

JBoss Federated SSO Framework

Presenter: Sohil Shah

Company: Red Hat, JBoss Division

Title: Software Engineer

Projects: JBoss Portal, JBoss Federated SSO

Date: February 14, 2008

Agenda

- Single Sign On
- Benefits of SSO
- High level overview
 - JBoss Federated SSO architecture
- Integration Details for developers/end-users
- Future Roadmap

What is Single Sign On (SSO)?

- Single Sign On (SSO) is a specialized form of user authentication that enables a user to be authenticated once, and gain access to resources on multiple systems/web applications during that session.
- It allows a federation/collection of applications that can trust each other and authenticate each other via a shared set of credentials

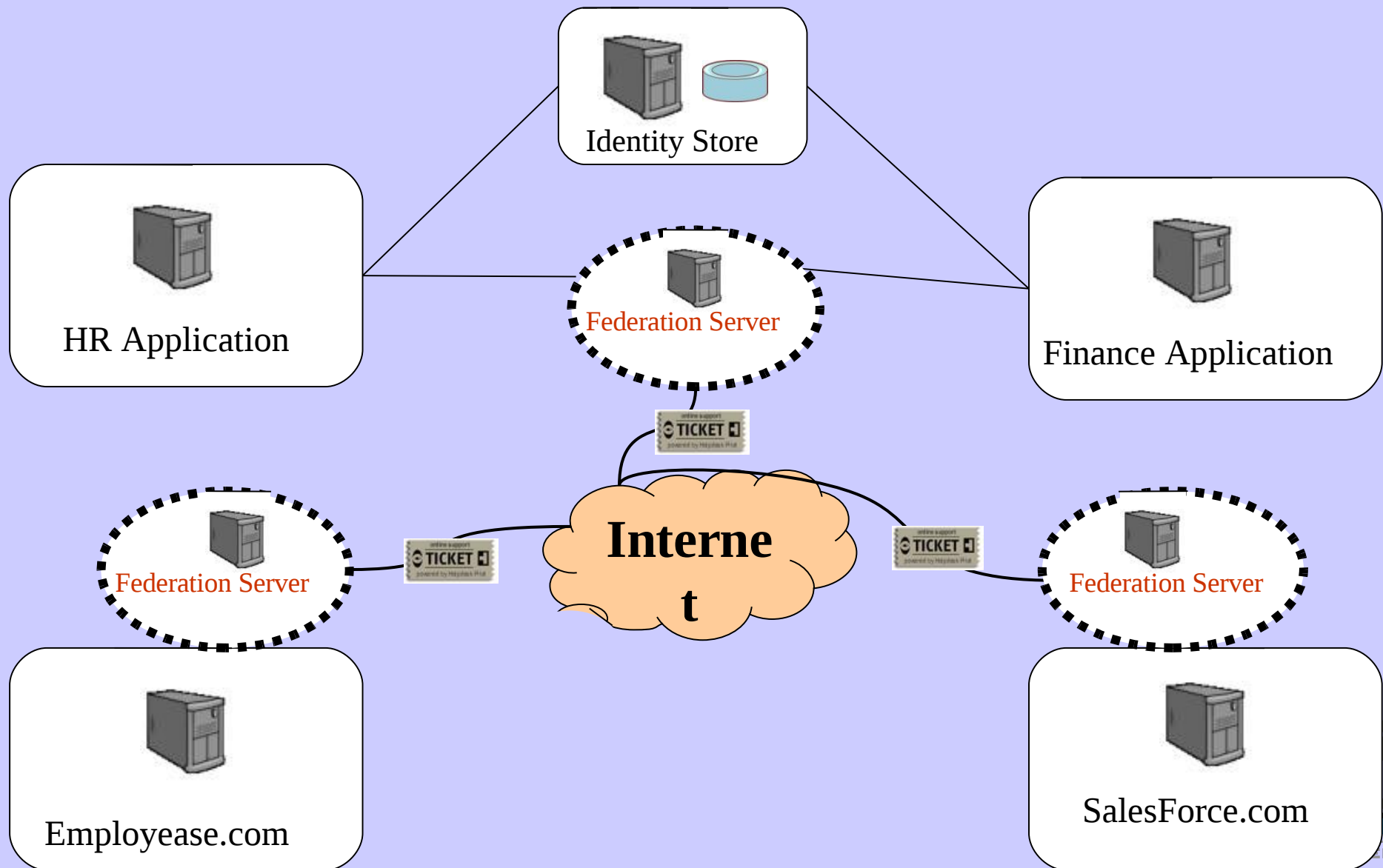
Benefits of Single Sign On

- Helps consolidate silos of identity stores that have cropped up over time with multiple web applications.
- Improves user account provisioning process dramatically.
- Provides a better end user experience using web SSO support.
- Improves efficiency when integrating user access to new applications including 3rd party ASP services like Salesforce.com.
- Enables secure intra-company access to applications between enterprises and their partners, suppliers, and customer organizations.

- Two Independent JBoss Seam applications running in different web domains

Demo

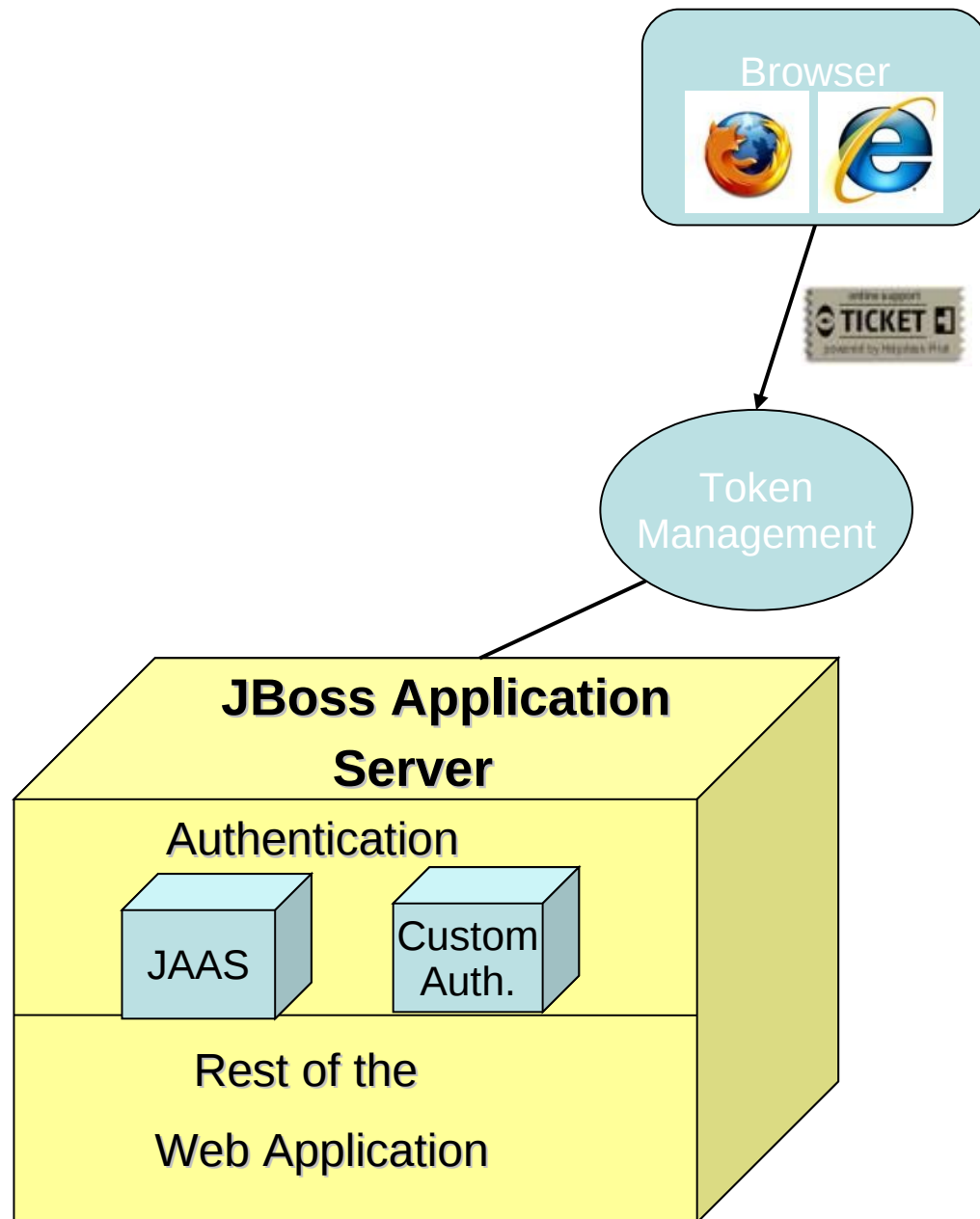
Architecture – Bird's Eye View



The JBoss Federated SSO Framework

- The JBoss Federated SSO Framework consists of the following components:
 - Token Management Framework
 - Identity Connector Framework
 - Web Application Coordinator
 - Federation Server

Token Marshalling Framework



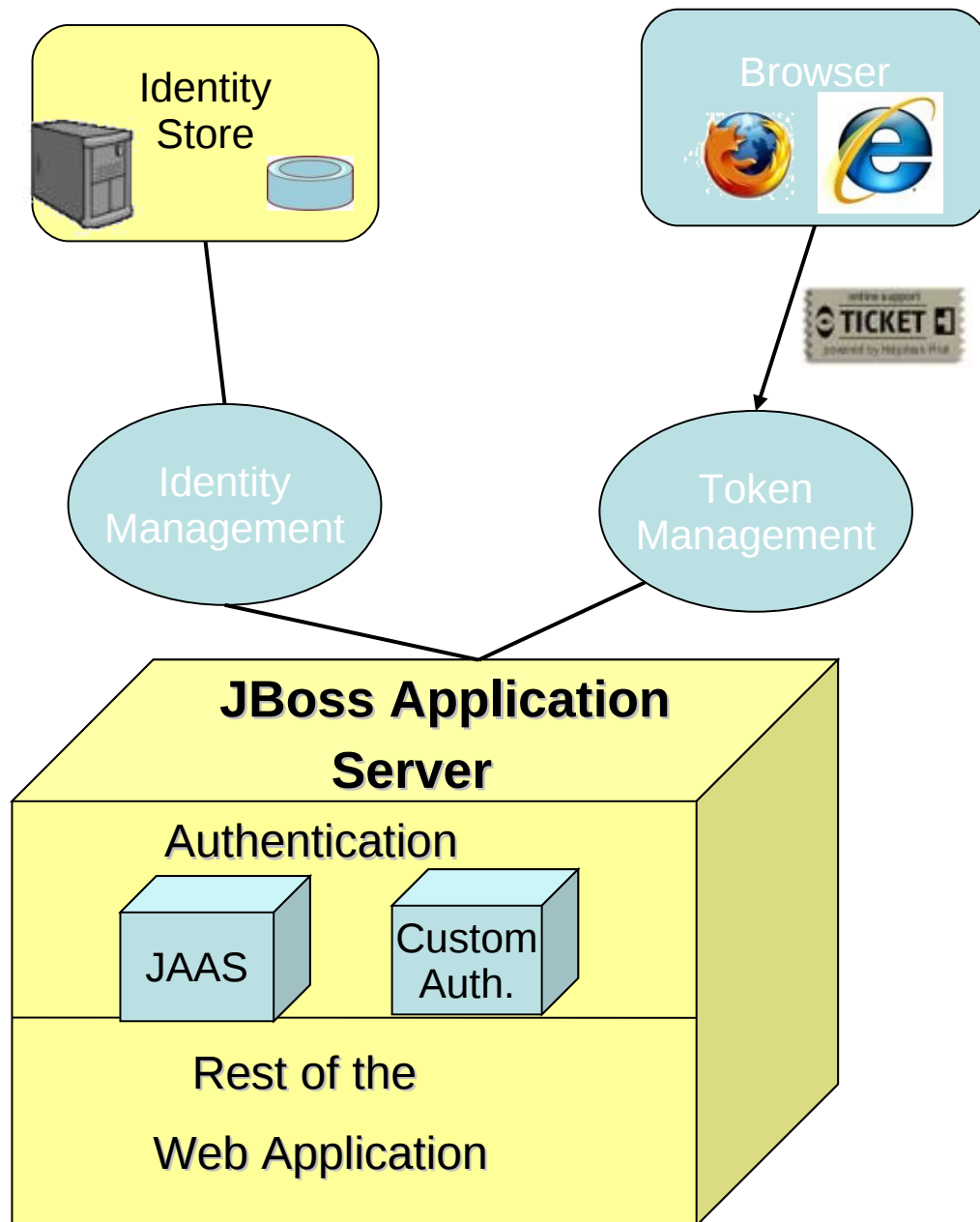
Token Marshalling Framework

- The Federation Token – an Authentication Assertion provided to the user for that particular web session.
- The framework comes with a SAML compliant marshaller/unmarshaller out-of-the-box.
- The pluggable nature of the framework allows you to plug-in other token formats (like Kerberos tickets, etc.) that are fit for your particular federation.

Token Marshalling Framework

```
<Response xmlns="urn:oasis:names:tc:SAML:1.0:protocol"
xmlns:saml="urn:oasis:names:tc:SAML:1.0:assertion"
  xmlns:samlp="urn:oasis:names:tc:SAML:1.0:protocol" IssueInstant="2006-11-
20T22:48:26.468Z"
  MajorVersion="1" MinorVersion="1"
  ResponseID="_e719a0cf3e6207007f589805d6f8b1a9">
  <Status> <StatusCode Value="samlp:Success"></StatusCode> </Status>
  <Assertion xmlns="urn:oasis:names:tc:SAML:1.0:assertion"
  AssertionID="_14e7d4285c55de2b24038fa5caf21852" IssueInstant="2006-11-20T22:48:26.468Z"
  Issuer="seam_hotel_reservation" MajorVersion="1" MinorVersion="1">
  <AuthenticationStatement AuthenticationInstant="2006-11-20T22:48:26.468Z"
    AuthenticationMethod="urn:oasis:names:tc:SAML:1.0:am:password">
    <Subject>
      <NameIdentifier Format="urn:oasis:names:tc:SAML:1.1:nameid-
format:unspecified">user1</NameIdentifier>
    </Subject>
  </AuthenticationStatement>
  <AttributeStatement>
    <Subject>
      <NameIdentifier Format="urn:oasis:names:tc:SAML:1.1:nameid-
format:unspecified">user1</NameIdentifier>
    </Subject>
    <Attribute AttributeName="secret" AttributeNamespace="jbossso:secret">
      <AttributeValue>82de2e3999283ce7c35a6f3575058ed3047363105df208f8acb66e0893de9999137b191
      </AttributeValue>
    </Attribute>
  </AttributeStatement>
  </Assertion>
</Response>
```

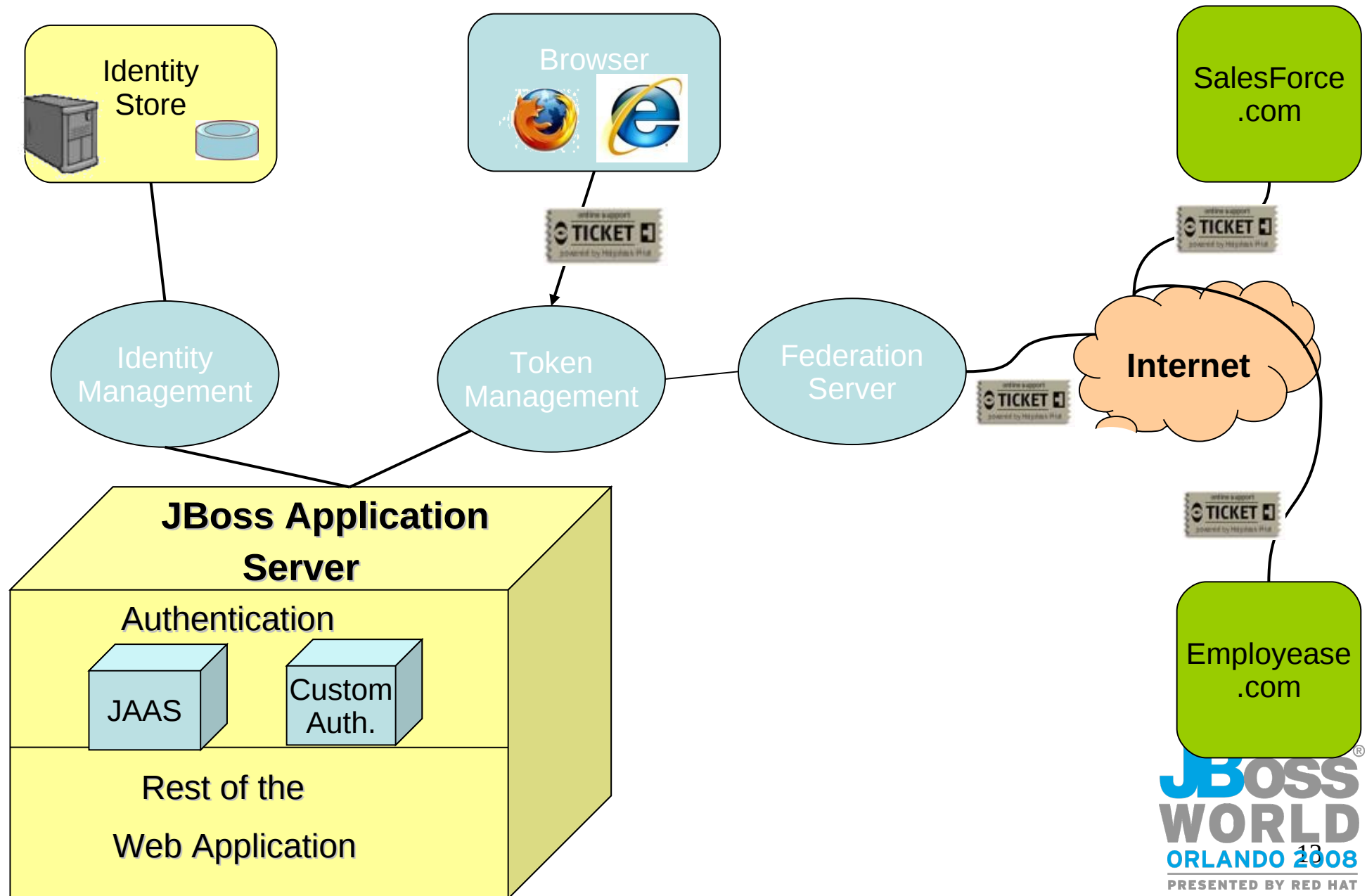
Identity Connector Framework



Identity Connector Framework

- A flexible/pluggable Java API to connect to central identity stores
- Comes with a built-in provider to connect to LDAP based stores
- Its pluggable nature lets an organization create custom providers that can integrate with custom identity stores or third party systems like SiteMinder, Netscape Identity Server etc

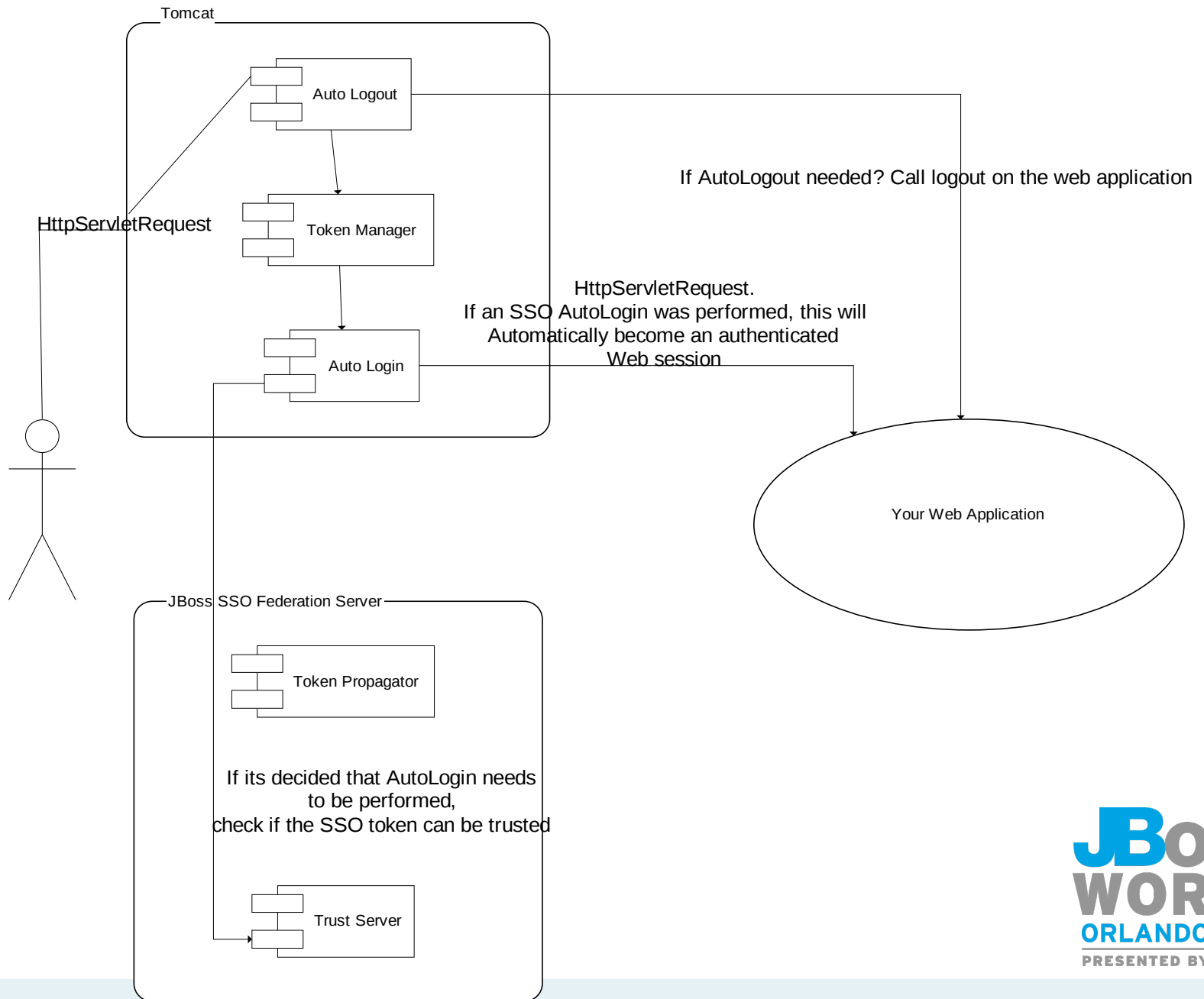
Federation Server



Federation Server

- A Federation Server is used for securely propagating the federation token across a federation of trusted web applications
- Think of Federation Server as a Single Sign On session manager allowing web applications to trust each other and not challenge each other for authentication once a Single Sign On session is successfully established
- The Federation Server is particularly useful for integrating third party ASP services like Salesforce.com into your federation, or even business partners looking to integrate their independent web applications
- A trust component is integrated with the Federation Server which informs the SSO Engine, that whether a particular authentication assertion from a partner application should be honored or not

How do I integrate this framework into my web application?



How do I integrate this framework into my web application?

- Step 1: Integrate an implementation of `org.jboss.security.idm.LoginProvider` into the SSO system
 - A LoginProvider represents a data source component that extracts data from Identity Stores like LDAP repositories, thirdparty identity management systems like SiteMinder, or even custom JDBC based systems. It serves as an abstraction to Identity data related to authentication on the system.
 - The core SSO Framework ships with an implementation of the LoginProvider which integrates with data stored in an LDAP repository.
 - The implementation is called **`org.jboss.security.idm.LDAPIdentityProvider`** and is tested to work with OpenLDAP and RedHat Directory Server

How do I integrate this framework into my web application?

- Brief Overview of LoginProvider

```
public Identity read(Principal principal) throws IdentityException;  
public boolean exists(Principal principal) throws IdentityException;  
public boolean login(Principal principal,byte[] password) throws IdentityException;  
public Collection readAllRoles() throws IdentityException;  
public void processSSOLoginNotification(LoginContext loginContext) throws  
IdentityException;
```

How do I integrate this framework into my web application?

- Step 2 : Register this LoginProvider with the SSO Engine using the following sso.cfg.xml file in the jboss-sso.sar/conf directory

```
<identity-management>
<login>
  <provider id="si://jboss-sso/ldap/login"
class="org.jboss.security.idm.ldap.LDAPIdentityProvider">
    <property name="connectionURL">{value}</property>
    <property name="username">{value}</property>
    <property name="password">{value}</property>
  </provider>
</login>
</identity-management>
```

How do I integrate this framework into my web application?

- Step 3: Configure the Federated SSO Trust Server
- This is done by modifying the following file:
/jboss-ss0.sar/conf/ss0.cfg.xml

Sample Configuration:

```
<ss0-processor>  
  <processor class="org.jboss.security.saml.JBossSingleSignOn">  
    <property  
name="trustServer">https://node1.jboss.org:8443/federate/trust</property>  
  </processor>  
</ss0-processor>
```

How do I integrate this framework into my web application?

- Step 4: Configure your own web application to activate the Federated SSO components
- Which Federated SSO components you need to configure depends on the type of authentication mechanism used by your web application. This configuration is split into two camps:
 - Integration for web applications using their own proprietary/custom authentication mechanism **(Non-JAAS)**
 - Integration for web applications using the container provided JAAS authentication mechanism

Web Applications using their own authentication mechanism (Non-JAAS)?

- Step 1: Integrate the proper tomcat valves inside your web application by configuring the /WEB-INF/context.xml of your web application as follows:
- Add the following valves in the proper order to your context.xml

```
<Context>  
  <Valve className="org.jboss.security.valve.PlainSSOAutoLogout"  
logoutURL="{url that performs logout on this web application}"/>  
  
  <Valve className="org.jboss.security.valve.PlainSSOTokenManager"  
assertingParty="{some value that identifies this web application as an SSO partner}"/>  
  
  <Valve className="org.jboss.security.valve.PlainSSOAutoLogin"/>  
</Context>
```

Web Applications using their own authentication mechanism?

- Step 2: When the authentication usecase is executed within your web application via the login screen or some other mechanism, part of executing that process, when the login is successful, send a notification of this event to the SSO Engine using the following API call:

```
org.jboss.security.saml.SSOManager.  
processManualLoginNotification(HttpServletRequest request,String user)
```

- Step 3 : When the SSOEngine performs an automatic login in response to a trusted SSO Token, it will send the following notification on your LoginProvider:

```
processSSOLoginNotification(LoginContext). Here, you can handle any web  
application environment necessary to setup an authenticated user session
```

- Note: When using this manner of authentication, web applications will be able to get the Principal logged in to the system using the following API call:

```
org.jboss.security.saml.SSOManager.getUserPrincipal(HttpServletRequest)
```

Web Applications using JAAS as the authentication mechanism?

- Step 1: Integrate the proper tomcat valves inside your web application by configuring the /WEB-INF/context.xml of your web application as follows:
- Add the following valves in the proper order to your context.xml

```
<Context>  
  <Valve className="org.jboss.security.valve.SSOAutoLogout" logoutURL="{url  
that performs logout on this web application}"/>  
  
  <Valve className="org.jboss.security.valve.SSOTokenManager"  
assertingParty="{some value that identifies this web application as an SSO partner}"/>  
  
  <Valve className="org.jboss.security.valve.SSOAutoLogin" authType="FORM"/>  
</Context>
```

Web Applications using JAAS as the authentication mechanism?

- Step 2: Integrate the JAAS Login Module with the web application. Here is a sample JAAS configuration file:

For more details with JAAS configuration look at:

<http://labs.jboss.com/wiki/Jbosssso>

```
<policy>
  <application-policy name="{your web application identifier}">
    <authentication>
      <login-module
code="org.jboss.security.idm.UsernameAndPasswordLoginModule" flag="required">
        <module-option name="unauthenticatedIdentity">guest</module-
option>
        <module-option name="password-stacking">useFirstPass</module-
option>
        <module-option
name="authenticatedRoles">Authenticated,RegisteredUsers</module-option>
      </login-module>
    </authentication>
  </application-policy>
</policy>
```

Out-of-the-Box Integration – JBoss Portal

- The Framework comes built-in
 - JBoss Portal Web Coordinator
 - JBoss Portal Identity Connector, leverages Portal's Identity component
- JBoss Portal leverages Tomcat/JAAS based authentication
- Integration is seamless
- You can easily create independent instances of JBoss Portal and loosely integrate them

Future Roadmap

- OpenID Support
- Support for federated web services using standards like WS-Security, WS-Trust, and WS-Policy
- Out-of-the-Box integration with Seam
- Out-of-the-Box integration with JBoss ESB
- Adding Support for more authentication methods like Certificate based, and Secure Remote Password (SRP) protocol
- Add federated provisioning to the Federation Server to facilitate business functions like account linking & account synchronization.

Questions?