JAAS - Leveraging External Authentication Based on Industry Standards

THE BEST-RUN BUSINESSES RUN SAP

JAAS Overview

Authentication on the SAP J2EE engine

Developing an authentication module

Deployment and Configuration

JAAS Overview

Authentication on the SAP J2EE engine

Developing an authentication module

Deployment and Configuration

Pluggable Authentication - JAAS

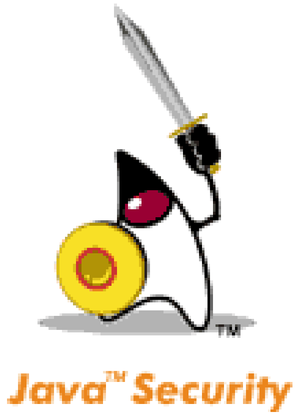
Interface defined by Java Authentication and Authorization Service (JAAS) standard

As of JDK 1.4 integral part of J2SE

Access control based on user credentials

User-centric approach with two components:

- Authentication (-> login modules)
- Authorization



http://java.sun.com/products/jaas

Login Modules, Login Context

Login Modules

- Application independent
- Represent the “authentication part” of JAAS
- Implement a specific type of authentication technology (UserID/Password, certificate, ..)
- Are plugged in under the application

Login Context

- Application specific
- Here the login modules are called

Subject, Principal, Credentials

javax.security.auth.Subject

- Represents source of a request
- May be any entity (person, service, …)
- May have many principals
- May own credentials

Principal

- Associated with a subject (after successful authentication)
- Represents subject’s identity

Credentials

- Not part of core JAAS class library
- Public credentials (e.g. public key)
- Private credentials (e.g. private key)

Login Module Configuration

Control Flags*

- **Required - The login module is required to succeed. If it succeeds or fails, authentication still continues to proceed down the login module list**
- **Requisite - The login module is required to succeed. If it succeeds, authentication continues down the login module list. If it fails, control immediately returns to the application (authentication does not proceed down the login module list).**
- **Sufficient - The login module is not required to succeed. If it does succeed, control immediately returns to the application (authentication does not proceed down the login module list). If it fails, authentication continues down the login module list**
- **Optional – The login module is not required to succeed. If it succeeds or fails, authentication still continues to proceed down the login module list**

Options

- **Additional configuration parameters for login module**
- **e.g. debug = true**

*** Source: http://java.sun.com/products/jaas/index-14.html**

JAAS Login Module – Authentication Success

Login Module 1 - required	pass	pass	pass	pass	fail	fail	fail	fail
Login Module 2 - sufficient	pass	fail	fail	fail	pass	fail	fail	fail
Login Module 3 - requisite	*	pass	pass	fail	*	pass	pass	fail
Login Module 4 - optional	*	pass	fail	*	*	pass	fail	*
Overall Authentication	pass	pass	pass	fail	fail	fail	fail	fail

Methods in a JAAS Login Module (1)

initialize()

- ➔**The initialize method is called to initialize the login module with the relevant authentication and state information.**

login()

- ➔**The login method is called to authenticate a Subject. This is phase 1 of authentication.**

commit()

- ➔**The commit method is called to commit the authentication process. This is phase 2 of authentication when phase 1 succeeds. It is called if the LoginContext's overall authentication succeeded (that is, if the relevant REQUIRED, REQUISITE, SUFFICIENT and OPTIONAL login modules succeeded).**

Source: http://java.sun.com/j2se/1.4.2/docs/guide/security/jaas/JAASLMDevGuide.html

Methods in a JAAS Login Module (2)

abort()

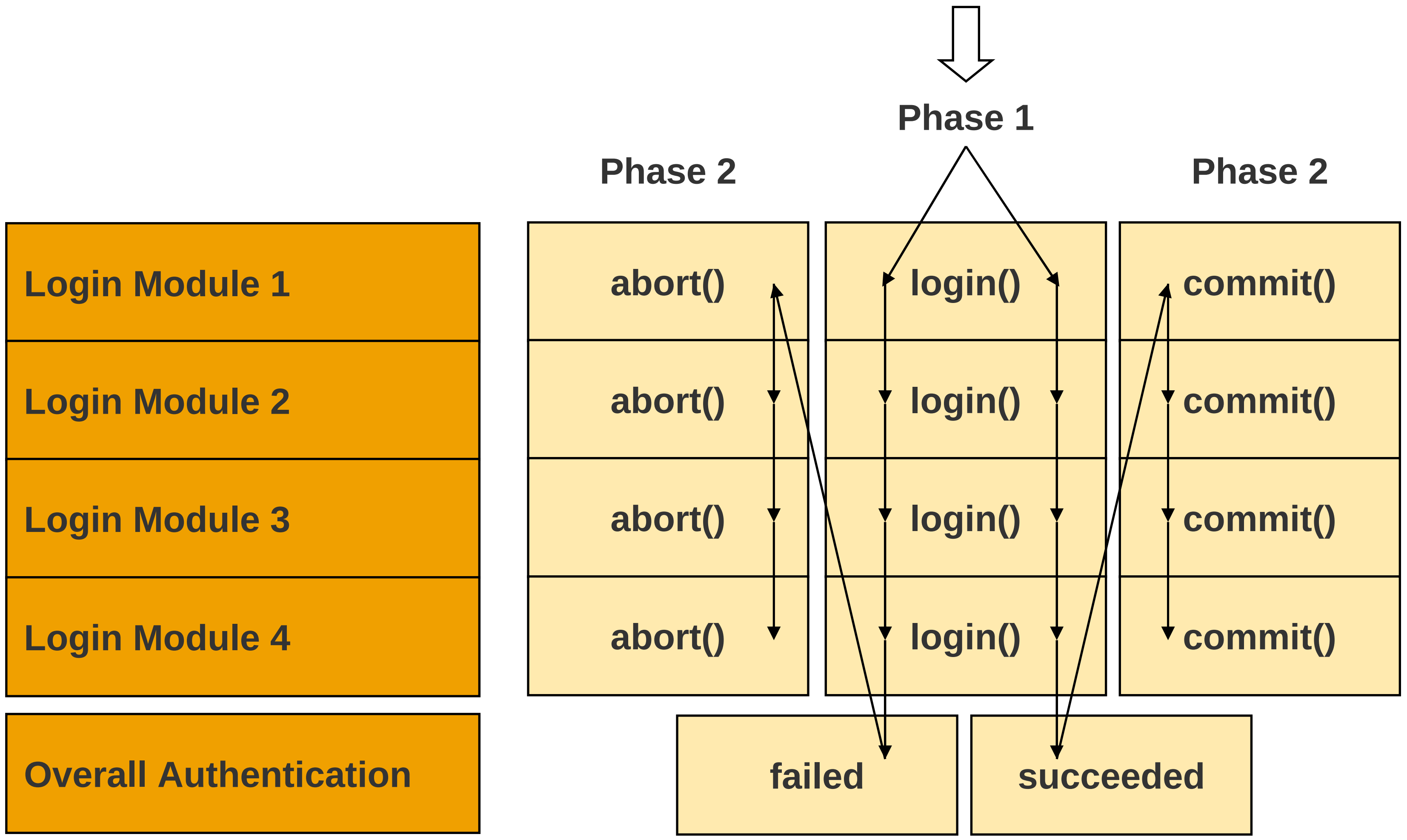
- ➔The abort method is called to abort the authentication process. This is phase 2 of authentication when phase 1 fails. It is called if the Login Context's overall authentication failed.

logout()

- ➔The logout method is called to log out a Subject.

Source: http://java.sun.com/j2se/1.4.2/docs/guide/security/jaas/JAASLMDevGuide.html

Login Module Authentication Process



JAAS Overview

Authentication on the SAP J2EE engine

Developing an authentication module

Deployment and Configuration

Authentication Methods on J2EE Engine

Anonymous/guest access

User ID / password

- Form-based
- Basic authentication

X.509 digital certificates

SAP Logon Tickets

External authentication methods

- SAML
- HTTP header variable authentication
- Others through JAAS interface

External authentication providers

- Integrated Windows Authentication (via IIS + SAP IISProxy)
- EAM products

Authentication – General

Applications running on the J2EE Engine have two options for authenticating users:

- **Container-based authentication:**
  - **The J2EE Engine handles authentication.**
  - **Applications running on the J2EE Engine run in anonymous mode and assume that the J2EE Engine handles authentication.**
- **UME-based authentication:**
  - **Applications running on J2EE Engine authenticate directly against SAP User Management Engine (UME) using the UME API.**

An integration of these two types of authentication is supported. Calls to the APIs of both the J2EE Engine and UME return the authenticated user.

Authentication Configuration - General

Authentication configuration is done in the J2EE Engine configuration in the responsible J2EE Service: “Security Provider”

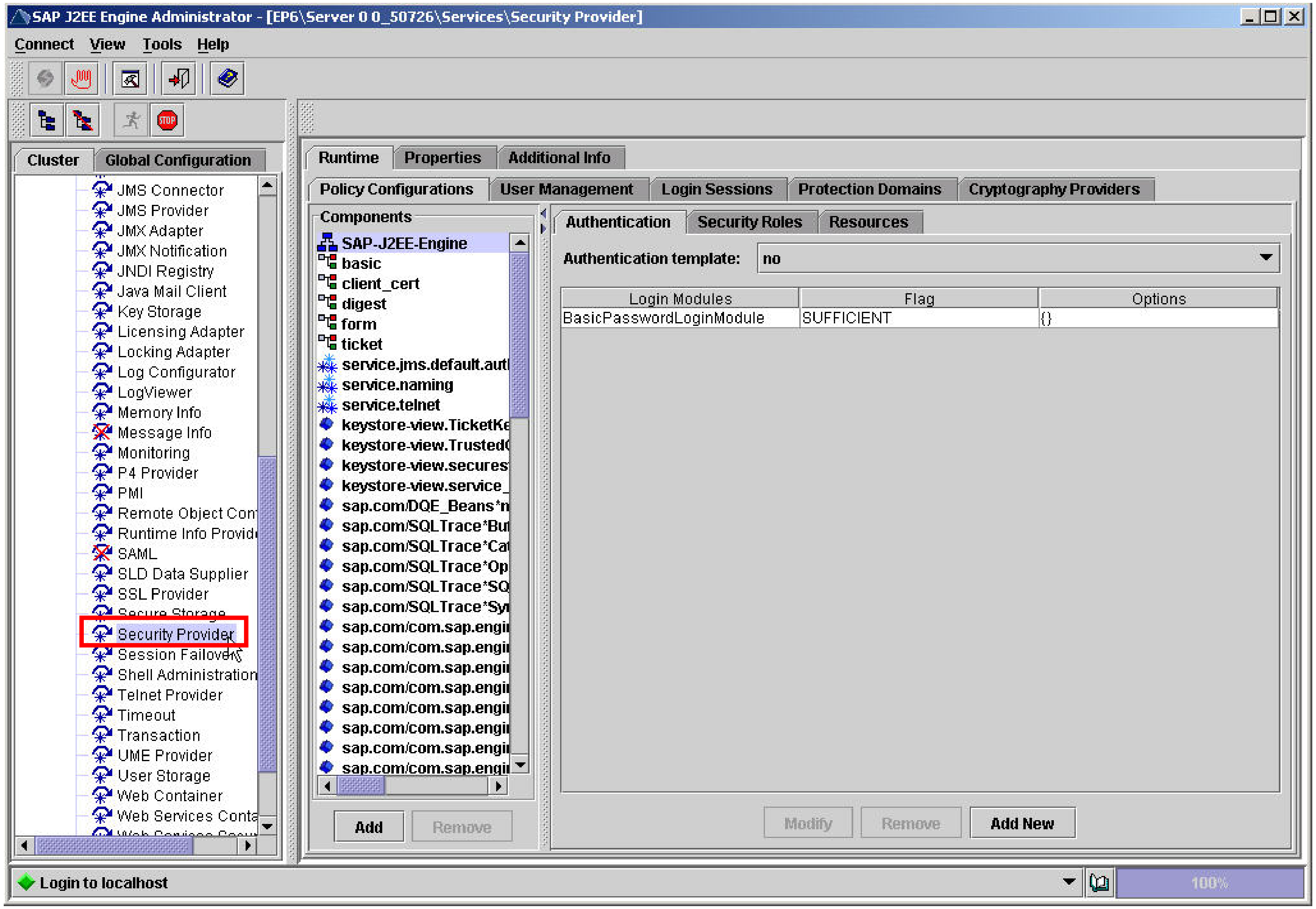
JAAS Login modules are bundled in “login module stacks”

The login module stack can include one or more login modules with some options and JAAS control flags attached to each module

The login module stack belongs to a policy configuration

In cases where UME-based authentication is used, references to the policy configurations are maintained in the authentication scheme configuration. The default is the configuration “ticket”

Authentication Configuration – Security Provider

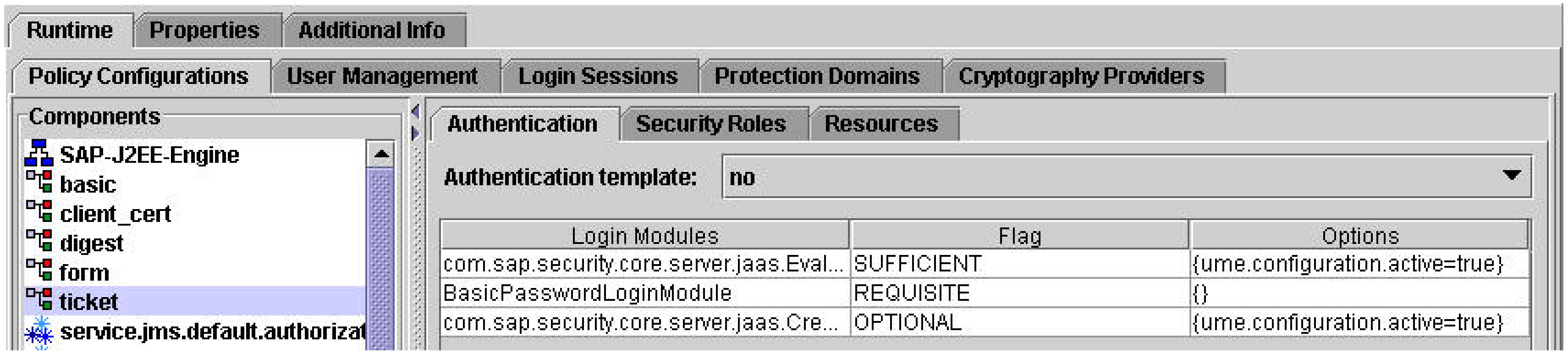


Authentication template: no

Login Modules	Flag	Options
BasicPasswordLoginModule	SUFFICIENT	{}

Example Login Module Stack - Ticket

com.sap.security.core.server.jaas.EvaluateTicket - Sufficient	pass	fail	fail	fail
BasicPasswordLoginModule - Requisite	*	pass	pass	fail
com.sap.security.core.server.jaas.CreateTicket - Optional	*	pass	fail	*
Overall Authentication	pass	pass	pass	fail



Authentication template: no

Login Modules	Flag	Options
com.sap.security.core.server.jaas.Eval...	SUFFICIENT	{ume.configuration.active=true}
BasicPasswordLoginModule	REQUISITE	{}
com.sap.security.core.server.jaas.Cre...	OPTIONAL	{ume.configuration.active=true}

Available Login Modules (1)

BasicPasswordLoginModule

- User name and password
- Basic or form-based authentication

DigestLoginModule

- More advanced form of the basic authentication type
- Password of the user is in digest form (encoded)

ClientCertificateLoginModule

- X.509 certificate logon

SAMLLoginModule

- Uses an SAML Browser/Artifact Profile

HeaderVariableLoginModule

- User name in HTTP header
- Used for Integrated Windows Authentication and EAM products

Available Login Modules (2)

EvaluateTicketLoginModule

- **Evaluates SAP Logon Tickets**

CreateTicketLoginModule

- **Creates SAP Logon Tickets**

SecuritySessionLoginModule

- **Uses tickets generated by security service on the engine**

Available Service Login Modules

EvaluateAssertionTicketLoginModule

- Verifies SAP Authentication Assertion tickets

CSILoginModule

- Login performed using the IIOP service

CallerImpersonationMappingLoginModule

- Caller Impersonation Authentication (JCA)

ConfiguredIdentityMappingLoginModule

- Configured Identity Mapping Authentication (JCA)

CredentialsMappingLoginModule

- Credentials Mapping Authentication (JCA)

PrincipalMappingLoginModule

- Principal Mapping Authentication (JCA)

Authentication Schemes (UME) - Overview

Define the authentication process

- Credentials to be supplied
- User interaction required (i.e. logon screens) – EP 6.0 only
- Priority of the authentication scheme (how strong it is)

Allows enforcement of different authentication mechanisms for different content (e.g. iViews)

Re-authentication required if the content requires a “stronger” authentication scheme

For EP the authentication scheme of an iView can be defined in the property editor of the Portal Content Studio

- Property AuthScheme
- Default value “default”

Authentication scheme is written into the SAP Logon Ticket

Example Authentication Scheme Configuration

```
<?xml version="1.0" encoding="UTF-8"?>
<document>
  <authschemes>
    <authscheme name="uidpwdlogon">
      <!– multiple login modules can be defined -->
      <authentication-template>
        <loginModuleName>
          ticket
        </loginModuleName>
      </authentication-template>
      <priority>20</priority>
      <frontendtype>2</frontendtype>
      <frontendtarget>com.sap.portal.runtime.logon.standard</frontendtarget>
    </authscheme>
…
  </authschemes>
  <authscheme-refs>
    <authscheme-ref name="default">
      <authscheme>uidpwdlogon</authscheme>
    </authscheme-ref>
  </authscheme-refs>…
```

new setting!:
authentication-template

Definition of an authenti-cation template

Definition of one authenti-cation scheme

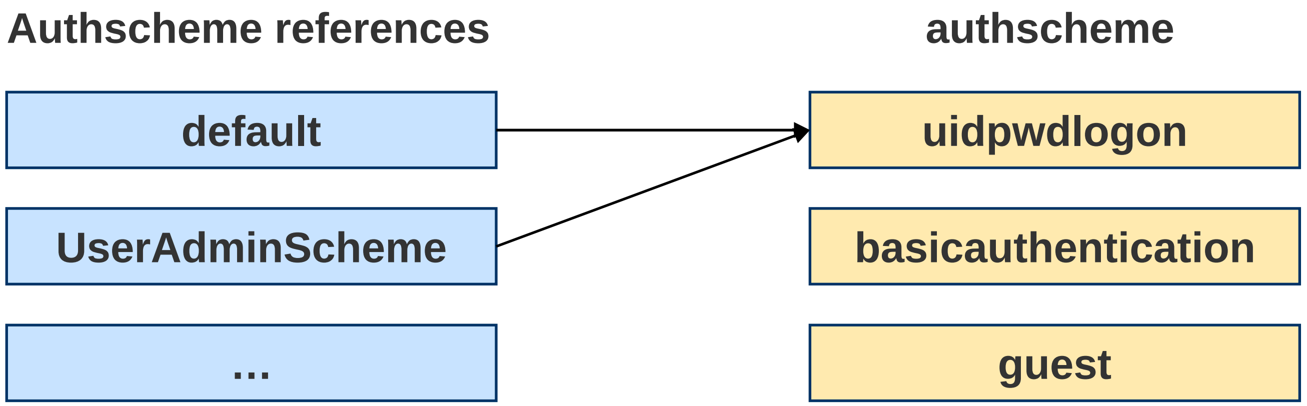
Definition of one authentication scheme reference

Authentication Scheme References (UME+EP)

Mapping of logical authentication schemes to authentication schemes defined in section <authschemes>

Reference names can be used in iView definitions (property "Authentication Scheme")

Authentication requirements can be changed without changing the iView property by only pointing the reference to a different authentication scheme



Logoff (EP)

It is possible to configure a URL that will be called during the log off process:

- "ume.logoff.redirect.url" specifies a URL to which users will be redirected after log off
- "ume.logoff.redirect.silent" specifies if the URL is called silently in a hidden iFrame (true) or not (false)

It is possible to send the parameter "logout_submit" containing any value to the EP to log a user off. This will NOT work if the request includes information which is used for logging users on (e.g. client certificate, basic authentication, NTLM, …)

JAAS Login Modules for WebAS Java 6.40

Extend the abstract class
`com.sap.engine.interfaces.security.auth.AbstractLoginModule`
(-> …\j2ee\cluster\serverX\bin\interfaces\security\security_api.jar)

Implement the following methods:

- initialize()
- login()
- commit()
- abort()
- logout()

```
import com.sap.engine.interfaces.security.auth.AbstractLoginModule

public class MyCustomLoginModule extends AbstractLoginModule {
    public void initialize(…){…}
    public boolean login() throws LoginException {…}
    public boolean commit() throws LoginException {…}
    public boolean abort() throws LoginException {…}
    public boolean logout() throws LoginException {…}
}
```

Supported Callbacks

Standard Callbacks

- javax.security.auth.callback.NameCallback()
  - Reads parameter j_user from HTTP servlet request
- javax.security.auth.callback.PasswordCallback()
  - Reads parameter j_password from HTTP servlet request

SAP Proprietary Callbacks (see NW ’04 documentation)

- com.sap.engine.lib.security.http.HttpGetterCallback()
  - Makes HTTP servlet request available to the JAAS login module
  - …\j2ee\cluster\serverX\bin\system\util.jar
- com.sap.engine.lib.security.http.HttpSetterCallback()
  - Makes HTTP servlet response available to the JAAS login module
  - …\j2ee\cluster\serverX\bin\system\util.jar

Necessary Actions in the login() Method

After the user name is known refresh user information before authentication is performed

- `refreshUserInfo(<user name>);`
- **Method throws a java.lang.SecurityException if user does not exist**
- **Necessary in order to always get the latest user information since user information in the cache might be out of date. One reason for an outdated cache could be that user information has been changed directly in the user repository and not via UME.**

Only one login module must put the user name (representing the authenticated user) into the shared state:

```
if (sharedState.get(AbstractLoginModule.NAME) == null) {
    sharedState.put(AbstractLoginModule.NAME, _userId);
    _nameSet = true;
}
```

If the shared state already holds a user name, you can check if the user in the shared state corresponds to the user you authenticated in your login module

Method login() – NameCallback/PasswordCallback

```
public boolean login() throws LoginException {
    NameCallback nameCallback = new NameCallback("user name: ");
    PasswordCallback pwdCallback = new PasswordCallback("password: ", false);
    try {
         callbackHandler.handle(new Callback[] {nameCallback, pwdCallback});
    } catch (java.io.IOException ioe) {
         throwUserLoginException(ioe, LoginExceptionDetails.IO_EXCEPTION);
    } catch (UnsupportedCallbackException uce) {
         _shoudBeIgnored=true;
         return false;
    }
    String _userId = nameCallback.getName();
    char[] password = pwdCallback.getPassword();
    pwdCallback.clearPassword();
    try  {
         refreshUserInfo(_userId);
    } catch (SecurityException e) {
         throwUserLoginException(e)
    }
    //check authentication …
    if succeeded return true
    else throwNewLoginException(“Wrong UserId/Password",
         LoginExceptionDetails.WRONG_USERNAME_PASSWORD_COMBINATION);
}
```

Method login() – httpGetterCallback()

```
public boolean login() throws LoginException {
    if (callbackHandler == null)
          throw new LoginException("Error: no CallbackHandler available " +
                    "to garner authentication information from the user");
    HttpGetterCallback httpGetterCallback = new HttpGetterCallback();
    httpGetterCallback.setType(HttpGetterCallback.HEADER);
    httpGetterCallback.setName(HEADER_NAME);
    succeeded = false;
    try {
          _callbackHandler.handle(new Callback[] {httpGetterCallback});
    } catch (UnsupportedCallbackException e) {
          _shouldBeIgnored = true;
          return false;
    } catch (IOException e) {
          throwUserLoginException(e, LoginExceptionDetails.IO_EXCEPTION);
    }
    _userId = (String) httpGetterCallback.getValue();
    // Refresh user information using refreshUserInfo(_userId) …
    //check authentication
    …
    if (succeeded) return true;
    else throwNewLoginException(“Wrong UserId/Password",
          LoginExceptionDetails.WRONG_USERNAME_PASSWORD_COMBINATION)
}
```

Method commit()

Add a Principal (and potentially credentials) to the subject

Method Principal.getName() must return the user's user ID

```
public boolean commit() throws LoginException {
    if (!_shouldBeIgnored) {
         if (_succeeded) {
             com.sap.engine.lib.security.Principal principal =
                    new com.sap.engine.lib.security.PrincipalPrincipal(_userId);
             _subject.getPrincipals().add(principal);
             // add credentials private/public to subject
             if (_userIdSet) {
                    _sharedState.put(AbstractLoginModule.PRINCIPAL, principal);
             }
         } else {
             _userId = null;
         }
         return true;
    } else {
         _shouldBeIgnored = false;
         return false;
    }
}
```

Method abort()

Clean out authentication and state information if phase 1 of authentication fails

```
public boolean abort() throws LoginException {
    if (!_shouldBeIgnored) {
         if (_succeeded) {
                   // clean out state
                   _user = null;
                   _succeeded = false;
         }
         return true;
    } else {
         _shouldBeIgnored = false;
         return false
    }
}
```

Method logout()

Log out a subject

Clean out authentication and state information

```
public boolean logout() throws LoginException {
    if (!_shouldBeIgnored) {
          if (_succeeded) {
                    // clean out state
                    subject.getPrincipals(Principal.class).clear();
                    _succeeded = false
          }
          return true;
    } else {
          shouldBeIgnored = false;
          return false;
    }
}
```

Logging/Tracing Inside the Login Module

For logging and tracing you use the standard logging/tracing functionality of the WebAS Java.

(-> …\j2ee\cluster\serverX\bin\system\logging.jar)

```
static Location LOCATION = Location.getLocation("com.demo.MyLoginModule");
…

LOCATION.debugT (…)
LOCATION.warningT (…)
```

References (you need an SDN account):

- https://www.sdn.sap.com/sdn/developerareas/ep.sdn?page=javadoc.htm
- https://www.sdn.sap.com/irj/servlet/prt/portal/prtroot/com.sapportals.km.docs/documents/a1-8-4/Tutorial%20-%20Logging%20and%20Tracing%20Mechanism%20in%20SAP.pdf

Custom Logon Frontend (for EP)

Modification of existing logon UI

- Modification of the logon UI is documented (-> Security Guide)

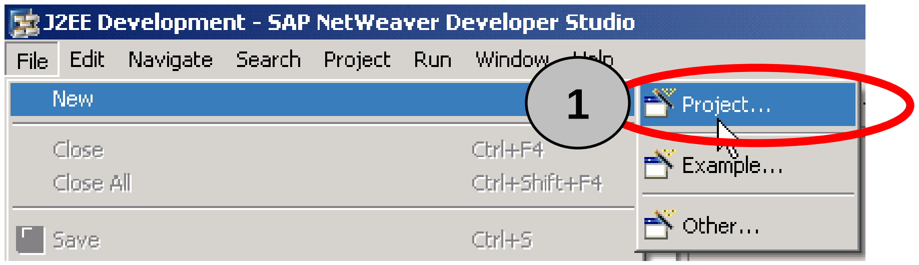
Own implementation

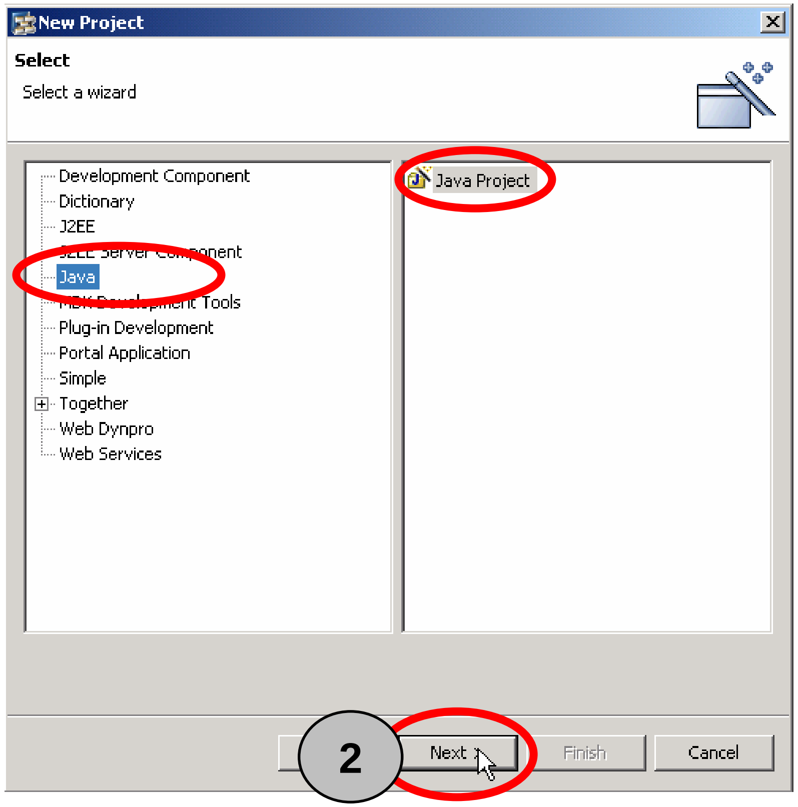
- Implement e.g. AbstractPortalComponent
- If using NameCallback and PasswordCallback, the following parameters are necessary in the logon form
  - j_user
  - j_password
- In order to specify a specific authentication scheme you need following parameter:
  - j_authscheme=<name of your authentication scheme>

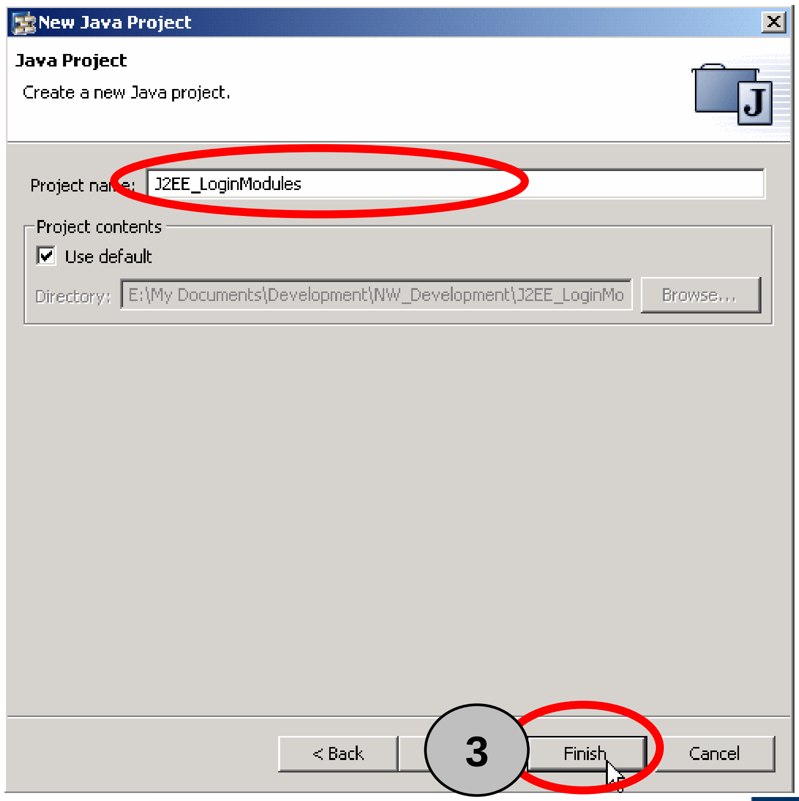
Steps to Create a New J2EE Library

1. Create a new Java project
2. Implement your login module
3. Create a jar file that includes your login module(s)
4. Create a new J2EE Engine library project
5. Configure provider.xml (versions, references)
6. Add logging information to your library
7. Build SDA File
8. Deploy SDA File to your server

Create New Java Project


J2EE Development - SAP NetWeaver Developer Studio
File Edit Navigate Search Project Run Window Help
New
Close Ctrl+F4
Close All Ctrl+Shift+F4
Save Ctrl+S
Project...
Example...
Other...
1





Project name: J2EE_LoginModules

☑ Use default

Directory: E:\My Documents\Development\NW_Development\J2EE_LoginMo

Implementation

Following libraries need to be included in your classpath settings:

- ...\j2ee\cluster\serverX\bin\system\util.jar
- …\j2ee\cluster\serverX\bin\interfaces\security\security_api.jar
- All other libraries you use

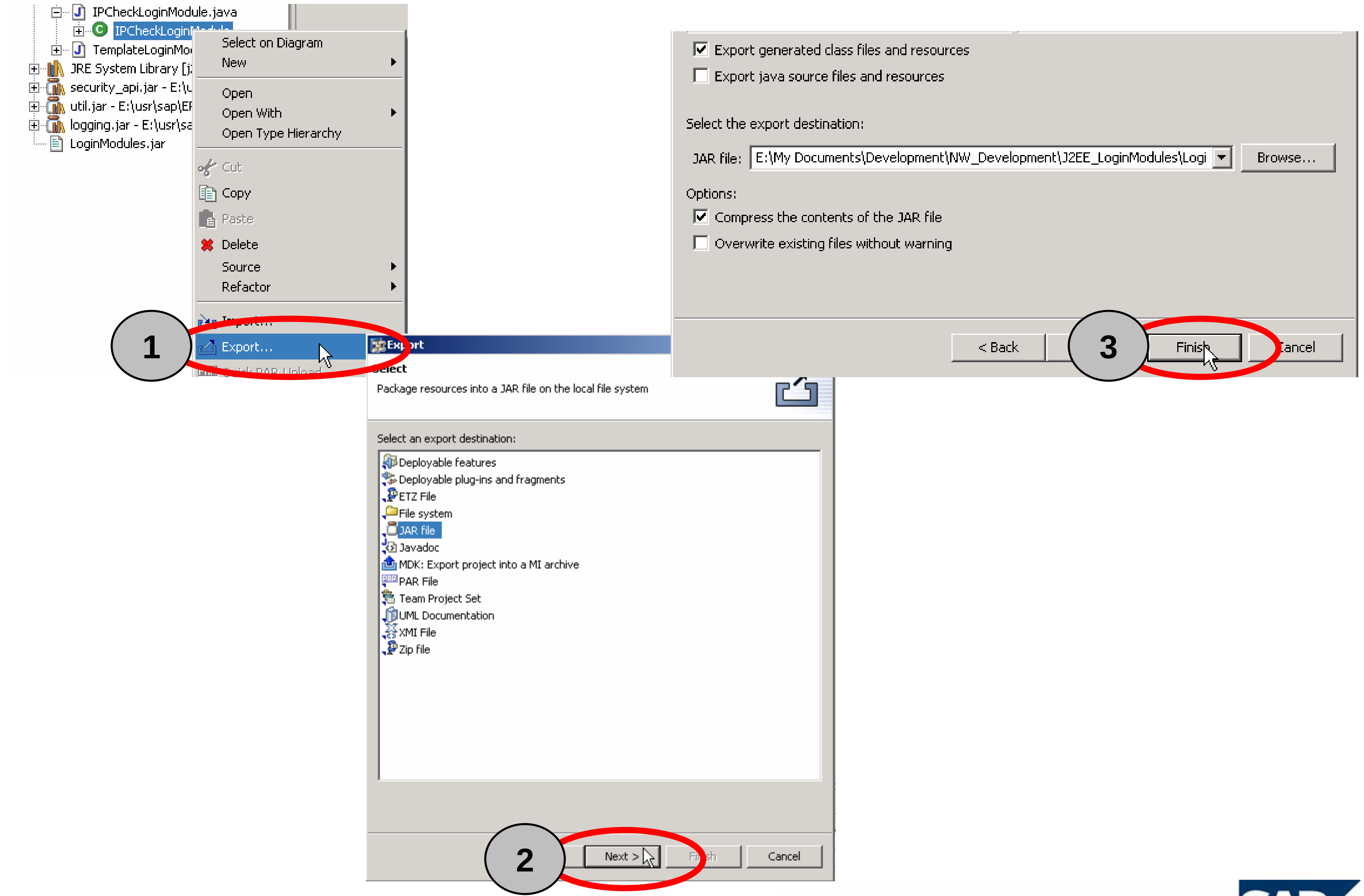
Classpath settings:

- Go to the properties of your Java project
- Go to “Java Build Path”
- Add the libraries as external jars

Implement your login module

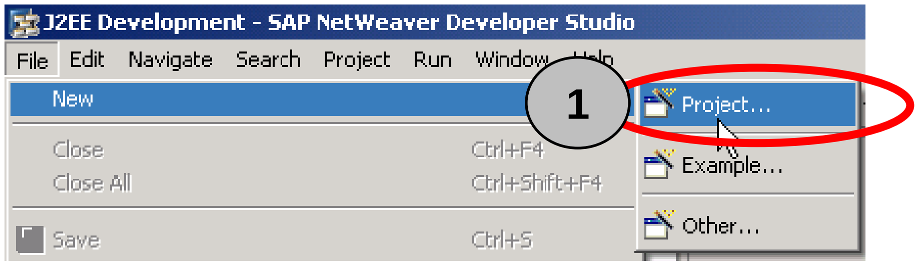
Compile the project

Create New Jar File

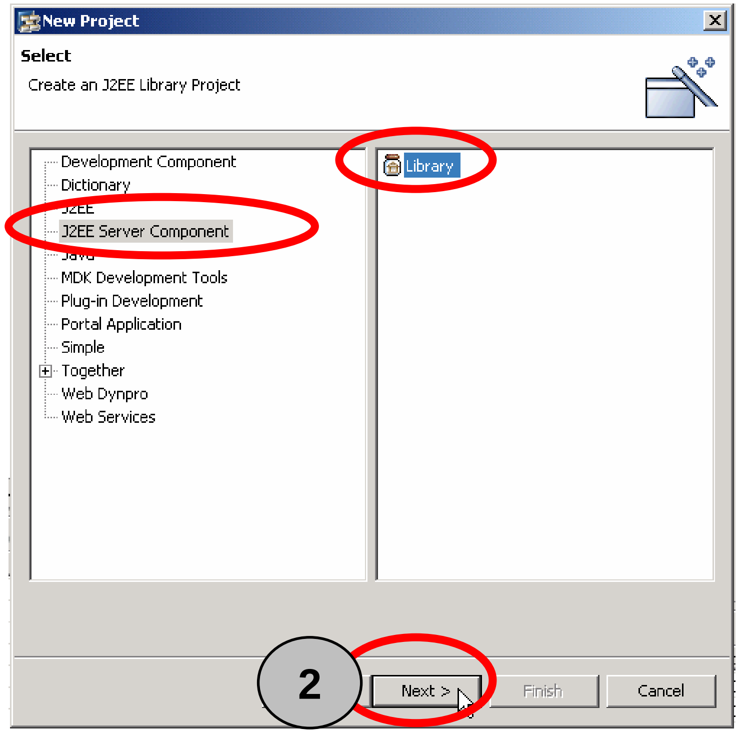


JAR file: E:\My Documents\Development\NW_Development\J2EE_LoginModules\Logi

Create a New J2EE Library Project



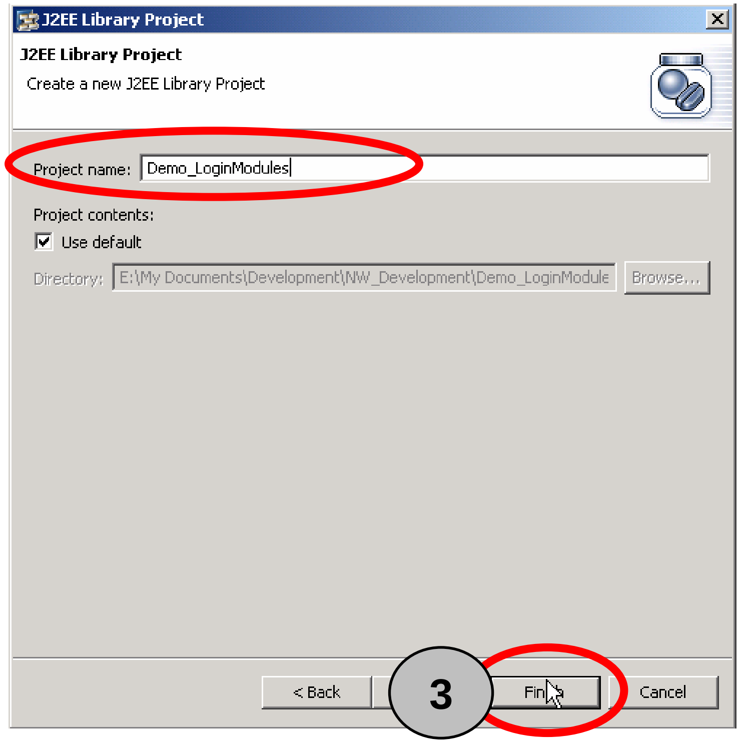
J2EE Development - SAP NetWeaver Developer Studio



New Project

Select

Create an J2EE Library Project



J2EE Library Project

J2EE Library Project

Create a new J2EE Library Project

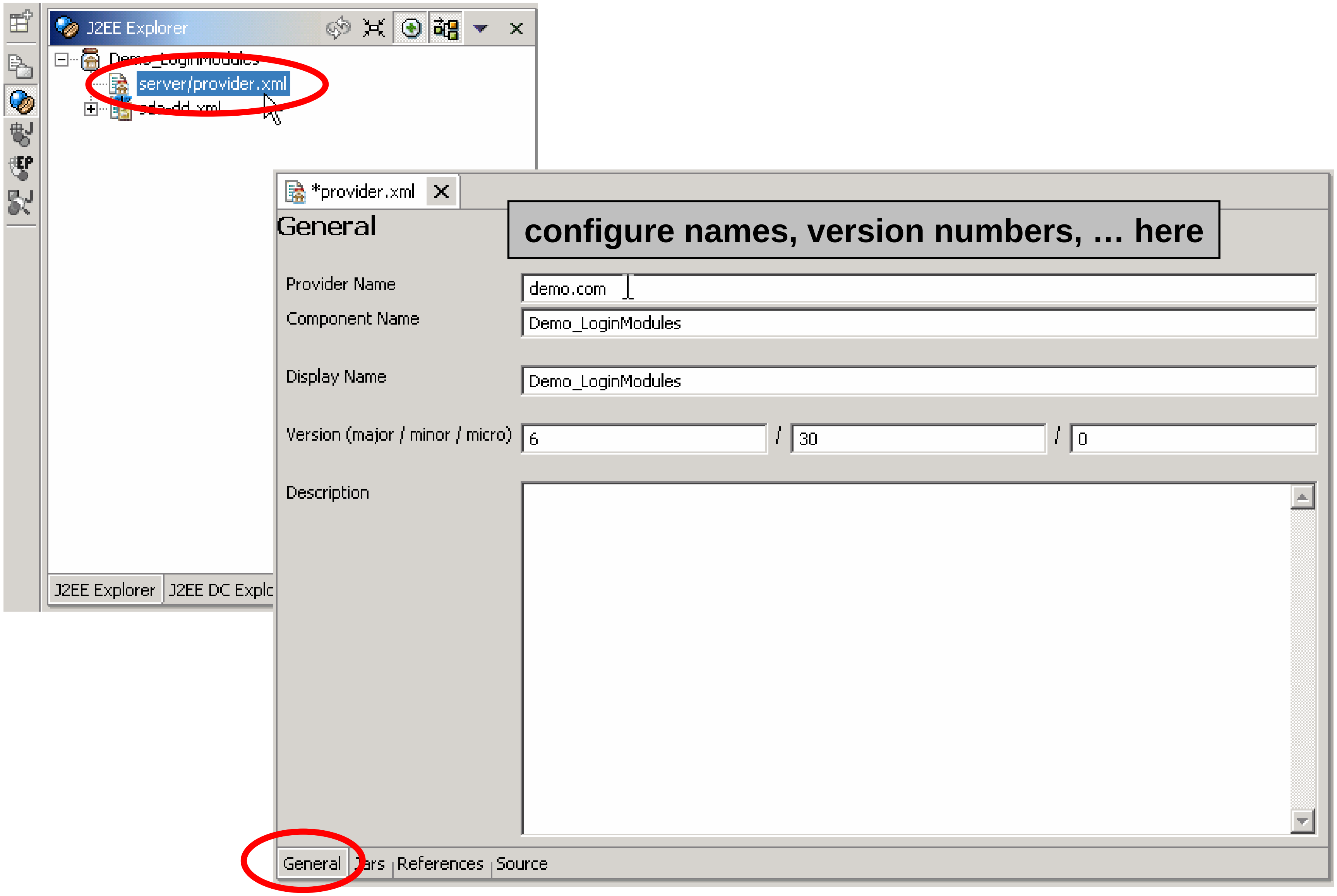
Project name: Demo_LoginModules

Project contents:

☑ Use default

Directory: E:\My Documents\Development\NW_Development\Demo_LoginModule

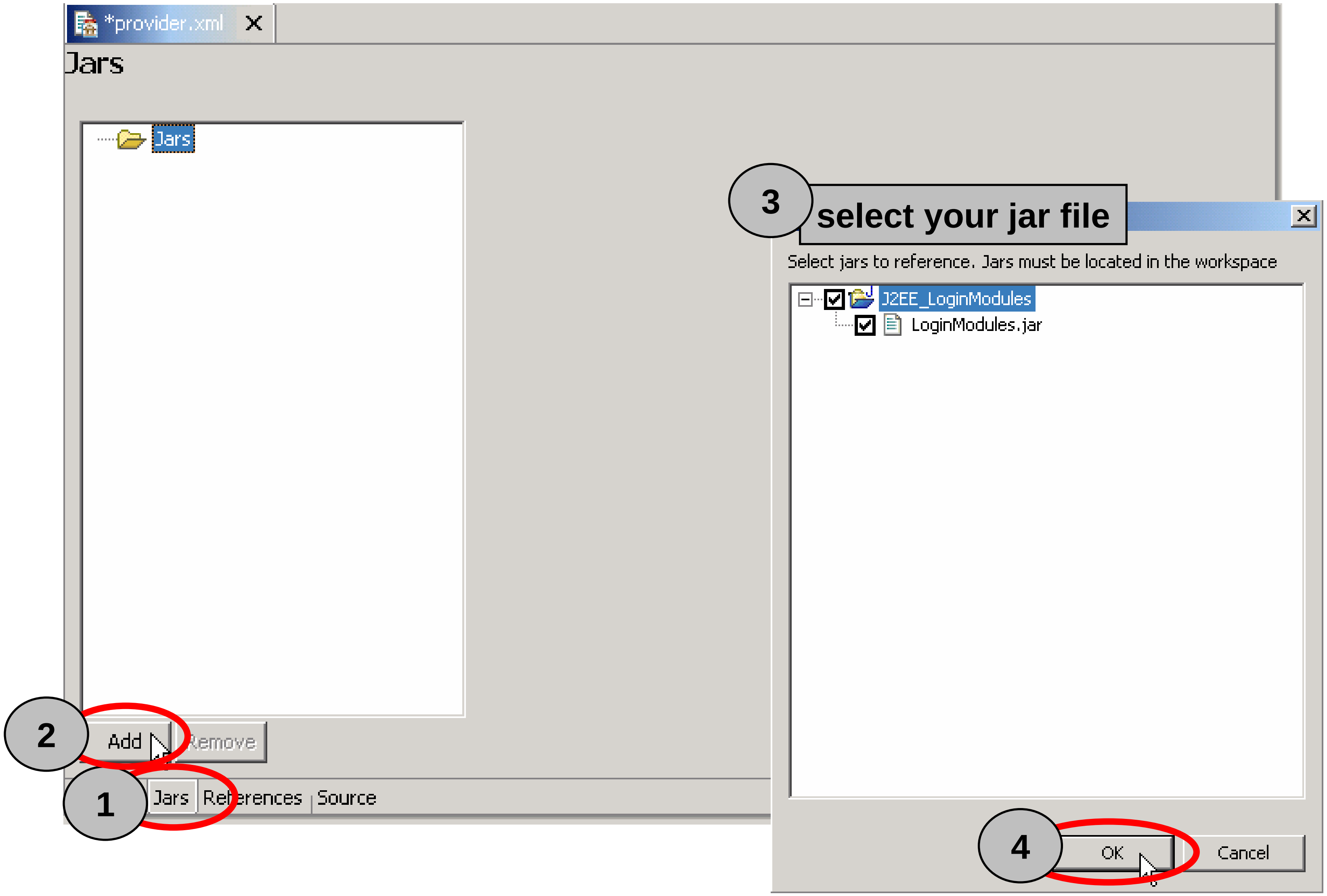
Configure Provider.xml



configure names, version numbers, … here

Provider Name	demo.com
Component Name	Demo_LoginModules
Display Name	Demo_LoginModules
Version (major / minor / micro)	6 / 30 / 0
Description	

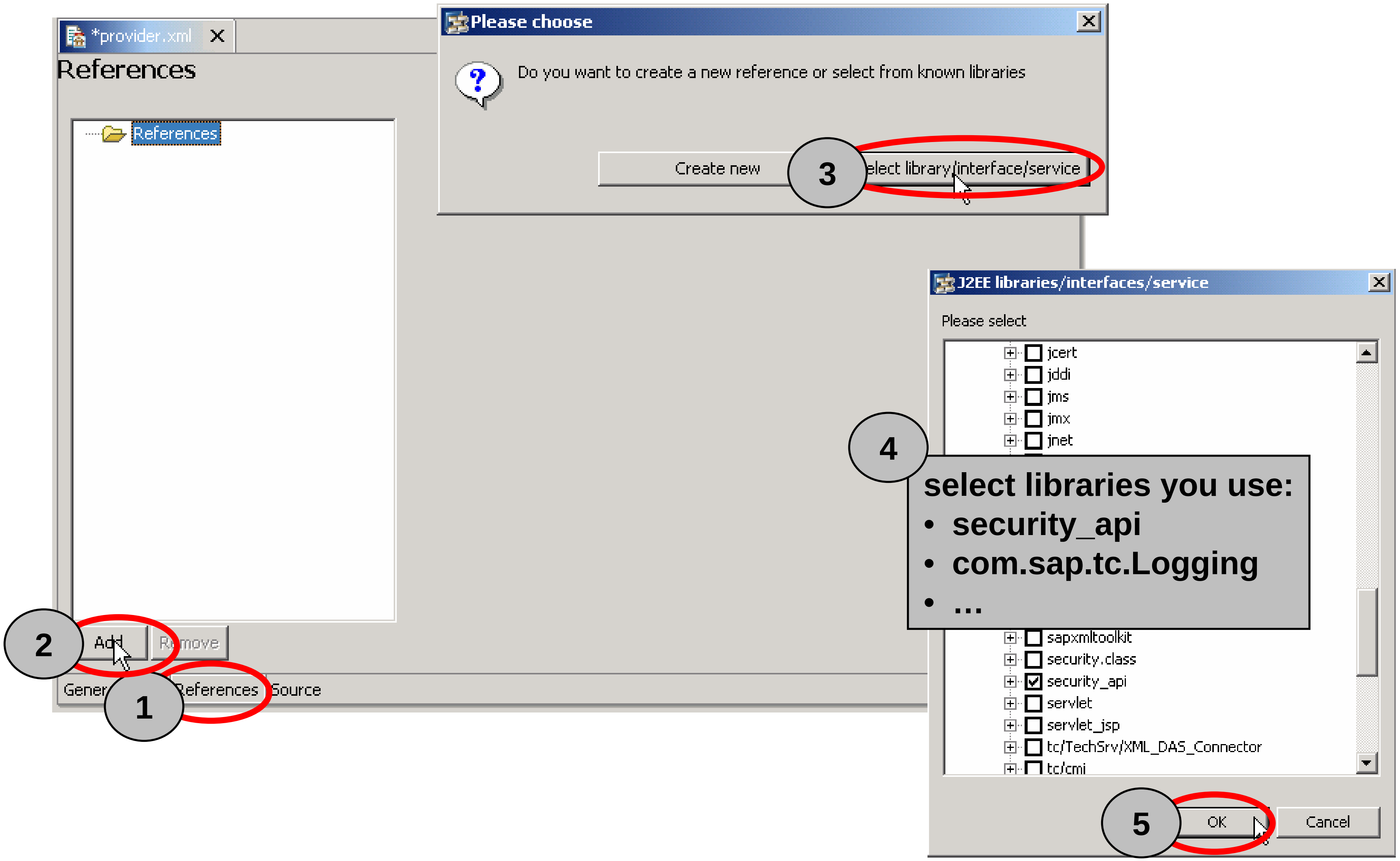
Add Jar File Containing Your Login Module(s)



Select jars to reference. Jars must be located in the workspace

- [✓] J2EE_LoginModules
  - [✓] LoginModules.jar

Add Library References



Do you want to create a new reference or select from known libraries

Create new | Select library/interface/service

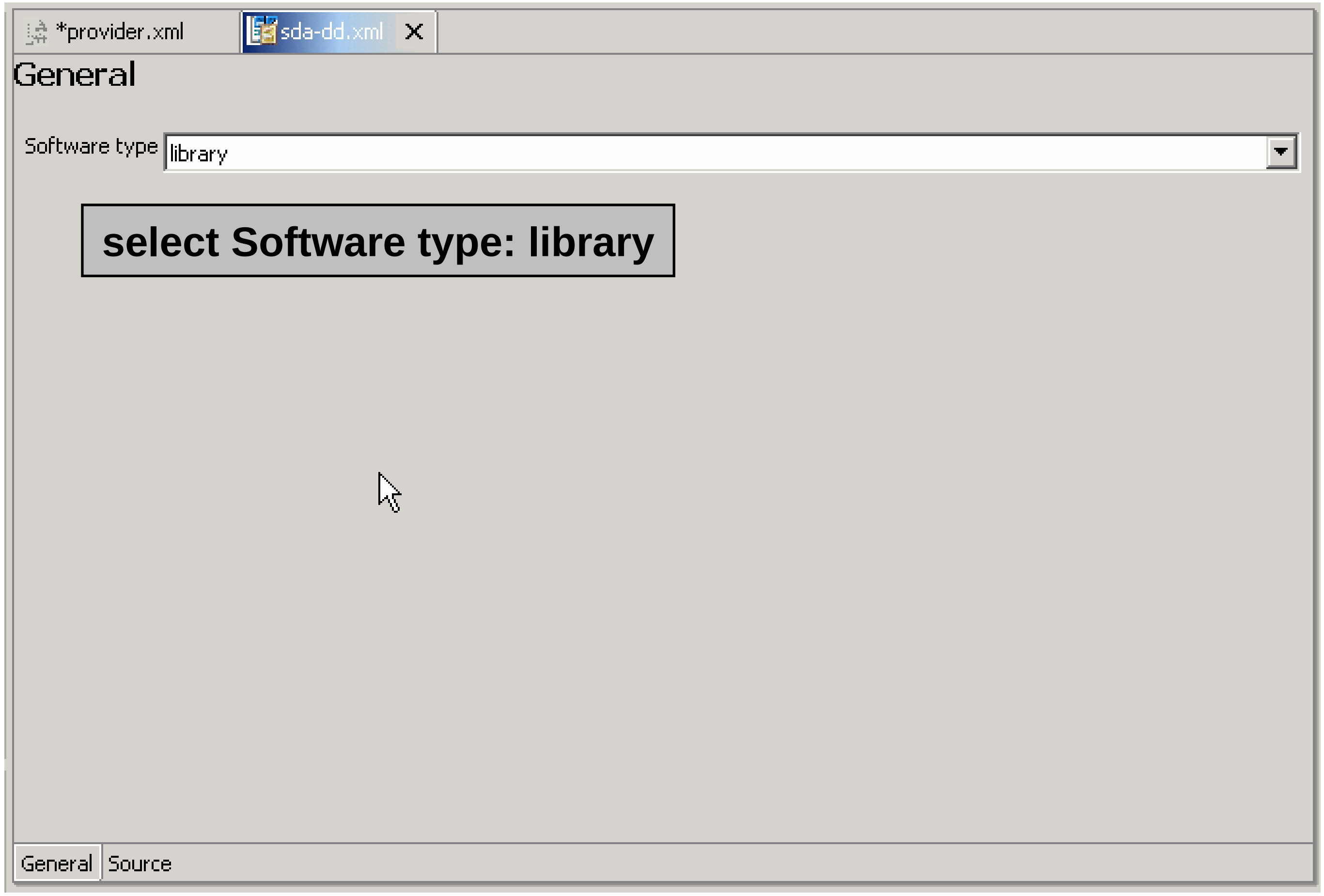
Please select

- jcert
- jddi
- jms
- jmx
- jnet
- sapxmltoolkit
- security.class
- [✓] security_api
- servlet
- servlet_jsp
- tc/TechSrv/XML_DAS_Connector
- tc/cmi

select libraries you use:

- **security_api**
- **com.sap.tc.Logging**
- **…**

Configure SDA Definition

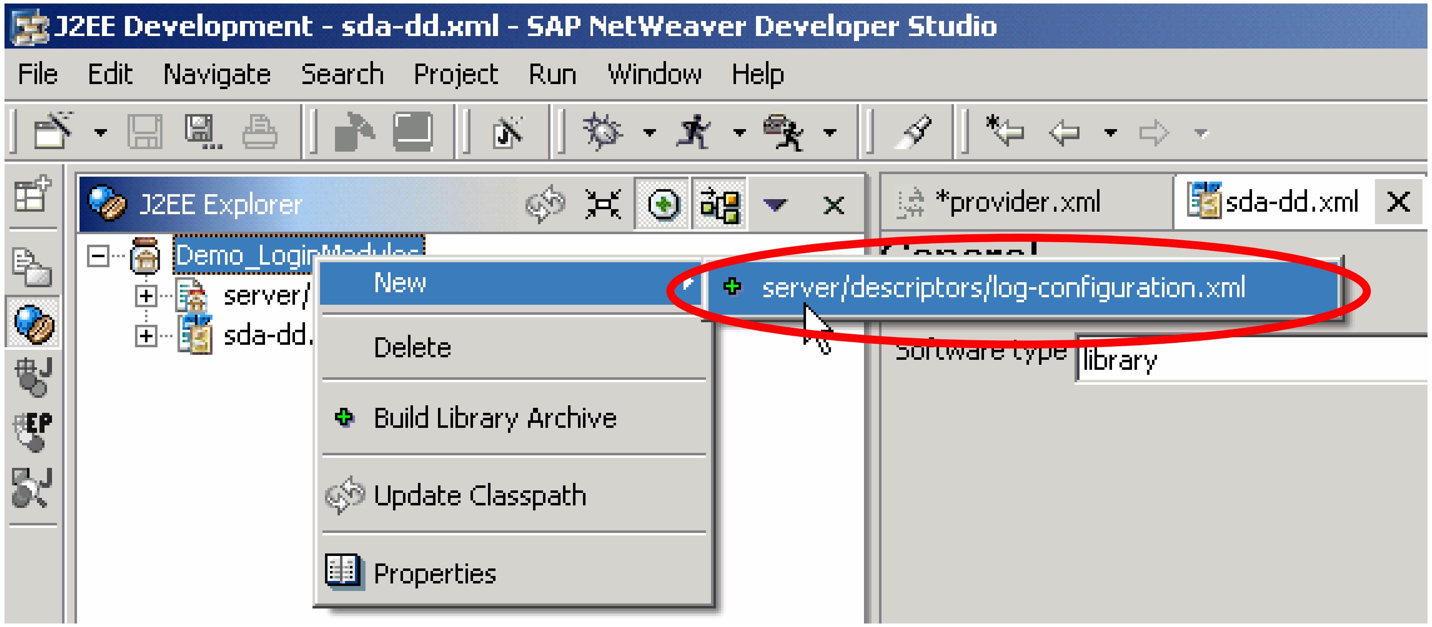


General

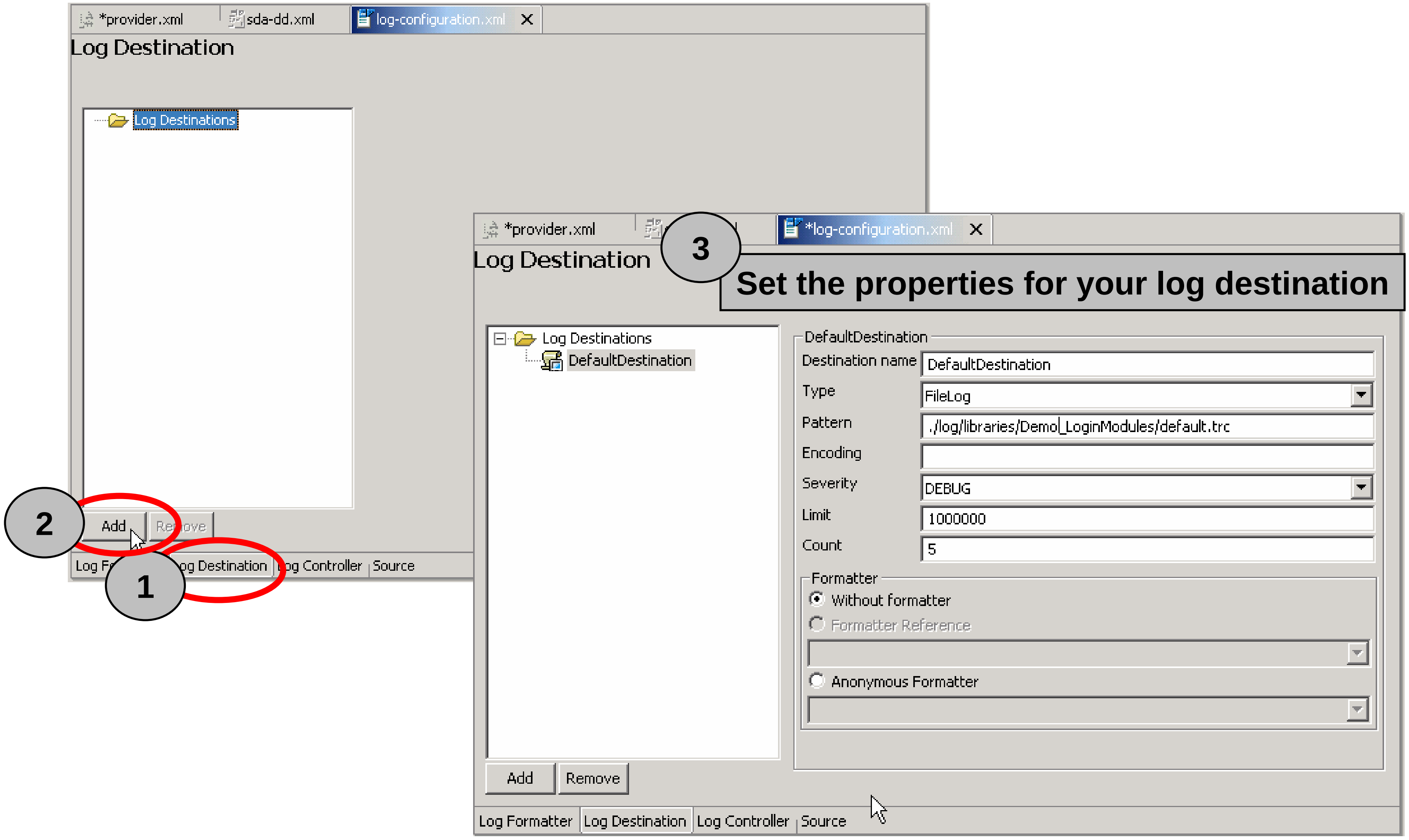
Software type: library

select Software type: library

Create a Log Configuration for Your Project


J2EE Development - sda-dd.xml - SAP NetWeaver Developer Studio
File Edit Navigate Search Project Run Window Help
J2EE Explorer
*provider.xml
sda-dd.xml
Demo_LoginModules
server/
sda-dd.
New
server/descriptors/log-configuration.xml
Delete
Build Library Archive
Update Classpath
Properties
Software type
library

Add and Configure a Log Destination



Set the properties for your log destination

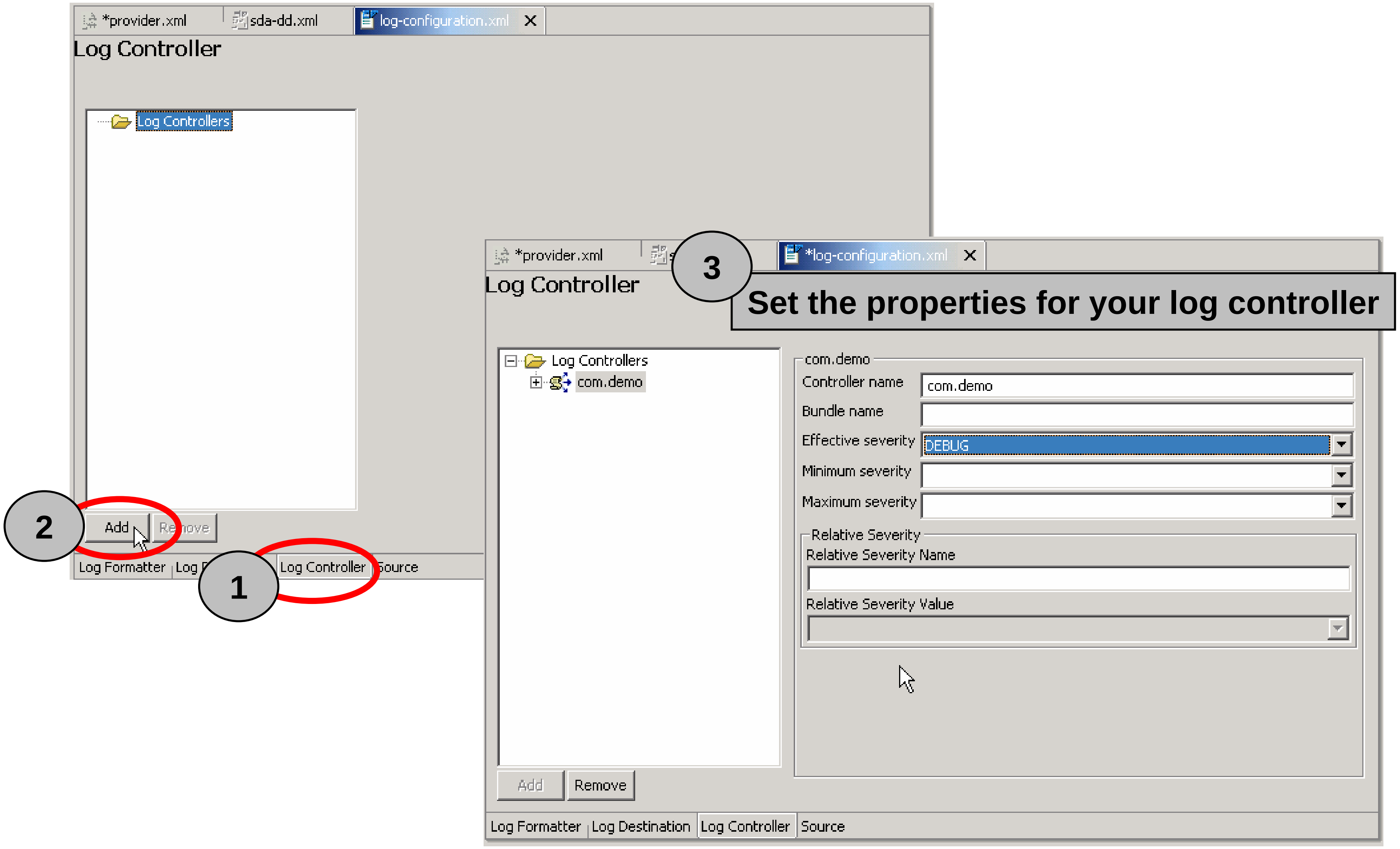
DefaultDestination

Destination name	DefaultDestination
Type	FileLog
Pattern	./log/libraries/Demo_LoginModules/default.trc
Encoding	
Severity	DEBUG
Limit	1000000
Count	5

Formatter

[✓ Without formatter]
Formatter Reference
Anonymous Formatter

Add and Configure Log Controller



Set the properties for your log controller

com.demo

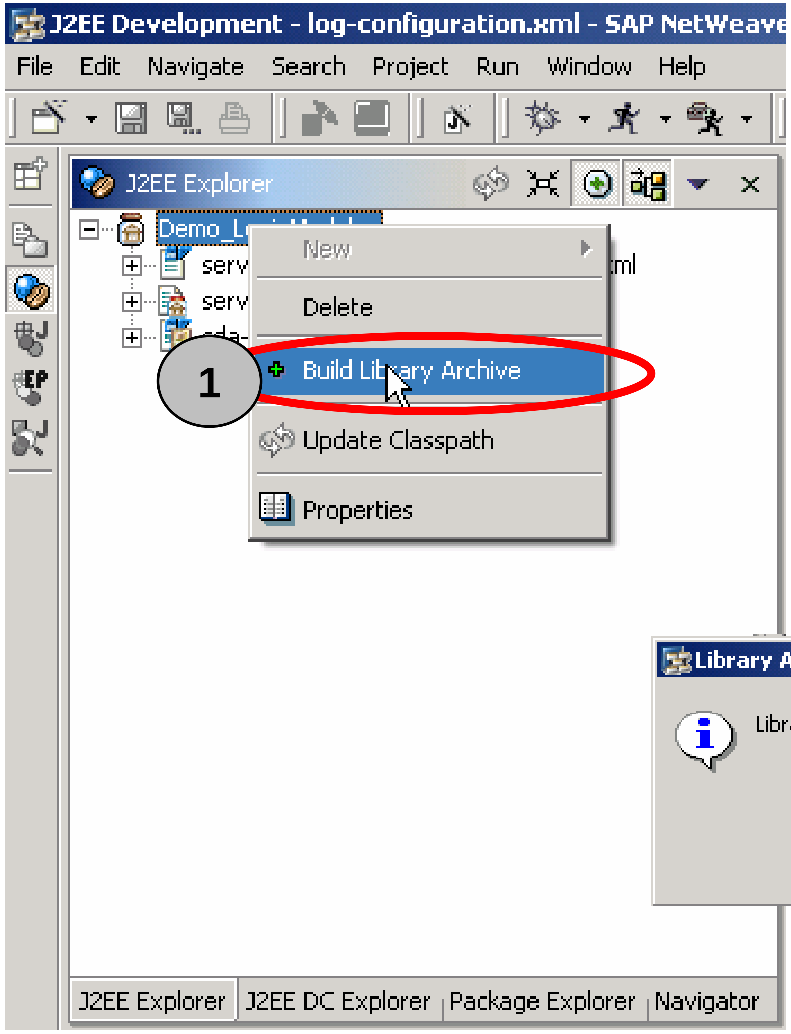
Controller name	com.demo
Bundle name	
Effective severity	DEBUG
Minimum severity	
Maximum severity	

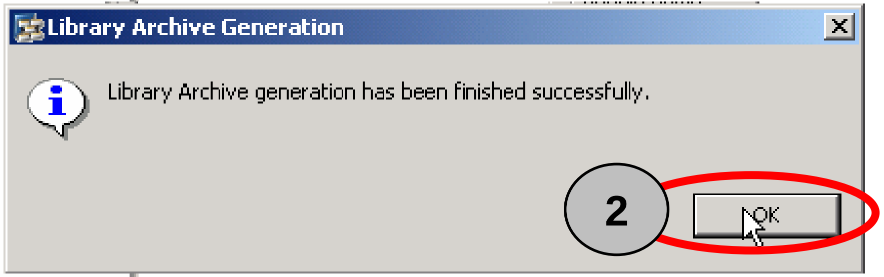
Relative Severity

Relative Severity Name

Relative Severity Value

Build SDA File





Library Archive generation has been finished successfully.

JAAS Overview

Authentication on the SAP J2EE engine

Developing an authentication module

Deployment and Configuration

Demo

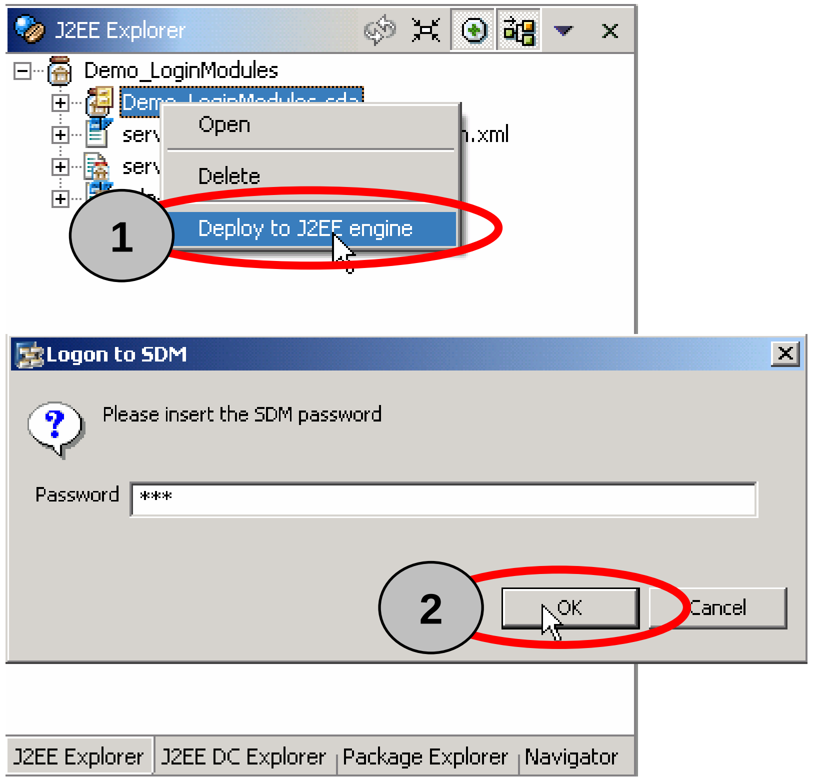
Demo

https://media.sdn.sap.com/public/eclasses/teched04/SCUR352_files/Default.htm#nopreload=1

Deployment from SAP NetWeaver Developer Studio

If you maintained the settings for your SAP WebAS Java correctly you will be able to deploy the SDA directly to your development system

- **Right-click on your SDA**
- **Select “Deploy to J2EE engine”**
- **You might need to enter your SDM password**



THE BEST-RUN BUSINESSES RUN SAP

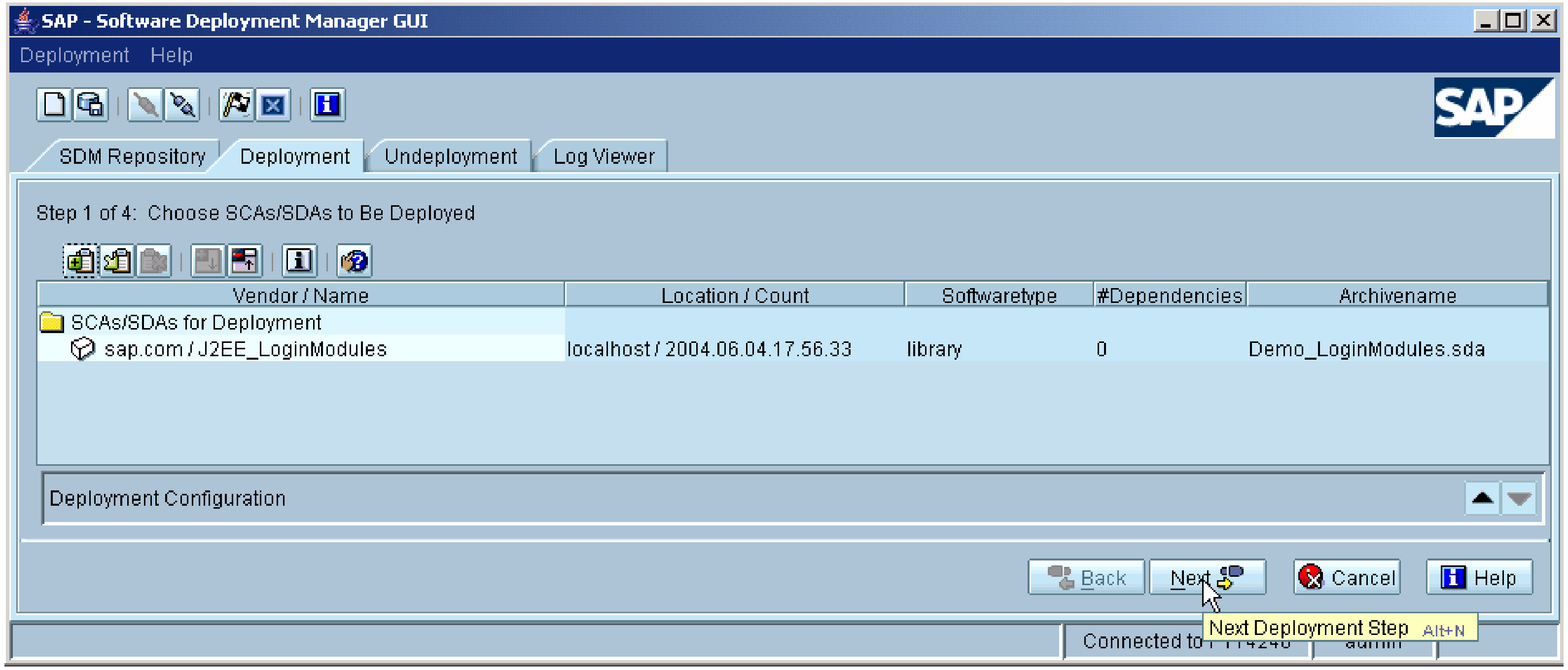
Deploy SDA File to Your Server With SDM

Start SDM Remote GUI

- UNIX: usr/sap/<SID>/<instance>/SDM/program/RemoteGui.sh
- Windows: usr\sap\<SID>\<instance>\SDM\program\RemoteGui.bat

Enter your SDM password (default: sdm)

Select your SDA file for deployment and deploy it



Step 1 of 4: Choose SCAs/SDAs to Be Deployed

Vendor / Name	Location / Count	Softwaretype	#Dependencies	Archivename
SCAs/SDAs for Deployment				
sap.com / J2EE_LoginModules	localhost / 2004.06.04.17.56.33	library	0	Demo_LoginModules.sda

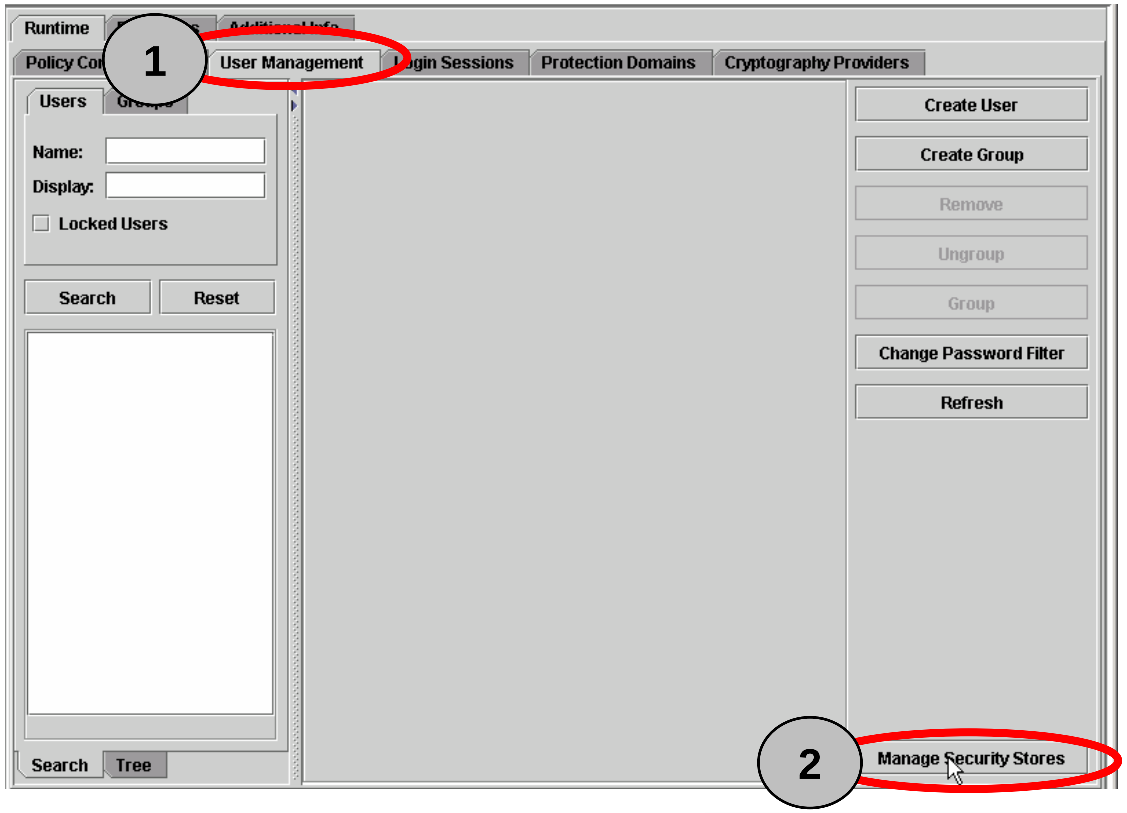
THE BEST-RUN BUSINESSES RUN SAP

Add New Login Modules to J2EE Engine (1)

Start Visual Administrator and connect to your server

Open configuration of the “Security Provider” service

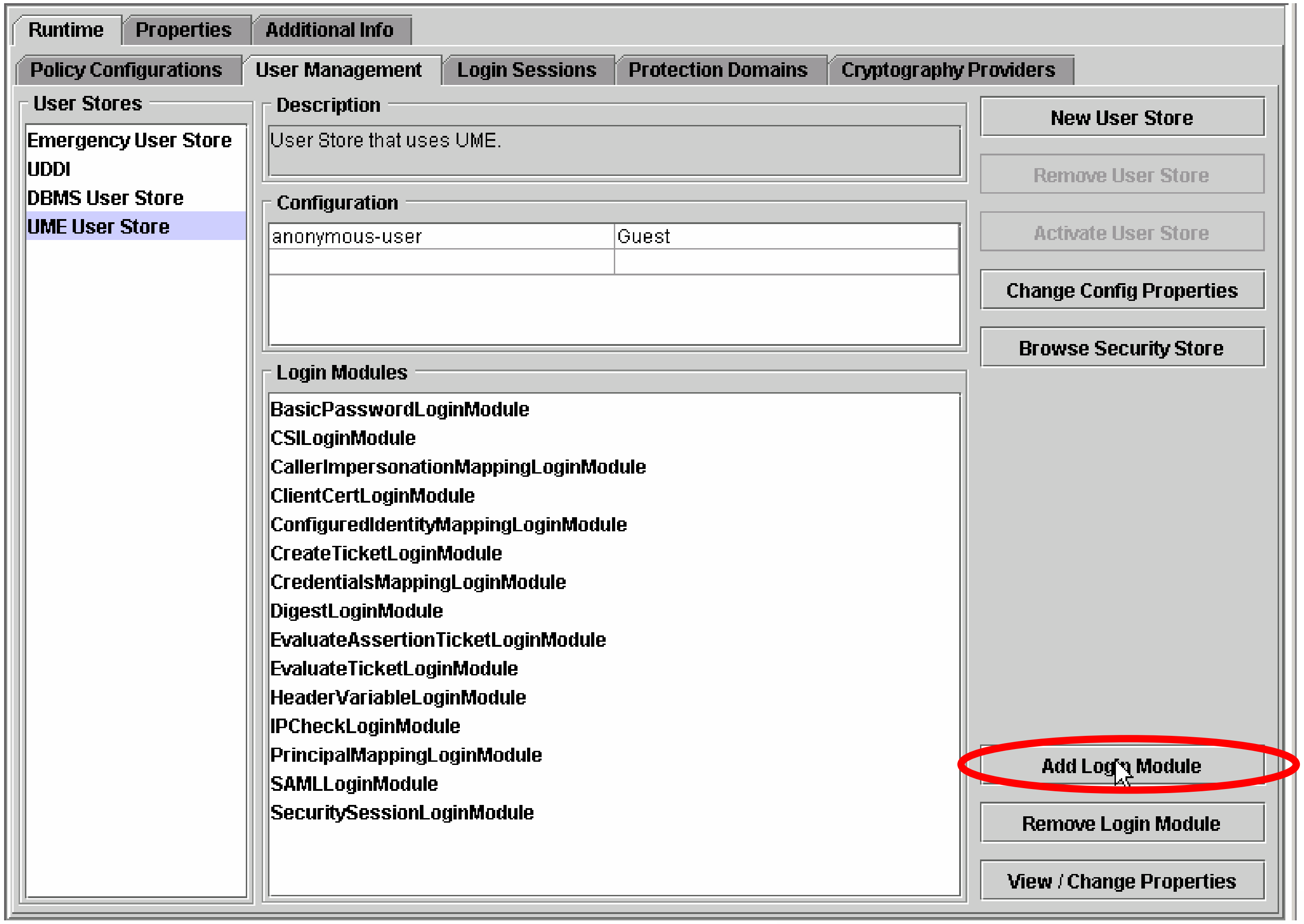
Switch to tab “User Management” and click on “Manage Security Stores”



Add New Login Modules to J2EE Engine (2)

Add Login Module to a specific user store

- UME User Store is sufficient in most cases



User Stores

- Emergency User Store
- UDDI
- DBMS User Store
- UME User Store

Description

User Store that uses UME.

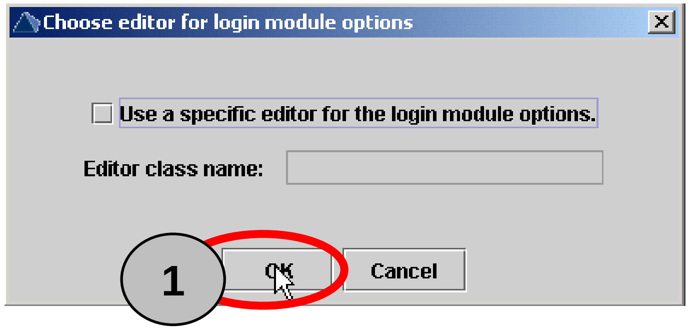
Configuration

anonymous-user	Guest

Login Modules

- BasicPasswordLoginModule
- CSILoginModule
- CallerImpersonationMappingLoginModule
- ClientCertLoginModule
- ConfiguredIdentityMappingLoginModule
- CreateTicketLoginModule
- CredentialsMappingLoginModule
- DigestLoginModule
- EvaluateAssertionTicketLoginModule
- EvaluateTicketLoginModule
- HeaderVariableLoginModule
- IPCheckLoginModule
- PrincipalMappingLoginModule
- SAMLLoginModule
- SecuritySessionLoginModule

Add New Login Modules to J2EE Engine (3)

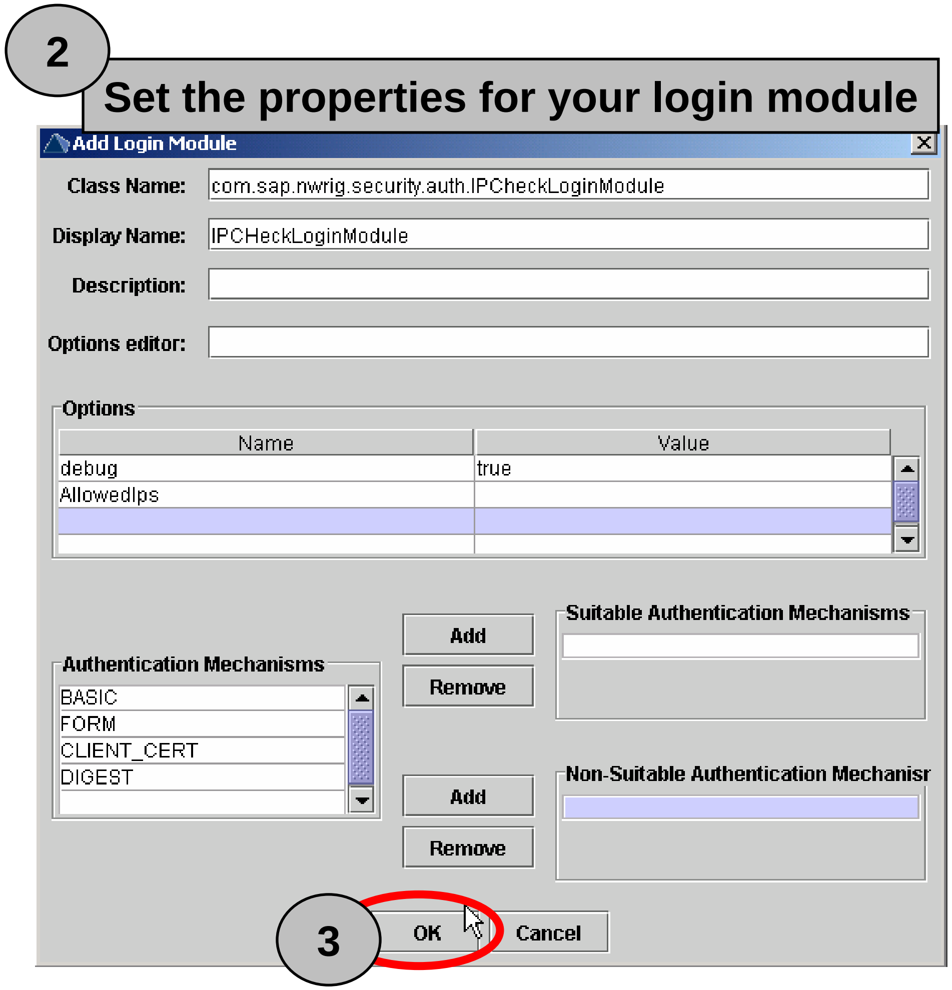


Choose editor for login module options

☐ Use a specific editor for the login module options.

Editor class name:

2 Set the properties for your login module



Add Login Module

Class Name: com.sap.nwrig.security.auth.IPCheckLoginModule

Display Name: IPCHeckLoginModule

Description:

Options editor:

Options

Name	Value
debug	true
AllowedIps	

Authentication Mechanisms

BASIC
FORM
CLIENT_CERT
DIGEST

Suitable Authentication Mechanisms

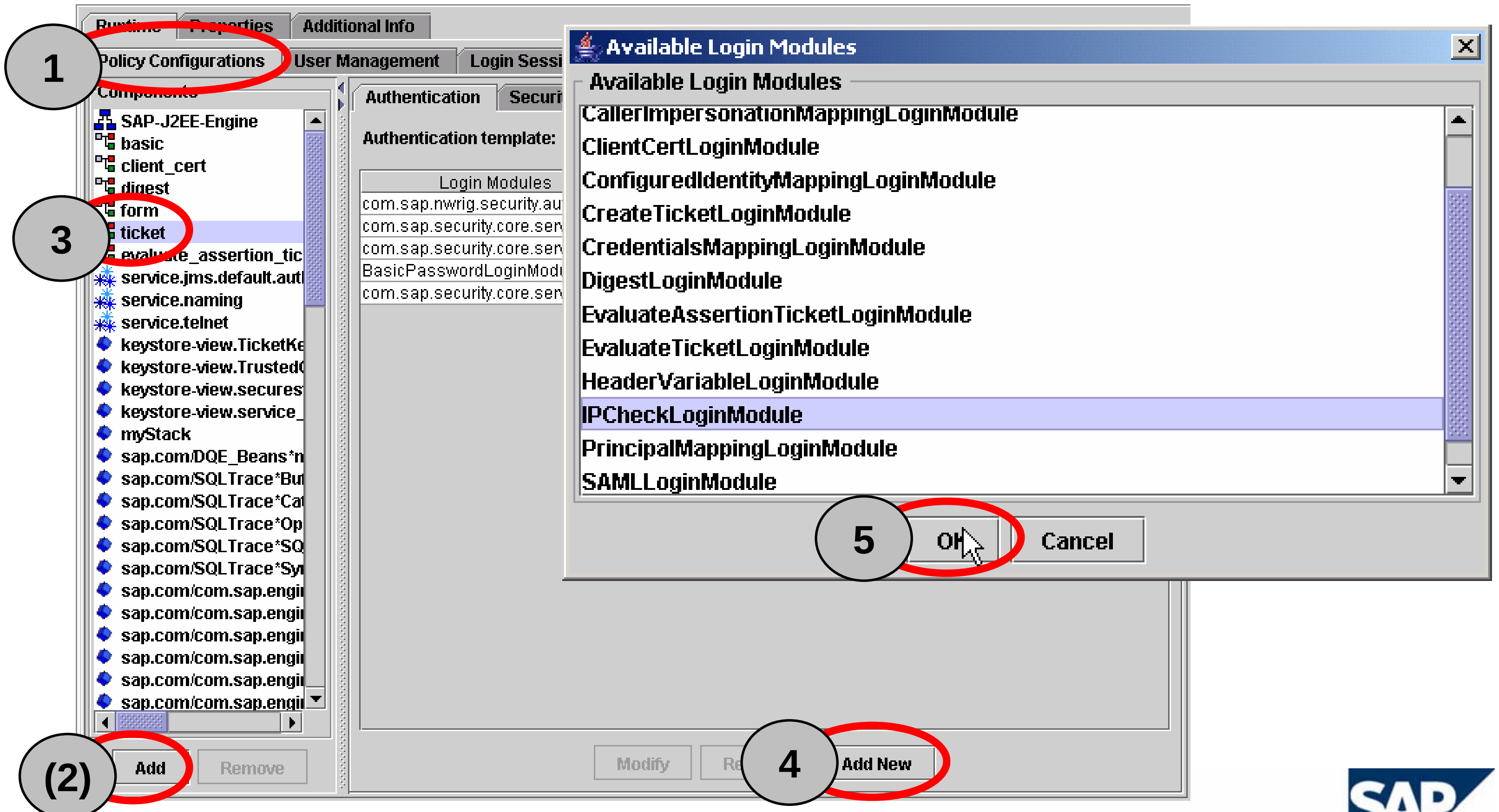
Non-Suitable Authentication Mechanisr

Add Login Module to a Login Module Stack

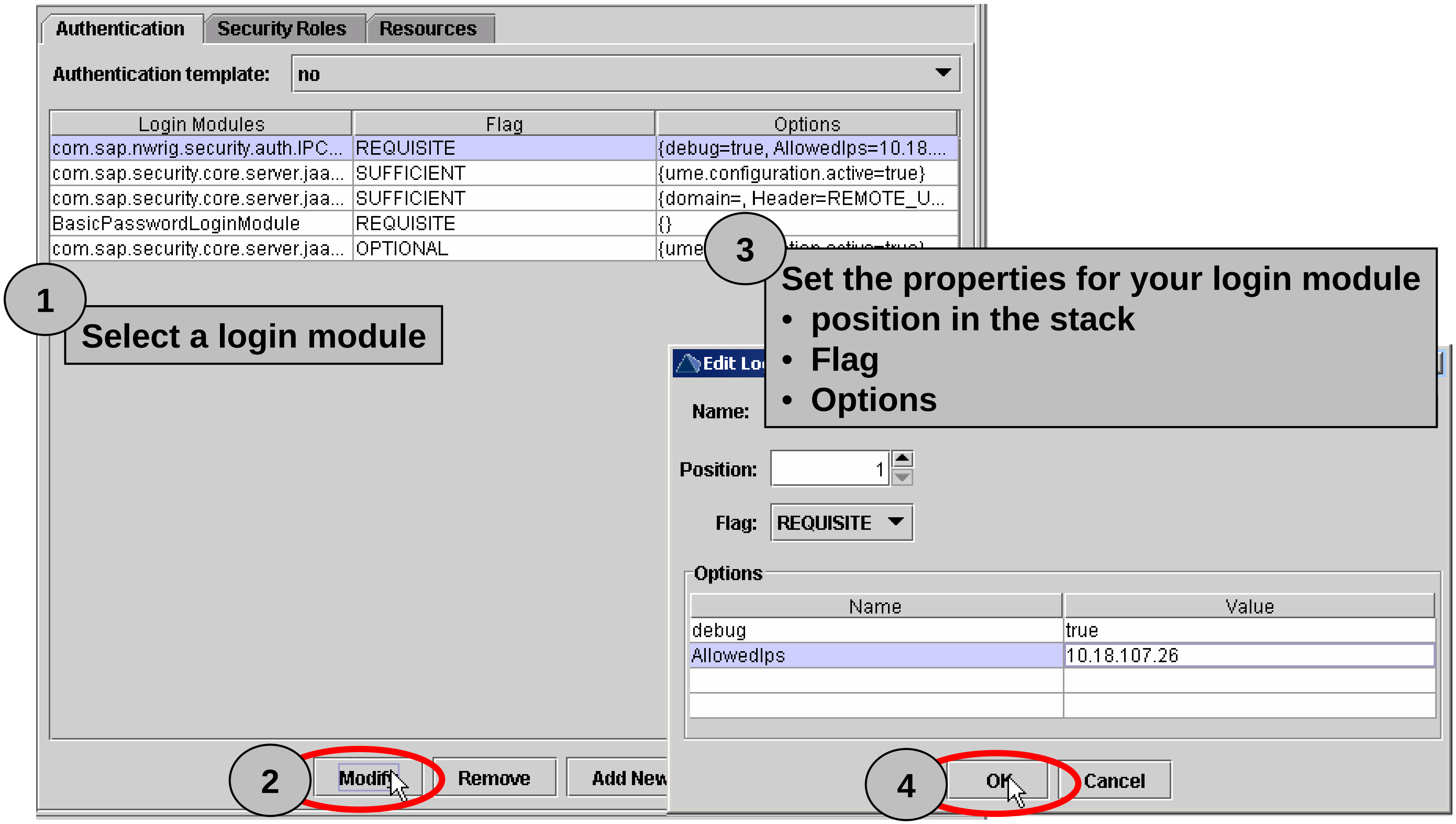
Go to tab "Policy Configurations"

Optionally add a new "Component"

Add new Login Module to a selected login module stack



New Login Module – Configuration



Authentication template: no

Login Modules	Flag	Options
com.sap.nwrig.security.auth.IPC...	REQUISITE	{debug=true, AllowedIps=10.18....
com.sap.security.core.server.jaa...	SUFFICIENT	{ume.configuration.active=true}
com.sap.security.core.server.jaa...	SUFFICIENT	{domain=, Header=REMOTE_U...
BasicPasswordLoginModule	REQUISITE	{}
com.sap.security.core.server.jaa...	OPTIONAL	{ume

1 Select a login module

2

3 Set the properties for your login module
- position in the stack
- Flag
- Options

Name:

Position: 1

Flag: REQUISITE

Options

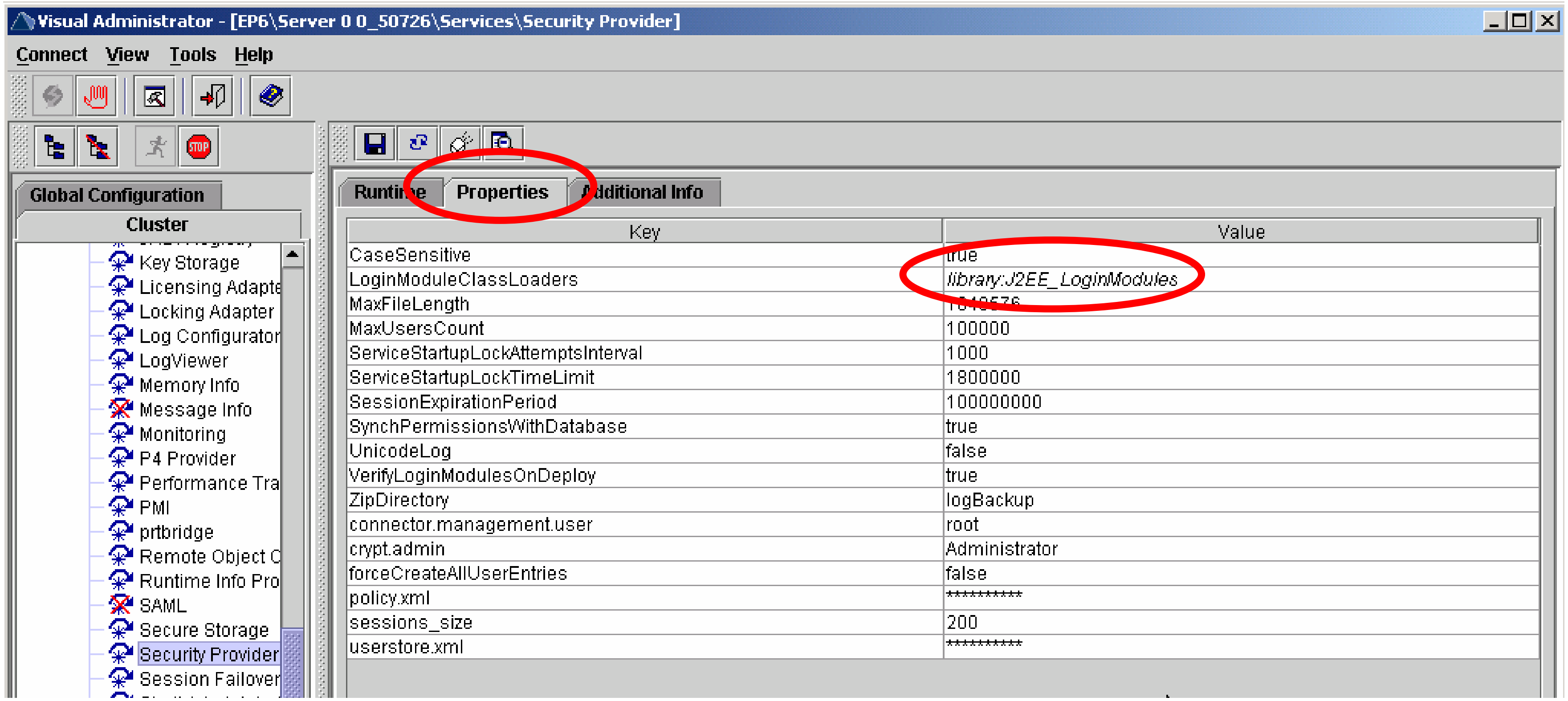
Name	Value
debug	true
AllowedIps	10.18.107.26

4

Classloader Reference to Login Module

Go to the properties View of the Security Provider service

Enter “library:<YourLibraryName>” as value for property “LoginModuleClassLoaders” (multiple entries: comma separated)



Key	Value
CaseSensitive	true
LoginModuleClassLoaders	*library:J2EE_LoginModules*
MaxFileLength	1048576
MaxUsersCount	100000
ServiceStartupLockAttemptsInterval	1000
ServiceStartupLockTimeLimit	1800000
SessionExpirationPeriod	100000000
SynchPermissionsWithDatabase	true
UnicodeLog	false
VerifyLoginModulesOnDeploy	true
ZipDirectory	logBackup
connector.management.user	root
crypt.admin	Administrator
forceCreateAllUserEntries	false
policy.xml	**********
sessions_size	200
userstore.xml	**********

THE BEST-RUN BUSINESSES RUN SAP

Summary

You are now able to

- Describe JAAS and the main ideas behind it
- Understand and configure the authentication mechanism of SAP WebAS 6.40 JAVA
- Develop a JAAS login module for SAP WebAS 6.40 Java
- Deploy and configure a JAAS login module on SAP WebAS 6.40 Java

Further Information

➔ Public Web:

www.sap.com

SAP Developer Network: www.sdn.sap.com ➔ SAP NetWeaver ➔ Security

➔ Related SAP Education Training Opportunities

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

➔ Related Workshops/Lectures at SAP TechEd 2004

SCUR251, Single Sign-On in Heterogeneous Landscapes

SCUR351, User Management and Authorizations: The Details

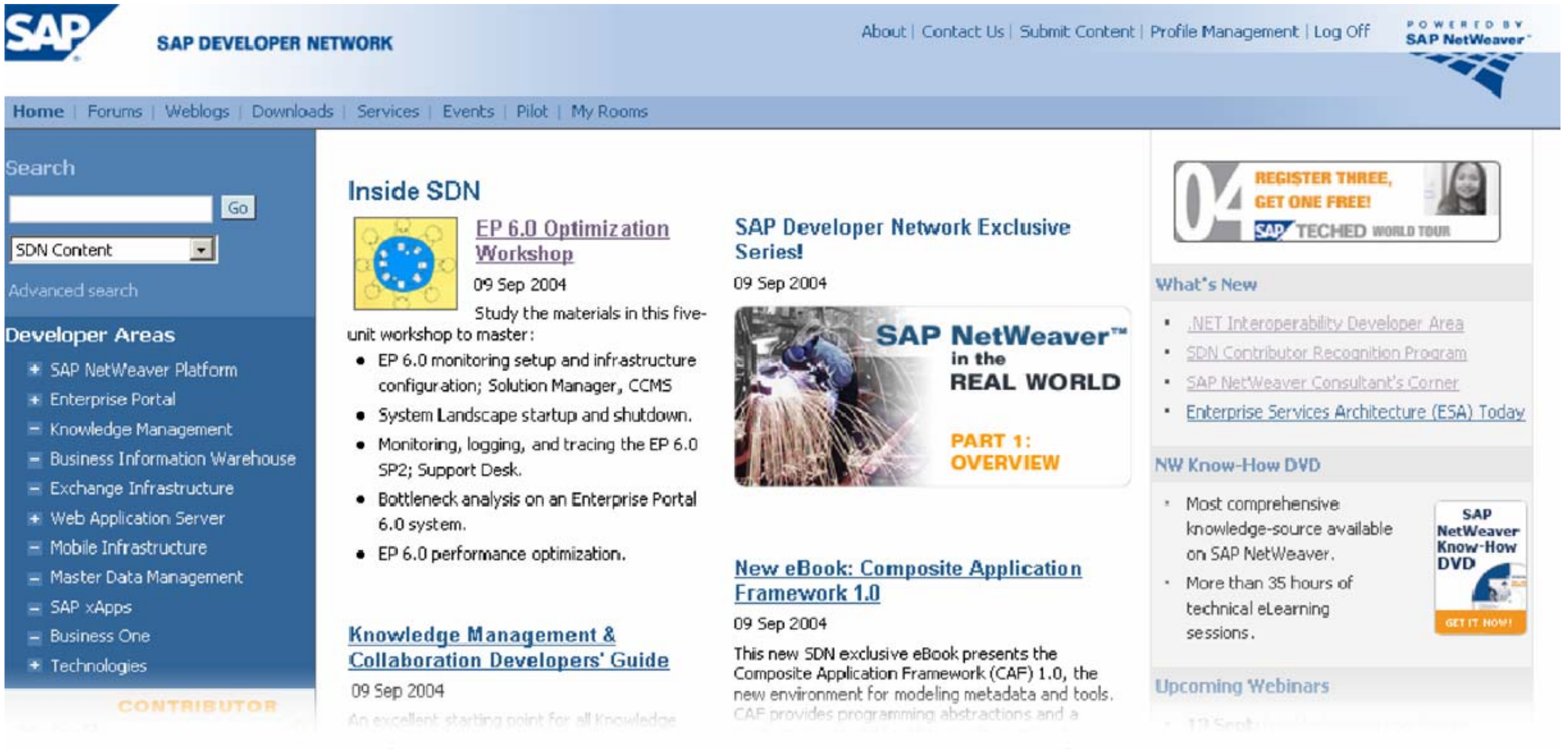
SCUR201, SAP Infrastructure Security

SAP Developer Network

Look for SAP TechEd ’04 presentations and videos on the SAP Developer Network.

Coming in December.

http://www.sdn.sap.com/



SAP DEVELOPER NETWORK

About | Contact Us | Submit Content | Profile Management | Log Off

POWERED BY SAP NetWeaver

Home | Forums | Weblogs | Downloads | Services | Events | Pilot | My Rooms

Search

Go

SDN Content

Advanced search

Developer Areas

- SAP NetWeaver Platform
- Enterprise Portal
- Knowledge Management
- Business Information Warehouse
- Exchange Infrastructure
- Web Application Server
- Mobile Infrastructure
- Master Data Management
- SAP xApps
- Business One
- Technologies

CONTRIBUTOR

Inside SDN

EP 6.0 Optimization Workshop

09 Sep 2004

Study the materials in this five-unit workshop to master:

- EP 6.0 monitoring setup and infrastructure configuration; Solution Manager, CCMS
- System Landscape startup and shutdown.
- Monitoring, logging, and tracing the EP 6.0 SP2; Support Desk.
- Bottleneck analysis on an Enterprise Portal 6.0 system.
- EP 6.0 performance optimization.

Knowledge Management & Collaboration Developers' Guide

09 Sep 2004

An excellent starting point for all Knowledge

SAP Developer Network Exclusive Series!

09 Sep 2004

SAP NetWeaver™ in the REAL WORLD

PART 1: OVERVIEW

New eBook: Composite Application Framework 1.0

09 Sep 2004

This new SDN exclusive eBook presents the Composite Application Framework (CAF) 1.0, the new environment for modeling metadata and tools. CAF provides programming abstractions and a

04 REGISTER THREE, GET ONE FREE! SAP TECHED WORLD TOUR

What's New

- .NET Interoperability Developer Area
- SDN Contributor Recognition Program
- SAP NetWeaver Consultant's Corner
- Enterprise Services Architecture (ESA) Today

NW Know-How DVD

- Most comprehensive knowledge-source available on SAP NetWeaver.
- More than 35 hours of technical eLearning sessions.

SAP NetWeaver Know-How DVD

GET IT NOW!

Upcoming Webinars

THE BEST-RUN BUSINESSES RUN SAP

Appendix

THE BEST-RUN BUSINESSES RUN SAP

References

JAAS Documentation

- http://java.sun.com/j2se/1.4.2/docs/guide/security/jaas/JAASRefGuide.html#LoginModule

JAAS LoginModule Developer Guide

- http://java.sun.com/j2se/1.4.2/docs/guide/security/jaas/JAASLMDevGuide.html

SAP NetWeaver ’04 Documentation

- http://help.sap.com (SAP NetWeaver – Release ’04)

SAP NetWeaver ’04 Custom Login Module Tutorial

- http://service.sap.com/security

SAP Logging and Tracing Tutorial

- https://www.sdn.sap.com/irj/servlet/prt/portal/prtroot/com.sapportals.km.docs/documents/a1-8-4/Tutorial%20-%20Logging%20and%20Tracing%20Mechanism%20in%20SAP.pdf

SAP Logging and Tracing JavaDocs

- https://www.sdn.sap.com/sdn/developerareas/ep.sdn?page=javadoc.htm

Copyright 2004 SAP AG. All Rights Reserved

THE BEST-RUN BUSINESSES RUN SAP