Securing WebSphere Application Server for z/OS

Mike Kearney
IBM Corporation

Session 8376
March 3, 2011

WebSphere Application Server Sessions

Room	Day	Time	Title	Speaker
208B	Monday	11:00	Lab	Multi
201A	Monday	11:00	The Value of the WebSphere Application Server Job Manager	Loos
205A	Monday	4:30	WebSphere Application Server for z/OS -- I am No Longer a Dummy but...	Hutchinson
205B	Tuesday	9:30	Performance Tuning for WebSphere Application Server for z/OS - Practical Advice	Everett
205A	Wednesday	4:30	WebSphere Application Server for z/OS: Tools and Tricks (Potpourri)	Loos and Co.
205A	Wednesday	6:00	WebSphere Application Server for z/OS: Helping Customers Help Themselves	Stephen
206B	Thursday	8:00	Securing WebSphere Application Server for z/OS	Kearney
206B	Thursday	9:30	Application Improvement and Savings Through Simplification	McCorkle
206B	Thursday	11:00	WebSphere Application Server for z/OS: Batch	Bagwell
206A	Thursday	12:15	WebSphere Application Server 101	Stephen
206B	Thursday	1:30	WebSphere Application Server for z/OS: Availability Considerations	Bagwell
206B	Thursday	3:00	WebSphere Application Server: z/OS Exploitation/Differentiation	Follis
206B	Thursday	4:30	Performance Tuning for WebSphere Application Server for z/OS - WAS and WLM Interactions and Concepts	Follis

SHARE in Anaheim 2011

Agenda

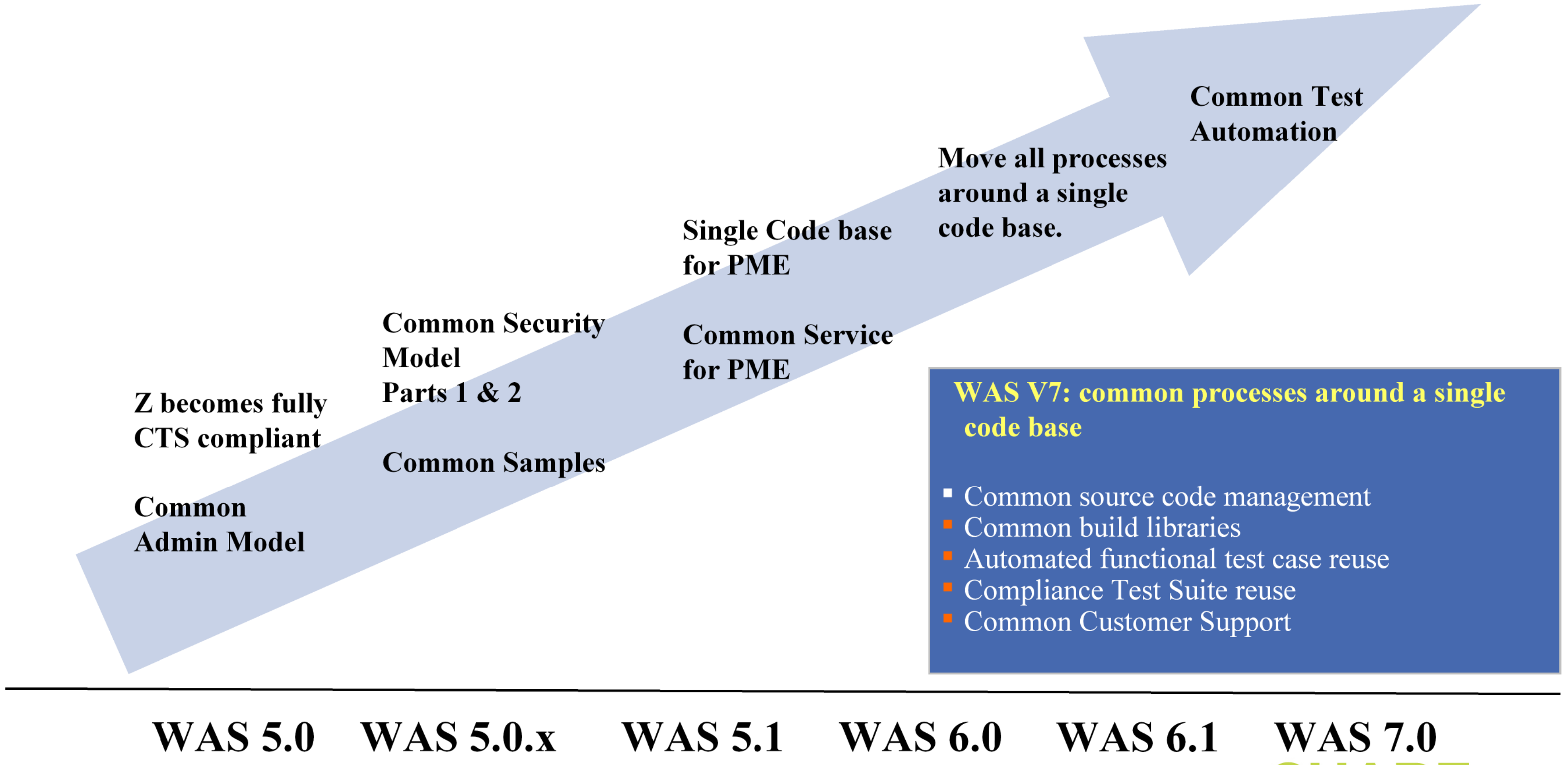
- Web Based Applications Authentication and Authorization
- EJB Applications Authentication and Authorization
- Web Services
- Additional Security Features

Note: This presentation will focus on WebSphere application Server V6.1 and above. Any WebSphere Application Server V7 specific features will be noted.

Note: All the features discussed apply to all platforms WebSphere Support. Any z/OS specific features will be noted.

Unifying the WAS Code Base

An organizational initiative that spans several releases aimed at merging our distributed and z/OS code and processes for the benefit of our customers



WAS V7: common processes around a single code base

- Common source code management
- Common build libraries
- Automated functional test case reuse
- Compliance Test Suite reuse
- Common Customer Support

WebSphere Security Principles

- **Secure by Default**
  - Starting with WAS V6.1, by design, we are secure out of the box.
  - WAS V7, additional defaults were changed
- **Ease of Use**
  - “Easy of use”, rich programming references, samples, etc.
    - Standard Compliance
    - Programming Flexibility
    - Simple to report and fix security vulnerability
    - Simple steps to configure
- **Defense in Depth**
  - WebSphere another layer of defense
- **Accountability**
  - Users held accountable for their actions
  - Ability to Audit
  - WebSphere Auditing added in WAS V7
- **Separation of Privileges**
  - No single person should have enough authority to cause a critical event to take place
- **Least Privilege**
  - Idea of granting just the least possible amount of privileges to permit a legitimate action with the idea of preventing the malicious behavior
- **Secure code is quality code**
  - Leave no Weakness in the code for exploitation

Why J2EE Security Model is important

- J2EE Security Model allows for
  - Security administration and management handle by the Infrastructure instead of custom applications.
  - Security implementation technology is independent (from application developer's view)
  - Application is expected to "lean"on server vendor
  - Authentication is not application responsibility
  - Applications deal only with authorization via declarations (in XML) and/or simple APIs
  - Container is the broker for Security
    - Applications "leans" on the WAS Container
    - WAS Container can administer Security or WAS Container "leans" on an optional pluggable Security Solution to manage the Security aspects of Users, Groups, and resource (roles).
- The J2EE Security specification is very high level and provides only minimal APIs

J2EE Security
Web Based Applications

- Authentication and Authorization is defined outside of the application using the Application's Deployment Descriptor.
- Located in the WAR file under web.xml
- Typically tools such as RAD or the AST are used to generate this xml file.

```
<web-app id="WebApp_ID">
    <security-constraint>
        <web-resource-collection>
            <web-resource-name>foo</web-resource-name>
            <url-pattern>myServlet</url-pattern>
            <http-method>GET</http-method>
            <http-method>PUT</http-method>
        </web-resource-collection>
        <auth-constraint>
            <role-name>myRole1</role-name>
        </auth-constraint>
        <user-data-constraint>
            <transport-guarantee>NONE</transport-guarantee>
        </user-data-constraint>
    </security-constraint>
    <login-config>
        <auth-method>BASIC</auth-method>
            <realm-name>MyRealm</realm-name>
    </login-config>
    <security-role>
        <role-name>myRole2</role-name>
    </security-role>
    <security-role>
        <role-name>MyRole1</role-name>
    </security-role>
</web-app>
```

- Define a **Web Resource**
  - The URI or URI Patter to protect
  - For static Http Method to protect such as GET or POST
  - For dynamic Http method (Servlet/JSP) to protect such as GET, PUT, POST, DELETE, HEAD, OPTION, TRACE
- Define **Authentication constraint**
  - List all the security roles needed to gain access to the Web Resource.
  - A User must be will belong to at least one of these roles.
- Define **User Data constraints**: allows you to specify the required transport guarantee that defines the communication between the client and the Web application.
  - None – no transport guarantee requires
  - Integral – ensures data cannot be changed in transit – SSL used
  - Confidential – ensures data cannot be viewed in transit – SSL used
- Define **Login Config**
  - Specify Basic Authentication (userID/Password) or Form Based Login
- Define **Security Roles**
  - List all the security roles that will be used by this application
  - Must include roles that were listed in the Authenticated Constraint plus any programmatic roles.

WebSphere User Registry

WebSphere Security requires a User Registry to be configured.

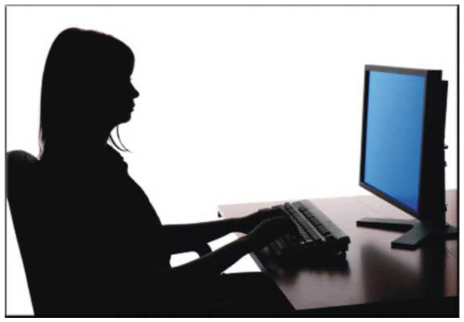
- Used during Authentication process to verify User Identity and construct the User’s group information as part of the Subject
- Used by WebSphere Authorization Mapping in order to map J2EE roles or Administrator roles to User or Groups.

User Registry - options:

- LocalOS (SAF)
- LDAP
- Custom
- Federated Repository

- z/OS Local Registry uses SAF plus…
  - Can use the mixed case password option for RACF.
    - Must use z/OS Version 1.7 or higher
    - local operating system registry
    - mixed case is turn on by using the SETROPTS PASSWORD(MIXEDCASE) command.
  - Can support the z/OS 1.9 Pass Phase
    - Requires z/OS 1.9
    - Requires WAS6.1.0.15

J2EE Web Authentication

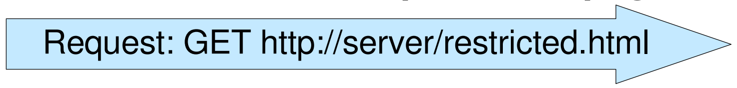


- WAS Container is responsible for the full aspects for Authentication.
  - Identify who you are …
  - No server side APIs or actions specified. Entirely responsibility of container.
  - Basic Authentication (e.g UserID/Password)
  - Form based login -custom login page
  - SSL mutual auth (e.g, Client Certificate)
  - Customized Login using JAAS

- Note that J2EE requires lazy authentication -users are not challenged until they attempt to use a secured resource

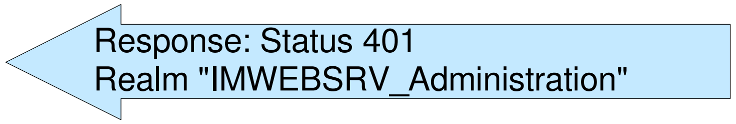
Basic Authentication

1. User clicks on link to protected page



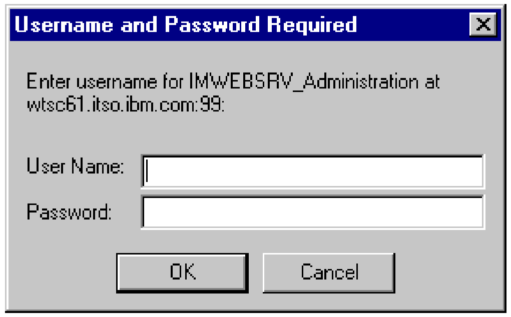
Request: GET http://server/restricted.html

2. Server checks authority and rejects request



Response: Status 401
Realm "IMWEBSRV_Administration"

3. Browser pop-up window prompts user for userId and password



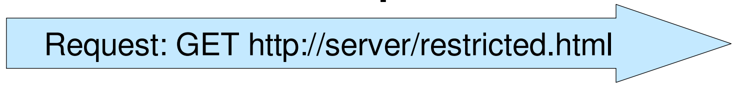
Username and Password Required

Enter username for IMWEBSRV_Administration at wtsc61.itso.ibm.com:99:

User Name:

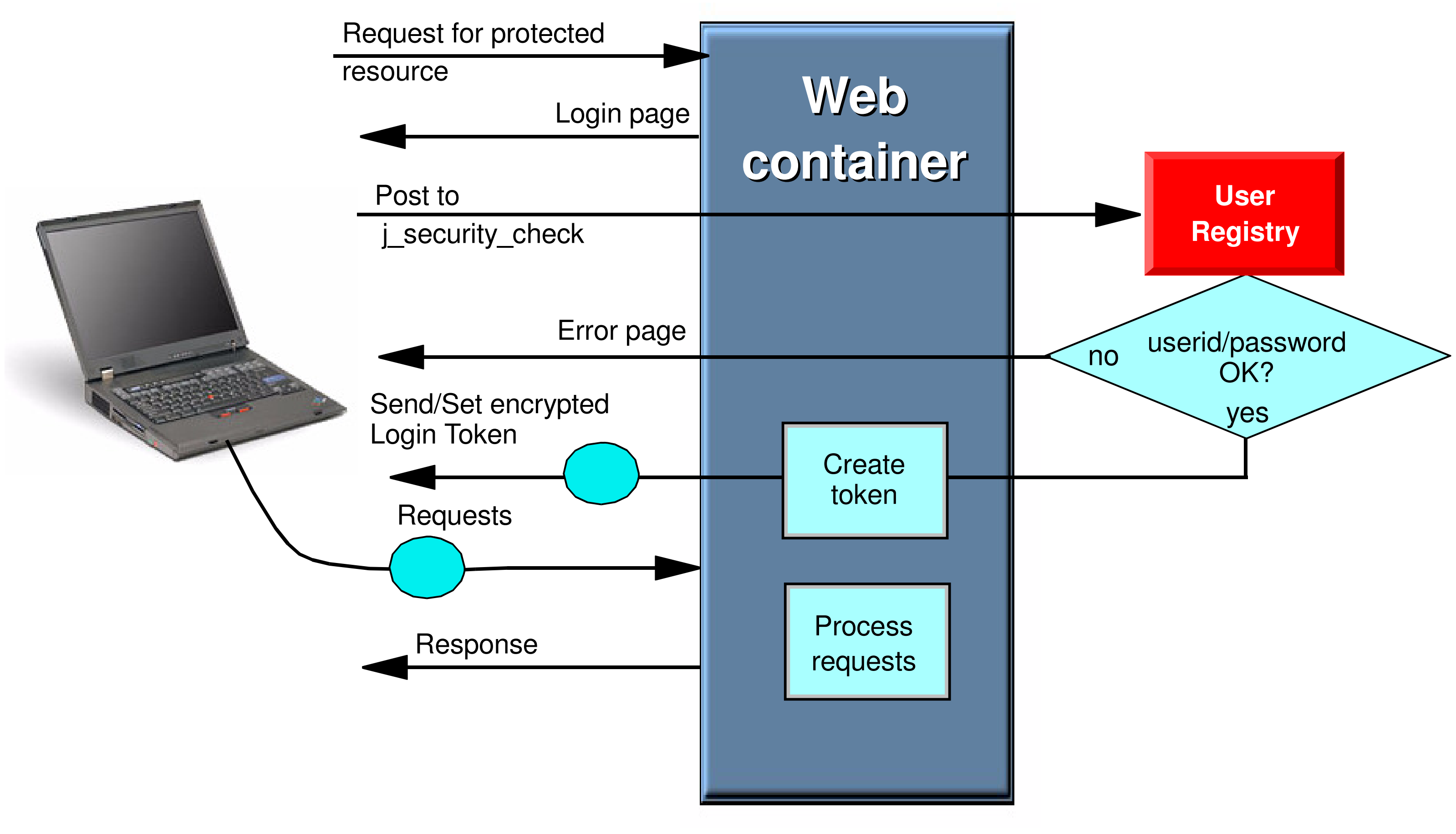
Password:

4. Browser resends request with userid and password in request header



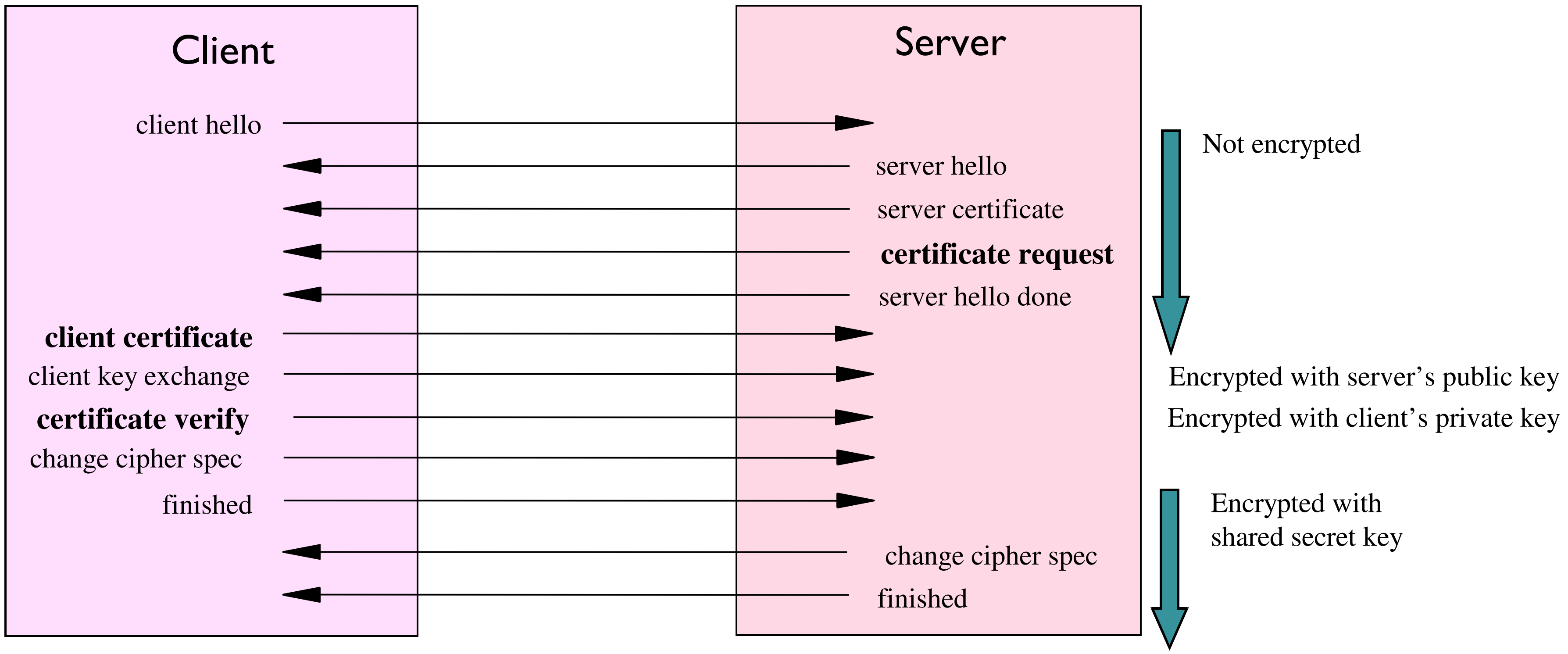
Request: GET http://server/restricted.html

Form-based login

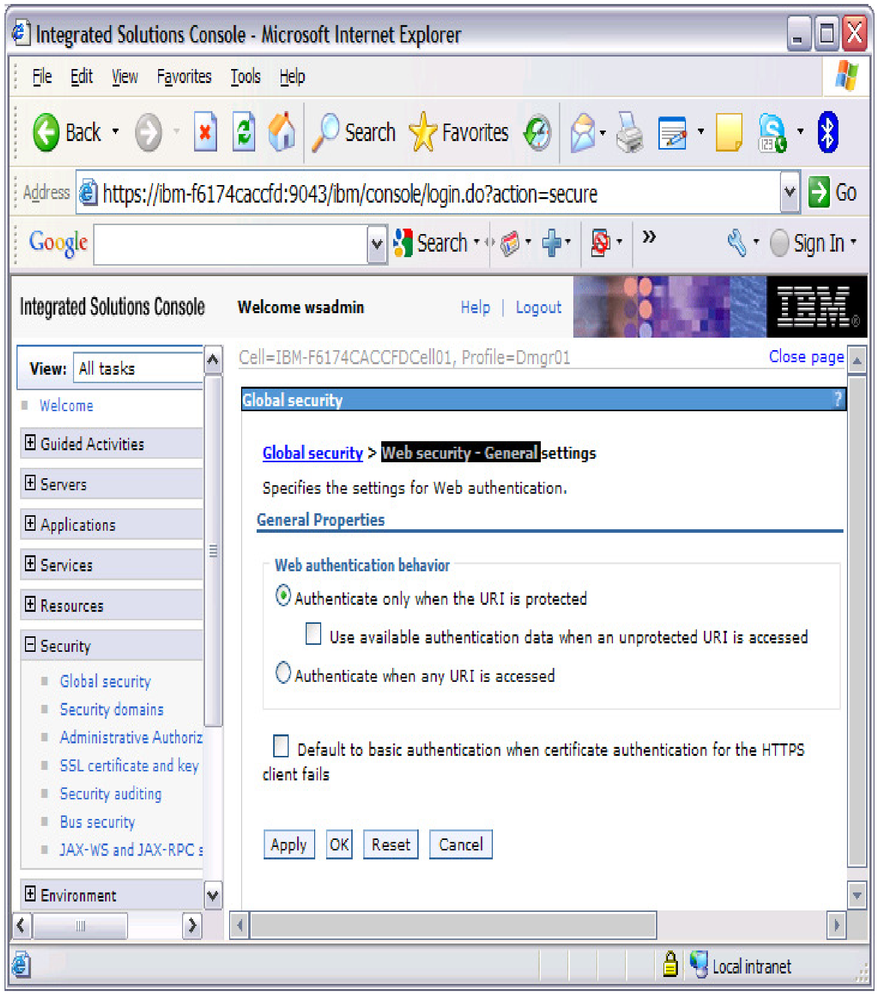


The Login Token is typically a LtpaToken cookie but not necessarily.

Certificate-based Authentication



Web Security General Settings



Global security > Web security - General settings

Specifies the settings for Web authentication.

General Properties

Web authentication behavior

[✓ Authenticate only when the URI is protected]

☐ Use available authentication data when an unprotected URI is accessed

○ Authenticate when any URI is accessed

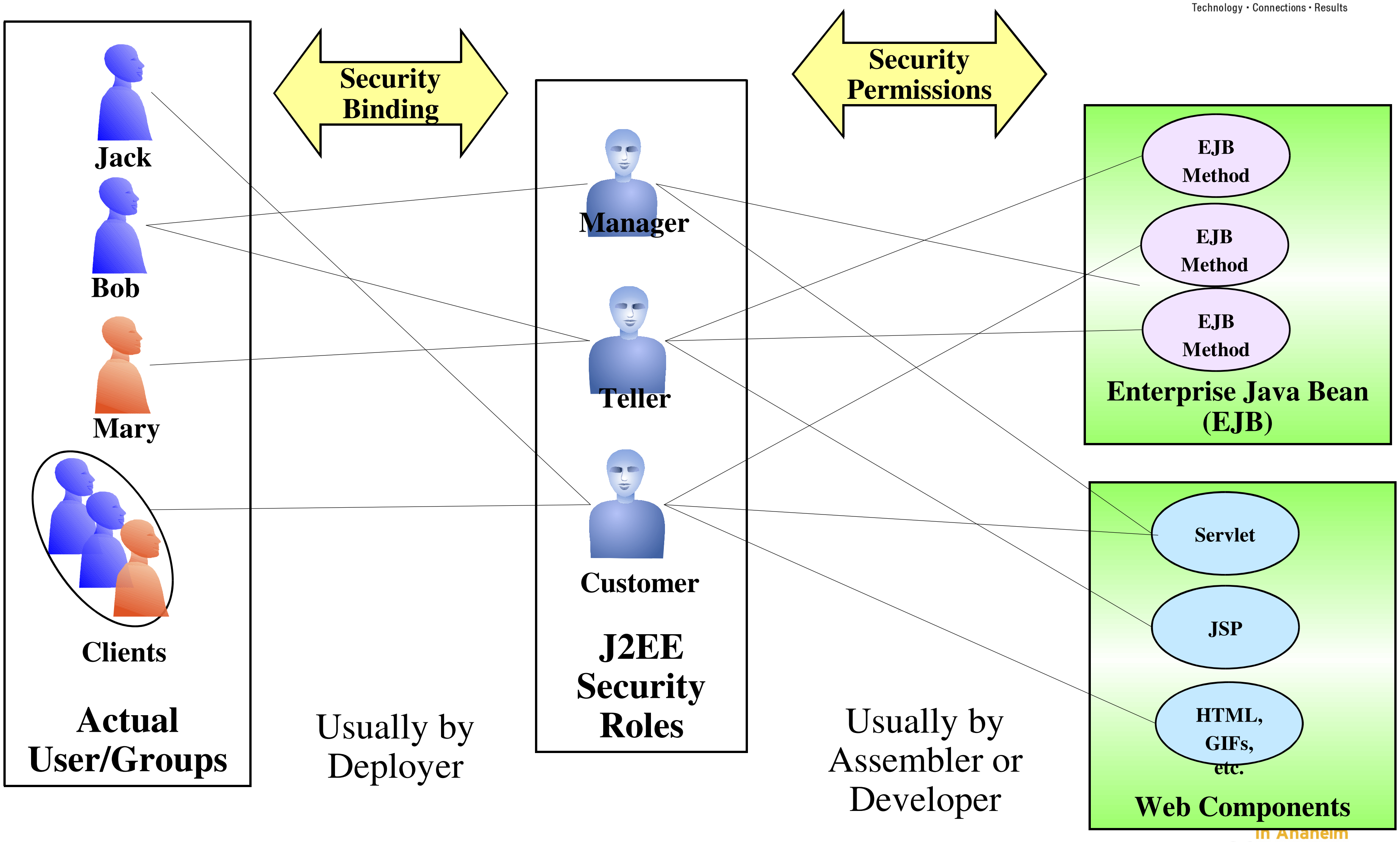
☐ Default to basic authentication when certificate authentication for the HTTPS client fails

- **Web authentication behavior**
  - **Authentication only when the URI is protected**
    - Authentication will only be performed for URI and Auth Methods that are protected via web.xml
    - Optionally the application can be aware of the authentication data when for unprotected URIs.
  - **Authenticated when any URI is accessed**
    - Regardless to the constraints define in web.xml, all URI will be forced to be authenticated.

J2EE Authorization Basics

- **Principals**
  - Things that can be authenticated: users, servers, etc
- **Roles**
  - An application centric name that represents a logical set of principals
    - □Used in Permissions and Constraints to specify who can do what
    - □Just string names. E.g.: “managers,”“customers”
- **Declarative Security**
  - "Declarative security refers to the means of expressing an application’s security structure, including security roles, access control, and authentication requirements in a form external to the application [code]." --J2EE 1.3 spec.
    - Security Roles
      - *Method permissions*
      - *RunAs information*
      - *Permission to –URL patterns (can be more than one)*
    - HTTP Methods (GET, POST, DELETE, etc)
    - Transport restrictions (none, integrity, confidential)
    - RunAs
- **Programmatic Security**
  - Allows for conditional checking of roles within the applications
  - Ability for a program to get the current userID.
  - For example, Manager role is required when depositing over $20,000
  - Assignment and management of the role is handled outside of the application

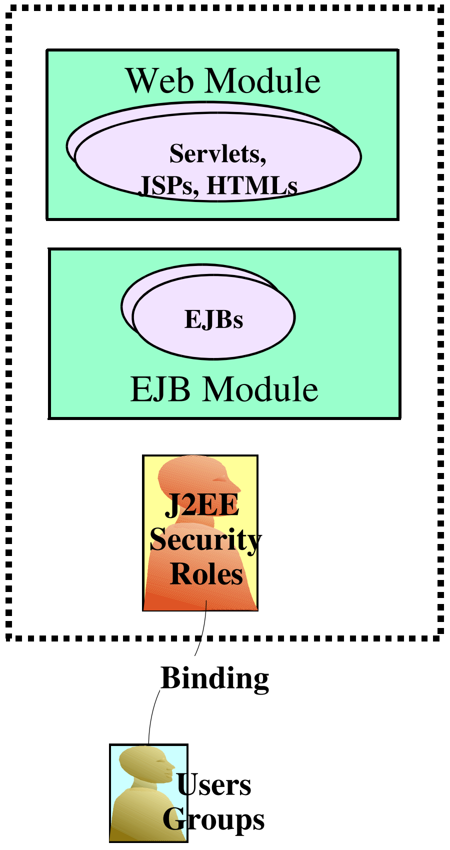
J2EE Role Mappings



Application Security Tasks and Roles

	Task	Role	Tools	Files Chg
1	Specific J2EE Programmatic Java API in code	Developer	RAD or any IDE	Java Code
2	Define J2EE Security Roles	Assembler	RAD, AST	application.xml
3	Map Developer J2EE roles to a bind-able referenced role	Assembler	RAD, AST	IBM binding files ibm-web-bnd.xmi
4	Specify the web constraints and declarative J2EE roles	Assembler	RAD, AST	web.xml
5	Map J2EE roles references from step 3 to users, groups, or both	Assembler or RACF Admin	WAS, RAD, AST	Ibm-appication-bnd.xml, JACC provider, or SAF

WAS for z/OS SAF Authorization

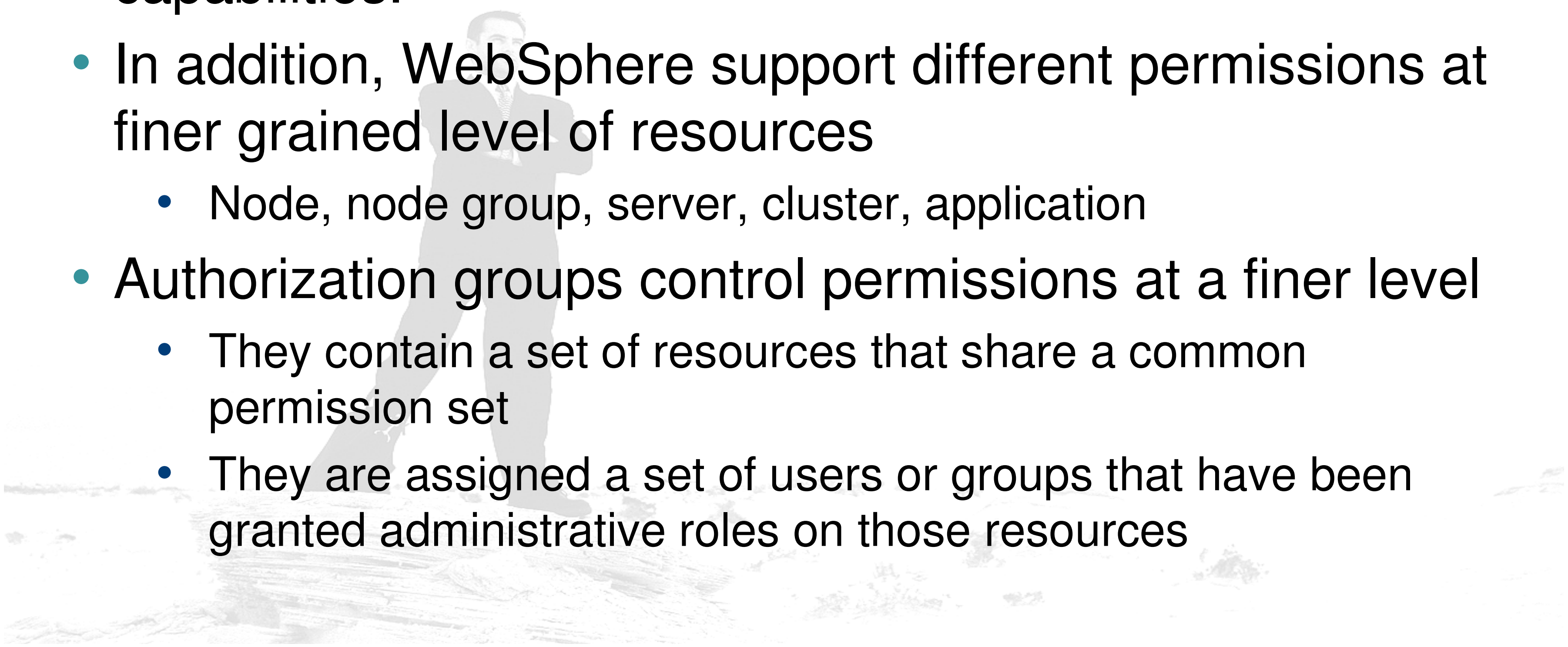


- You can either use WebSphere Authorization or SAF Authorization to manage your Role to User Mappings.
- **WebSphere Authorization**, the administrator roles and application roles are managed within WAS using the WAS Administration console and the deployment descriptor.
- **WebSphere SAF authorization**, the administrator roles and application roles are managed within SAF. Any Administration and/or Application roles configured via the WAS Administration console will be ignored.
- In addition, the application deployment information for "Everyone", "All Authenticated", and "User/group to role" attributes are ignored and managed within the SAF Authorization Management facilities.
- SAF manages roles using the EJBROLE SAF Class and the SAF Profile represents the role.
  - **RDEFINE EJBROLE (safPrefix.)myrole UACC(NONE)**
  - **PERMIT (safPrefix.)myrole CLASS(EJBROLE) ID(User1) ACCESS(READ)**

z/OS Security Domain Name V61
z/OS SAF Prefix V7 and beyond

- **optionalSecurityDomainName** was renamed to **SAF Prefix** in WAS7 to remove any confusion with the Multiple Security Domain Feature delivered in V7.
- SAF Prefix is established during the installation task using the z/OS customization Dialogs.
- The specification of a security domain prefix affects the specific EJBROLE profiles.
- When enabled, the EJBROLE profile role can be scoped down to a cell level. For example, I can have a different User have administrator role access to different cells
  - Production Cell might have
    - RDEFINE EJBROLE (PRODCELL.administrator UACC(NONE)
    - PERMIT (PRODCELL. administrator CLASS(EJBROLE) ID(User1) ACCESS(READ)
  - Test Cell might have
    - RDEFINE EJBROLE (TESTCELL.administrator UACC(NONE)
    - PERMIT (TESTCELL.administrator CLASS(EJBROLE) ID(User2) ACCESS(READ)

Administrative Privileges



- WAS Administration offers a separation of privilege model with multiple roles with different administration capabilities.
- In addition, WebSphere support different permissions at finer grained level of resources
  - Node, node group, server, cluster, application
- Authorization groups control permissions at a finer level
  - They contain a set of resources that share a common permission set
  - They are assigned a set of users or groups that have been granted administrative roles on those resources

Web Applications Programmatic APIs

- **isUserInRole** (String role-name): Returns true if the remote user is granted the specified security role. Returns false, if the remote user is not granted the specified role, or no user is authenticated
- **getUserPrincipal**(): Returns the java.security.Principal object containing the remote user name
- **getRemoteUser**(): Returns the user name the client used for authentication (String)

Example:

```
public void doGet(HttpServletRequest request, HttpServletResponse response) {
        // to get remote user using getUserPrincipal()
        java.security.Principal principal = request.getUserPrincipal();
        String remoteUser = principal.getName();
        // to get remote user using getRemoteUser()
        remoteUser = request.getRemoteUser();
        // to check if remote user is granted Manager role, using isUserInRole
        boolean isMgr = request.isUserInRole("Manager");
}
```

J2EE Security
Enterprise Java Bean Based Applications

- Role Authorization and the runAs identity can be defined outside of the application using the Application's Deployment Descriptor or defined using annotations with in the Java Source code.
- Located in the EJB jar under ejb-jar.xml
- Typically tools such as RAD or the AST are used to generate this xml file.

```
<ejb-jar id="ejb-jar_ID">
….
<assembly-descriptor>
    <security-role>
        <role-name>myRole</role-name>
    </security-role>
    <method-permission>
        <role-name>myRole</role-name>
        <method>
            <ejb-name>myEJB</ejb-name>
            <method-intf>Home</method-intf>
            <method-name>*</method-name>
        </method>
    </method-permission>
</assembly-descriptor>
</ejb-jar>
```

- Define **Security Roles**
  - List all the security roles that will be used by this application
  - Must include roles that were listed in the Authenticated Constraint plus any programmatic roles.
- Define **Security Identity**
  - Specifies the security identity to be used to invoke methods in a particular EJB
  - Options
    - *Run as the caller identity*
    - *Run as a role and then the role is associated with an identity.*
    - *Run as a specified Identity*
    - *Run using the server identity*

EJB Applications Programmatic APIs

- **IsCallerInRole** (String role-name)
  - Returns true if the bean caller is granted the specified security role
  - If the caller is not granted the specified role, or if the caller is not authenticated, it returns false
  - If the specified role is granted **Everyone** access, it always returns true
  - Must have security role reference defined in the deployment descriptor
- **getCallerPrincipal()**:
  - Returns the java.security.Principal object containing the bean caller name
  - If the caller is not authenticated, it returns a principal containing UNAUTHENTICATED name

Example:

```
public void myEJBmethod() {
          …
          // to get bean's caller using getCallerPrincipal()
          java.security.Principal principal = context.getCallerPrincipal();
          String callerId= principal.getName();
          // to check if bean's caller is granted Mgr role
          boolean isMgr = context.isCallerInRole("Mgr");
          …
}
```

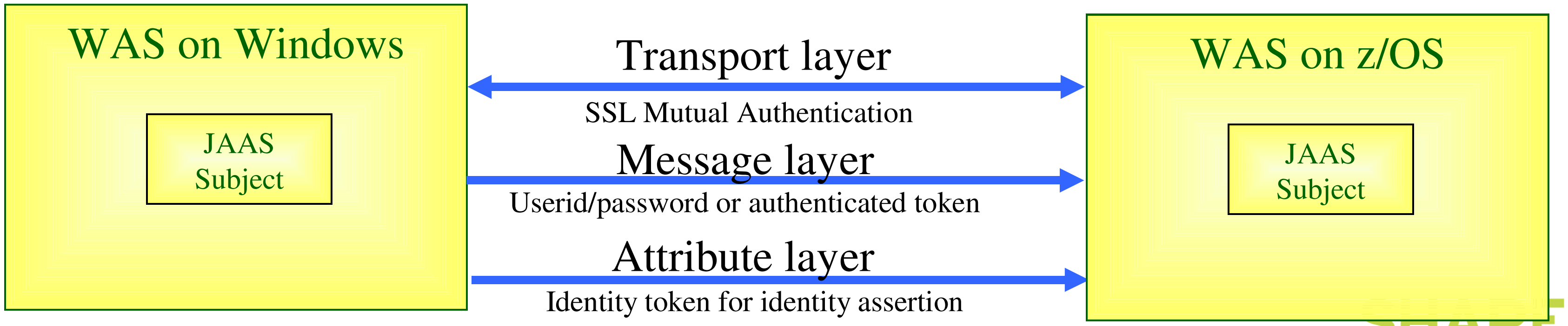
J2EE
EJB Security Annotation New WAS7!

- Beginning with WAS7 and EE5, EJB authorization can be specified in the Java Source Files instead of using the deployment Descriptor.
- Ddd
  - **@PermitAll –** The given method or all the methods for the EJB are accessible by everyone.
  - **@DenyAll –** The given method for the EJB can not be accessible by anyone.
  - **@RolesAllowed –** The given method or all the methods for the EJB can be accessed by users associated with the list of roles.
  - **@DeclareRoles –** To define all the roles for a given EJB.
  - **@RunAs –** Specifies the user Identity to be used.

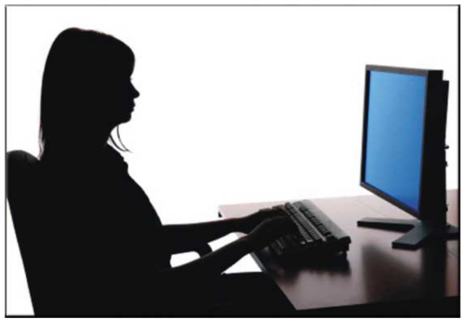
CSIv2 Overview

- CSIv2 defines the Security Attribute Service (SAS) that enables interoperable authentication, delegation and privileges
  - CSIv2 SAS supports SSL and interoperability across J2EE vendors (starting with J2EE 1.3 specification)
- Provides 3 layers of authentication, as shown in the table below:

Transport layer	Uses SSL client certificate as the identity	Attribute layer has the highest priority, followed by the message layer, and then the transport layer. If a client sends all three, only the identity token from the attribute layer is used
Message layer	Uses an user ID/password or an authenticated token with an expiration	
Attribute layer	Uses Identity token to support Identity assertion of an upstream server	


WAS on Windows
JAAS Subject
Transport layer
SSL Mutual Authentication
Message layer
Userid/password or authenticated token
Attribute layer
Identity token for identity assertion
WAS on z/OS
JAAS Subject

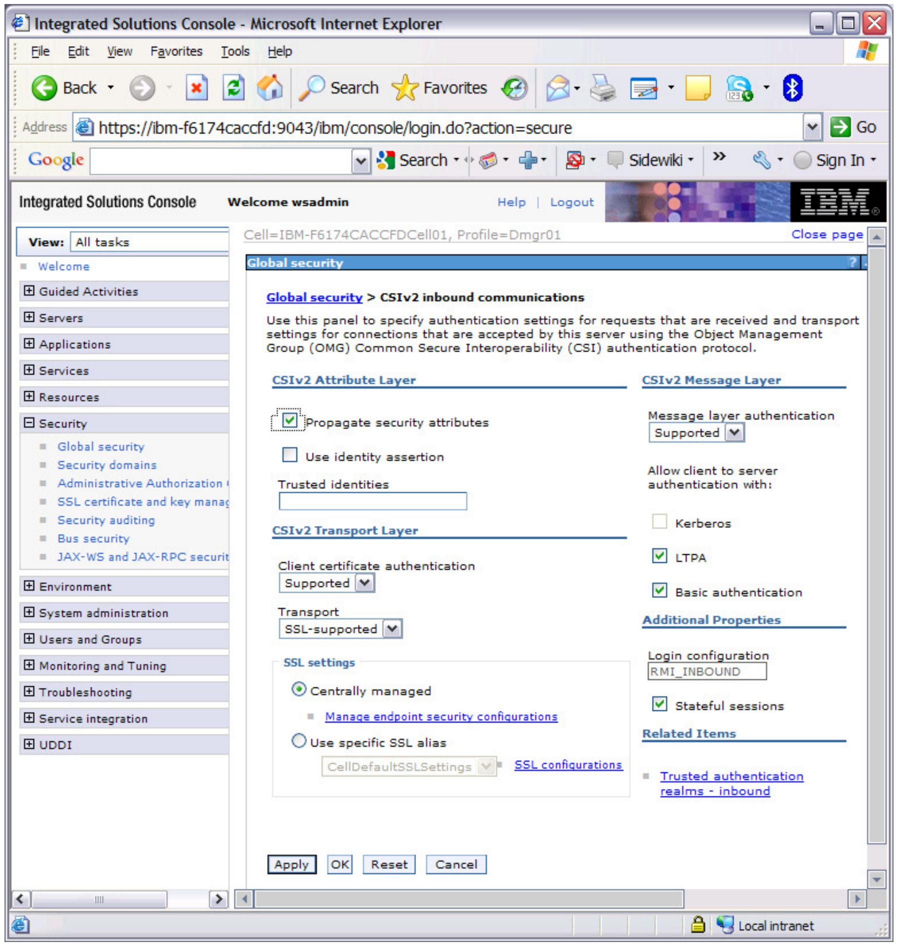
J2EE EJB Authentication



- Similar to Web Applications, the WAS EJB Container is responsible for the full aspects for Authentication.

- **Uses Common Secure Interoperability Version 2 (CSiV2)**
  - Defined by Object Management Group (OMG) standard to provide open, secure interoperability common framework across J2EE servers
  - CSIv2 Protocol facilitates interoperability by serving as the higher-level protocol under which secure transports (SSL/TLS) can be unified
  - Distinguishes between network level (transport layer) and application level (message layer) authentication
    - Transport layer supports PKI client certificates authentication using SSL
    - Message layer supports the exchange of security attributes
      - Standard provided for several token types including basic authentication, asserted identities, Kerberos, etc
      - WAS of course adds LTPA tokens as an additional type

EJB Leverages RMI/IIOP Security using CSIV2 – inbound communications


Integrated Solutions Console - Microsoft Internet Explorer

Address https://ibm-f6174caccfd:9043/ibm/console/login.do?action=secure

Integrated Solutions Console   Welcome wsadmin   Help | Logout

View: All tasks

Cell=IBM-F6174CACCFDCell01, Profile=Dmgr01

Global security

Global security > CSIv2 inbound communications

Use this panel to specify authentication settings for requests that are received and transport settings for connections that are accepted by this server using the Object Management Group (OMG) Common Secure Interoperability (CSI) authentication protocol.

CSIv2 Attribute Layer

- [✓] Propagate security attributes
- [ ] Use identity assertion

Trusted identities

CSIv2 Transport Layer

Client certificate authentication: Supported

Transport: SSL-supported

SSL settings

- [✓ Centrally managed] Manage endpoint security configurations
- Use specific SSL alias: CellDefaultSSLSettings   SSL configurations

CSIv2 Message Layer

Message layer authentication: Supported

Allow client to server authentication with:

- [ ] Kerberos
- [✓] LTPA
- [✓] Basic authentication

Additional Properties

Login configuration: RMI_INBOUND

- [✓] Stateful sessions

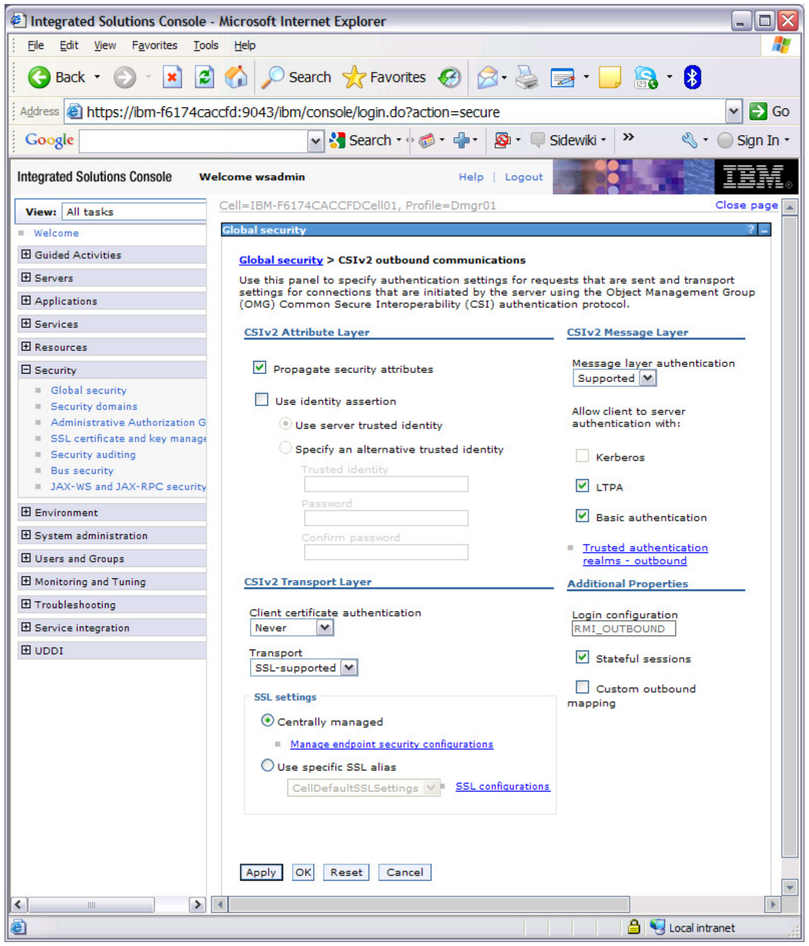
Related Items

- Trusted authentication realms - inbound

Apply | OK | Reset | Cancel

- **Identity Assertion** – When enabled, the server permits an identity that was asserted from an upstream server. It requires the Trusted Identities to contain upstream serverID that you trust to assert.
- **Message Layer authentication** – Specifies if authentication is required, supported (optional), or none. Also need to specify the authentication types supported ie LTPA, Kerberos, or basic Authentication.
- **Client Certificate Authentication** – Specifies required, Supported or none.
- **Transport –** Specify if SSL is required, supported (optional) or none.

EJB Leverages RMI/IIOP Security using CSIV2 – outbound communications



Global security > CSIv2 outbound communications

Use this panel to specify authentication settings for requests that are sent and transport settings for connections that are initiated by the server using the Object Management Group (OMG) Common Secure Interoperability (CSI) authentication protocol.

CSIv2 Attribute Layer

- [✓] Propagate security attributes
- [ ] Use identity assertion
  - (●) Use server trusted identity
  - ( ) Specify an alternative trusted identity
    - Trusted identity
    - Password
    - Confirm password

CSIv2 Message Layer

Message layer authentication: Supported

Allow client to server authentication with:

- [ ] Kerberos
- [✓] LTPA
- [✓] Basic authentication
- Trusted authentication realms - outbound

CSIv2 Transport Layer

Client certificate authentication: Never

Transport: SSL-supported

SSL settings

- (●) Centrally managed
  - Manage endpoint security configurations
- ( ) Use specific SSL alias: CellDefaultSSLSettings — SSL configurations

Additional Properties

Login configuration: RMI_OUTBOUND

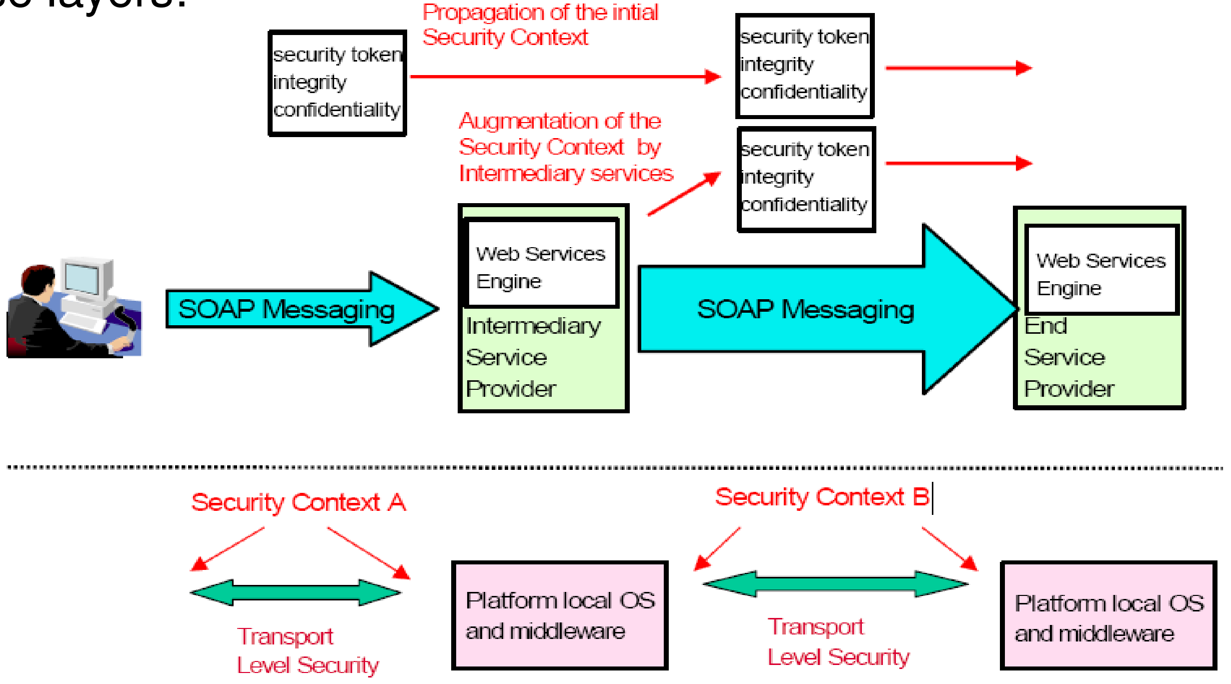
- [✓] Stateful sessions
- [ ] Custom outbound mapping

Integrated Solutions Console - Microsoft Internet Explorer
https://ibm-f6174caccfd:9043/ibm/console/login.do?action=secure
Welcome wsadmin
Cell=IBM-F6174CACCFDCell01, Profile=Dmgr01
Apply
OK
Reset
Cancel

- **Identity Assertion** – The Server will perform an identity assertion going outbound. Either a ServerID or some specified userid/password can be used.
- **Message Layer authentication** – Specifies if authentication is required, supported (optional), or none. Also need to specify the authentication types supported ie LTPA, Kerberos, or basic Authentication.
- **Client Certificate Authentication** – Specifies required, Supported or none.
- **Transport –** Specify if SSL is required, supported (optional) or none.

Web Services security protocol layers

- Web services messaging relies on two protocol layers. Security can be implemented at each of these layers:
  - **The Transport layer**: HTTP, RMI/IIOP, WebSphere MQ, and so on typically carry authentication information in headers, with optional additional security provided by encapsulation in the SSL/TLS protocol.

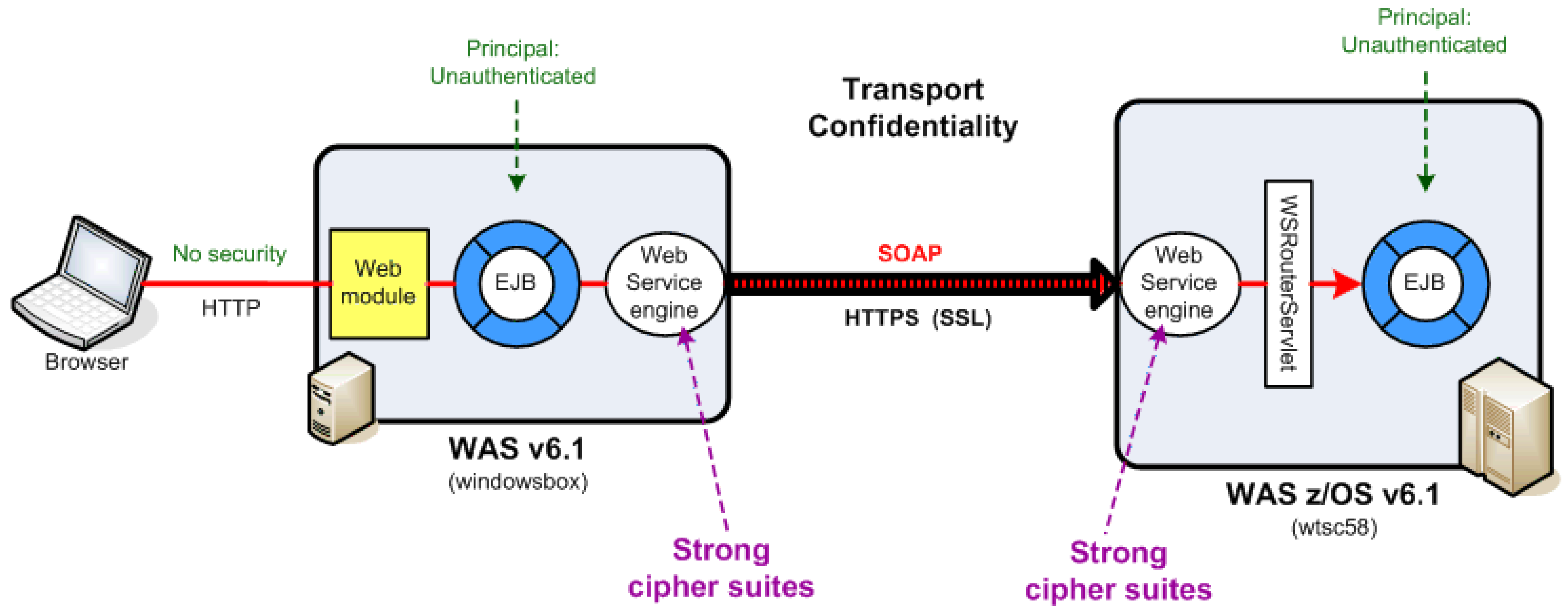


- **The SOAP or Message layer**: The WS-Security specifications indicate how SOAP XML messages can carry security assertions and contexts.

Web Services Transport layer security

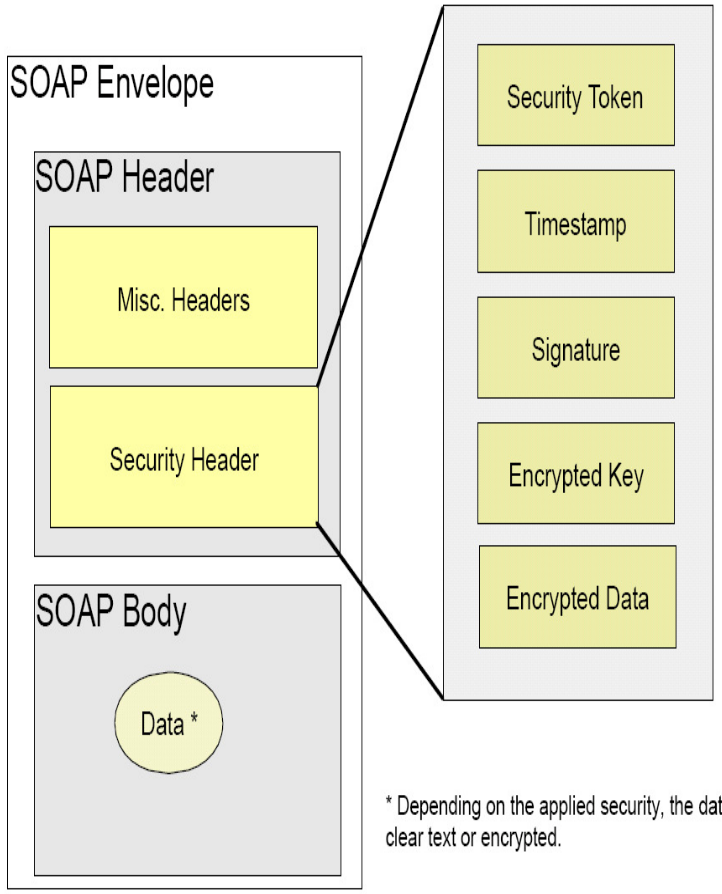
- SSL is the most popular way to encrypt communication between business partners over the Internet.
- It simply creates a secure pipeline between two nodes and encrypts all traffic flowing between the nodes.
  - SSL provides a straightforward way to provide **confidentiality**.
  - It also includes a built-in communication **integrity** check.
  - Connection layer **authentication** is achieved by the client always authenticating the server, and optimally being authenticated by the server, through the exchange of X.509 certificates.
- HTTPS (SSL over HTTP) has the following advantages:
  - It can be used to provide a very fast and secure transport for Web services.
  - It provides authentication through either HTTP Basic Authentication or a client X.509 certificate.
  - It provides integrity between the client and server by using asymmetric key cryptography to establish authenticity of server and client and to securely share a secret key.
  - It provides confidentiality between the client and server through efficient shared key cryptography.
  - It has good support for a broad array of hardware accelerators.
  - It is mature and similarly implemented by most vendors, and therefore, is subject to few interoperability problems.
- JMS: SSL can be used between messaging engines.

For Example: Web services transport security confidentiality via SSL scenario



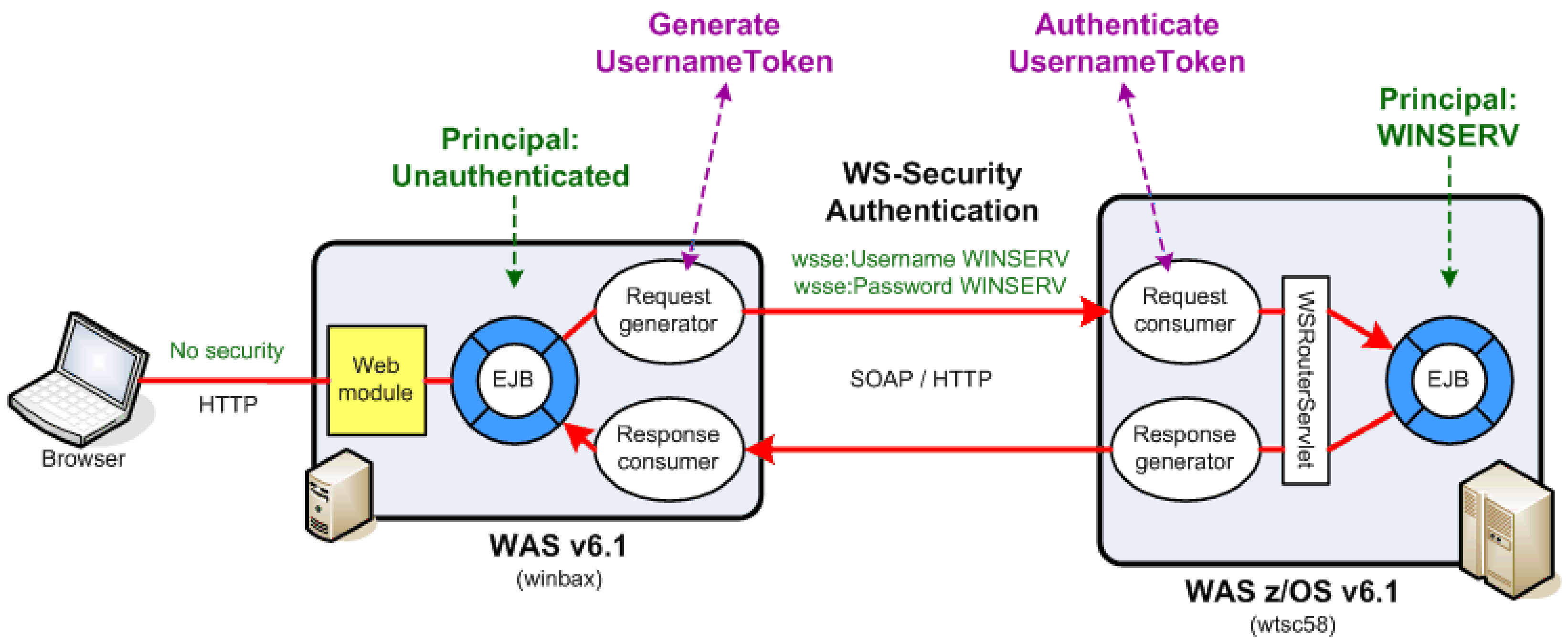
Web Services Message level security

- WS-Security provides a general purpose mechanism for associating security tokens with messages.
  - Typical tokens in WebSphere-based Web services are user name and password, X.509 certificates, and LTPA tokens.
- WS-Security supports the following authentication mechanisms via the insertion of a security token:
  - **Basic Authentication**: The security token includes the user name and password information, and is generated as <wsse:UsernameToken> with <wsse:Username> and <wsse:Password>.
  - **Signature**: The security token includes the X.509 certificate of the signer of the data and is generated as <ds:Signature> with <wsse:BinarySecurityToken>.
  - **ID assertion**: ID assertion includes a user name only, since the identity is asserted, and is generated as <wsse:UsernameToken> with <wsse:Username>.
  - **Custom**: This mechanism includes a custom-defined token.
  - **LTPA**: Use of an LTPA token is a WebSphere-specific customer token, generating a <wsse:UsernameToken> with <wsse:Username>



* Depending on the applied security, the data is clear text or encrypted.

For Example: Web service message security authentication scenario



Web Services Decision Tree

	Can use Transport Level	**Can use WS-Security**
Ability to encrypt the entire message	Yes	Yes
Ability to only encrypt a portion of the message	No	Yes
Ability to handle 1 identity	Yes	Yes
Ability to handle authentication/Assertion of multiple identities	No	Yes
Ability to handle non-repudiation ie: show origin (authentication and content (integrity) of the message	No	Yes
Non SOAP message	Yes	No
Identity is in the transport header	Yes	No
Identity is in the SOAP message	No	Yes
SOAP message being passed in multiple transport types	No	Yes

WebSphere Auditing added in WAS7

Designed to support a variety of Audit points such as Authentication, Authorization, Principal/Credential mapping, User registry and Identity management, Logouts,

A solid Auditing Strategy may help by giving the organization the critical information needed when a penetration occurs.

WebSphere® Security

WebSphere® Service Integration Bus (SIB)

WebSphere® Web Services Security

WAS flat Audit File optionally configured as virtually tamper proof using signing and encryption.

z/OS SMF Type 83 subtype 5.

- Look for SMF Data Area Book to be updates
- The SMF Dump utility will be updated to document SMF83 subtype 5.

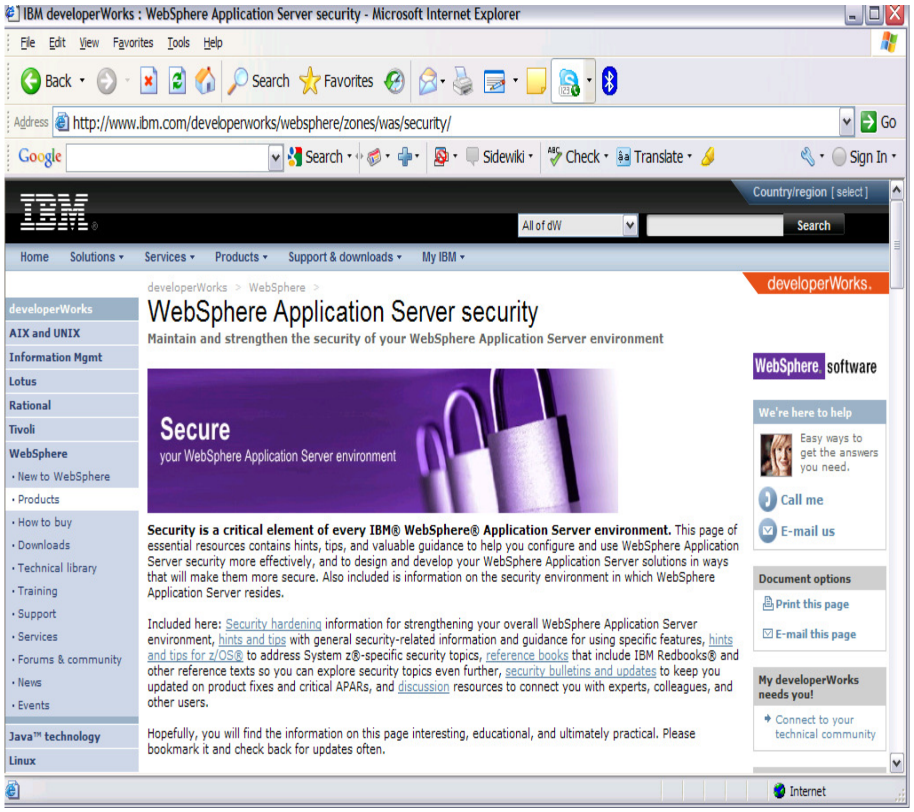
RACF for z/OS and WebSphere for Distributed Systems

- **Technology preview… IBM RACF Remote Authorization provider**
  - Available via the z/OS Download site
  - **http://www-03.ibm.com/systems/z/os/zos/downloads/**
    - Available to z/OS RACF licensed customers
- **Enables WebSphere authorization requests to be processed by z/OS RACF**
  - Centralized Audit and Authorization
- Utilizes WebSphere “plug points”
  - Java Authorization Contract for Containers (JACC) for Authorization
  - Trust Association Interceptor (TAI++)
    - “Pluggable" module whose responsibilities are:
      - *Validation of trust with the perimeter authentication service – such as the WebSeal reverse proxy*
      - *Extraction of credential information from the request*
        - *Subsequently used by authorization providers*

Provides ability to use RACF services to centralize access control policy and auditing on z/OS, while leveraging ITAMeb and WebSeal for authentication, edge of the network coarse grain access control and reverse proxy capabilities.

Visit Our Website

http://www.ibm.com/developerworks/websphere/zones/was/security/


IBM developerWorks : WebSphere Application Server security - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://www.ibm.com/developerworks/websphere/zones/was/security/ Go
Country/region [select]
All of dW Search
Home Solutions Services Products Support & downloads My IBM
developerWorks
AIX and UNIX
Information Mgmt
Lotus
Rational
Tivoli
WebSphere
New to WebSphere
Products
How to buy
Downloads
Technical library
Training
Support
Services
Forums & community
News
Events
Java™ technology
Linux
developerWorks > WebSphere >
WebSphere software
We're here to help
Easy ways to get the answers you need.
Call me
E-mail us
Document options
Print this page
E-mail this page
My developerWorks needs you!
Connect to your technical community
Secure
your WebSphere Application Server environment
Internet

WebSphere Application Server security

Maintain and strengthen the security of your WebSphere Application Server environment

Security is a critical element of every IBM® WebSphere® Application Server environment. This page of essential resources contains hints, tips, and valuable guidance to help you configure and use WebSphere Application Server security more effectively, and to design and develop your WebSphere Application Server solutions in ways that will make them more secure. Also included is information on the security environment in which WebSphere Application Server resides.

Included here: Security hardening information for strengthening your overall WebSphere Application Server environment, hints and tips with general security-related information and guidance for using specific features, hints and tips for z/OS® to address System z®-specific security topics, reference books that include IBM Redbooks® and other reference texts so you can explore security topics even further, security bulletins and updates to keep you updated on product fixes and critical APARs, and discussion resources to connect you with experts, colleagues, and other users.

Hopefully, you will find the information on this page interesting, educational, and ultimately practical. Please bookmark it and check back for updates often.

- How to harden your environment
- Hints and Tips
- FAQ
- Reference Material
- Security Bulletin
- Blog and discussion

Thank
You

धन्यवाद
Hindi
多謝
Traditional Chinese
ขอบคุณ
Thai
Спасибо
Russian
Gracias
Spanish
Merci
French
Thank You
English
شكراً
Arabic
Obrigado
Brazilian Portuguese
Danke
German
多谢
Simplified Chinese
Grazie
Italian
நன்றி
Tamil
ありがとうございました
Japanese
Teşekkürler
turkish
감사합니다
Korean