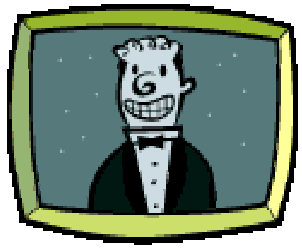


Secure and Reliable Web Services

Guy Crets

Integration Consultant
Apogado

Overall Presentation Goal

Web Services as basis for real-life Integration,
based on WS-Security and WS-ReliableMessaging

Speaker's Qualifications

- IT Consultant since 1987
- Managing Partner at Apogado
- Doing integration for the last 9 years: from screen-scraping and JMS to SAP Netweaver
- Speaks frequently on EAI, ESB and WS-*
- Guest lecturer at UAMS
- JavaPolis Steering Member

Waiting for WS-* …

WS-Security + WS-ReliableMessaging +…

Web Services can provide one single standard for secure and reliable communication. But after 6 years, it's time to nail things down.

Web Services - SOAP

- **XML** over **HTTP**
- Envelop: Header and body

```
<s:Envelop xmlns:s="http://.../soap/envelop">
  <s:Header >
      …
  </s:Header >
  <s:Body >
    <o:Order xmlns:o="http://...">
      <o:Product>...</o:Products>
      <o:Amount>...</o:Amount>
    </o:Order>
  </s:Body >
</s:Envelop>
```

Web Services

- SOAP spec dates back from July 2000 !
- WSDL: description of web services
- UDDI: discovery of web services
- Focus on synchronous request/reply
- XML over HTTP without SOAP
  - REST
  - B2B protocols
- Limited standardization of standard messages
  - Some use of B2B XML standards
  - E.g. WSDLs from Open Applications Group

WS (draft) standards

Messaging

- SOAP 1.1, 1.2
- WS Referral
- WS Routing
- WS-Addressing
- WS-MessageData
- WS-Enumeration
- WS-Eventing
- SOAP-over-UDP

XML

- ✓ XML
- ✓ Namespaces
- ✓ Information Set

Metadata

- WSDL 1.1, 2.0
- WS-Policy
- WS-PolicyAssertions
- WS-PolicyAttachment
- WS-Discovery
- WS-MetadataExchange
- WS-RM Policy
- UDDI 1.0, 2.0, 3.0
- WS Inspection Language

Messaging (2)

- ✓ WS-Notification
- ✓ WS-BaseNotification
- ✓ WS-BrokeredNotification
- ✓ WS-ReliableMessaging
- ✓ WS-Reliabiltiy
- ✓ ASAP
- ✓ WS-MessageDelivery
- ✓ WS-Acknowledgement
- ✓ WS-Callback

Attachments

- ✓ SwA SOAP with Attachments
- ✓ DIME / WS-Attachments
- ✓ MTOM (XOP)

More WS-* standards...

Security

- WS-Security: SOAP Message Security
- WS-Security: UsernameToken Profile
- WS-Security: X.509 Certificate Token Profile
- WS-Security: SAML Profile
- WS-SecureConversation
- WS-SecurityPolicy
- WS-Trust
- WS-Federation
- WS-Federation Active Requestor Profile
- WS-Federation Passive Requestor Profile
- WS-Security: Kerberos Binding
- Web Single Sign-On Interoperability Profile
- Web Single Sign-On Metadata Exchange Protocol

Business Process

- ✓ XLANG
- ✓ WSFL
- ✓ WS-BPEL (BPEL4WS)
- ✓ WS-Choreography
- ✓ WS-CDL
- ✓ WSCL (HP)
- ✓ WSCI

Management

- WS-Management
- WS-Management Catalog
- WS-DM
  - WS-MUWS part 1
  - WS-MUWS part 2
  - WS-MOWS
- WS-Manageability

And more ...

State / Context

- WS-Transfer
- WS-Resource
- WS-ResourceProperties
- WS-ResourceLifetime
- WS-ServiceGroup
- WS-BaseFaults
- WS-CAF
  - WS-Context
  - WS-CF

More security

- ✓ XML Signing
- ✓ XML Encryption
- ✓ SAML
- X-KMS
  - X-KISS
  - X-KRSS
- XACML

Transactions

- ✓ WS-Coordination
- ✓ WS-AtomicTransaction
- ✓ WS-BusinessActivity
- ✓ WS-T(X)M
- ✓ BTP

Miscellaneous

- ✓ WS-Remote Portlets
- ✓ WS-Provisioning

"The Web Services Standards Mess"

(Eric Newcomer, Iona)

The WS-* mix

- SOAP 1.1 → SOAP 1.2
- WSDL 1.1 → WSDL 2.0
- **WS-Addressing**
- **WS-ReliableMessaging**
- **WS-Security**
- UDDI → WS-MetaDataExchange
- SOAP with Attachments → MTOM/XOP
- ...

WS-Addressing

```
<s:Envelop xmlns:s="http://.../soap/envelop">
  <s:Header xmlns:wsa="...">
    <wsa:MessageID>
      uuid:aaaabbbb-cccc-dddd-eeee-wwwwwwwwwwww
    <wsa:MessageID>
    <wsa:To>…</wsa:To>
    <wsa:Action>
      http://../CreateOrder
    </wsa:Action>
    <wsa:From>…</wsa:From>
  </s:Header >
  <s:Body >
    <o:Order xmlns:o="http://...">
      <o:Product>...</o:Products>
      <o:Amount>...</o:Amount>
    </o:Order>
  </s:Body>
</s:Envelop>
```

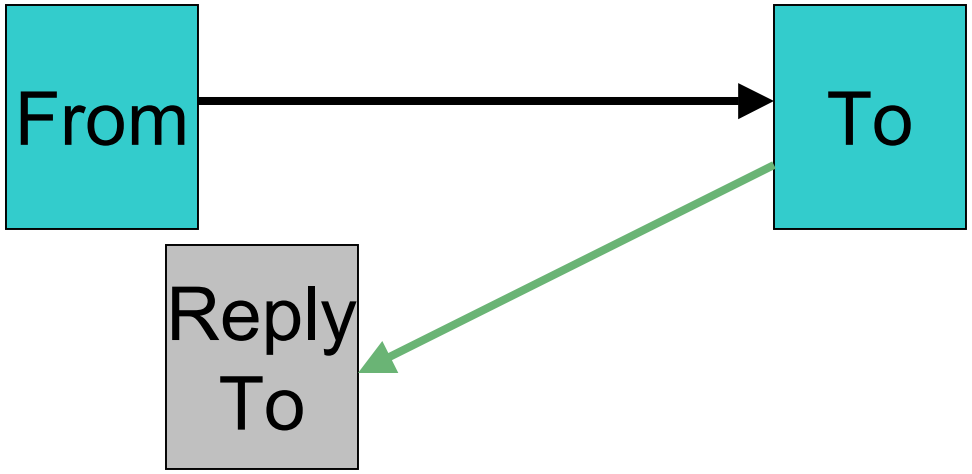
WS-Addressing

- Web service **Endpoint References**
- Message Information Headers

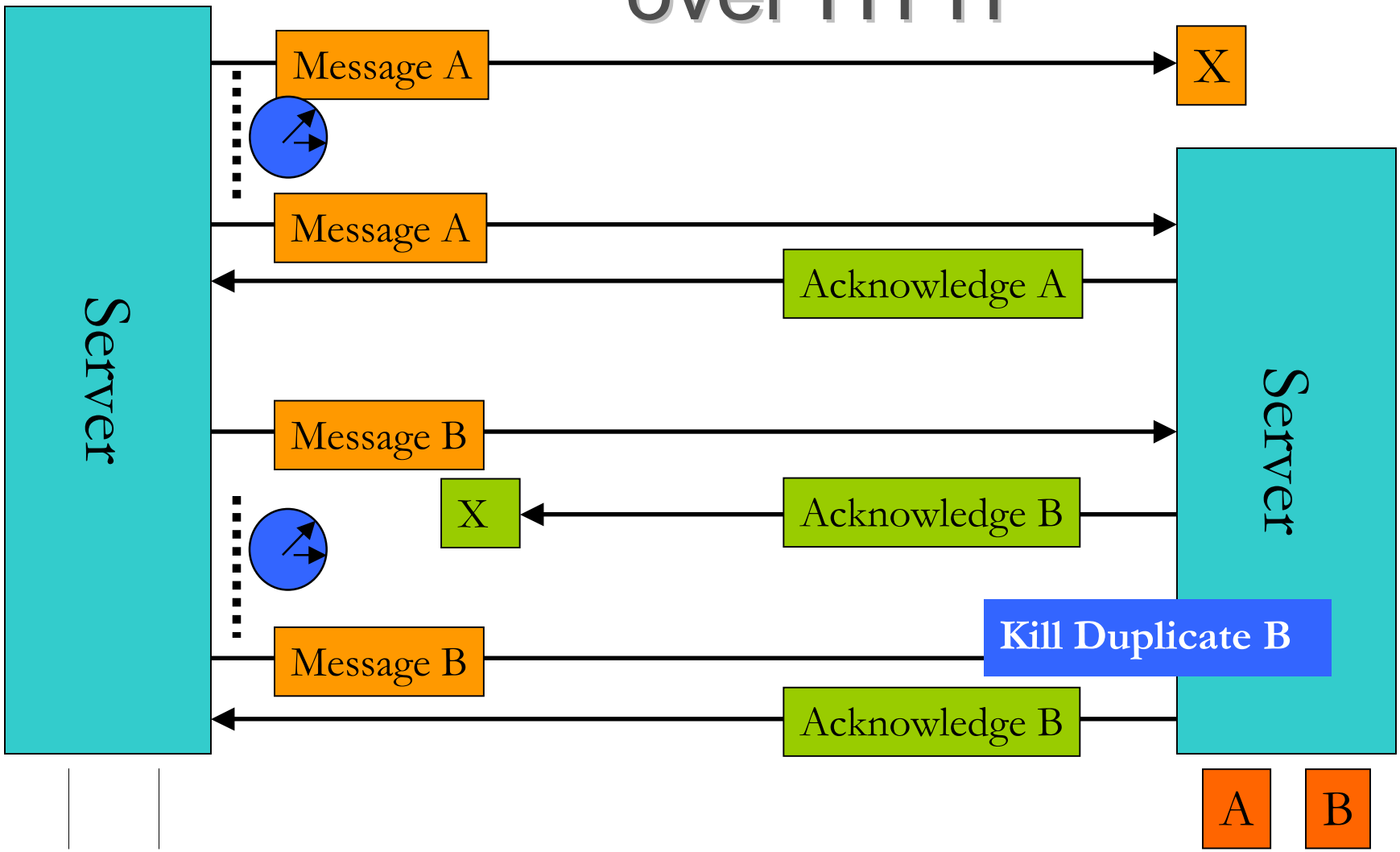
  wsa:MessageID, wsa:RelatesTo

  wsa:Action

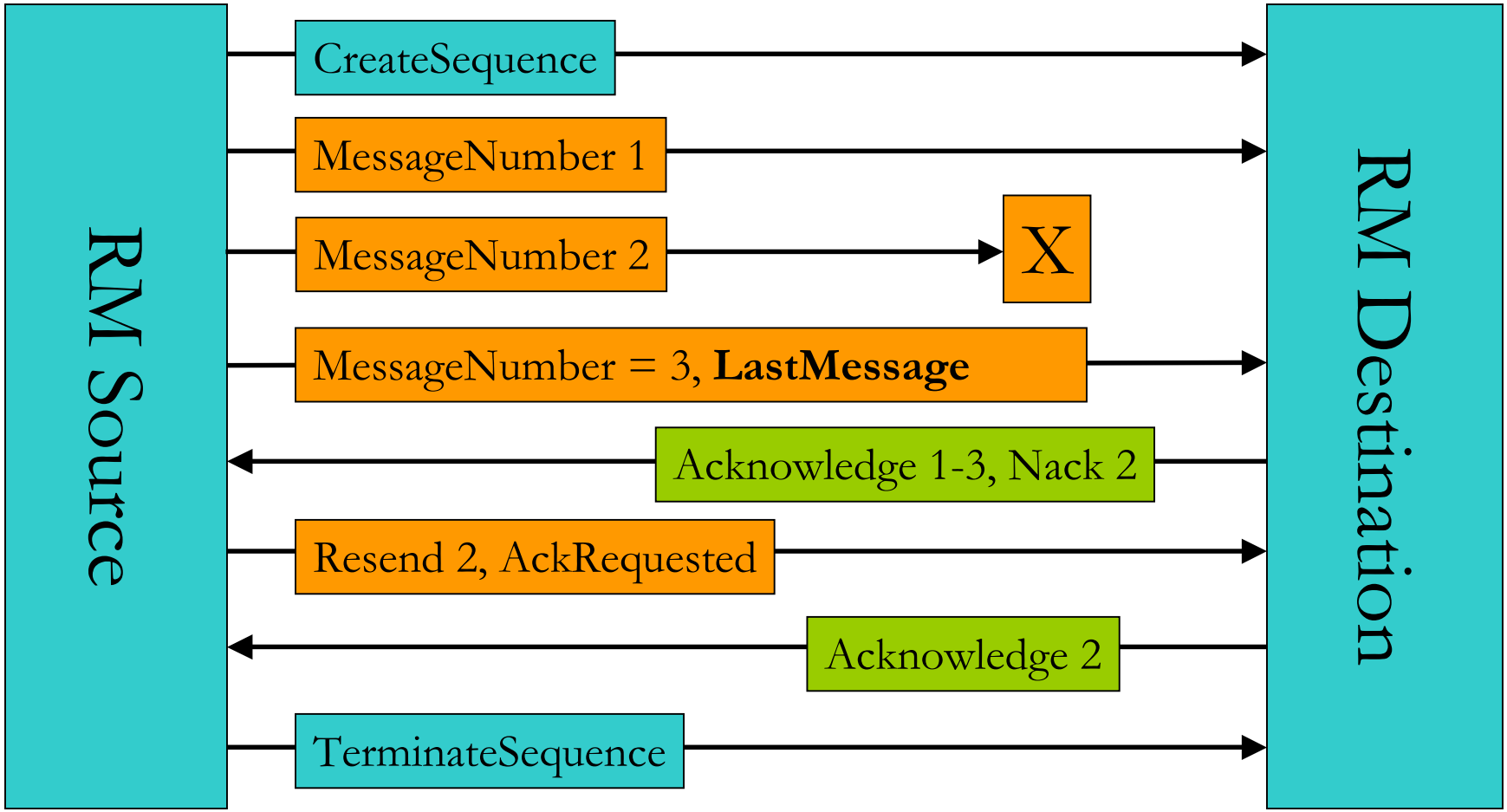
  wsa:To, wsa:From, wsa:ReplyTo, wsa:FaultTo



Reliable Messaging over HTTP



WS-RM protocol

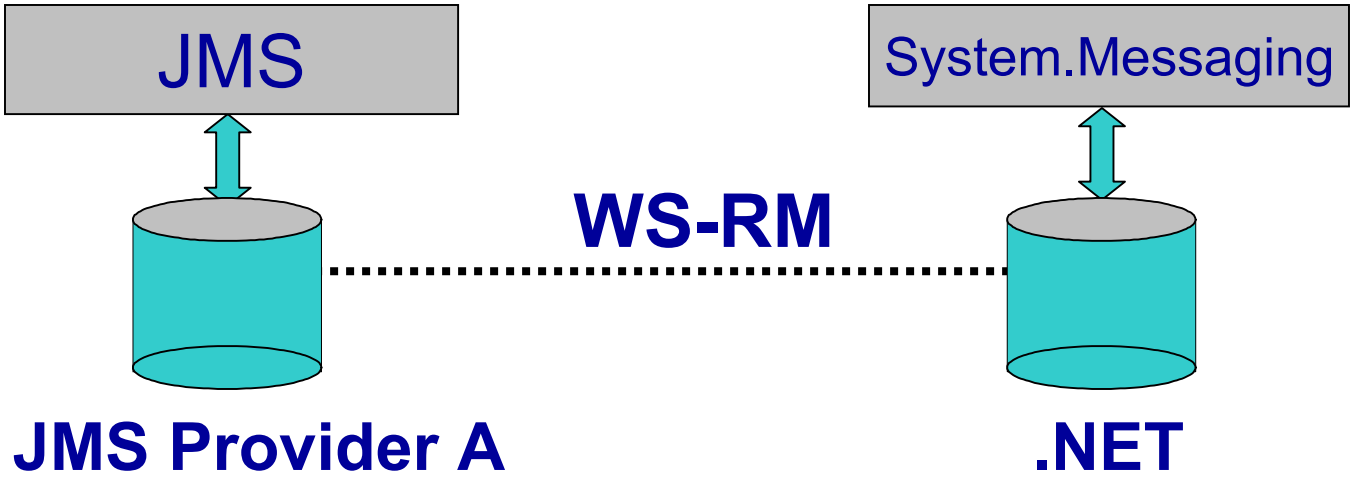


Reliable Sessions or Queued Messaging?

- WS-RM says nothing about **durability**
  - Persistent vs. Transient sequences
  - Persistent sequence survive re-starts, crashes, ...
- Microsoft WCF (Indigo)
  - Queued Messaging: use MSMQ
  - Maybe queued Messaging based on WS-RM in WCF 1.1 ?

WS-RM - Impact

- WS-RM will have ***MAJOR*** impact !!!
  - Products from different vendors at each side ~ B2B
  - Messaging becomes a commodity



  - Requires **Queued Messaging**

SOAP over e-mail ?

- Described (non-normative)
- SMTP is quite reliable
- Basic API's available
- Well-known addressing scher
- Limited support
  - CapeClear, Apache

"This is where our trails divide, Luke. You have my E-mail address, right?"

SOAP over FTP?

WS-Security

- OASIS standard(s)
- Authentication, Integrity, Privacy
- Profiles
  - **X509**, **UserName**, *Kerberos*, *SAML*, ...
- Stable
  - Compatible implementations
- Builds on
  - W3C XML Signature and XML Encryption

WS-Security

Username Profile 1.0

- Clear-text password

```
<soap:Envelope ...>
  <soap:Header>
    <wsse:SecurityToken>
      <wsse:UserName>guy</wsse:UserName>
      <wsse:Password>password</wsse:Password>
    </wsse:SecurityToken>
  </soap:Header>
  <soap:Body>
    ...
```

WS-Security

Username Profile 1.0

```
<wsse:Security>
 <wsse:UsernameToken>
  <wsse:Username>Guy Cret
  <wsse:PasswordType="wsse:PasswordDigest">
    D2A12DFE8D9F0C6BB82C89B091DF5C8A872F94DC
  </wsse:PasswordType>
  <wsse:Nonce>EFD89F06CCF  C89</wsse:Nonce>
  <wsu:Created>2005-11-  5:01:30Z</wsu:Created>
 </wsse:UsernameToken>
</wsse:Security>
```

UserName Toke Profile 1.1

- Derive key from password
- Encryption
- Integrity (MAC)

Hash(Nounce+TimeStamp+Password)

WS-Security - Signing

```
<s:Envelope>
  <s:Header>
    <wsse:Security>
      <ds:Signature>
        <ds:SignedInfo>
          ...
          <ds:Reference URI="#body">
          ...
        </ds:SignedInfo>
      </ds:Signature>
    </wsse:Security>
  </s:Header>
  <s:Body id="body">...
  </s:Body>
<s:Envelope>
```

XML Signature

XML Signature

```
<Signature Id="MyFirstSignature" xmlns="http://www.w3.org/2000/09/xmldsig#">
  <SignedInfo>
    <CanonicalizationMethod
      Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
    <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"/>
    <Reference URI="" />
      <Transforms>
        <Transform
          Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>
      </Transforms>
      <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <DigestValue>j6lwx3rvEPO0vKtMup4NbeVu8nk=</DigestValue>
    </Reference>
  </SignedInfo>
  <SignatureValue>MC0CFFrVLtRlk=...</SignatureValue>
  <KeyInfo>
    <KeyValue>
      ......
   </KeyValue>
  </KeyInfo>
</Signature>
```

XML Signature

```
<Signature>
    <SignedInfo>

   (CanonicalizationMethod)
        (SignatureMethod)
        (<Reference (URI=)? >
            (Transforms)?
            (DigestMethod)
            (DigestValue)
        </Reference>)+
    </SignedInfo>
    (SignatureValue)
    (KeyInfo)?
    (Object)*
</Signature>
```

Object to be signed

References = `SignedInfo`

URI:

- External document
  `URI="http://www…/…"`
- Document itself (root)
  `URI=""`
- Part of document
  `URI="#PurchaseOrder"`
- Attachments

KeyInfo = certificate

Canonicalization

- C14N CanonicalizationN ('C'+14 chars +'N')
- "Standardize" the XML document
  - Standard encoding (UTF-8)
  - Line breaks: #xA (new line)
  - Attributes: normalize white space
    - single quotes → double quotes
    - quotes wihtin quotes → "
  - Remove XML and DTD declarations
  - Empty: <element/> → <element></element>
  - Namespaces declarations: remove unused, sort
  - …

Canonicalization

XMLStarlet Command Line XML Toolkit

```
<?xml version="1.0" encoding="ascii" ?>
<x:test xmlns:y="http://apogado.com/y"
      xmlns:x="http://apogado.com/test"
        xmlns="http://apogado.com/z" >
    <x:emtpy  a='"a"'/>
  <element b="b">bbbb
  </element>
</x:test>
```

```
<x:test xmlns:x="http://apogado.com/test">
    <x:emtpy a="&quot;a&quot;"></x:emtpy>
  <element xmlns="http://apogado.com/z" b="b">bbbb
  </element>
</x:test>
```

Step by step

- For each reference
  - Transform (usually c14n)
  - Calculate digest
  - Create `<Reference>`
- For `<SignedInfo>` (containing all References)
  - Canonicalize
  - Calculate digest
  - Encrypt digest (= sign)

    Result in `<SignatureValue>`

```
<Reference URI="" />
  <Transforms>
    <TransformAlgorithm=" "/>
  </Transforms>
  <DigestMethod Algorithm=" "/>
  <DigestValue> </DigestValue>
</Reference>
```

"Indirect" signing

1. **Hash of every reference**
2. **Hash of the hashes**
3. **Sign the "hash of the hashes"**

Sign the hash of the hashes

```
<s:Header id="header">
  <element>1</element>
  <element>2</element>
<s:Header>
```

```
<s:Body id="body">
  <o:Order xmlns:o="...">
    <o:Product>...</o:Products>
    <o:Amount>...</o:Amount>
  </o:Order>
</s:Body>
```

Transform
Digest (hash)

```
<Reference URI="#header" />
  <Transforms>
    <TransformAlgorithm=" "/>
  </Transforms>
  <DigestMethod Algorithm=" "/>
  <DigestValue> </DigestValue>
</Reference>
<Reference URI="#body" />
  <Tranforms> … </Transforms>
  <DigestMethod Algorithm=" "/>
  <DigestValue> </DigestValue>
</Reference>
```

- Transform (Canonicalize)
- Digest
- Encrypt

```
<Signature … >
 <SignedInfo>
   <CanonicalizationMethod … />
   <SignatureMethod>
   <Reference URI= >…</Reference><Reference URI= ></Reference>
  </SignedInfo> >…</Reference
  <SignatureValue>hTHQJyd3C6ww…</SignatureKeyValue>
```

X509Token Profile

```
<wsse:Security>
  <wsse:BinarySecurityToken ValueType="wsse:X509v3"
                            EncodingType="wsse:Base64Binary"
                            wsu:Id="X509Token">
    FIgEZzCRF1EgILBAgIQEmtJZc0rqrKh5i...
  </wsse:BinarySecurityToken>
  <ds:Signature>
    <ds:KeyInfo>
      ….
      <wsse:SecurityTokenReference>
        <wsse:Reference URI="#X509Token"/>
      </wsse:SecurityTokenReference>
    </ds:KeyInfo>
  </ds:Signature>
</wsse:Security>
```

Certificate:

- Container for public key
- Identity owner of private key
- Attested by the CA

XML Security - Signature

```
<S:Envelope xmlns:S="…" >
  <S:Header>
    <wsse:Security S:mustUnderstand="1" xmlns:wsse="…" >
      <wsse:BinarySecurityToken ValueType="wsse:X509v3"
                                EncodingType="wsse:Base64Binary"
                                wsu:Id="X509Token">
                                FIgEZzCRF1EgILBAgIQEmtJZc0rqrKh5i...
      </wsse:BinarySecurityToken>
      <ds:Signature xmlns:ds="…">
        <ds:SignedInfo>
          <ds:CanonicalizationMethod Algorithm="…" />
          <ds:SignatureMethod Algorithm="…" />
          <ds:Reference URI="#body">
            <ds:Transforms>
              <ds:Transform Algorithm="…" />
            </ds:Transforms>
            <ds:DigestMethod Algorithm="…" />
            <ds:DigestValue>EULddytSo1...</ds:DigestValue>
          </ds:Reference>
        </ds:SignedInfo>
```

XML Security - Signature

```
        <ds:SignatureValue>
          XLdER8=ErToEb1l/vXcMZNNjPOV...
        </ds:SignatureValue>
        <ds:KeyInfo>
          <wsse:SecurityTokenReference>
            <wsse:Reference URI="#X509Token"/>
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
      </ds:Signature>
    </wsse:Security>
  </S:Header>
  <S:Body wsu:Id="body">
    <StatusRequest xmlns="http://www.apogado.com/Order">
       <OrderNumber>1234</OrderNumber>
    </StatusRequest>
  </S:Body>
</S:Envelope>
```

XML Security - Timestamps

- Addition to XML Signature
- wsu → Web Services Utility

```
<s:Envelope xmlns:s="">
  <s:Header>
    <wsu:Timestamp>
      <wsu:Created>2005-03-03T01:42:00Z</wsu:Created>
      <wsu:Expires>2005-03-04T01:00:00Z</wsu:Expires>
    </wsu:Timestamp>
    ...
  </s:Header>
  <s:Body>
    ...
  </s:Body>
</s:Envelope>
```

WS-Security developments

- SAML Token Profile
- Security Roadmap
- WS-Trust
- InfoCard
- Real world, secure web service: Paypal
- Security in Hardware

Windows Vista™ Developer Center

Cyper Criminals Target Web Services

Yahoo, Google, PayPal Seek To Close Security Holes

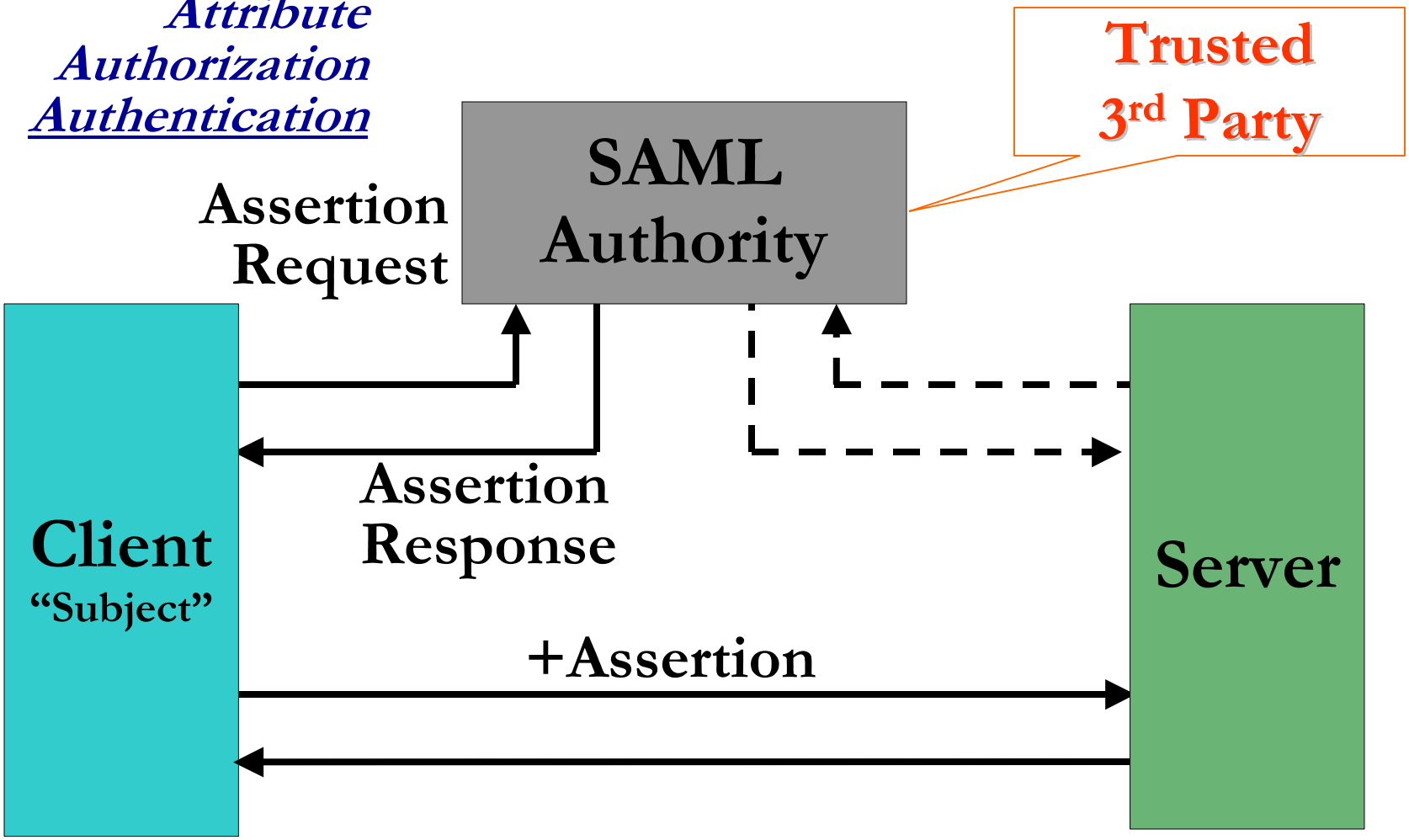
SAML

The Security Assertions Markup Language is an XML-based framework for Web services that enables the exchange of authentication and authorization information among business partners.

- Pre-dates WS-*
- Use-cases: Single Sign-On, Authorization Service, Back-office transaction
- OASIS included SAML in WS-Security
- Strong focus on Single Sign-On from browser

SAML

Attribute
Authorization
Authentication



Protocol: HTTP, SMTP, SOAP, JMS, ebXML, …

SAML Assertion

```
<saml:Assertion xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
  MajorVersion="2" MinorVersion="0"
  AssertionID="buGxcG4gILg5NlocyLccDz6iXrUa"
  Issuer="www.trustcompany.com"
  IssueInstant="2005-03-15T17:05:37.795Z">
  <saml:Conditions NotBefore="2005-03-15T17:00:37.795Z"
                   NotOnOrAfter="2005-03-15T17:10:37.795Z"/>
  <saml:AuthenticationStatement
      AuthenticationMethod="urn:oasis:names:tc:SAML:2.0:am:password"
      AuthenticationInstant="2005-03-15T17:05:17.706Z">
    <saml:Subject>
      <saml:NameIdentifier
          NameQualifier=http://www.tcompany.com
          Format="http://www.customformat.com/">
        uid=GuyCrets
      </saml:NameIdentifier>
      <saml:SubjectConfirmation>
        <saml:ConfirmationMethod>
          urn:oasis:names:tc:SAML:2.0:cm:artifact-01
        </saml:ConfirmationMethod>
      </saml:SubjectConfirmation>
    </saml:Subject>
  </saml:AuthenticationStatement>
```

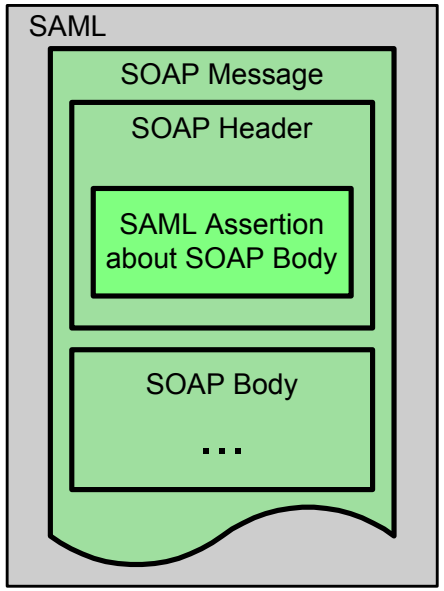
Assertion Can also be Digitally Signed

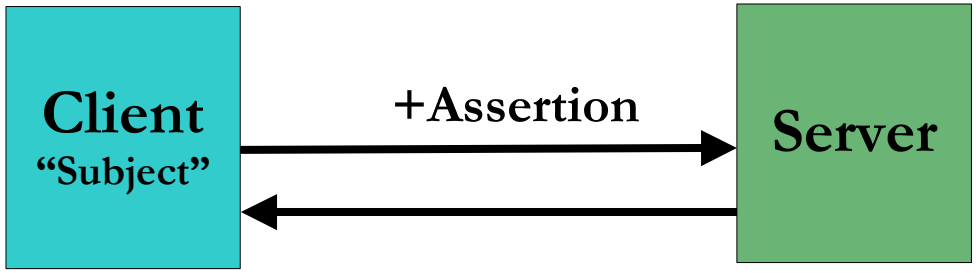
WS-Security & SAML

SAML in front of WS-Security

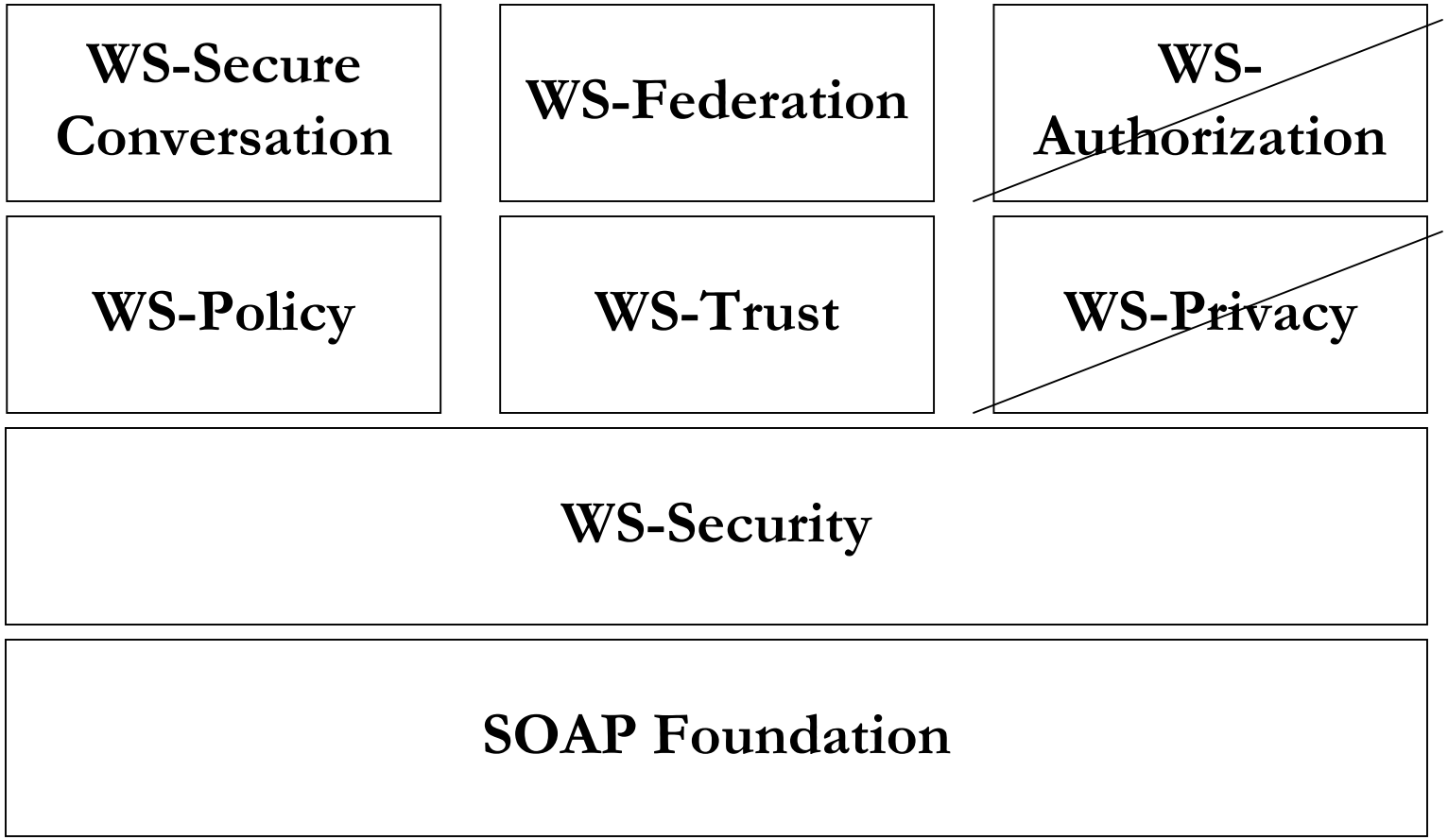
```
<S:Envelope>
  <S:Header>
    <wsse:Security>
      <saml:Assertion
        AssertionID="_a75adf55-01d7-40cc-929f-dbd8372ebdfc"
        IssueInstant="2005-03-15T00:46:02Z"
        Issuer="www.opensaml.org"
        ..."
      </saml:Assertion>

      <wsse:SecurityTokenReference wsu:Id="STR1">
        <wsse:KeyIdentifier wsu:Id=" …"
           ValueType="http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1
           _a75adf55-01d7-40cc-929f-dbd8372ebdfc
        </wsse:KeyIdentifier>
      </wsse:SecurityTokenReference>
    </wsse:Security>
  </S:Header>
  <S:Body>
    …
  </S:Body>
</S:Envelope>
```



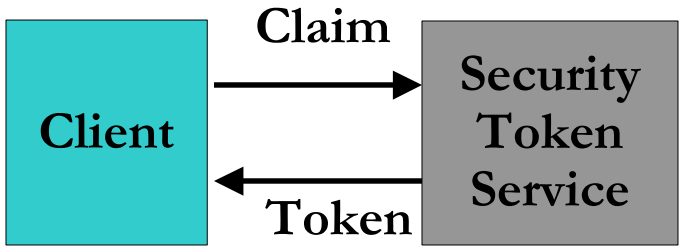


WS-Security *Roadmap*

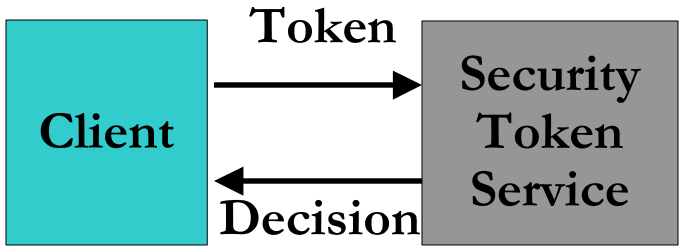


WS-Trust

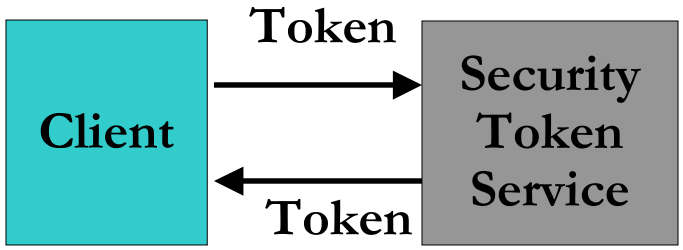
- Issuance
  - ~ SAML Authentication



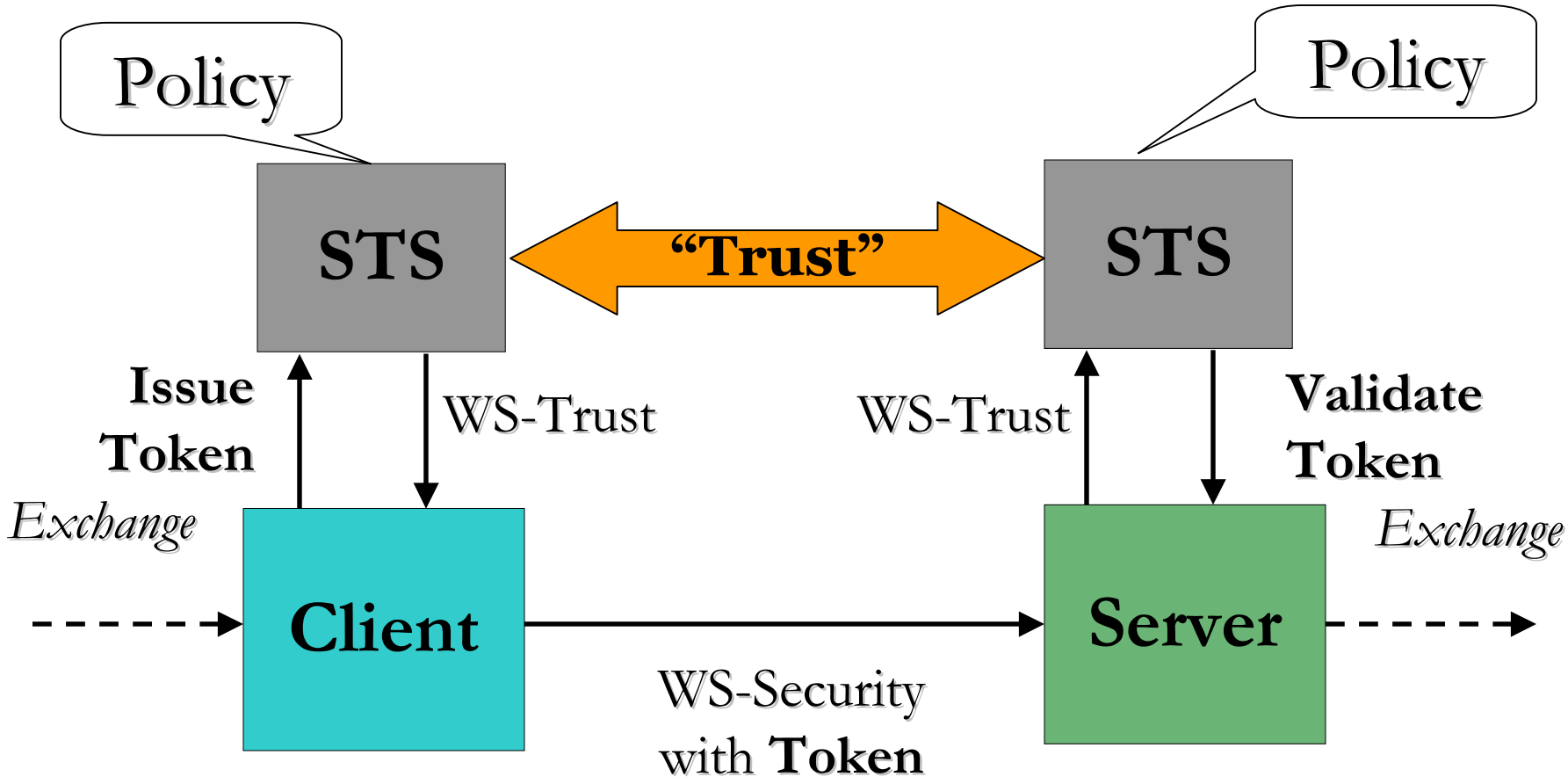
- Validation



- Exchange
  - Convert X509 or SAML to Kerberos



WS-Trust



Microsoft InfoCard

Users select "identity"

S-MEX

-Security Policy

-Trust

g. SAML

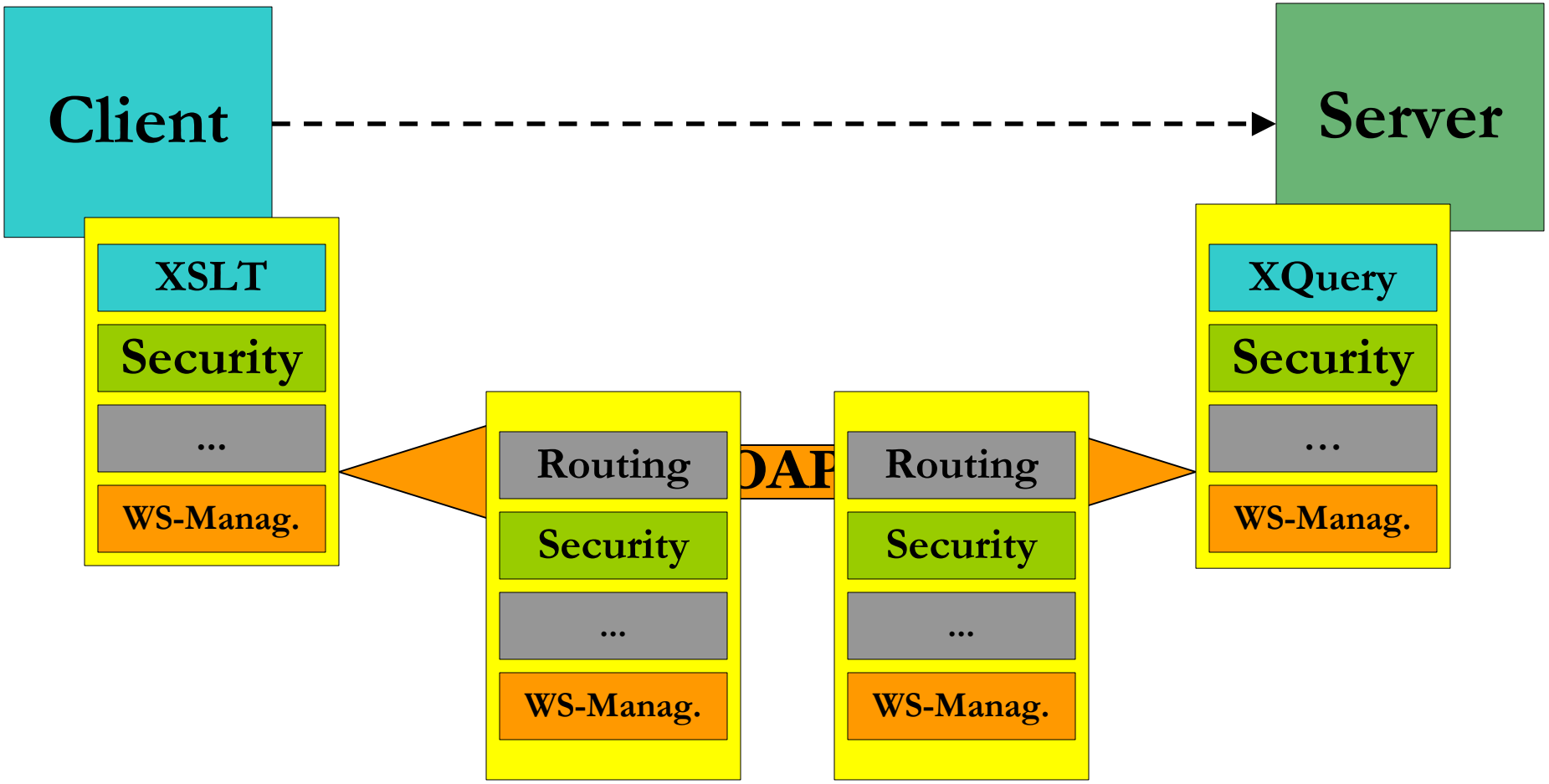
WS-Security

Specialized WS Security products & vendors

- Agents / PEP
  - Proxies or plugged into WS-Stack
- Overlap between tools/products for Securing & Managing web services
- WS-Policy support

- Features
  - Enforce policies (PEP)
  - Sign, validate
  - Encrypt/decrypt
  - Support WS-Security, SAML, …
  - Access Control - Integrate with LDAP and Identity Mgt. Solutions
  - Data validation:
    - against WSDL
    - against schema's
  - (Reverse) Proxy
  - Detect Denial-Of-Service
  - Audit trail
  - Route message

WS stack



Service "mediation"

Real Web Services Security

- Salesforce.com
  - Userid & password (no WS-Security)
  - Returns session-id and new server URL

    e.g. https://**na1-api**.salesforce.com/services/Soap/c/7.0
- Amazon S3
  - Signature: RFC 2104

    HMAC-SHA1 of

    "AmazonS3"+ OPERATION + Timestamp

    e.g. AmazonS3CreateBucket2005-01-31T23:59:59.183Z
- PayPal
  - PaypalUses HTTPS with client certificate or "Signature"

PayPalTech.com

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope
  ...
  <SOAP-ENV:Header>
    <RequesterCredentials xmlns="urn:ebay:api:PayPalAPI" ... >
      <Credentials xmlns="urn:ebay:apis:eBLBaseComponents">
        <Username>business_api1.test.be</Username>
        <Password>V9VTSDBRJXXH6TCS</Password>
        <Signature>AFcWxV21C7fd0v3bYYYRCp...</Signature>
        <Subject/>
      </Credentials>
    </RequesterCredentials>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body>
    <RefundTransactionReq xmlns="urn:ebay:api:PayPalAPI">
```

WS/XML firewalls

- Sarvega's XPE 2000
- Forum Systems' XWall
- DataPower's XS40 XML Security Gateway (IBM)

- Westbridge Technology's XML Message Server
- Vordel's VordelSecure
- Reactivity's Reactivity XML Firewall
- Digital Evolution

- CISCO AON

EAI – WS – B2B

B2B

ROSETTANET

ebXML Creating A Sir

EDIINT AS2

EDI VAN
Value Added Network

ODETTE

Transaction Delivery Network

Firewall

WS

Web Services

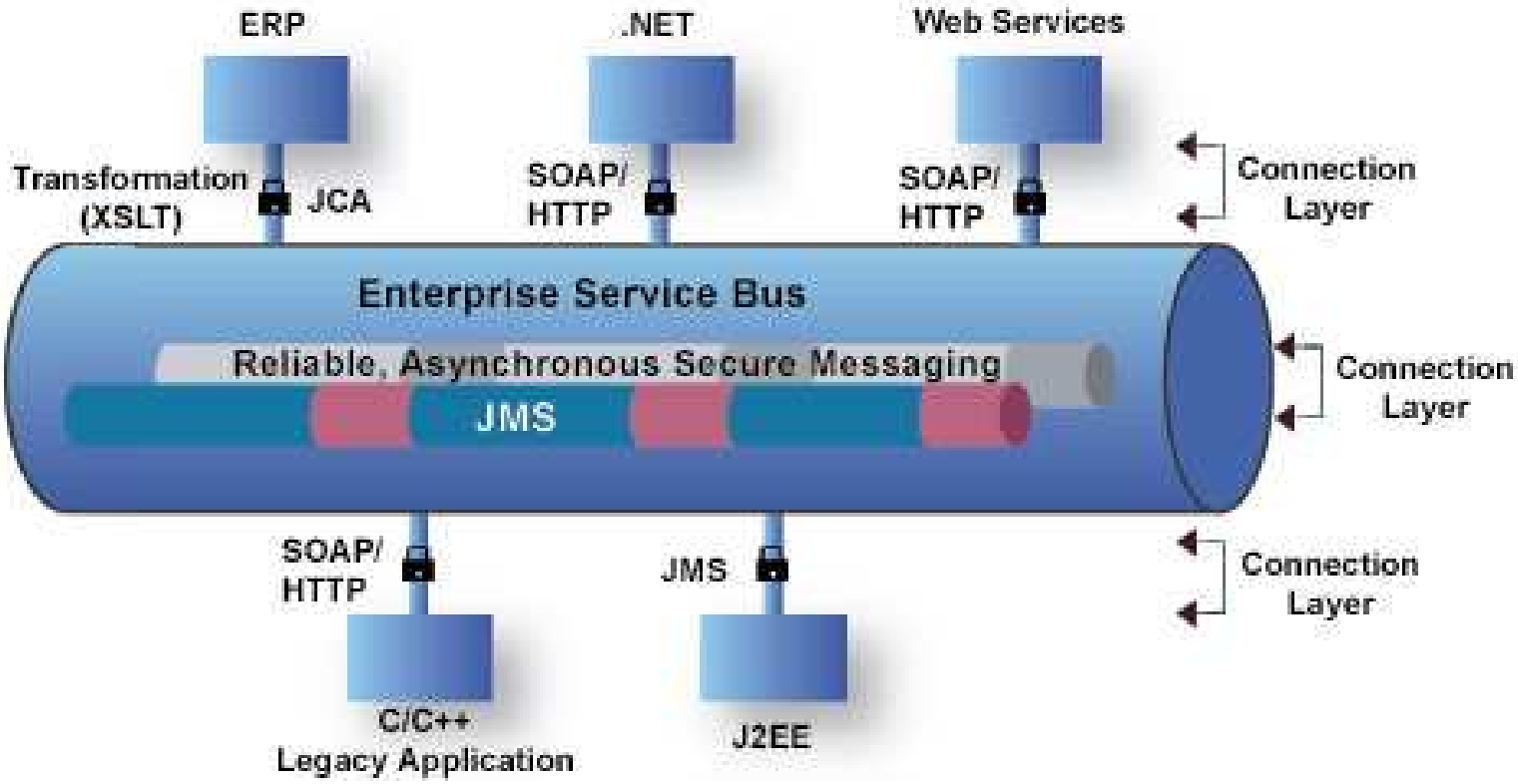
Used for request/reply (RPC) within organizations

Communication "Bus"

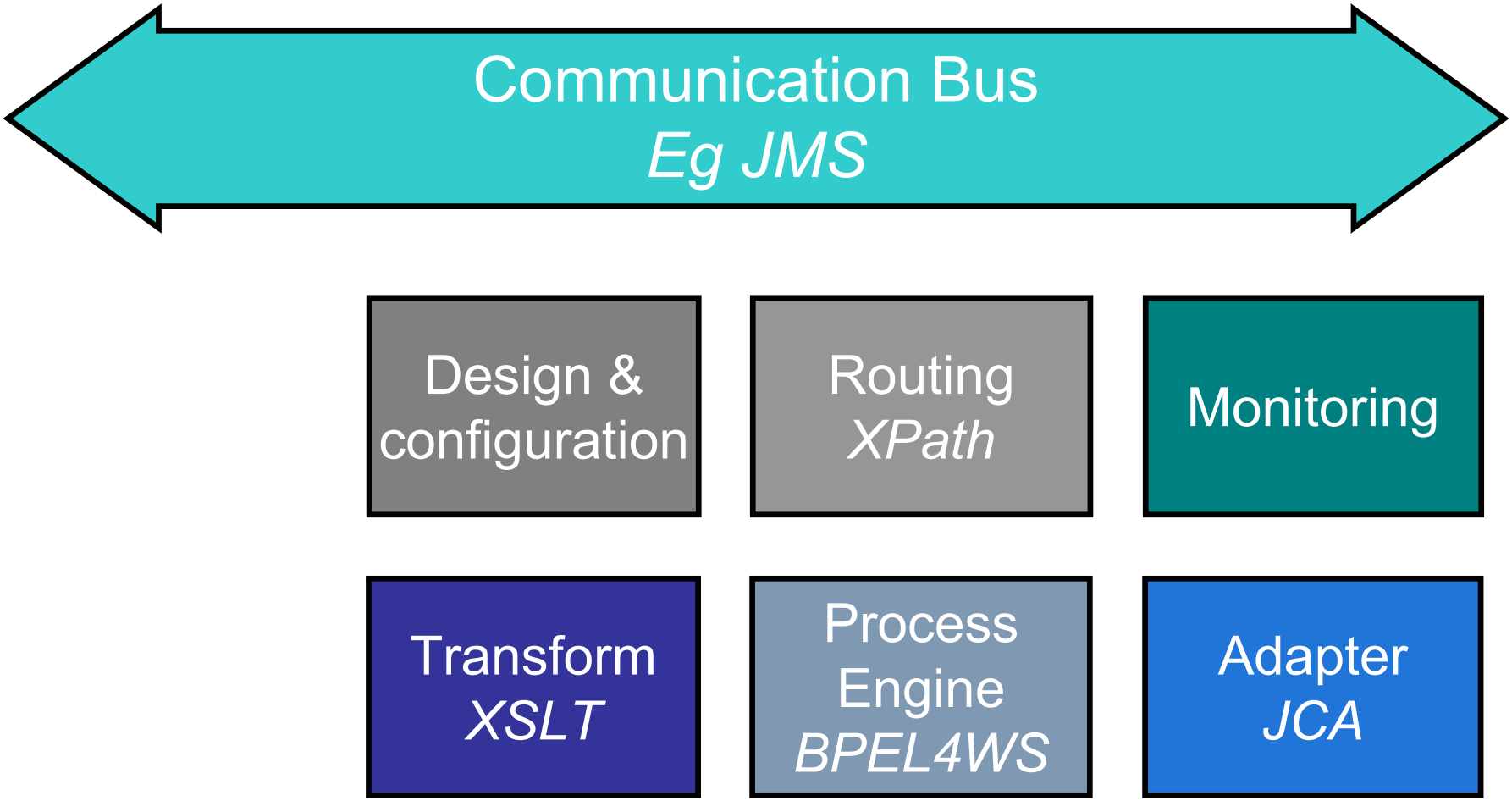
Messaging used for both request/reply (RPC) and asynchronous communication

EAI

EAI:
Enterprise Service Bus



Enterprise Service Bus



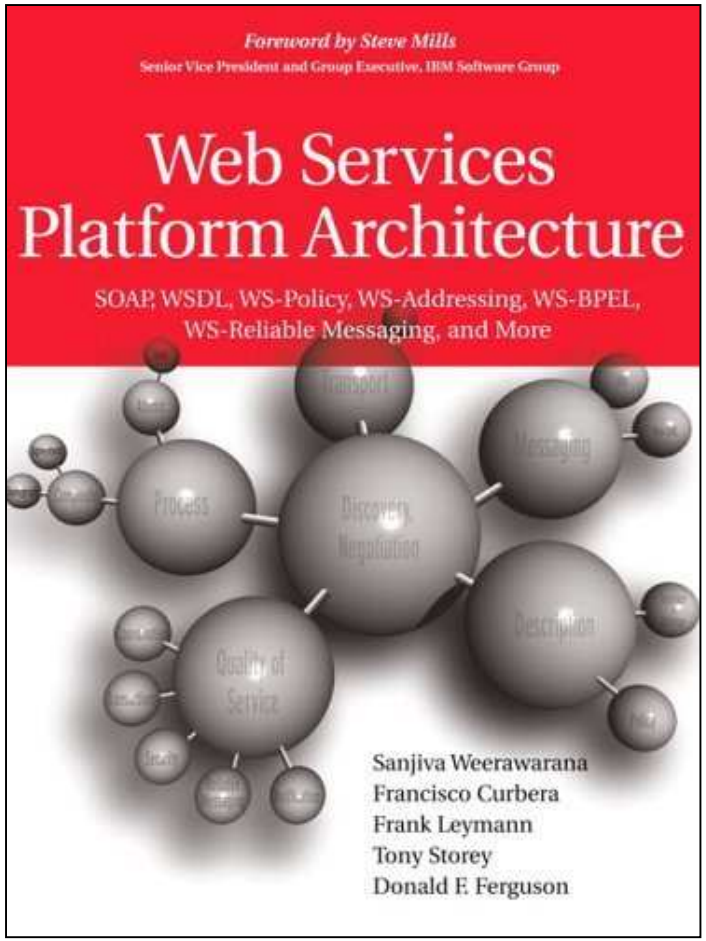
B2B - External connectivity

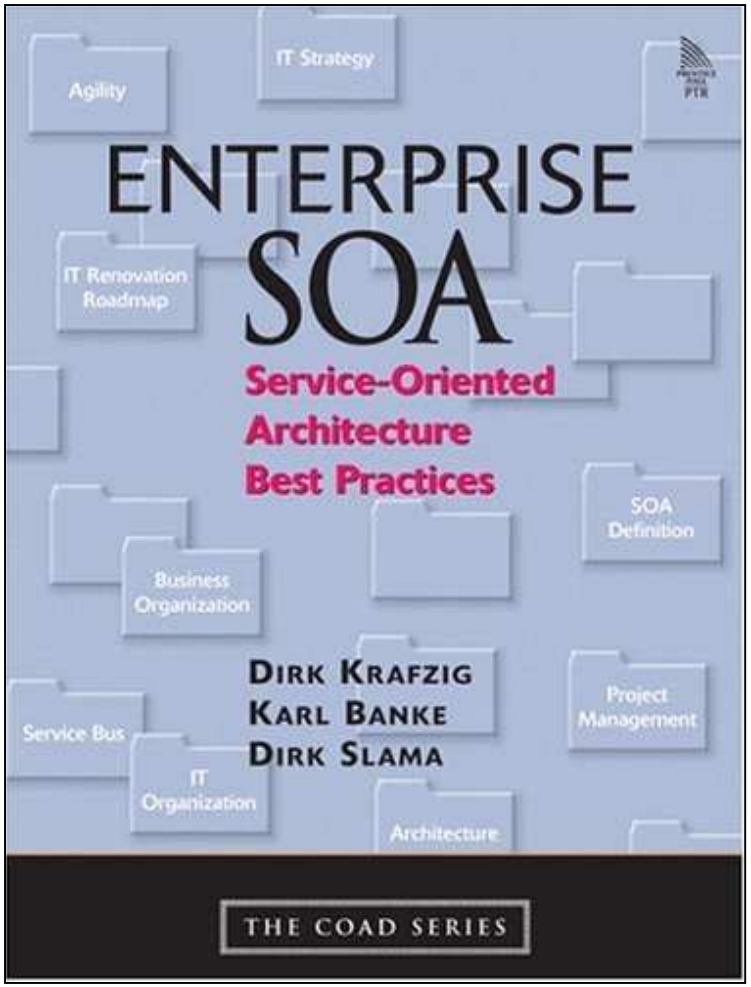
- *EDI VAN*
- RosettaNet
  - CIDX
  - PIDX
- ebXML
- EDIINT AS1/AS2/AS3
- BizTalk Framework 2.0
- FTP, FTPS (over SSL), SFTP (SSH), …

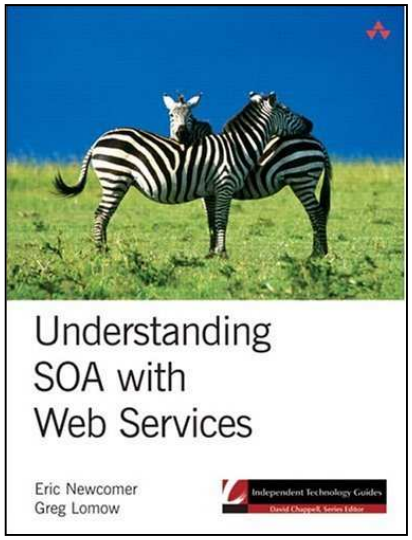
B2B

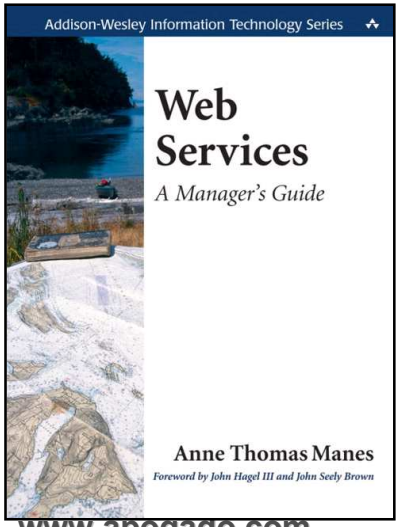
- Almost **no Web Services**
  - SwA: BizTalk Framework and ebXML
  - XML over HTTP, FTP, ...
  - EDIINT: can carry XML, but mostly EDIFACT & X12
- Acknowledgements
  - EDIINT: Message Disposition Notification
- Security
  - SSL of course
  - RosettaNet & EDIINT: S/MIME and PKCS7
  - ebXML: XML Signing (pre-dates WS-Security)

Recommended Reading

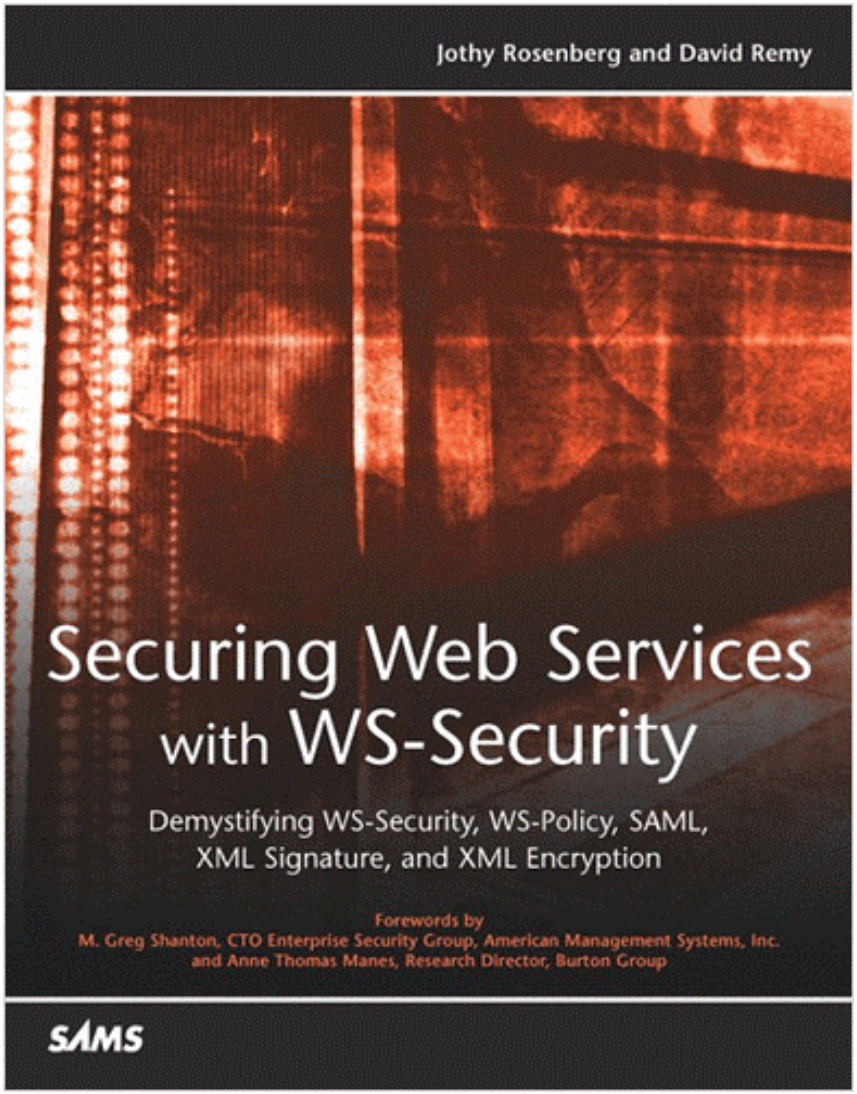


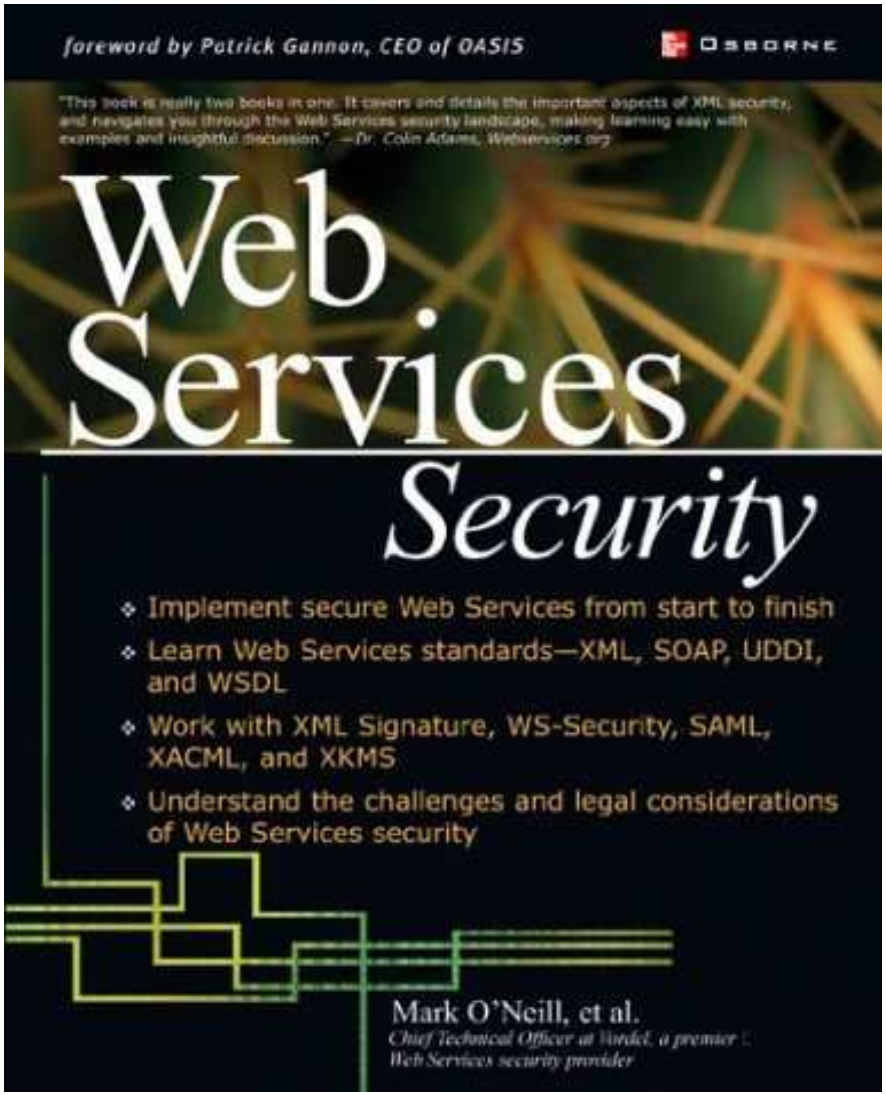






Recommended Reading





soapUI

SoapUI 1.0b1

File Window

Projects
- Amazon
  - AWSECommerceServicePortType
    - BrowseNodeLookup
    - CartAdd
    - CartClear
    - CartCreate
    - CartGet
    - CartModify
    - CustomerContentLookup
    - CustomerContentSearch
    - Help
    - ItemLookup
    - ItemSearch
      - Request 1
    - ListLookup
    - ListSearch
    - MultiOperation
    - SellerListingLookup
    - SellerListingSearch
    - SellerLookup
    - SimilarityLookup
    - TransactionLookup
  - Amazon Tests
    - Simple TestCase
- jira
  - JiraSoapService

Interface Properties

Property	Value
Definition URL	http://webservices....
Binding	{http://webservice...
PortType	AWSECommerceSer...

Windows Details

Amazon.AWSECommerceServicePortType.ItemSearch: Request 1

http://soap.amazon.com/onca/soap?Service=AWSECo... UTF-8

```
<soapenv:Envelope xmlns:soapen
   <soapenv:Body>
      <ns:ItemSearch>
         <ns:SubscriptionId>05
         <ns:Request>
            <ns:Title>Oxford</
            <ns:SearchIndex>Bo
         </ns:Request>
      </ns:ItemSearch>
   </soapenv:Body>
</soapenv:Envelope>
```

```
<SOAP-ENV:Envelope xmlns:SOAP-ENC="http://scher
   <SOAP-ENV:Body>
      <ItemSearchResponse xmlns="http://webserv
         <OperationRequest>
            <HTTPHeaders>
               <Header Name="UserAgent" Value="
            </HTTPHeaders>
            <RequestId>00DDEVKCNJVWAR6PTMEX</Re
            <Arguments>
               <Argument Name="Service" Value="
            </Arguments>
            <RequestProcessingTime>0.1598229408
         </OperationRequest>
         <Items>
            <Request>
               <IsValid>True</IsValid>
               <ItemSearchRequest>
                  <SearchIndex>Books</SearchIn
                  <Title>Oxford</Title>
               </ItemSearchRequest>
            </Request>
            <TotalResults>24490</TotalResults>
            <TotalPages>2449</TotalPages>
            <Item>
```

- no response -

```
09:25:33 : INFO : Got response for [AWSECommerceServicePortType.ItemSearch:Test Book Search] in 301ms (7352 bytes)
09:27:18 : INFO : Got response for [AWSECommerceServicePortType.ItemSearch:Test Music Search] in 281ms (7607 bytes)
09:27:59 : INFO : Got response for [AWSECommerceServicePortType.ItemSearch:Test Software Search] in 280ms (7607 bytes)
09:28:20 : INFO : Got response for [AWSECommerceServicePortType.ItemSearch:Test Software Search] in 9324ms (6365 bytes)
```

Conclusions

- WS-standards are "settling"
  - WS-Security + WS-RM + WS-Addressing
  - More patience (why does it take so long?)
- Lessons from previous technologies, e.g. EDI
- WSDL first, know your XML (Schema's)
- Make your web service secure
  - And "Asynchronous"
- EAI/ESB as "stepping stone"

Q&A

Guy Crets

guy.crets@apogado.com

+32.(0)479.27.36.58

Apogado CVBA

www.apogado.com

www.integrationengineers.com

WWW.JAVAPOLIS.COM