JAAS Login Modules
For SAP WebAS JAVA 6.40
Meeting will be recorded!
Oliver Nocon
SAP NetWeaver RIG EMEA, SAP AG
JAAS Overview

Authentication SAP WebAS Java 6.40

Development (6.40)

Deployment and Configuration (6.40)
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

- application independent
- represent the “authentication part” of JAAS
- implement a specific type of authentication technology (UserId/Password, certificate, ..)
- are plugged into the J2EE Engine under the application

Login Context

- application specific
- here the login modules are called
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

## JAAS Login Module – Authentication Success

<table>
<thead>
<tr>
<th>Login Module</th>
<th>1 - required</th>
<th>2 - sufficient</th>
<th>3 - requisite</th>
<th>4 - optional</th>
<th>Overall Authentication</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>pass</td>
<td>fail</td>
<td>fail</td>
<td>pass</td>
<td>pass</td>
</tr>
<tr>
<td>Login Module 1 - required</td>
<td>pass</td>
<td>pass</td>
<td>pass</td>
<td>fail</td>
<td>fail</td>
</tr>
<tr>
<td></td>
<td>fail</td>
<td>fail</td>
<td>fail</td>
<td>pass</td>
<td>fail</td>
</tr>
<tr>
<td>Login Module 2 - sufficient</td>
<td>fail</td>
<td>fail</td>
<td>fail</td>
<td>pass</td>
<td>fail</td>
</tr>
<tr>
<td></td>
<td>fail</td>
<td>fail</td>
<td>fail</td>
<td>pass</td>
<td>fail</td>
</tr>
<tr>
<td>Login Module 3 - requisite</td>
<td>*</td>
<td>pass</td>
<td>fail</td>
<td>*</td>
<td>pass</td>
</tr>
<tr>
<td></td>
<td>pass</td>
<td>fail</td>
<td>fail</td>
<td>*</td>
<td>fail</td>
</tr>
<tr>
<td>Login Module 4 - optional</td>
<td>*</td>
<td>pass</td>
<td>fail</td>
<td>*</td>
<td>pass</td>
</tr>
<tr>
<td></td>
<td>*</td>
<td>fail</td>
<td>fail</td>
<td>*</td>
<td>fail</td>
</tr>
<tr>
<td>Overall Authentication:</td>
<td>pass</td>
<td>pass</td>
<td>pass</td>
<td>fail</td>
<td>fail</td>
</tr>
<tr>
<td></td>
<td>fail</td>
<td>fail</td>
<td>fail</td>
<td>fail</td>
<td>fail</td>
</tr>
<tr>
<td></td>
<td>fail</td>
<td>fail</td>
<td>fail</td>
<td>fail</td>
<td>fail</td>
</tr>
</tbody>
</table>
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](http://java.sun.com/j2se/1.4.2/docs/guide/security/jaas/JAASLMDevGuide.html)
Login Module Authentication Process

Login Module 1
Login Module 2
Login Module 3
Login Module 4

Overall Authentication:

Phase 1
- abort() → login() → commit()

Phase 2
- abort() → login() → commit()
- abort() → login() → commit()
- abort() → login() → commit()

Failed
Succeeded
JAAS Overview

Authentication (6.40)

Development (6.40)

Deployment and Configuration (6.40)
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
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 takes care of 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 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 a JAAS control flags attached to each module

The login module stack belongs to a policy configuration.

In case 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
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
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 digested (encoded).

ClientCertificateLoginModule
- X.509 certificate logon

SAMLLoginModule
- uses a 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
- login performed using download.ear.
- uses tickets generated by security service on the engine.
Available Service Login Modules

CSILoginModule
- login performed using the IIOP service.

CallerImpersonationMappingLoginModule
- Caller Impersonation Authentication

ConfiguredIdentityMappingLoginModule
- Configured Identity Mapping Authentication

CredentialsMappingLoginModule
- Credentials Mapping Authentication

EvaluateAssertionTicketLoginModule
- verifies SAP Authentication Assertion tickets

PrincipalMappingLoginModule
- Principal Mapping Authentication
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)

Allow to enforce different authentication mechanisms for different content (e.g., iViews)

Re-authentication required in case 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>
</document>
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

```
Authscheme references       authscheme
default                   uidpwdlogon
UserAdminScheme           basicauthentication
...                       guest
```
It is possible to configure a url that will be called during the logout 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, ...)

* planned for SAP NetWeaver SP Stack 05
Extend the abstract class

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

implement the following methods:

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

```java
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 {...}
}
```
Method initialize()

Call the method initialize() of the superclass AbstractLoginModule

Initialize the login module with the relevant authentication and state information.

Retrieve login module configuration:

- `options.get("<ParameterName>");`
- configuration of options is done through the visual administrator

```java
public void initialize(Subject subject, CallbackHandler callbackHandler,
                      Map sharedState, Map options) {
    super.initialize(subject, callbackHandler, sharedState, options);
    _subject = subject;
    _callbackHandler = callbackHandler;
    _sharedState = sharedState;
    _options = options;

    // initialize any configured options
    _debug = "true".equalsIgnoreCase((String)options.get("debug"));
}
```
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)

  - Makes HTTP servlet request available to the JAAS login module
    - ...\j2ee\cluster\serverX\bin\system\util.jar
  - 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 sharedState:

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

You may check if the user in the shared state corresponds to the user you authenticated in your login module in case the shared state already holds a user name.
Method login() – NameCallback/PasswordCallback

```java
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) {
        _shouldBeIgnored=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);
}
```
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

```java
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;
    }
}
```
Clean out authentication and state information if phase 1 of authentication fails.

```java
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

```java
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;
    }
}
```
For logging and tracing you use the standard logging/tracing functionality of the WebAS Java.

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

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

References (you need an SDN account):

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
- In case of using NameCallback and PasswordCallback 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

1. Click on 'New Project...'

2. Select 'J2EE Application'

3. Enter 'J2EE_LoginModules' as the project name and click 'Finish'

© SAP AG 2004, Title of Presentation / Speaker Name / 38

THE BEST-RUN BUSINESSES RUN SAP
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

1. Select 'Export' from the menu.

2. Select 'Packaging' and click 'Next'.

3. Click 'Finish' to complete the process.

Options:
- Export generated class files and resources
- Export Java source files and resources
- Synchronize with the source code
- Overwrite existing files without warning
- Compress the contents of the JAR file
- No compression

Select the export destination:

- JAR file: [Path]

Browse...
Create a New J2EE Library Project

1. Select "Project..." from the New menu.
2. Create a new J2EE Library Project.
3. Specify the project name and choose the project contents.

© SAP AG 2004. Title of Presentation / Speaker Name / 41
Configure Provider.xml

Configure names, version numbers, … here
Add Jar File Containing Your Login Module(s)

1. Jars
2. Add/Remove
3. Select your jar file
4. OK/Cancel
Add Library References

1. select libraries you use:
- security_api
- com.sap.tc.Logging
- ...

2. Add / Remove

3. Create new / select library/interface/service

4. J2EE libraries/interfaces/service

5. OK / Cancel
Configure SDA Definition

select Software type: library
Create a Log Configuration for Your Project
Add and Configure a Log Destination

1. Add and configure a log destination.
2. Set the properties for your log destination.
Add and Configure Log Controller

1. Set the properties for your log controller

2. Add and Remove Log Controller

3. Log Controller Configuration
Build SDA File

1. Build Library Archive

2. Library Archive generation has been finished successfully.
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
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
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
Add New Login Modules to J2EE Engine (3)

1. 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
   - Class Name: com.sap.security.auth.IPCheckLoginModule
   - Display Name: IPCheckLoginModule
   - Description:
   - Options editor:
   - Options:
     - debug: true
     - AllowedOps:
   - Authentication Mechanisms
     - BASIC
     - FORM
     - CLIENT_CERT
     - DIGEST

3. OK
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

1. Select a login module

2. Modify

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

4. OK / Cancel

Select a login module

Set the properties for your login module

Options:
Go to the properties View of the Security Provider service

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

SAP NetWeaver ’04 Custom Login Module Tutorial
- http://service.sap.com/security

SAP Logging and Tracing Tutorial

SAP Logging and Tracing JavaDocs
No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Some software products marketed by SAP AG and its distributors contain proprietary software components of other software vendors.

Microsoft®, WINDOWS®, NT®, EXCEL®, Word®, PowerPoint® and SQL Server® are registered trademarks of Microsoft Corporation.

IBM®, DB2®, DB2 Universal Database, OS/2®, Parallel Sysplex®, MVS/ESA, AIX®, S/390®, AS/400®, OS/390®, OS/400®, iSeries, pSeries, xSeries, zSeries, z/OS, AFP, Intelligent Miner, WebSphere®, Netfinity®, Tivoli®, Informix and Informix® Dynamic ServerTM are trademarks of IBM Corporation in USA and/or other countries.

ORACLE® is a registered trademark of ORACLE Corporation.

UNIX®, X/Open®, OSF/1®, and Motif® are registered trademarks of the Open Group.

Citrix®, the Citrix logo, ICA®, Program Neighborhood®, MetaFrame®, WinFrame®, VideoFrame®, MultiWin® and other Citrix product names referenced herein are trademarks of Citrix Systems, Inc.

HTML, DHTML, XML, XHTML are trademarks or registered trademarks of W3C®, World Wide Web Consortium, Massachusetts Institute of Technology.

JAVA® is a registered trademark of Sun Microsystems, Inc.

JAVASCRIPT® is a registered trademark of Sun Microsystems, Inc., used under license for technology invented and implemented by Netscape.

MarketSet and Enterprise Buyer are jointly owned trademarks of SAP AG and Commerce One.

SAP, R/3, mySAP, mySAP.com, xApps, xApp, SAP NetWeaver and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries all over the world. All other product and service names mentioned are the trademarks of their respective companies. Data contained in this document serves information purposes only. National product specifications may vary.
Weitergabe und Vervielfältigung dieser Publikation oder von Teilen daraus sind, zu welchem Zweck und in welcher Form auch immer, ohne die aus-drückliche schriftliche Genehmigung durch SAP AG nicht gestattet. In dieser Publikation enthaltene Informationen können ohne vorherige Ankündigung geändert werden.

Die von SAP AG oder deren Vertriebsfirmen angebotenen Softwareprodukte können Softwarekomponenten auch anderer Softwarehersteller enthalten.

Microsoft®, WINDOWS®, NT®, EXCEL®, Word®, PowerPoint® und SQL Server® sind eingetragene Marken der Microsoft Corporation.

IBM®, DB2®, DB2 Universal Database, OS/2®, Parallel Sysplex®, MVS/ESA, AIX®, S/390®, AS/400®, OS/390®, OS/400®, iSeries, pSeries, xSeries, zSeries, z/OS, AFP, Intelligent Miner, WebSphere®, Netfinity®, Tivoli®, Informix und Informix® Dynamic ServerTM sind Marken der IBM Corporation in den USA und/oder anderen Ländern.

ORACLE® ist eine eingetragene Marke der ORACLE Corporation.

UNIX®, X/Open®, OSF/1® und Motif® sind eingetragene Marken der Open Group.

Citrix®, das Citrix-Logo, ICA®, Program Neighborhood®, MetaFrame®, WinFrame®, VideoFrame®, MultiWin® und andere hier erwähnte Namen von Citrix-Produkten sind Marken von Citrix Systems, Inc.


JAVA® ist eine eingetragene Marke der Sun Microsystems, Inc.

JAVASCRIPT® ist eine eingetragene Marke der Sun Microsystems, Inc., verwendet unter der Lizenz der von Netscape entwickelten und implementierten Technologie.

MarketSet und Enterprise Buyer sind gemeinsame Marken von SAP AG und Commerce One.