

CICS Web services security scenarios

Nigel Williams (IBM Design Center)

nigel_williams@uk.ibm.com

Session 4136B

Agenda

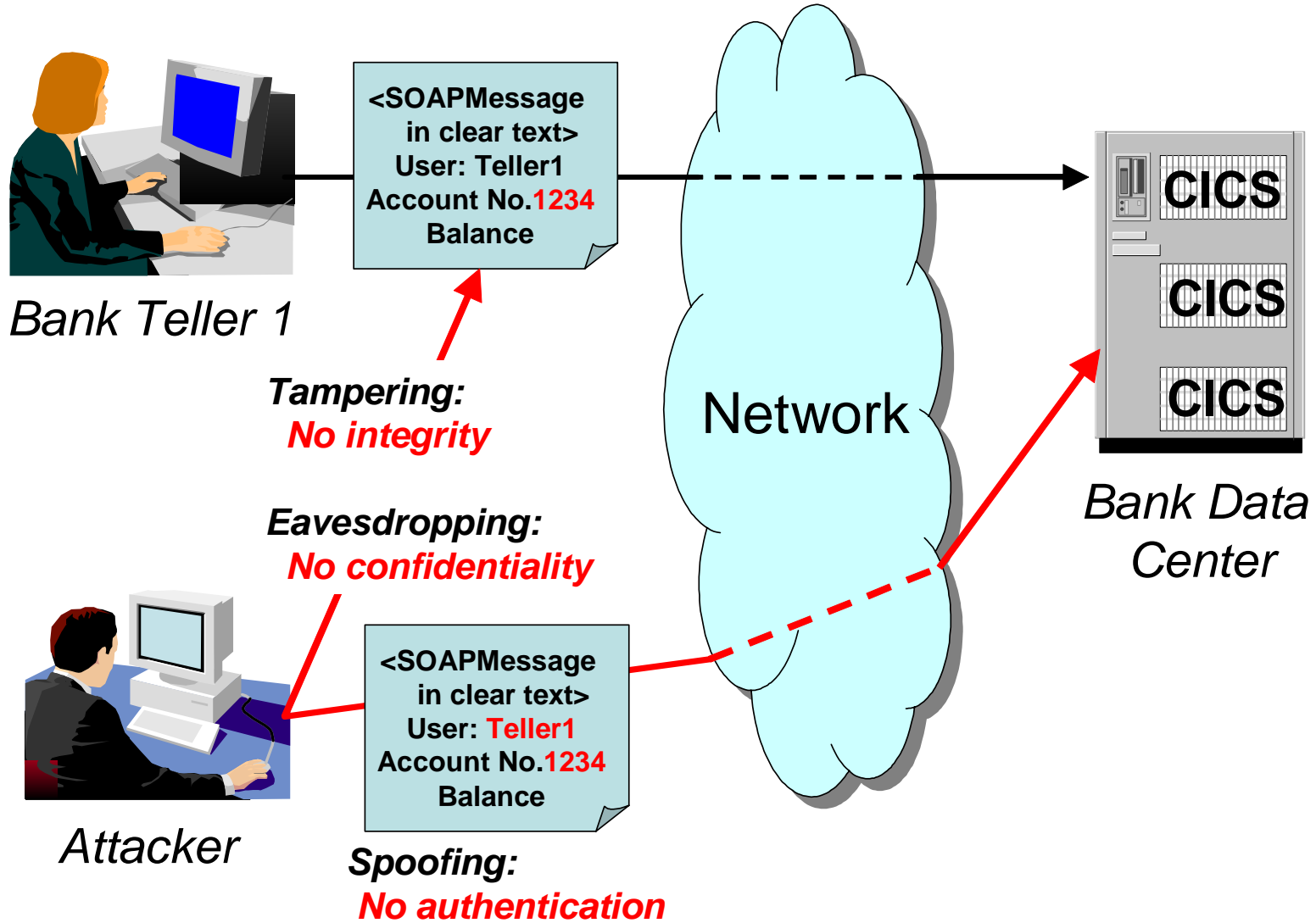
- An overview of some of the technologies that can be used when securing CICS Web Services:
 - [Security challenges](#)
 - [Transport level security](#)
 - [Message level security](#)

- Detailed look at some WS-Security scenarios:
 1. [Setting user ID on URIMAP definition](#)
 2. [Basic authentication](#)
 3. [Identity Assertion](#)
 4. [XML Digital Signatures](#)
 5. [XML Encryption](#)

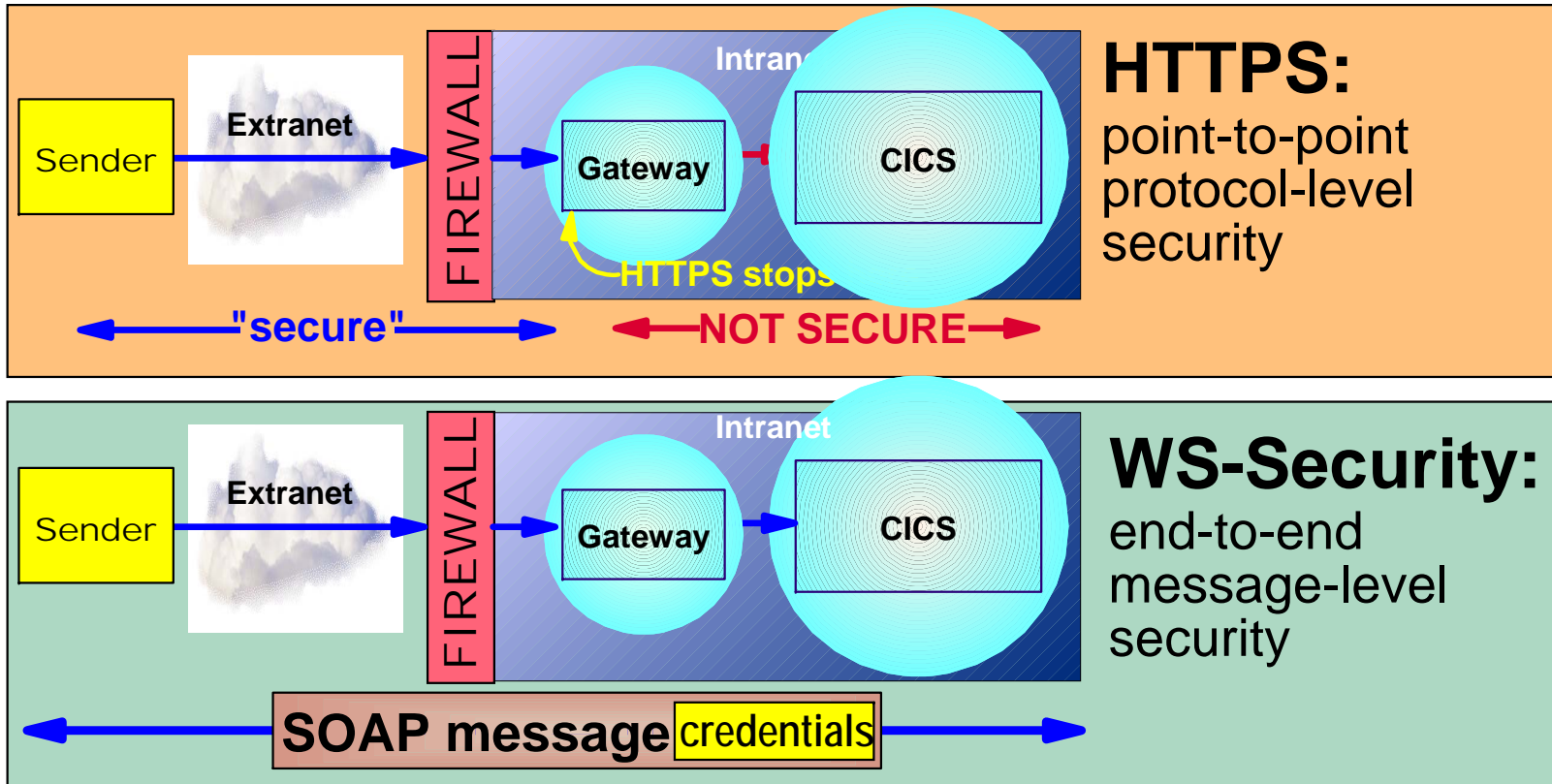
Thanks to the following people for their input to this presentation: the redbook team Robert Herman and Luis Aused Lopez. And also Fraser Bohm, Arnauld Desprets and Patrick Kappeler.

Technology Overview

Security challenges



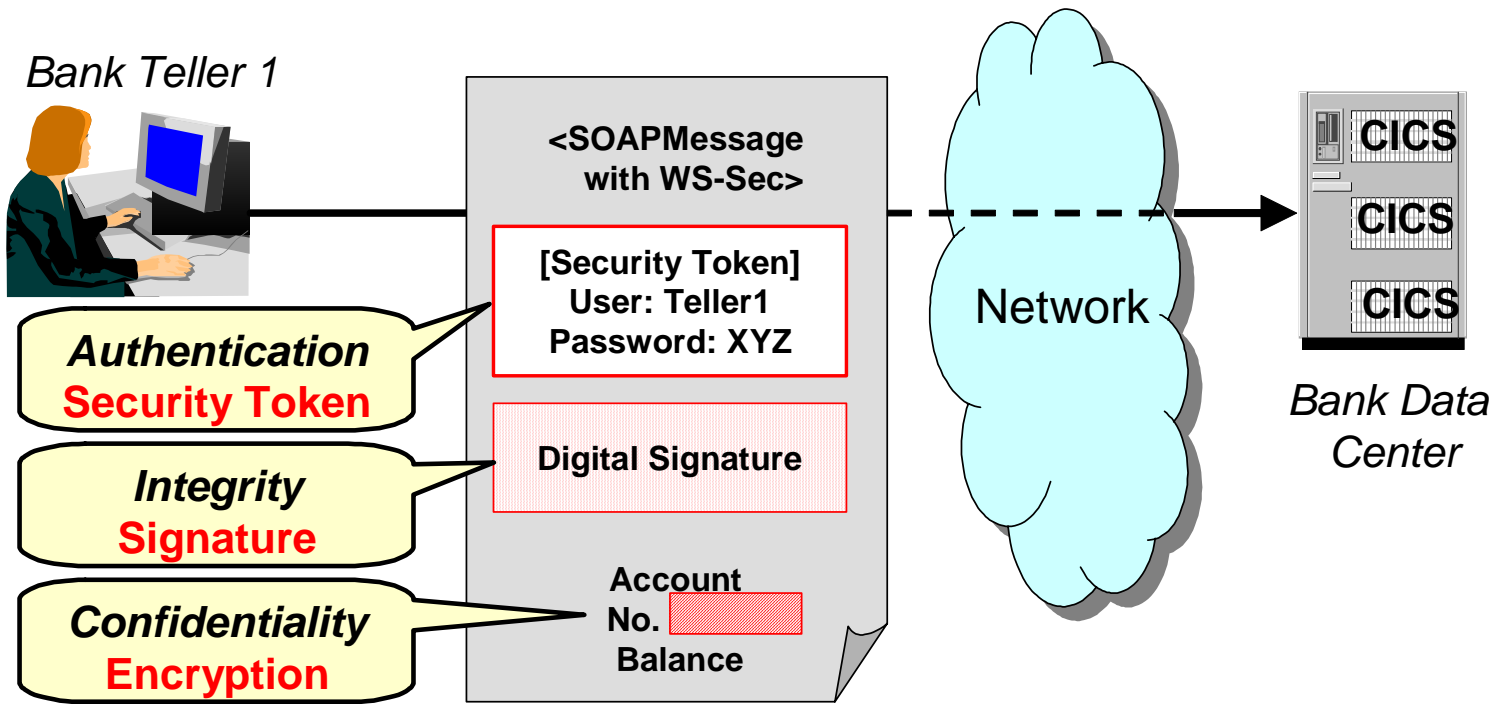
Transport security or message level security ?



Without WS-Security, CICS Web Services can be secured using any technique that can be applied to a browser based application, such as HTTPS

However, this still leaves a number of gaps:- end-to-end (rather than point-to-point) credentials, signature, encryption, etc.

Message level security



The SOAP message shown above contains three pieces of security data:

- A **security token** (UsernameToken) used to authenticate and identify user Teller1
- A **digital signature** to ensure that no one illegally modifies message while it is in transit.
- An account balance XML element which is **encrypted** to ensure confidentiality

CICS provided support for WS-Security

Support for the following is provided by the CICS WS-Security Message handler, DFHWSSE1: Shipped via **APAR PK22736 (PTFs UK15271 and UK15261)**

- **Authentication** based on basic authentication (UsernameToken) or X.509 certificate (BinarySecurityToken)
- **Identity Assertion** based on trust tokens (basic authentication or X.509 certificate)
- **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, with key wrap RSA 1_5 and AES 128,192& 256 or triple-DES.
- **Encryption** of the SOAP Body content with the above algorithms.

User-written SOAP header processing program

A user-written **message handler program** can:

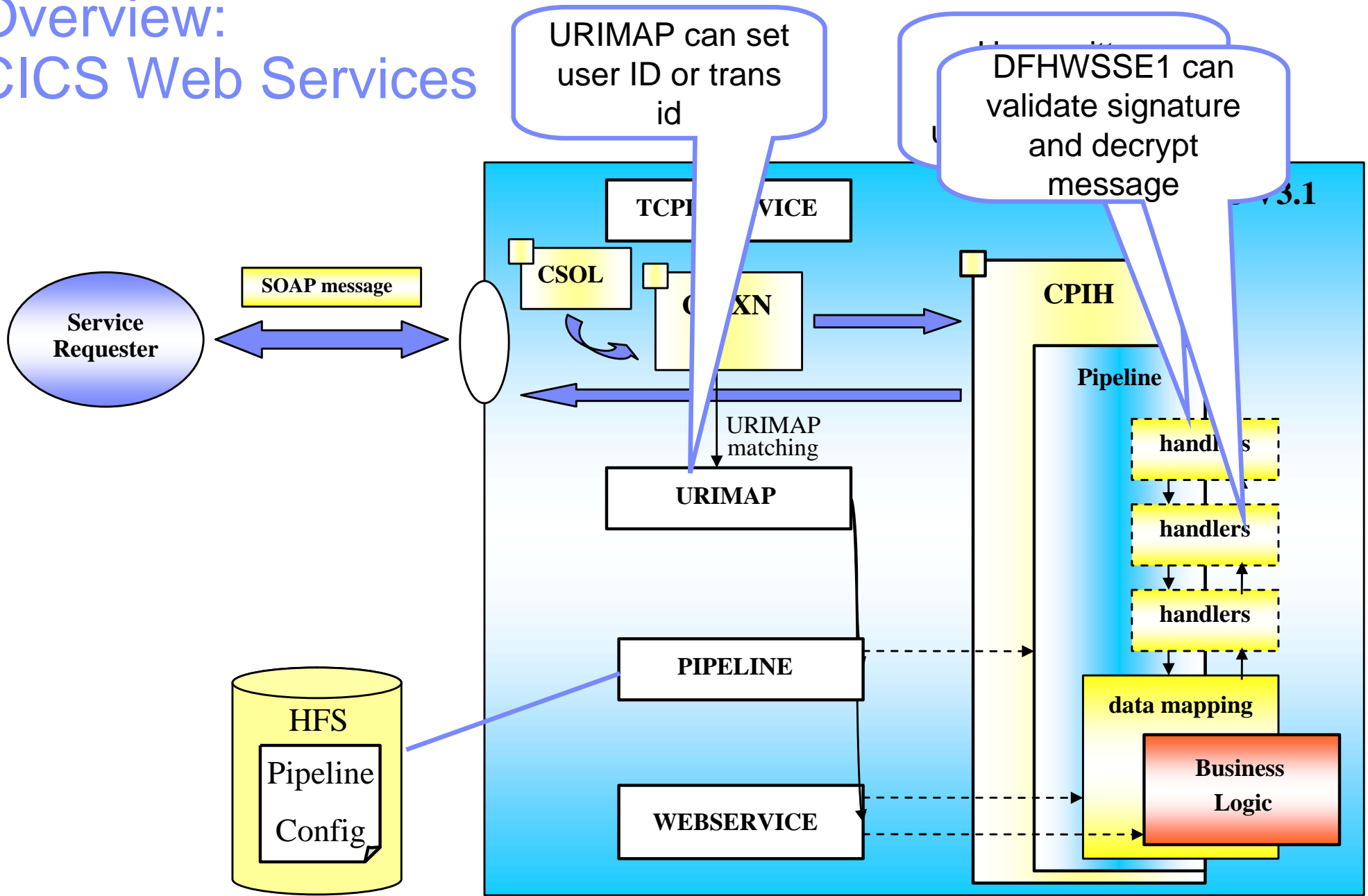
- extract the UsernameToken from the SOAP header
- validate the username and password
- set the user ID of the CICS task to the username passed in the header

SSL can be used for integrity and confidentiality

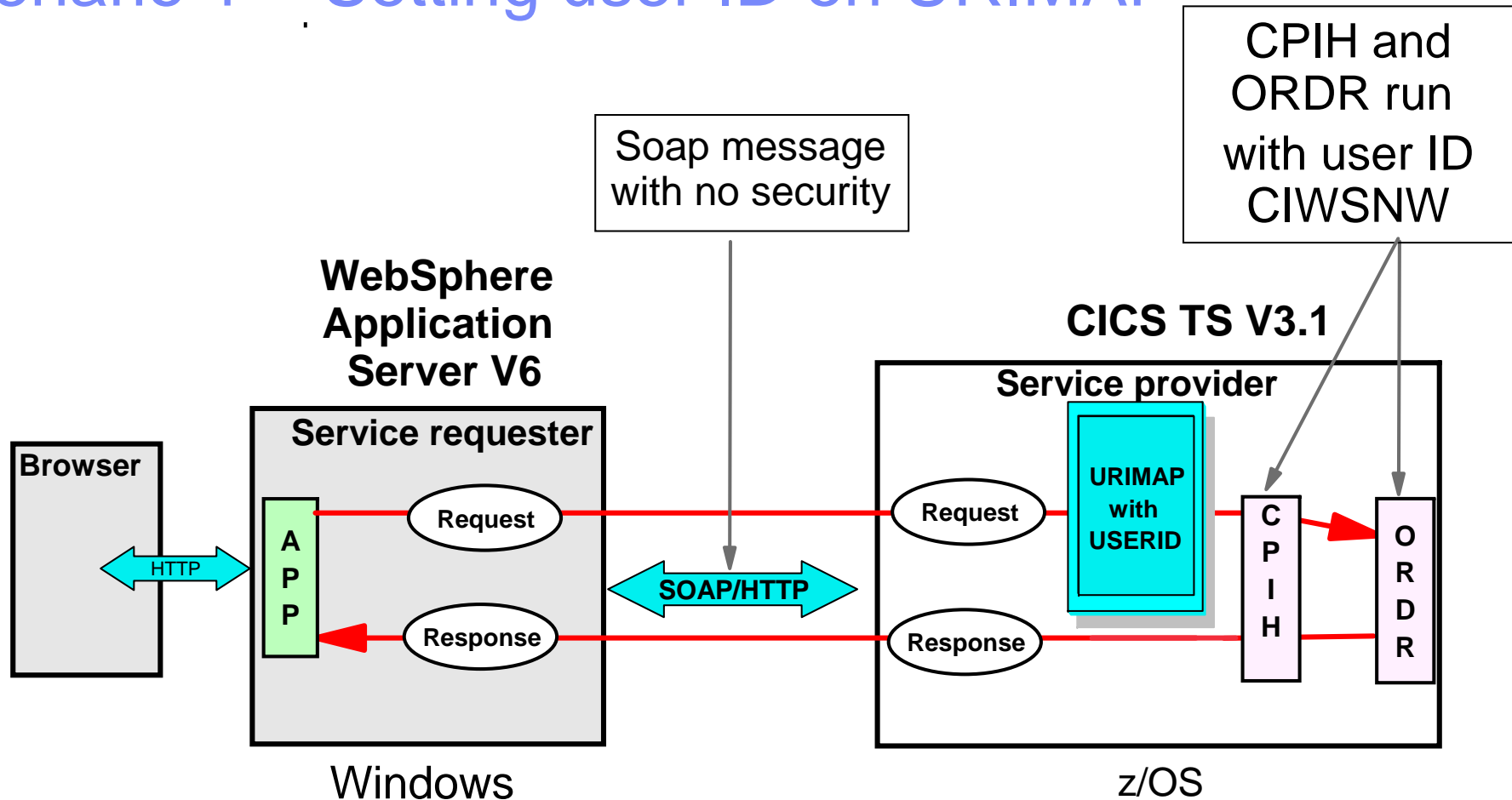
This '**hand-crafted**' approach is very appropriate for applications with 'simple' security requirements and has significant performance benefits.

Security scenarios

Overview: CICS Web Services



Scenario 1 – Setting user ID on URIMAP



- User ID (**CIWSNW**) for transaction is specified in URIMAP definition
- Transaction id **ORDR** is dynamically chosen for pipeline alias transaction based on type of request (default trans id is CPIH)
- All placeOrder service requests run with same user ID

URIMAP definition

CEDA DEFine Urimap(SECPORDR)

Urimap : SECPORDR

Group : S3C1

Description ==>

Status ==> Enabled Enabled | Disabled

USAge ==> Pipeline Server | Client | Pipeline

UNIVERSAL RESOURCE IDENTIFIER

SCHEME ==> HTTP HTTP | HTTPS

HOST ==> *

(Lower Case) ==>

PAth ==> **/exampleApp/placeOrder**

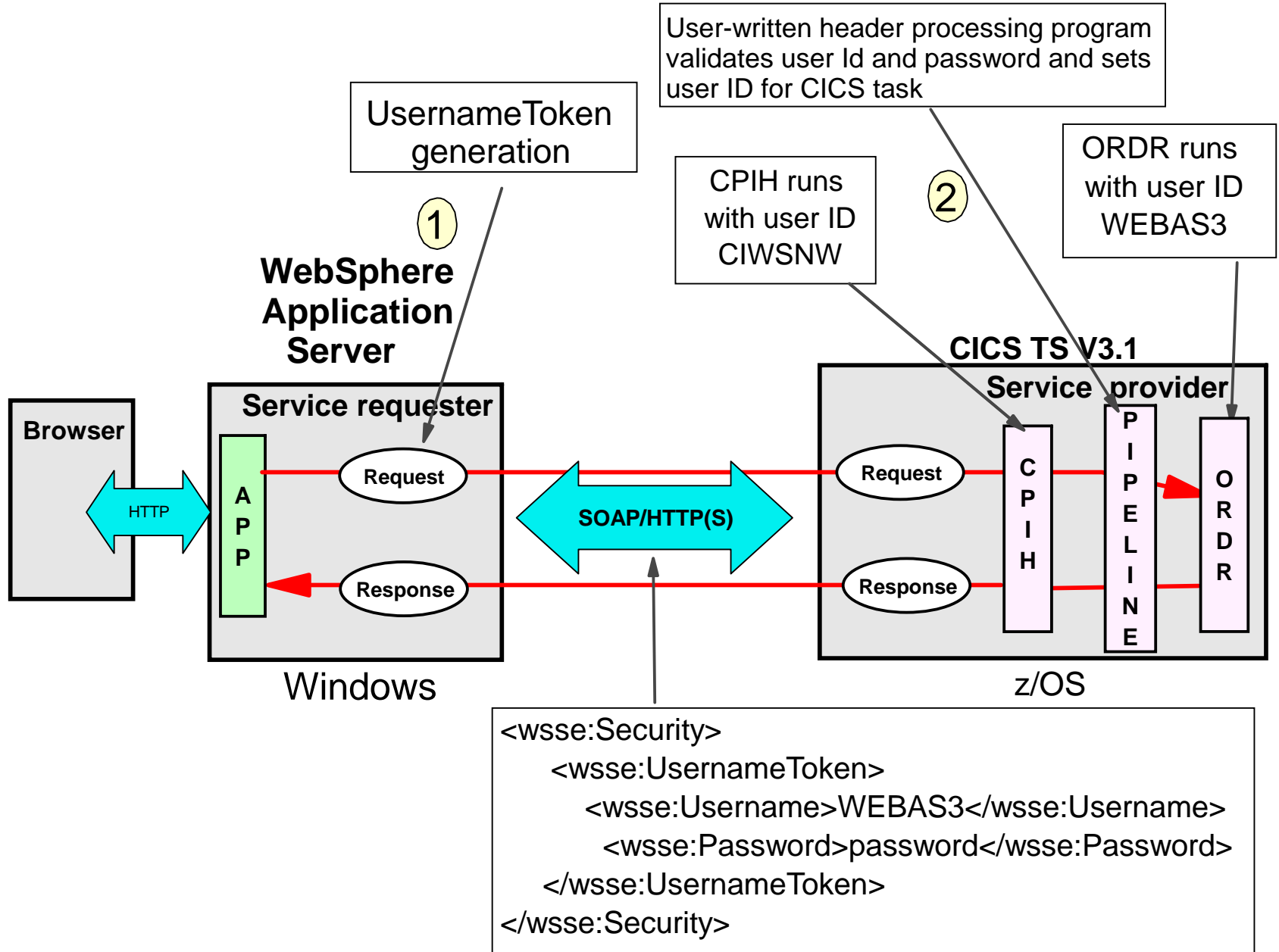
ASSOCIATED CICS RESOURCES

Webservice ==> **placeOrder**

SECURITY ATTRIBUTES

USERid ==> **CIWSNW**

Scenario 2 – Basic Authentication



Basic authentication configuration

WebSphere

- Configure the request generator
 - Include UsernameToken in SOAP message request

CICS

- Pipeline configuration file
 - Run user-written header processing program which validates user Id and password and sets user ID for CICS task

Web Deployment Descriptor
✕

Web Service Client Security Extensions

Editor for Web Service Client Security Extensions

▶ **Component Scoped References**

▼ **Service References**

Service References

- service/DFHOXCMNService
- service/DFHOXCMNService2
- service/DFHOXCMNService3

Add
Edit
Remove

▼ **Port QName Bindings**

The following are port qualified name bindings of the selected service reference

- DFHOXCMNPort

Add
Edit
Remove

▶ **Client Service Configuration Details**

▶ **Default Mappings**

▼ **Request Generator Configuration**

The following is the Request Generator Configuration for the selected Port QName Binding

- ▶ **Details**
- ▶ **Integrity**
- ▶ **Confidentiality**
- Security Token**
- ▶ **Add Timestamp**
- ▶ **Property**

▼ **Response Consumer Configuration**

The following is the Response Consumer Configuration for the selected Port QName Binding

- ▶ **Required Integrity**
- ▶ **Required Confidentiality**
- ▶ **Required Security Token**
- ▶ **Add Timestamp**
- ▶ **Property**

Client Deployment Descriptor

Request Generator Configuration

Response Consumer Configuration

Client Binding configuration

Request Generator Configuration

Response Consumer Configuration

Overview | Servlets | Filter | Security | References | W5 Handler | Pages | Variables | **W5 Extension** | W5 Binding | Extensions | Source

*Web Deployment Descriptor

Editor for Web services client bindings

▶ **Component scoped references**

▼ **Service references**

The following are Web services referenced in this client binding

- service/DFHOXCMNService
- service/DFHOXCMNService2
- service/DFHOXCMNService3**

Add Remove

▶ **Service References Details**

▼ **Port Qualified Name Binding**

Port qualified name bindings of the selected service reference

- DFHOXCMNPort**

Add Edit

Client Deployment Descriptor

- Request Generator Configuration
- Response Consumer Configuration

Client Binding configuration

- Request Generator Configuration
- Response Consumer Configuration

▶ **Security Request Generator Binding Configuration**

Security configuration for generating request messages

- ▶ Trust Anchor
- ▶ Certificate Store List
- ▶ Token Generator
- ▶ Key Locators
- ▶ Key Information
- ▶ Signing Information
- ▶ Encryption Information
- ▶ Property

▼ **Security Response Consumer Binding Configuration**

Security configuration for consuming response messages

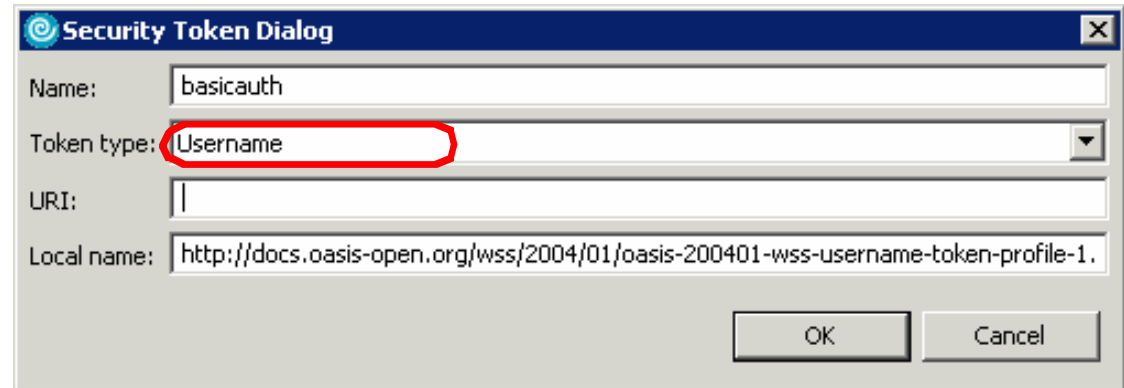
- ▶ Trust Anchor
- ▶ Certificate Store List
- ▶ Token Consumer
- ▶ Key Locators
- ▶ Key Information
- ▶ Signing Information

Overview | Servlets | Filter | Security | References | WS Handler | Pages | Variables | WS Extension | WS Binding | Extensions | Source

Configure service requester for basicauth

Client deployment descriptor:

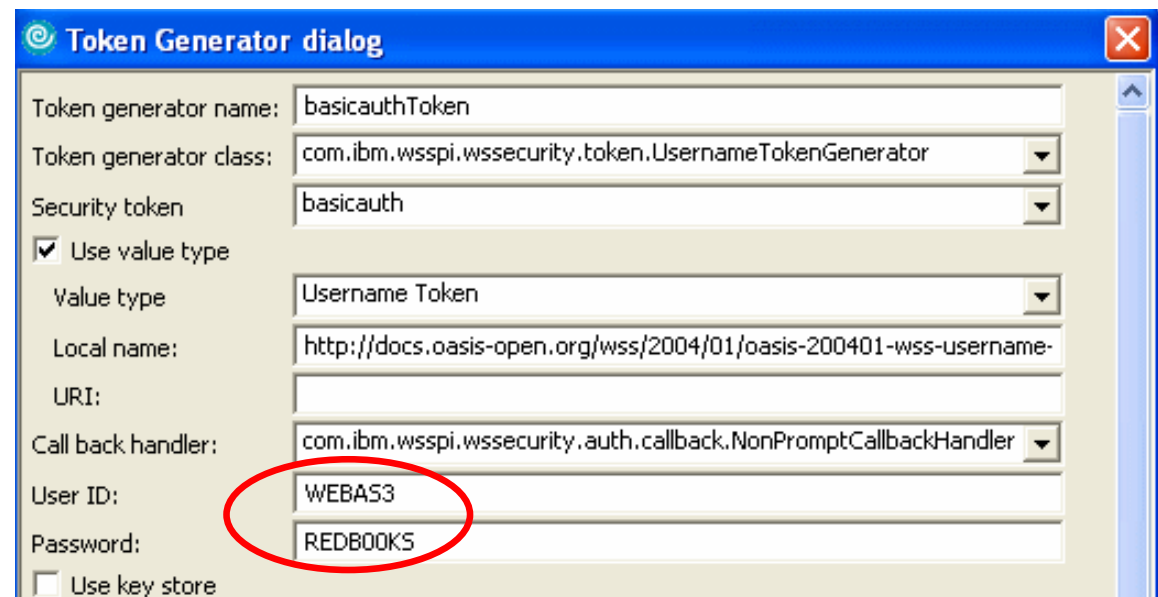
Specify the **type** of security token



The Security Token Dialog window shows the configuration for a security token. The Name field is set to 'basicauth'. The Token type dropdown is set to 'Username' and is highlighted with a red circle. The Local name field contains the URI 'http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.'. The URI field is empty. The OK and Cancel buttons are visible at the bottom right.

Client binding configuration:

Specify the security token **information**



The Token Generator dialog window shows the configuration for a token generator. The Token generator name is 'basicauthToken'. The Token generator class is 'com.ibm.wsspi.wssecurity.token.UsernameTokenGenerator'. The Security token is 'basicauth'. The 'Use value type' checkbox is checked. The Value type dropdown is set to 'Username Token'. The Local name field contains the URI 'http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-'. The URI field is empty. The Call back handler is 'com.ibm.wsspi.wssecurity.auth.callback.NonPromptCallbackHandler'. The User ID field is 'WEBAS3' and the Password field is 'REDBOOKS', both highlighted with a red circle. The 'Use key store' checkbox is unchecked.

CICS pipeline configuration for basicauth

```
<service>
  <terminal_handler>
    <cics_soap_1.2_handler>
      <headerprogram>
        <program_name>CIWSSECH</program_name>
        <namespace>http://docs.oasis-open.org/wss/2004/01/oasis-
          200401-wss-wssecurity-secext-1.0.xsd</namespace>
        <localname>Security</localname>
        <mandatory>true</mandatory>
      </headerprogram>
    </cics_soap_1.2_handler>
  </terminal_handler>
</service>
```

Header processing program CIWSSECH

Pseudo code

Check if invoked for RECEIVE-REQUEST, else exit

Check for correct URI, else exit

Obtain WS-Security header from DFHHEADER container

Parse the header

Verify the credentials extracted

Put user ID into DFHWS-USERID container

What user ID is used to run Web service request ?

INQUIRE TASK

STATUS: RESULTS - OVERTYPE TO MODIFY

Tas(0000274) Tra(CPIH) Sus Tas Pri(001)

Sta(U) Use(CIWSNW) Uow(BDFAD304E2ED1740) Hty(RZCBNOTI)

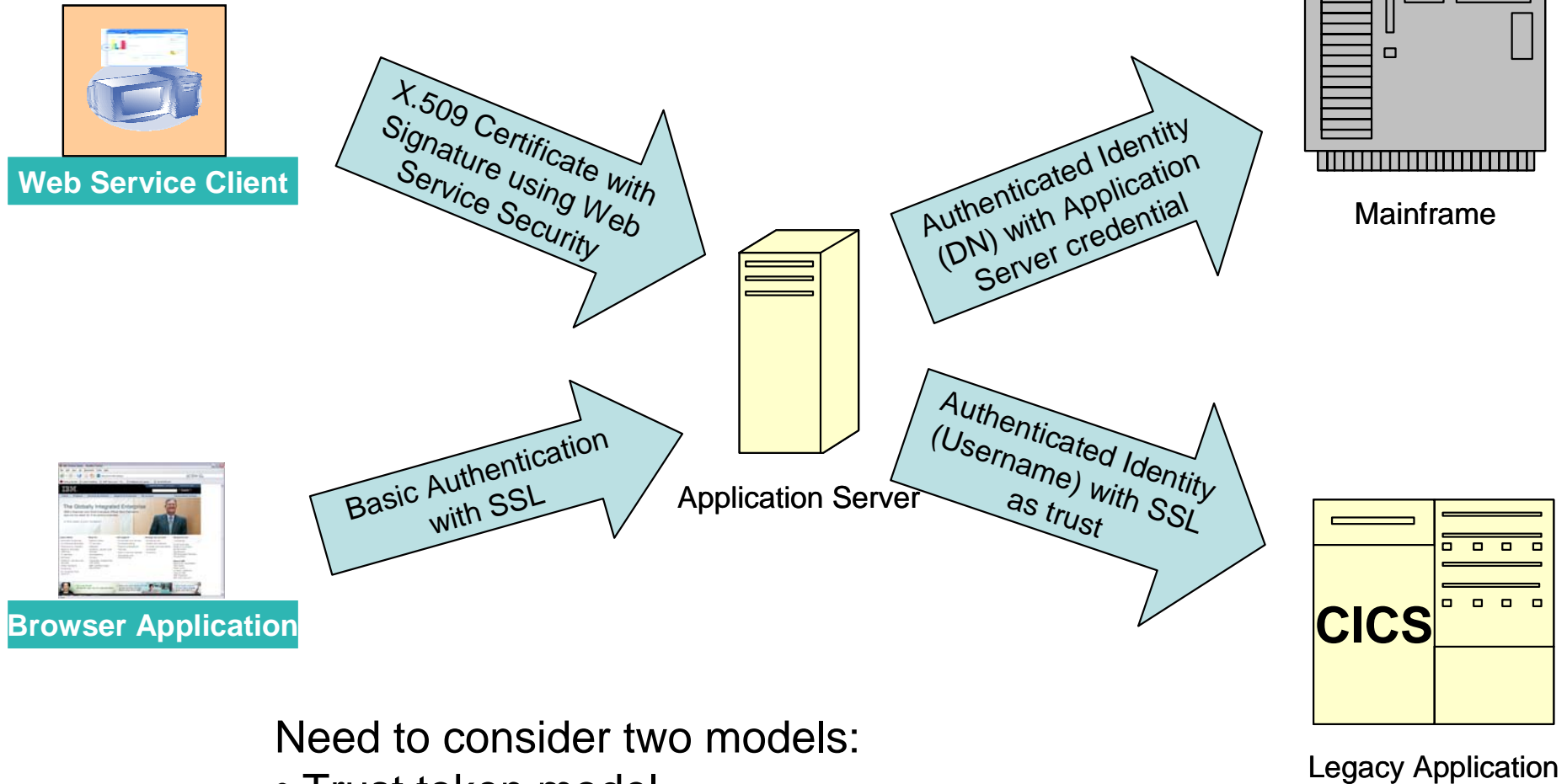
Tas(0000275) Tra(ORDR) Sus Tas Pri(001)

Sta(U) Use(**WEBAS3**) Uow(BDFAD304E4DEE18E) Hty(EDF)

SYSID=S3C1 PPLID=A6POS3C1

ORDR transaction runs with user ID **WEBAS3**, that is the user ID passed in the SOAP Security header.

Scenario 3 - What is Identity assertion ?



Need to consider two models:

- Trust token model
- Blind trust model

CICS provided support for ID assertion

- Requirements specified using **<authentication>** element of DFHWSSE1 configuration in pipeline configuration file e.g

```
<service_handler_list>  
  <wsse_handler>  
    <dfhwsse_configuration version="1">  
      <authentication trust="signature" mode=" basic">  
        </authentication>  
      </dfhwsse_configuration>  
    </wsse_handler>  
  </service_handler_list>
```

- CICS processes two tokens
 - **Identity token** (the asserted identity)
 - **Trust token** (used to check that the sender is authorized to assert identities)
- Trust relationship is established using surrogate definitions

Trust options for service provider pipeline

trust	mode	meaning
basic	basic	<ul style="list-style-type: none"> ● The trust token is a username token with a password ● The identity token is a second username token without a password. CICS puts this username in container DFHWS-USERID, and this user ID is used to run transactions in CICS.
	signature	<ul style="list-style-type: none"> ● The trust token is a username token with a password ● The identity token is an X.509 certificate. CICS puts the user ID associated with the certificate in container DFHWS-USERID, and this user ID is used to run transactions in CICS.
signature	basic	<ul style="list-style-type: none"> ● The trust token is an X.509 certificate. ● The identity token is a username token without a password. <p>The identity token and the body must be signed with the X.509 certificate.</p>
	signature	<ul style="list-style-type: none"> ● The trust token is an X.509 certificate. ● The identity token is a second X.509 certificate. <p>The identity token and the body must be signed with the 1st X.509 certificate (the trust token).</p>

Trust options for service requester pipeline

trust	mode	meaning
signature	basic	<p>CICS adds the following tokens to the message:</p> <ul style="list-style-type: none">• The trust token is an X.509 security token.• The identity token is a username with no password. <p>The certificate used to sign the identity token and message body is specified by the <certificate_label>.</p> <p>The user ID placed in the identity token is the contents of the DFHWS-USERID container (which, by default, contains the running task's user ID).</p>

Comparison of trust token and blind trust models

Trust token model

- WAS sends trust token and identity token for each request
- Trust token options are usernametoken or X.509 certificate
- CICS validates trust token for each request

Advantages

- CICS supplied function
- Works for any transport mechanism

Disadvantages

- Trust re-established for each request
- Performance

Blind trust model

- WAS only sends identity token
- Trust needs to be established using a transport based mechanism (e.g SSL client authentication)

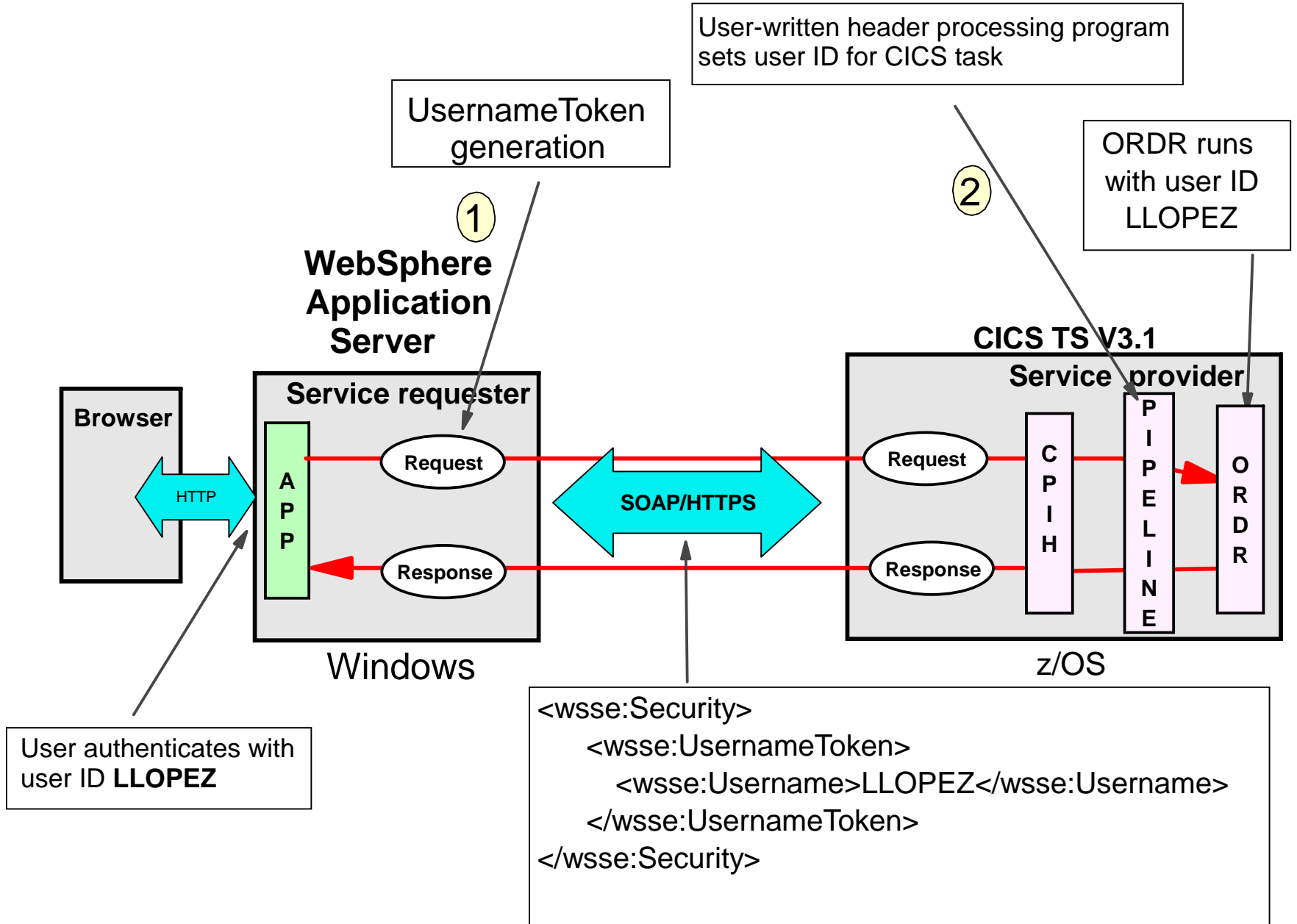
Advantages

- Trust can be established at socket connection and is not required for each request (if connection persists)
- Performance

Disadvantages

- Trust model dependent on transport mechanism
- User-written approach because DFHWSSE1 does not support blind trust

Scenario 3 – Identity Assertion



ID assertion configuration

WebSphere

- Configure the request generator
 - Include UsernameToken in SOAP message request based on Run As identity

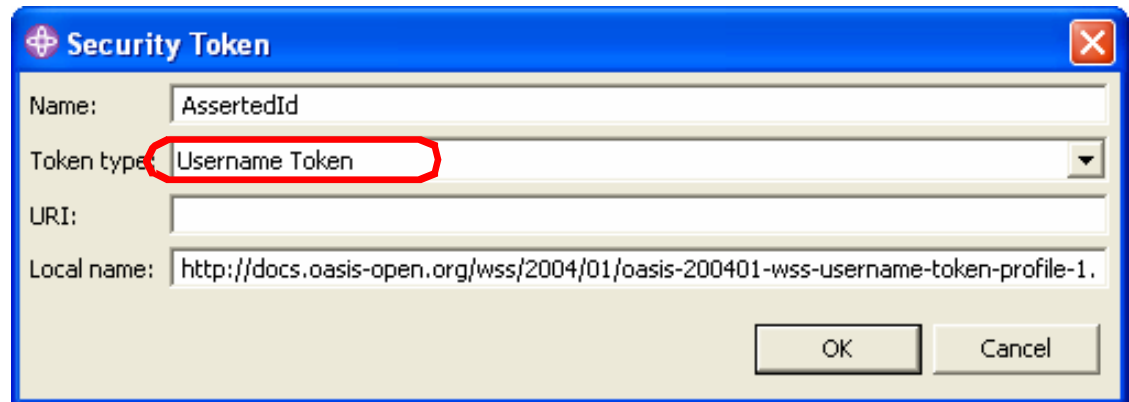
CICS

- Pipeline configuration file
 - Run user-written header processing program which sets user ID for CICS task

Configure service requester for ID assertion

Client deployment descriptor:

Specify the **type** of security token



The screenshot shows a dialog box titled "Security Token" with a blue header bar. It contains the following fields:

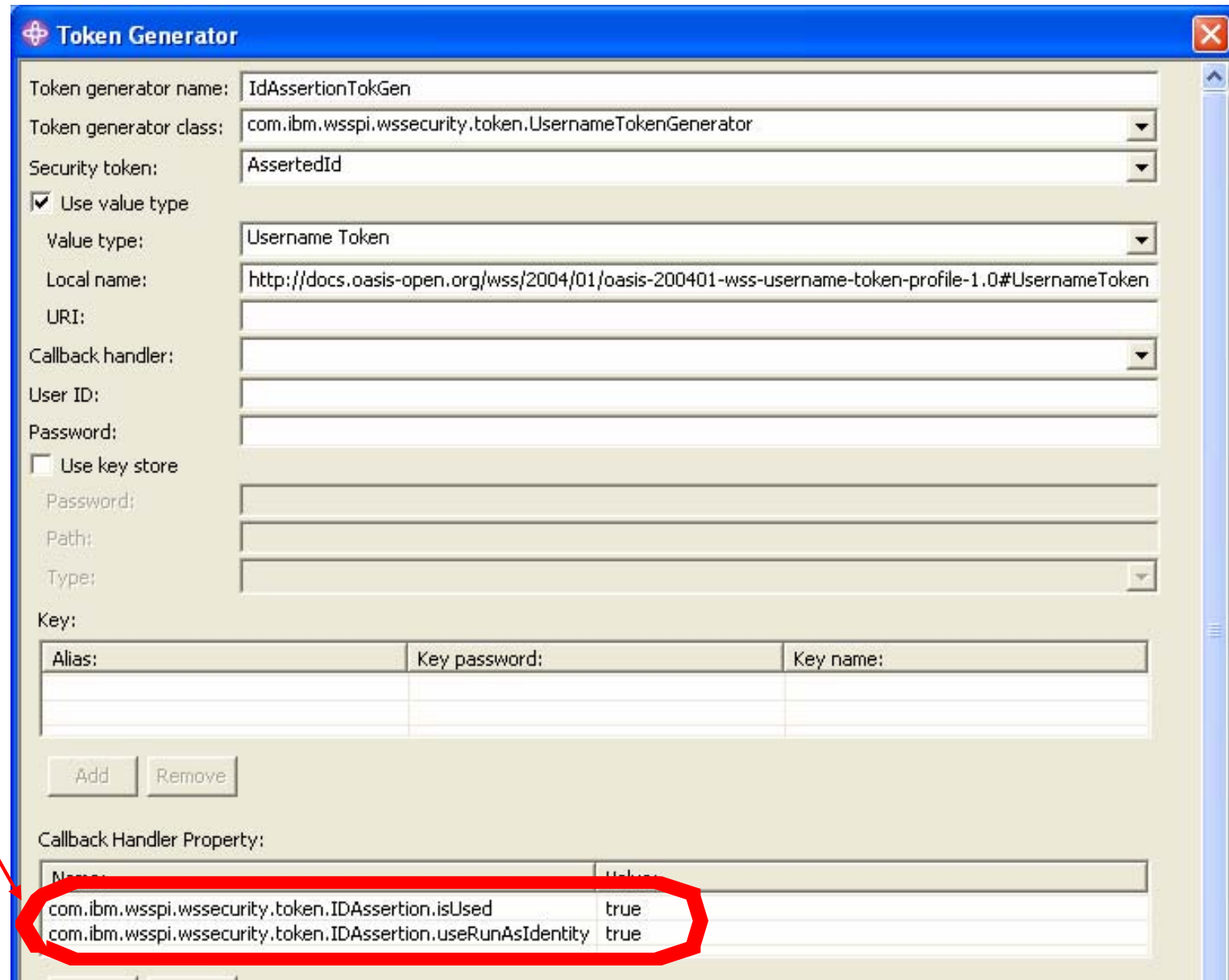
- Name: AssertedId
- Token type: Username Token (highlighted with a red circle)
- URI: (empty)
- Local name: http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.

At the bottom right, there are "OK" and "Cancel" buttons.

Configure service requester for ID assertion

Client binding configuration:

Specify the security token generation information for Identity assertion



Token generator name: IdAssertionTokGen

Token generator class: com.ibm.wsspi.wssecurity.token.UsernameTokenGenerator

Security token: AssertedId

Use value type

Value type: Username Token

Local name: http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#UsernameToken

URI:

Callback handler:

User ID:

Password:

Use key store

Password:

Path:

Type:

Alias:	Key password:	Key name:

Add Remove

Callback Handler Property:

Name:	Value:
com.ibm.wsspi.wssecurity.token.IDAssertion.isUsed	true
com.ibm.wsspi.wssecurity.token.IDAssertion.useRunAsIdentity	true

CICS pipeline configuration for ID assertion

```
<service>
  <terminal_handler>
    <cics_soap_1.2_handler>
      <headerprogram>
        <program_name>CIWSSECS</program_name>
        <namespace>http://docs.oasis-open.org/wss/2004/01/oasis-
          200401-wss-wssecurity-secext-1.0.xsd</namespace>
        <localname>Security</localname>
        <mandatory>true</mandatory>
      </headerprogram>
    </cics_soap_1.2_handler>
  </terminal_handler>
</service>
```

Header processing program CIWSSECS

Pseudo code

Check if invoked for RECEIVE-REQUEST, else exit

Check for correct URI, else exit

Obtain WS-Security header from DFHHEADER container

Parse the header

Put user ID into DFHWS-USERID container

What user ID is used to run Web service request ?

INQUIRE TASK

STATUS: RESULTS - OVERTYPE TO MODIFY

Tas(0000115) Tra(CPIH) Sus Tas Pri(001)

Sta(U) Use(CIWSNW) Uow(BF7020A462194C09) Hty(RZCBNOTI)

Tas(0000116) Tra(ORDR) Sus Tas Pri(001)

Sta(U) Use(**LLOPEZ**) Uow(BF7020A462C6FA09) Hty(EDF)

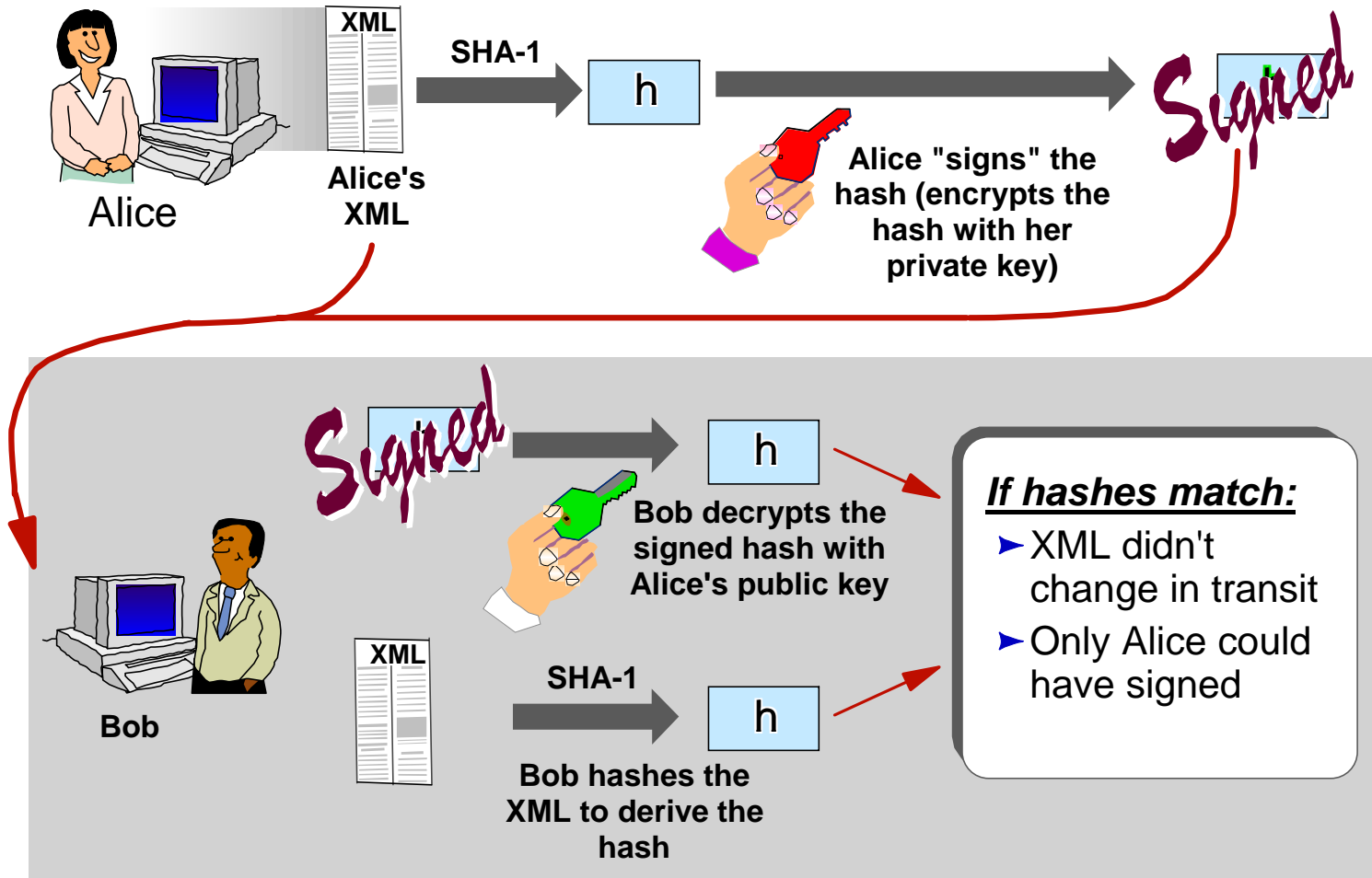
SYSID=S3C1 PPLID=A6POS3C1

ORDR transaction runs with user ID **LLOPEZ**, that is the user ID used to authenticate with the WebSphere Application Server

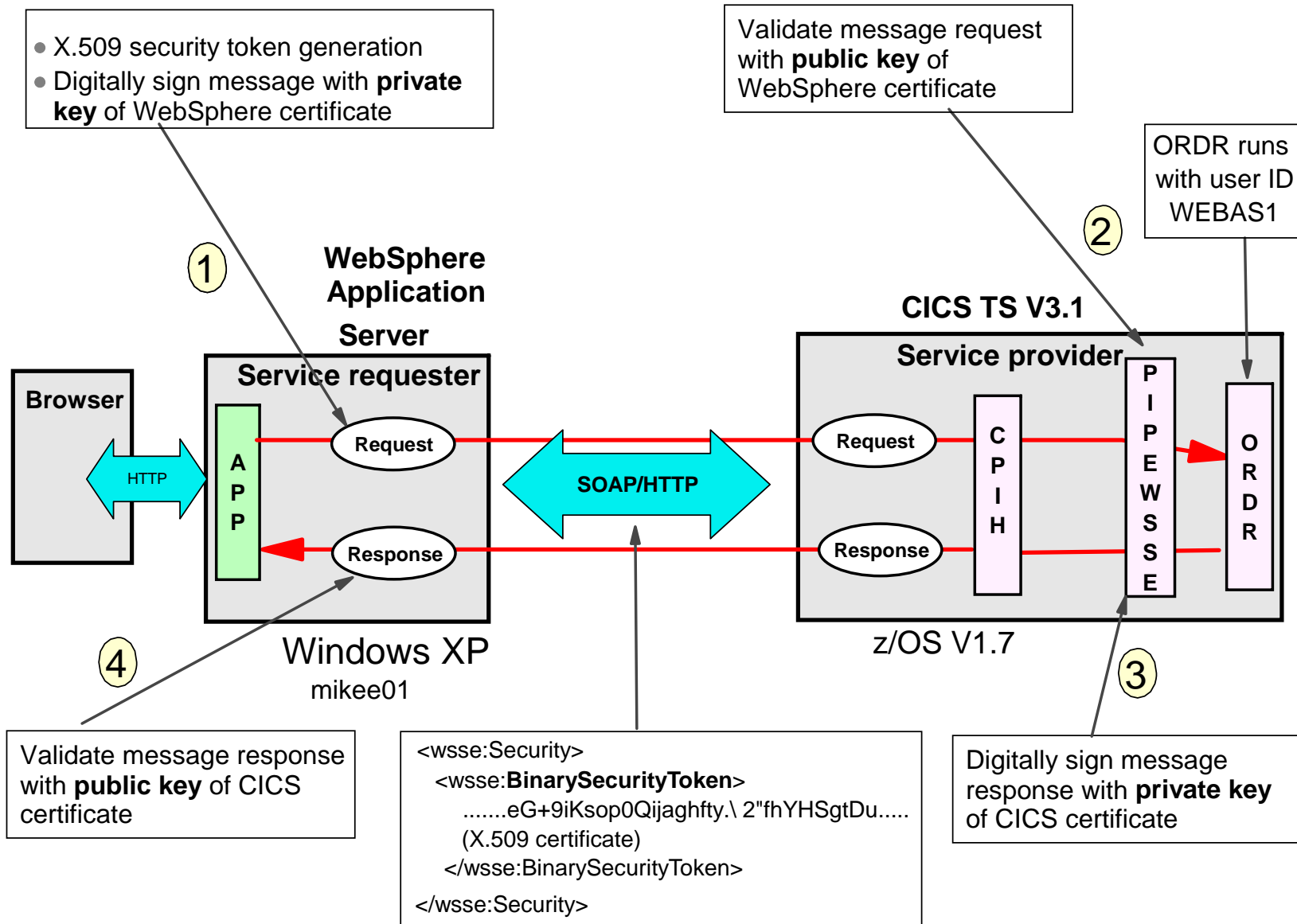
Important:

This simplistic scenario assumes that the distributed and host user registries are synchronised. When this is not the case, a mapping can be done in the service requester, for example, using Tivoli Federated Identity Manager (TFIM).

Scenario 4 – What is an XML Digital Signature ?



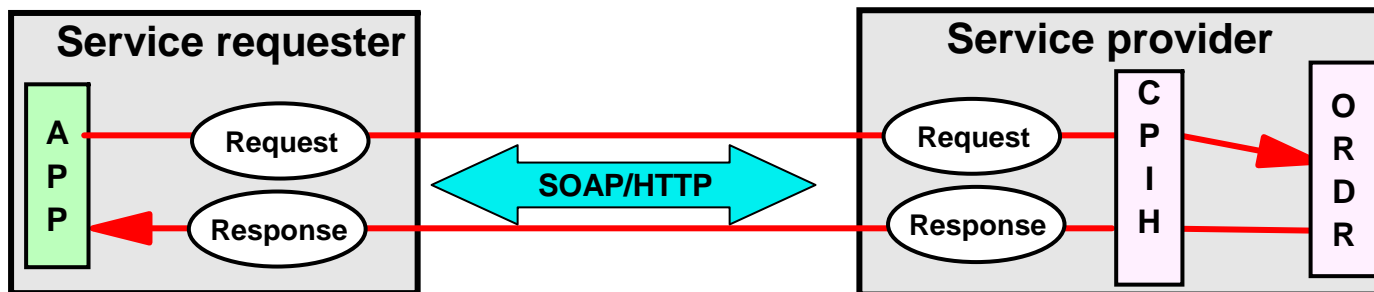
Scenario 4 – Signing a SOAP message



Key pairs and certificates

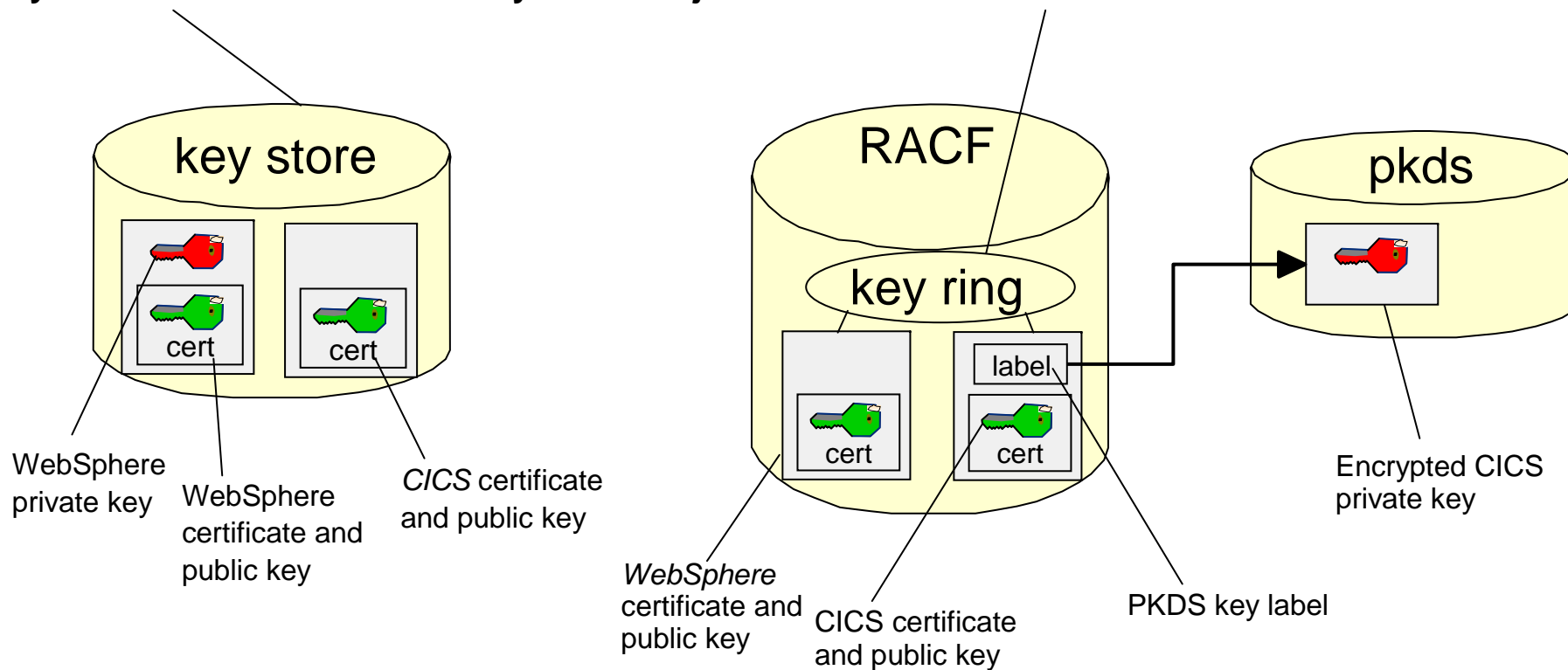
WebSphere Application Server V6.1

CICS TS V3.1



key store name = c:\SG247206\keystore\was.jks

KEYRING=Ciws.Ciwss3c1



Creating CICS certificate in RACF

RACDCERT ID(**CIWS3D**) **GENCERT**

SUBJECTSDN(CN('CICSCERT'))

T ('Ciwss3c1-cert')

OU('PSSC')

O ('ITSO')

L ('Endicott')

SP('New York')

C ('US'))

NOTBEFORE(DATE(2005-01-01) TIME(00:00:00))

NOTAFTER (DATE(2014-12-31) TIME(23:59:59))

KEYUSAGE (**DOCSIGN DATAENCRYPT**)

WITHLABEL('CICSCERT')

SIZE(1024)

ICSF

RACDCERT ID(CIWS3D) **CONNECT**(ID(CIWS3D) LABEL('CICSCERT'))

RING(Ciws.Ciwss3c1))

CICS WS-Security Cryptographic Hardware Requirements

- **ICSF** (Integrated Cryptographic Service Facility) **must** be configured with cryptographic devices and started
 - ICSF is a software component of z/OS which provides an application programming interface which CICS uses to request cryptographic services

- Cryptographic hardware requirements depend on System z server (z9, z990, z890 etc)

- On the z9 an optimal cryptographic hardware configuration is a combination of CPACF and CEX2
 - **CPACF** (CP Assist for Cryptographic Functions) for shared secret key functions
 - **CEX2** (Crypto Express2 Feature) for public key and shared secret key functions

Importing WebSphere certificate into RACF

```
RACDCERT ID(WEBAS1) ADD('CIWS.WASCERT.DER')  
WITHLABEL('WASCERT') TRUST
```

```
RACDCERT ID(CIWS3D) CONNECT(ID(WEBAS1)  
LABEL('WASCERT') RING(Ciws.Ciwss3c1))
```

Digital signature configuration

WebSphere

- Configure the request generator
 - Sign SOAP body of message request
 - Include BinarySecurityToken (X.509 certificate)
- Configure the response consumer
 - Validate signed response

CICS

- Pipeline configuration file
 - Validate signed message
 - Authentication using X.509 certificate
 - Sign SOAP body of message response

Configure service requester for signing

Client deployment descriptor:
 Create integrity message part

1

Integrity

Integrity name: int_body|

Order: 1

Message Parts:

Parts Dialect	Parts Keyword
http://www.ibm.com/websphere/webservices...	body

Token Generator

Token generator name: gen_dsigtgen

Token generator class: com.ibm.wsspi.wssecurity.token.X509TokenGenerator

Security token:

Use value type

Value type: X509 certificate token v3

Local name: http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token

URI:

Callback handler: com.ibm.wsspi.wssecurity.auth.callback.X509CallbackHandler

User ID:

Password:

Use key store

Password: itso

Path: C:\SG247206\keystore\was.jks

Type: JKS

Key:

Alias:	Key password:	Key name:
wascert_rsa	itso	CN=wascert, O=IBM, C=US

Client binding configuration:

2

Specify the security token generation information for WebSphere certificate

Configure service requester for signing (cont ...)

Client binding configuration:

Create key locator

3

Key Locator

Key locator name: gen_dsiglocator

Key locator class: com.ibm.wsspi.wssecurity.keyinfo.KeyStoreKeyLocator

Use key store

Password: itso

Path: C:\SG247206\keystore\was.jks

Type: JKS

Key:

Alias:	Key password:	Key name:
wascert_rsa	itso	CN=wascert, O=IBM, C=US

Client binding configuration:

4

- Specify key information type and key locator
- Attach **WebSphere certificate** to request
 - Sign with WebSphere **private key**

Key Information

Key information name: gen_dsigkeyinfo

Key information type: STRREF

Key information class: com.ibm.ws.webservices.wssecurity.keyinfo.STRReferenceContentGenerat

Use key locator

Key locator: gen_dsiglocator

Key name: CN=wascert, O=IBM, C=US

Use token

Token: gen_dsigtgen

Property:

Configure service requester for signing (cont ...)

Client binding configuration:

Specify signing information including signature algorithm

5

Signing Information

Signing Information Name: sign_body

Canonicalization method algorithm: http://www.w3.org/2001/10/xml-exc-c14n#

Show only FIPS compliant algorithms

Signature method algorithm: http://www.w3.org/2000/09/xmldsig#rsa-sha1

Key information name: sign_kinfo

Key information element: gen_dsigkeyinfo

Part Reference

Part reference name: sign_part

Integrity part: int_body

Show only FIPS compliant algorithms

Digest method algorithm: http://www.w3.org/2000/09/xmldsig#sha1

OK Cancel

Client binding configuration:

6

Message part to sign

Note: signing information also needs to be specified for the **response consumer** configuration (not shown here) and the client application needs to be re-deployed

CICS pipeline configuration for signature

```
<service_handler_list>
  <wsse_handler>
    <dfhwsse_configuration version="1">
      <authentication mode="signature">
        <algorithm>http://www.w3.org/2000/09/xmlsig#rsa-sha1</algorithm>
      </authentication>
      <expect_signed_body/>
      <sign_body>
        <algorithm>http://www.w3.org/2000/09/xmlsig#rsa-
          sha1</algorithm>
        <certificate_label>CICSCERT</certificate_label>
      </sign_body>
    </dfhwsse_configuration>
  </wsse_handler>
</service_handler_list>
```

Signed SOAP request message

```
<soapenv:Envelope>
  <soapenv:Header>
    <wsse:Security S:mustUnderstand="1"
      xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
      <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>
  </soapenv:Header>
  <soapenv:Body>
    <p635:ca_request_id>01ORDR</p635:ca_request_id>
    <p635:ca_return_code>0</p635:ca_return_code>
    ... [truncated]
  </soapenv:Body>
</soapenv:Envelope>
```

X.509 cert

Signature

SOAP body

What user ID is used to run Web service request ?

INQUIRE TASK

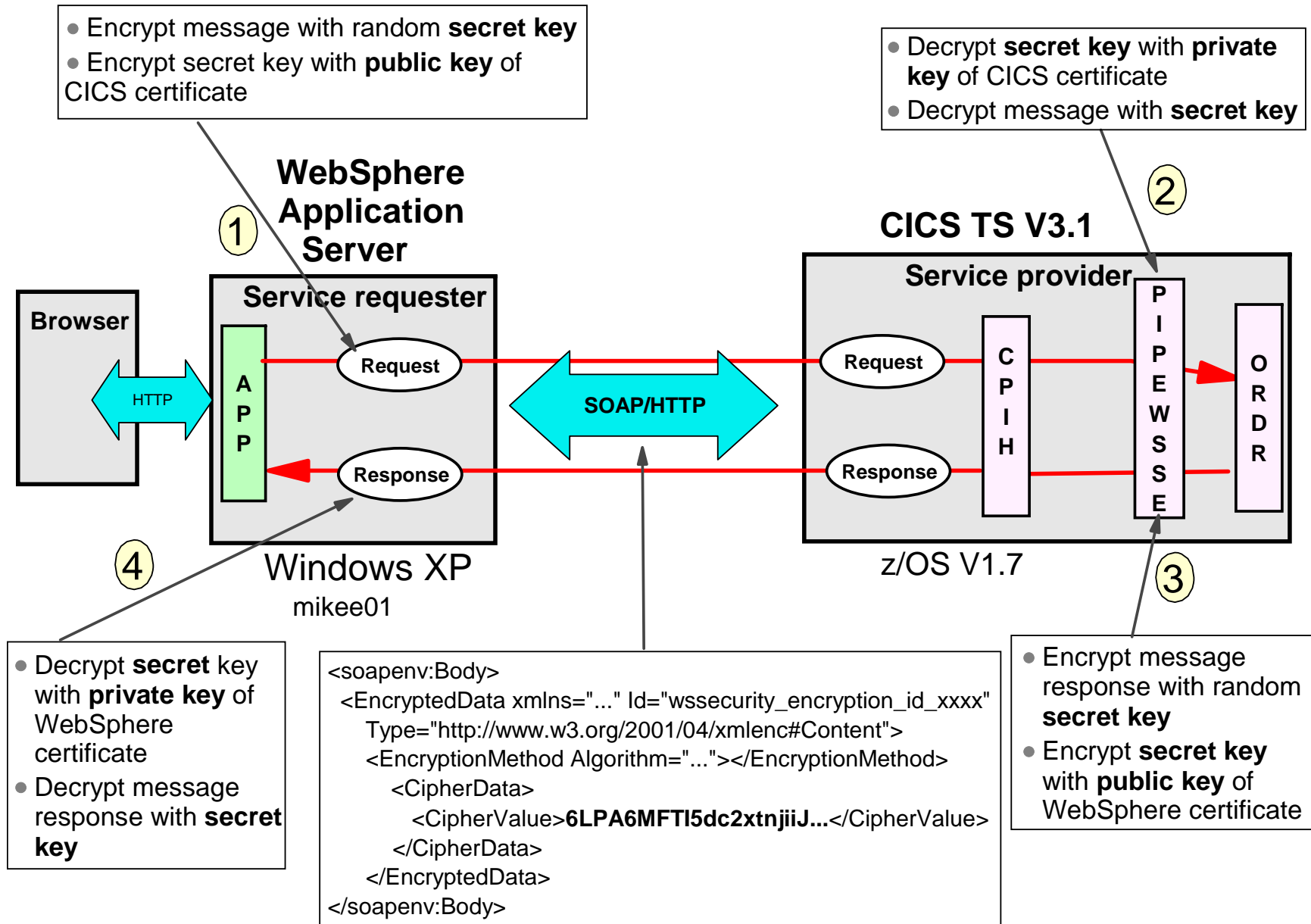
STATUS: RESULTS - OVERTYPE TO MODIFY

```
Tas(0000055) Tra(CPIH)           Sus Tas Pri( 001 )
    Sta(U ) Use(CIWSNW ) Uow(BF3C2E9E2114284E) Hty(RZCBNOTI)
Tas(0000056) Tra(ORDR)           Sus Tas Pri( 001 )
    Sta(U ) Use(WEBAS1 ) Uow(BF3C2E9E24801B8B) Hty(EDF      )
```

SYSID=S3C1 PPLID=A6POS3C1

ORDR transaction runs with user ID WEBAS1 which is the owning user ID of the WebSphere certificate

Scenario 5 – Encrypting a SOAP message



XML Encryption configuration

WebSphere

- Configure the request generator
 - Encrypt SOAP body of message request
- Configure the response consumer
 - Decrypt encrypted response

CICS

- Pipeline configuration file
 - Decrypt encrypted request
 - Encrypt response

Configure service requester for encryption

Client deployment descriptor:
 Create confidentiality message part

1

Confidentiality

Confidentiality name:

Order:

Message Parts:

Parts Dialect	Parts Keyword
<input type="text" value="http://www.ibm.com/websphere/webservic"/>	<input type="text" value="bodycontent"/>

Client binding configuration:

2

Create key locator

Key Locator

Key locator name:

Key locator class:

Use key store

Password:

Path:

Type:

Key:

Alias:	Key password:	Key name:
<input type="text" value="cicscert"/>	<input type="text" value="itso"/>	<input type="text" value="CN=CICSCERT, O=ITSO, C=US"/>

Configure service requester for encryption (cont...)

Client binding configuration :
Specify key information for
encrypting secret key with
CICS public key

3

Key Information

Key information name: genenkeyinfo

Key information type: KEYNAME

Key information class: com.ibm.ws.webservices.wssecurity.keyinfo.KeyNameContentGenerator

Use key locator

Key locator: gen_encklocator

Key name: CN=CICSCERT, O=ITSO, C=US

Encryption Information

Encryption Name: enc_body

Show only FIPS compliant algorithms

Data encryption method algorithm: http://www.w3.org/2001/04/xmlenc#tripleDES-cbc

Key encryption method algorithm: http://www.w3.org/2001/04/xmlenc#rsa-1_5

Key information name: enc_keyinfo

Key information element: genenkeyinfo

Confidentiality part: conf_body

Client binding configuration:

4

- Specify encryption algorithm used to encrypt the data
- Specify encryption algorithm used to encrypt the secret key

Note: not going to show all screenshots for encryption here !! Don't forget to specify the **response consumer** configuration for encryption too (not shown here)

CICS pipeline configuration for encryption

```
<service_handler_list>  
  <wsse_handler>  
    <dfhwsse_configuration version="1">  
      <expect_encrypted_body/>  
      <encrypt_body>  
        <algorithm>http://www.w3.org/2001/04/xmlenc#tripledes-cbc  
      </algorithm>  
      <certificate_label>WASCERT</certificate_label>  
    </encrypt_body>  
  </dfhwsse_configuration>  
</wsse_handler>  
</service_handler_list>
```

Encrypted SOAP request message

```
<soapenv:Envelope>
  <soapenv:Header>
    <wsse:Security S:mustUnderstand="1"
      xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
      <EncryptedKey xmlns="http://www.w3.org/2001/04/xmlenc#">
        <EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
        <ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
          <ds:KeyName>CN=CICSCERT, T=Ciwss3c1-cert, OU=PSSC, O=ITSO,
            L=ENDICOTT, ST=NEW YORK, C=US
          </ds:KeyName>
        </ds:KeyInfo>
        <CipherData>
          <CipherValue>rN8nTy+IlIPN/g4 [truncated] </CipherValue>
        </CipherData>
      </EncryptedKey>
    </wsse:Security>
  </soapenv:Header>
  <soapenv:Body>
    <EncryptedData xmlns="http://www.w3.org/2001/04/xmlenc#">
      <EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc"/>
      <CipherData>
        <CipherValue>y3FFMZ4ckOZjfpyskgrNHQP9Pr [truncated] </CipherValue>
      </CipherData>
    </EncryptedData>
  </soapenv:Body>
</soapenv:Envelope>
```

Key info

Encrypted Secret key

Encrypted Data

Summary

Main messages

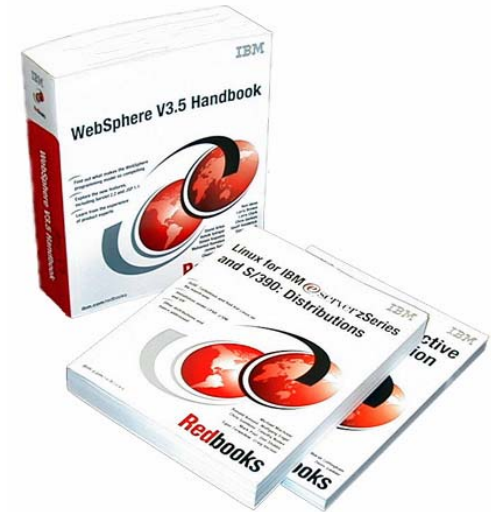
Main considerations for designing a security solution for CICS Web services:

- How to authenticate
- How to transport security credentials in the message
- How to ensure confidentiality and data integrity
- Whether to use WS-Security, transport security or both
- CICS supplied WS-Security support or user-written message handler
- Performance considerations
- Whether identity assertion is required and how to establish trust

Further information

ITSO Redbooks

- “Implementing CICS Web Services” (SG24-7206-2)
- “Web Services Handbook for WebSphere Application Server 6.1” (SG247257)



Information Centers

- CICS

<http://publib.boulder.ibm.com/infocenter/cicsts/v3r1/index.jsp>

- WebSphere

<http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp>

Questions and Answers

© IBM Corporation 2007. All Rights Reserved.

The workshops, sessions and materials have been prepared by IBM or the session speakers and reflect their own views. They are provided for informational purposes only, and are neither intended to, nor shall have the effect of being, legal or other guidance or advice to any participant. While efforts were made to verify the completeness and accuracy of the information contained in this presentation, it is provided AS IS without warranty of any kind, express or implied. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, this presentation or any other materials. Nothing contained in this presentation is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software.

References in this presentation to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. Product release dates and/or capabilities referenced in this presentation may change at any time at IBM's sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way. Nothing contained in these materials is intended to, nor shall have the effect of, stating or implying that any activities undertaken by you will result in any specific sales, revenue growth or other results.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon many factors, including considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve results similar to those stated here.

All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics may vary by customer.

The following are trademarks of the International Business Machines Corporation in the United States and/or other countries. For a complete list of IBM trademarks, see www.ibm.com/legal/copytrade.shtml

AIX, CICS, CICSplex, DB2, DB2 Universal Database, i5/OS, IBM, the IBM logo, IMS, iSeries, Lotus, OMEGAMON, OS/390, Parallel Sysplex, pureXML, Rational, RCAF, Redbooks, Sametime, System i, System i5, System z, Tivoli, WebSphere, and z/OS.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft and Windows are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.