# CICS Identity and Security

*Leigh Y Compton*
*IBM zGrowth Team*

*lcompton@us.ibm.com*

#SHAREorg

SHARE in Pittsburgh 2014

## Abstract

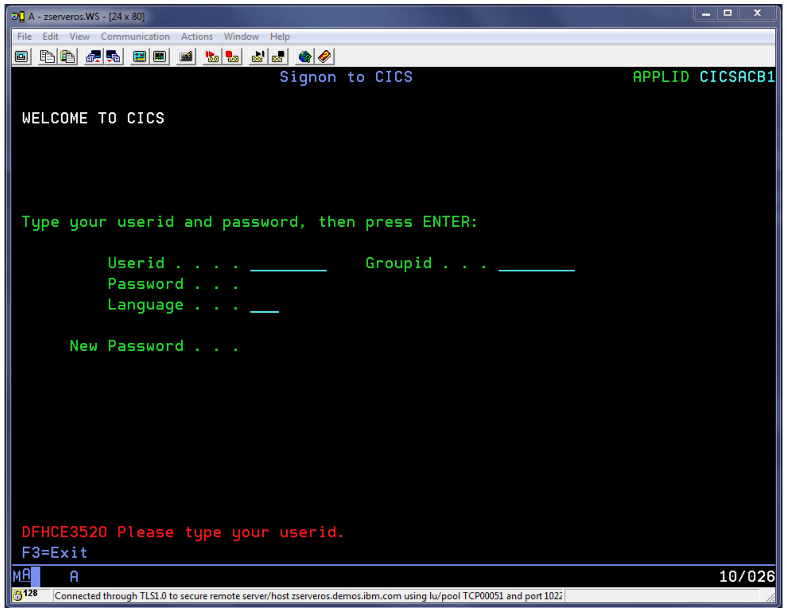
- User identity and security is critical to businesses today.  This session will show how CICS TS V5 participates in today's infrastructure to ensure a secure environment for your applications and data.

## Abstract

- Traditional CICS security
- Security principles
- Security in an interconnected world
  - Network security
  - WS-Security

# Traditional CICS security

- Sign on (i.e. Authentication)
  - User ID plus password (or pass phrase)
  - Terminal oriented
- Authorization checks
  - z/OS Security Manager
    - RACF, ACF2, or Top Secret
  - Transaction security
  - Resource security
  - Command security

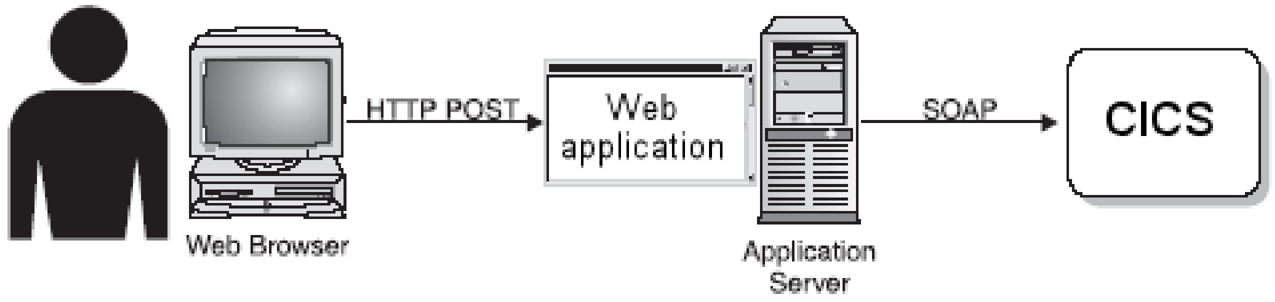

A - zserveros.WS - [24 x 80]
File Edit View Communication Actions Window Help

```
                              Signon to CICS                     APPLID CICSACB1

WELCOME TO CICS




Type your userid and password, then press ENTER:

         Userid . . . . ________     Groupid . . . ________
         Password . . .
         Language . . . ___

     New Password . . .










DFHCE3520 Please type your userid.
F3=Exit
MA      A                                                                 10/026
```

128 Connected through TLS1.0 to secure remote server/host zserveros.demos.ibm.com using lu/pool TCP00051 and port 1022

## Security for modern CICS applications

- Non-terminal access
  - No “sign on” as such
  - Requires alternate means of authentication
    - EXEC CICS VERIFY PASSWORD
    - X.509 Certificate
- Intermediate systems



## Authentication

- Real or genuine, from Greek: αυθεντικός = 'authentes' = author
- Establishing or confirming something (or someone) as authentic
  - Claims made by or about the thing are true
  - Authenticating a person often consists of verifying their identity.
- In computer security:
  - The process of attempting to verify the digital identity of the sender
    - Such as a request to log in
- Sender being authenticated may be:
  - A person using a computer
  - A computer itself
  - A computer program.

-- Wikipedia -- http://en.wikipedia.org/wiki/Authentication

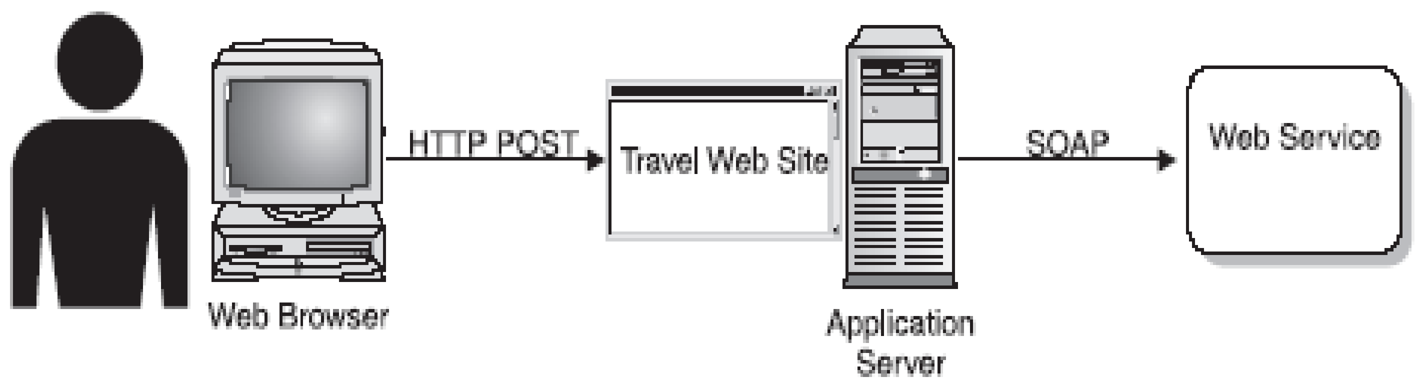
## Authentication

- Real or genuine, from Greek: αυθεντικός = 'authentes' = author
- The act of establishing or confirming something (or someone) as authentic, that is, that claims made by or about the thing are true. Authenticating an object may mean confirming its provenance, whereas authenticating a person often consists of verifying their identity. Authentication depends upon one or more authentication factors.
- In computer security, authentication is the process of attempting to verify the digital identity of the sender of a communication such as a request to log in. The sender being authenticated may be a person using a computer, a computer itself or a computer program.

-- Wikipedia -- http://en.wikipedia.org/wiki/Authentication

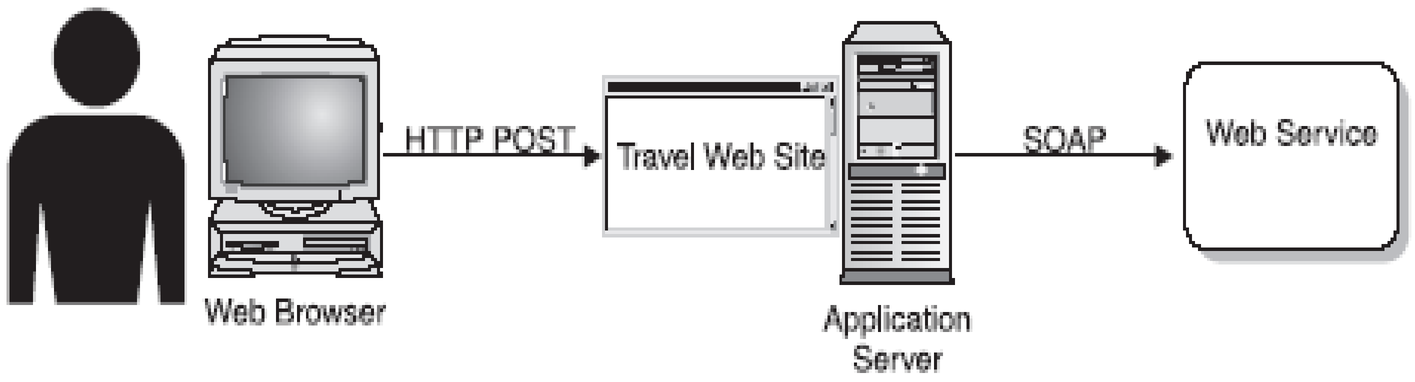
## Identity Propagation

- Supports a downstream server in accepting the client identity that is established on an upstream server, with credentials that allow for authentication.



## Identity Assertion

- Supports a downstream server in accepting the client identity that is established on an upstream server, without having to authenticate again. The downstream server trusts the upstream server.



## Authorization

- A part of the operating system that protects computer resources by only allowing those resources to be used by resource consumers that have been granted authority to use them.
- Resources include individual files or items data, computer programs, computer devices and functionality provided by computer applications.
- Examples of consumers are computer users, computer programs and other devices on the computer.

-- Wikipedia -- http://en.wikipedia.org/wiki/Authorization

## Confidentiality

- Assures that information in storage and in-transit are accessible only for reading by authorized parties.
- Encryption is used to assure message confidentiality.
  - Encryption is the process of scrambling data so as to render it unreadable to all but the holder of the correct decryption key

## And more …

- Intermediaries
  - A basic tenet of Web services is that requests for services may travel through multiple intermediates.
- Non-repudiation
  - Requires that neither the sender nor the receiver of a message be able to legitimately claim they didn't send/receive the message.
- Message integrity
  - Ensures that information, either in storage or in-transit cannot be modified intentionally or unintentionally.
  - Digital signatures are used to assure message integrity.

## Notes

A basic tenet of Web services is that requests for services may travel through multiple intermediates, which may be different divisions of the service company and, in many instances, different companies.

Therefore, from a security point of view, moving to a Web services architecture entails moving from security models based on a client/server paradigm to one where there are multiple intermediates that may or may not be trusted.

Surprising to many is the fact that traditional, transport level security protocols such as SSL and Kerberos decrypt the message at each endpoint thus inappropriately exposing information.

## Across the security divide

- The objective of loosely coupled integration is often completely lost when security integration is added:
  - Linking separate applications often means linking separate user access security processes
  - It's best to manage user identity and access policies outside individual applications
  - Centralized security solutions quickly run out of steam
  - Federated identity across multiple security domains bring greater flexibility

-- Phil Wainewright, Solving the web services identity crisis, Loosely Coupled Digest, April 2004

## Notes

What is the real issue in web services security?

When techies talk about it, they're typically talking about intruders intercepting trusted XML messages and substituting malicious code. Business people, who take this kind of wire-level security for granted, are more concerned about tracking the identities and activities of users who log on legitimately: "Threats come internally within an organization — from within your secure environment," says Christopher Crowhurst, VP enterprise architecture at computer-based assessment provider Thomson Prometric.

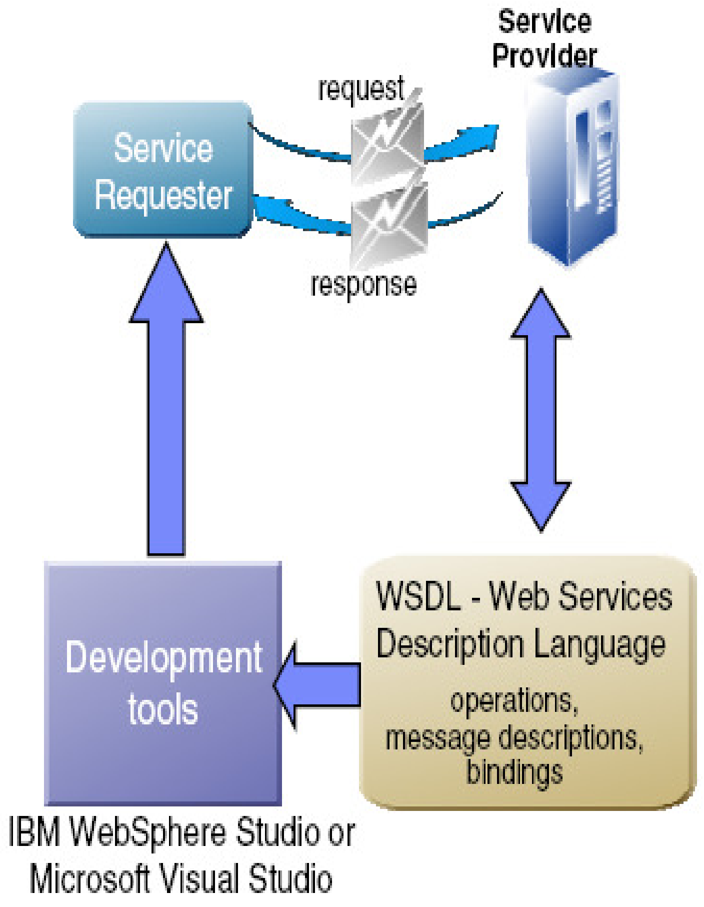
Of course technologists have a duty to master the fine detail of securing the web services infrastructure. But it's no good focussing all your efforts on guarding against XML hackers, when failing to get to grips with the complexities of access and identity management in a web services project could expose sensitive data to the wrong users.

"Questions of identity are perhaps more sophisticated than the security of XML itself," says Mark O'Neill, CTO of web services security vendor Vordel. "If your web service is being accessed through a portal, you have to have a way of tying the web service to the user. The issue is that you have someone authenticating in one place and using a service that's somewhere else." The two processes may be independently secure, but maintaining security when the applications are linked together means integrating their separate mechanisms for authenticating users and controlling access rights.

--- Phil Wainewright, Solving the web services identity crisis,

## Web Services

- SOAP uses XML messages for a request and response model of conversation between programs
- WSDL describes everything a requester needs to use a service.
- UDDI can be used to publish details of one or more services.


Service Provider
request
Service Requester
response
Development tools
WSDL - Web Services Description Language
operations, message descriptions, bindings
IBM WebSphere Studio or Microsoft Visual Studio

## CICS and Web Services Standards

- HTTP: Transport layer *
- SOAP: Describes the Web Services message formats
- WSDL: Describes the interface to a Web Service
- WS-I Basic Profile: Interoperability between providers and requesters using SOAP
- WS-Coordination: Extensible coordination framework
- WS-AtomicTransaction: For transactionality
- WS-Security: Authentication and encryption of all or part of a message
- WS-Trust: Extensions for requesting and issuing security tokens
- XOP and MTOM: optimizing the transmission and format of a SOAP message

* *WebSphere MQ can also be used as the transport layer*

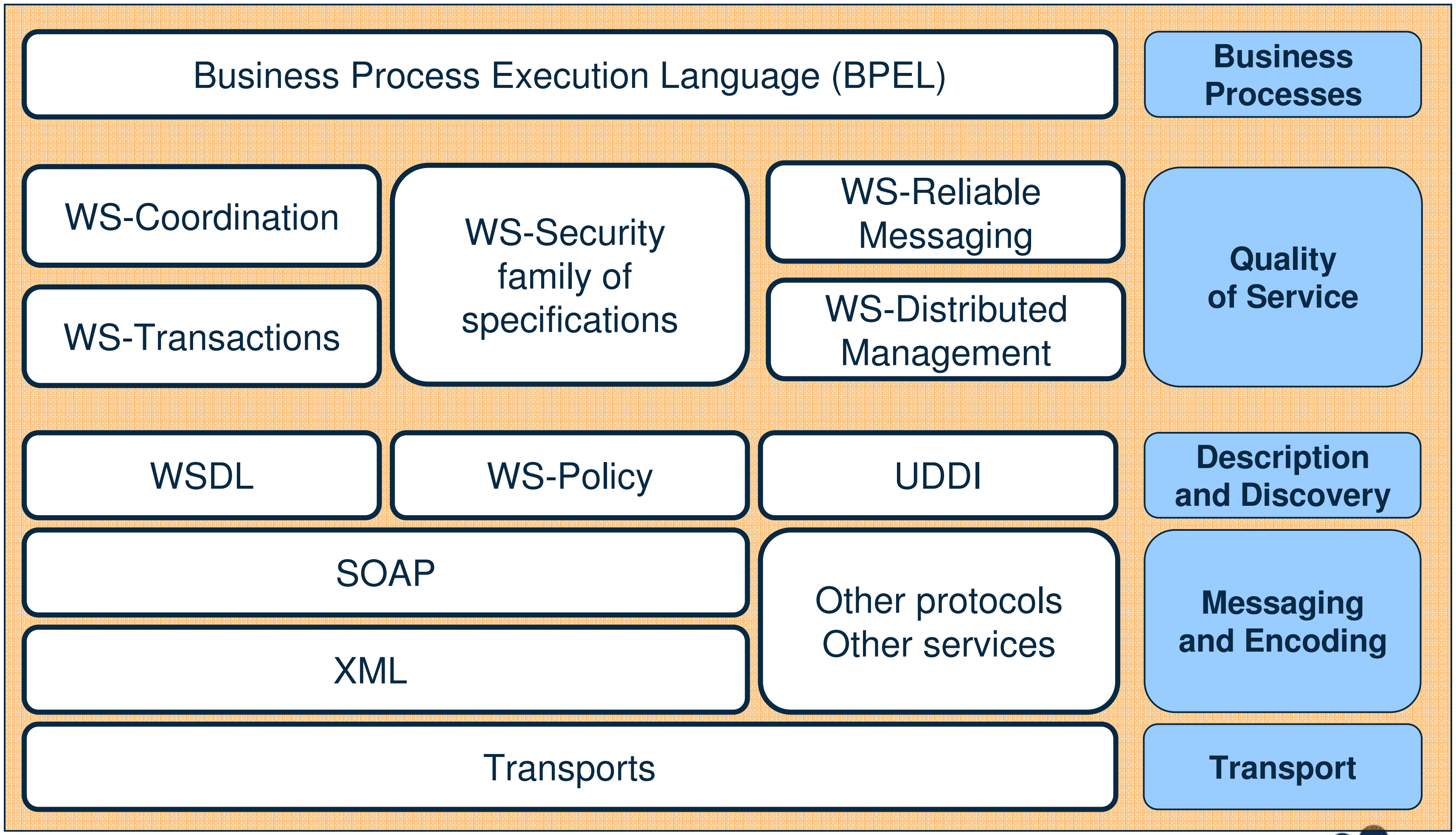
## Notes

CICS support for Web services conforms to a number of industry standards and specifications.

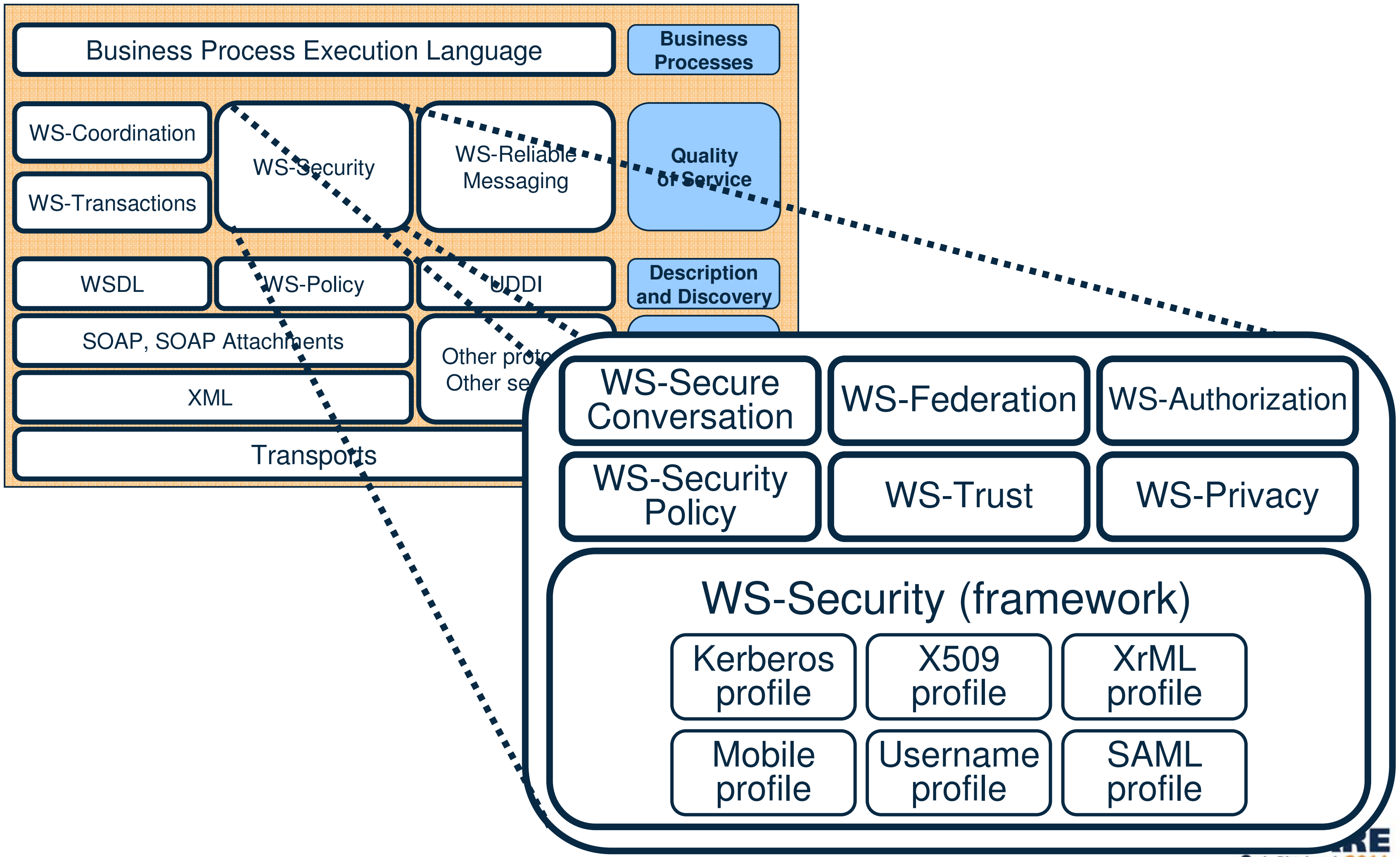
- Extensible Markup Language Version 1.0
- SOAP 1.1 and 1.2
- SOAP 1.1 Binding for MTOM 1.0
- SOAP Message Transmission Optimization Mechanism (MTOM)
- Web Services Addressing 1.0
- Web Services Atomic Transaction Version 1.0
- Web Services Coordination Version 1.0
- Web Services Description Language Version 1.1 and 2.0
- Web Services Security: SOAP Message Security
- Web Services Trust Language
- WSDL 1.1 Binding Extension for SOAP 1.2
- WS-I Basic Profile Version 1.1
- WS-I Simple SOAP Binding Profile Version 1.0
- XML (Extensible Markup Language) Version 1.0
- XML-binary Optimized Packaging (XOP)
- XML Encryption Syntax and Processing
- XML-Signature Syntax and Processing

CICS is compliant with the supported Web services standards and specifications, in that it allows you to generate and deploy Web services that are compliant.

## The Web Services “Stack”


Business Process Execution Language (BPEL)
Business Processes
WS-Coordination
WS-Transactions
WS-Security family of specifications
WS-Reliable Messaging
WS-Distributed Management
Quality of Service
WSDL
WS-Policy
UDDI
Description and Discovery
SOAP
XML
Other protocols Other services
Messaging and Encoding
Transports
Transport

# Web Services and Security



## Notes

CICS® Transaction Server for z/OS® provides support for a number of related specifications that enable you to secure SOAP messages.

The Web Services Security (WSS): SOAP Message Security 1.0 specification describes the use of security tokens and digital signatures to protect and authenticate SOAP messages.

Web Services Security protects the privacy and integrity of SOAP messages by, respectively, protecting messages from unauthorized disclosure and preventing unauthorized and undetected modification. WSS provides this protection by digitally signing and encrypting XML elements in the message. The elements that can be protected are the body, or any elements within the body or the header. Different levels of protection can be given to different elements within the SOAP message.

The Web Services Trust Language (WS-Trust) uses the secure messaging mechanisms of WS-Security to define additional primitives and extensions for the issuance, exchange and validation of security tokens. WS-Trust also enables the issuance and dissemination of credentials within different trust domains.

## Security considerations with SOAP messaging

- how to include security credentials in the message
- how to use element-wise encryption: expose some parts for routing, hide critical data from unauthorized parties
- how to use digital signatures
- security must persist from originator to processing end-point, for the life of the transaction
- security survives call to external business partner
- use with, or instead of, protocol-level security

## Securing Web services

- Securing the transport
  - Authentication
  - Encryption
- Encrypting message
  - SSL
  - WS-Security
- Identity propagation and assertion
  - WS-Security
  - WS-Trust
- Signing messages
  - WS-Security

## Securing the transport

- CICS supports Secure HTTP messages
  - `https://hostname:port/ ...`
  - Using SSL or TLS
- Client authentication
- Mutual authentication
- Encryption

## Extended support for cryptographic standards

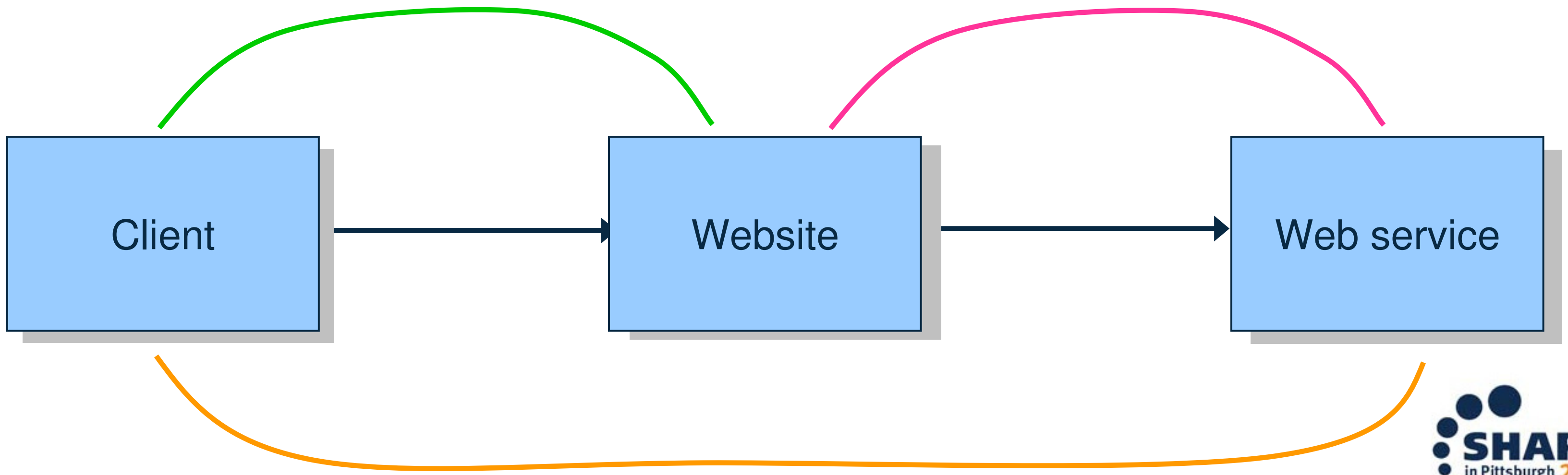
- Support for TLS v1.1 and v1.2
  - APAR PM97207 for CICS TS v5.1
  - Assures compliance with NIST SP800-131A
  - Adds cipher suites from FIPS 140-2
  - Choose FIPS or non-FIPS mode at CICS start-up
- NIST SP800-131A
  - Requirement for US agencies to transition to stronger cryptographic algorithms and longer keys
- FIPS 140-2
  - Security requirements for cryptographic modules

## Defining cipher suites

- CIPHERS – a parameter on resource defintions
  - TCPIPSERVICE
  - URIMAP
  - IPCONN
- Specified as
  - String of hexadecimal 2-character values
    - 35363738392F303132330A1613100D
  - CIPHERs XML file
    - allvalidciphers.xml
    - fipsciphers.xml
    - strongciphers.xml

## HTTPS may not be enough for Web services

- HTTPS is transport-level security
  - Point-to-point
  - Lasts only for duration of the connection
  - All or nothing encryption
  - Weak integrity concept
  - Does not support other security mechanisms



## WS-Security: SOAP Message Security

- A foundational set of SOAP message extensions for building secure Web services
  - Defines new elements to be used in SOAP header for message-level security
- Defines the use of formerly incompatible proven and emerging security technologies:
  - Kerberos, PKI, HTTPS, IPSEC, XrML
  - XML Signature, XML Encryption, XKMS from W3C
  - SAML, XACML from OASIS
- OASIS WS-Security 1.0 standard
  - http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss
  - widely supported in application servers and development tools from several vendors, including IBM, Microsoft, Oracle …

## WS-Security: SOAP Message Security

- Flexible, composable specification
  - Designed to be used as basis for securing Web services
  - Supports a wide variety of industry security models including PKI, Kerberos, and SSL
- Builds upon existing security technologies
  - XML DSIG
  - XML Encryption
- Provides support for:
  - multiple security token formats
  - multiple trust domains
  - multiple signature formats
  - multiple encryption technologies

## Review: SOAP Message Structure

- The SOAP specification defines the “envelope”
  - The envelope wraps the message itself
  - The message is a different vocabulary
  - A namespace prefix is used to distinguish vocabularies
- WS-Security defines the **`<Security>`** element, which allows security extensions to be placed in **`<soapenv:header>`**
  - Username/password
  - Encryption details
  - x.509 certificate
  - Kerberos ticket
  - XrML
  - SAML

## The WS-Security `<Security>` element

The WS-Security specification defines a vocabulary that can be used inside the SOAP envelope. **`<wsse:Security>`** is the “container” for security-related information

```
<S:Envelope
   xmlns:S="http://www.w3.org/2002/06/soap-envelope">
   <S:Header>
      <wsse:Security
         xmnls:wsse="http://schemas.xmlsoap.org/ws/2003/07/secext">
         Security information
      </wsse:Security>
   </S:Header>
   <S:Body> Application specific content </S:Body>
</S:Envelope>
```

## Security Tokens for the `<Security>` element

- A **Security Token** is a collection of one or more "claims"
  - A claim is a declaration made by some entity, such as name, identity, key, group, privilege, capability, etc.
  - "username" is an example of an unsigned security token
- A **Signed Security Token** is one that is cryptographically signed by a specific authority
  - An X.509 certificate is a signed security token
  - A Kerberos ticket is also a signed security token
- An **XML Security Token** is one that is defined with a separate XML schema rather than simple or encrypted text
  - SAML and XrML are examples
  - Can be included directly in **`<wsse:Security>`** container

## WS-Security `<UsernameToken>` element

This element can be used to provide a user name within a `<wsse:Security>` element, for Basic Authentication

```
<S:Envelope
   xmlns:S="http://www.w3.org/2002/06/soap-envelope">
   <S:Header>
      <wsse:Security
         xmnls:wsse="http://schemas.xmlsoap.org/ws/2003/07/secext">
         <wsse:UsernameToken wsu:ID="myToken">
            <wsse:Username>compton</wsse:Username>
            <wsse:Password>up2date</wsse:Password>
         </wsse:UsernameToken>
      </wsse:Security>
   </S:Header>
   <S:Body> Application specific content</S:Body>
</S:Envelope>
```

## WS-Security <BinarySecurityToken> element

Signed security tokens, such as a Kerberos ticket or x.509 certificate are binary content. They must be encoded for inclusion in the wsse:Security container.

```
<S:Envelope
   xmlns:S="http://www.w3.org/2002/06/soap-envelope">
   <S:Header>
      <wsse:Security
         xmnls:wsse="http://schemas.xmlsoap.org/ws/2003/07/secext">
         <wsse:BinarySecurityToken wsu:ID="myToken"
            ValueType="wsse:X509"
            EncodingType="wsse:Base64Binary>
            XIFNWZz99UUbalqIEmJZc0
         </wsse:BinarySecurityToken>
      </wsse:Security>
   </S:Header>
   <S:Body> Application specific content</S:Body>
</S:Envelope>
```

## X.509 Certificate Token

- Public key infrastructure
- Binding between a public key and attributes
  - Subject name
  - Issuer name
  - Serial number
  - Validity interval
  - Others
- Uses:
  - Authentication
  - Encryption
  - Digital Signature

## Kerberos Token

- Kerberos performs authentication
  - as a trusted third-party authentication service
  - using shared secret key cryptography
- Kerberos provides a means of verifying the identities of principals
  - without relying on authentication by the host operating system
  - without basing trust on host addresses
  - without requiring physical security of all the hosts on the network
  - under the assumption that packets that travel along the network can be read, modified, and inserted at will.

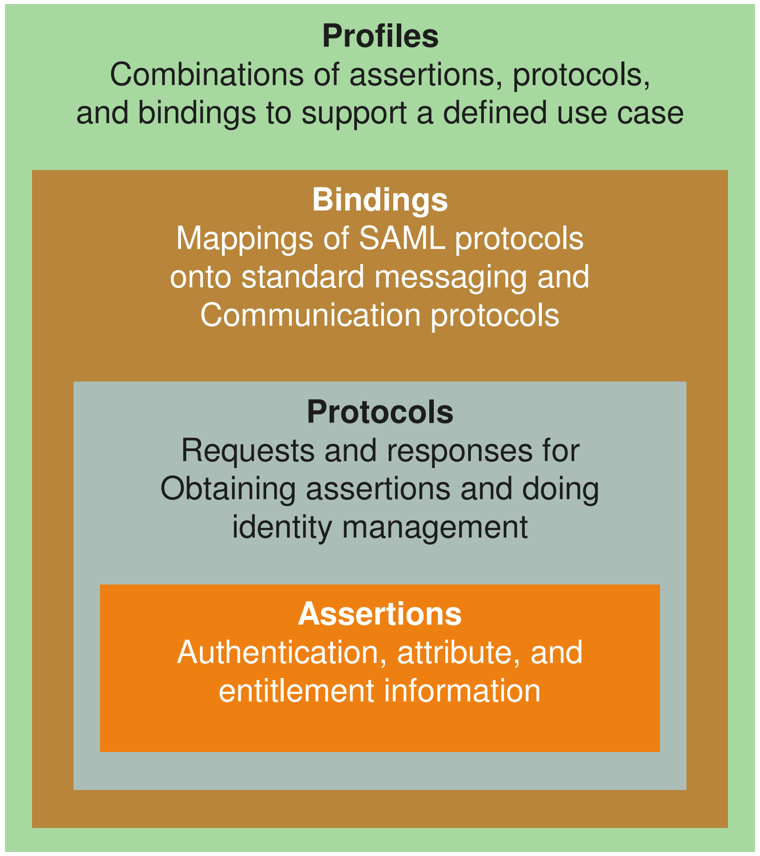
## WS-Security `<saml:Assertion>` element

SAML assertions are attached to SOAP messages by placing assertion elements inside a <wsse:Security> header

```
<S:Envelope
   xmlns:S="http://www.w3.org/2002/06/soap-envelope">
   <S:Header>
      <wsse:Security
         xmnls:wsse="http://schemas.xmlsoap.org/ws/2003/07/secext">
           <saml:Assertion
                      AssertionID="_a75adf55-01d7-40cc-929f-
dbd8372ebdfc"
                      IssueInstant="2003-04-17T00:46:02Z"
                      Issuer="www.opensaml.org"
                      MajorVersion="1"
                      MinorVersion="1" . . .
           </saml:Assertion>
         </wsse:Security>
   </S:Header>
   <S:Body> Application specific content</S:Body>
</S:Envelope>
```

## SAML – Security Assertion Markup Language

- OASIS open standard
- “XML based framework for describing and exchanging security information between on-line business partners.”
- SAML dates from 2001; most recent update from 2005
- Used for:
  - Web SSO
  - Attribute-based authorization
  - Web service security



**Profiles**
Combinations of assertions, protocols, and bindings to support a defined use case

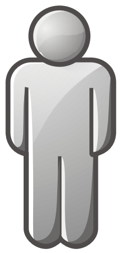
**Bindings**
Mappings of SAML protocols onto standard messaging and Communication protocols

**Protocols**
Requests and responses for Obtaining assertions and doing identity management

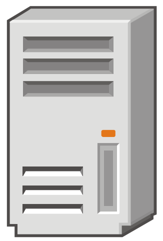
**Assertions**
Authentication, attribute, and entitlement information

## SAML – Roles

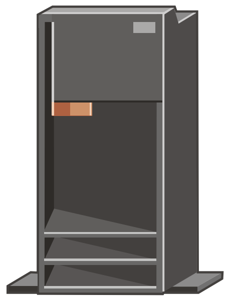
- Principal (or subject of the assertion)



- Identity Provider/Security Token Service (or asserting party)



- Service Provider (or relying party)



## SAML Assertions

- Assertions
  - Collection of statements about principal
  - Made by a SAML authority
- Assertion types
  - Authentication
  - Attribute
  - Authorization decision

## Using XML Digital Signatures with SOAP

- XML Digital Signatures tells us how to sign arbitrary XML content
- How do we use XML Signatures with SOAP messages?
  - WS-Security defines a new element in the SOAP header to hold XML Signature(s) on the content
  - Standardization of these elements allows implementations from different vendors to interoperate with signatures
  - WS-I Basic Security Profile specifies usage details to ensure interoperability

## Example: SOAP with XML Signature

```
<S:Envelope
  xmlns:S="http://www.w3.org/2002/06/soap-envelope">
  <S:Header>
    <wsse:Security S:mustUnderstand="1"
      xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/07/secext">
      <wsse:BinarySecurityToken EncodingType="wsse:Base64Binary">
        MIIDQTCC4ZzO7tIgerPlaid1q ... [truncated]
      </wsse:BinarySecurityToken>
      <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        ....signature data....
      </ds:Signature>
    </wsse:Security>
  </S:Header>
  <S:Body>
    <m:OrderAircraft quantity="1" type="777" config="Atlantic"
       xmlns:m="http://www.boeing.com/AircraftOrderSubmission"/>
  </S:Body>
</S:Envelope>
```

## WS-Security uses W3C XML Encryption

- **`<EncryptedData>`** element replaces the content being encrypted. It contains:
  - **`<EncryptionMethod>`** Algorithm used to encrypt the data
  - **`<CipherData>`**
    - **`<CipherValue>`** Element containing the encrypted data
- **`<EncryptedKey>`** element placed in security header contains:
  - **`<EncryptionMethod>`** Algorithm used to encrypt symmetric key
  - **`<KeyInfo>`** Identifier of key used to encrypt symmetric key
  - **`<CipherData>`**
    - **`<CipherValue>`** Encrypted symmetric key value
  - **`<ReferenceList>`** List of **`<DataReference>`**s to content encrypted with this symmetric key

## Example: entire <body> contents encrypted

```
<PayBalanceDue xmlns='http://example.org/paymentv2'>
  <Name>John Smith<Name/>
  <CreditCard Limit='5,000' Currency='USD'>
    <Number>4019 2445 0277 5567</Number>
    <Issuer>Bank of the Internet</Issuer>
    <Expiration>04/02</Expiration>
  </CreditCard>
</PayBalanceDue >
```

Unencrypted original content
Red text is data to be encrypted
Green text is left unencrypted

```
<EncryptedData xmlns='http://www.w3.org/2001/04/xmlenc#'
  Type='http://www.isi.edu/in-notes/iana/assignments/media-types/text/xml'>
  <CipherData><CipherValue>A23B4C6</CipherValue></CipherData>
</EncryptedData>
```

“PayBalanceDue” element identity is hidden in encrypted form. We can’t even see what kind of transaction it is.

(The real cipher would be longer than this)

## Example: one element and subelements encrypted

```
<PayBalanceDue xmlns='http://example.org/paymentv2'>
  <Name>John Smith<Name/>
  <CreditCard Limit='5,000' Currency='USD'>
    <Number>4019 2445 0277 5567</Number>
    <Issuer>Bank of the Internet</Issuer>
    <Expiration>04/02</Expiration>
  </CreditCard>
</PayBalanceDue >
```

Unencrypted original content
Red text is data to be encrypted
Green text is left unencrypted

```
<PayBalanceDue xmlns='http://example.org/paymentv2'>
  <Name>John Smith<Name/>
  <EncryptedData xmlns='http://www.w3.org/2001/04/xmlenc#'
    Type='http://www.isi.edu/in-notes/iana/assignments/media-types/text/xml'>
    <CipherData><CipherValue>A23B4C6</CipherValue></CipherData>
  </EncryptedData>
</PayBalanceDue >
```

<CreditCard> group was replaced by <EncryptedData> element

## Example: element text (only) encrypted

```
<PayBalanceDue xmlns='http://example.org/paymentv2'>
  <Name>John Smith<Name/>
  <CreditCard Limit='5,000' Currency='USD'>
    <Number>4019 2445 0277 5567</Number>
    <Issuer>Bank of the Internet</Issuer>
    <Expiration>04/02</Expiration>
  </CreditCard>
</PayBalanceDue >
```

Unencrypted original content
Red text is data to be encrypted
Green text is left unencrypted

```
<PayBalanceDue xmlns='http://example.org/paymentv2'>
  <Name>John Smith<Name/>
  <CreditCard Limit='5,000' Currency='USD'>
    <Number>
     <EncryptedData xmlns='http://www.w3.org/2001/04/xmlenc#'
       Type='http://www.isi.edu/in-notes/iana/assignments/media-types/text/xml'>
       <CipherData><CipherValue>A23B4C6</CipherValue></CipherData>
     </EncryptedData>
    </Number>
    <Issuer>Bank of the Internet</Issuer>
    <Expiration>04/02</Expiration>
  </CreditCard>
</PayBalanceDue >
```

Text was replaced by <EncryptedData> element

## CICS Support for WS-Security

- Support is provided by the CICS WS-Security Message Handler
  - DFHWSSE1: shipped via APAR PK22736 in CICS TS V3.1
- SOAP Message Security
  - Signature validation of inbound message signatures, for RSA-SHA1 & DSA-SHA1
  - Signature generation for the SOAP Body on outbound messages using RSA-SHA1
  - Decryption of encrypted data in inbound messages
    - AES 128, 192 & 256 or Triple DES
  - Encryption of the SOAP Body content with the above algorithms
- Various mechanisms for deriving a User ID from an inbound message
  - UsernameToken Profile
  - X.509 Certificate Token Profile
  - SAML Token Profile (V5.2 or Feature Pack for Security Extensions)
  - Kerberos Token Profile (V5.2)

## Signature Generation & Validation

- CICS will verify the validity of any signatures present in inbound SOAP messages.
- Invalid signatures will cause a Security Fault to be raised.
- A CICS Pipeline can be configured to sign the body of outbound SOAP messages.
- The label of the certificate to use and the algorithm required are specified in the Pipeline configuration file.

## Encryption & Decryption

- CICS will decrypt any encrypted elements present in inbound SOAP messages
- Invalid encryption elements will cause a Security Fault to be raised
- A CICS Pipeline can be configured to encrypt the body of outbound SOAP messages
- The label of the certificate to use and the algorithm required are specified in the Pipeline configuration file.

## Certificate & Key Management

- Certificates and Keys are stored in a Key Ring file
  - Managed by the z/OS Security Manager
- The Key Ring used for WS-Security is the same one used for SSL support
  - SAML support introduces optional additional Key Ring file
    - Configured in CICS Security Trust Server region(s)
- Certificates must be propagated between the Requesters and Providers

## Support for UsernameTokens via SOAP Header Handler

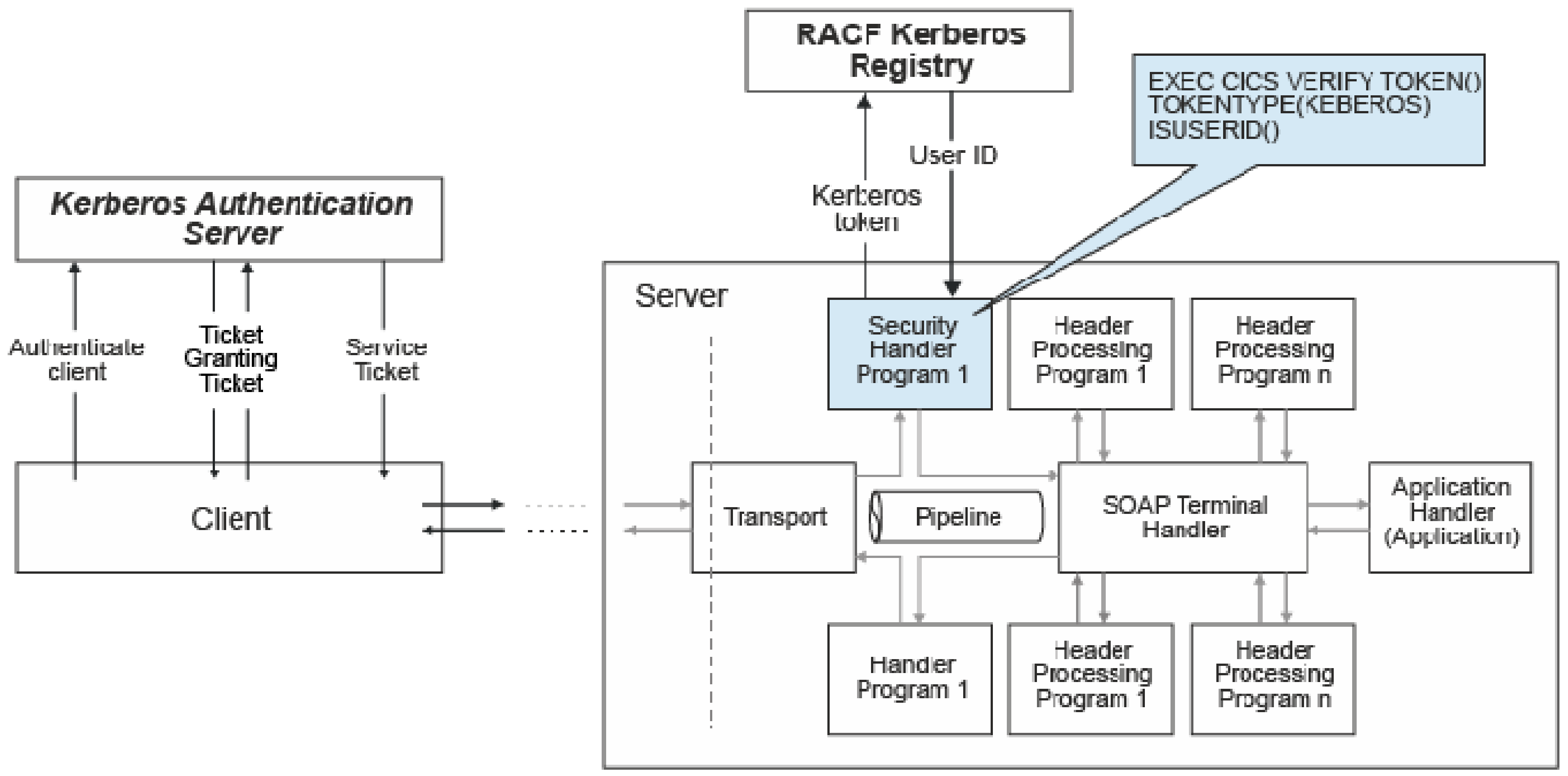
- In deployments where SSL is sufficient to satisfy the integrity and confidentiality requirements, then a simple Header processing program can be used to extract a User ID from an inbound message

- See Implementing Web Services in CICS
  - Redbook SG24-7206

## CICS Support for Kerberos

- WS-Security authentication
  - Provider mode
  - Asserted identity
  - Token validation
  - User ID extracted
- API command
  - Validate Kerberos token
  - Extract User ID
- IBM Network Authorization Service for z/OS
  - Implemented by External Security Manager

## Kerberos and CICS


RACF Kerberos Registry
EXEC CICS VERIFY TOKEN() TOKENTYPE(KEBEROS) ISUSERID()
User ID
Kerberos token
Kerberos Authentication Server
Authenticate client
Ticket Granting Ticket
Service Ticket
Client
Server
Security Handler Program 1
Header Processing Program 1
Header Processing Program n
Transport
Pipeline
SOAP Terminal Handler
Application Handler (Application)
Handler Program 1
Header Processing Program 1
Header Processing Program n

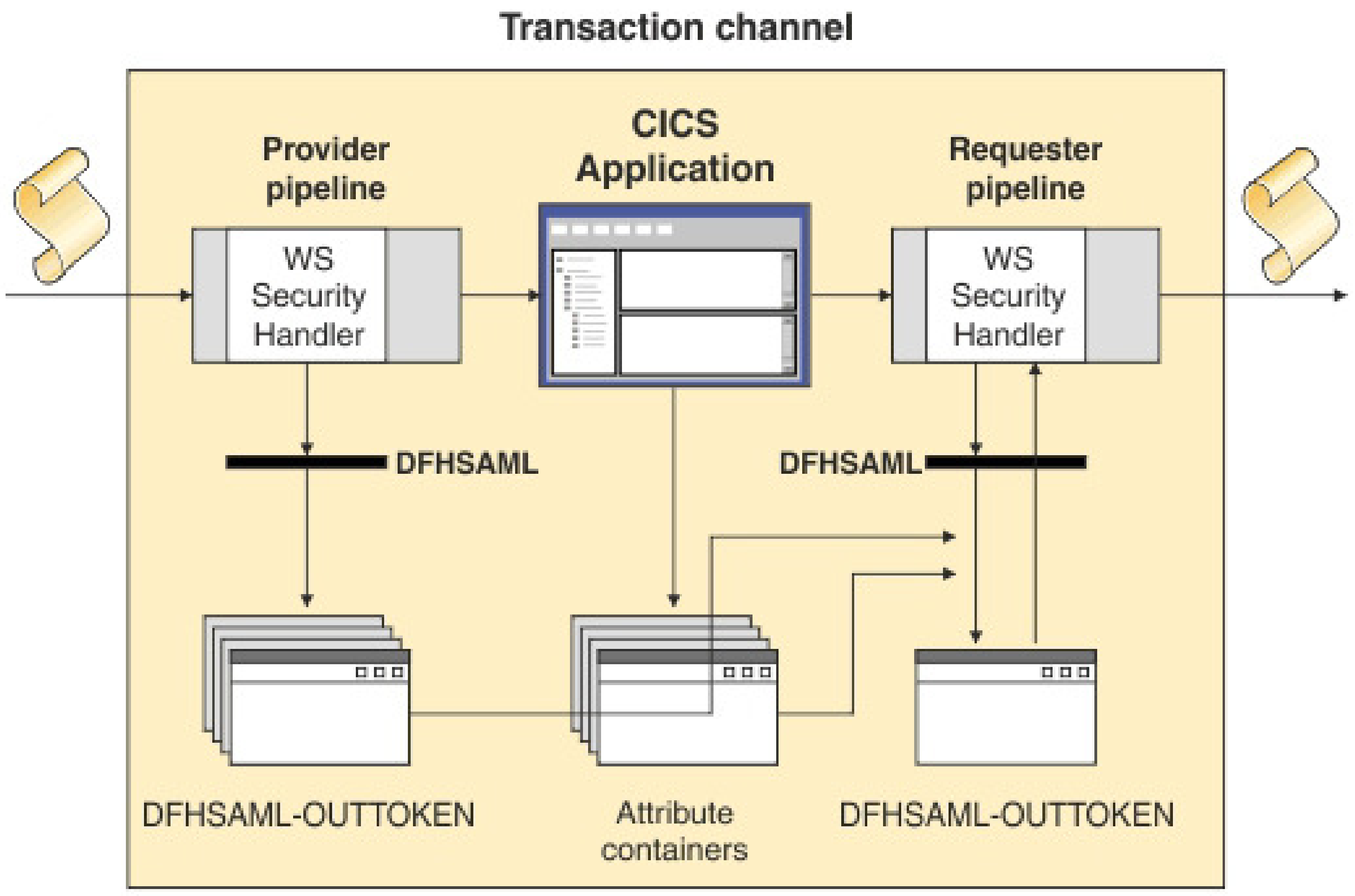
## Verify Token

```
>>-VERIFY TOKEN(data-area)--TOKENLEN(data-value)---------------->

>--+---------------------------+-------------------------------->
   '-+-TOKENTYPE--(--cvda--)-+-'
     '-KERBEROS--------------'

                                  .-BIT------------------.
>--+---------------------------+--+----------------------+------>
   '-ISUSERID--(--data-area--)-'  +-DATATYPE--(--cvda--)-+
                                  '-BASE64---------------'

>--+----------------------+--+--------------------+------------><
   '-ESMREASON(data-area)-'  '-ESMRESP(data-area)-'
```

## CICS Support for SAML

- SAMLCore1.1 and SAMLCore2.0
  - No support for protocols
- WS-Security authentication
  - Token validation
  - Extraction of SAML parts for inbound messages
  - Addition of SAML token to SOAP request
  - Augmentation of SAML token before it is added to outbound message
- API
  - Linkable interface: DFHSAML
  - Channel and containers
  - Create tokens
  - Validate tokens
  - Extract SAML parts
  - Augment SAML assertions

## SAML processing in CICS



## Conclusions

- Security is many things
  - Authentication
    - Identification Credentials
    - Identity Propagation
    - Identity Assertion
  - Authorization
  - Confidentiality
    - Encryption
  - Integrity
    - Digital Signatures
    - Non-repudiation
- Open standards assist with security interoperability

## Further Resources

- WS-Security specifications
  - http://www.oasis-open.org/specs/index.php#wssv1.0
- WS-I Basic Security Profile V1.0
  - http://www.ws-i.org/Profiles/BasicSecurityProfile-1.0.html
- IBM Redbooks
  - Implementing CICS Web Services, SG24-7206
  - Patterns: Extended Enterprise SOA and Web Services, SG24-7135
- WS-Trust
  - February 2005, http://www.ibm.com/developerworks/library/specification/ws-trust/

## Interesting reading

- Securing Web Services
  - http://www.techweb.com/wire/security/20020508_security
- Best Practices for Web services: Web services security
  - http://www-128.ibm.com/developerworks/webservices/library/ws-best11/
  - http://www-128.ibm.com/developerworks/webservices/library/ws-best12/
- Web Service Security: Scenarios, Patterns, and Implementation Guidance
  - http://www.gotdotnet.com/codegallery/codegallery.aspx?id=67f659f6-9457-4860-80ff-0535dffed5e6
- Solving the web services identity crisis
  - http://www.looselycoupled.com/stories/2004/crisis-id0622.html
- Mainframe security changes as Web services arrive
  - http://searchwebservices.techtarget.com/tip/0,289483,sid26_gci1202408,00.html?asrc=SS_CLA_301932&psrc=CLT_26

## FYI

- Asymmetric key algorithms (Public-key cryptography)
  - RSA
    - encryption algorithm patented by Rivest, Shamir, and Adleman
    - patent expired in 2000
  - DSA
    - Digital Signature Algorithm, U.S. Government standard
  - DES
    - Data Encryption Standard, U.S. Government standard
  - AES
    - Advanced Encryption Standard, U.S. Government standard
- Cryptographic hash functions
  - SHA-1
    - Secure Hash Algorithm, developed by NSA, U.S. Government standard
    - used in many security applications and protocols
      - TLS, SSL, PGP, SSH, S/MIME, and IPSec
  - MD5
    - Message-Digest algorithm 5