# Adding Security to your J2EE Applications

THE BEST-RUN BUSINESSES RUN SAP

# Agenda

**Review: J2EE Security**

**Network Topology**

**Usermanagement**

**Authentication**

**Secure Storage**

# J2EE Security Security Models

## J2EE supports two different Security Models

- Declarative Security
  - Access control linked to the resource
  - Decouples Access Control from application logic
  - Easy to implement and maintain
- Programmatic Security
  - Access control within Java code
  - More flexible but linked to application logic
  - More work to implement

# J2EE Security Security Standards

## Java Authentication and Authorization Services (JAAS)

- authenticate and enforce user-based, group-based, or role-based access controls.

## Java Cryptographic Extensions (JCE)

- key generation and key agreement
- symmetric, asymmetric, block, and stream ciphers
- Message Authentication Codes
- secure streams and sealed objects

## Java Secure Sockets Extensions (JSSE)

- Java version of SSL (Secure Sockets Layer) and TLS (Transport Layer Security) protocols

# Agenda

Review: J2EE Security

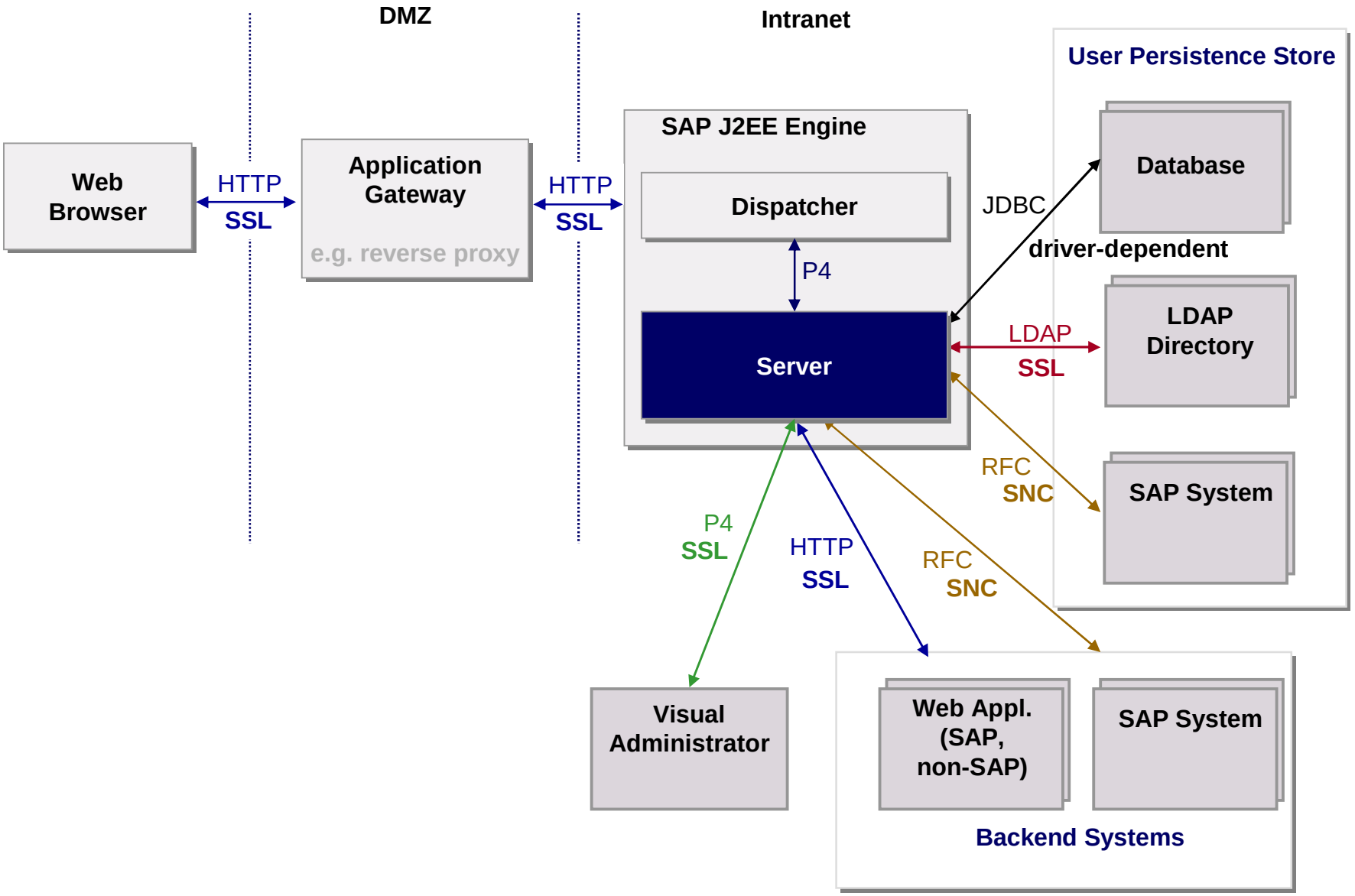
Network Topology

Usermanagement

Authentication

Secure Storage

# J2EE Engine Protocols Overview



# SAP J2EE Engine Ports

## SAP J2EE Engine Dispatcher ports default to 50000+100*sid+port_index

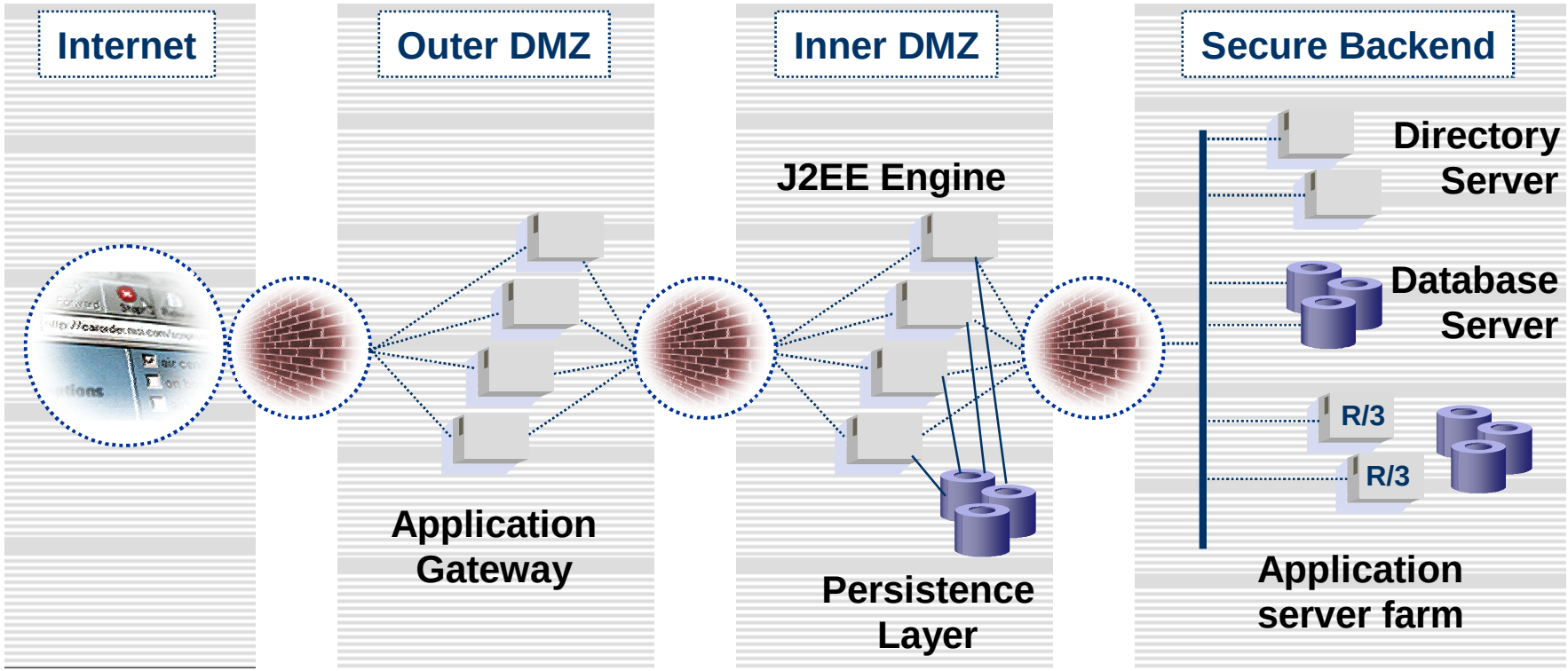
| Index | Dispatcher Service |
|---|---|
| 0 | HTTP Port |
| 1 | HTTPS Port |
| 2 | IIOP Initial Context Port |
| 3 | IIOP SSL Port |
| 4 | P4 Port |
| 5 | P4 HTTP Tunneling Port |
| 6 | P4 SSL Port |
| 7 | IIOP Port |
| 8 | Telnet Port |
| 9 | Monitor Port |
| 10 | JMS Port |

| Index | Server Cluster Service |
|---|---|
| 20+N*5 | Join Port |
| 20+N*5+1 | Debug Port |

For Server Cluster Services, N is the number of the server element

# Secure Network Topology

## Use security zones to improve infrastructure protection



# Agenda

Review: J2EE Security

Network Topology

Usermanagement

Authentication

Secure Storage

HTTPS Client

Secure Store and Forward

Configuration and Destination Service

Virus Scanner

# SAP Web Application Server 6.40

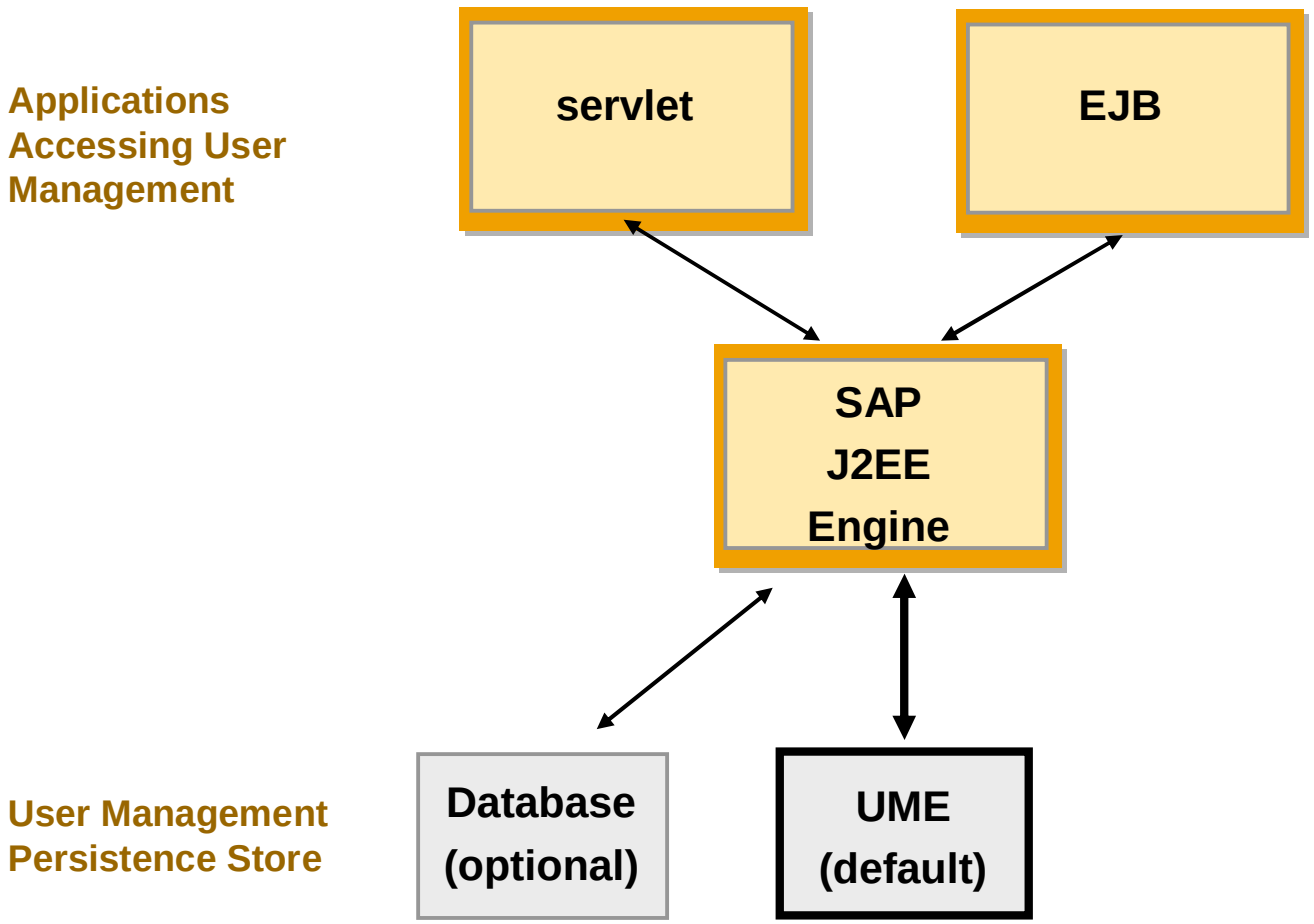
## New Features:

- Pluggable user store concept
  - Local database (as with 6.20)
  - User Management Engine (UME)
- User Management Engine (UME) integration
  - When installing the SAP J2EE Engine 6.40, UME is automatically deployed.

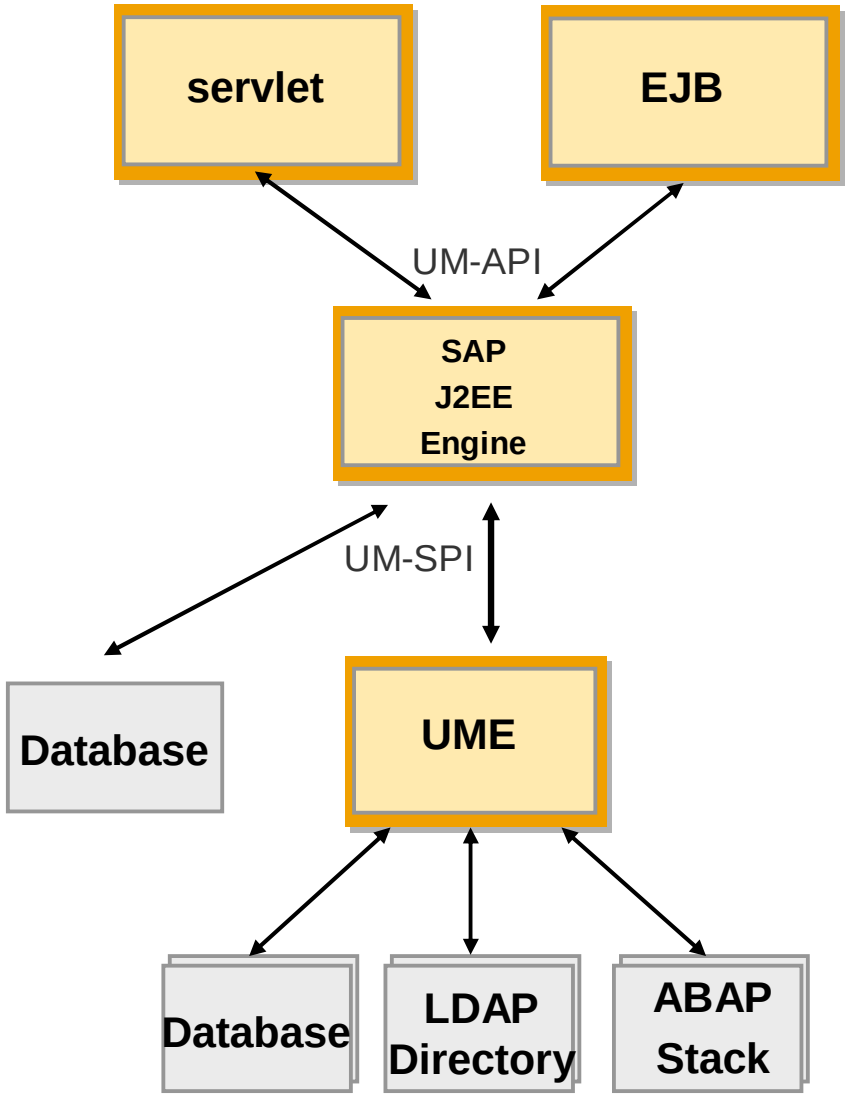
## Benefits of using UME as user store:

- Provides several persistence options
  - LDAP directory
  - ABAP user management (SAP WebAS 6.20 or higher)
  - Database
- Augments user accounts with identity information
- Comes with Web-based administration

# SAP J2EE 6.40 Userstore used by applications



# Usage of UME by applications in SAP J2EE 6.40

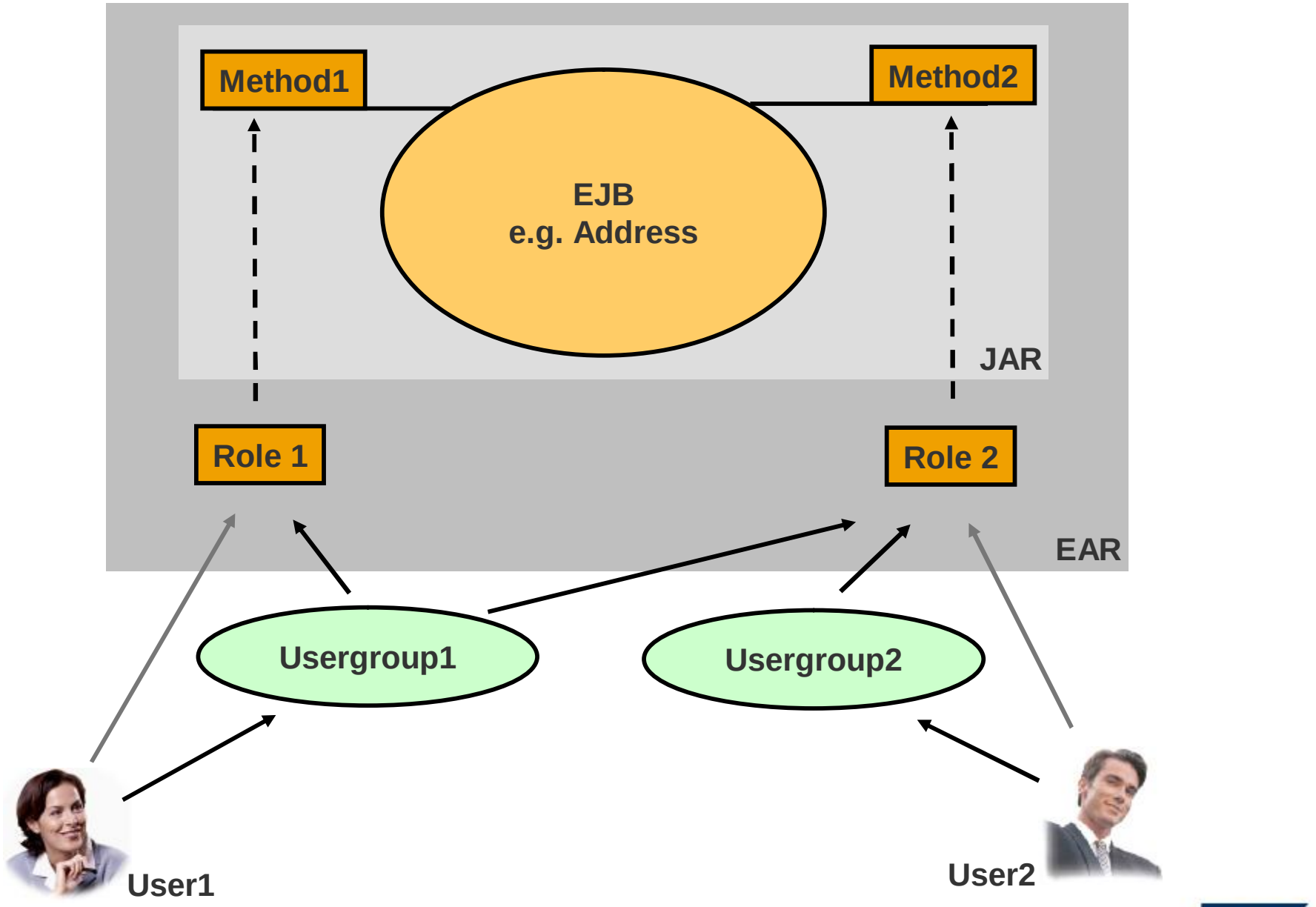


# J2EE Security Security Models

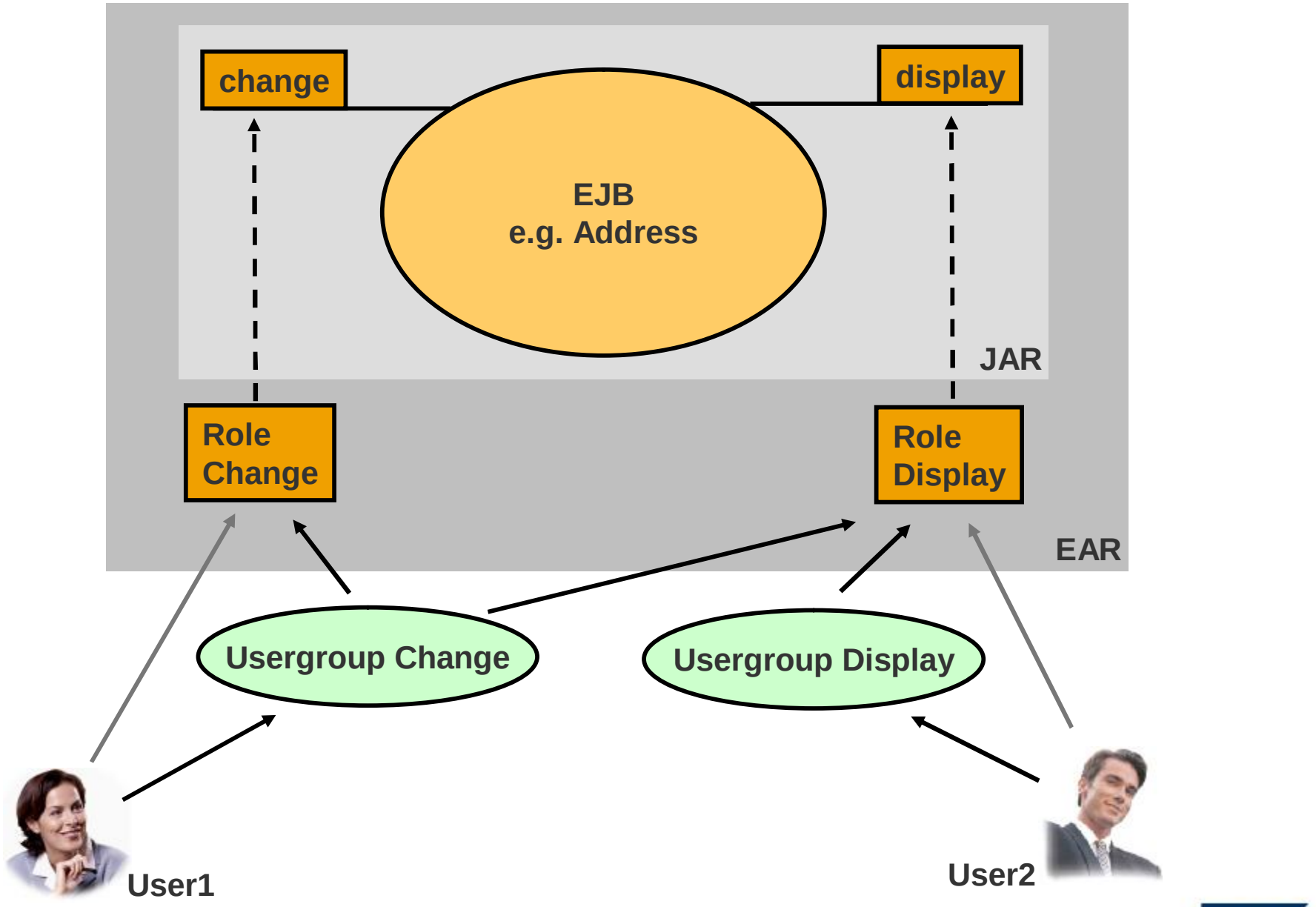
## J2EE supports two different Security Models

- Declarative Security
  - Access control linked to the resource
  - Decouples Access Control from application logic
  - Easy to implement and maintain
- Programmatic Security
  - Access control within Java code
  - More flexible but linked to application logic
  - More work to implement

# J2EE Role Concept Declarative Security



# J2EE Role Concept (Example) Declarative Security



# *Programmatic Security I: Security Role References*

**When security checks are hard-coded within the business code,** *security role references* **should be used.**

**The Bean Provider declares all *security role references* used within the bean's source code within the *deployment descriptor.***

**The Assembler or Deployer assigns *security roles* to defined *security role references.***

**Therefore, different *security roles* can be used within different operational scenarios** without changing the source code**.**

# Programmatic Security II: Security Checks within the Source Code

**The interface *javax.ejb.EJBContext* defines two methods for accessing the security context:**

- *getCallerPrincipal()*
  **Returns the *java.security.Principal* that** identifies **the** user**.**

- *isCallerInRole**(java.lang.String roleName)***
  **Tests if the caller** belongs to **a given** *security role*
  **(*security role references* should be used).**

**Security checks should only be hard-coded for special exceptions.**

**Security checks on the level of accessing methods are implicitly performed by the EJB Container.**

# Programmatic Security II: Hard Coded Security Checks - Sample

```
public class PayrollBean ... {
  EntityContext ejbContext;

  public void updateEmployeeInfo(EmplInfo info) {
    oldInfo = ... read from database;
    // The salary field can be changed only by
    // callers who have the security role "payroll"
    if (info.salary != oldInfo.salary &&
      !ejbContext.isCallerInRole("payroll")) {
      throw new SecurityException(...);
    }
  ...
  }
  ...
}
```

# UME Roles - Permissions and Actions

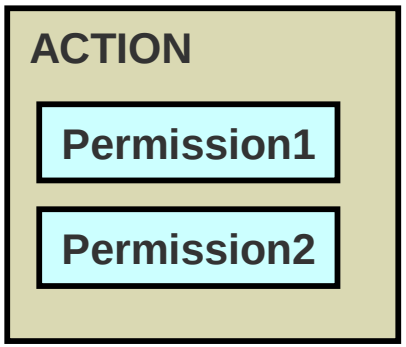
Java applications that want to restrict access can use Java Permissions (derived from a user's role).

An application can define and check many Java Permissions, to provide a flexible access schema.

It has to be easy for Administrators to assign the correct permissions to users à Actions are introduced

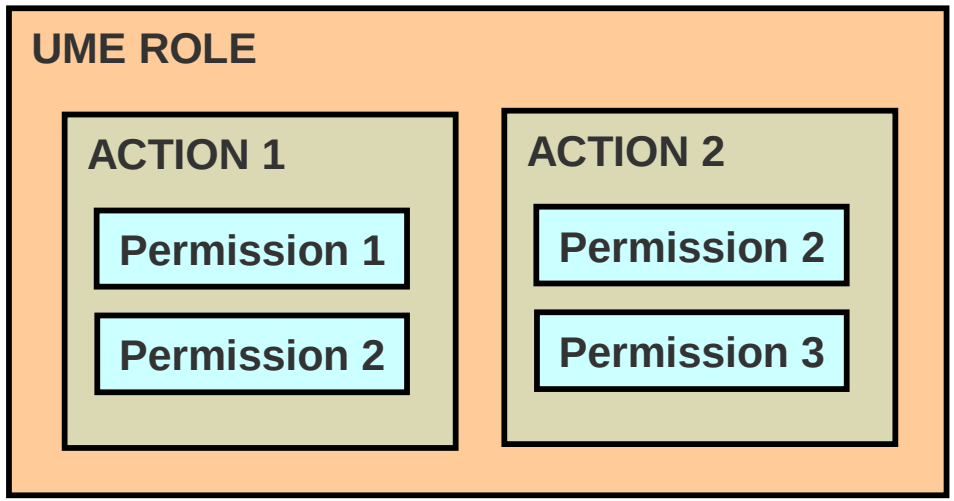
Service Actions describe the possible actions in an application.

Service Actions are a collection of Java Permissions.



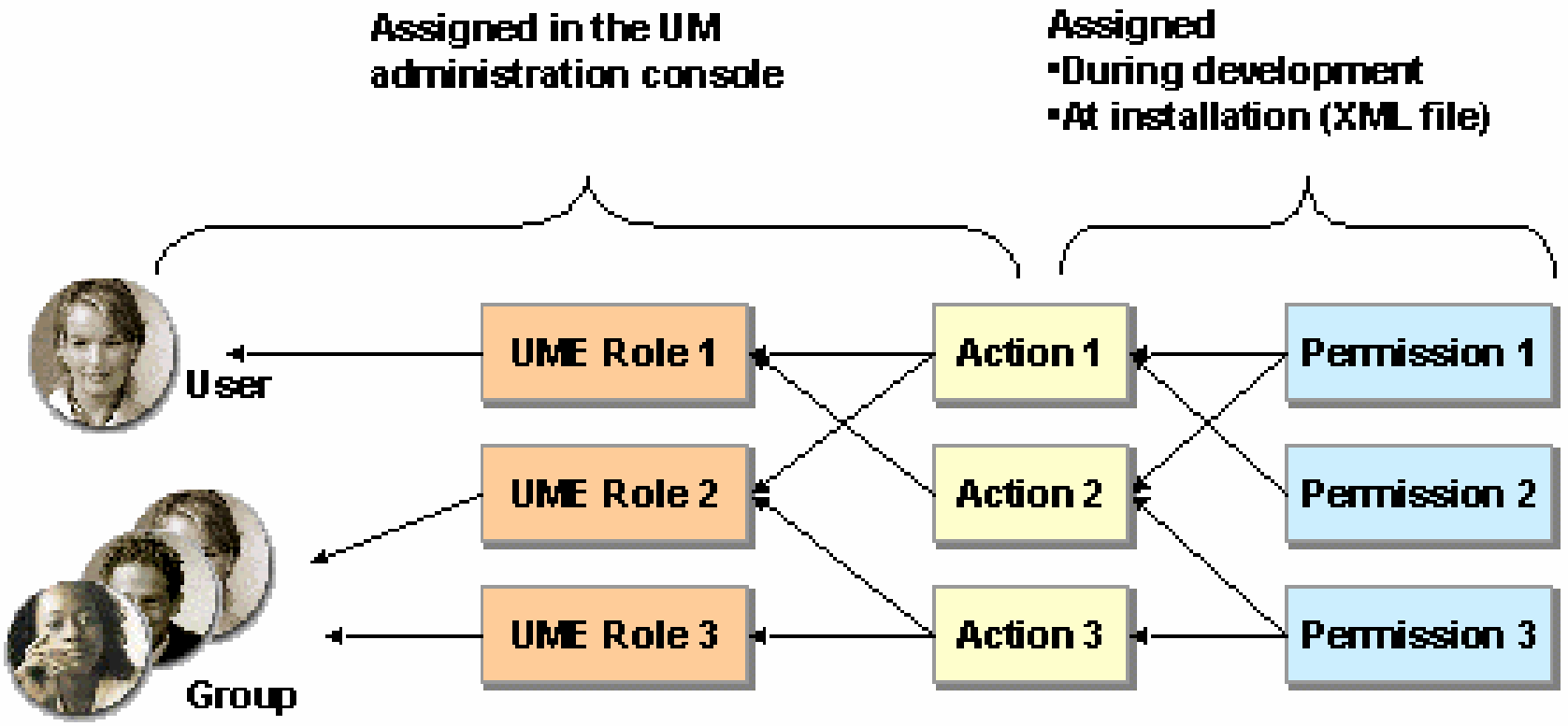
# UME Roles - Actions and Roles

- A service can define any number of Service Actions.
- Actions themselves can't be assigned to users, they have to be grouped into Roles.
- Roles can and probably will consist of Actions from different Services.




- An Administrator assigns these Roles to users.

# UME Roles - Summary



# UME Roles ↔ J2EE Roles

## J2EE roles concept

- **Characteristics of J2EE roles:**
  - Abstract logical grouping of users defined by the developer
  - Defined for one specific application (within the deployment descriptor)
  - Only consists of a name and a description
  - Security role references:
    - additional role name changes (at deploy time)
    - mappings of already existing security roles
- **Declarative security:**
  - Container-managed authorization
- **Programmatic security:**
  - Method(s) to check if a caller of an EBJ or web resource belongs to a specific role.
- **Differences**
  - UME roles are application independent (J2EE role ó UME action)
  - No declarative security at UME role concept,but programmatic security is more powerful.

# Agenda

Review: J2EE Security

Network Topology

Usermanagement

Authentication

Secure Storage

# Authentication

## Select Authentication Methods for your application:

- Basic authentication
- Digest Authentication
- Form based authentication
- Client certificate based authentication

## Authentication via SSL Certificates

- Decide whether certificates are
  - Not used at all
  - Optional
  - Required

# Java Authentication and Authorization Services (JAAS)

**JAAS uses login modules for authentication**

- Login modules get user information via callbacks
- If standard information is not sufficient, SAP proprietary handlers can be used:
  - HttpGetterCallback – used to obtain arbitrary information from the request
  - HttpSetterCallback – used to pass information back to the callback handler which will attach this information to the response
- Standard information available is only User/Pass phrase, all other information requires a Callback

**Further information about JAAS in the documentation or at http://java.sun.com/products/jaas/**

# SAML Support

## Security Assertions Markup Language

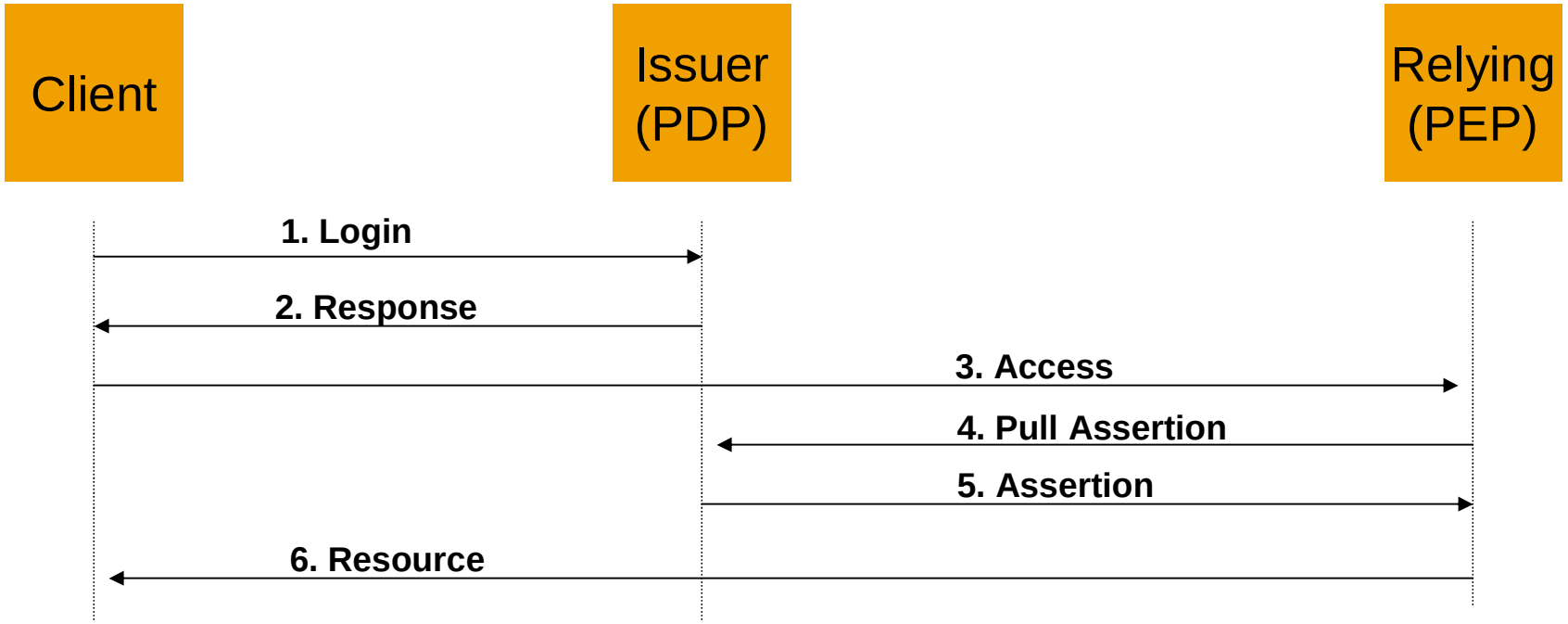
- XML-based framework for exchanging security information
- OASIS Standard

## Only SAML client for authentication available (destination site)

## Support limited

- Only browser artifact scenario supported
- Digital signatures for SOAP documents are ignored
- No support for additional “Condition” elements
- The received assertion may only contain one authentication statement
- The authentication statement must contain the NameIdentifier
- AuthorizationDesicionStatement and AttributeStatement are ignored

# SAML Process




**PDP: Policy Decision Point**
**PEP: Policy Enforcement Point**

# Agenda

Review: J2EE Security

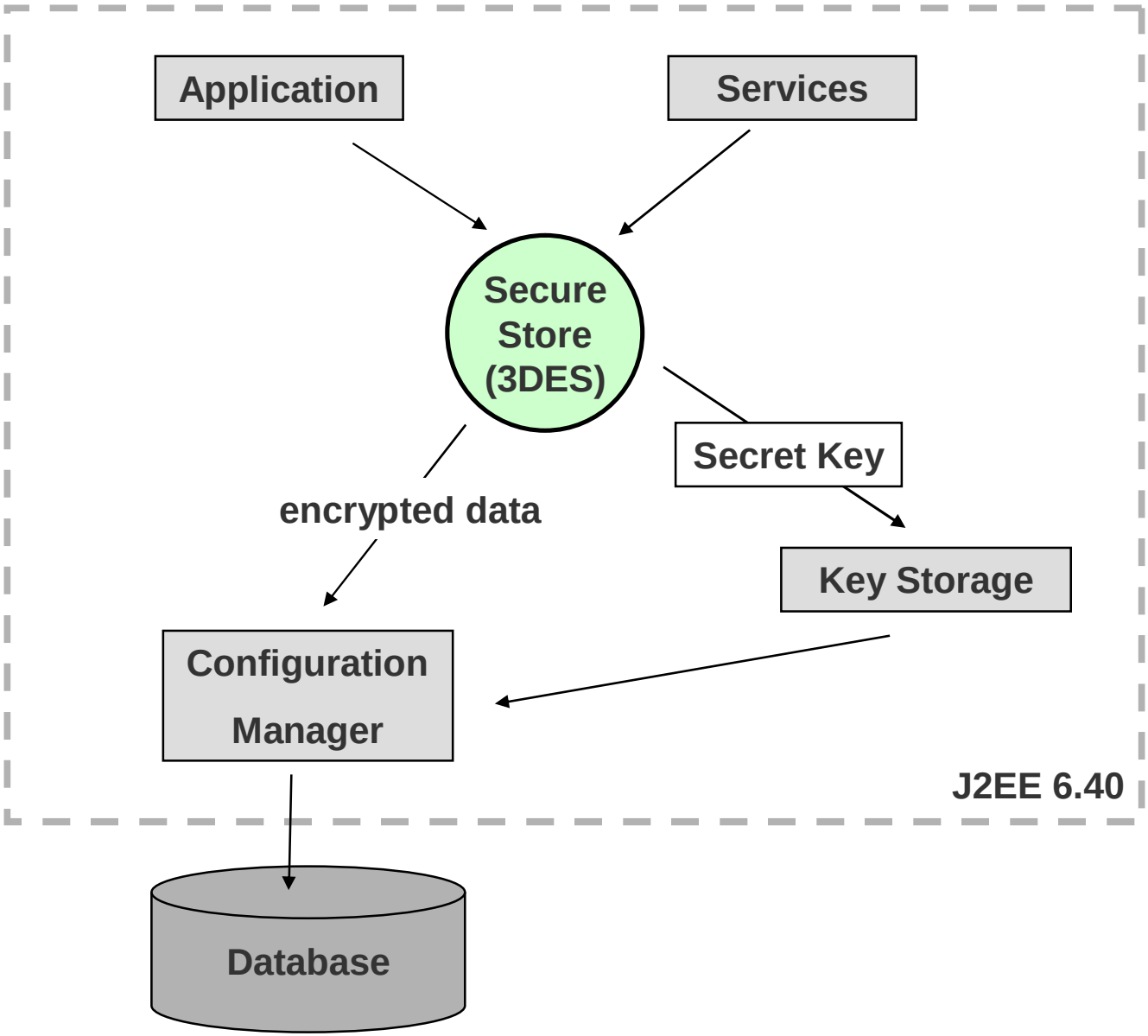
Network Topology

Usermanagement

Authentication

Secure Storage

# Overview Secure Storage



# Secure Storage Service

## Encrypted storage of application data

- Central or decentralized storage possible
- Application passes objects to be protected (e.g. passwords)
- Optional strong symmetric encryption (3DES)
- Individual encryption keys per application, transparent key management for applications
- Service API for applications
  - *SecStore.storeObject(…)*
  - *SecStore.retrieveObject(…)*
  - *SecStore.deleteObject(…)*
- Central administration via Visual Administrator

# Secure Storage Functions

## Packages provided:

- `com.sap.security.core.server.securestorage`
- `com.sap.security.core.server.securestorage.command`
- `com.sap.security.core.server.securestorage.exception`
- `com.sap.security.core.server.securestorage.gui`
- `com.sap.security.core.server.securestorage.remote`

## Important interfaces:

- **`SecureStorageRuntimeInterface` for initialization**

  **from package**
  `com.sap.security.core.server.securestorage`

- **`RemoteSecureStorageClientContextInterface` for data access**

  **from package**
  `com.sap.security.core.server.securestorage.remote`

# Keystore Service

## Central protected storage of application keys

- Central storage of X.509 public/private key pairs and certificates
- Code based permission control access to the different keystore views
- Applications use the KeystoreManager interface
  - ***KeystoreManager.*** *createKeystoreView****(…)***
  - ***KeystoreManager. get****Keystore****(…)***
  - ***KeystoreManager. getProperty(…)***
- Central administration via Visual Administrator

# Questions and Answers

**Email: Security@sap.com**

**http://service.sap.com/Security**

**http://service.sap.com/NetWeaver**

# Security Role Exercise I / Netweaver Developer Studio

1. BASIC Authentication and Realm Name: HelloWorld Protected Area — assign → Web Project Hello World: web.xml

2. Role: HelloWorldGroup — assign → Web Project Hello World: web.xml

3. Security Constraint 0 — assign → Web Project Hello World: web.xml

4. HelloWorld Protected Area
   URL Pattern: /*
   HTTP Method: GET, POST
   HelloWorldGroup
   — assign → Security Constraint 0

5. No Security Role Mapping in web-j2ee-engine.xml !!!!

# Exercise

**Building the WAR file**

- Right click on the HelloWorld_Sec_Web_E and select build WAR file

**Building the EAR file**

- Right click on the HelloWorld_Sec_Ear_E and select build EAR file

**Deploy the EAR file to the local J2EE server**

- Expand the HelloWorld_Sec_Ear_E project node and right click on the HelloWorld_Sec_Ear_E.ear file and select Deploy to J2EE Server

**Test your application link.**

**http://localhost:50000/HelloWorld_Sec_E/HelloWorld.jsp**

# Security Role Exercise II / Deployment Tool of the J2EE Engine

7. In the Security Provider create a New User with password.

8. In the Security Provider for your HelloWorld component add the Role “HelloWorldGroup“

Test it!

# Copyright 2003 SAP AG. All Rights Reserved