User Authentication in SAP Portal and Java Environments: Know Which Options to Pick and How to Wield Them

Nicholas Holshouser

SAP Labs
Contributors

Nicholas Holshouser
Platinum Technical Consultant, SAP Labs

Serge Muts
Senior Technical Consultant, SAP Labs
What We’ll Cover ...

Introduction
The User Management Engine
SAP NetWeaver Portal Authentication
Header-Based Authentication
X.509 Certificate-Based Authentication
Authentication Demos
Troubleshooting
Wrap-up
What We’ll Cover ...

Introduction

The User Management Engine
SAP NetWeaver Portal Authentication
Header-Based Authentication
X.509 Certificate-Based Authentication
Authentication Demos
Troubleshooting
Wrap-up
Learning Objectives

As a result of this session, you will be able to:

- Understand implementation of login stacks, login modules, and control flags in the SAP J2EE Engine
- Configure the authentication methods to suit your security requirements
- Understand the role of authschemes.xml in leveraging SAP J2EE Engine login stacks for SAP NetWeaver Portal authentication
- Configure your SAP NetWeaver Portal to authenticate users using Header Authentication
- Configure your SAP NetWeaver Portal to authenticate users using X.509 client certificates
The User Management Engine (UME) provides a centralized user management for all Java applications

- It can be configured to work with user management data from multiple data sources

The UME is seamlessly integrated in the J2EE Engine of SAP NetWeaver Application Server Java (SAP NetWeaver AS Java) as its default user store

- It can be administrated using the administration tools of SAP NetWeaver AS Java
The J2EE Engine implements the *Java Authentication and Authorization Service (JAAS)* standard to support various authentication methods

- This means you can choose the required authentication mechanisms for your applications

Applications running on the J2EE Engine can either use declarative or programmatic authentication

- Both types of authentication rely on the same underlying technology, login modules, and login module stacks
- Programmatic authentication additionally uses authentication schemes
- SAP ships login modules and authentication schemes to support various authentication mechanisms
The SAP NetWeaver Portal and Security Services

The SAP NetWeaver Portal is a Java application that runs entirely within the NetWeaver Application Server.

As a consequence, SAP NetWeaver Portal can use all the Identity Management and Security services offered by the NetWeaver Application Server (Java), including:

- UME Provider Service
- Security Provider Service
- Key Storage Service
- SSL Provider Service
- Secure Storage Service

User management in the SAP NetWeaver Portal is provided by the UME service in the J2EE Engine.

Authentication is performed by the Security Provider Service.
Introduction

The User Management Engine

SAP NetWeaver Portal Authentication
Header-Based Authentication
X.509 Certificate-Based Authentication
Authentication Demos
Troubleshooting
Wrap-up
The User Management Engine (UME)

The UME provides a comprehensive user store implementation, offering LDAP-enabled directories, RDBMS, and the SAP System as persistence options.

The UME also provides initial identity management features by storing other user attributes in addition to technical account information.

Source:

http://help.sap.com/saphelp_nw04/helpdata/en/57/a21f407b402402e100000000a1550b0/frameset.htm
UME runs as a service in the J2EE Engine of the SAP NetWeaver AS Java and is set up as the default user store of the J2EE Engine.
UME can be configured to read and write user-related data from and to multiple data sources, such as:

- Lightweight Directory Access Protocol (LDAP) directories
- The system database of the J2EE Engine, and
- The ABAP user management of a NetWeaver Application Server
The user management administration console enables administrators to perform routine administration tasks such as creating or searching for users and groups, and assigning users and groups to roles.

- **Security settings**
  - You can define a password policy including settings such as minimum and maximum length of passwords, number of failed logons before a user is locked, and so on.

- **Self-service scenarios**
  - UME provides self-service scenarios that allow users to register themselves as new users or to change their own data (address, password, and so on). It is also possible to set up an approval workflow, whereby administrators approve newly registered users.
What We’ll Cover ...

Introduction
The User Management Engine
SAP NetWeaver Portal Authentication
Header-Based Authentication
X.509 Certificate-Based Authentication
Authentication Demos
Troubleshooting
Wrap-up
SAP NetWeaver Portal authentication verifies the identity of a user who attempts to log in to the portal or to access sensitive portal content.

SAP NetWeaver Portal authentication is tightly integrated with J2EE Engine authentication.
The standard authentication methods offered by the J2EE Engine are:

- Anonymous/guest access
- User ID/password
  - Form-based
  - Basic authentication
- X.509 digital certificates
- SAP Logon Tickets
- External authentication methods
  - HTTP header variable authentication
  - Security Assertion Markup Language (SAML)
  - Through Java Authentication and Authorization Services (JAAS)
    - SPNEGO available – Kerberos as of NW04 SPS15 and NW2004s
Authentication configuration is done in the J2EE Engine configuration

Responsible J2EE service: “security provider”

JAAS Login modules are bundled in “login module stacks” (policy configurations)

A policy configuration can include one or more login modules with a JAAS control flag attached to each module

In case the SAP NetWeaver Portal is used, references to the policy configurations are maintained in the authentication scheme configuration
### JAAS Pluggable Authentication – Login Stacks and Control

#### Flag Values

<table>
<thead>
<tr>
<th>Login Module</th>
<th>Flag</th>
</tr>
</thead>
<tbody>
<tr>
<td>EvaluateLoginTicket</td>
<td>Sufficient</td>
</tr>
<tr>
<td>ClientCert</td>
<td>Sufficient</td>
</tr>
<tr>
<td>BasicPassword</td>
<td>Requisite</td>
</tr>
</tbody>
</table>

**SUFFICIENT** – the login module is not required to succeed. If the authentication is successful, control returns to application; otherwise, the authentication proceeds.

**OPTIONAL** – the login module is not required to succeed. Authentication proceeds down the list if the module has succeeded or has failed.

**REQUIRED** – the login module is required to succeed. Authentication proceeds down the list of modules if the module has succeeded or has failed.

**REQUISITE** – the login module is required to succeed. If successful, the authentication proceeds down the list; otherwise, control returns to the application – that is, the authentication does not proceed.

---

© SAP AG 2006, 19

THE BEST-RUN BUSINESS
**Example Login Module Stack – Ticket**

<table>
<thead>
<tr>
<th>Module</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>com.sap.security.core.server.jaas.EvaluateTicket</code> – Sufficient</td>
<td>pass</td>
</tr>
<tr>
<td><code>BasicPasswordLoginModule</code> – Requisite</td>
<td>* pass</td>
</tr>
<tr>
<td><code>com.sap.security.core.server.jaas.CreateTicket</code> – Optional</td>
<td>* pass</td>
</tr>
</tbody>
</table>

**Overall Authentication:**

<table>
<thead>
<tr>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>pass</td>
</tr>
</tbody>
</table>

---

### Login Module Details

<table>
<thead>
<tr>
<th>Component</th>
<th>Flag</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>com.sap.security.core.server.jaas.EvaluateTicket</code></td>
<td>SUFFICIENT</td>
<td><code>(ume.configuration.active=true)</code></td>
</tr>
<tr>
<td><code>BasicPasswordLoginModule</code></td>
<td>REQUISITE</td>
<td><code>{}</code></td>
</tr>
<tr>
<td><code>com.sap.security.core.server.jaas.CreateTicket</code></td>
<td>OPTIONAL</td>
<td><code>{ume.configuration.active=true}</code></td>
</tr>
</tbody>
</table>

---

© SAP AG 2006, 20
JAAS Pluggable Authentication – Login Modules

- **BasicPasswordLoginModule** – Performs a JSP logon using basic or form authentication (user name and password)

- **ClientCertificateLoginModule** – Performs a certificate logon to the J2EE Engine

- **CertPersisterLoginModule** – This login module automatically maps certificates provided by the user to his or her user account while logging on the first time

- **DigestLoginModule** – Performs a more advanced form of the basic authentication type. Here the password of the user is digested (encoded).

- **SAMLLoginModule** – Uses a SAML browser/artifact profile

- **HeaderVariableLoginModule** – Reads a user ID from the HTTP header variable and then uses this user ID to authenticate the user

- **CSILoginModule** – login performed using the IIOP service

- **EvaluateTicketLoginModule** – Evaluates SAP Logon Tickets

- **CreateTicketLoginModule** – Creates SAP Logon Tickets after successful logon

The SAP J2EE Engine also enables you to define your own stack of login modules or to create, import, and use your own login modules.
The J2EE Engine implements the *Java Authentication and Authorization Service (JAAS)* standard to support various authentication methods

- This enables you to choose the required authentication mechanisms for your applications

Applications running on the J2EE Engine can either use declarative or programmatic authentication

- Both types of authentication rely on the same underlying technology, login modules and login module stacks
- Programmatic authentication additionally uses authentication schemes
- SAP ships login modules and authentication schemes to support various authentication mechanisms
Declarative and Programmatic Authentication

- Explains the difference between declarative (container-based) authentication and programmatic (UME) authentication. The type of authentication that an application uses has consequences for the login module stack it uses and where you configure authentication.

Login Modules and Login Module Stacks

- Provides conceptual and configuration information about login modules and login module stacks. These are part of the Java Authentication and Authorization Services (JAAS) standard.

- Both declarative and programmatic authentication are based on login modules and login module stacks

- Login modules define authentication logic. Login module stacks enable you to define a sequence of authentication logic performed for an application.
Authentication Schemes

- Provides conceptual and configuration information on authentication schemes that are used in programmatic authentication only

Configuring Authentication Mechanisms

- Describes how to configure different types of authentication, including basic authentication, authentication with client certificates, authentication using SAML assertions or SAP Logon Tickets
Declarative and Programmatic Authentication

Declarative authentication (also known as container-based authentication):

- The Web container (in this case, the J2EE Engine) handles authentication
- A component running on the J2EE Engine declares its protected resources and its desired authentication mechanism in its deployment descriptor
- When a protected resource of this component is accessed, the container in which the component runs triggers authentication
Programmatic authentication (also known as UME authentication):

- Components running on the J2EE Engine authenticate directly against the UME using the UME API
- The component explicitly triggers authentication
- The authentication process then is controlled by the authentication framework

Web Dynpro applications and portal iViews always use programmatic (UME) authentication.

J2EE Web applications can use either declarative or programmatic authentication, depending on which the developer decides to use.
## Integration of Authentication

<table>
<thead>
<tr>
<th>Application Type</th>
<th>Type of Authentication</th>
<th>Where is login module stack defined?</th>
</tr>
</thead>
<tbody>
<tr>
<td>J2EE Web applications</td>
<td>Declarative authentication</td>
<td>Declared in the <code>Web.xml</code> deployment descriptor of the J2EE Web application</td>
</tr>
<tr>
<td>J2EE Web applications</td>
<td>Programmatic authentication</td>
<td>This depends on how the application is programmed. Applications can define an authentication scheme in their calls to the API. By default, if they do not define an authentication scheme, these applications use the login module stack referenced by <code>default</code> in the authentication schemes file.</td>
</tr>
<tr>
<td>Web Dynpro applications</td>
<td>Programmatic authentication</td>
<td>Web Dynpro applications always use the login module stack referenced by <code>default</code> in the authentication schemes file</td>
</tr>
<tr>
<td>Portal iViews</td>
<td>Programmatic authentication</td>
<td>An iView property defines which authentication scheme the iView uses. The authentication scheme references a login module stack.</td>
</tr>
</tbody>
</table>
Login Module

- Authentication on the J2EE Engine is performed using predefined authentication classes, referred to as login modules. A login module contains an implementation Java class that defines the authentication logic. For more information, see Login Modules.

Login Module Stack

- On the J2EE Engine, you can use or define groups of login modules that contain different authentication logic. Such groups are referred to as login module stacks.

Each login module stack enables you to choose different combinations of authentication for applications you create or for each of the components on the J2EE Engine with applied security restrictions. For more information, see Login Module Stacks.
Login Module Stacks

In the J2EE Engine, you can use or define groups of login modules that contain different authentication logic.

Such groups are referred to as login module stacks. Each login module stack enables you to choose different combinations of authentication for the applications you create, or for each of the components on the J2EE Engine.
Authentication Templates

Predefined login module stacks, which are also referred to as authentication templates, on the J2EE Engine:

SAP-J2EE-Engine – this is a default configured login module stack that can be used by everyone

Basic – allows for Basic Authentication, supported by the Web container

Client – allows for client certificate authentication, supported by the Web container

Digest – allows for digest authentication, supported by the Web container

Form – allows for form authentication, supported by the Web container

Ticket – used for creating and verifying logon tickets

Evaluation assertion ticket – used for verifying assertion tickets (tickets used between systems)
Authentication schemes are used in UME authentication only

- Portal iViews and Web Dynpro applications always use authentication schemes
- J2EE Web applications may use them depending on whether they were programmed to use UME or container-based authentication

Authentication schemes are defined in the authschemes.xml file, which you can change in the config tool
An authentication scheme is a definition of what is required for an authentication process. This includes:

- The login module stack that is used to determine whether a user is granted access to an application
- The user interfaces that are used to gather the information required to authenticate a user
- Priority, allowing authentication schemes to be ordered
Configuring Authentication Mechanisms

The J2EE Engine is shipped with a range of login modules that support the most common authentication mechanisms. In this section, you find out how to use the login modules to set up the authentication mechanism that you require.

- **Using Basic Authentication (User ID and Password)**
  - By default, the J2EE Engine uses basic authentication for applications that are set up to use basic or form authentication

- **Using Client Certificates for User Authentication**
  - Use client certificate authentication for applications that require a higher level of security

- **Using Security Session IDs for SSO Between Applications**
  - By default, the standard JSESSIONID mechanism is used to exchange the information about the user’s identity between the Web applications when performing SSO

- **Using Logon Tickets for SSO**
  - Use SAP Logon Tickets for SSO in an SAP system environment
Configuring Authentication Mechanisms (cont.)

Using SSO with Resource Adapters

- You can use SSO when working with resource adapters

Using Header Variables or Integrated Windows Authentication for User Authentication

- You can use header variable authentication to delegate user authentication to any external product that authenticates the user and returns an authenticated user ID as part of the HTTP header. Integrated Windows authentication is an example of header variable authentication.
Using SAML Assertions for SSO

- Users can use SAML assertions to access the J2EE Engine. The SAP J2EE Engine accepts SAML assertions for SSO, but it cannot issue such assertions.

Using Anonymous Logon to Access the Portal

- Anonymous logon allows users to access the portal in anonymous mode, without providing any form of authentication. This may be required for external portals that allow Internet users to access anonymous content.
Customer Authentication Modules

This release includes individual login modules for user authentication against a local database, LDAP-enabled directory, or the SAP System.

Extended configuration options are also provided for configuring application-specific authentication schemes.

Source:

http://help.sap.com/saphelp_nw04/helpdata/en/57/a21f407b402402e100000000a1550b0/frameset.htm
Authentication on the Portal

Authentication provides a way of verifying the user’s identity before he or she is granted access to the portal. Once the user has been authenticated, he or she is issued an SAP Logon Ticket that allows him or her to access all the applications, information, and services in SAP NetWeaver Portal using SSO.

- Since many of these applications may contain sensitive data, it is imperative that the user in question can be identified and their identity authenticated.

In the portal, authentication is defined using authentication schemes that are assigned to iViews. Users log on to the portal with a specific authentication scheme, and this is stored in the user’s logon ticket.

- If a user needs to access an iView that requires a stronger authentication scheme, he or she must re-authenticate as specified by the stronger authentication scheme.
Introduction
The User Management Engine
SAP NetWeaver Portal Authentication
Header-Based Authentication
X.509 Certificate-Based Authentication
Authentication Demos
Troubleshooting
Wrap-up
HTTP Header Authentication

Authentication Authority (intermediate)

Initial logon

Access

Web Access Management (WAM) Token

Identity information within HTTP request

Access

EP

ERP

Workflow

CRM

Internet

ESS

Groupware

THE BEST-RUN BUSINESSES RUN SAP
Header-Based Authentication

There is one login module that handles both

- Integrated Windows authentication (NTLM and Kerberos)
- HTTP header variable authentication (for integration of external WAM products)

Class name

- com.sap.security.core.server.jaas.HeaderVariableLoginModule

Configuration in the “Security Provider” service of the J2EE Engine

Login module options (parameters)

<table>
<thead>
<tr>
<th>Name</th>
<th>Values</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ume.configuration.active</td>
<td>true</td>
<td>true</td>
</tr>
<tr>
<td>Header</td>
<td>&lt;header name&gt;</td>
<td>REMOTE_USER</td>
</tr>
<tr>
<td>windows_integrated</td>
<td>true/false</td>
<td>false</td>
</tr>
<tr>
<td>Domain</td>
<td>comma separated list</td>
<td>&lt;empty&gt;</td>
</tr>
</tbody>
</table>
Initial authentication is done with third-party WAM

After initial authentication, the WAM issues an authentication header

- Apache will be used to add a header for authentication to simulate this

Configure the login stack in the J2EE Engine to accept header-based authentication

- Add HeaderVariableLoginModule if necessary
- Add HeaderVariableLoginModule to the ticket login stack
- Add CreateTicketLoginModule to the ticket login stack
- Test the header-based authentication

Prerequisites:

- The UME user ID and header user ID must be identical
## Example of Header-Based Authentication Login Stack

<table>
<thead>
<tr>
<th>Login Modules</th>
<th>Flag</th>
<th>Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>EvaluateTicket</td>
<td>SUFFICIENT</td>
<td>{ume.configuration.active=true}</td>
</tr>
<tr>
<td>HeaderVariable</td>
<td>OPTIONAL</td>
<td>{ume.configuration.active=true, Header=&lt;header_name&gt;}</td>
</tr>
<tr>
<td>CreateTicket</td>
<td>SUFFICIENT</td>
<td>{ume.configuration.active=true}</td>
</tr>
<tr>
<td>BasicPassword</td>
<td>REQUISITE</td>
<td>{}</td>
</tr>
<tr>
<td>CreateTicket</td>
<td>OPTIONAL</td>
<td>{ume.configuration.active=true}</td>
</tr>
</tbody>
</table>
Security Measures for HeaderVariableLoginModule

Block risk of user impersonation!
- Be aware of header spoofing

Safeguard J2EE Engine HTTP(S) ports from direct access by users
- Prevent opportunity to bypass the proxy for J2EE Engine access

Configure SSL with mutual authentication between the Web server and the J2EE Engine
Header-Based Authentication

Demo
These steps simulate inserting a user ID in the header; this is normally performed by a WAM product

- **Browse to** `c:\Program Files\Apache Group\apache2\conf` and open `httpd.conf` in a text editor
- **Verify that** Apache runs under hostname `apache` (ServerName)
- **Add RequestHeader to** `httpd.conf` (e.g., under ServerSignature section):
  - `RequestHeader add HTTP_USER “AGS350_01”`
- **Add RewriteRules to** `httpd.conf`
  - `<IfModule mod_rewrite.c>
  - Rewrite Engine On
  - RewriteLog logs/rewrite.log
  - RewriteLogLevel 3 (use 0 in production setting)
  - #EP Rules
  - RewriteRule ^/(irj.*) http://portal:50000/$1?%{QUERY_STRING} [P,L]
  - RewriteRule ^/(logon.*) http://portal:50000/$1?%{QUERY_STRING} [P,L]
  - </IfModule>`
Apache Setup (cont.)

Add proxy rules to httpd.conf

- `<IfModule mod_proxy.c>`
- ProxyRequests Off
- ProxyPreserveHost On

- ProxyPassReverse /irj http://portal:50000/irj
- ProxyPassReverse /logon http://portal:50000/logon
- `</IfModule>`

Save the httpd.conf file and restart Apache

Test proxy by typing `http://apache:8080/irj` in Internet Explorer (IE)

Verify that the portal login page should come up with `http://portal:50000/irj/portal`
Add HeaderVariableLoginStack to the Security Store in the Visual Administrator:

- Start the Visual Administrator from Start – Programs – SAP J2EE Engine P25 – Visual Administrator
- In the Visual Administrator, choose Security Provider
- Choose the User Management tab, click “edit,” and choose Manage Security Stores
- The currently active user store and the login modules for that user store are displayed
- Choose Add-Login Module
- A dialog box appears, prompting you to choose an editor for the login module option. Choose OK.
- A dialog box appears, prompting you to add a login module
Add HeaderVariableLoginStack to the Security Store in the Visual Administrator: (cont.)

- Fill in the fields as follows:
  - Class Name com.sap.security.core.server.jaas.HeaderVariableLoginModule
  - Display Name HeaderVariableLoginModule
  - Choose OK

The HeaderVariableLoginModule now appears in the list of login modules for the active user store
Add HeaderVariableLoginModule to Ticket authentication template in the Security Store

- In the Visual Administrator, choose Security Provider
- Choose Policy Configurations → Authentication
- Add the login module HeaderVariableLoginModule to the ticket login module stack

Change the parameters of the HeaderVariableLoginModule:

- Set the options for the HeaderVariableLoginModule as follows:
  1. Ume.configuration.active = true
  2. Header = HTTP_USER

Complete the login stack as displayed on the next slide
Test Header-Based Authentication

- Start HttpWatch in Internet Explorer (icon in tool bar)
- Browse to URL http://portal:50000/irj; verify that the basic login form is shown
- Verify headers in HttpWatch and clear HttpWatch
- Browse to URL http://apache:8080/irj; verify that the login is direct using headers
- Check whether the portal URL gets passed back to the client when you click on “User Management”
- Open http.conf and change ProxyPreserveHost to “Off.” Restart Apache and retest logging into http://apache:8080/irj with HttpWatch active. Click on User Management and see if the hostnames listed in HttpWatch are consistent.
- Consider the security implications for external user access
Introduction  
The User Management Engine  
SAP NetWeaver Portal Authentication  
Header-Based Authentication  
X.509 Certificate-Based Authentication  
Authentication Demos  
Troubleshooting  
Wrap-up
Authentication: X.509 Client Certificates

Every user is issued an X.509 certificate

The certificate is used for logging on to the system

The J2EE Engine maps the certificate to a valid user account

Two login modules are used:

1. ClientCertLoginModule – Performs a certificate logon to the J2EE Engine

2. CertPersisterLoginModule – Automatically maps certificates provided by the user to his or her user account while logging on the first time

Alice
X.509 Client Certificate Authentication – Prerequisites

- Users import valid client certificate into the Web browser
- The J2EE Engine is configured to support SSL
- The public-key certificates belonging to the users exist as files in the file system with either the extension .crt (DER-encoded or Base-64-encoded) or .cert (Base-64-encoded)
- The issuing Certificate Authority’s (CA’s) root certificate either exists in the TrustedCAs view in the Key Storage service or is available in the file system as a DER-encoded or Base-64-encoded certificate
- UME property ume.logon.allow_cert is set to true
Configuring the J2EE Engine to Accept Client Certificate Authentication – Overview

Key Storage Service
- Import Certificate Authority’s (CA’s) root certificate as a certificate entry in the TrustedCAs view

SSL Provider Service
- Request or require client certificates for authentication
- Import CA root certificate in Trusted Certification Authorities list

Security Provider Service
- Adjust login module stacks to accept client certificates using ClientCertLoginModule and CertPersisterLoginModule

Result: Client certificates can be used to authenticate users. On first logon, the user will be given the option of mapping the user ID to the certificate.
Certificate Authentication

Demo
Check the sheet titled “Last-Minute Tips” provided with this session for details relevant to the exercise

Users import valid client certificate into the Web browser
  - Import the provided client certificate into IE (see “Last-Minute Tips” sheet)

The J2EE Engine is configured to support SSL
  - Already set up in advance of this session
The issuing CA’s root certificate either exists in the TrustedCAs view in the Key Storage service or is available in the file system as a DER-encoded or Base-64-encoded certificate (see “Last-Minute Tips” sheet)

- Import the provided CA root certificate into the J2EE Engine

Adjust the login stack to accept X.509 certificates

- Add the CertPersisterLoginModule to the Security Store (detailed in next slide)
- Add the ClientCertLoginModule and CertPersisterLoginModule to the login stack

UME property ume.login.allow_cert is set to true

Test the logon
Import Valid Client Certificate Into the Web Browser

1. In MS-IE, go to Tools, Internet Options, and select Content tab
2. Click on certificate; on the Personal tab select Import
3. Follow the steps in the wizard to import the Personal Information Exchange file AGS350_01.p12
4. Use rigteched for the password
5. You should get an “Import was successful” message and the certificate will show up in the list; close the window
6. In Internet Options, select the Security tab, and choose “Local intranet” and “Custom Level”
7. Disable the “Don’t prompt for client certificate selection when no certificates or only one certificate exists” option
Import the Root Certificate CA.CRT of the CA Issuing the Client Certificates
Import the Root Certificate CA.CRT of the CA Issuing the Client Certificates – Result
Client Certificate Settings in the SSL Service
Add CertPersisterLoginModule

In the Visual Administrator, choose Security Provider. Choose the User Management tab, switch to change mode, and choose Manage Security Stores.

The currently active user store and the login modules for that user store are displayed. As the CertPersisterLoginModule requires the UME as the user store, you can only perform the following steps if the UME user store is activated.

Choose Add Login Module. A dialog box, prompting you to choose an editor for the login module option, appears. Choose OK.

A dialog box prompting you to add a login module appears

Fill in the fields as follows:

- **Class Name:** com.sap.security.core.server.jaas.CertPersisterLoginModule
- **Display Name:** CertPersisterLoginModule
- Choose OK

CertPersisterLoginModule now appears in the list of login modules for the active user store
Modify Ticket Login Module Stacks for Services That Accept Client Certificate Authentication
In the pop-up window, select certificate AGS350_01

User = AGS350_01, password = abc123
Check user AGS350_01 in the Security Provider Service

Verify that the certificate mapping was successful
Test X.509 Certificate Authentication – Subsequent Login

Open a new browser window with URL https://portal:50001/irj

Now login should be successful
Introduction
The User Management Engine
Sap NetWeaver Portal Authentication
Header-Based Authentication
X.509 Certificate-Based Authentication
Authentication Demos
Troubleshooting
Wrap-up

What We’ll Cover ...
Basic Troubleshooting for Authentication

- Isolate the error by reviewing the appropriate logs
  - Use the Log Viewer service of the Visual Administrator
  - Check default.trc and security logs
  - Increase logging level using the Log Configurator service

- Understand where errors come from
  - Is the error from an intermediate device or software?

- Validate your configuration
  - Use a second set of eyes to review the system configuration and check for errors in the setup before you begin debugging
  - Document the setup procedure you used so that you can check it for accuracy and so that it is repeatable if it works properly

- SEARCH for SAP Notes to see if your problem is already solved!
Introduction
The User Management Engine
SAP NetWeaver Portal Authentication
Header-Based Authentication
X.509 Certificate-Based Authentication
Authentication Demos
Troubleshooting
Wrap-up
For the most recent changes in UME, see SAP Note 720590 User Management Engine (UME) on WAS 6.30 and higher

For information on configuring UME, see UME Configuration. Here you can find information on configuring the data sources that UME uses to read and write user management data, and other configuration options.

For information on administration with UME, see UME User Administration. The UME provides an administration console for performing administrative tasks such as searching for and creating users, groups, and roles.

For reference material on UME, see UME Reference. This includes information on UME properties and configuration files.

For information on UME tools in the portal, see User Administration and User Management Configuration in the Portal Administration Guide
Resources (cont.)

Public Web

www.sap.com
NetWeaver Developer’s Guide: www.sdn.sap.com/sdn/developersguide.sdn*
SAP Developer Network: www.sdn.sap.com*
SAP Customer Services Network: www.sap.com/services/

Related SAP Education Training Opportunities

http://www.sap.com/education/

* Requires login credentials to the SAP Service Marketplace
Related Content from SMP

http://service.sap.com/NW04*
http://service.sap.com/NW2004S*
http://service.sap.com/security*

(Check under Security in Detail -> Secure User Access -> Authentication and SSO for How-To Guides)

* Requires login credentials to the SAP Service Marketplace
<table>
<thead>
<tr>
<th>Note ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>720590</td>
<td>User Management Engine (UME) on WAS 6.30 and higher</td>
</tr>
<tr>
<td>843061</td>
<td>NW04: SSO2 ticket generation and ticket evaluation in Java</td>
</tr>
<tr>
<td>818947</td>
<td>UME Consistency Check Tool</td>
</tr>
<tr>
<td>718383</td>
<td>NW04: Supported UME Data Sources and Change Options</td>
</tr>
<tr>
<td>701205</td>
<td>Single Sign-On using SAP Logon Tickets</td>
</tr>
<tr>
<td>812047</td>
<td>NW04: SSO to J2EE 6.40 fails</td>
</tr>
<tr>
<td>869852</td>
<td>Central Note for UME Web Dynpro UIs</td>
</tr>
<tr>
<td>894958</td>
<td>NW04s(700) Central Note for Security</td>
</tr>
<tr>
<td>894170</td>
<td>NW04s(700) Central Note for User Management Engine (UME)</td>
</tr>
<tr>
<td>894446</td>
<td>ECO: Using UME logon application in SAP e-commerce</td>
</tr>
<tr>
<td>891151</td>
<td>UME and EP 7.0 support in SAP E-Commerce applications</td>
</tr>
<tr>
<td>757692</td>
<td>Changing the hostname for J2EE Engine 6.40 installation</td>
</tr>
<tr>
<td>711093</td>
<td>Release Restriction Note for Web AS 6.40</td>
</tr>
</tbody>
</table>
The central hub for the SAP technology community

- Everyone can connect, contribute and collaborate- consultants, administrators and developers
- Focus around SAP NetWeaver and SAP xApps

High quality of technical resources

- Articles, how-to guides, weblogs, collaborative areas, discussion forums and downloads, toolkits and code-samples

A collaboration platform, not a one-way street

- SAP experts from customers, partners and SAP

SDN is powered by SAP NetWeaver™

- Built on the SAP Enterprise Portal
- Featuring collaboration capabilities of SAP Knowledge Management
7 Key Points to Take Home

• The User Management Engine (UME) is a core component of the SAP NetWeaver AS Java

• Authentication is highly configurable

• The UME uses JAAS to support multiple authentication methods

• X.509 certificates are the most secure authentication method

• Using SSL is required for X.509 certificates

• Header authentication requires further configuration at the network level to fully secure the Web AS

• Always use the latest SAP Notes and guides whenever you do configuration on the UME
Your Turn!

Questions?

How to contact me:
nicholas.holshouser@sap.com