

IBM WebSphere Application Server V4.0

WebSphere V4.0 and J2EE Security



Topics



- Security changes
- J2EE 1.2 Security Architecture
- EJB 1.1 Security Responsibilities
- Web Components Security
- WebSphere V4.0 Security Administration
- Pluggable Authentication Registry



WebSphere Security: Changes

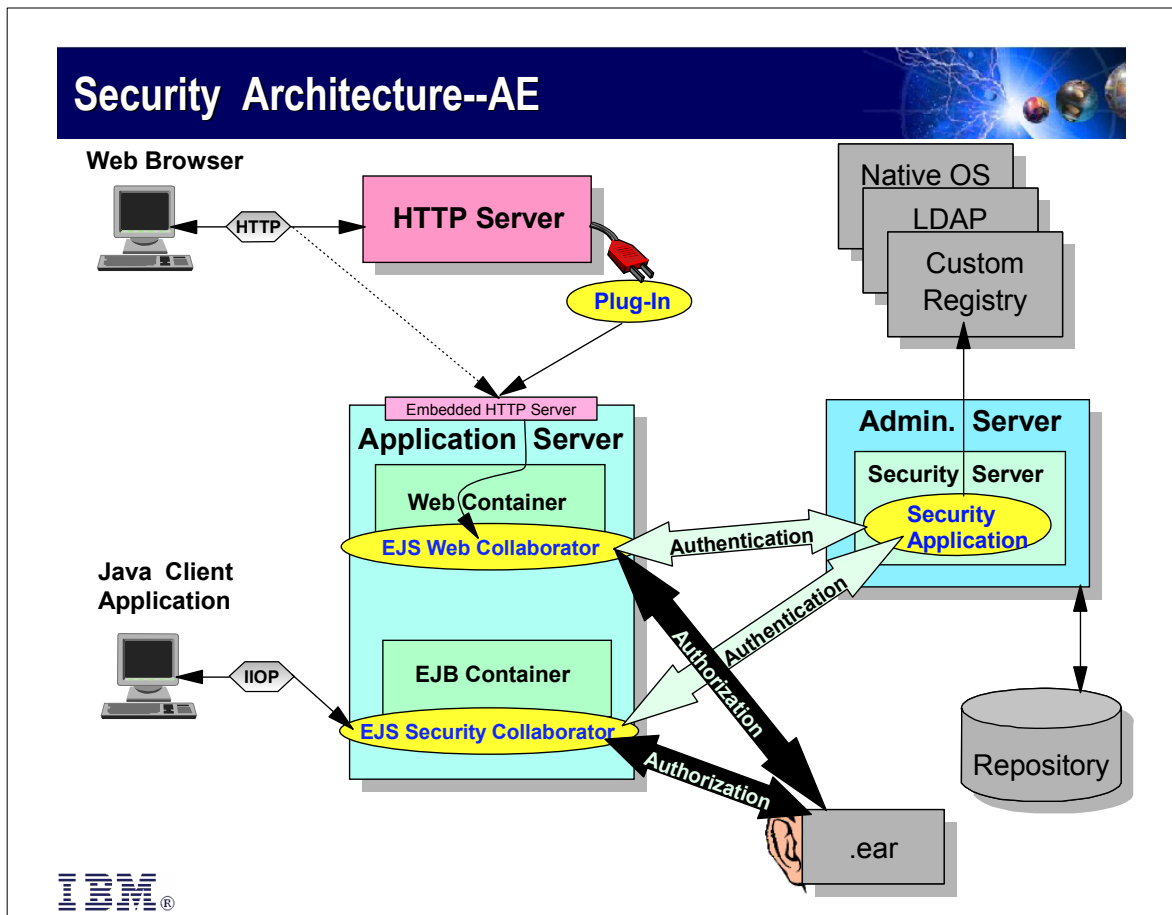


- WebSphere security model is modified to fit J2EE 1.2 requirements (EJB 1.1 and Servlet 2.2)
- In WebSphere V3.5, each method has a list of users/groups who could access the EJB methods
 - ▶ WAS V3.5 has method permission based model
 - ▶ EJB 1.0 addressed security only in a limited fashion
- J2EE 1.2 uses the concept of **security roles** to administer method permissions
 - ▶ Authorization model is changed from Permission-based model to Role-based model
 - ▶ Method permissions are addressed differently



- ▶ With J2EE comes the EJB 1.1 specification, which has a whole section (section 15) on security. The introduction of the Security Management begins "The deployment aspect of security management has changed significantly since EJB 1.0."
- ▶ In WAS V3.x, we grouped methods into Method Groups to assign permissions. These permissions were created and mapped to end users (or groups) after the application was installed.
- ▶ In the EJB 1.1 security model, only the mapping can be modified after install. Everything else is accomplished (by the Assembler or the Developer) before the Deployer gets the .ear file to install.

Security Architecture--AE



▶ AEs/AEd uses Local OS security only.

▶ Security Flow::

1. Request comes in.

▶ Request goes to Collaborator.

▶ Collaborator sends 401 to plug-in/HTTP Server.

▶ 401 goes back to browser.

▶ Request comes back with user ID/PW (or Certificate as appropriate).

2. If WebSeal is enabled, HTTP Server authenticates the user. The request goes to Collaborator in the appropriate Application Server. Otherwise, request goes straight to Collaborator.

3. Collaborator forwards to Security Server, which either:

a) Authenticates user, or

b) Authenticates Web Server and trusts that the user is 'real' (WebSeal)

4. Security Server tells Collaborator that user is (or is not) valid.

5. Collaborator checks permissions in the deployment descriptor to determine if the request will be served.

One thing to note: Because the security information is in the deployment descriptors, in the EAR file, and because in a server group configuration the EAR file may be on more than one machine, editing the EAR file directly can have some very serious side effects especially in respect to the security policy. This is why editing the installed EAR file is not recommended. If it is necessary to edit the installed EAR file, be EXTREMELY careful to keep all of the EAR files in synch.

J2EE Security Architecture



- Forms of Container-Based Security
 - ▶ Declarative - Outside the code (i.e., method level)
 - ▶ Programmatic - Within the code

- User Authentication Requirements
 - ▶ Must support login sessions - Web single signon
 - ▶ Login Mechanisms:
 - HTTP Basic Authentication
 - SSL Mutual Authentication
 - Form-based login
 - ▶ Unauthenticated access

- Authorization Requirements
 - ▶ Code Authorization - i.e., Access for the bean
 - ▶ Caller Authorization - i.e., Access for the end-user of the bean



- ▶ J2EE uses 'Single Signon' to mean that a user should have to authenticate only when crossing a security policy domain boundary.

- ▶ Unauthenticated Access - Web containers report that a user has not authenticated by returning 'null' from the `HttpServletRequest` method `getUserPrincipal`. However, EJB specification requires that `getCallerPrincipal` never returns null. When a call is made by a servlet to a bean, the container must be able to provide a principal. This may be a single distinguished principal, a principal per server, per session, or per application. The deployer or system administrator will select which principal to use. See J2EE 1.2 Specification Section 3.4.1.4.

- ▶ Code Authorization is based on J2SE and is J2EE component-specific.

- ▶ Client Authorization, as required by J2EE, says that the user's identity be propagated within a single application within a single J2EE product. That is to say that `getCallerPrincipal` must return the same value as returned for the first bean in the call chain. (J2EE 1.2 Specification, Section 3.5.2)

EJB 1.1 Security - Who does what?



■ 15.2 Bean Provider's Responsibilities

"The Bean Provider is responsible for declaring in the **security-role-ref** elements of the deployment descriptor all the security role names used in the enterprise bean code."

■ 15.3 Application Assembler's Responsibilities

"The Application Assembler (which could be the same party as the Bean Provider) may define a **security view** of the enterprise beans contained in the ejb-jar file."

"If the Bean Provider has declared any security role references using the **security-role-ref** elements, the Application Assembler must link all the security role references listed in the security-role-ref elements to the security roles defined in the security-role elements."

"The Applications Assembler defines **method permissions** for each security role."

■ 15.4 Deployer's Responsibilities

"The Deployer is responsible for ensuring that an assembled application is secure after it has been deployed in the target operational environment."

"The Deployer's job is to map the security view that was specified by the Application Assembler to the mechanisms and policies used by the security domain in the target operational environment."



► Section numbers correspond to EJB 1.1 Specification sections of the same name.

For the sake of illustration...



- Developers are going to write "Account and Transfer"
 - ▶ WebSphere sample extracted for labs and renamed "MyBank"
 - ▶ Transfer bean developer will use "Manager" and "Client"
 - ▶ Account bean developer will use "Supervisor" and "User"
- Application Assembler dictates two roles:
 - ▶ "Banker_Role" to handle 'Manager' and 'Supervisor' tasks
 - ▶ "Customer_Role" to handle 'Client' and 'User' tasks



Layers of Abstraction for EJBs

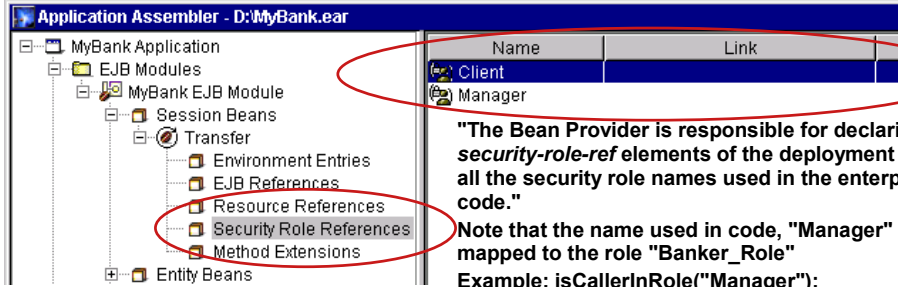
■ EJB Module Level (Declarative) Security Roles



The screenshot shows the 'Application Assembler' interface for 'D:\MyBank.ear'. The left pane shows a tree view with 'MyBank Application' expanded to 'EJB Modules' and then 'MyBank EJB Module'. Under 'MyBank EJB Module', 'Security Roles' is highlighted. The right pane shows a table with two columns: 'Name' and 'Description'. Two rows are visible: 'Banker_Role' and 'Customer_Role'. Red circles highlight the 'Security Roles' folder in the tree and the two rows in the table.

The Application Assembler may define a **security view** of the enterprise beans contained in the ejb-jar file.

■ Bean Level (Programmatic) Security Role Reference



The screenshot shows the 'Application Assembler' interface for 'D:\MyBank.ear'. The left pane shows a tree view with 'MyBank Application' expanded to 'EJB Modules' and then 'MyBank EJB Module'. Under 'MyBank EJB Module', 'Transfer' is expanded to 'Security Role References', which is highlighted. The right pane shows a table with two columns: 'Name' and 'Link'. Two rows are visible: 'Client' and 'Manager'. Red circles highlight the 'Security Role References' folder in the tree and the two rows in the table.

"The Bean Provider is responsible for declaring in the *security-role-ref* elements of the deployment descriptor all the security role names used in the enterprise bean code."

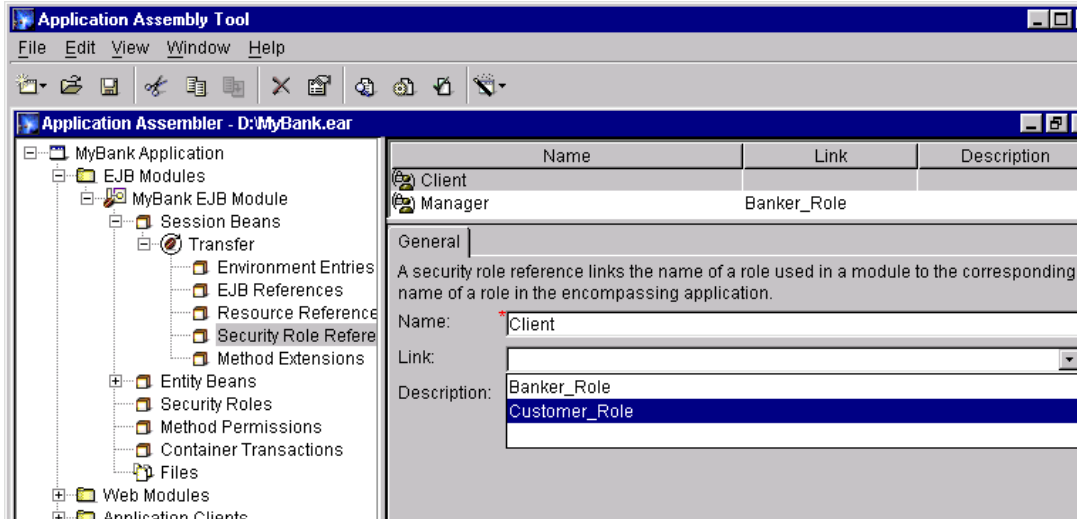
Note that the name used in code, "Manager" is not yet mapped to the role "Banker_Role"
Example: `isCallerInRole("Manager");`



- ▶ In this example, the Application Assembler defines the roles of Banker_Role and Customer_Role in the first step.
- ▶ Developers may or may not use the same names for these capabilities. In this case, the developer calls the super user "Manager", and the not-so-super user "Client".
- ▶ On the next slide, these will be linked.

Application Assembler's Task - Linking

- Developer has provided security role references
- Assembler links them to security roles



Layers of Abstraction for Servlets

■ Web Module Level (Declarative) Security Roles

The screenshot shows the 'Application Assembler - D:\MyBank.ear' interface. On the left, a tree view shows the project structure: MyBank Application > Web Modules > MyBank Web Module > Security Roles. On the right, a table lists the roles:

Name	Description
Banker_Role	
Customer_Role	

Red circles highlight the 'Security Roles' folder in the tree and the role entries in the table.

■ Servlet Level (Programmatic) Security Role References

The screenshot shows the 'Application Assembler - D:\MyBank.ear' interface. On the left, a tree view shows the project structure: MyBank Application > Web Modules > MyBank Web Module > Web Components > Security Role References. On the right, a table lists the role references:

Name	Link
Supervisor	Banker_Role
User	Customer_Role

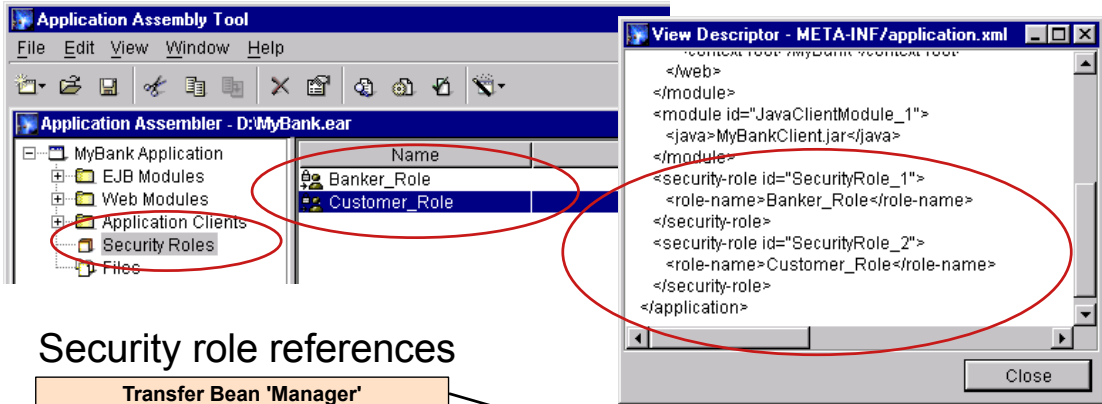
Red circles highlight the 'Security Role References' folder in the tree and the role reference entries in the table.

Example: `isUserRole("Supervisor");`

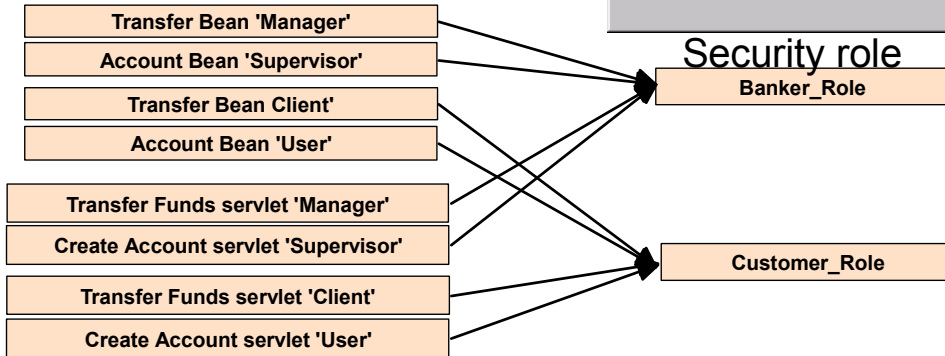
* Note that in this view, the Application Assembler has already linked the Role References to Roles.



Security Roles



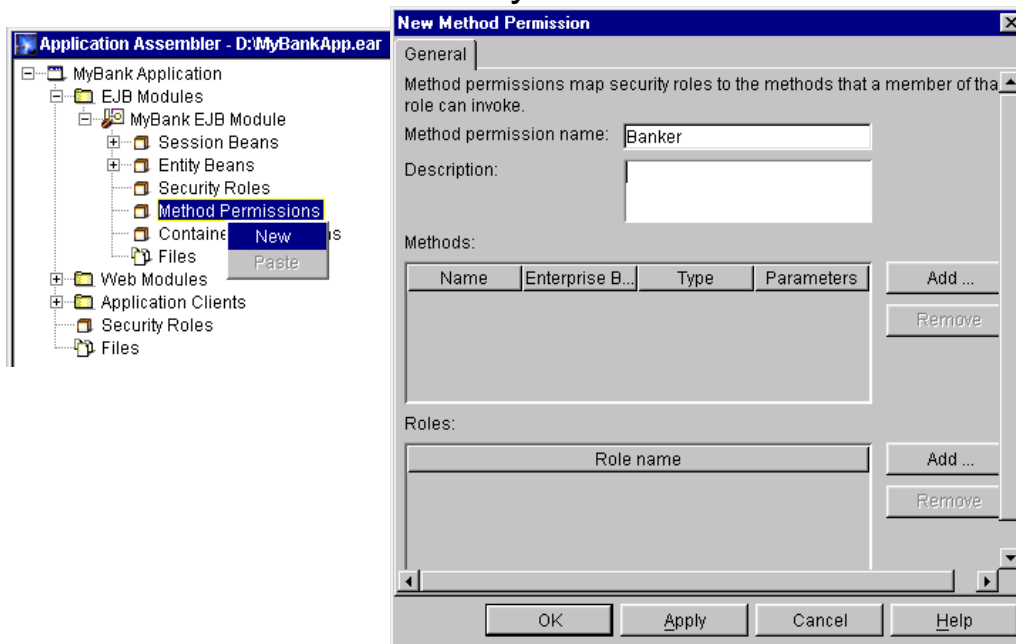
Security role references



- ▶ Right-clicking objects in the properties notebook will often allow you to view the associated deployment descriptor in raw XML form.

EJB Method Permissions

■ Declarative form of security



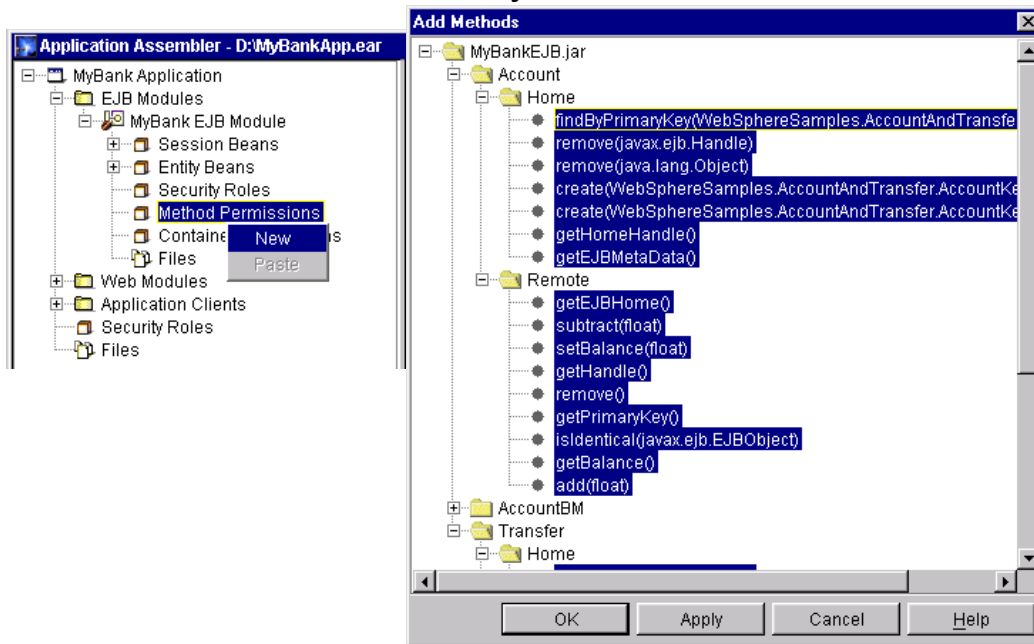
IBM®

Steps to configure EJB Method Permissions:

1. Expand the EJB Module, right-click Method Permissions--> New
2. Assign a permission name

EJB Method Permissions

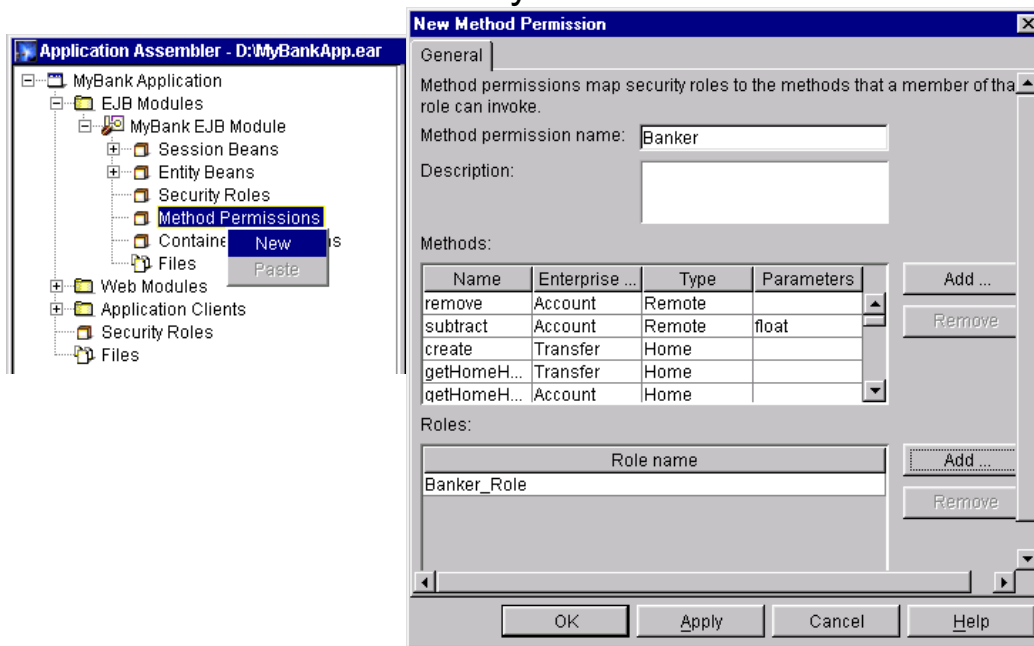
■ Declarative form of security



3. Add methods from the beans to the permission

EJB Method Permissions

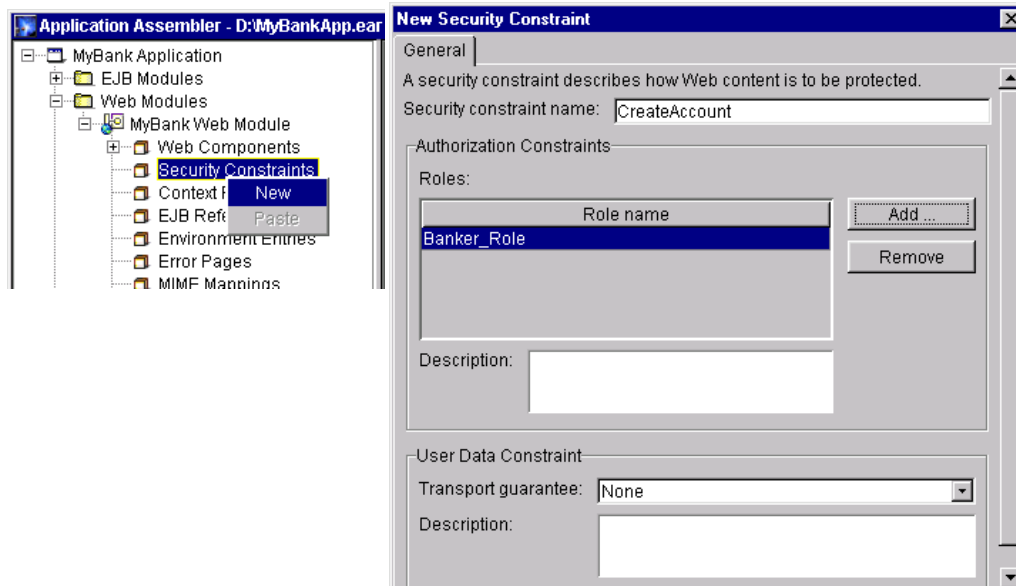
■ Declarative form of security



4. Assign one or more roles that will have access to these methods
5. Click Apply or OK

Servlet Method Permissions

- First, Create a Security Constraint
 - ▶ Add one or more Role Names



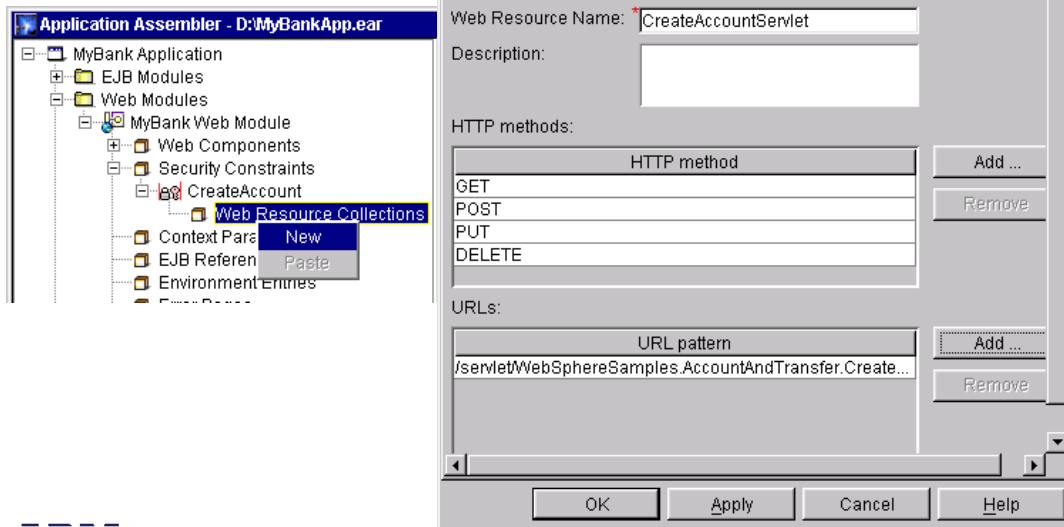
IBM®

- ▶ A security constraint consists of a Web resource collection, an authorization constraint, and a user data constraint.
- ▶ Authorization Constraints specify the user roles that are permitted access to this resource collection.
- ▶ User Data Constraints - Transport guarantee indicates how data communicated between the client and the server is to be protected. Specifies that the protection for communications between the client and server is None, Integral, or Confidential.
 - ▶ None means that the application does not require any transport guarantees.
 - ▶ Integral means that the application requires that the data sent between the client and the server must be sent in such a way that it cannot be changed in transit.
 - ▶ Confidential means that the application requires that the data must be transmitted in a way that prevents other entities from observing the contents of the transmission.
- ▶ In most cases, Integral or Confidential indicates that the use of SSL is required.

Servlet Method Permissions

- Next, Create a Web Resource Collection

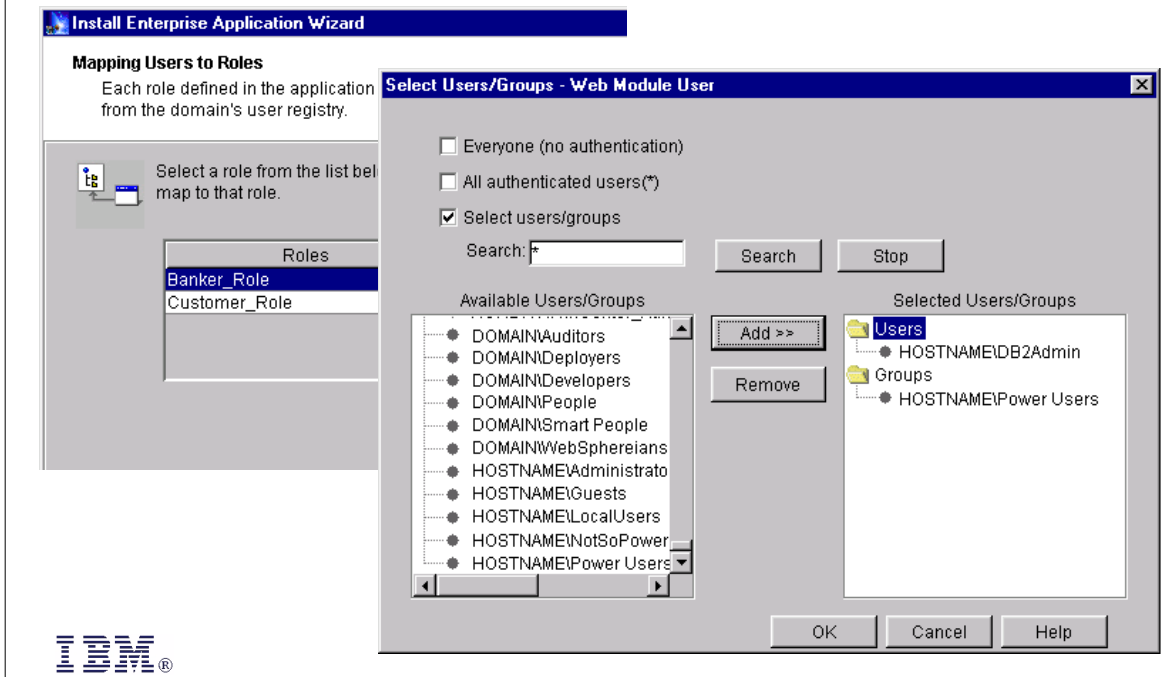
- ▶ Add HTTP Methods
- ▶ Add a URL pattern



- ▶ If no HTTP methods are specified, then the security constraint applies to all HTTP methods.
- ▶ If multiple security constraints are specified, the container uses the "first match wins" rule when processing a request to determine what authentication method to use, or what authorization to allow.

Administrative Console

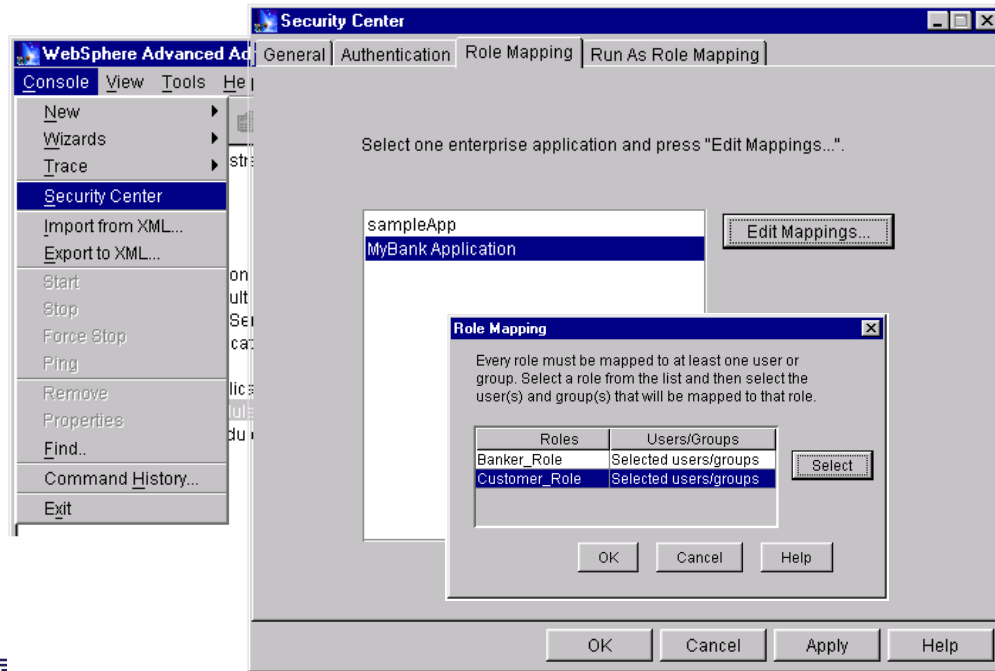
- Role Mappings can be part of the installation process



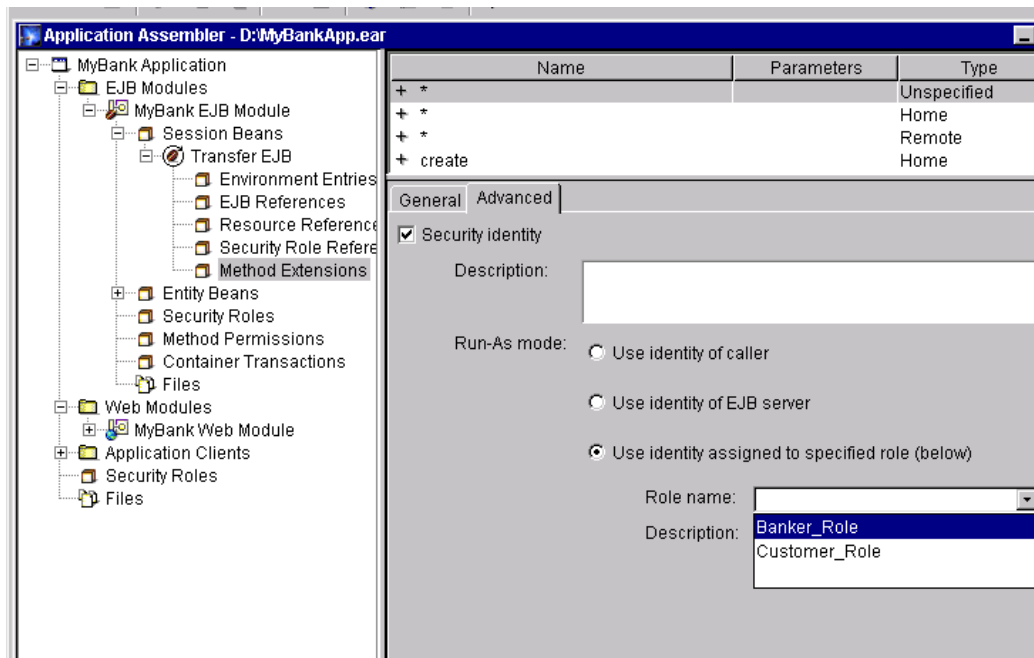
- ▶ When you search the user registry as part of a large domain, it may require several seconds to several minutes to fetch all the users and groups.

Administrative Console

- ...or a manual process after an .ear is installed

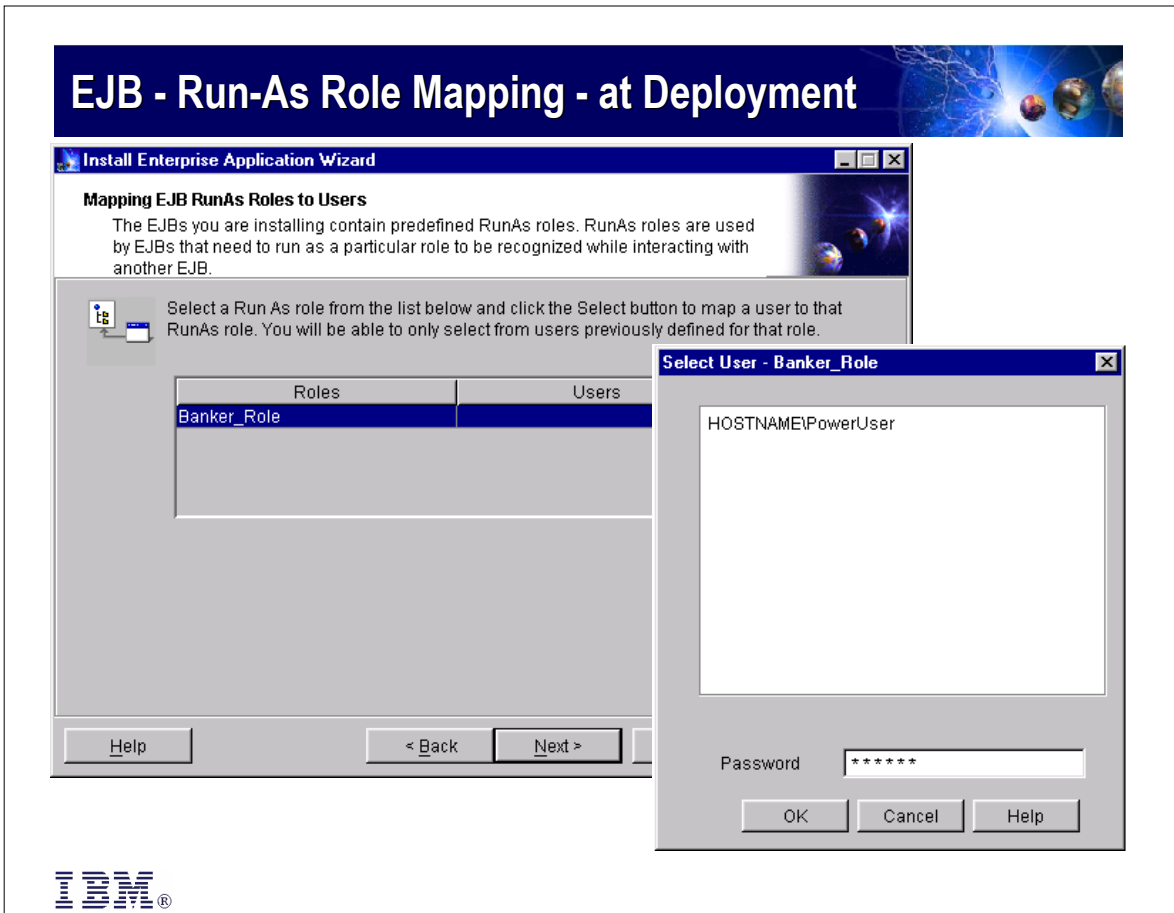


EJB Delegation - "Run As" Mode - at Assembly



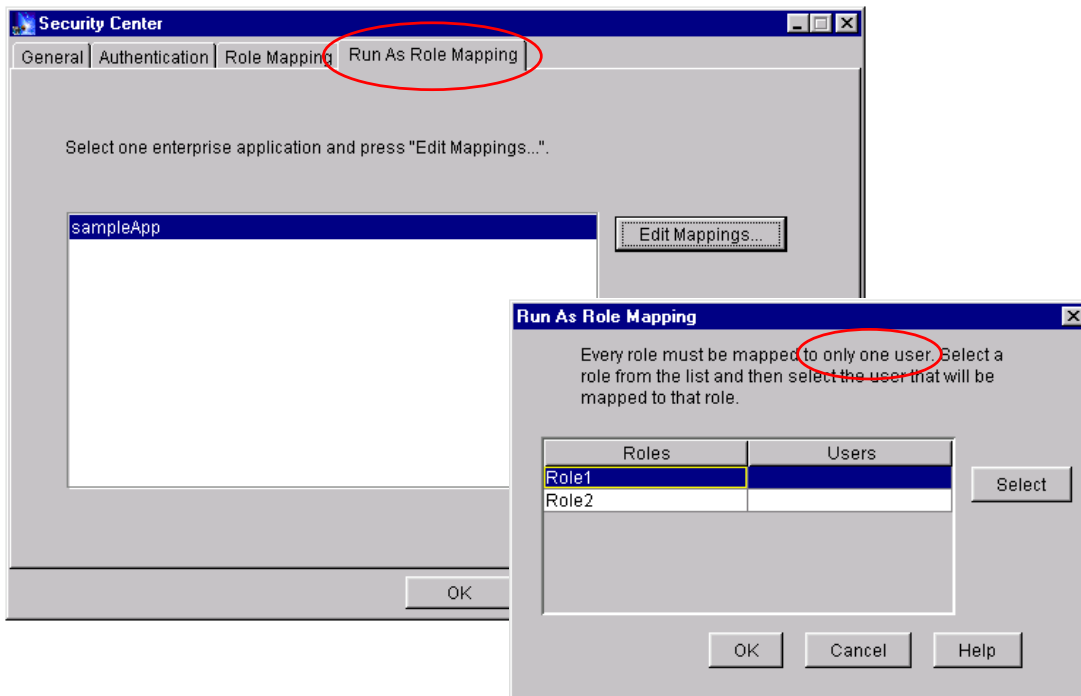
- ▶ To set the delegation mode on an EJB or a method on a bean, use AAT to specify the security identity. Selecting a role here will allow the Deployer to assign a user ID within that role to be the delegated identity.
- ▶ Note that delegation roles can be set on the whole bean, all the Home Interface methods, all the Remote Interface methods, or on individual methods.

EJB - Run-As Role Mapping - at Deployment



- Only users specified as part of the 'Banker_Role' role are visible at this point. You must select an individual user from the list

EJB Run-As Role Mapping - Existing Application



- ▶ If not mapped at Deployment, or if the mapping is to be changed, use the Run As Role Mapping tab in the Security Center.

WebSphere V4.0: EJB 1.1 Security Responsibilities

Function	Responsibility (Role)	Tool
Security role names used internally within the code	Application Component Provider	VAJ, Editor
Specify Security Roles	Application Assembler	AAT
Method Permissions (map security role to methods)	Application Assembler	AAT
Security Role Reference	Application Assembler or Deployer	AAT
Security Role Binding to Principal → User → Group → Special Subjects	Deployer	AAT, Admin Console
Principal mapping for inter-enterprise bean calls (RUN-AS)	Deployer	Admin Console



► Bean Provider

- Implement the EJB without hard-coding the security policies and mechanisms into the business methods.

► Application Assembler (could be the Bean Provider)

- Define security roles for an application composed of one or more EJBs.
- Security role is a semantic grouping of permissions that a given type of users of the application must have to use the application.
- Define method permissions for each security role.
- Its is a permission to invoke a specified group of methods of the enterprise beans' home and remote interfaces.
- Present a simplified security view of the EJB to the Deployer.

► Deployer

- Deal with small set of security roles rather than a large number of individual methods.
- Responsible for assigning principals or groups (of principals) to the security roles defined by the Application Assembler for the EJB in the deployment descriptor.
- Configure principal mapping for inter-enterprise bean calls (RUN-AS) and principal mapping for resource manager access (JDBC, etc).

- **"Special Subjects"** under Security Role Binding refers to "Everyone" or "All Authenticated Users" options for access.

WebSphere V4.0: Default Security Settings



For EJB and Web-Resources

Setting	WAS 3.5.x	WAS 4.0
Security NOT defined for any method in EJB, servlet	<ul style="list-style-type: none">• EJB: Denied• Web-Resource: Grant	<ul style="list-style-type: none">• EJB: Grant• Web-Resource: Grant
Default setting for methods with NO security definition in resources, where at least one method has Security defined	<ul style="list-style-type: none">• EJB: Denied• Web-Resource: Grant	<ul style="list-style-type: none">• EJB: Denied• Web-Resource: Grant



- ▶ When security is turned on and at least one bean method has security defined, access to all other methods of any instance *of that EJB* in the domain will be denied. That is, if you define security on the Transfer bean, the Account bean will still be unprotected.

WebSphere V4.0 AEs: Where is security info saved?



Security Info	Tool	Location Stored
<ul style="list-style-type: none"> • Global security • Authentication • SSL settings 	Admin Console	<ul style="list-style-type: none"> • server-cfg.xml • server-cfg.xml • server-cfg.xml
Application Security Roles	AAT	EAR DD application.xml
Security Role Binding to Users, Groups, Special Subjects	AAT, Admin Console	EAR DD ibm-application-bnd.xmi
EJB <ul style="list-style-type: none"> • Security Role • Role Reference • Method Permission • Sec. Identity - RunAs • RunAs Mapping 	<ul style="list-style-type: none"> • AAT • AAT • AAT • AAT • Admin Console 	<ul style="list-style-type: none"> • EJB DD ejb-jar.xml • EJB DD ejb-jar.xml • EJB DD ejb-jar.xml • EJB DD ibm-ejb-jar-ext.xmi • EAR DD ibm-application-bnd.xmi
Web Resources <ul style="list-style-type: none"> • Security Role • Role Reference • Sec. Constraints 	<ul style="list-style-type: none"> • AAT • AAT • AAT 	<ul style="list-style-type: none"> • WAR DD web.xml • WAR DD web.xml • WAR DD web.xml



WebSphere V4.0 AE: Where is security info saved?

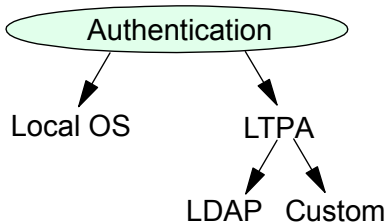
Security Info	Tool	Location Stored
<ul style="list-style-type: none"> • Global security • Authentication • SSL settings 	Admin Console	WAS repository
Application Security Roles	AAT	EAR DD application.xml
Security Role Binding to Users, Groups, Special Subjects	AAT, Admin Console	EAR DD ibm-application-bnd.xmi, WAS repository
EJB <ul style="list-style-type: none"> • Security Role • Role Reference • Method Permission • Sec. Identity - RunAs • RunAs Mapping 	<ul style="list-style-type: none"> • AAT • AAT • AAT • AAT • Admin Console 	<ul style="list-style-type: none"> • EJB DD ejb-jar.xml • EJB DD ejb-jar.xml • EJB DD ejb-jar.xml • EJB DD ibm-ejb-jar-ext.xmi • WAS repository
Web Resources <ul style="list-style-type: none"> • Security Role • Role Reference • Sec. Constraints 	<ul style="list-style-type: none"> • AAT • AAT • AAT 	<ul style="list-style-type: none"> • WAR DD web.xml • WAR DD web.xml • WAR DD web.xml



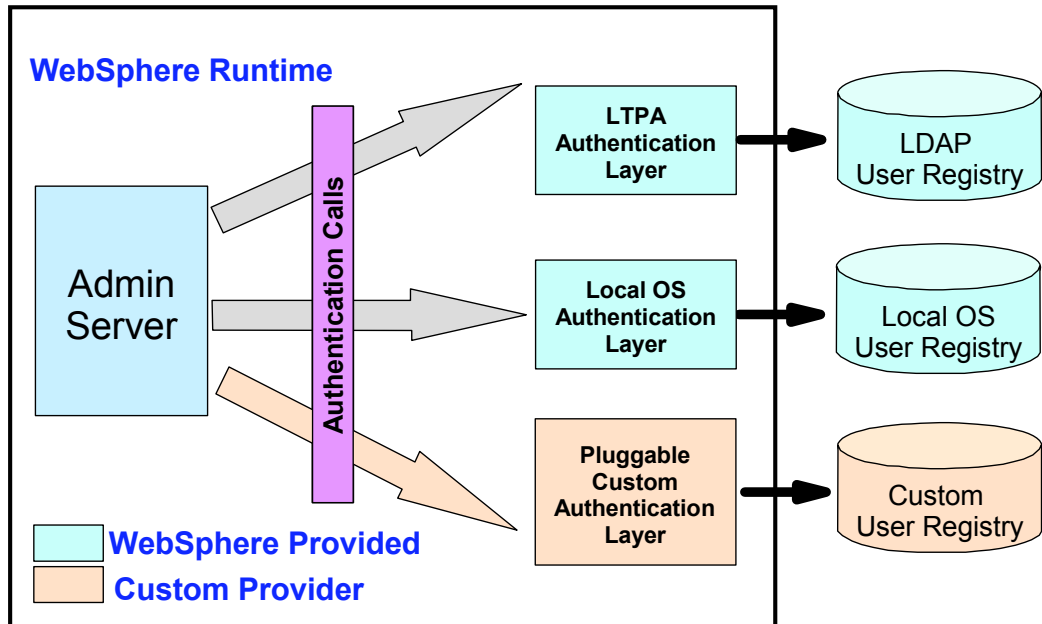
- ▶ Only Mappings are stored in the repository. The rest of security information, roles, role references, permissions, and so on, is stored in the appropriate XML files in the expanded representation of the .ear file in the <was_home>/installedApps directory.
- ▶ Mappings can be stored by AAT in the ibm-application-bnd.xmi file as preferences. If there is mapping information in the repository, it takes precedence over the information in the .xmi file.

WebSphere V4.0: Authentication Registries

- WebSphere V3.5 supports following registries
 - ▶ Local OS registries
 - ▶ Centralized registries- LDAP using supported LTPA service
- New addition in WebSphere V4.0 AE
 - ▶ Pluggable CUSTOM-REGISTRY interface
 - Implemented as LTPA
 - Allows WebSphere applications to take advantage of new registries.



WebSphere V40 Security: Custom Registry



- ▶ WebSphere Security in Version 3.5 supported two ways to look up users for authentication:
 - ▶ Lightweight Third Party Authentication (LTPA) verifies users stored in a Lightweight Directory Access Protocol (LDAP) directory.
 - ▶ Local OS verifies users stored in the Local Operating System registry.
- ▶ WebSphere Version 4.0 adds a new option for Pluggable Custom Authentication to verify users stored in a custom user registry.
- ▶ This allows WebSphere to cooperate with any given registry through a custom authentication layer, implemented by the customer.

How Pluggable Custom Registry Works



- Application developer
 - ▶ Implements the methods in the **CustomRegistry** interface
 - com.ibm.ejs.security.registry package
 - ▶ This layer of code interacts with the custom user registry

- Administrator
 - ▶ Selects Custom pluggable Registry option and specifies the class

- Admin/Security Server
 - ▶ Calls CustomRegistry methods to perform authentication for applications



- ▶ CustomRegistry interface in com.ibm.ejs.security.registry package
- ▶ GUI for pluggable Registry is being designed



Some of the methods of **CustomRegistry interface**:

// User-related methods

```
public boolean isValidUser(String securityName)
public List getUsers()
public List getUsers(String pattern)
public String getUserDisplayName(String securityName)
public String getUniqueId(String securityName)
```

// Group-related methods

```
... // Similar to user methods
```

// Authentication methods

```
public String checkPassword(String userId, String password)
public String mapCertificate(X509Certificate cert)
```



Security Center GUI

The screenshot displays the Security Center GUI with the following components:

- Security Center (Main Window):**
 - Tabbed interface: General | Authentication | Role Mapping | Run As Role Mapping
 - Authentication Mechanism: Local Operating System, Lightweight Third Party Authentication (LTPA)
 - LTPA Settings:
 - Token Expiration: 30
 - Enable Single Sign On (SSO)
 - Domain: [Empty]
 - Limit to SSL connections only
 - Enable WebSeal trust association
 - Buttons: Generate Keys, Import Key..., Export Key...
 - LDAP: Custom User Registry
 - Custom User Registry Settings:
 - Security Server ID: [Empty]
 - Security Server Password: [Empty]
 - Custom User Registry Class Name: [Empty]
 - Special Custom Settings button

- Specify Custom Settings (Dialog Box):**
- Table with columns: Name, Value
- Buttons: OK, Cancel, Help

A blue arrow points from the 'Special Custom Settings' button in the main window to the 'Value' column in the 'Specify Custom Settings' dialog box.

Summary



■ Topics Covered

- ▶ Security changes - Method-based to Roles-based
- ▶ J2EE 1.2 Security Architecture
- ▶ EJB 1.1 Security Responsibilities
- ▶ Web Components Security
- ▶ WebSphere V4.0 Security Administration
- ▶ Pluggable Authentication Registry

■ Resources

- ▶ InfoCenter
- ▶ Specifications
 - <http://java.sun.com/j2ee/docs.html> - Scroll down to Specification links



Lab Preview - Security Lab



- Put on the System Administrator hat
- Create users in Operating System user registry
- Put on the Application Assembler hat
- Configure security in AAT
 - ▶ Assign Security Constraints
 - ▶ Create web resource collections
 - ▶ Map roles to permissions
- Turn on Security in the Admin Console
- Put on Deployer hat
- Install and test Enterprise Application in AE

