa:ReplyTo>
          <wsa:Address>http://example.com/business/client1</wsa:Address>
    </wsa:ReplyTo>
  </S:Header>
  <S:Body>
    <f:Delete xmlns:f="http://example.com/fabrikam">
      <maxCount>42</maxCount>
    </f:Delete>
  </S:Body>
</S:Envelope>
```

Ejemplo de respuesta

```
<S:Envelope xmlns:S="http://www.w3.org/2003/05/soap-envelope"
    xmlns:wsa="http://www.w3.org/2005/08/addressing">
  <S:Header>
    <wsa:To>http://example.com/business/client1</wsa:To>
    <wsa:Action>http://example.com/fabrikam/DeleteAck</wsa:Action>
     <wsa:MessageID>http://example.com/otheruniquestring</wsa:MessageID>
    <wsa:RelatesTo>http://example.com/someuniquestring</wsa:RelatesTo>
</S:Header>
  <S:Body>
    <f:DeleteAck xmlns:f="http://example.com/fabrikam"/>
  </S:Body>
</S:Envelope>
```

Aspectos destacables

- Identificación de mensajes

- Respuestas desacopladas
  - Respuestas correctas
  - SOAP Faults
  - Correlación de mensajes

MTOM

Message Transmission Optimization Mechanism

Motivación

- SOAP
  - Provee una forma estándar de estructurar mensajes utilizando XML
  - Define mecanismos para utilizar distintos protocolos de transporte para el envío de mensajes
  - Especifica un modelo de procesamiento que indica cómo se deben procesar los mensajes
  - Una forma de adjuntar datos no-XML a los mensajes.

Cómo lo hace?

- SOAP utiliza la codificación a Base64 para convertir datos binarios a XML

- XMLSchema soporta el uso de este tipo de serialización
  - tipo de datos *xsd:base64binary*

Codificación base64

- Base 64 toma los datos binarios y los convierte en una serie de caracteres ASCII
- Toma 3 octetos de bits y los convierte en 4 caracteres del estándar ASCII
  - Se usan solo 64 caracteres del estándar
  - a-z, A-Z, 0-9, + y /
- Ejemplo
  - Mary had a little lamb...
  - TWFyeSBoYWQgYSBsaXR0bGUgbGFtYi4uLiA=

Codificación base64

Texto ASCII	Mary had
Representación hexadecimal	4D 61 72 79 20 68 61 64
Representación binaria agrupada en bytes	01001101 01100001 01110010 01111001 00100000 01101000 01100001 01100100
Representación binaria agrupada de a 6 bits	010011 010110 000101 110010 011110 010010 000001 101000 011000 010110 010000=
Representación decimal de los bloques de 6 bits	19 22 05 50 30 18 01 40 24 22 16=
Codificación base64	TWFyeSBoYWQ=

Sin embargo…

- Tamaño del texto un %25 más grande
- En base64 6 bits son codificados con 8 bits
  - Base64(010011) = T (8 bits)
- Gran impacto en datos binarios de gran tamaño!

MTOM (Message Transmission Optimization Mechanism)

- MTOM es un estándar de la W3C con el propósito de optimizar la transferencia de archivos binarios entre Web Services.

- MIME gran protagonista!

MTOM

- Compuesto por:
  - Abstract SOAP Transmission Optimization Feature
    - mecanismo de optimización abstracto para mensajes SOAP
  - Optimized MIME multipart/Related Serialization of SOAP messages
    - Implementación basada en XOP e indep. del medio de transporte
  - HTTP SOAP Transmission Optimization Feature
    - extensión para ser usada con HTTP

Ejemplo: Serialización original

```
HTTP/1.1 200 OK
Content-Length: 4542508
Content-Type: text/xml; charset=utf-8

<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/ ">
  <s:Body>
    <GetDataResponse xmlns="http://tempuri.org/ ">
      <GetDataResult>
   UesDBAoAAAAAAGRoaTsAAAAAAAAA..../Cj8KBxAEAG9xMgAHAFBBQ0syMDA=
      </GetDataResult>
    </GetDataResponse>
  </s:Body>
</s:Envelope>
```

Ejemplo: Serialización MTOM

```
HTTP/1.1 200 OK
Content-Length: 3407478
Content-Type:          multipart/related;          type="application/xop+
start="<http://tempuri.org/0>";          boundary="uuid:774aed72-9fbd-43ba-
24a5a025899e+id=2"; start-info="text/xml"
Server: Microsoft-HTTPAPI/1.0
MIME-Version: 1.0
Date: Fri, 03 Sep 2010 18:33:41 GMT

--uuid:774aed72-9fbd-43ba-a09a-24a5a025899e+id=2
Content-ID: <http://tempuri.org/0>
Content-Transfer-Encoding: 8bit
Content-Type: application/xop+xml;charset=utf-8;type="text/xml"

<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Body>
    <GetDataResponse xmlns="http://tempuri.org/">
      <GetDataResult>
        <xop:Include
          href="cid:http%3A%2F%2Ftempuri.org%2F1%2F634191248212968750"
            xmlns:xop="http://www.w3.org/2004/08/xop/include"/>
      </GetDataResult>
    </GetDataResponse>
  </s:Body>
</s:Envelope>

--uuid:774aed72-9fbd-43ba-a09a-24a5a025899e+id=2
Content-ID: <http://tempuri.org/1/634191248212968750>
Content-Transfer-Encoding: binary
Content-Type: application/octet-stream

PK##
□□□□□dhi;□□□□□□□□□□□□      □□□META-INF/PK##
```

Ejemplo: Serialización MTOM

- Content-Type:
  - Deja de ser text/xml para utilizar multipart/related
  - Se agregan a su vez tres elementos:
    - Tipo de adjuntos: application/soap+xml
    - Boundary: frontera de cada fragmento del mensaje multiparte
    - Boundary incial: indica dónde comienza el mensaje

Ejemplo: Serialización MTOM

- Cada Boundary contiene la siguiente información:
  - Content-type: tipo de mensaje contenido en el boundary
  - Content-ID: identificador del boundary

- En el mensaje soap se sustituye el archivo binario por una referencia a una boundary

Mensajería Confiable

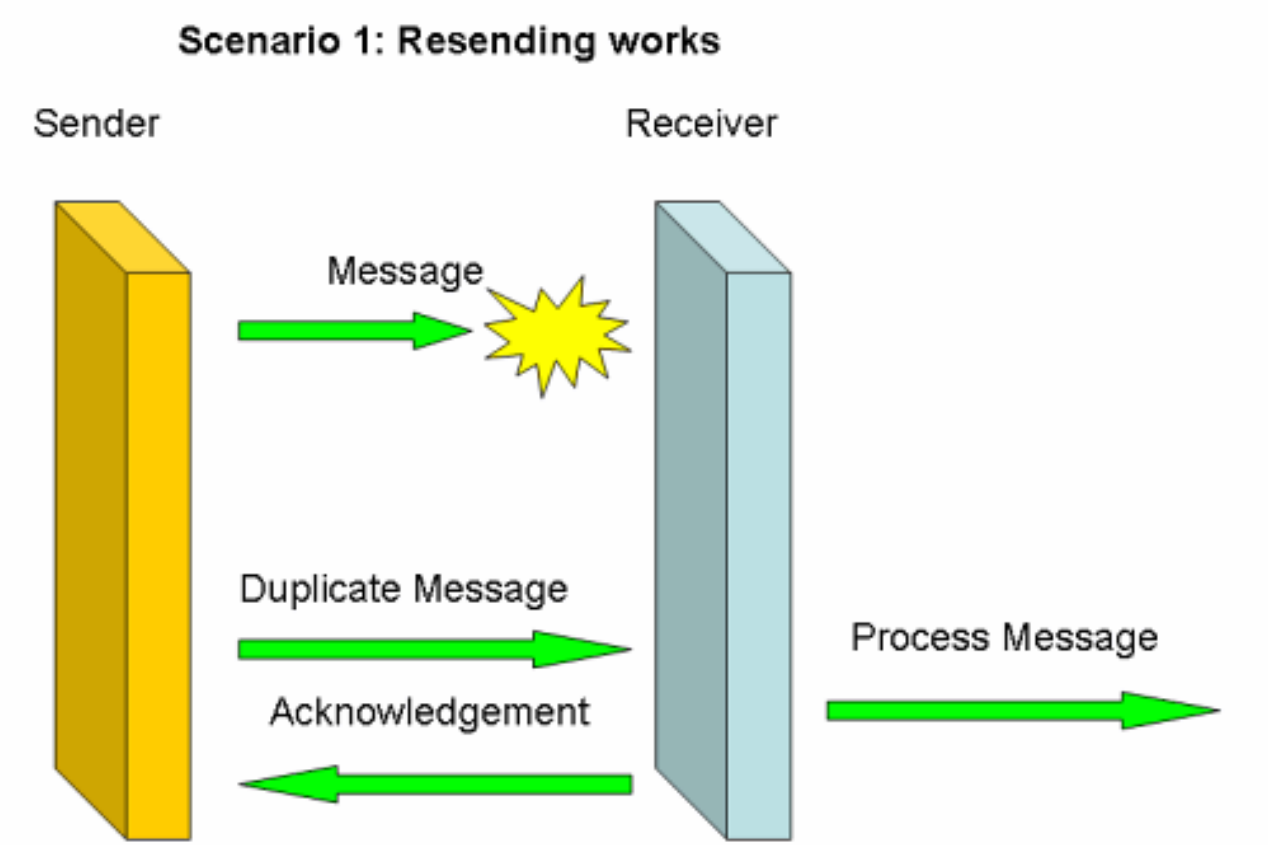
WS-ReliableMessaging

Motivación

- Errores inesperados en la comunicación
  - Fallas en la red
  - Caídas de servicio final
  - Intermediarios no disponibles

- ¿Cómo podemos resolver estos problemas?

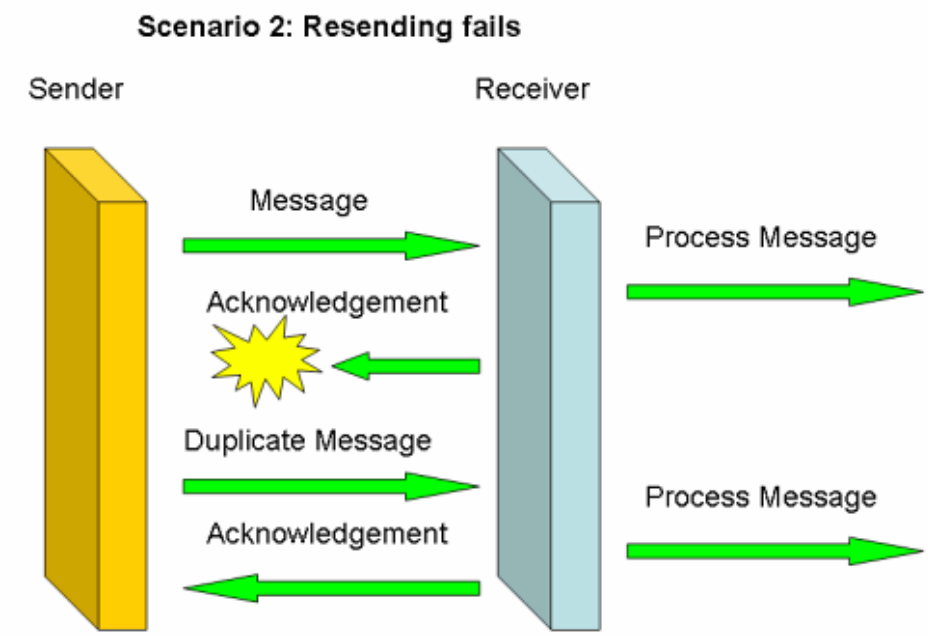
Reintentos


Scenario 1: Resending works
Sender
Receiver
Message
Duplicate Message
Process Message
Acknowledgement

Reintentos

- Podemos reintentar el envío del mensaje
- Capa de red
  - TCP!
- Capa de aplicación
  - Reintentos programados
  - Cliente más inteligente

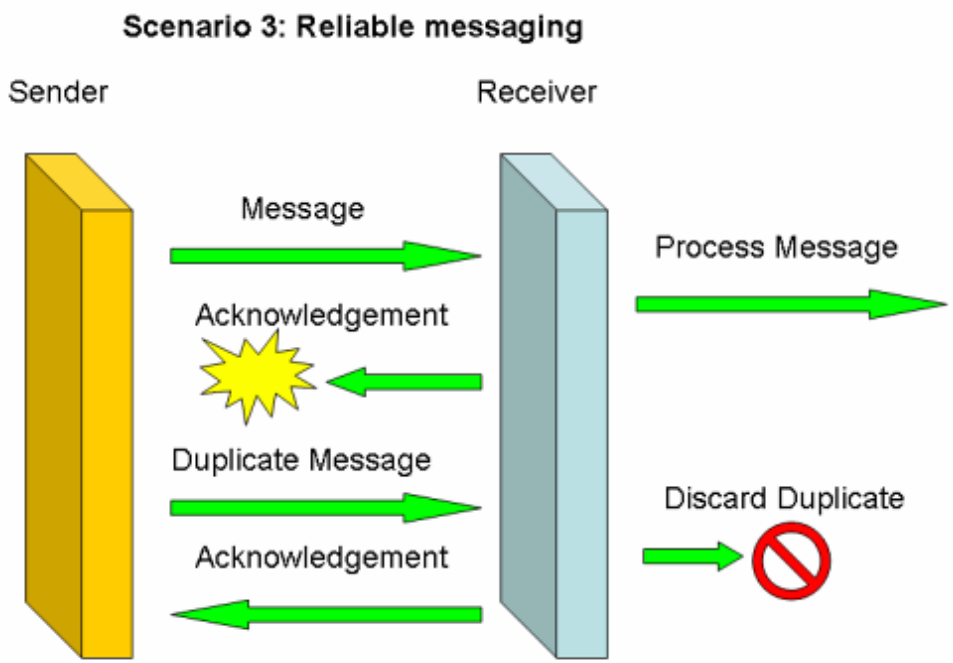
Sin embargo…



- Las respuestas se pueden perder
- Forzar duplicados
- Doble procesamiento de mensajes
  - Retiro bancario!

¿Opciones?

Reintentos y descartar duplicados



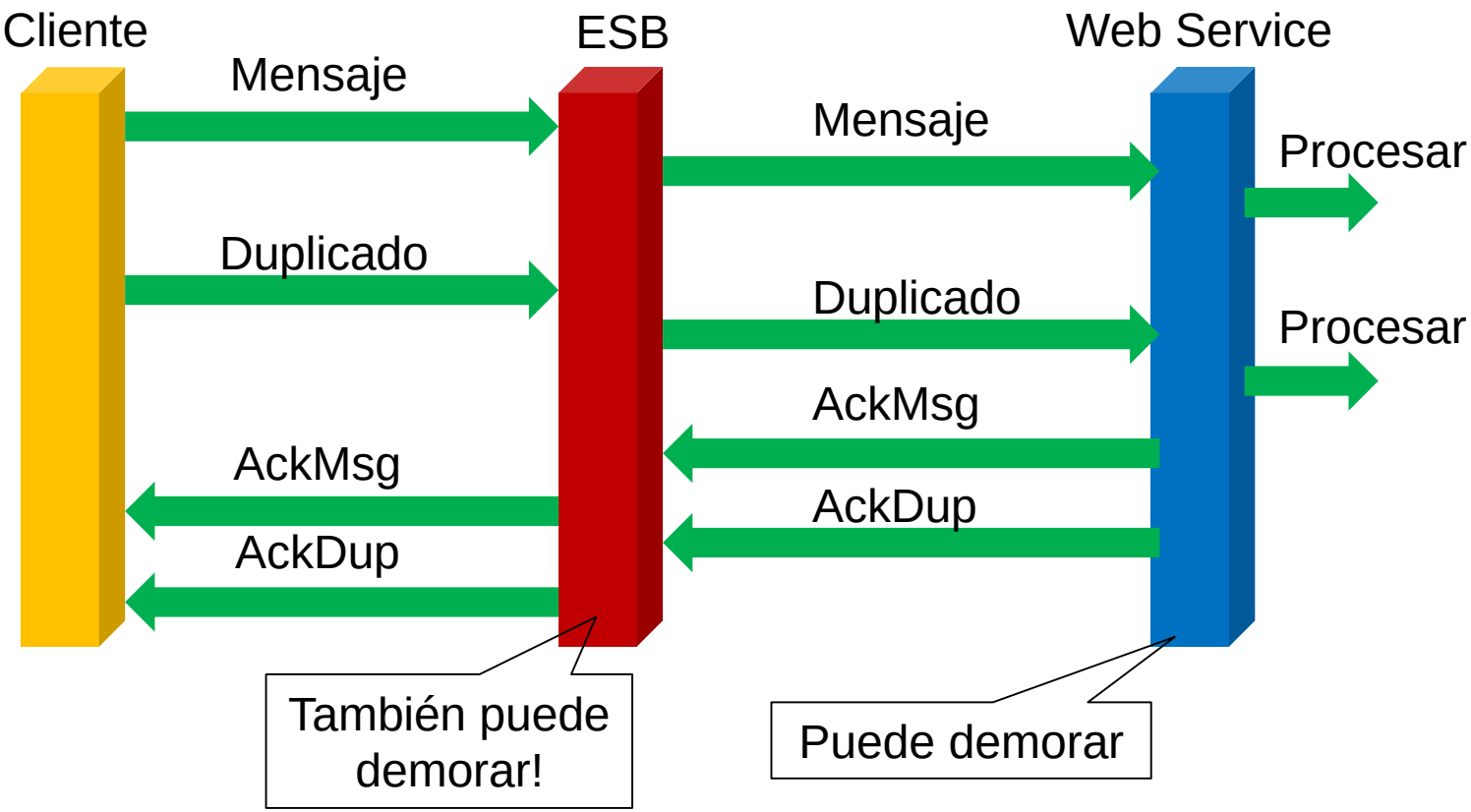
- Reenviar mensaje
  - TCP
  - Capa de aplicación
    - Cliente más inteligente
- Descartar duplicados
  - Capa de aplicación
    - Servicio más inteligente
- ¿Y si debemos mantener el orden de envío?

Escenarios

- Pago de facturas
  - No hay efecto frente a duplicados
- Recarga de celulares
  - Debe ponerse atención
  - ¿Es un duplicado o una nueva recarga?
- Transferencia bancaria
  - El orden importa
    1. Transferencia entre mis cuentas
    2. Desde una de mis cuentas a otro banco

¿Y los intermediarios?

- ¿Cómo diferenciamos una demora de una falla?



Problemas…

- Confiabilidad implementada en cada organización
- Confiabilidad es parte de la lógica de negocio
- Cada vez que dos organizaciones comenzaban una relación, acordaban un protocolo para garantizar confiabilidad
- Volver a tratar mismos problemas en cada integración!

WS-ReliableMessaging

- Estándar de la OASIS
  - Actualmente en versión 1.2
- Dar una solución estándar a la confiabilidad independiente de la lógica de negocio
  - Comunicación sincrónica y asincrónica
- No confundir confiabilidad con confidencialidad!!
  - Confidencialidad aplica a seguridad de datos

Garantía de entrega

- *AtLeastOnce* (al menos una vez)
- *AtMostOnce* (a lo sumo una vez)
- *ExactlyOnce* (exactamente una vez)
- *InOrder* (en orden, más alguno de los anteriores)

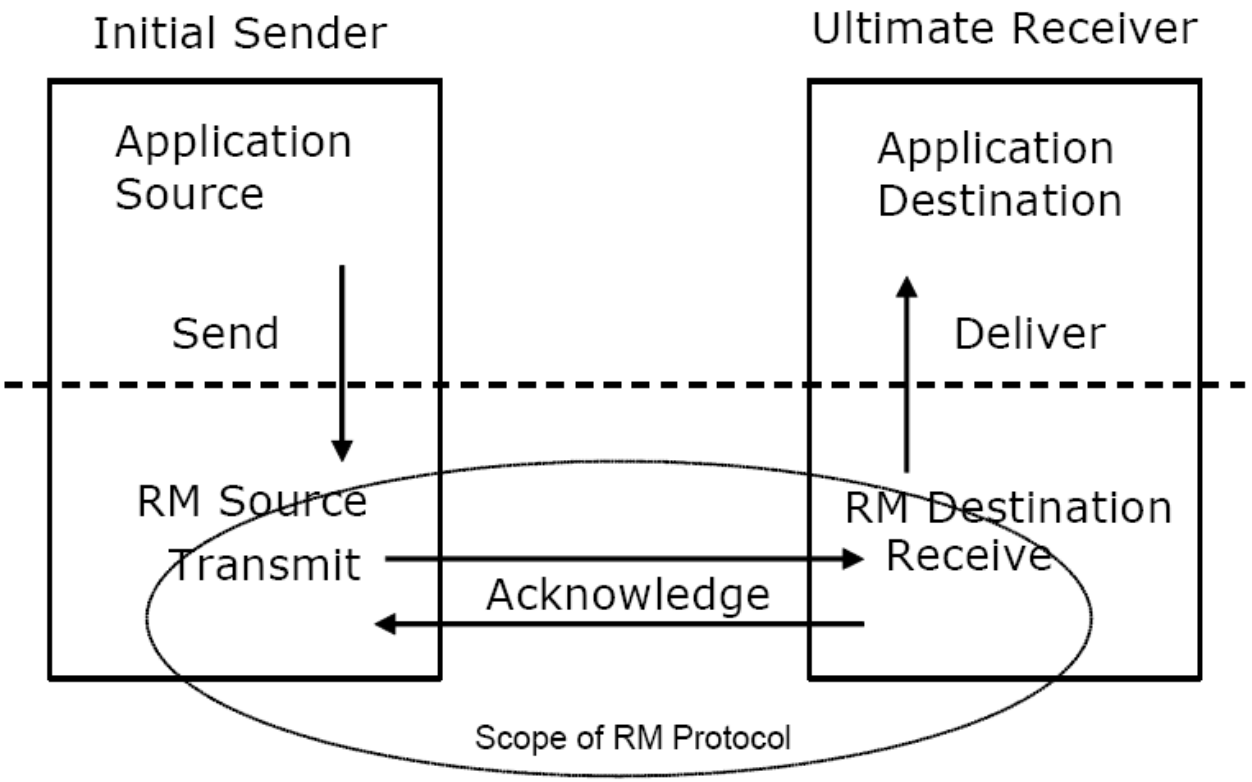
Comparativa

Política	Descripción	Observaciones
A lo sumo una vez	Solo se manda una vez el mensaje. Si ocurre una caída de red o servicio, el cliente recibe el error pero no lo vuelve a reenviar.	Puede ocurrir que el servicio nunca procese el mensaje. Cliente y servicio sin complejidad.
Al menos una vez	El mensaje se reenvía hasta que se recibe una confirmación de su procesamiento. Cada vez que se recibe un error de comunicación o se asume que el mensaje se perdió, este se vuelve a enviar.	El servicio puede llegar a procesar dos veces o más el mismo mensaje. Complejidad en el cliente.

Comparativa

Política	Descripción	Observaciones
Exactamente una vez	En caso de error de comunicación o pérdida del mensaje, este se reenvía. El servicio recibe solo una vez el mensaje. Elimina duplicados.	Costoso a nivel de performance e implementación. Complejidad en cliente y servicio
En orden	El servicio asegura un procesamiento ordenado de los mensajes enviados por el cliente.	Muy costoso a nivel de performance e implementación Complejidad en cliente y servicio

Modelo


Initial Sender
Ultimate Receiver
Application Source
Application Destination
Send
Deliver
RM Source
Transmit
RM Destination
Receive
Acknowledge
Scope of RM Protocol

Modelo

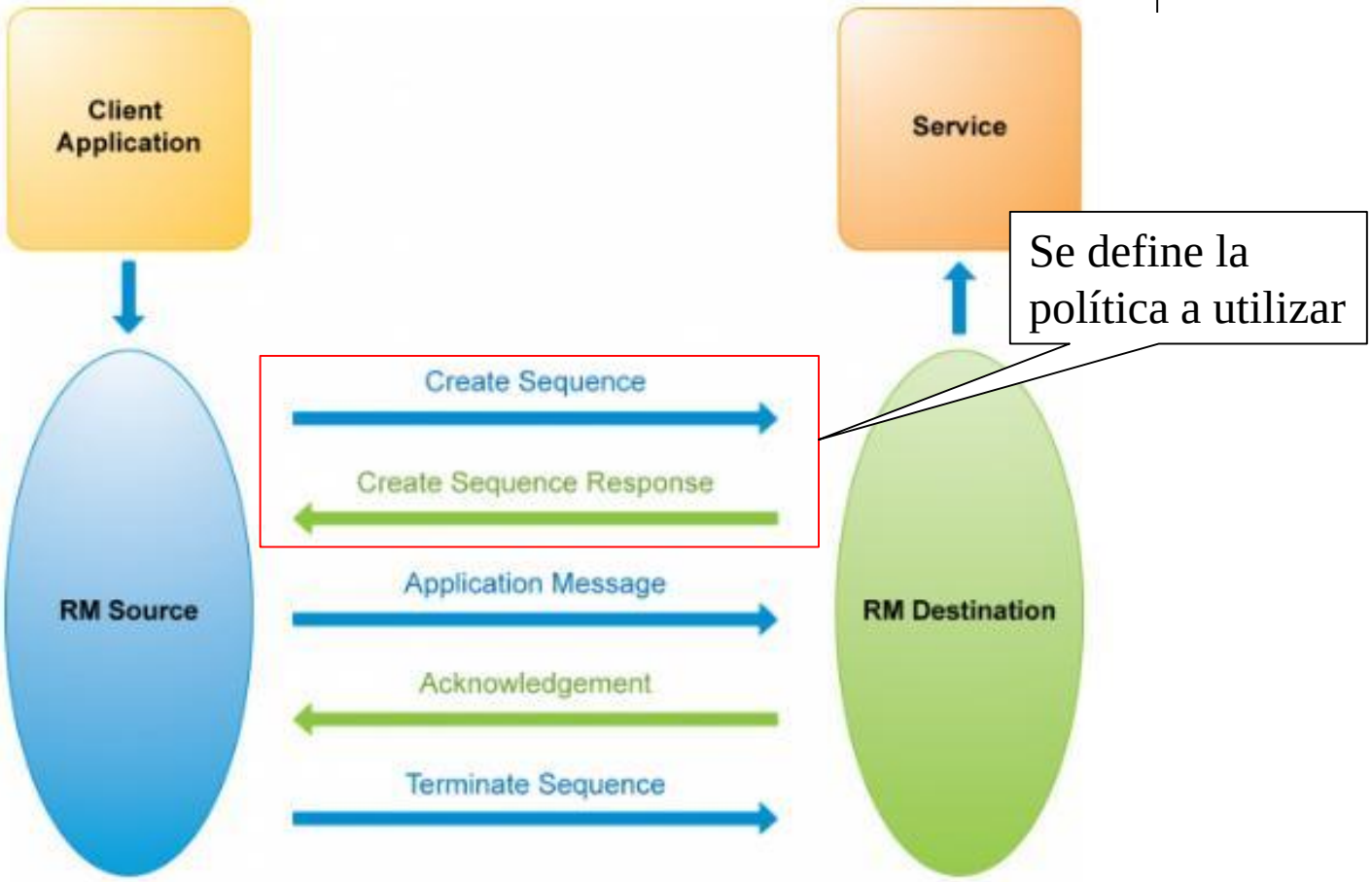
- Application source
  - Aplicación que implementa la lógica de negocio del cliente y consume el servicio
- Application destination
  - Web Service que implementa cierta lógica de negocio

Modelo

- RM Source
  - Solicita la creación y finalización de una sesión confiable
  - Agrega los cabezales SOAP relativos a WS-RM
  - Reenvía los mensajes de ser necesario

- RM Destination
  - Responde a las solicitudes de creación y finalización de sesiones confiables
  - Acepta y confirma la recepción de mensajes
  - Descarta mensajes duplicados (opcional)
  - Reordena los mensajes en caso que lleguen desordenados (opcional)

Intercambio de mensajes entre RMS y RMD



Creación y terminación de Secuencia

- Una secuencia es creada y terminada por el cliente
- Cuando una secuencia es creada, se retorna al cliente un identificador de la misma para poder realizar el envío de mensajes en su contexto.
- Una secuencia delimita al conjunto de mensajes al cual se le va a aplicar la garantía de entrega especificada.
- Una secuencia tiene un tiempo de vida útil

Creación de una secuencia

```
...
<s:Envelope>
    <S:Body>
        <wsrm:CreateSequence>
            <wsrm:AcksTo>
                <wsa:Address>http://Business456.com/serviceA/789</wsa:Address>
            </wsrm:AcksTo>
        </wsrm:CreateSequence>
    </S:Body>
</S:Envelope>
...
```

```
...
<S:Body>
    <wsrm:CreateSequenceResponse>
        <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
    </wsrm:CreateSequenceResponse>
</S:Body>
...
```

Números de secuencia

- Cada mensaje posee un número de secuencia
  - Empieza en uno y se incrementa en uno por cada mensaje
- Se utilizan para confirmar la recepción de los mensajes

Envío de Mensajes

```
...
<s:Envelope>
    <S:Header>
        <wsa:MessageID>...</wsa:MessageID>
        <wsa:To>http://example.com/serviceB/123</wsa:To>
        <wsa:From>
            <wsa:Address>http://Business456.com/serviceA/7</wsa:Address>
        </wsa:From>
        <wsa:Action>http://example.com/serviceB/123/request</wsa:Action>
        <wsrm:Sequence>
            <wsrm:Identifier>http://Business456.com/RM/ABC</wsrm:Identifier>
            <wsrm:MessageNumber>1</wsrm:MessageNumber>
        </wsrm:Sequence>
    </S:Header>
    <S:Body>
        <!-- Aquí van los datos a transmitirse entre las aplicacione -->
    </S:Body>
</S:Envelope>
...
```

Cabezales WS-RM

Cabezales WS-RM en respuesta

```
<soap:header>
  <wsrm:sequenceacknowledgement>
    <wsrm:identifier>http://Business456.com/RM/ABC</wsrm:identifier>
    <wsrm:acknowledgementrange lower="1" upper="1" />
    <wsrm:acknowledgementrange lower="3" upper="3" />
  </wsrm:sequenceacknowledgement>
</soap:header>
```

Algunas consideraciones

- Mensaje NAck (NoAcknowledge)
  - Mensaje enviado por el RM Destination al RM Source para indicarle que no recibió un determinado mensaje.

- Mensaje AckRequested
  - Mensaje enviado por el RM Source al RM Destination para solicitar la confirmación de recepción de un mensaje.

Limitaciones

- Web Service debe ser idempotente
  - El Web Service se puede consumir múltiples veces y siempre arrojar el mismo resultado
    - Si el servicio elimina un archivo, no es idempotente
    - Si el servicio sólo usa transacciones de base de datos podría llegar a ser idempotente.

- Persistencia de mensajes
  - Este requisito **NO** es exigido por WS-RM.

Resumen

- WS-SecureConversation
  - Sesiones seguras
- WS-Trust
  - Seguridad federada
- WS-Addressing
  - Asincronismo e identificación de mensajes
- MTOM
  - Envío eficiente de datos no-XML
- WS-ReliableMessaging
  - Confiabilidad en la entrega de mensajes

Fin

LINS

Laboratorio de Integracion de Sistemas