Security Application Architecture Development and Integration Overview

Bill O'Donnell
IBM Corporation
Lead WebSphere Security Architect

August 5, 2010

Agenda

- Design and Architecture
- Web Based Applications Authentication and Authorization
- EJB Applications Authentication and Authorization
- Web Services
- Additional Security Features
- Securing DB2

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

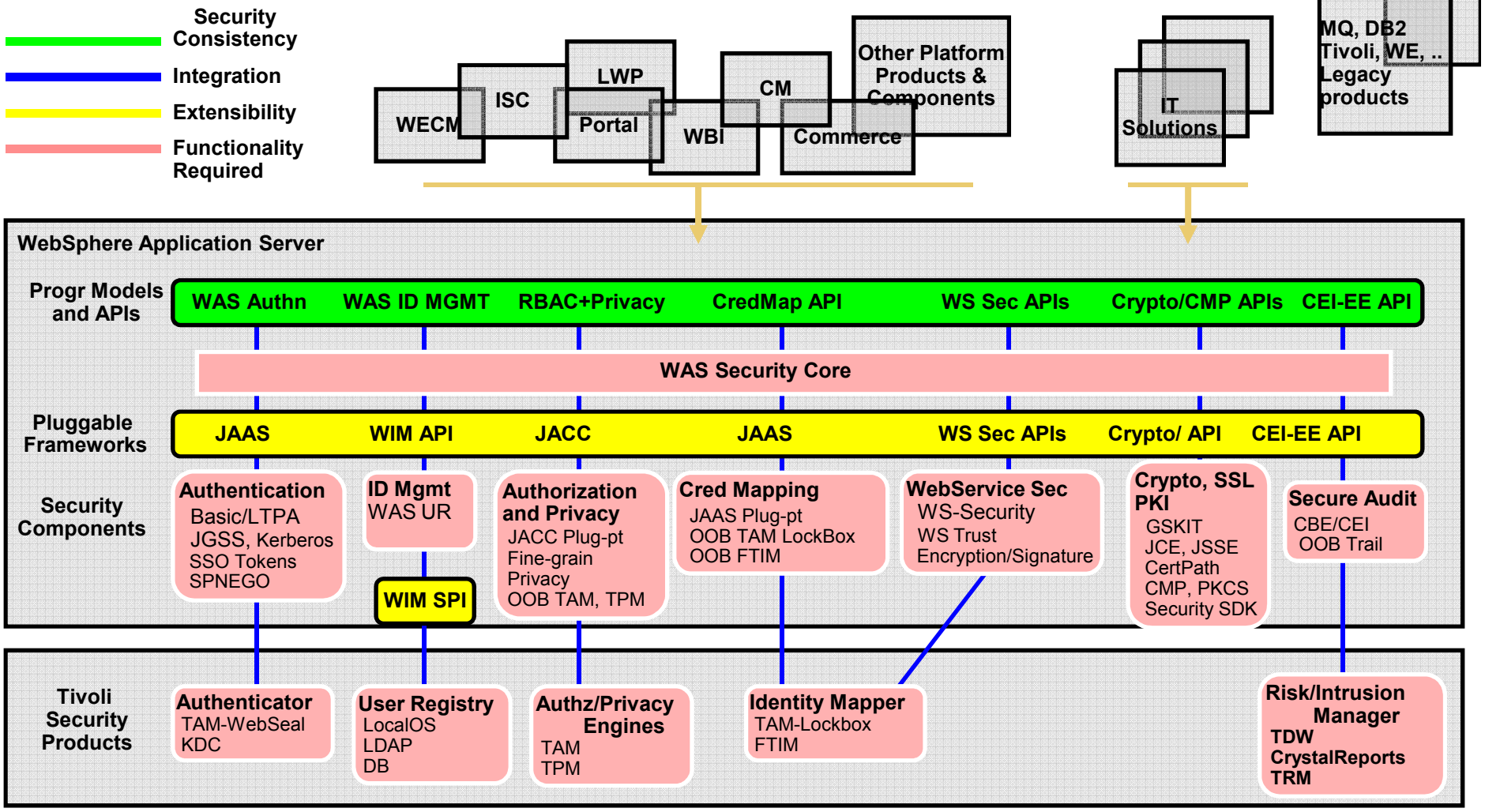
Note: All the features discussed apply to all platforms WebSphere Support. Any z/OS specific features will be noted.

1074 – Security Application Architecture Development and Integration Overview

Design Architecture

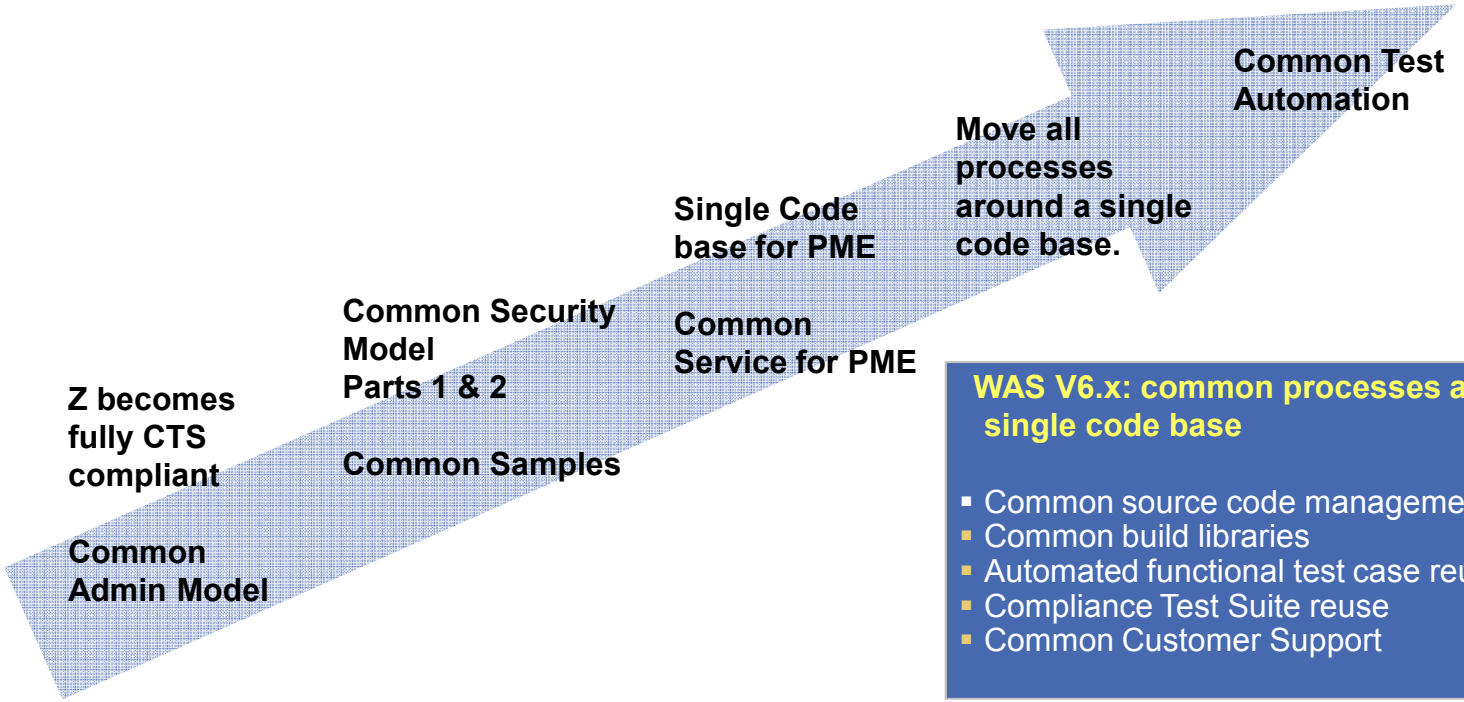
SHARE in Boston

Security Architecture

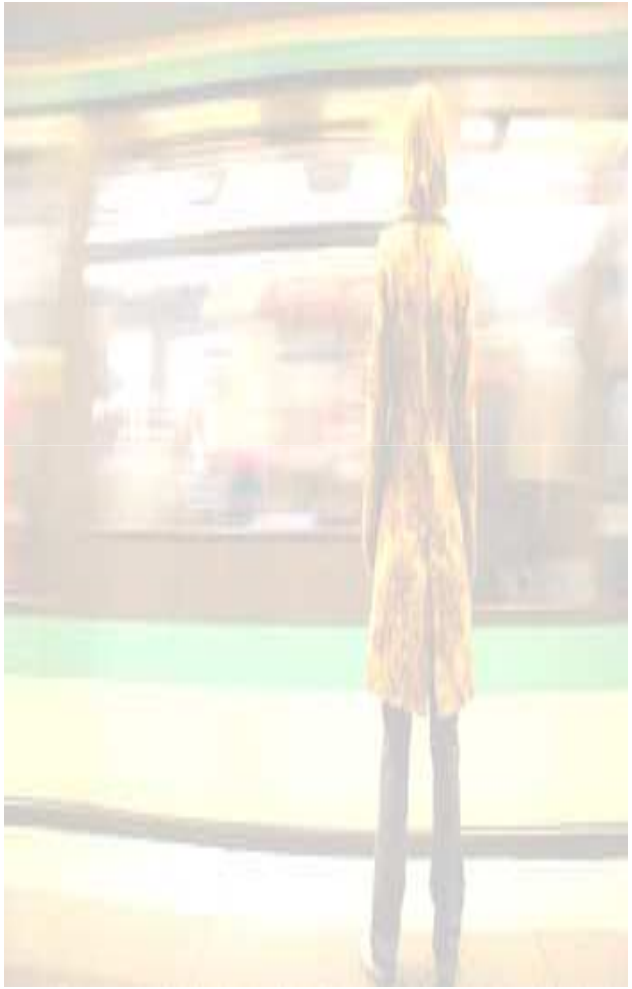


Unifying the WAS Code Base

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

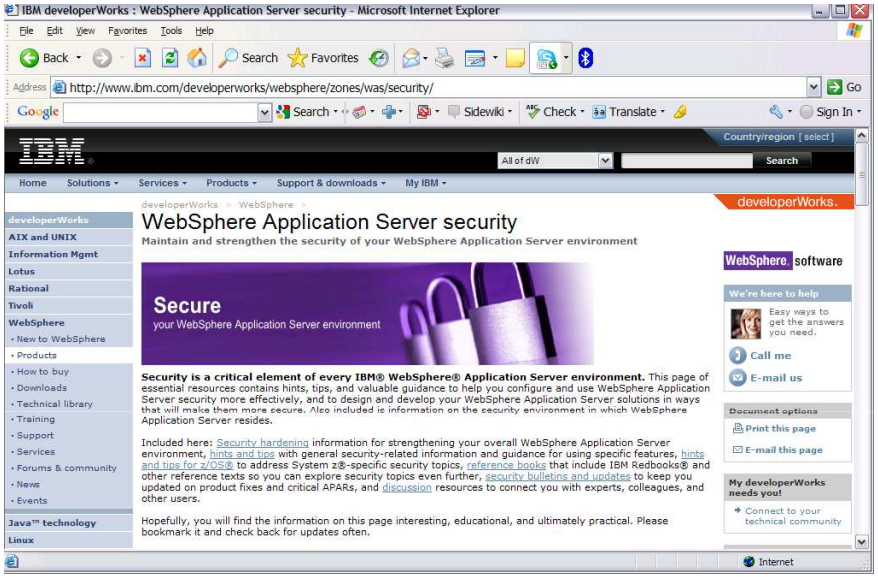


WebSphere Security Principles



- **Secure by Default**
  - Starting with WAS V6.1, by design, we are secure out of the box.
  - WAS V7, additional defaults were change
- **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

WebSphere Application Server Secure by Default



- Since WAS V6.1, is secured by default
  - Security is enabled.
  - Use WebSphere options that are considered most secure.
  - Additional configuration may be required to meet your business requirements

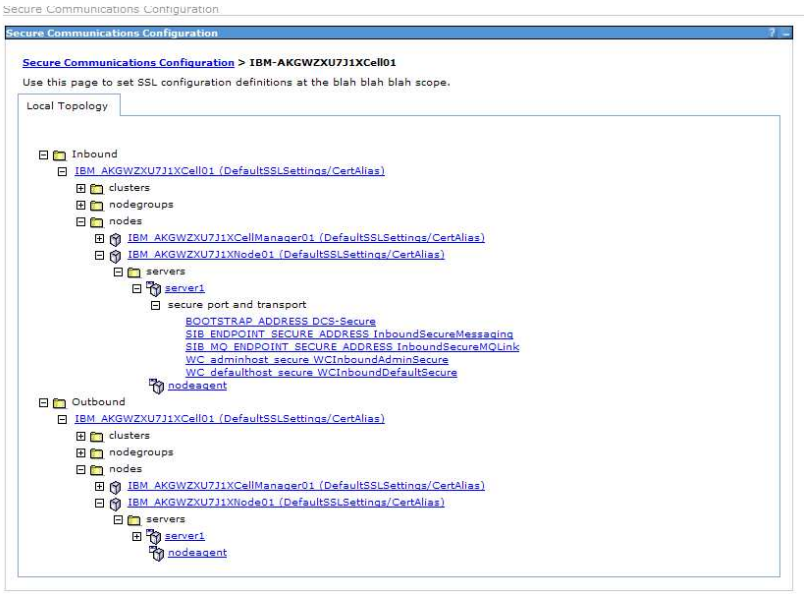
- WebSphere Application Server Web Site
  - We publish and maintain a detailed Security hardening documentation
    - Consideration for a secure environment for WebSphere
    - Consideration in properly securing your Applications
    - Post any additional WAS options that should be considered.
  - http://www.ibm.com/developerworks/websphere/zones/was/security/.

Ease of Use

- **Administrative Security enabled out of the box**
  - Prior to WAS61, Security enablement resulted in both Administration Security and Application Security enabled
  - Administration Security and Application Security enablement is now separate
  - Administrative Security is enabled out of the box to properly secure WAS Server environment
- **Simplified Security Configuration and Administration**
  - Simplified administrative console security panels
  - New Security enablement wizard
  - Security Configuration reporting Tool
  - WebSphere Application Server V7 allows for multiple security configuration within a Cell.
- **Automatically generating server IDs**
  - You no longer need to specify a server user ID and password during security configuration, unless using a mixed cell environment

Ease of Use (Continue)

- **Simplified WebSphere key and Certificate Management**
  - Allow you to use the key management tools from the console
  - Make it easier to configure SSL
  - Manage Web server and plug-in certificates from the console
  - Use the Trust Manager to automatically trust host or signers
  - Make it easier to refresh an expiring certificate
  - Certificates Expiration Monitor
  - Easy utilities to handle Certificate exchange
- **Federated Repository (VMM)**
  - Common identity management programming interface
  - User Identity profile and relationship management
  - Multiple changing of registry
- **Signal Sign On (SSO) options**
  - SPNEGO and Kerberos
    - Be sure to see our red book http://www.redbooks.ibm.com/abstracts/sg247771.html?Open
  - SAML for Web Servers WAS 7.0.0.7 and above
  - LTPA

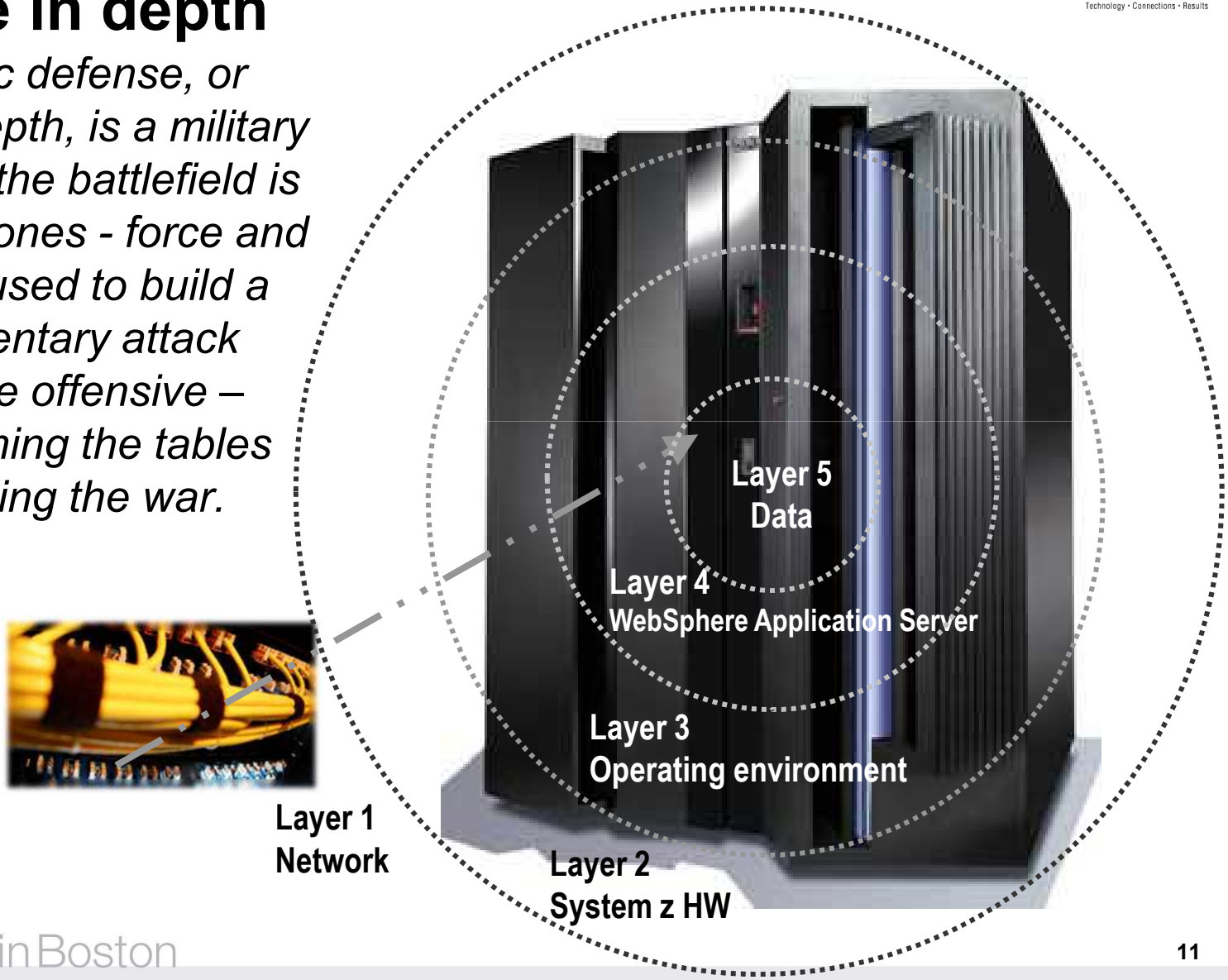


Ease of Use (Continue)

- **Common Criteria Assurance Level 4 (EAL4)**
  - Certification that provides an independent assessment, analysis, and Testing in providing customers the confidence that a given product will be effective in delivering key security functions.
  - WebSphere Application Server V7 was designed to meet or exceed the security capabilities of WAS V6.1 including the EAL4 requirements.
    - **But,** *the US CCEVS is no longer certifying software products as Common Criteria EAL compliant because they are moving to a new security standard referred to as "Protection Profiles". The "Protection Profiles" requirements for Middleware software have been made available.*
- **Fully FIPS Compliant**
  - Supports the Federal Information Processing Standards(FIPS) 140-2 Government standards.

Defense in depth

The elastic defense, or defense in depth, is a military tactic where the battlefield is broken into zones - force and space are used to build a complementary attack against the offensive – thereby turning the tables and winning the war.



z/OS for example defense in action

Let's start with …

Zone 1: The **Network**

- z/OS Communications Server with integrated intrusion detection capabilities

Zone 2: The **System z9** server

- EAL 5 Common Criteria certified LPARs
- System z cryptographic technology featuring the Crypto Express2
- HiperSockets security benefits compared to communication alternatives like TCP/IP
- Storage Protection Keys can granularly protect address spaces
- Coming soon: IP Security with zIIPs

Zone 3: The **z/OS** operating environment

- EAL 4+ Common Criteria certified multi-level security
- Public Key Infrastructure (PKI) services

Zone 3 cont'd: The **z/OS** operating environment

- Centralized system level auditing to know who is trying to access your system and when
- Resource Access Control Facility (RACF) integrates with z/OS to provide centralized enterprise level security & auditing up through the software stack
- EAL4+ for Controlled Access Protection Profile (CAPP) and Labeled Security Protection Profile (LSPP).
- Tivoli support for z/OS security, including access, identity, compliance, et al

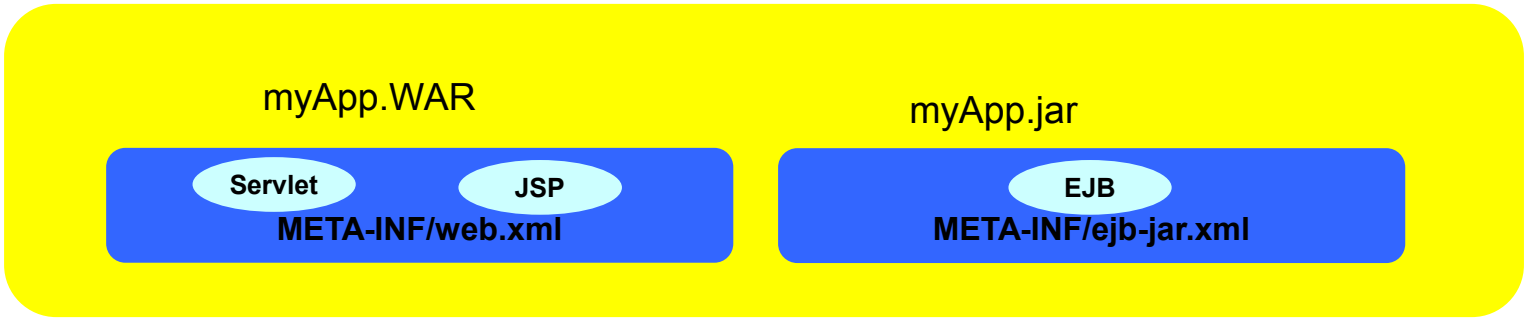
Zone 4: **WebSphere Application Server**

- Java 2 Platform Enterprise Edition (J2EE) security
- EAL4 Common Criteria certified
- Federal Information Processing Standards (FIPS) 140-2 Compliance

Why J2EE Security Model is important

- Defense in Depth calls for all layers to be secure. Too often, the application layers is ignored.
- J2EE Security Model is “very” abstract and 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 Model



- The **web.xml** is the deployment descriptor for the Web modules for the application and contains the security-constraints.
- The **ejb-jar.xml** is the deployment descriptor for the EJB modules for the application and contains the security method-permissions.

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.

Administrative Privilege

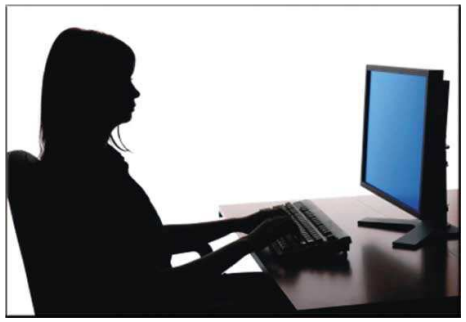
- WAS Administration offers a separation of privilege model in which multiple roles with different administration capabilities.
- In addition, WebSphere support different permissions at finer grained level of resources
  - Node, node group, server, cluster, application
- Authorization groups control 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

1074 – Security Application Architecture Development and Integration Overview

Web Based Application
Authentication and Authorization

SHARE in Boston

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

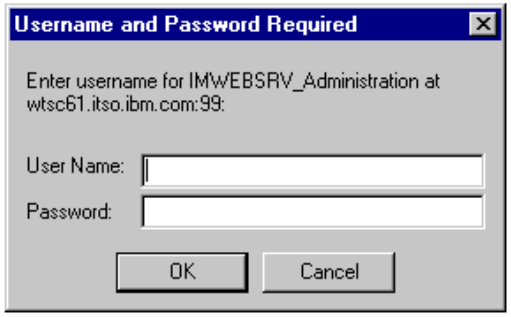
1. User clicks on link to protected page

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

2. Server checks authority and rejects request

Response: Status 401
Realm "IMWEBSRV_Administration"

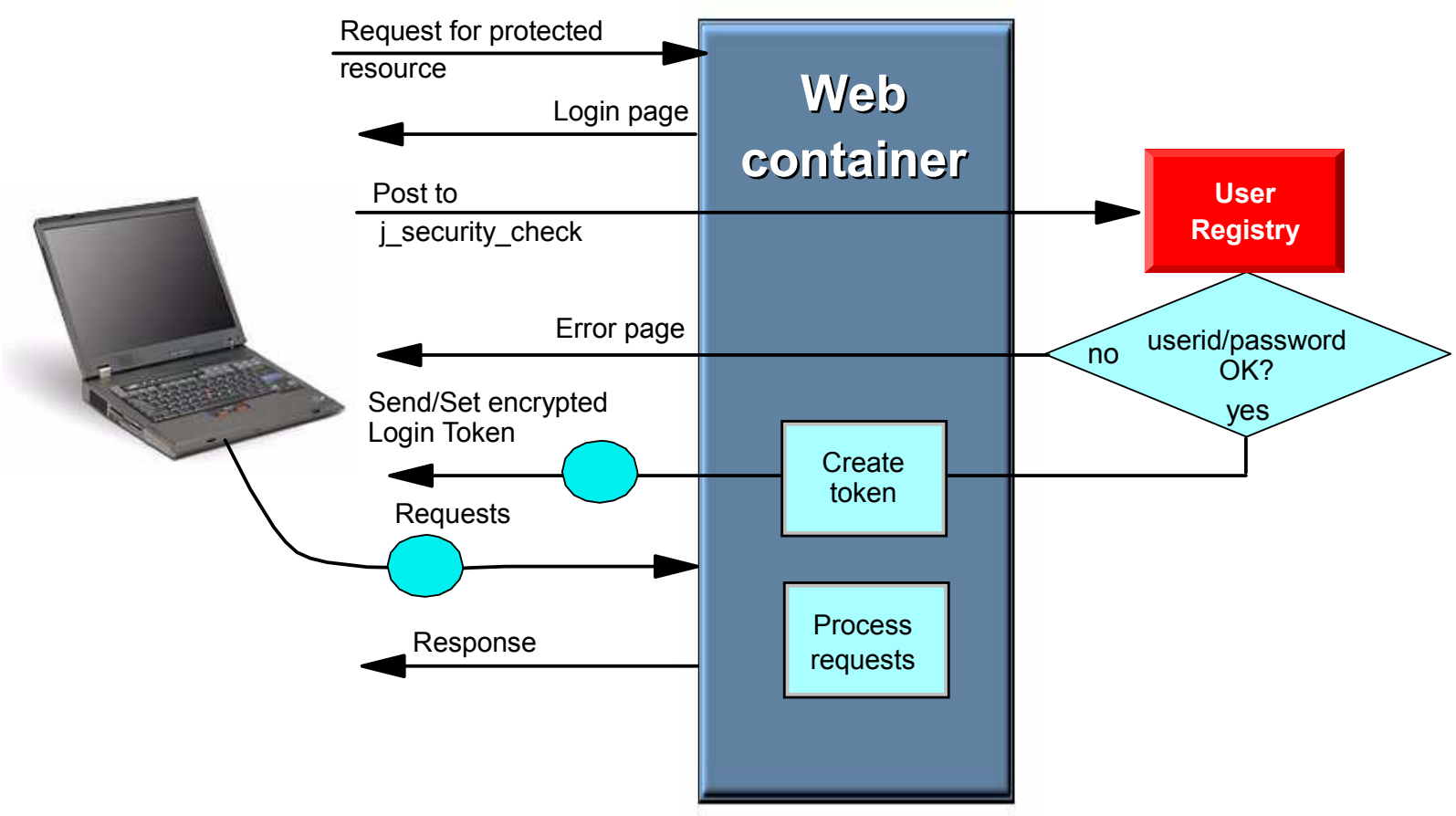
3. Browser pop-up window prompts user for userId and password



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

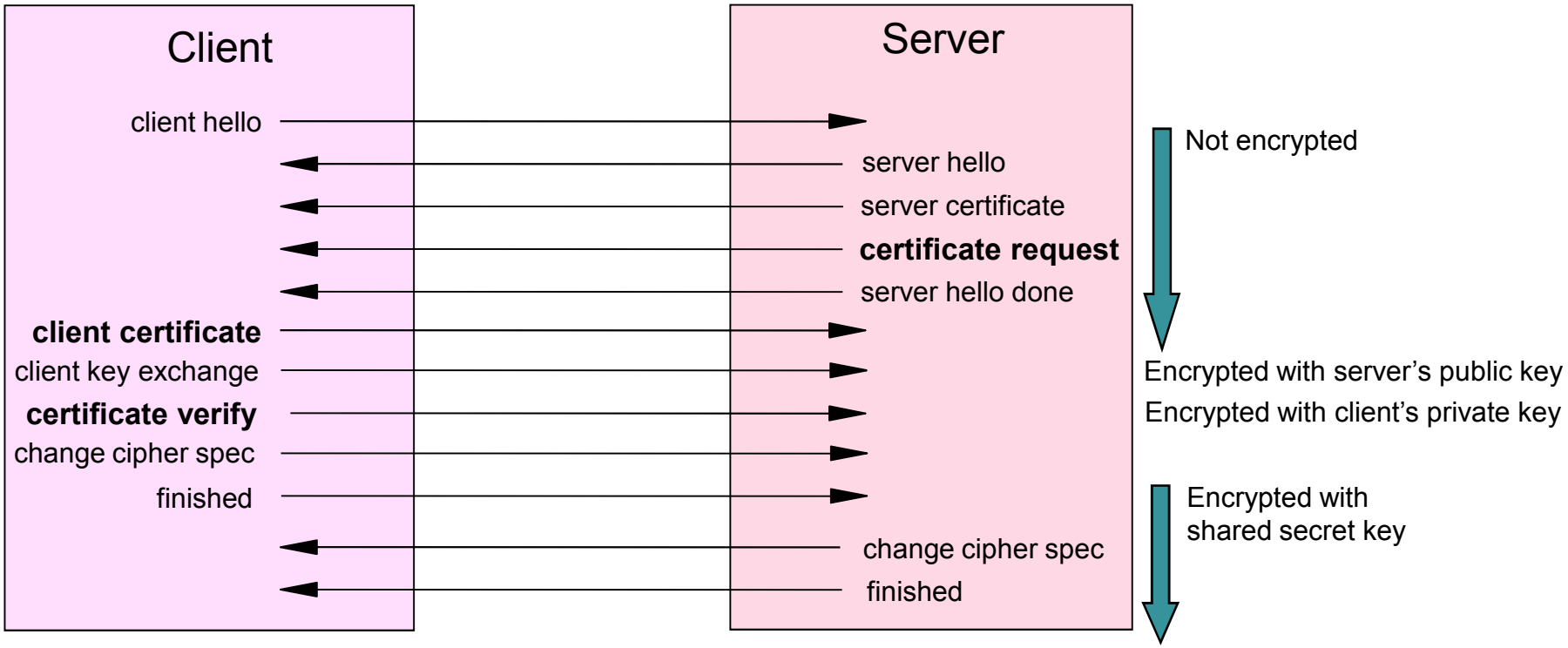
Request: GET http://server/restricted.html

Form-based login



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

Certificate-based Authentication



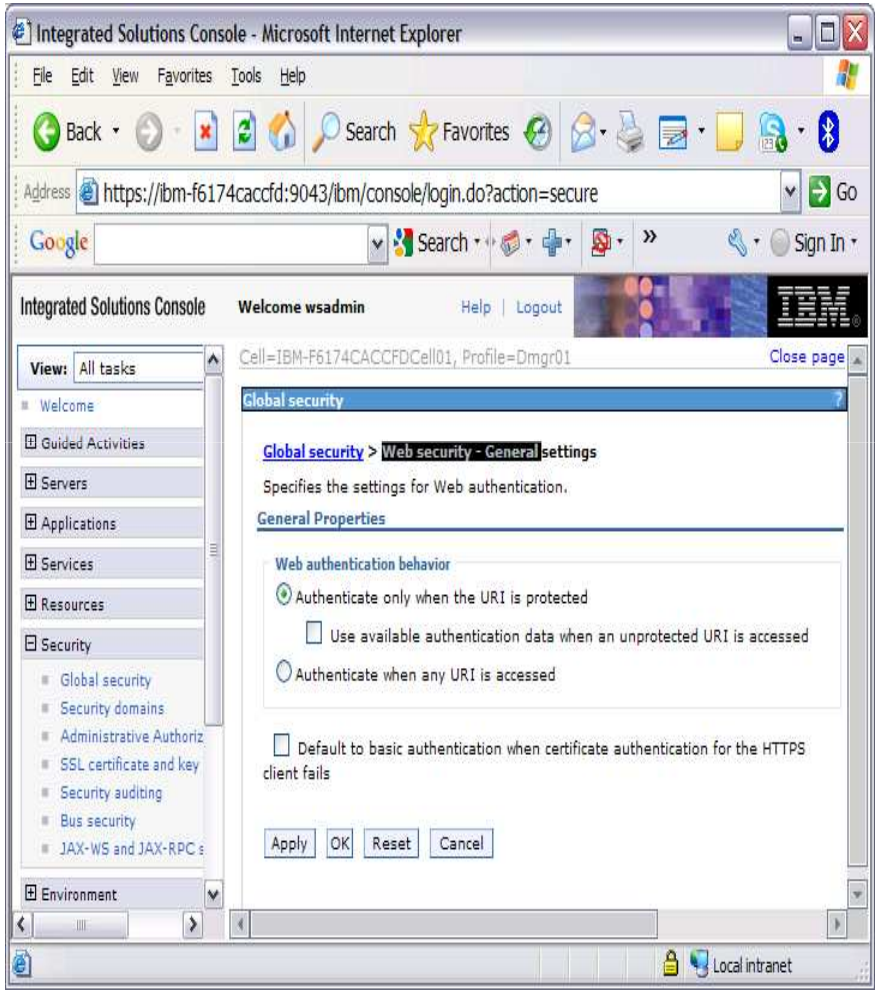
J2EE Security
Web Based Applications

- 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.

Web Security General Settings



- **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

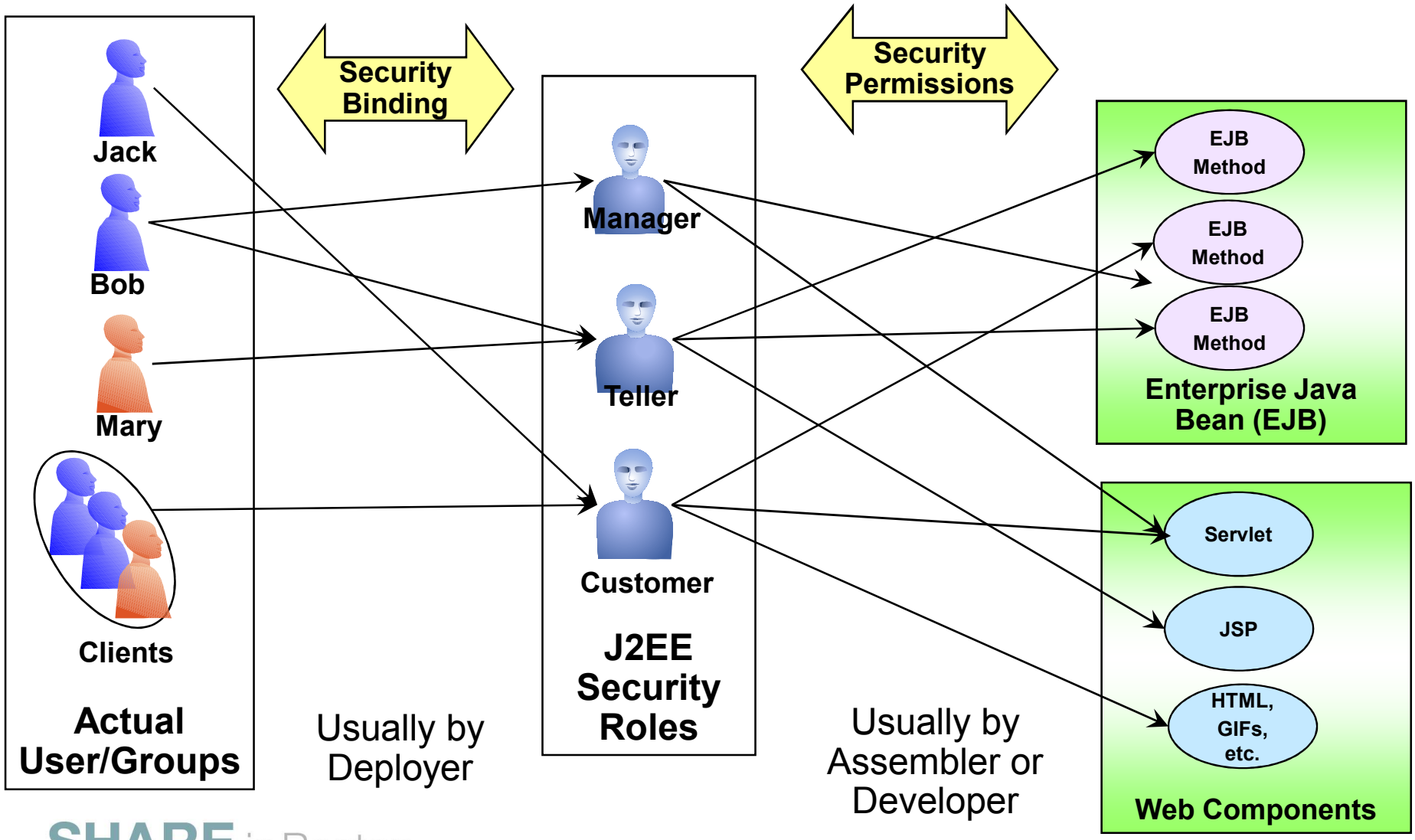
Customer

Teller

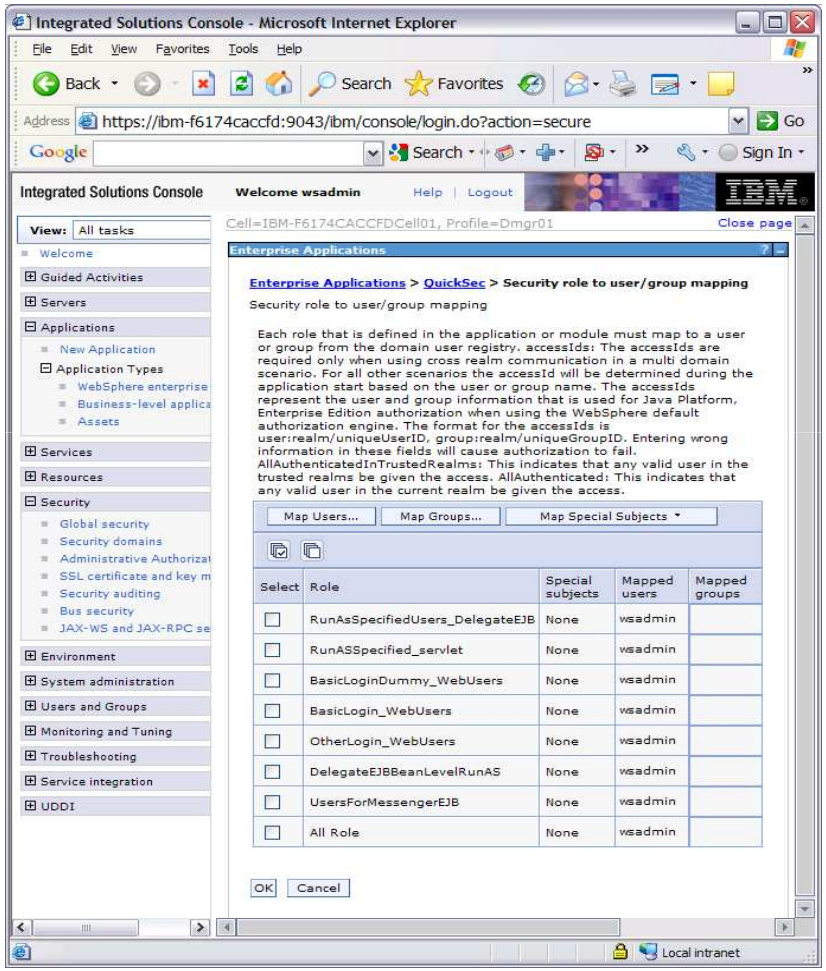
Manager

- **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

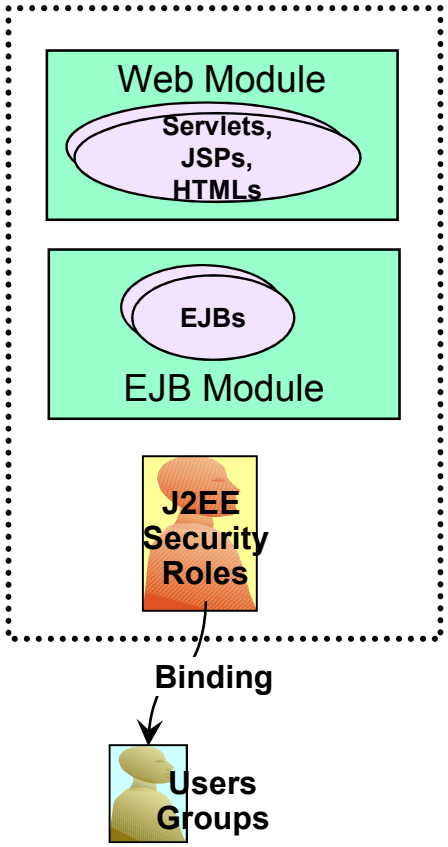


J2EE Security Role mapping



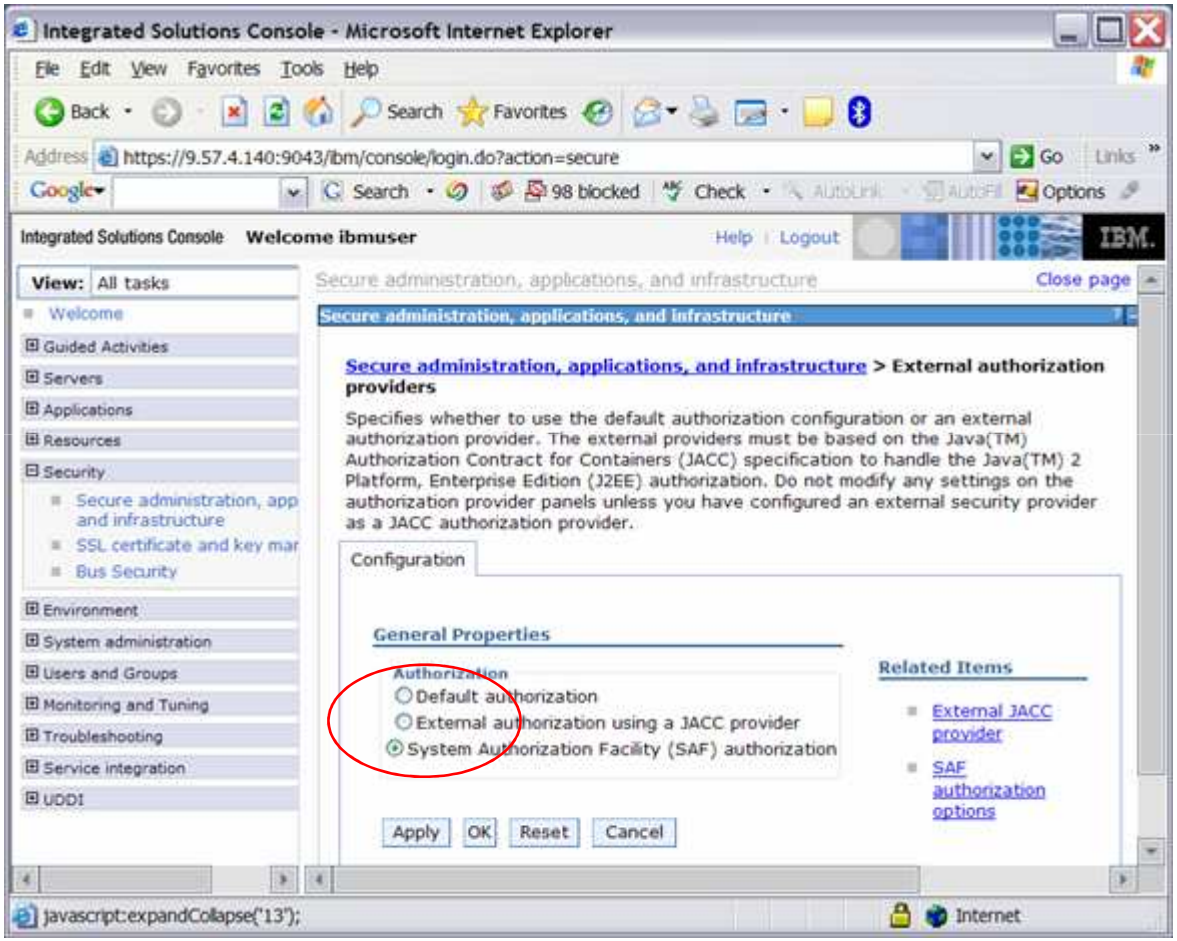
- J2EE roles are mapped to either user(s) or group(s) or both.
- Can be mapped during application deployment or changed after deployment.
- During the mapping process, WebSphere allows user and group patter searches against the configured User Account Repository (User Registry).
- If using zOS SAF authorization, these mappings are handled within RACF or equivalent product.

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)**

SAF Authorization can be Enabled using GUI



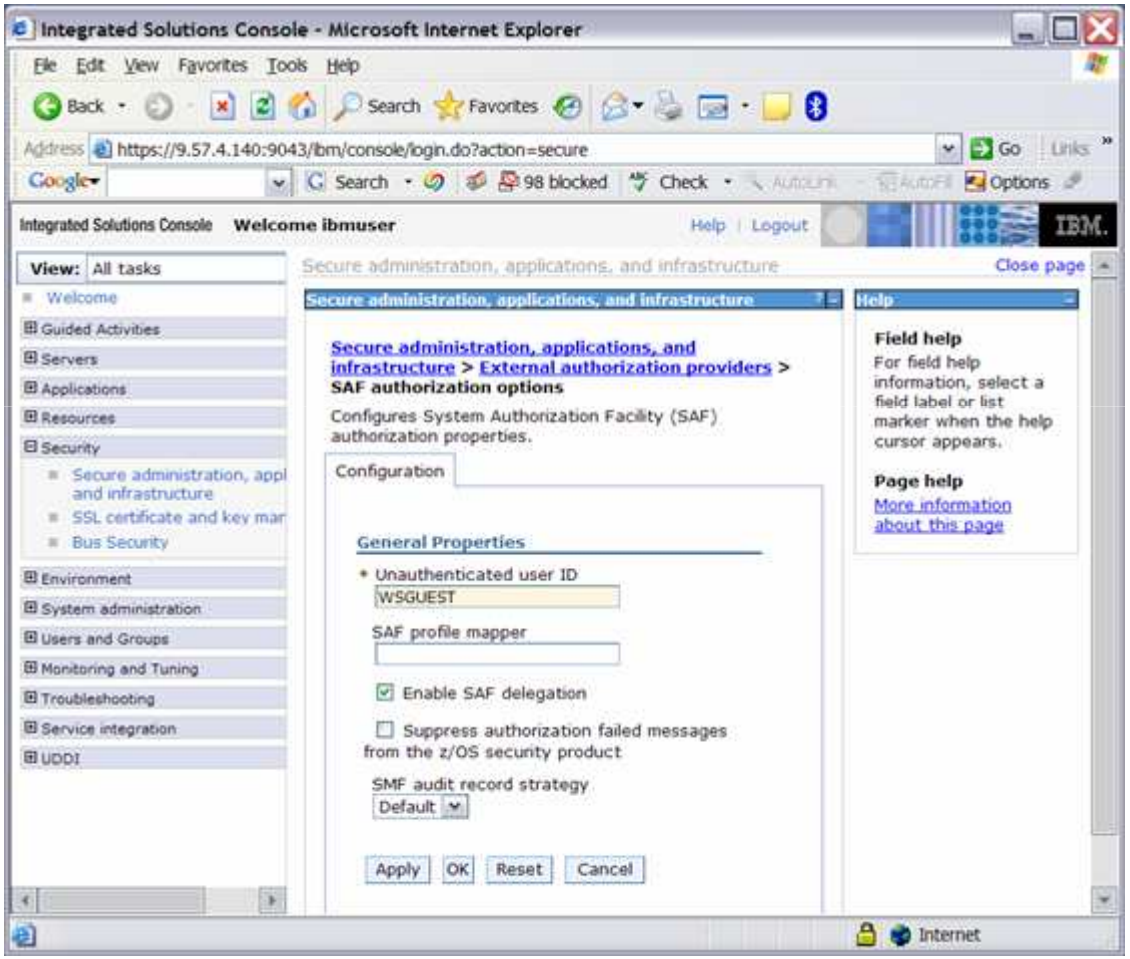
- From the main Security panel >> External Authorization Provider panel, you have the options for
  - WebSphere Authorization
  - JACC Provider
  - SAF authorization
- Note, on this screen, you have a link for SAF authorization options.

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)

WAS for z/OS
Specific GUI enablement's

- On the SAF Authorization Options you have the ability to:
  - Specify the unauthenticated User ID.
  - Specify a SAF profile mapper.
  - Enable SAF Delegation.
  - SAF logging options
  - Specify SAF Prefix (WAS7)



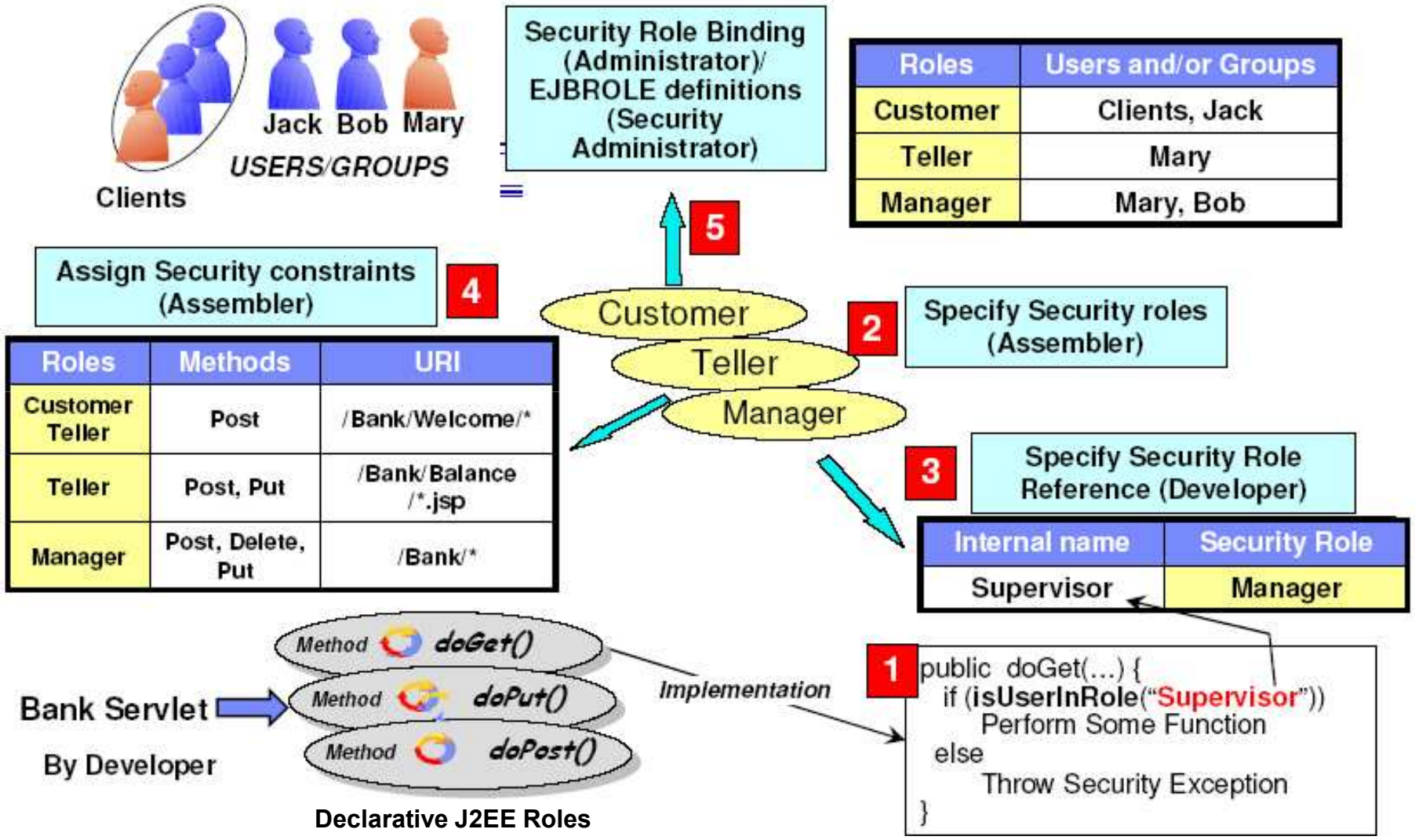
Web Applications Programmatic APIs

- **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: Servlet, JSP Role Based Authorization



Declarative J2EE Roles

Programmatic J2EE Roles

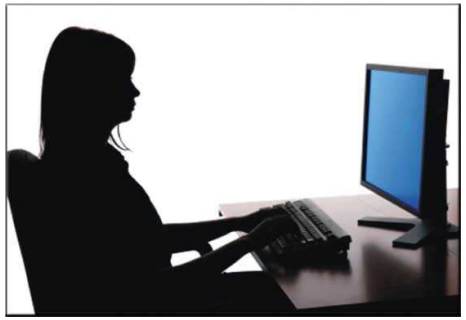
Application Security Tasks and Roles

	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

1074 – Security Application Architecture Development and Integration Overview

EJB Application
Authentication and Authorization

J2EE EJB Authentication



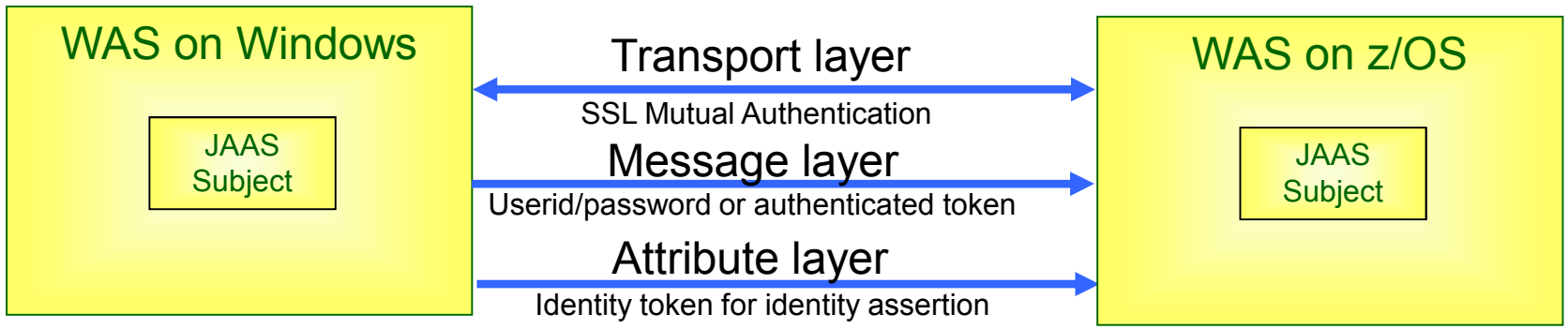
- Similar to Web Applications, the WAS EJB Container is responsible for the full aspects for Authentication.

- **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

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	



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>
```

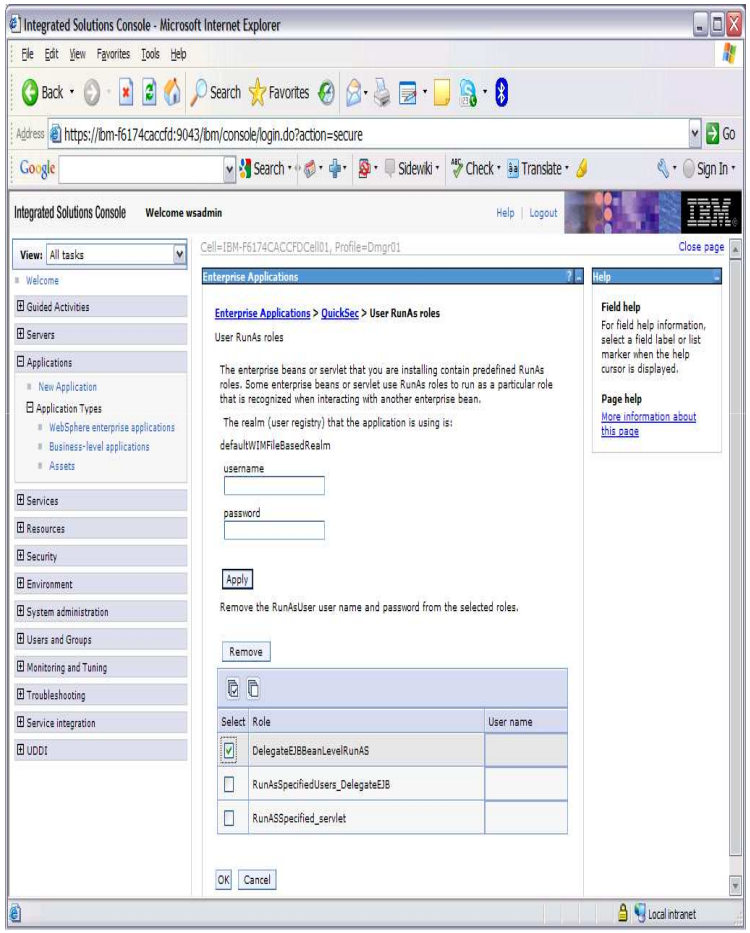
- Define **Security Roles**
  - List all the security roles that will be used by this application
  - Must include roles that were listed in the Authenticated Constraint plus any programmatic roles.
- Define **Security Identity**
  - Specifies the security identity to be used to invoke methods in a particular 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*

Changing Identity: “Run-As” Option

- EJB methods have the ability to run using different identity
  - There are several different “Run-As” identities that you can choose from
  - Run-As specification applies to all the methods of the EJB
  - IBM extension, allow for specify “Run-As” options for different methods within the same EJB

Run As Options	Description
Caller	Run as the identity of the caller subject
Specific Identity	Run as a the specified users
Role	Run as the role that mapped to a specified Identity
Server Identity	Run as the Server Identity **(but be careful Authorization using Server Autogen ID not supported)**

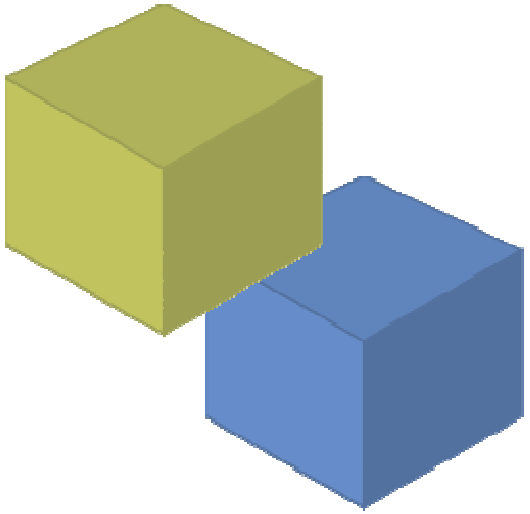
WebSphere Role Delegation

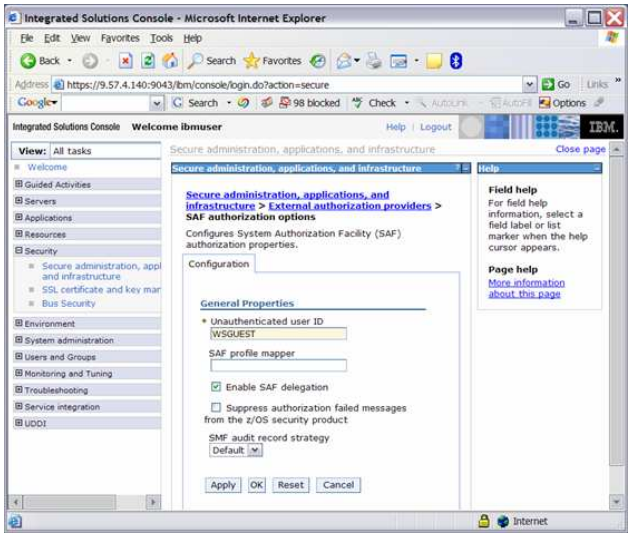


- WebSphere offers a Delegation to User Identity be represented as a J2EE role.
- For example, an application can be established to run with RunAs Role of *rolea*. *rolea* can be mapped as *Usera*. In this case, WebSphere will setup the identity context as *Usera*. RunAs Role is defined in the Deployment Descriptor.
- For example
  - The Deployer will deploy *app1* and set “RunAS role” to *role1*.
  - In the WebSphere Console, you can map the rule to a specific userID and password.

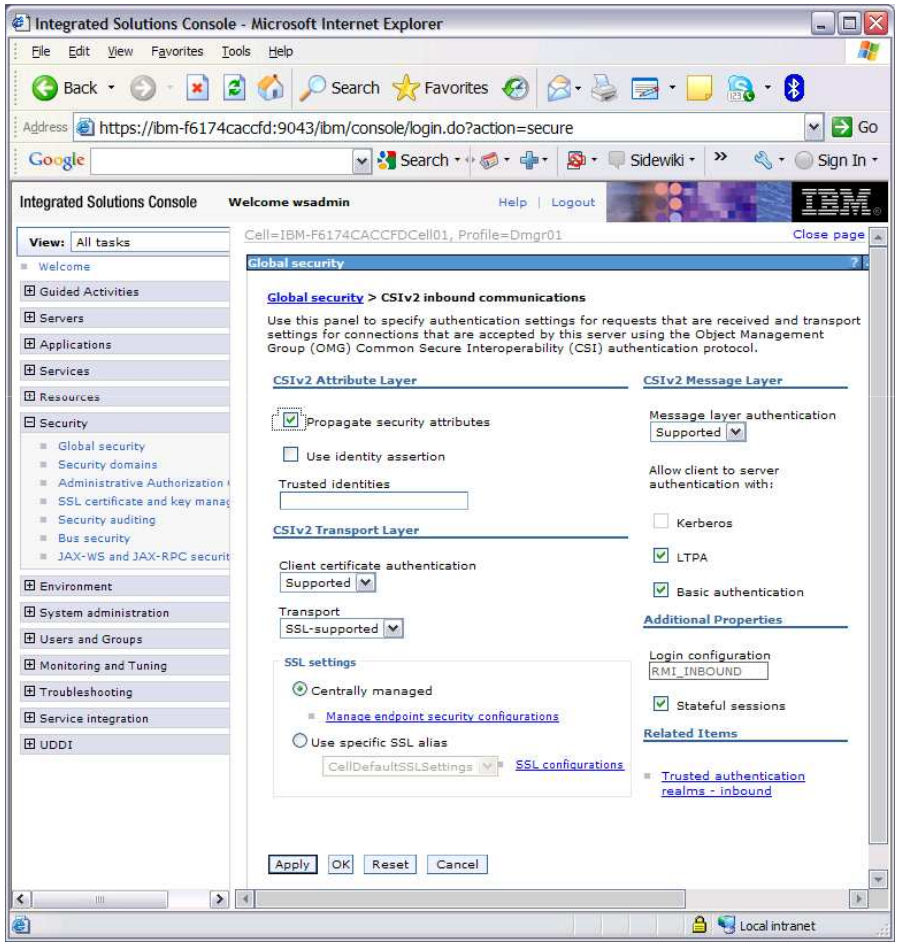
WAS for z/OS SAF Delegation

- SAF Delegation will use the specified J2EE role name to determine the Thread Identity and will synchronize with the Userid specified in the SAF EJBROLE profile's APPLDATA.
- Basically this is similar to the rest of the WebSphere family except the Userid will be obtained from SAF. .
- For example, the customer would define to SAF using the RACF command RDEFINE EJBROLE myRole UACC(NONE) APPLDATA('myUserID') ID specified must be a valid SAF ID.
- Inherently, this requires SAF Authorization to be enabled.
- Security configuration must have SAF Delegation enabled.
  - GUI – Secure administration, applications, and infrastructure > External authorization providers > SAF authorization options > SAF Delegation.



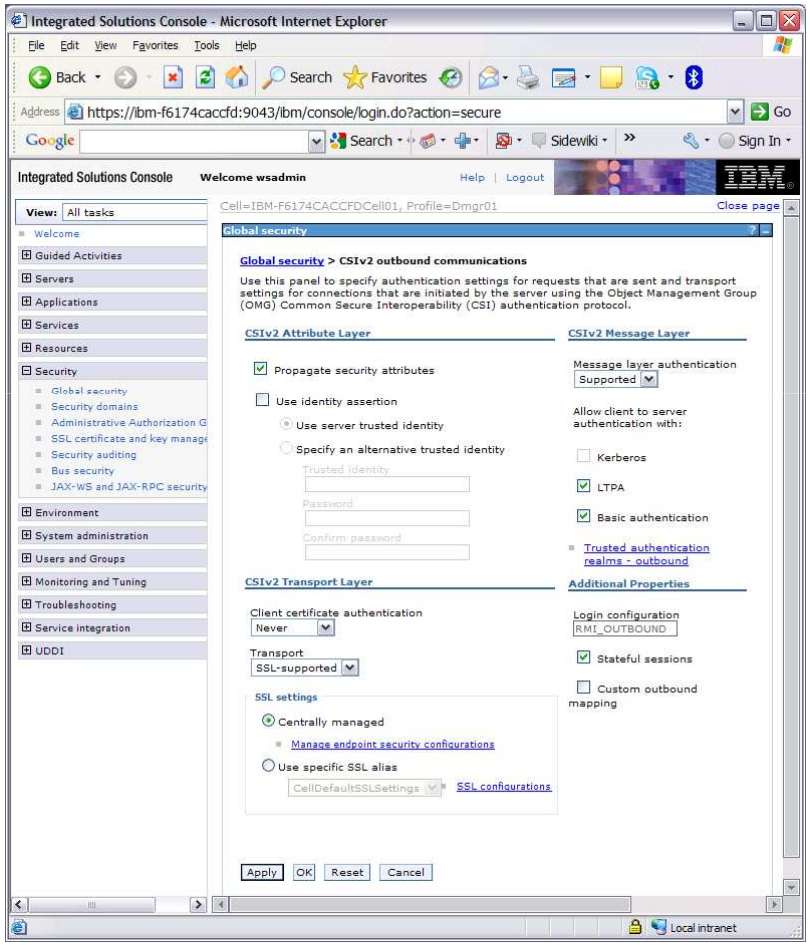


EJB Leverages RMI/IIOP Security using CSIV2 – inbound communications



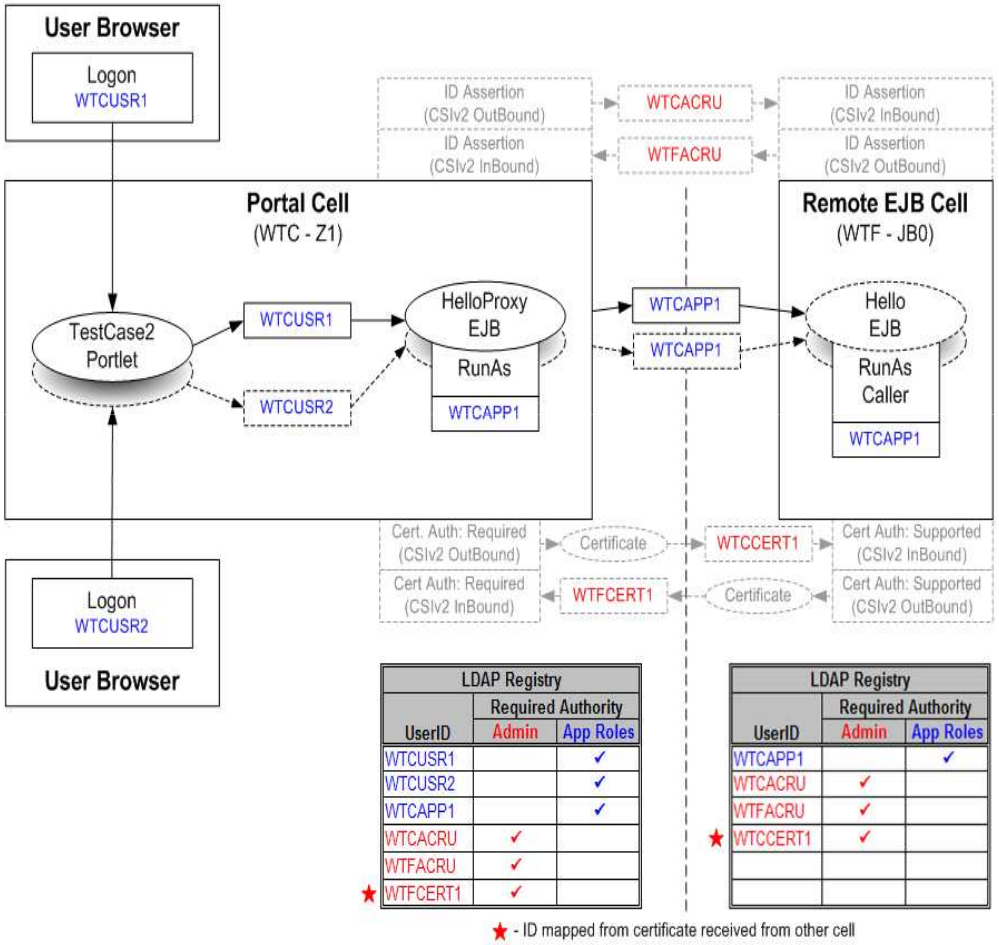
- **Identity Assertion** – When enabled, the server permits an identity that was asserted from an upstream server. It requires the Trusted Identities to contain upstream serverID that you trust to assert.
- **Message Layer authentication** – Specifies if authentication is required, supported (optional), 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



- **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.

Example:
Local EJB on Portal runAS Role
Remote EJB on WAS runAs Caller



- Portal CSIv2 config
  - Inbound/Outbound
    - Basic Authentication none
    - Client Certificate Required
    - Identity Assertion enabled
    - SSL required
- Remote WAS
  - Inbound/Outbound
    - Basic Authentication none
    - Client Certificate Required
    - Identity Assertion enabled
    - SSL required
- Results
  - Logon to Portal using WTCUSER1
  - Local EJB will run as WTCAPP1
  - Remote EJB will also run as WTCAPP1

J2EE Authorization Basics

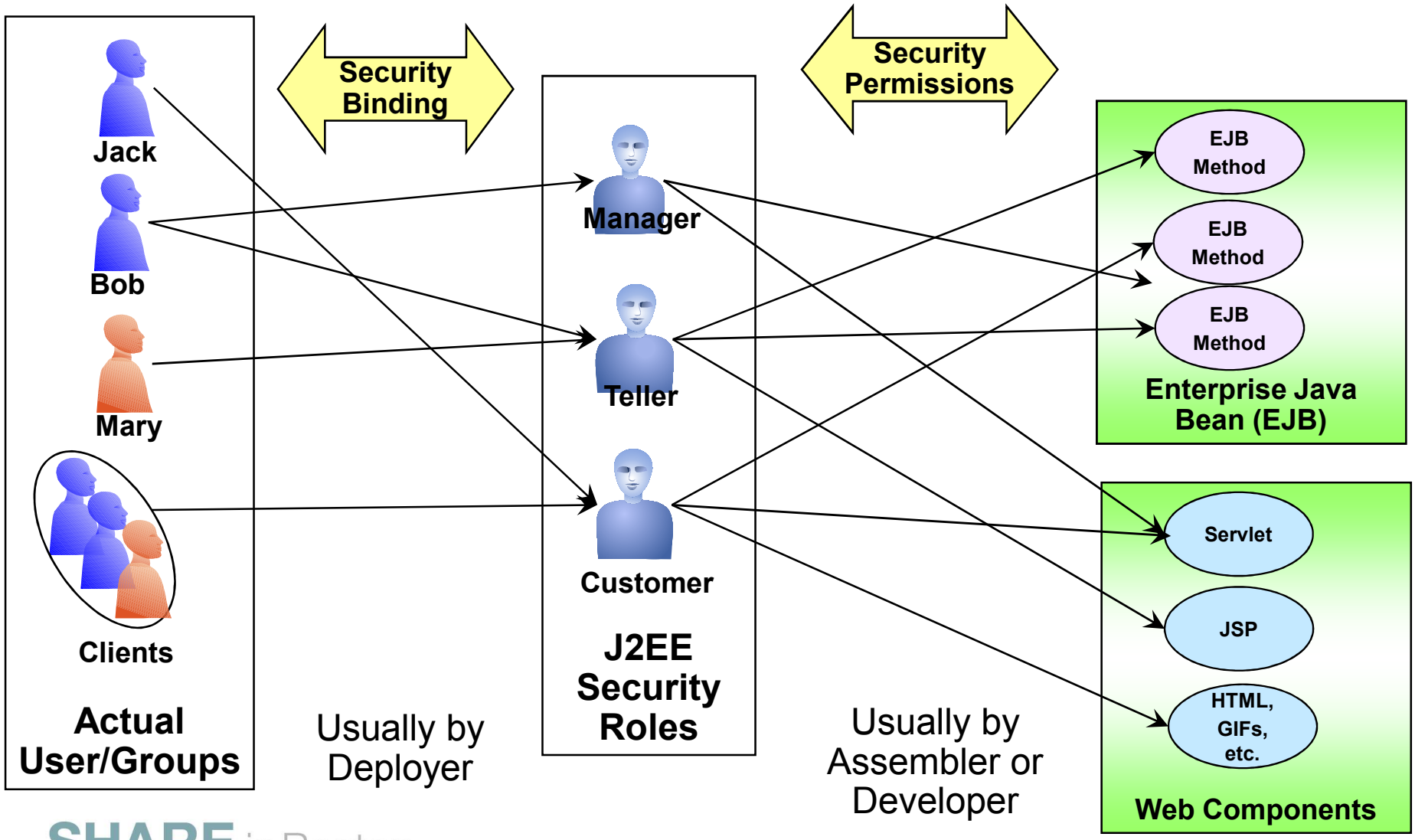
Customer

Teller

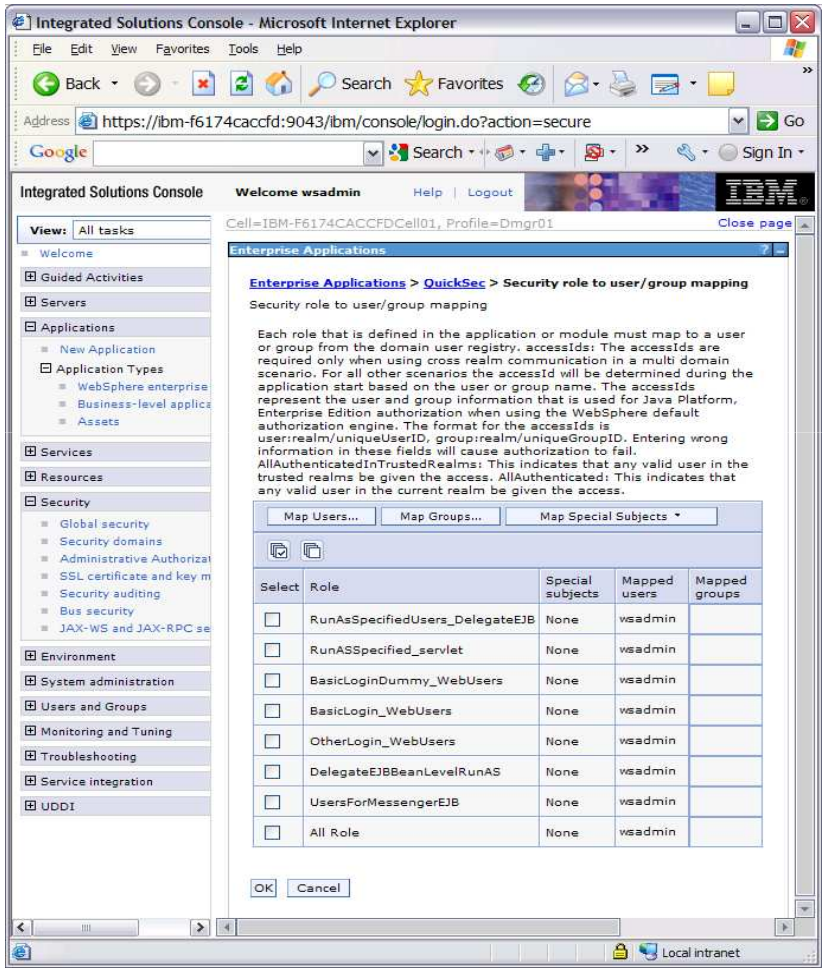
Manager

- **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



J2EE Security Role mapping



- J2EE roles are mapped to either user or group or both.
- Can be mapped during application deployment or changed after deployment.
- During the mapping process, WebSphere allows user and group patter searches against the configured User Account Repository (User Registry).
- As discussed before, WAS for z/OS customer has the option to use SAF Authorization instead.

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

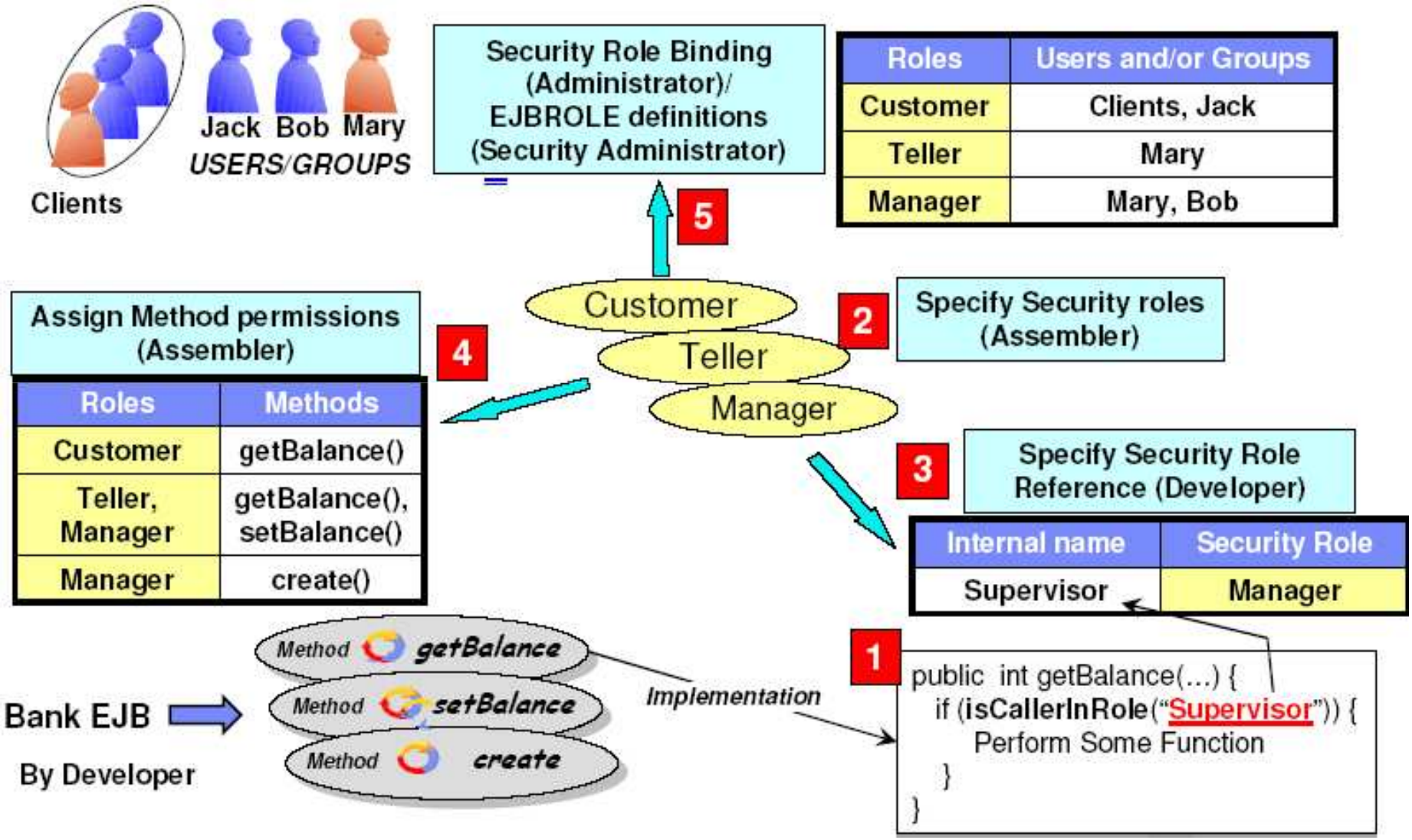
Example:

```
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.

J2EE Security: EJB Role Based Authorization



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-ejb-bnd.xmi
4	Specify the declarative J2EE roles	Assembler	RAD, AST	ejb-jar.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

1074 – Security Application Architecture Development and Integration Overview

Java 2 Security

Java 2 Security

- Java 2 security provides code level access control
  - Can code call “this” Java API
  - Only provides code authorization, no user authorization to code.
- Do you “really” need Java 2 Security?
  - Most likely not
    - Performance penalty
      - *High performance penalty prior to WAS6.1*
      - *WAS 6.1.0.9 and above, performance improved a lot* when the read only subject is enabled*, but the more program permissions added to logic, the worse the performance.*
    - Very difficult to manage
    - Provides no value in improvement of security with respect to external attacks.
- Recommendation
  - Most customers do not use Java 2 Security
  - Do not use unless you “really” need it

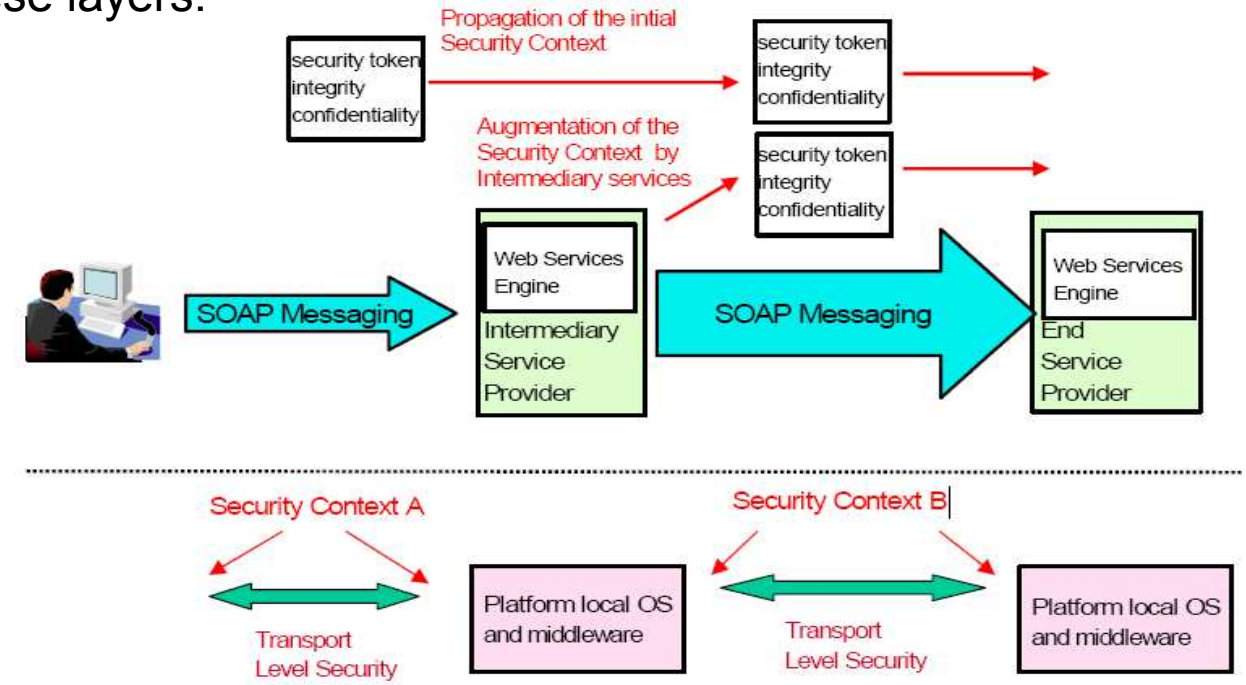
1074 – Security Application Architecture Development and Integration Overview

Web Services Application
Authentication

SHARE in Boston

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.

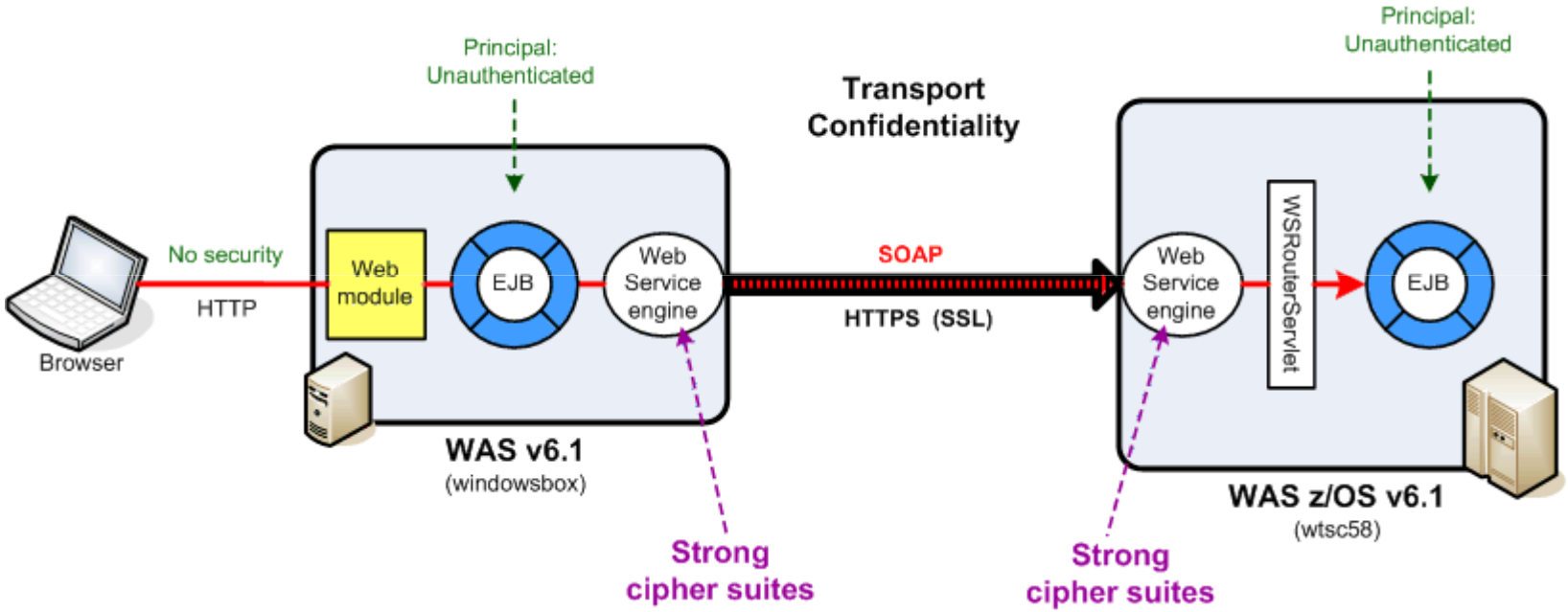


- **The SOAP or Message layer**: The WS-Security specifications indicate how SOAP XML messages can carry security assertions and contexts.

Web Services Transport layer security

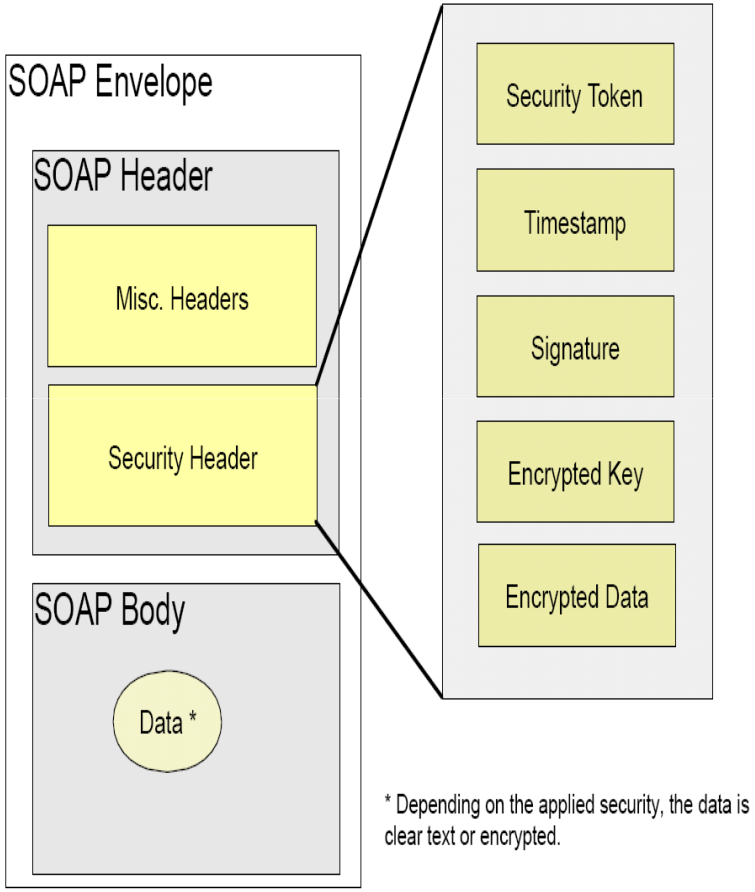
- SSL is the most popular way to encrypt communication between business partners over the Internet.
- It simply creates a secure pipeline between two nodes and encrypts all traffic flowing between the nodes.
  - SSL provides a straightforward way to provide **confidentiality**.
  - It also includes a 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

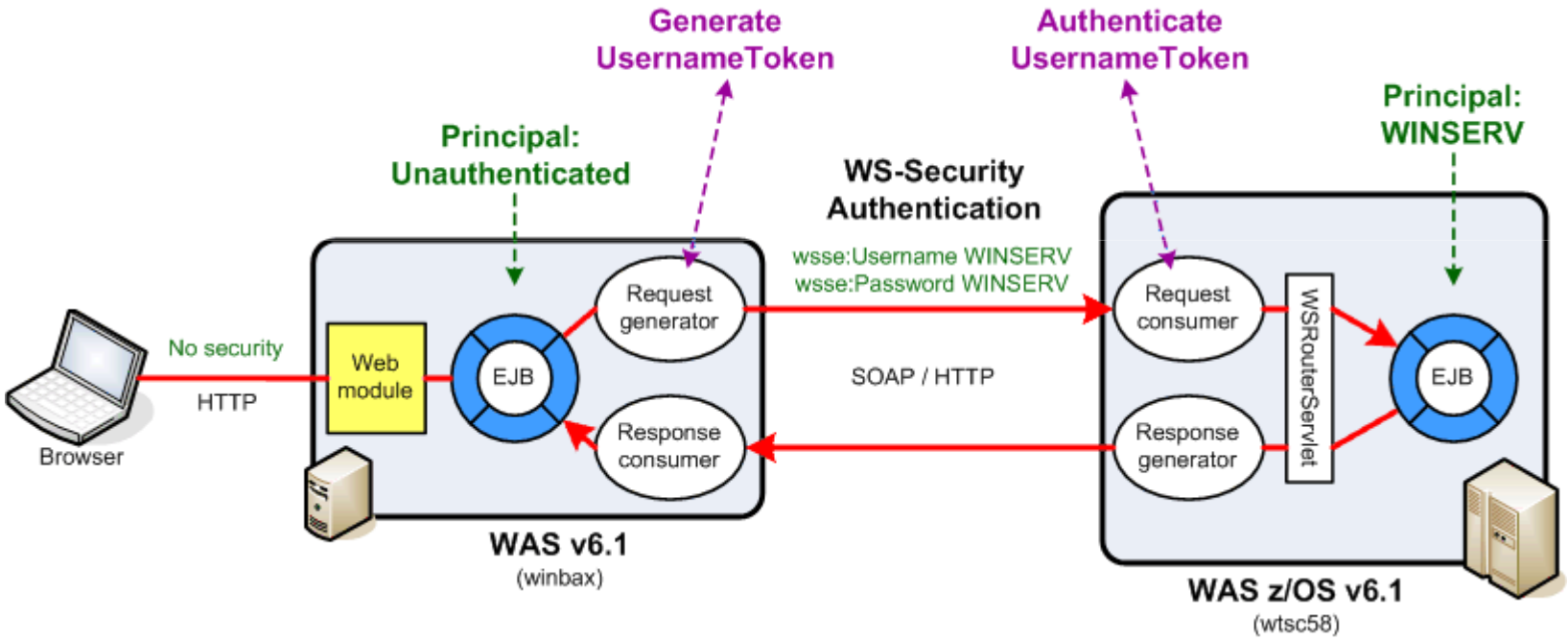


Web Services Message level security

- 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>



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

WS-Security can easily be configured using WAS7 Policy Set feature

- **A policy set is a collection of policies.**
- **A policy is a definition of a Quality of Service (QoS).**
- **Simplifies the QoS configuration model**
  - Central repository of reusable policy sets
  - Default policy sets for common configurations
  - Policy sets can now be applied dynamically at runtime as well as at development time via RAD.

JAX-RBC VS JAX-WS

- Some aspects of JAX-WS 2.0 are merely evolutionary to JAX-RPC 1.1, other parts are revolutionary
- Main Programming Differences
  - Data binding between Java and XML is most notable changes
  - JAX-WS leverages heavily on the use of Annotations.
  - JAX-WS added SOAP 1.2 standard. Not a big deal.
  - Removes the Service Endpoint Interface to using a more POJO class style
  - Client Port Lookup changed slightly. Not a big deal.
  - JAX-WS introduces RESTful Web Services which is the successor for SOAP based Web Services.
  - JAX-WS added support for MTOM and SAAJ which allows for optimized transmission of binary data useful for sending attachment data.
  - Ability to support a asynchronous operations
- Recommendations
  - Just getting started with Web Services, we strongly encourage you to move towards JAX-WS
  - Already invested in JAX-RBC, consider a migration to JAX-WS if you need any of the following:
    - use message-oriented API
    - MTOM, SOAP 1.2
    - better support for XML schema
    - asynchronous programming model
- Good Resources
  - http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com.ibm.websphere.express.doc/info/exp/ae/rwbs_migjaxrpc2jaxws.html
  - http://www.ibm.com/developerworks/webservices/library/ws-tip-jaxwsrpc.html
  - http://www.ibm.com/developerworks/webservices/library/ws-tip-jaxwsrpc2.html

1074 – Security Application Architecture Development and Integration Overview

Additional Features

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 - similar to dWAS, your options for WAS61

- LocalOS UR
- LDAP UR
- Custom UR
- Federated Repository (VMM)

- 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

SAML Support as of WAS 7.0.0.7

- OASIS Web Service Security SAML Token Profile 1.1
  - SAML Assertion V1.1 and V2.0
- Configurable via policy sets
  - Targets JAX-WS
  - Leverages Custom Token Support
- API allowing customer to create and consume SAML Assertion
  - Utility available for building customer SSO solutions independent from Web Services.
  - Does not support using SAMLToken in WSS API
- Supports External STS
  - We tested TFIM and Microsoft Geneva
  - No STS shipped with Product
  - Provide API to request and validate SAML assertion via WS-Trust V1.2 and V1.3 protocol

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.

WAS for z/OS and z/Linux Cryptography

- The z/OS Daemon Address space uses z/OS system SSL
- WebSphere DMGR, Node Agent, and Application Server leverages Java Cryptograghy Extensions (JCE) and Java Secure Socket Extensions (JSSE).
- Supports HW Crypto Accelerator and Secure Key
  - Secure Key meaning Crypto Keys are physically stored in the HW.
  - Accelerate meaning Crypto Operations execute on the HW card. Secure Key is not required to achieve Accelerator.
  - z/OS uses IBMJCECCA provider
  - z/Linux uses PKCS11 provider

- Note: For all other platforms, JDK only supports Secure Key for a number of 3rd party card provider. For more information on what supported, see http://www.ibm.com/developerworks/java/jdk/security/60/secguides/pkcs11implDocs/IBMPKCS11SupportList.html

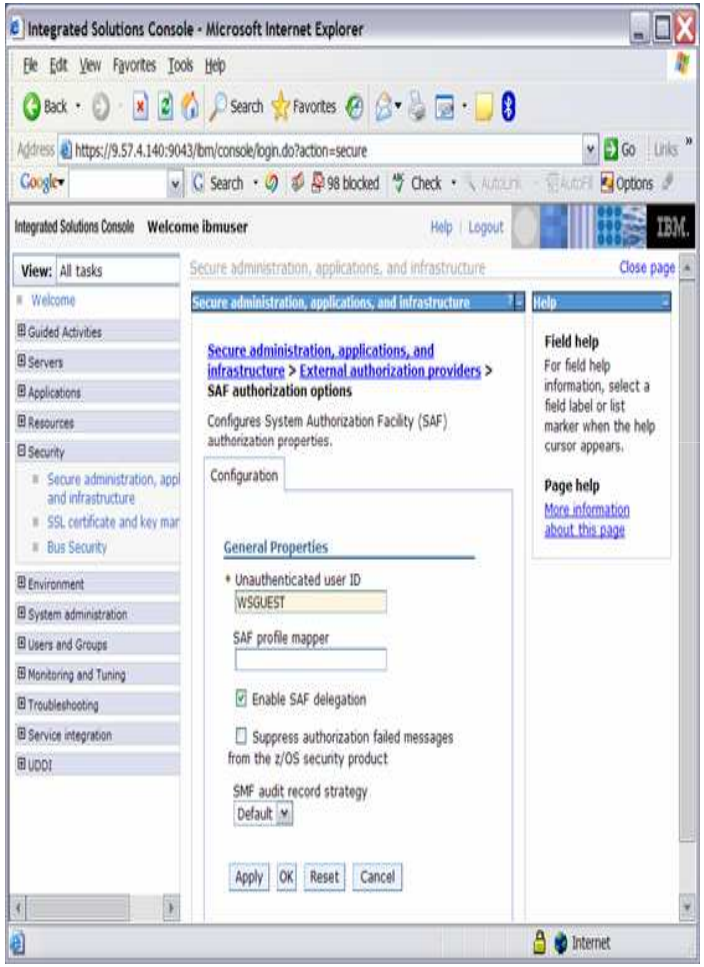
WAS for z/OS
Sync to Thread Option

- The Operating System User Identity is synchronized with the WAS Subject or delegated RunAs identity in the servlet or EJB thread.
- Any access outside of the WAS container such as accessing a file, the thread idenity will be assign the ID of the SAF user instead of the Identity of the server.
- To activate this function you must have all of the following set:
  - Using SAF **U**ser **R**egistry (LocalOS) or Identity Mapping.
  - Application must include within its deployment descriptor an env-entry of com.ibm.websphere.security.SyncToOSThread set to true.
  - Security configuration must have Sync to Thread enabled
    - GUI – Secure administration, applications, and infrastructure > z/OS security options > Enable application server and z/OS thread identity synchronization.
  - ***New in Version 6.1!*** SAF Authorized to use Sync to Thread
    - CR must have CONTROL ACCESS to SAF resource FACILITY BBO.SYNC.<cell short name>.<cluster short name>
    - OR CR must have READ ACCESS to SAF resource FACILITY BBO.SYNC.<cell short name>.<cluster short name> and SR must have READ ACCESS to the SAF resource of SURROGATE BBO.SYNC.<authenticated User ID>

WAS for z/OS
Sync to Thread Example

- A simple test application attempts to access a UNIX file named ***/tmp/test.txt***.
- For testing purposes, we set file ***/tmp/test.txt*** UNIX file permission so both the WAS Servant User ***CBSYMSR1*** and ***USER3*** do NOT have access.
- Lets see what our security violation messages will render.
- When running with Synch to Thread DISABLED
  - ICH408I USER(**CBSYMSR1**) GROUP(CBCFG1) NAME(WAS APPSVR SR) ***/tmp/test.txt*** CL(FSOBJ) ID(01E6E2C8C6E2F800010300003C3C0000) **INSUFFICIENT AUTHORITY TO OPEN** ACCESS INTENT(R--) ACCESS ALLOWED(OTHER ---) EFFECTIVE UID(0000002113) EFFECTIVE GID(0000002300)
  - Shows that User ***CBSYMSR1*** attempted to access file ***/tmp/test.txt***
- When running with Synch to Thread ENABLED
  - ICH408I USER(**USER3**) GROUP(GROUP1) NAME(CB390 USER3) ***/tmp/test.txt*** CL(FSOBJ) FID(01E6E2C8C6E2F800010300003C3C0000) **INSUFFICIENT AUTHORITY TO OPEN** ACCESS INTENT(R--) ACCESS ALLOWED(OTHER ---) EFFECTIVE UID(0000033114) EFFECTIVE GID(0000033333)
  - Shows that User ***USER3*** attempted to access file ***/tmp/test.txt***

WAS for z/OS
Unauthenticated User



- z/OS requires that all Users have some kind of identity.
- In particular, if we enable Sync to Thread for an unauthenticated WAS User, we need to use some Identity that can be synced to the operating system.
- In zWAS, the Unauthenticated Users are represented as a default ID for non-authenticated Users.
- Default ID must be defined in SAF based product. This ID is usually setup with limited access to z/OS resources.
- The restricted ID will use a default SAF ID of WSGUEST or ID specified in the configuration.
- If SAF authorization is enabled, zWAS administrator can specify the default ID to be used.
  - During the installation dialog process
  - GUI – Secure administration, applications, and infrastructure > External authorization providers > SAF authorization options > Unauthenticated User ID.

WAS for z/OS
SAF Profile Mapper

- RACF has a restriction of 240 characters for SAF profiles which means a potential issue for role names over 240 characters. Future they will remove restriction.
- In addition, SAF profiles do not support profile names containing any white space or extended code page characters.
- In order to get around this restriction, zWAS will allow the customers to develop a SAF EJB Role Mapper Class which will simply map J2EE roles to SAF EJBRole Profiles.
- Inherently, this requires SAF Authorization to be enabled.
- Code a SAF Profile Mapper Class

```
public class SAFRoleMapperImpl1 {
    String domainPrefix = null;
    public void initialize(Properties context) {
        domainPrefix = context.get(SAFRoleMapper.DOMAIN_NAME); }
    public String getProfileNameFromRole(String app, String role) {
        String profile = app + "." + role;
        If (domainPrefix != null) {
                    profile = domainPrefix + "." + profile; }
        profile = profile.replaceAll("\\%", "#");
        profile = profile.replaceAll("\\&", "#");
        profile = profile.replaceAll("\\*", "#");
        profile = profile.replaceAll("\\s","#"); return profile; } }
```

WAS for z/OS
Trusted Applications

- In general, if you are using any aspects of the SAF Security (LocalOS UR, SAF Authorization, Sync to Thread, etc) we recommend that customers to enable Trusted Application
- This feature is critical to z/OS System Integrity Statement which we will see covered later in this presentation.
- To enable Trusted Application, the Server must have SAF Access of READ to CLASS FACILITY and profile of BBO.TRUSTEDAPPS.<cell>.<cluster>
- For example, a RACO/ACEE can not be created for asserted identity credential (no password) unless the Trusted Applications is enabled

1074 – Security Application Architecture Development and Integration Overview

DB2

SHARE in Boston

Introducing DB2 security objects

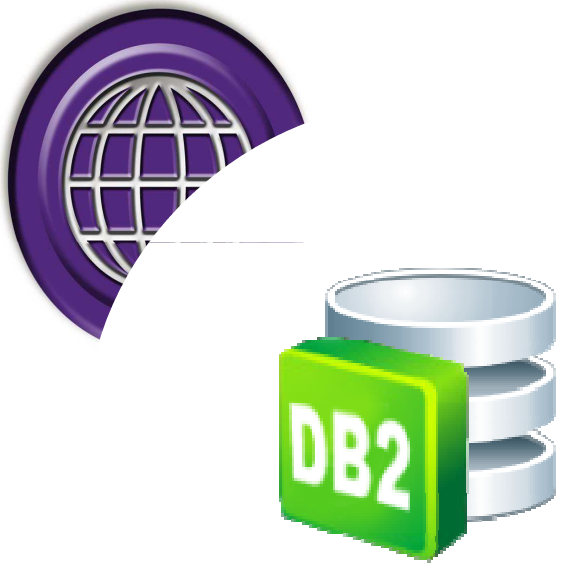
- DB2 ***TRUSTED CONTEXT***
  - A new object used to control users and applications access to DB2
- DB2 ***ROLE***
  - A new object that can be granted privileges or own objects

Associating an application with a trusted context

- Application attributes are verified before associating it with a trusted context such as the system user id and where the request originated
- Allows a unique set of privileges to be associated with an application preventing the misuse of privileges when not accessing through the trusted context
- Controls what end users can be associated with an application eliminating the need to manage RACF user credential from trusted servers

CREATE TRUSTED CONTEXT

- Provide a system ID and connection attributes necessary to associate a trusted context to a connection
  - ☑ **IP Address or host name of remote application**
  - ☑ **JOBNAME of local application**
  - ☑ **Encryption requirements**
  - ☑ **Enabled or disabled by administrator**
- Provide optional list of users can be associated with the trusted connection
- Provide authentication requirements for users in list of users
- Provide optional ROLE to control application privileges
- Provide optional RACF SERVAUTH profile to control access by network zones
- Provide optional SECURITY LABEL can be associated with the connection

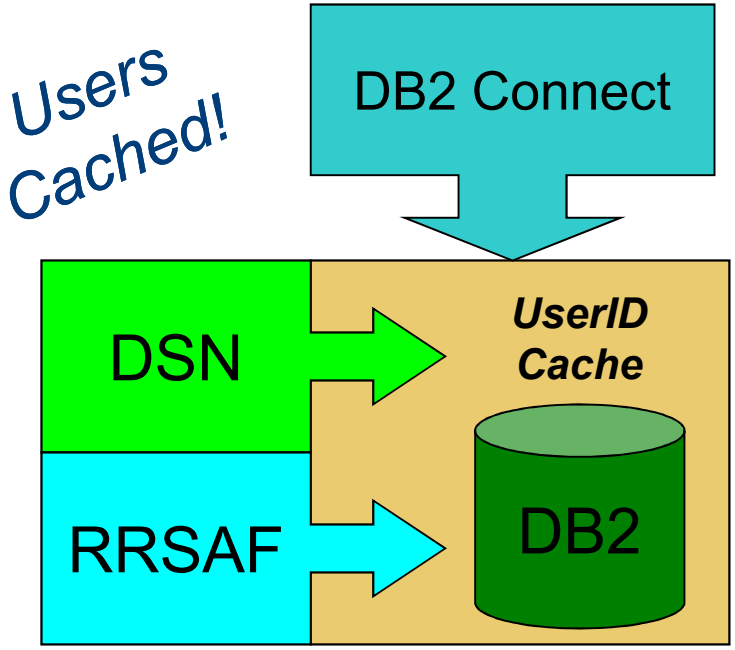


Trusted Context Example

```
CREATE TRUSTED CONTEXT CTX1
        BASED UPON CONNECTION
        USING SYSTEM AUTHID WASADMIN
        WITH USE FOR SAM, JOE, PETE, MARY
        WITHOUT AUTHENTICATION
        ATTRIBUTES (ADDRESS '9.67.40.219')
        ENCRYPTION HIGH
        SECURITY LABEL SAFEZONE
        ENABLE;
```

Establishing a Trusted Connection

- An application can be associated with a trusted context using:
  - DDF
  - RRS Attach
  - DSN ASUSER
  - BATCH



Client Exploitation

- New CLI and JDBC Client Driver APIs
  - JDBC example:

```
Cookie=getDB2TrustedPooledConnection(authid, pwd, …);
getDB2Connection(Cookie, newUser, newPassword, …);
```

- Websphere Application Server
  - Database property:

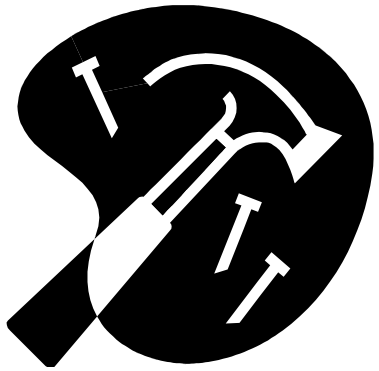
```
propagateClientIdentityUsingTrustedContext
```

Special Trusted Context Privileges

- Once an application is associated with a trusted context, it can:
  - Acquire additional privileges through a ROLE
  - Acquire a RACF security label
  - Efficiently switch user associated with connection on transaction boundary
  - Allow objects created to be owned by the ROLE

CREATE ROLE

- Creates a database entity that can have one or more privileges granted to it
- Role associated with a DB2 process when a connection is associated with a trusted context
- Means to acquire context specific privileges
- Can own DB2 objects when trusted context is defined with “Role as Object Owner”



Role Example

```
CREATE ROLE CTXROLE;

CREATE TRUSTED CONTEXT CTX1
      BASED UPON CONNECTION
      USING SYSTEM AUTHID ADMIN1
      DEFAULT ROLE CTXROLE
      WITH ROLE AS OBJECT OWNER
      ATTRIBUTES(ADDRESS'9.67.40.219')
      ENABLE;

GRANT DBADM TO CTXROLE;
```

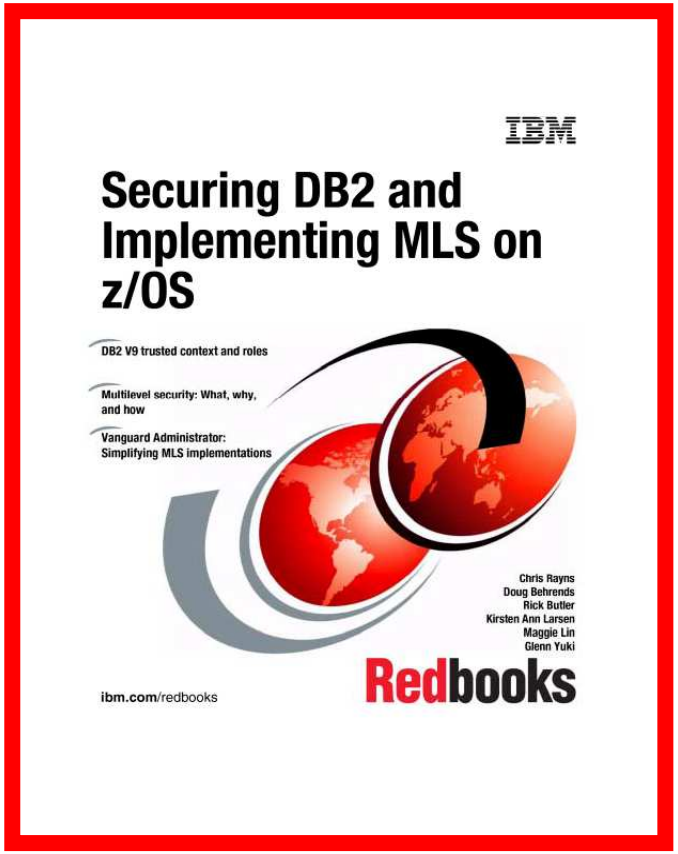
Best Practices using new features

- Secure an existing Application Server
- Secure DBA Activities
- Allow DBA to run as another USER
- Allow remote IDs to be included in z/OS audit logs

DB2 Security Redbook

SG24-6480

- Multilevel security
- Trusted Context
- Roles



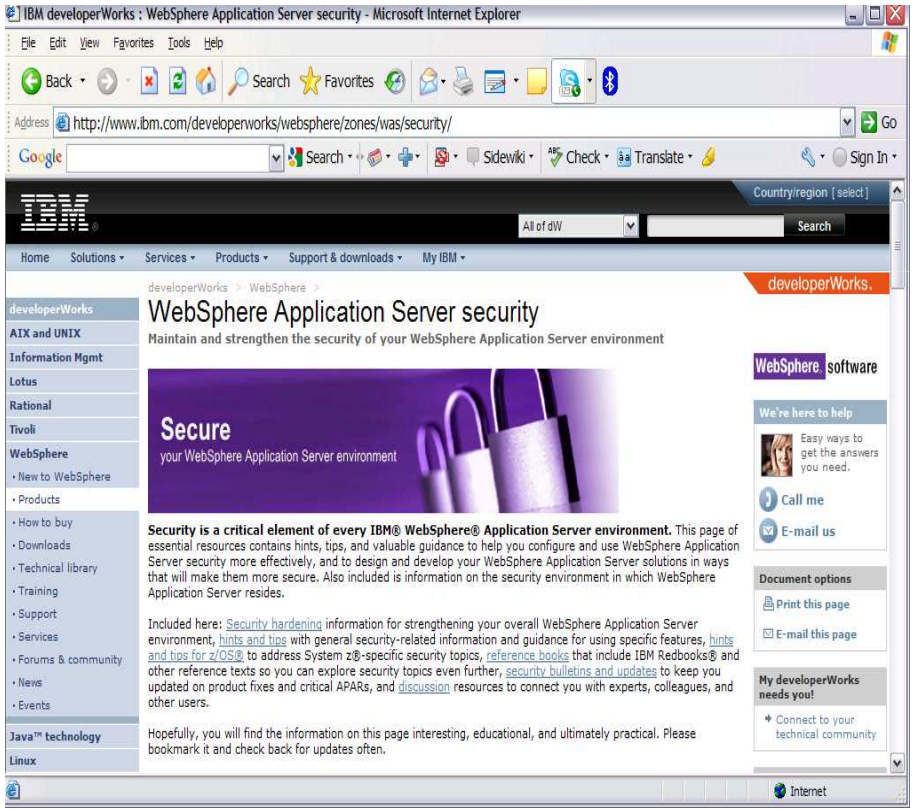
1074 – Security Application Architecture Development and Integration Overview

Final Word

SHARE in Boston

Visit Our Website

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



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

WebSphere Application Server Sessions

Session Number	Day	Time	Title	Speaker
1076	Monday	11:00	WAS z/OS – Architecting Mixed-Platform Cells	Don Bagwell
1179	Monday	1:30	WAS z/OS – Java Out Of Memory (OOM) Analysis Hands-on Lab	Michael Stephen & Ken Irwin
1065	Monday	4:30	WebSphere Application Server Latest Status	Dave Follis
1089	Tuesday	11:00	WebSphere for z/OS Migration – Walk Through, Warnings and Feedback	Mike Stephen and Mickey Scott
1107	Tuesday	3:00	WebSphere for z/OS – I am no longer a dummy but…	Don Bagwell
1064	Tuesday	4:30	Which platform should I use for my WebSphere Application?	Mickey Scott
1147	Wednesday	3:00	Introduction to using IBM Support Assistant for WebSphere Application Server for z/OS	John Hutchinson
1143	Wednesday	4:30	Avoiding the potholes on the WebSphere Application Server for z/OS Onramp	Mike Loos
1172	Wednesday	6:00	WAS z/OS – Making Use of Optimized Local Adapters	Don Bagwell
1074	Thursday	8:00	Security Architecture – How does WebSphere play?	Bill O'Donnell
1075	Thursday	9:30	WebSphere Application Server Version 7 Management Strategies	Paul Houde
1077	Thursday	1:30	WAS z/OS – High Availability Architectural Considerations	Don Bagwell

धन्यवाद

多謝

ขอบคุณ

Спасибо

Gracias

Merci
French

Thank You

شكرا

Obrigado

多谢

Danke

Grazie

நன்றி

ありがとうございました

Teşekkürler

감사합니다