Copyright © 2003 Jorgen Thelin / Cape Clear Software

# Web Services Security Tutorial

A Web Services Security Overview and Implementation Tutorial

CAPE CLEAR

www.capeclear.com

## Copyright Notice

- Copyright © 2003 Cape Clear Software.
  All rights reserved.

- The document is not intended for production and is furnished “as is” without warranty of any kind.
- All warranties on this document are hereby disclaimed including the warranties of merchantability and fitness for a particular purpose.

- The presentation content is provided for your personal information only. Any commercial or non-commercial use of the presentation in full or part, including use of any text or graphics requires a license from Cape Clear Software Inc.

- This presentation is protected by the US and EU copyright regulations and international treaties.

- Trademarks
  - Cape Clear, Cape Clear Software, Cape Clear Studio, Cape Clear Server, and Cape Clear Manager are trademarks of Cape Clear Software in the United States and other countries.
  - Other company, product, and service names mentioned in this product may be trademarks or service marks of others.

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## Abstract

- This tutorial provides an assessment of the various security concerns and implications for XML Web Services, and the different means to address them.
- A framework is presented outlining the variety of measures and approaches for achieving end-to-end security for Web Services, leveraging any pre-existing security environments where possible.
- The various technical security aspects of authentication, authorization, confidentiality and integrity are explored, along with how they affect Web Services and how they relate to the business-driven security concepts of identity, single-sign-on, privacy, trust and non-repudiation.
- An overview is provided of the emerging XML security standards such as XML Digital Signatures (XML-DSIG), XML Encryption, Security Assertions Markup Language (SAML) and WS-Security, including how they combine to address the fundamental security requirements of line-of-business Web Services.
- Examples are shown of a common technique for implementing the security requirements for a Web Service application through the use of custom or pre-built client-side and server-side interceptor plugins, in a manner similar to existing Aspect-oriented programming concepts.
- Finally, some lessons from the initial experiences implementing and using Web Services security are provided, along with advice and guidance for future projects.

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## Modules

- Web Services Security Requirements
- Web Services Security Framework
- Web Services Usage Scenarios
- Security Credentials and Identity
- Emerging XML Security Standards
- Security Implementation using Interceptors
- Lessons from Implementing Web Services Security

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## Resources

- CapeScience
  - Papers, articles, tutorials, and webcasts for Web Services developers
  - http://www.capescience.com
- Jorgen Thelin's Weblog
  - Weblog covering enterprise systems development, and especially Web Services
  - http://www.TheArchitect.co.uk/weblog/

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

# A Web Services Security Framework

Jorgen Thelin
Chief Scientist
Cape Clear Software Inc.

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## 4 Main Concerns of a Security Framework

- Authentication – identity
  - Who is the caller?
  - How do we prove they are who they say they are?
- Authorization – access control
  - What is the caller authorized to do?
  - Is the caller permitted by perform the operation it is requesting?
- Confidentiality – encryption
  - How do we prevent snoopers viewing our messages and data?
- Integrity – tamper-proofing
  - How do we prevent messages being tampered with between sender and receiver?

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

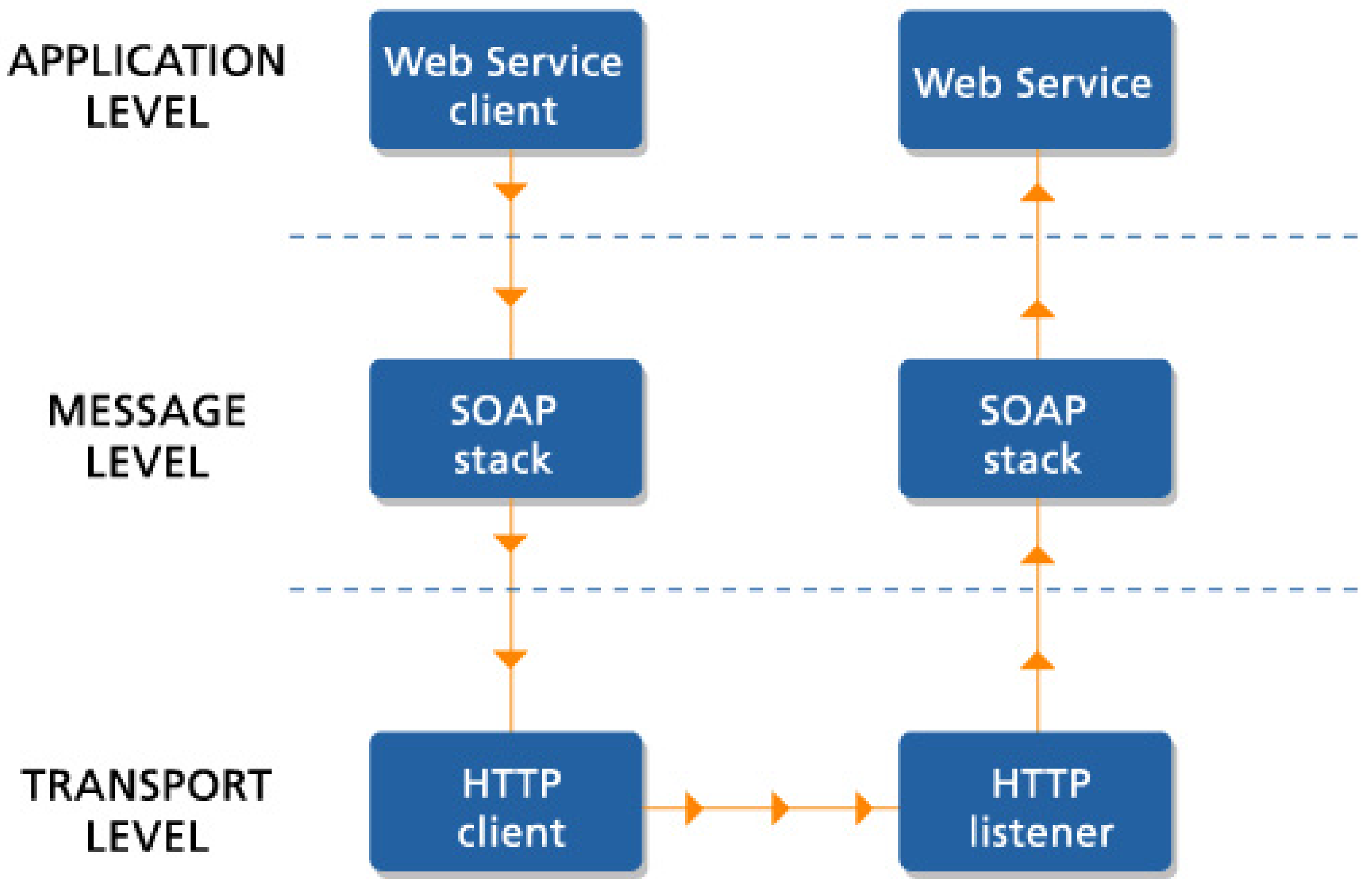
## Non-Repudiation

- This is ultimately the major business requirement for a security framework
  - Can a trading partner possibly claim that:
    - They didn't send a message
    - They sent a different message from the one you received
  - Requires framework support for:
    - **Authentication** – we know who sent the message
    - **Integrity** – the message did not change in transit
    - **Audit record storage** – we can prove what happened

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## Web Service Interaction Levels



CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## Transport Level Security

- Uses existing Web tier technology such as HTTP and SSL
- Authentication
  - HTTP authentication schemes – Basic or Digest
  - SSL client side certificates
- Authorization
  - URL access control policies in the web tier
  - J2EE Servlet declarative security constraints
- Confidentiality
  - SSL encrypted connections
- Integrity
  - Point-to-point SSL encryption to avoid data interception

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## Message level security

- Security data built in to the XML message text – usually as additional SOAP header fields
- Authentication
  - SSO (single sign-on) header tokens
  - SAML authentication assertions
- Authorization
  - SSO session details
  - SAML attribute assertions
- Confidentiality
  - XML Encryption specification
- Integrity
  - XML Digital Signatures specification

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## Application level security

- A Web Service application handles its own security scheme – for example, UDDI
- Authentication
  - App specific authentication messages
  - App specific credential headers in other messages
  - App maintains its own security domain
- Authorization
  - App performs its own access control checks
- Confidentially
  - App can apply an encryption scheme to some or all data fields
- Integrity
  - XML Digital Signature can be used for tamper detection
  - App specific integrity data such as MD5 hash can be sent for some or all data fields

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## Conclusions – Key Issues

- A Web Services security framework must support existing security products
- Must be an end-to-end framework to avoid any security gaps
- New XML security specifications are not yet stabilized or proven
- Use existing proven Web tier security infrastructure until XML security specifications and infrastructure is validated
- WS-I Basic Security Profile will deliver a standardized XML security infrastructure over time

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## Web Service Usage Scenarios

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## Current Usage Scenarios

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## Web Services – Current Usage Scenarios

- Enterprise Application Integration
- Re-use of Existing Business Logic
- Deploying Applications across Firewalls
- EJB Component Reuse
- Ad-hoc Reuse

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## Enterprise Application Integration

- Usage Scenario Description:
  - SOAP can be used to integrate Java and EJBs with logic deployed in other enterprise systems such as CORBA and .NET.
  - The best initial projects for Web Services in organizations often involve the reuse of existing back-end systems – with Web Services used to expose them in a new way.
  - This approach has the added benefit that the focus of the project has been the Web Services rather than developing some new business logic.
- Security Implications:
  - For internal integration, the security implications for this have tended to depend on factors such as the sensitivity of the internal information being passed around and whether the information ever moves beyond the internal firewall at any point (which can happen if, for example, branch offices are connected over the Internet).

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## Re-use of Existing Business Logic

- Usage Scenario Description:
  - Exposing back-end logic to multiple types of clients at the same time, such as Visual Basic and Java GUIs.
  - Many projects have an attractive value proposition for using mainstream developers (Visual Basic programmers, for example) to develop the front-end clients while reserving the EJB programmers (a relatively small percentage of the very best software developers) for developing business logic.
- Security Implications:
  - The security implications for this have tended to depend on factors such as the sensitivity of the internal information being passed around (with authentication and access control being common security solutions) and whether the information ever moves beyond the internal firewall at any point (which can happen if, for example, branch offices are connected over the Internet) SSL has tended to be used in the cases where deployments have been across a combination of intranets and the Internet.

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## Deploying applications across firewalls

- Usage Scenario Description:
  - SOAP (when HTTP is used as the transport layer) can be used to integrate applications or clients across firewalls.
  - This has been particularly useful for projects deadlines that need to avoid the organizational issues usually involved with firewalls.
  - This also has been useful for projects that involved integrating with business partners with heterogeneous firewall security requirements.
- Security Implications:
  - The security implications of what is essentially a shortcut are often ignored due to tight deadlines.

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## EJB Component Reuse

- Usage Scenario Description:
  - The UDDI repository can be used by organizations to make their existing business systems available for reuse within their organizations.
  - The value proposition to organizations for such projects is not just the rapid return on investment but also new opportunities.
- Security Implications:
  - Because this is for internal use, organizations to date have been happy with the various user identification systems in the UDDI registries.

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## Ad-hoc Reuse

- Usage Scenario Description:
  - Web Services technology allows organizations to expose existing (business) logic for reuse in ad-hoc EAI projects.
  - This is done by generating WSDL for existing logic (typically component-based logic such as Java, CORBA, or Enterprise JavaBeans) and registering them in a UDDI registry.
  - An EAI project can then be reduced to looking up the registry for a suitable service.  An example is a company implementing the logic for credit card validation once, but making it available for reuse anywhere it is needed.
- Security Implications:
  - The security implications for such projects have tended to be as varied as the projects.

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## Emerging Usage Scenarios

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## Web Services – Emerging Usage Scenarios

- Point-to-point system integration
- Enterprise application integration
- Technology integration
- Business partner collaboration
- Composite business processes
- Reducing I.T. lifecycle costs
- I.T. investment protection

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## Point-to-point system integration

- Usage Scenario Description:
  - Web Services are ideal when ‘Lite’ internal integration needs exist within an organization. ‘Lite’ integration is the transfer of data between two or more systems. A typical scenario is when a company’s employee information needs to be passed into various downstream applications.
  - The threshold, however, stands at more complex integration technology: for example, transaction processing, business process automation, and so on. Web Services excels at communicating data, but currently not at operational processing. When composition of business services is required in a single atomic operation with complex workflow, Web Services do not yet provide such mechanisms.
- Security Implications:
  - The security implications for such point-to-point integration projects will largely depend on factors such as the sensitivity of the internal information being passed around and whether the information ever moves beyond the internal firewall at any point (which can happen if, for example, branch offices are connected over the Internet).
  - Simple communication security technology such as SSL is usually sufficient to address the security problems here.

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## Enterprise application integration

- Usage Scenario Description:
  - Bridging across a complex architecture comprised of multiple systems residing on multiple platforms using different object models based on different programming languages has previously required complex and expensive EAI technology, but Web Services provides a more effective communication technology for this than traditional EAI technology.
  - However in many instances, Web Services currently lack many of the enterprise features of an EAI solution, especially around process management, transactions, administration, and so on, although this will change over time.
- Security Implications:
  - The security implications for such technology integration projects will probably be the most critical technical issue. There are currently no standards for mapping security features across all the different possible technologies being integrated, and this is even true when using established EAI technology to some extent.
  - Web services platform products are now starting to provide a unifying security layer when integrating disparate technologies by including implementations of all the basic security features such as user authentication, access control, activity auditing and reporting that are required for enterprise applications.

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## Technology integration

- Usage Scenario Description:
  - One of the largest categories of usage scenarios for web services at the moment is about the integration of diverse applications build on various different implementation technologies – i.e. true technology integration. This can involve such simple things are Microsoft VB clients talking to Java EJB systems – something that just 12 months ago was considered virtually impossible to achieve.

- Security Implications:
  - Crossing a technology gap such as this usually highlights a corresponding security gap that needs to be addresses also.
  - So for example, a Microsoft VB (Visual Basic) program will most likely be obtaining user identity information from the Windows ActiveDirectory system and the native NT Authentication scheme, while a Java program this VB program needs to talk to may be using JAAS (Java Authentication and Authorization Services) technology to access an LDAP repository and the EJB (Enterprise JavaBeans) declarative security system to control access.
  - Web service platforms and security product vendors typically need to address the security gap associated with the technology gap being bridged in one of two ways:
    - Use products and technology that can "map" credentials and user information between the different security schemes (e.g. mapping Windows ActiveDirectory credentials to LDAP credentials). This can obviously prove increasingly harder as the number of technologies being used increases. This is where products such as Quadrasis' EASI product can add great value in an organization.
    - Provide a unifying security layer in the web services platform that to a large extent can replace the other existing security control mechanisms.

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## Business partner collaboration

- Usage Scenario Description:
  - Until the introduction of Web Services standards, business partners faced a difficult task to integrate their systems. Solutions were almost always once-off, customer integrations. They were difficult to implement and difficult to maintain. Changes at either partner could easily unravel the entire system. Collaboration between multiple partners was strictly the domain of very large companies.
  - For example, a yellow-pages site may be created for automotive parts vendors. A parts-provider may thus desire to provide a Web Service to integrate their services into the marketplace through the UDDI registry.
  - Web Services offer a standards-based way for business partners to collaborate. The usual business and organizational issues will still be the substantive amount of work that is done with a new business partnership. However, a common technology framework ensures that the focus is the business benefits rather than resolving technological integration problems.
- Security Implications:
  - The key security requirement here is for standards to exist to avoid the need to implement a custom security solution for each different partner being communicated with, in the same way that the interaction technology has typically converged to SOAP and WSDL.

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## Composite business processes

- Usage Scenario Description:
  - Once backend services are available in a standardized manner through exposing them with XML Web Services technologies and standards like SOAP and WSDL, it makes the task of reusing these core business services in new applications and new usage scenarios significantly simpler.
  - New business processes can be created by combining together the existing business process components in innovative and exciting new ways, without having to worry about the traditional technology barriers that have hindered much of this work in the past.
- Security Implications:
  - However, this can easily lead to exactly the same sorts of problems with security gaps as found in the Technology Integration usage scenarios unless all the web services being composed utilize the same set of XML security standards. This clearly highlights the importance of mature implementations of standards that have been widely adopted in the industry.

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## Reducing I.T. lifecycle costs

- Usage Scenario Description:
  - There are a number of factors that make Web Services a better choice than older technologies from the perspective of lifecycle costs:
    - Web Services are comparatively cheaper to implement, lowering the investment part of any return-on-investment calculation.
    - Web Services are generally quicker to implement (assuming productivity tools like CapeStudio are used). This results in a faster time to market and lower development costs.
    - Lower ongoing maintenance and transaction costs. For example, because tools like CapeStudio automatically expose application logic without coding, changes can be implemented quickly and seamlessly.
- Security Implications:
  - The trend towards the web services platform providing the unified security policy enforcement layer also creates considerable cost savings in that using a single security system considerably reduces staff training and operations costs.

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## I.T. investment protection

- Usage Scenario Description:
  - By allowing the functionality of existing I.T. systems to be published and re-used through SOAP, WSDL and UDDI is considerably more cost effective than re-designing from scratch. Adding a web services interface onto an existing legacy system can provide a new lease of life for the system, and take away much of the immediate pressure to replace highly complex systems immediately.
  - Using web service technology as the standardized form for publishing and re-using application services also helps to protect future I.T. investment, by providing a degree of separation between the interface definition and the underlying implementation.
- Security Implications:
  - The use of web service security standards based on XML similarly provide a level of future proofing as the implementation of this security framework can be changed while still relying on the technology-neutrality of standards based on XML communications.

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

# Identity, Security and XML Web Services

Jorgen Thelin

Chief Scientist
Cape Clear Software Inc.

E-mail: Jorgen.Thelin@capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## Abstract

- The use of security credentials and concepts of single-sign-on and "identity" play a big part in Web Services as developers start writing enterprise-grade line-of-business applications. An overview is provided of the emerging XML security credential standards such as SAML, along with various "identity" standards such as Passport and Liberty. We examine how "identity aware" Web Service implementations need to be, and the value a Web Services platform can add in reducing complexity in this area, with lessons drawn from experiences using J2EE technology for real-world security scenarios.

CAPE CLEAR www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## Agenda

- The Concept of Identity
- Web Services and Identity
- Interoperable XML Security and Identity
- Examples of Security Credentials in SOAP
- Single-sign-on
- Identity Awareness in Web Services

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## A Definition of Identity

- Definition from Cambridge Dictionaries Online:
  - **Identity**
    - [ noun ]
    - Who a person is, or the qualities of a person or group which make them different from others
    - http://dictionary.cambridge.org/define.asp?key=identity*1+0

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## What is Identity?

- At its most basic, the concept of Identity is about:
  - Who you are
  - How you prove who you are
  - What that allows you to do

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## Identity – Who are you?

- An identity equates to a particular <u>subject</u> or principal
  - For example: Joe Bloggs …
  - … Who lives at 123 My Street, Your Town
- Usually equates to a <u>person</u>, but could also be a <u>group</u>, <u>corporation</u>, or even something like an automated <u>software agent</u> component
- Subjects must be distinguishable

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## Identity – Proof of identity

- How do you prove who you are?
- In real life, this is usually thru some official documents such as:
  - Driving License
  - Passport
- In computing terms, a user has a set of security credentials such as:
  - username + password
  - X509 certificates

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## Identity – Permissions

- What does this identity prove about us?
- What does this identity allow us to do?

- Some real life examples:
  - Holding a UK passport proves I am a UK Citizen
  - Losing my passport does not stop me being a UK Citizen; it just makes it harder to prove that I am.

  - A standard driving license shows I am allowed to drive a car
  - I am not allowed to drive a Heavy Goods Vehicle unless I hold a HGV Driving License

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## Identity – Permissions and Credentials

- The permissions and entitlements for an identity is ultimately determined by the set of credentials that were presented to assert that identity.

- Permissions and credentials are use to make policy enforcement decisions

  - Am I allowed to drive a Heavy Goods Vehicle?
  - Am I allowed to work in the UK?
  - Am I allowed to work in the US?

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## Web Services and Identity

- How does this affect Web Services?
- Security and Identity is a fundamental requirement of any real-world deployment of a Web Services application
- Ultimately all security policy decisions are based on the caller’s identity
- The challenge is to how to represent and prove a caller’s identity in an open and interoperable way.

CAPE CLEAR®
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## Web Services and Identity 2

- Security and identity considerations for a Web Services application:
  - Authentication
    - Who is the caller?
    - How did they prove their identity?
    - Do we trust the source of these credentials?
  - Authorization
    - What is the caller allowed to do?
  - Attributes
    - What other facts do we know about the caller?
      - For example, e-mail address, department, employee number
    - How do we use this attribute information in the application?
      - For example, customizing the data returned based on display preferences

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## Web Services and Identity 3

- To achieve interoperable security and identity, web services require the following
- Standard ways to:
  - Representing security credential data in XML
    - Eg. SAML – Security Assertions Markup Language specification
  - Obtaining credential data
    - Eg. Single-sign-on services such as Microsoft Passport or Liberty Alliance specifications
  - Transport credential data in a SOAP message
    - Eg. SOAP header fields defined in the WS-Security specification

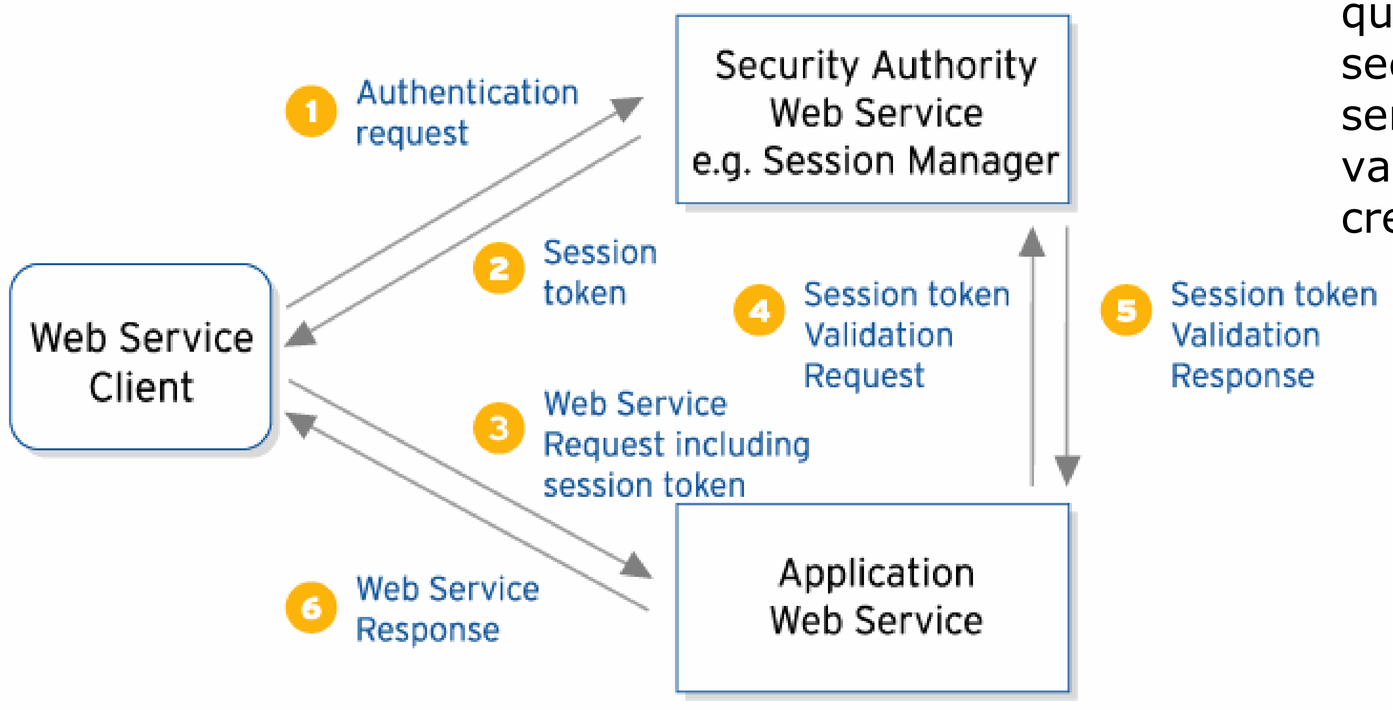
Copyright © 2003 Jorgen Thelin / Cape Clear Software

## Types of Security Tokens

- The WS-Security specification set defines the following tokens:
  - Unsigned security tokens
    - Username
  - Signed security tokens
    - X.509 certificates (binary)
    - Kerberos tickets (binary)
  - XML security tokens
    - Any XML token, such as SAML
    - Usually self verifying / signed

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## Typical XML Security Dialogue – Non Self-Validating Credentials

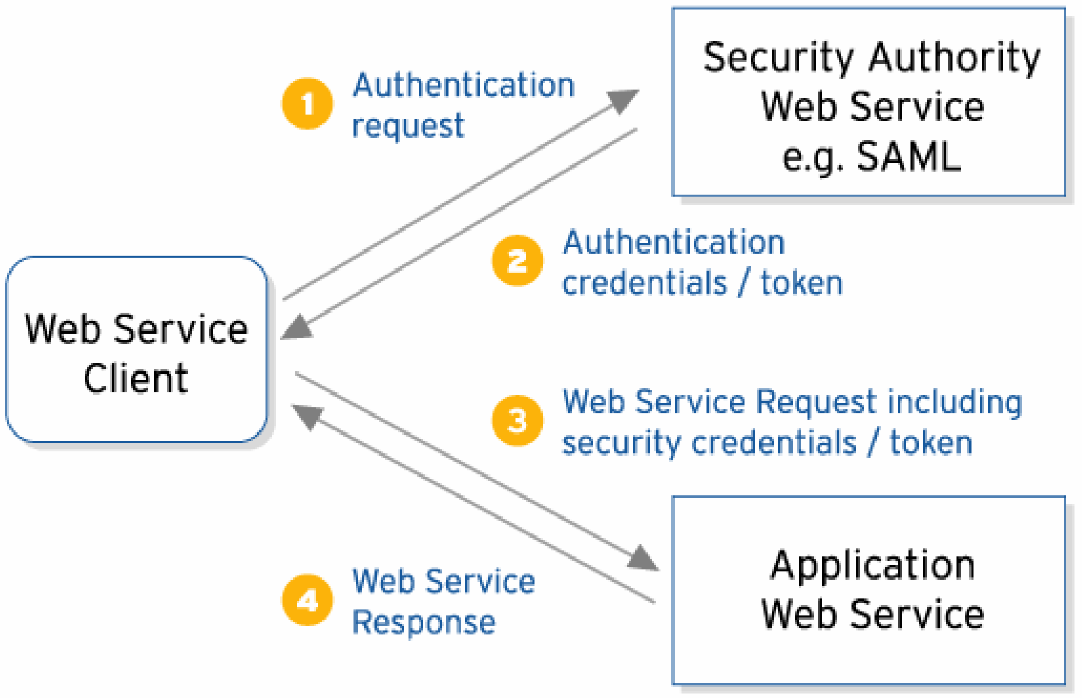

Security Authority Web Service e.g. Session Manager
1 Authentication request
2 Session token
Web Service Client
4 Session token Validation Request
5 Session token Validation Response
3 Web Service Request including session token
6 Web Service Response
Application Web Service

Need to query the security service to validate the credentials

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## Typical XML Security Dialogue – Self Validating Credentials


Web Service Client
Security Authority Web Service e.g. SAML
Application Web Service
1 Authentication request
2 Authentication credentials / token
3 Web Service Request including security credentials / token
4 Web Service Response

No need to query the security service to validate the credentials.

Usually done by the security authority digitally signing the credentials.

CAPE CLEAR www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## SAML v1.0

- SAML – Security Assertions Markup Language
  - An XML-based framework for exchanging security information
  - A specification published by the OASIS organization
- The SAML specification defines:
  - How to represent security credentials ("Assertions" in SAML parlance) using XML
  - An XML message exchange protocol for querying a SAML Authority service
- SAML does not define:
  - How to obtain security credentials ("Assertions") in the first place

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## SAML Assertion Types

- SAML Authentication Assertions
  - The results of an authentication action performed on a *subject* by a *SAML authority*
- SAML Attribute Assertions
  - Attribute information about a *subject*
- SAML Authorization Assertions
  - Authorization permissions that apply to a *subject* with respect to a specified *resource*

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## A Username Token in WS-Security SOAP Header

```
<SOAP:Envelope xmlns:SOAP="...">
    <SOAP:Header>

        <wsse:Security
          xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/12/secext">

         <wsse:UsernameToken>
           <wsse:Username>jthelin</wsse:Username>
           <wsse:Password Type="wsse:PasswordDigest">
           XYZabc123
           </wsse:Password>
           <wsse:Nonce>
           h52sI9pKV0BVRPUolQC7Cg==
           </wsse:Nonce>
         </wsse:UsernameToken>

          ...
        </wsse:Security>

     </SOAP:Header>

     <SOAP:Body Id="MsgBody">
        <!-- SOAP Body data -->
     </SOAP:Body>
    </SOAP:Envelope>
```

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## A Binary X509 Certificate in WS-Security SOAP Header

```
<wsse:Security
  xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/12/secext">

  <wsse:BinarySecurityToken  Id="X509Token"
     xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/12/secext"
     ValueType="wsse:X509v3"
     EncodingType="wsse:Base64Binary"
  >
     MIIEZzCCA9CgAwIBAgIQEmtJZc0...
  </wsse:BinarySecurityToken>

  <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">

    <ds:SignedInfo> ... </ds:SignedInfo>
    <ds:SignatureValue> ... </ds:SignatureValue>

    <ds:KeyInfo>
      <wsse:SecurityTokenReference>
        <wsse: Reference URI="#X509Token" />
      </wsse:SecurityTokenReference>
    </ds:KeyInfo>
  </ds:Signature>

  ...
</wsse:Security>
```

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## A SAML Assertion in WS-Security SOAP Header

```
<wsse:Security
  xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/12/secext">

  <saml:Assertion
    xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
    MajorVersion="1"
    MinorVersion="0"
    AssertionID="SecurityToken-mc375268"
    Issuer="mycompany"
    IssueInstant="2002-07-23T11:32:05.6228146-07:00" >
    ...
  </saml:Assertion>

  <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">

    <ds:SignedInfo> ... </ds:SignedInfo>
    <ds:SignatureValue> ... </ds:SignatureValue>

    <ds:KeyInfo>
      <wsse:SecurityTokenReference>
        <saml:AssertionIDReference>
          SecurityToken-mc375268
        </saml:AssertionIDReference>
      </wsse:SecurityTokenReference>
    </ds:KeyInfo>
  </ds:Signature>

  ...
</wsse:Security>
```

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## Single-sign-on Services

- SSO Services provide:
  - a single point of logon and authentication
  - a standardized way to obtain suitable credentials to prove the authenticated identity
- The main contenders using XML are:
  - Liberty Alliance
  - Microsoft Passport
  - Proprietary security products such as Netegrity SiteMinder are adding direct SAML interfaces
  - WS-Trust – new spec for standardized XML interface
- Still remains an area needing standardization

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## Liberty Alliance

- The Liberty Alliance Project is a cross-industry group aiming to establish an open standard for federated network identity
- http://www.projectliberty.org/
- The Liberty specification v1.0 has two main facets:
  - Single sign-on
  - Identity federation

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## Microsoft .NET Passport

- Microsoft .NET Passport is a suite of Web-based services that makes using the Internet and purchasing online easier and faster for users.
- http://www.passport.com/
- .NET Passport provides users with
  - Single sign-in (SSI)
  - Fast purchasing capability at participating sites
- Microsoft is upgrading the current Passport facilities to
  - Provide an XML interface
  - Support federation
  - Use Kerberos v5 as the underlying mechanism for securely exchanging credentials

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## The Need for a Sign-on Standard – WS-Trust

- The need remains for a “sign-on standard” to avoid reliance on proprietary interfaces
- WS-Trust
  - A proposed specification in the WS-Security family
  - Provides a standardized interface for acquiring security tokens
  - Still very early in the standardization process, but the most likely candidate for a common interface
  - http://msdn.microsoft.com/library/en-us/dnglobspec/html/ws-trust.asp

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## Identity-awareness in Web Services

- Do web services themselves need to be identity-aware?
  - Not really, in most cases
  - A mature web services platform product such as Cape Clear Server can handle almost all the "boilerplate" work of authentication and enforcement of access control lists

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## Identity-awareness in Web Services - 2

- Most standard authentication and authorization functions are best done in a uniform manner by the platform, rather than being implemented on an application-by-application basis
  - Interceptor plugins allow this to be a deployment policy decision rather than an implementation decision
- Web Service application only needs to be Identity-aware if it needs to use attributes asserted for the caller
  - For example, reading the delivery address from the user’s MS Passport record

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## Ultimate Web Services platform security

- Ultimate goal will be declarative security functions for web services just like EJB
  - So, having declarative statements of:
    - Permitted authentication realms / single-sign-on services
    - Required transport security attributes
      (for example, "Callers must use encrypted / SSL connections")
    - Required message security attributes
      (for example, "Messages must be digitally signed")
    - Role-based access control lists applied at the granularity of the operation / method call.
- This places control of security to application <u>administrators</u> rather than developers.

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## Summary

- “Identity” is one of the fundamental concepts in all Web Service security mechanisms
- Having a standard XML-based serialized form of credentials is vital for true end-to-end interoperability
- Standardization of specifications for credential exchange and single-sign-on using XML and SOAP are still incomplete, so true interoperability is not yet possible.
- Use a mature Web Services runtime platform such as Cape Clear Server to handle most “boilerplate” security tasks such as enforcing authentication and authorization requirements

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## XML Security Standards

Current and Emerging Specifications attempting to provide standardization of XML security infrastructure

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

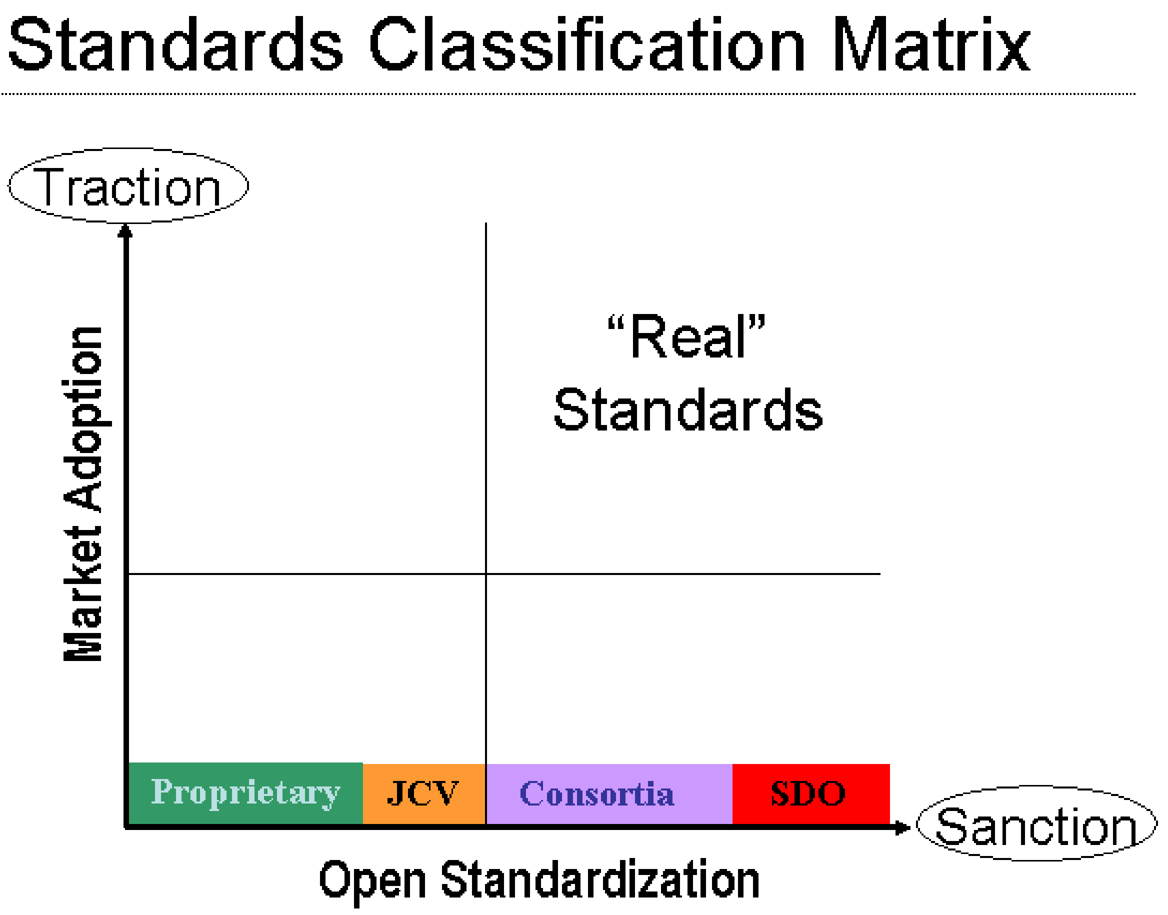
## Specifications and Standards

- There are lots of specifications flying around concerning Web Services
- Not all specifications are, or will be, “real” standards
- The hard part is working out which specifications will “win” and become part of the standard infrastructure
- Vendors and Architects need to plot an “intercept trajectory” for emerging standards

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## “Real” Standards



CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## When is a Specification not a Standard?

- Real standards are:
  - Published by a "recognized" standards development organization – eg. W3C, OASIS
  - Created through a process that allows public comment and feedback
  - Agreed and approved by a committee or group consisting of wide and diverse membership
  - Published at a final or definitive status, such as "W3C Recommendation"
  - Publicly available for reference - most usually by publication on the Internet.
  - Achieving both traction (usage) and sanction (backing)
- Everything else is just a specification hoping to become a standard!

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## Security Standards Overview

- There are several specifications for various aspects of XML and Web Services Security
- The standardization process is still at a very early stage in the evolution
- The front runner specifications are:
  - XML Digital Signatures
  - XML Encryption
  - SAML
  - WS-Security
  - WS-Trust
  - WS-Policy
  - WS-Secure Conversation
  - WS-Security Policy

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## XML Digital Signatures

- **Source:** W3C
- **Status:** Final

- **Purpose:**
  - Specifies a process for digitally signing data and representing the result in XML
  - Define the processing rules and syntax for XML digital signatures

- **Notes:**
  - A serialised form in XML is defined for the signature
  - The signatures can be applied to information in any form, not just XML-formatted information
  - The specification specifically excludes encryption.

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## XML Encryption

- **Source:** W3C
- **Status:** Final

- **Purpose:**
  - Specifies a process for encrypting data and representing the result in XML such that it is only discernable to the intended recipients and opaque to all others

- **Notes:**
  - The information that is encrypted can be arbitrary data (including an XML document), an XML element, or XML element content
  - The result is an XML Encryption element that contains or identifies the cipher data
  - The standard is generally accepted in the industry, although not yet in widespread use

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## SAML

- **Source:** OASIS
- **Status:** Final

- **Purpose:**
  - Uses XML to encode authentication and authorization information in “assertions”

- **Notes:**
  - Defines a standardized XML format for credential and security assertion data
  - The authentication and authorization information can be moved around systems within or between organizations
  - SAML is platform-independent and language-independent
  - A key objective of SAML is to allow organizations to exchange date regardless of the security system they use

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## WS-Security

- **Owner:** Microsoft/IBM/Verisign – Now OASIS WSS-TC
- **Status:** WIP for OASIS standardization process

- **Purpose:**
  - Provides a model for many levels of security needed for web services.
  - A general-purpose mechanism to associate security-tokens with messages
  - Describes how to encode binary security tokens in messages using SOAP Headers
  - Includes enhancements to SOAP to provide quality of protection mechanisms

- **Notes:**
  - Builds on top of XML Digital Signatures and XML Encryption specifications
  - WS-Security Addendum adds
    - Facility for timestamp and TTL headers
    - Provides greater protection when passing around passwords and security certificates

- **More Info:**
  - http://www-106.ibm.com/developerworks/library/ws-secure/
  - http://www-106.ibm.com/developerworks/library/ws-secureadd.html

  - WS-Security AppNotes - provide guidance to implementers of the WS-Security specification:
  - http://www-106.ibm.com/developerworks/webservices/library/ws-secapp/

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## WS-Security - Security Token Types

- The WS-Security specification set defines the following tokens:
  - Unsigned security tokens
    - Username
  - Signed security tokens
    - X.509 certificates (binary)
    - Kerberos tickets (binary)
  - XML security tokens
    - Any XML token, such as SAML
    - Usually self verifying / signed

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## WS-Security Profile for XML-based Tokens

- **Owner:** Microsoft/IBM/Verisign – Now OASIS WSS
- **Status:** WIP for OASIS standardization process
- **Purpose:**
  - Describes a general framework to enable XML-based security tokens to be used with WS-Security
- **Notes:**
  - Two profiles that use this general framework are provided for:
    - Security Assertion Markup Language (SAML)
    - eXtensible rights Markup Language (XrML).
- **More Info:**
  - http://www-106.ibm.com/developerworks/library/ws-sectoken.html

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## WS-Trust

- **Owner:** Microsoft/IBM/Verisign/RSA
- **Status:** Initial public draft release – Soliciting comments

- **Purpose:**
  - Uses the secure messaging mechanisms of WS-Security to define additional primitives and extensions for the issuance, exchange and validation of security tokens.

- **Notes:**
  - WS-Trust also enables the issuance and dissemination of credentials within different trust domains.

- **More Info:**
  - http://www-106.ibm.com/developerworks/webservices/library/ws-trust/

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## WS-Policy

- **Owner:** BEA/Microsoft/IBM/SAP
- **Status:** Initial public draft release – Soliciting comments

- **Purpose:**
  - **WS-Policy Framework**
    - Defines a general purpose model and corresponding syntax to describe and communicate Web services policies
    - Allows Service consumers can discover the information they need to know to be able to access services from a Service Provider
    - http://www-106.ibm.com/developerworks/webservices/library/ws-polfram/
  - **WS-Policy Attachments**
    - Provides a general-purpose mechanism for associating policy assertions with subjects (services).
    - Provides two approaches for making assertions:
      - policy assertions defined as part of the definition of the subject
      - policy assertions defined independently of and associated through an external binding to the subject
    - http://www-106.ibm.com/developerworks/webservices/library/ws-polatt/
  - **WS-Policy Assertions**
    - Specifies a set of common message policy assertions that can be specified within a policy
    - http://www-106.ibm.com/developerworks/webservices/library/ws-polas/

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## WS-Secure Conversation

- **Owner:** Microsoft/IBM/Verisign/RSA
- **Status:** Initial public draft release – Soliciting comments

- **Purpose:**
  - Defines mechanisms for establishing and sharing security contexts, and deriving keys from security contexts, to enable a secure conversation

- **Notes:**
  - Built on top of the WS-Security and WS-Policy models to provide secure communication between services
  - WS-Security focuses on the message authentication model but not a security context, and thus is subject to several forms of security attacks which this specification deals with

- **More Info:**
  - http://www-106.ibm.com/developerworks/webservices/library/ws-secon/

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## WS-Security Policy

- **Owner:** Microsoft/IBM/Verisign/RSA
- **Status:** Initial public draft release – Soliciting comments
- **Purpose:**
  - Defines a model and syntax to describe and communicate security policy assertions within a larger Policy Framework
  - Covers assertions for security tokens, data integrity, confidentiality, visibility, security headers and the age of a message.
- **More Info:**
  - http://www-106.ibm.com/developerworks/webservices/library/ws-secpol/

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## The Extensibility / Composability of XML

- XML is designed to be inherently extensible
- XML allows composable data structures by supporting nested content
  - Extra data can be
- Namespaces allow unique identification of data content
- Composability does not require any registration with a central authority, just a unique namespace

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## Combining Standards / Specifications

- Due to the extensibility features of XML and SOAP, all XML and Security Specifications can generally be combined independently of each other

- For example, add SOAP Headers for:
  - WS-Security X509 Token header to assert identity
  - WS-Policy header to signify:
    - Text encoding requirements
    - Supported languages

- On occasions, ordering of combinations can be significant
  - For example, do you “encrypt” then “digitally sign”, or “digitally sign” then “encrypt”

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## WS-I Basic Security Profile

- From the charter for the new WS-I Basic Security Profile work group:
  - The BSP-WG will develop an interoperability profile dealing with transport security, SOAP message security, and other Basic Profile-oriented security considerations of Web Services
- Although this will not cover all aspects of the emerging XML Security specifications, it will certainly solidify the base levels.

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## Conclusion

- Only partial agreement on the "real standards" at the moment
  - Rival XML security specifications are still emerging
  - XML security standards have not yet been widely adopted
- New XML security standards are not yet proven (so probably contain "holes")
- WS-I Basic Security Profile will deliver a standardized XML security infrastructure

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## Customization Using Interceptors

Using an interceptor-based framework for providing customized client-side and server-side Web Service extension behaviour

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

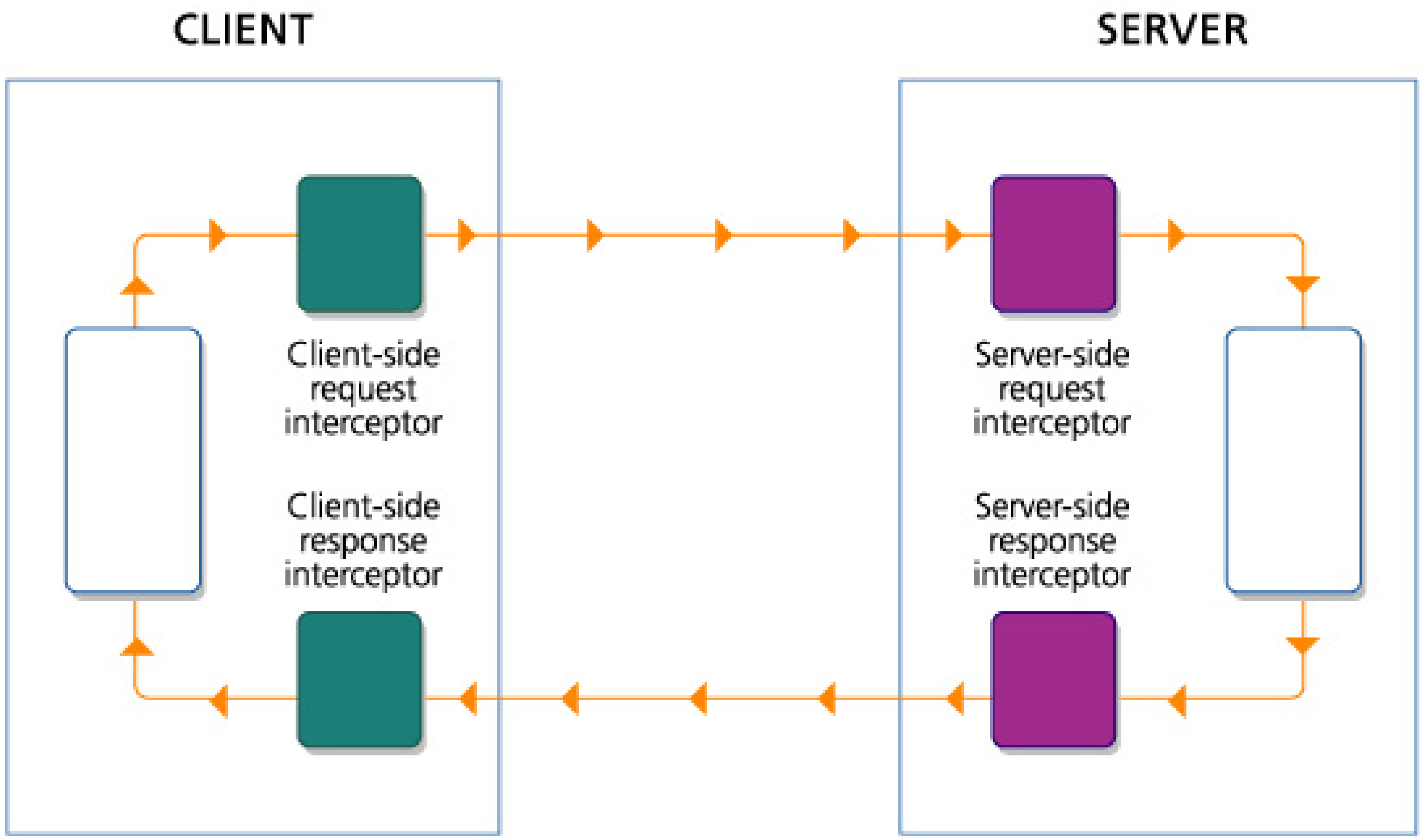
## Interceptors

- Interceptors are a general purpose concept for customizing and controlling message processing
- Interceptors provide a framework for changing the steps involved in processing a message
- Interceptors can be used both server-side and client side

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

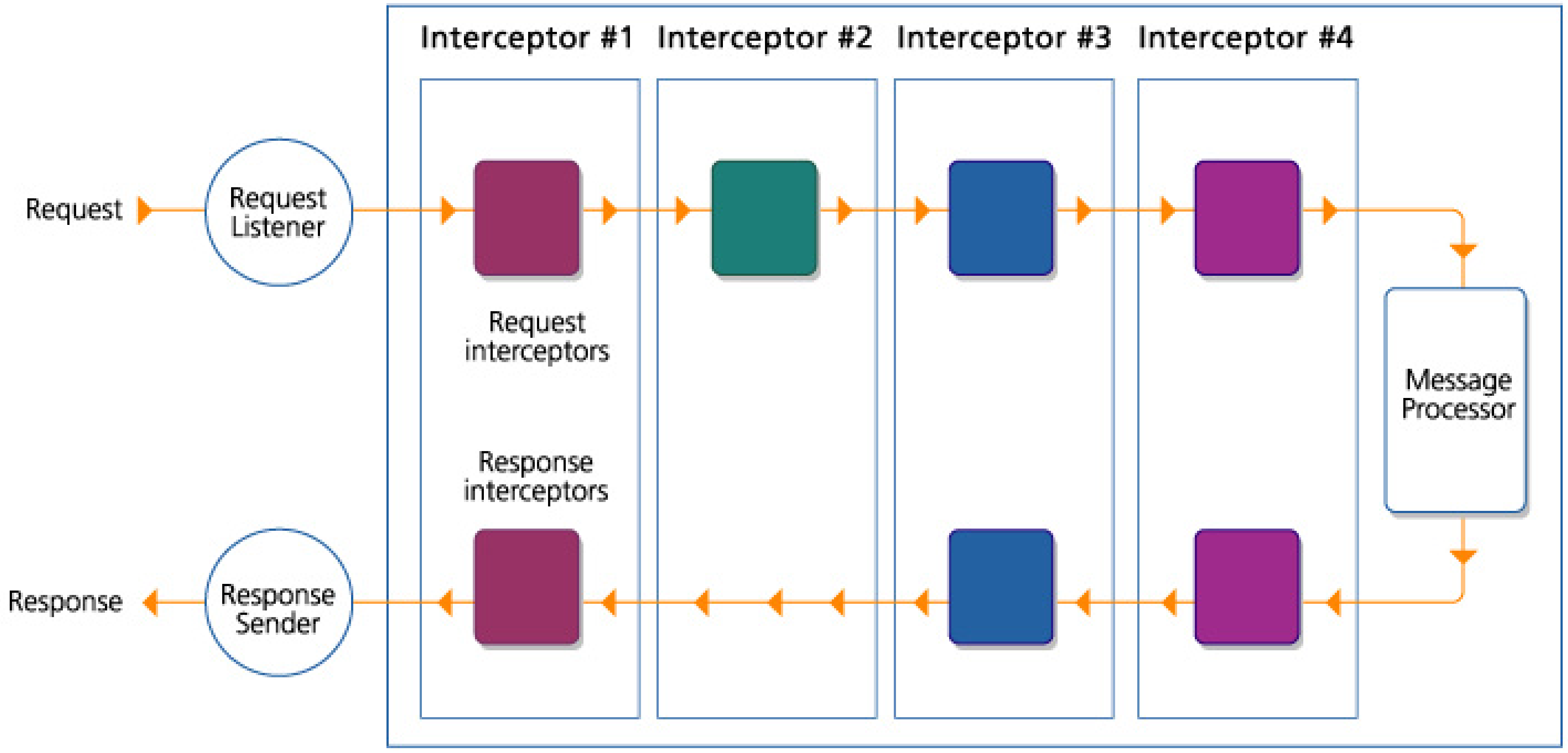
## Interceptor Plugin Architecture


CLIENT
SERVER
Client-side request interceptor
Client-side response interceptor
Server-side request interceptor
Server-side response interceptor

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

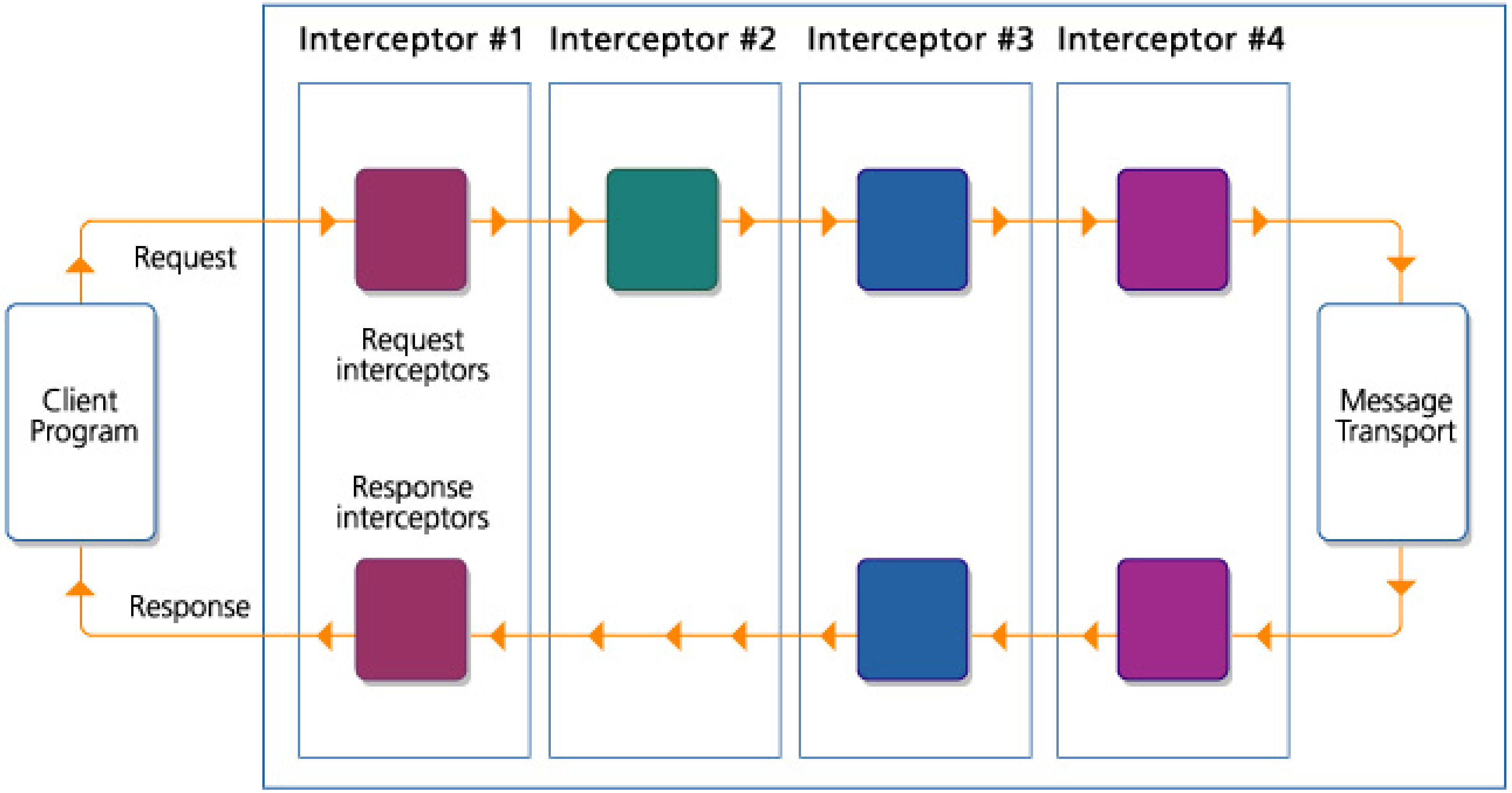
# Interceptor Plugins – Server-side



CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## Interceptor Plugins – Client-side

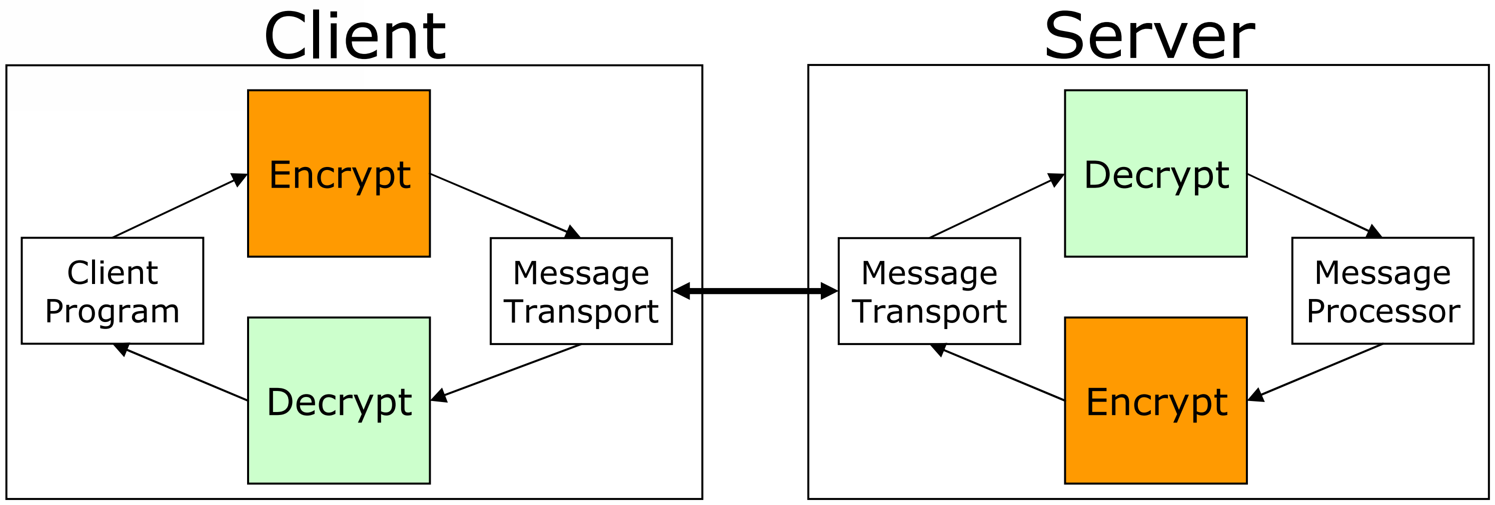


CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## Inbound and Outbound Interception Points

- Often have similar functionality at "opposite" interceptor points
  - Eg. Server-side **response** interceptor may **en**crypt data in the reply just like the client-side **request** interceptor **en**crypted data in the request message
- The **request** message is:
  - Outbound for the client
  - Inbound for the server
- The **response** message is:
  - Inbound for the client
  - Outbound for the server

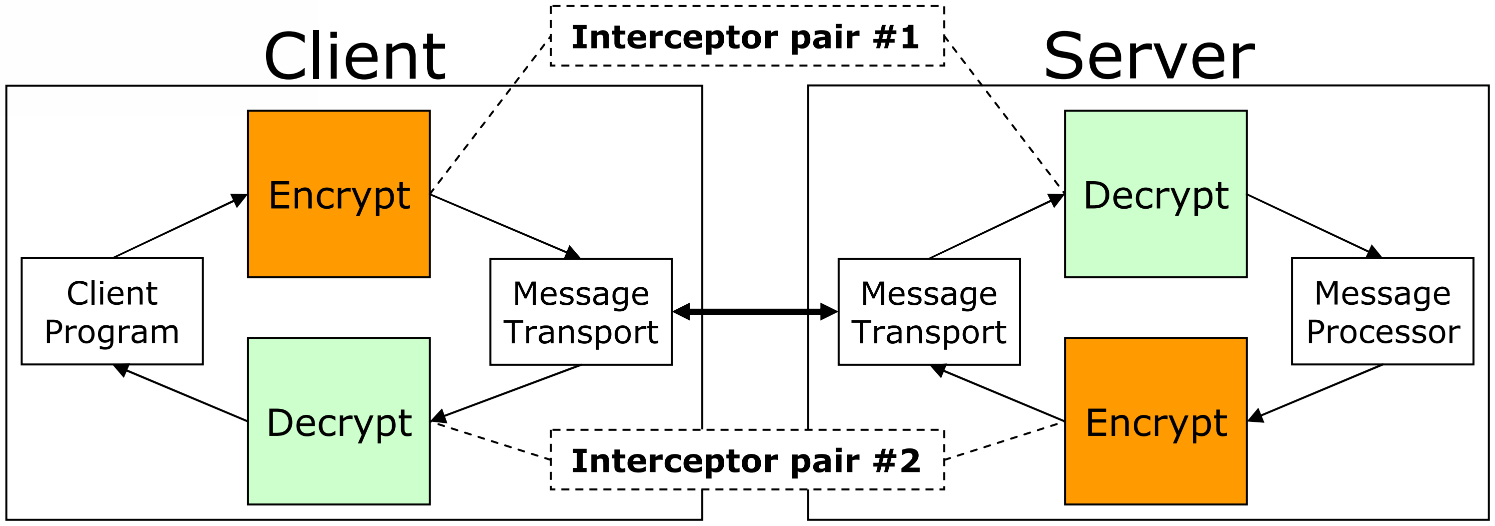


CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## The "Interceptor-Pair" Concept

- Usually require a **<u>pair</u>** of interceptors to fulfil a task
  - One on the client side
    - eg. Request interceptor which <u>encrypts</u> the data in a field
  - One on the sever side
    - eg. Request interceptor which <u>decrypt</u> the field data
- The functions of the interceptor pair need to match up
  - Eg. Both request interceptors cannot both encrypt or both decrypt
- This is often referred to as an input-output interceptor pair



CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## JAX-RPC Handlers (aka "SOAP Interceptors")

- JAX-RPC Handlers
  - Provide a standardized interface for a "SOAP interceptor" in Java
  - Operate on the SOAP object model defined by SAAJ (SOAP with Attachments API for Java)
  - Are part of the JAX-RPC extension module which will be in J2SE 1.4 and J2EE 1.4
  - Provides a MessageContext object for passing call-related context information between interceptors and callbacks

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## JAX-RPC handler interface: javax.xml.rpc.handler.Handler

- JAX-RPC interface javax.xml.rpc.handler.Handler
  - Interception Operations:
    - `boolean` **`handleRequest`**`( MessageContext context )`
      - Called with each request message
    - `boolean` **`handleResponse`**`( MessageContext context )`
      - Called with each response message
    - `boolean` **`handleFault`**`( MessageContext context )`
      - Called with each fault message
  - Interrogation Operations:
    - `QName[]` **`getHeaders`**`()`
      - The names of the header blocks processed by this Handler
  - Lifecycle Operations:
    - `void` **`init`**`( HandlerInfo config )`
      - Called at the start of the lifecycle of the Handler instance
    - `void` **`destroy`**`()`
      - Called at the end of the lifecycle for the Handler instance.

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## Microsoft Web Service Extensions (WSE)

- Web Services Enhancements 1.0 for Microsoft .NET
- WSE is a class library that implements additional Web Services functionality and advanced protocols
  - Diagnostics and tracing
  - WS-Security
  - WS-Routing
  - WS-Referral
- Provides a SoapContext object for passing call-related context information between interceptors and callbacks

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## WSE Filter Interfaces

- Interface **SoapOutputFilter**
  - Interception Operations:
    - `void ProcessMessage( SoapEnvelope envelope)`
  - Lifecycle Operations:
    - *Standard object constructor*
    - *Standard object destructor*
- Interface **SoapInputFilter**
  - Interception Operations:
    - `void ProcessMessage( SoapEnvelope envelope )`
  - Interrogation Operations:
    - `bool CanProcessHeader( XmlElement header, SoapContext context )`
  - Lifecycle Operations:
    - *Standard object constructor*
    - *Standard object destructor*

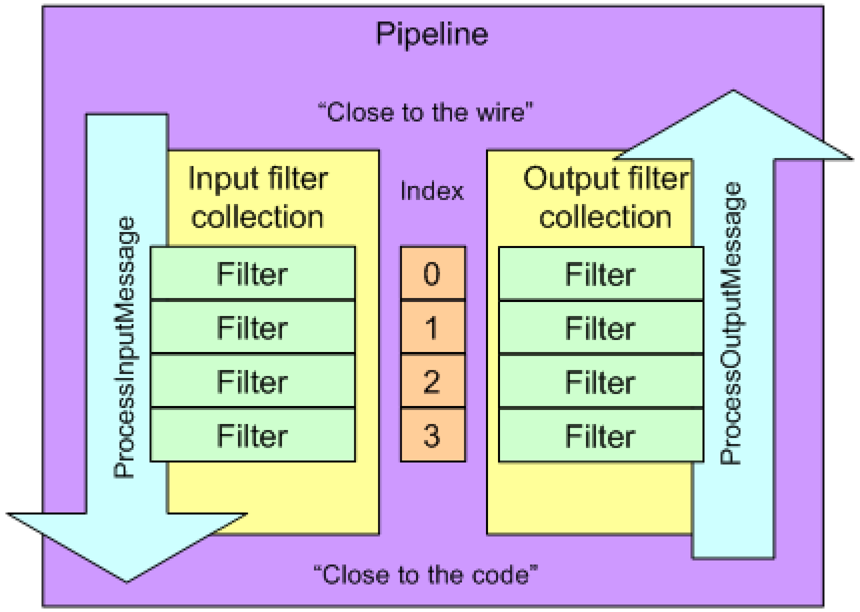
CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## Microsoft Web Service Extensions (WSE) Framework

- WSE is based on the architectural model of a pipeline of **filters** that process inbound and outbound SOAP messages
- The pipeline controls execution order
- Output filters are called in reverse order
- Filters can be integrated with ASP.NET or used in standalone code

**Filter ordering in the WSE pipeline**



Source: MSDN

More info: http://msdn.microsoft.com/library/en-us/dnwebsrv/html/insidewsepipe.asp

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## WSE Built-in Filters

| Namespace | Input Filter | Output Filter | Purpose |
|---|---|---|---|
| Microsoft .Web.Services .Diagnostics | TraceInputFilter | TraceOutputFilter | Write messages to log files to help with debugging |
| Microsoft .Web.Services .Security | SecurityInputFilter | SecurityOutputFilter | Authentication, signature and encryption support (WS-Security) |
| Microsoft .Web.Services .Timestamp | TimestampInputFilter | TimestampOutputFilter | Timestamp support (WS-Security) |
| Microsoft .Web.Services .Referral | ReferralInputFilter | ReferralOutputFilter | Dynamic updates to routing paths (WS-Referral) |
| Microsoft .Web.Services .Routing | RoutingInputFilter | RoutingOutputFilter | Message routing (WS-Routing) |

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## Server-side Interceptors

- Both JAX-RPC Handlers and .NET WSE Filters fit naturally as server-side interceptors
  - Inbound interception points handle requests
  - Outbound interception points handle response

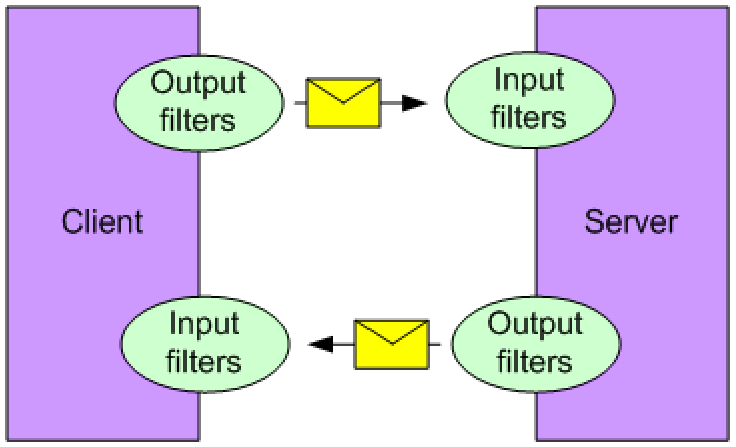
CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## Client-side Interceptors – WSE Filters

- .NET WSE Filters fit in naturally as client-side interceptors too
    - Output filters handle requests
    - Input filters handle responses
- Can directly reuse filters written for server-side use
    - Eg. Decryption interceptor is always an input filter

**The WSE filter model**



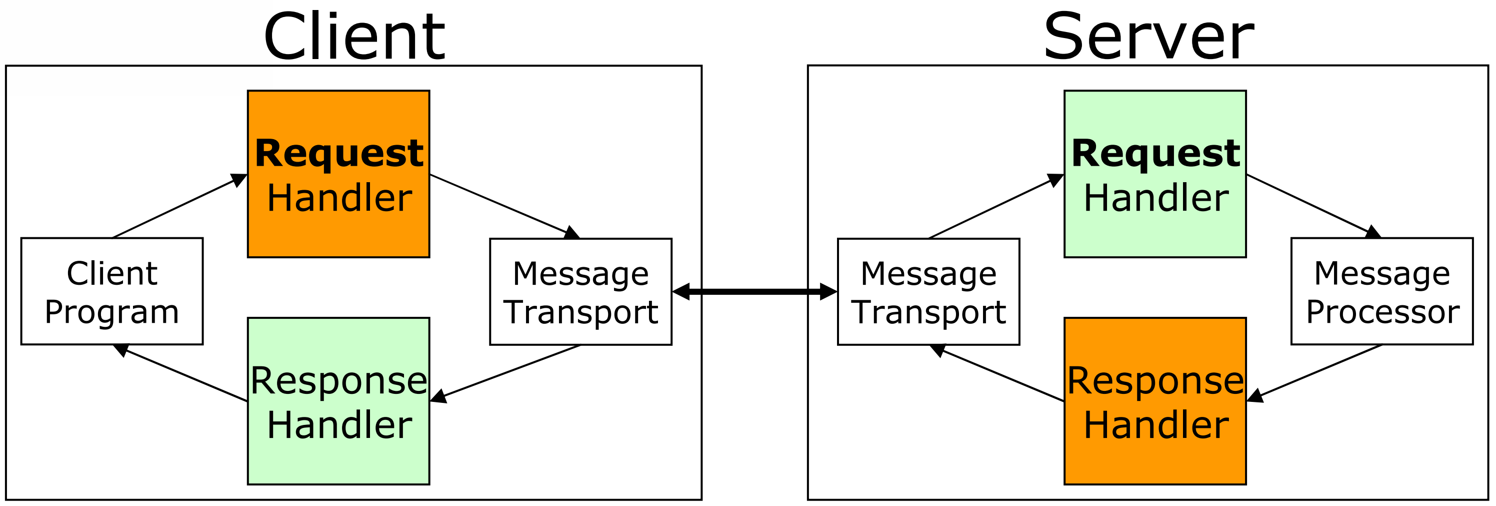
Source: MSDN

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## Client-side Interceptors – JAX-RPC Handlers

- JAX-RPC Handlers do not fit so cleanly as client-side interceptors:
  - **handleRequest** needs to do different things on the client-side (outbound) and server-side (inbound)
  - **handleResponse** needs to do different things on the client-side (inbound) and server-side (outbound)
- Need to write different JAX-RPC Handlers for client and server-side, or use a "configuration flag" to swap behaviour round



CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## Combining Interceptors in “Chains”

- Interceptors are usually independent so can be combined to achieve a desired function

- For example, implementing a security policy for a Web Service:
  - An interceptor to <u>decrypt</u> a SOAP message which was transmitted using XML Encryption
  - An interceptor to recognise and decode <u>SAML authentication assertion</u> carried in the WS-Security header of the SOAP message
  - An interceptor to perform a <u>role-based access control check</u> that the caller is a member of the group of people permitted to call this operation

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## Aspect-oriented Programming (AOP)

- Aspect-oriented Programming (AOP) is a way of implementing **separation of concerns** (SOC) in software.
- AOP make it possible to **modularize** crosscutting "aspects" or "concerns" of a system.
  - For example:
    - Logging policies
    - Diagnostics
    - Transactional contexts
    - Security policy checks
- Separation of Concerns (SOC) makes software much easier to develop, construct and understand
- More info on AOP / AOSD at: http://aosd.net/

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## Interceptors as AOP

- Interceptors are a form of Aspect-oriented Programming (AOP)
  - Plug in an interceptor to deal with a specific function such as message validation or logging
  - Can change the external visible behaviour by adding an interceptor to modify data before or after processing
  - Can change the external visible behaviour by adding an interceptor to “short-circuit” processing
- Interceptors provide an ideal way to implement reusable “policy” aspects of a system
  - For example: Access-control checks
- Interceptors provide an extensibility framework for Web Service protocols and applications
  - For example: Adding WS-Security credentials

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## Example – Custom Authentication Headers

- Scenario:
  - Client program needs to communicate with a Web Service which requires an session-based authentication dialog
  - When calling a "start session" operation, the response message contains a custom header which must be resubmitted with all future requests

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## Example – Custom Authentication Headers

- Implementation:
  - A custom client-side interceptor can be written to deal with this specific situation
  - Client-side response interceptor:
    - Preserve the authentication token found in the SOAP header of the response message
  - Client-side request interceptor:
    - Add any preserved authentication token into a SOAP header of the next request message

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## Example Code – Custom Auth Headers

```
public class SoapCorrelationHeaderInterceptor
   implements Handler
{
   SOAPHeaderElement correlationHeader;  // OK for single-threaded client

   String correlationHeaderElementNamespace;
   String correlationHeaderElementName;
   Name correlationHeaderName;

   public void init( HandlerInfo info )
   {
      Map cfg = info.getHandlerConfig();

      correlationHeaderElementNamespace = (String) cfg.get( "header.ns" );
      correlationHeaderElementName = (String) cfg.get( "header.name" );
   }

   public void destroy()
   {
   }

   public QName[] getHeaders()
   {
      return new QName[] {
         new QName( correlationHeaderElementNamespace, correlationHeaderElementName )
      };
   }

    // More below
}
```

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## Example Code – Custom Auth Headers

```
public boolean handleResponse( MessageContext context )
{
    try {
        // Dig into the response message and extract the contents of the token header

        SOAPMessage soapMessage = ((SOAPMessageContext)context).getMessage();
        SOAPEnvelope soapEnvelope = soapMessage.getSOAPPart().getEnvelope();
        SOAPHeader soapHeaders = soapEnvelope.getHeader();

        if (this.correlationHeaderName == null) {
            this.correlationHeaderName = soapEnvelope.createName(
                correlationHeaderElementName, null,
                correlationHeaderElementNamespace );
        }

        if (soapHeaders != null) {
            this.correlationHeader =
                extractNamedHeader( correlationHeaderName, soapHeaders );
        }
    }
    catch (SOAPException se) {
        throw new JAXRPCExceptionImpl( se );
    }

    return true;  // Continue processing
}
```

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## Example Code – Custom Auth Headers

```
protected SOAPHeaderElement extractNamedHeader(
        Name headerName, SOAPHeader soapHeaders )
        throws SOAPException
    {
        Iterator iter =
            soapHeaders.getChildElements( headerName );

        if (iter.hasNext()) {
            SOAPHeaderElement soapHeaderField =
                (SOAPHeaderElement) iter.next();

            // Remove from SOAP Message element tree
            return soapHeaderField.detachNode();
        }
        else {
            return null;
        }
    }
```

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

# Example Code – Custom Auth Headers

```
public boolean handleRequest( MessageContext context )
{
    if (this.correlationHeader != null) {
        try {
            // Dig into the request message and add the contents of the token header

            SOAPMessage soapMessage = ((SOAPMessageContext)context).getMessage();
            SOAPEnvelope soapEnvelope = soapMessage.getSOAPPart().getEnvelope();
            SOAPHeader soapHeaders = soapEnvelope.getHeader();
            if (soapHeaders == null) { soapHeaders = soapEnvelope.addHeader(); }

            soapHeaders.addChildElement( this.correlationHeader );
        }
        catch (SOAPException se) {
            throw new JAXRPCExceptionImpl( se );
        }
    }

    return true;  // Continue processing
}

public boolean handleFault( MessageContext context )
{
    return true;  // Continue processing
}
```

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## Conclusion

- Interceptors provide a highly extensible processing architecture
- Interceptors allow incremental enhancements to functionality through writing small amounts of code
- Interceptors are the way enhanced protocol support is being added to SOAP platforms
- Interceptors provide an ideal way to implement custom security logic for Web Services

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

# Web Services Security Tutorial - Wrap-up

A Web Services Security Overview and Implementation Tutorial

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## Modules Covered

- Web Services Security Requirements
- Web Services Security Framework
- Web Services Usage Scenarios
- Security Credentials and Identity
- Emerging XML Security Standards
- Security Implementation using Interceptors
- Lessons from Implementing Web Services Security

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## Lessons from the First Wave

- Existing Web tier security infrastructure usually sufficient for many internal projects
- Necessary to accommodate third-party security products already in use in the organization
- End-to-end framework is necessary to avoid security gaps
- Operational security procedure best practices for Web services have yet to be developed
- XML security specifications have not yet been widely adopted
- Rival XML security specifications are still emerging
- Lack of experience and training on XML security standards is holding back adoption

CAPE CLEAR
www.capeclear.com

Copyright © 2003 Jorgen Thelin / Cape Clear Software

## Recommendations for the future

- Track usage scenarios in an organization to determine security levels
- Start with “proof-of-concept” projects to gain experience
- Integration with Microsoft .NET security schemes will be vital
- WS-I Basic Security Profile will reduce interoperability problems with current XML Security standards.
- Don’t throw away the organization’s existing security infrastructure
- Plan to implement end-to-end security

CAPE CLEAR
www.capeclear.com