



JBoss Enterprise Application Platform 5

Negotiation User Guide

for JBoss Enterprise Application Platform 5

Edition 5.2.0

Last Updated: 2017-10-13

JBoss Enterprise Application Platform 5 Negotiation User Guide

for JBoss Enterprise Application Platform 5
Edition 5.2.0

Darran Lofthouse

Eva Kopalová

Edited by

Petr Penicka

Russell Dickenson

Scott Mumford

Legal Notice

Copyright © 2012 Red Hat, Inc.

This document is licensed by Red Hat under the [Creative Commons Attribution-ShareAlike 3.0 Unported License](#). If you distribute this document, or a modified version of it, you must provide attribution to Red Hat, Inc. and provide a link to the original. If the document is modified, all Red Hat trademarks must be removed.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux ® is the registered trademark of Linus Torvalds in the United States and other countries.

Java ® is a registered trademark of Oracle and/or its affiliates.

XFS ® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL ® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js ® is an official trademark of Joyent. Red Hat Software Collections is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack ® Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Abstract

The JBoss Negotiation Guide is aimed at system administrators and developers, who wish to set up the SPNEGO authentication on their JBoss Enterprise Application Platform. This guide provides instructions for its configuration and additional details on the setup of the AdvancedLdapLoginModule, which allows integration of the SPNEGO authentication with an LDAP server.

Table of Contents

CHAPTER 1. INTRODUCTION	4
1.1. SPNEGO AUTHENTICATION PROCESS	4
1.2. CONFIGURATION OVERVIEW	4
CHAPTER 2. APPLICATION SERVER CONFIGURATION	6
2.1. ADDING THE SPNEGO AUTHENTICATOR	6
2.2. DEFINING SERVER SECURITY DOMAIN	6
2.3. DEFINING APPLICATION SECURITY DOMAIN	8
2.4. ROLE MAPPING	9
2.4.1. Setting up Role Mapping with a Roles Properties File	9
2.4.2. Setting up Role Mapping with an LDAP Server	9
2.4.2.1. Defining Initial LDAP Context with GSSAPI	10
2.4.2.2. Defining DN Search	10
2.4.2.2.1. User Authentication	11
2.4.2.3. Defining Role Search	11
2.4.3. Examples of LDAP Configuration with the SPNEGO Module	12
2.4.3.1. Chained Configuration on FreeIPA	13
2.4.3.2. Chained Configuration on Active Directory	13
CHAPTER 3. TRACE LOGGING	15
3.1. CONFIGURING MESSAGE TRACING	15
CHAPTER 4. PASSING AUTHENTICATION PROPERTIES TO THE SERVER	17
4.1. PASSING THE PROPERTIES FROM THE COMMAND LINE	17
4.2. ADDING THE PROPERTIES TO THE SYSTEM PROPERTIES	17
4.2.1. Multiple KDCs	18
CHAPTER 5. CONFIGURING MICROSOFT ACTIVE DIRECTORY	19
5.1. USER ACCOUNT FOR THE APPLICATION SERVER	19
5.1.1. Creating Server User	19
5.2. EXPORTING KEYTAB	23
CHAPTER 6. CONFIGURING FREEIPA	25
6.1. CREATING SERVICE PRINCIPAL	25
6.2. EXPORTING KEYTAB	27
CHAPTER 7. CONFIGURING WEB BROWSERS	29
7.1. CONFIGURING INTERNET EXPLORER	29
7.2. CONFIGURING FIREFOX	31
CHAPTER 8. NEGOTIATION TOOLKIT	34
8.1. FRONT PAGE	34
8.2. BASIC NEGOTIATION	35
8.3. SECURITY DOMAIN TEST	37
8.4. SECURED	39
CHAPTER 9. CONFIGURING WEB APPLICATIONS	41
APPENDIX A. ADVANCED LDAP LOGIN MODULE: FULL LDAP AUTHENTICATION	42
A.1. CONFIGURATION	42
A.1.1. Defining Initial LDAP Context	42
A.1.2. Defining DN Search	43
A.1.3. User Authentication	43
A.1.4. Defining Role Search	44

A.2. EXAMPLES OF FULL LDAP AUTHENTICATION	45
A.2.1. Full LDAP Authentication for Active Directory	46
A.2.2. Full LDAP Authentication for Free IPA	47
APPENDIX B. REVISION HISTORY	49

CHAPTER 1. INTRODUCTION

JBoss Negotiation is a component of JBoss Enterprise Application Platform, which provides the SPNEGO-based (Simple and Protected Negotiation) SSO (Single Sign On) mechanism.

JBoss Negotiation is located in `$JBOSS_HOME/jboss-as/common/lib/jboss-negotiation.jar`

SPNEGO is a Generic Security Services Application Program Interface (GSSAPI) mechanism for client-server authentication. It allows silent authentication to remote systems and access to security services. It can also delegate user credentials to a remote system so the remote system can contact further systems on behalf of the user.

1.1. SPNEGO AUTHENTICATION PROCESS

Generally, the client sends the input credentials to the server and the login module of the server verifies the credentials against its credential store when a user is authenticating to a server. SPNEGO authentication differs in several aspects:

1. The application server authenticates itself against the KDC and obtains a ticket before it can authenticate the user.
2. Only then, the server prompts the client to authenticate. The client responds with a SPNEGO token and the server uses its own ticket to decode client's ticket and then responds to the client.
3. A client can request the server to authenticate itself if required.
4. A client can delegate its credentials to the server so that the server can call other systems on behalf of the calling client.

JBoss Negotiation is typically useful in the following scenario:

- The user logs into a desktop computer with a log in that is governed by an Active Directory domain or FreeIPA.
- The user launches a web browser and accesses a web application that uses JBoss Negotiation.
- The web browser transfers the desktop credentials to the web application.



IMPORTANT

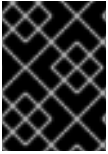
You can configure Active Directory and FreeIPA to use JBoss Negotiation (refer to [Chapter 6, Configuring FreeIPA](#) and [Chapter 5, Configuring Microsoft Active Directory](#)).

1.2. CONFIGURATION OVERVIEW

To have your environment configured to use JBoss Negotiation, you need to do the following:

- Configure your application server to use JBoss Negotiation (refer to [Chapter 2, Application Server Configuration](#)).
- Optionally configure Active Directory or FreeIPA to use JBoss Negotiation (refer to [Chapter 5, Configuring Microsoft Active Directory](#) or [Chapter 6, Configuring FreeIPA](#)).

- Configure client web browsers to use JBoss Negotiation (refer to [Chapter 7, Configuring Web Browsers](#)).
- Test the setup with Negotiation Toolkit (refer to [Chapter 8, Negotiation Toolkit](#)).
- Configure your web applications to use JBoss Negotiation (refer to [Chapter 9, Configuring Web Applications](#)).



IMPORTANT

Before you configure your applications to use JBoss Negotiation test the setup with Negotiation Toolkit (refer to [Chapter 8, Negotiation Toolkit](#)).

CHAPTER 2. APPLICATION SERVER CONFIGURATION

To configure JBoss Negotiation to run on JBoss Enterprise Application Platform, you need to do the following:

- Extend the core authentication mechanism to support JBoss Negotiation (add the SPNEGO authenticator).
- Define the application security domain, which allows an application to communicate with the application server through the SPNEGOLoginModule.
- Define the server security domain, which allows the application server to authenticate itself to the KDC for the first time.

You may also need to configure the realm properties to allow the server to locate the authentication realm (Kerberos realm) if the server was not previously configured to do so.

JBoss Negotiation comes with Negotiation Toolkit, a web application, which allows you to test your SPNEGO setup. Consider using the application before testing on your own web applications (refer to [Chapter 8, Negotiation Toolkit](#)).

2.1. ADDING THE SPNEGO AUTHENTICATOR

To add the SPNEGO authenticator to the core authentication mechanism, do the following:

1. Open the `$JBOSS_HOME/server/PROFILE/deployers/jbossweb.deployer/META-INF/war-deployers-jboss-beans.xml` file for editing.
2. Locate the property `authenticators`.
3. Add the following entry to the property:

```
<property name="authenticators">
  <map class="java.util.Properties" keyClass="java.lang.String"
valueClass="java.lang.String">
    <entry>
      <key>SPNEGO</key>

      <value>org.jboss.security.negotiation.NegotiationAuthenticator</valu
e>
    </entry>
```

The key value is arbitrary; however, if you want to use the Negotiation Toolkit to test your server setup, make sure you use the `SPNEGO` value as the tool works only with the SPNEGO authenticator with this name.

2.2. DEFINING SERVER SECURITY DOMAIN

The application server must define a security domain to be able to authenticate to the KDC for the first time.



IMPORTANT

Krb5LoginModule can use a local credentials cache; however, this option is incompatible with the `storeKey` option, which is required by SPNEGO. Make sure the module does not use the local credentials cache.

To define a server security domain, do the following:

1. Open the `$JBOSS_HOME/server/$PROFILE/conf/login-config.xml` file for editing.
2. Define the application policy element with the authentication element with the following options:

storeKey

If `true` the private key is cached in the Subject (set to `true`).

useKeyTab

If `true` the key is loaded from a keyTab file (set to `true`).

principal

The attribute needs to state the full name of the principal to obtain from the keyTab file.

keyTab

The attribute defines the full path to the keyTab file with the server key (key for encrypting the information between the server and KDC).

doNotPrompt

If `true` password prompting is turned off (as this is a server, set to `true`).

debug

If `true` the system logs additional debug information to STDOUT.

Example 2.1. Server security domain

```
<application-policy name="host">
  <authentication>
    <login-module code="com.sun.security.auth.module.Krb5LoginModule"
      flag="required">
      <module-option name="storeKey">true</module-option>
      <module-option name="useKeyTab">true</module-option>
      <module-option
name="principal">HTTP/testserver@KERBEROS.JBOSS.ORG</module-option>
      <module-option
name="keyTab">/home/jboss_user/testserver.keytab</module-option>
      <module-option name="doNotPrompt">true</module-option>
      <module-option name="debug">true</module-option>
    </login-module>
  </authentication>
</application-policy>
```

2.3. DEFINING APPLICATION SECURITY DOMAIN

To allow an application to communicate with the application server through the SPNEGOLoginModule, you need to define the application security domain on the application server.

To define the application security domain, do the following:

1. Open the `$JBOSS_HOME/jboss-as/server/$PROFILE/conf/login-config.xml` file for editing.
2. Define a new application policy with the following chained configuration:

- o The SPNEGOLoginModule and its configuration with the following options:

```
<module-option name="password-stacking">useFirstPass</module-option>
```

The password-stacking option activates client-side authentication of clients with other login modules. Set the `password-stacking` option to `useFirstPass`, so the module looks first for a shared user name and password with `javax.security.auth.login.name` and `javax.security.auth.login.password` respectively (for further information refer to Password Stacking in the *Security Guide*).

```
<module-option name="serverSecurityDomain"> DomainName</module-option>
```

The `serverSecurityDomain` option defines the server security domain, which defines the authentication module (Kerberos) and server authentication properties (refer to [Section 2.2, “Defining Server Security Domain”](#)).

- o The login module which returns the roles of the authenticated user and its configuration options. You can make use of the `UsersRolesLoginModule` that obtains the user roles from a properties file or `AdvancedLdapLoginModule`, which obtains user roles from an LDAP server following GSSAPI. For further information refer to [Section 2.4, “Role Mapping”](#).

Example 2.2. Application Security Domain

```
<application-policy name="SPNEGO">
  <authentication>
    <login-module
      code="org.jboss.security.negotiation.spnego.SPNEGOLoginModule"
      flag="requisite">
      <module-option name="password-stacking">useFirstPass</module-
option>
      <module-option name="serverSecurityDomain">host</module-
option>
    </login-module>
    <login-module
      code="org.jboss.security.auth.spi.UsersRolesLoginModule"
      flag="required">
      <module-option name="password-stacking">useFirstPass</module-
option>
      <module-option name="usersProperties">props/spnego-
users.properties</module-option>
      <module-option name="rolesProperties">props/spnego-
roles.properties</module-option>
```

```

        </login-module>
    </authentication>
</application-policy>

```

In [Example 2.2, “Application Security Domain”](#) we have defined an application security domain called **SPNEGO** with two login modules:

- `org.jboss.security.negotiation.spnego.SPNEGOLoginModule` provides SPNEGO user authentication;
- `org.jboss.security.auth.spi.UsersRolesLoginModule` returns the roles of the user authenticated by the `SPNEGOLoginModule` (the roles are filtered from a users properties file).

2.4. ROLE MAPPING

Once the user has been authenticated against the KDC (this occurs through `org.jboss.security.negotiation.spnego.SPNEGOLoginModule`), the application server needs to obtain the user roles. The authentication can use either the `org.jboss.security.auth.spi.UsersRolesLoginModule` to obtain user roles from a `roles.properties` file or the `org.jboss.security.negotiation.AdvancedLdapLoginModule` to obtain user roles from an LDAP server.

2.4.1. Setting up Role Mapping with a Roles Properties File

To allow SPNEGO to obtain the roles of an authenticated user from a `roles.properties` file, do the following:

1. In the application security domain, set the second login module of the SPNEGO authentication to `org.jboss.security.auth.spi.UsersRolesLoginModule` (refer to [Example 2.2, “Application Security Domain”](#)) and provide the module options. Refer to `UsersRolesLoginModule` in the *Security Guide*.
2. If the application security domain is defined in the `$JBOSS_HOME/server/$PROFILE/conf/login-config.xml` file, define the user roles in the `$JBOSS_HOME/server/$PROFILE/conf/props/spnego-users.properties` file. Use the following pattern: *fullyQualifiedUserName=comma-separatedListOfRoles*

Example 2.3. roles.properties file

```

# A roles.properties file for use with the UsersRolesLoginModule
darranl@KERBEROS.JBOSS.ORG=Users,Admins

```

2.4.2. Setting up Role Mapping with an LDAP Server

The `AdvancedLdapLoginModule` allows you to obtain the roles of a user, who was previously authenticated against a KDC with the `SPNEGOLoginModule`. The `AdvancedLdapLoginModule` is based on the `LdapExtLoginModule` and follows GSSAPI.

**NOTE**

In this chapter we discuss the module in chained configuration with the SPNEGOLoginModule; however, you can use the module for both authentication and role look up from an LDAP server. For further details on such configuration, refer to [Appendix A, *Advanced LDAP Login Module: Full LDAP Authentication*](#)

To make use of the AdvancedLdapLoginModule in the chained configuration with the SPNEGOLoginModule, you need to chain it with the SPNEGOLoginModule in the SPNEGO application security domain: set the second login module of SPNEGO authentication to `org.jboss.security.negotiation.AdvancedLdapLoginModule` (refer to [Example 2.2, “Application Security Domain”](#)).

To set up the role mapping to an LDAP server, you need to do the following:

- Define InitialLdapContext properties: these properties are used to obtain LDAP connection (refer to [Section 2.4.2.1, “Defining Initial LDAP Context with GSSAPI”](#) ; for details on the Java API refer to <http://download.oracle.com/javase/6/docs/api/javax/naming/ldap/InitialLdapContext.html>).
- Define DN (Distinguished Name) properties: these properties are used to search for the authenticated user on the LDAP server (refer to [Section 2.4.2.2, “Defining DN Search”](#)).
- Define role search properties: these properties govern the role search on the LDAP server ([Section 2.4.2.3, “Defining Role Search”](#)).

The properties set on the login mode are passed into the InitialLdapContext constructor; that means you can make use of any of the options supported by the LdapCtxFactory.

2.4.2.1. Defining Initial LDAP Context with GSSAPI

To obtain the initial LDAP Context, define the following module properties for the AdvancedLdapLoginModule in the application security domain ([Section 2.3, “Defining Application Security Domain”](#)):

bindAuthentication

defines the authentication type (set the property value to `GSSAPI` to use GSSAPI-based authentication).

jaasSecurityDomain

defines the security domain that is used to obtain the subject required for the connection (refer to [Section 2.2, “Defining Server Security Domain”](#) for information defining the required `jaasSecurityDomain`).

2.4.2.2. Defining DN Search

After the module has created the LDAP initial context, it takes the provided username and searches for the user DN. To define the properties of the search, provide the following properties:

baseCtxDN

defines the fixed DN of the context to search for the user; consider that this is not the Distinguished Name of the location where the actual users are located but DN of the location where the objects containing the users are located (that is, for Active Directory, this is the DN with the user account).

baseFilter

defines the search filter used to locate the context of the user to authenticate; the input username/userDN as obtained from the login module callback substitutes the `{0}` expression. This substitution behavior comes from the standard `DirContext?.search(Name, String, Object[], SearchControls? cons)` method. A common example search filter is `(uid={0})`

searchTimeLimit

defines the timeout for the user and role search in milliseconds (defaults to 10000, that is 10 seconds).

**NOTE**

To disable the user DN search, omit the `baseCtxDN` property; the provided username will be used as the DN in the login module.

2.4.2.2.1. User Authentication

If the `AdvancedLdapLoginModule` is not the first login module and a previous login module has already authenticated the user, user authentication is skipped.

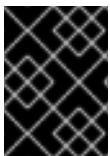
For user authentication, you can define the following property:

allowEmptyPassword

If empty (`length==0`) passwords are passed to the LDAP server. An empty password is treated as an anonymous login by an LDAP servers. Set the property to `false` to reject empty passwords or to `true` to allow the LDAP server to validate an empty password (the default is `false`).

2.4.2.3. Defining Role Search

The `AdvancedLdapLoginModule` passes the properties that define the search for a particular user and its roles to the LDAP server.

**IMPORTANT**

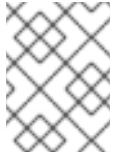
The following role search settings are similar to the `LdapExtLoginModule` settings; however, the recursion now finds the roles listed within a DN.

rolesCtxDN

defines the fixed DN of the context to search for user roles; consider that this is not the Distinguished Name of the location where the actual roles are but the DN of the location where the objects containing the user roles are (that is, for Active Directory, this is the DN where the user account is).

roleFilter

defines the search filter used to locate the roles of the authenticated user. The input username/userDN as obtained from the login module callback substitutes the `{0}` expression in the filter definition. The authenticated userDN substitutes the `{1}` in the filter definition. An example search filter that matches the input username is `(member={0})`. An alternative that matches the authenticated userDN is `(member={1})`.

**NOTE**

If you omit the `roleFilter` attribute, the role search will use the UserDN as the DN to obtain the `roleAttributeID` value.

roleAttributeID

defines the name of the role attribute of the context that corresponds to the name of the role. If the `roleAttributesDN` property is set to `true`, this property is the DN of the context to query for the `roleNameAttributeID` attribute. If the `roleAttributesDN` property is set to `false`, this property is the attribute name of the role name.

roleAttributesDN

defines if the role attribute contains the fully distinguished name of a role object or the role name. If `false`, the role name is taken from the value of the user's role attribute. If `true`, the role attribute represents the distinguished name of a role object. The role name is taken from the value of the `roleNameAttributeID` attribute of the corresponding object. In certain directory schemas (for example, Microsoft Active Directory), role (group) attributes in the user object are stored as DNs to role objects and not as simple names. In such case, set this property to `true`. The default value of this property is `false`.

roleNameAttributeID

defines the role attribute of the context which corresponds to the name of the role. If the `roleAttributesDN` property is set to `true`, this property is used to find the name attribute of the role object. If the `roleAttributesDN` property is set to `false`, this property is ignored.

recurseRoles

defines if the recursive role search is enabled. The login module tracks already added roles to handle cyclic references.

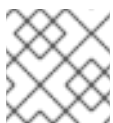
searchScope

allows to limit the search scope to one of the following (the default value is `SUBTREE_SCOPE`):

- `OBJECT_SCOPE` - searches the named roles context only.
- `ONELEVEL_SCOPE` - searches directly in the named roles context.
- `SUBTREE_SCOPE` - searches only the object if the role context is not a `DirContext?`. If the roles context is a `DirContext?`, the subtree rooted at the named object and the named object itself are searched.

searchTimeLimit

defines the timeout for the user/role searches in milliseconds (defaults to 10000, that is 10 seconds).

**NOTE**

Both searches use the same `searchTimeLimit` setting.

2.4.3. Examples of LDAP Configuration with the SPNEGO Module

The following configurations of the SPNEGO authentication uses the SPNEGOLoginModule and LDAP login module. This chained configuration is identical for FreeIPA and Active Directory with the exception of the `baseFilter` value, which defines the name to search for in LDAP identified by the SPNEGOLoginModule (for the relevant Idiff dump refer to [Section A.2.1, “Full LDAP Authentication for Active Directory”](#) and [Section A.2.2, “Full LDAP Authentication for Free IPA”](#)).

Note that the `password-stacking` property is set to `useFirstPass` on both login modules to allow the SPNEGOLoginModule to pass the name of the authenticated user to the AdvancedLdapLoginModule.

2.4.3.1. Chained Configuration on FreeIPA

The following configuration shows the AdvancedLdapLoginModule chained after the SPNEGOLoginModule for FreeIPA:

```
<application-policy name="SPNEGO_FREEIPA">
  <authentication>
    <login-module
      code="org.jboss.security.negotiation.spnego.SPNEGOLoginModule"
      flag="requisite">
      <module-option name="password-stacking">useFirstPass</module-
option>
      <module-option name="serverSecurityDomain">host</module-option>
    </login-module>

    <login-module
      code="org.jboss.security.negotiation.spnego.AdvancedLdapLoginModule"
      flag="required">
      <module-option name="password-stacking">useFirstPass</module-option>

      <module-option name="bindAuthentication">GSSAPI</module-option>
      <module-option name="jaasSecurityDomain">host</module-option>
      <module-option
name="java.naming.provider.url">ldap://kerberos.jboss.org:389</module-
option>

      <module-option
name="baseCtxDN">cn=users,cn=accounts,dc=jboss,dc=org</module-option>
      <module-option name="baseFilter">(krbPrincipalName={0})</module-
option>

      <module-option name="roleAttributeID">memberOf</module-option>
      <module-option name="roleAttributeIsDN">>true</module-option>
      <module-option name="roleNameAttributeID">cn</module-option>

      <module-option name="recurseRoles">>true</module-option>
    </login-module>
  </authentication>
</application-policy>
```

2.4.3.2. Chained Configuration on Active Directory

The following configuration shows the AdvancedLdapLoginModule chained after the SPNEGOLoginModule for Active Directory:

```
<application-policy name="SPNEGO_AD">
  <authentication>
    <login-module
      code="org.jboss.security.negotiation.spnego.SPNEGOLoginModule"
      flag="requisite">
      <module-option name="password-stacking">useFirstPass</module-option>
      <module-option name="serverSecurityDomain">host</module-option>
    </login-module>

    <login-module
      code="org.jboss.security.negotiation.spnego.AdvancedLdapLoginModule"
      flag="required">
      <module-option name="password-stacking">useFirstPass</module-option>

      <module-option name="bindAuthentication">GSSAPI</module-option>
      <module-option name="jaasSecurityDomain">host</module-option>
      <module-option
name="java.naming.provider.url">ldap://VM104:3268</module-option>

      <module-option
name="baseCtxDN">CN=Users,DC=vm104,DC=gsslab,DC=rdu,DC=redhat,DC=com</modu
le-option>
      <module-option name="baseFilter">(userPrincipalName={0})</module-
option>

      <module-option name="roleAttributeID">memberOf</module-option>
      <module-option name="roleAttributeIsDN">>true</module-option>
      <module-option name="roleNameAttributeID">cn</module-option>

      <module-option name="recurseRoles">>true</module-option>
    </login-module>
  </authentication>
</application-policy>
```

CHAPTER 3. TRACE LOGGING

To enable logging for JBoss Security and so also for the authenticator of JBoss Negotiation, do the following:

1. Open the `$JBOSS_HOME/server/$PROFILE/conf/jboss-log4j.xml`
2. Add the following to enable full TRACE logging for `org.jboss.security`:

```
<category name="org.jboss.security">
  <priority value="TRACE"/>
</category>
```

3. Optionally allow additional logging for the `com.sun.security.auth.module.Krb5LoginModule` login module. To do so, set the `debug` option to `true`:

```
<module-option name="debug">true</module-option>
```

4. Set the system property `-Dsun.security.krb5.debug=true` to get verbose output of the entire GSSAPI negotiation process.

3.1. CONFIGURING MESSAGE TRACING

You can log the exchanged messages selectively at TRACE level. Both, the Request and Response messages, can be logged and that either as Hex or as Base64 or both.

The base category for message tracing is `org.jboss.security.negotiation.MessageTrace`. If you enable TRACE logging for this category, all request and response messages are logged at the TRACE level in both Hex and in Base64 encoding.

Example 3.1. Configuration for tracking all messages

```
<category name="org.jboss.security.negotiation.MessageTrace">
  <priority value="TRACE"/>
</category>
```

To reduce the logging to either just request or just response messages, append `.Request` or `.Response` to the category value.

Example 3.2. Configuration for tracking only request messages (messages are logged in both Hex and Base64)

```
<category name="org.jboss.security.negotiation.MessageTrace.Request">
  <priority value="TRACE"/>
</category>
```

Example 3.3. Configuration for tracking only response messages (messages are logged in both Hex and Base 64)

-

```
<category name="org.jboss.security.negotiation.MessageTrace.Response">
  <priority value="TRACE"/>
</category>
```

To have messages logged in a particular encoding, append `.Hex` or `.Base64` to the category value.

Example 3.4. Message tracking with defined encoding

```
<category
name="org.jboss.security.negotiation.MessageTrace.Request.Hex">
  <priority value="TRACE"/>
</category>

<category
name="org.jboss.security.negotiation.MessageTrace.Request.Base64">
  <priority value="TRACE"/>
</category>

<category
name="org.jboss.security.negotiation.MessageTrace.Response.Hex">
  <priority value="TRACE"/>
</category>

<category
name="org.jboss.security.negotiation.MessageTrace.Response.Base64">
  <priority value="TRACE"/>
</category>
```

CHAPTER 4. PASSING AUTHENTICATION PROPERTIES TO THE SERVER

After you have set up JBoss Negotiation, you need to make sure to pass the Kerberos realm properties to JBoss Application Server:

java.security.krb5.realm

the Kerberos realm the server authenticates against

java.security.krb5.kdc

KDC hostname



NOTE

Skip this step if you are running your JBoss installation on a host which is already configured to authenticate against a KDC.

For further information about the properties, refer to [Java Generic Security Services \(Java GSS\) and Kerberos](#).

You can pass the properties to the server either from the command line or add them to the server properties.

4.1. PASSING THE PROPERTIES FROM THE COMMAND LINE

To send the properties to the server from the command line, substitute *KERBEROS.JBOSS.ORG* with your realm and issue the `run` command with the respective Java properties:

- On Red Hat Enterprise Linux, run the following command:

```
./run.sh -Djava.security.krb5.realm=KERBEROS.JBOSS.ORG -
Djava.security.krb5.kdc=kerberos.security.jboss.org
```

- On Windows, run the following command:

```
run.bat Djava.security.krb5.realm=KERBEROS.JBOSS.ORG -
Djava.security.krb5.kdc=kerberos.security.jboss.org
```

These properties are valid only until the server shutdown and you need to pass them to the server on every start.

4.2. ADDING THE PROPERTIES TO THE SYSTEM PROPERTIES

To make the properties permanent and have an application server start always with the SPNEGO mechanism, define the properties in the `$JBOSS_HOME/server/$PROFILE/deploy/properties-service.xml` descriptor. Make sure the properties are loaded before the first authentication attempt (JBoss does not allow any incoming HTTP connections before the server has started up fully).

Open the descriptor and add the following attribute to the `jboss:type=Service, name=SystemProperties` MBean:

```
<attribute name="Properties">  
  java.security.krb5.kdc=kerberos.security.jboss.org  
  java.security.krb5.realm=KERBEROS.JBOSS.ORG  
</attribute>
```

4.2.1. Multiple KDCs

If you are using one or more slave KDCs in addition to your master KDC, list the KDCs in a colon-separated list after the `java.security.krb5.kdc` system property. The system will use the provided alternative KDC if the master KDC is not available.

Example 4.1. Running a server with multiple KDCs

```
./run.sh -  
Djava.security.krb5.realm=KERBEROS.JBOSS.ORG:SLAVE_KDC.JBOSS.ORG -  
Djava.security.krb5.kdc=kerberos.security.jboss.org
```

CHAPTER 5. CONFIGURING MICROSOFT ACTIVE DIRECTORY

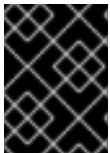


IMPORTANT

The domain details in this chapter differ from the domain details used in the rest of this guide.

To configure Active Directory to authenticate user through JBoss Negotiation you need to do the following:

- Create a server user account and configure it as a Service Principal Name (SPN) account: the user of the Service Principal Name account (SPN account) acts as a connection between the Kerberos server, the Active Directory and the JBoss web server.
- Generate a keytab file for the server user and export it to the application server. The application server uses the keytab to authenticate to KDC in AD.



IMPORTANT

Make sure you are using an Active Directory domain controller. It is not possible to use a Windows machine with accounts managed locally.



WARNING

Instructions in this guide apply to Windows 2003 and may differ from the instructions relevant for your Windows operating system.

5.1. USER ACCOUNT FOR THE APPLICATION SERVER

To configure an SPN account for the application server on the AD domain controller, you need **Setspn** and **Ktpass**. The command line utilities are part of Windows Server 2003 Support Tools and serve for mapping the server user name to the application server and its HTTP service.

The utilities are available on [Microsoft web pages](#).

You need to create a regular user account for the server in the AD domain (make sure it is a user account, not a computer account) and map the account to the service account.

5.1.1. Creating Server User

To create a new user for the server, do the following:

1. Go to **Start** → **Administrative Tools** → **Active Directory Users and Computers**
2. In the **Active Directory Users and Computers** window, go to **Action** → **New** → **User**

New Object - User

Create in: vm104.gsslab.rdu.redhat.com/Users

First name: testserver Initials:

Last name:

Full name: testserver

User logon name: testserver @vm104.gsslab.rdu.redhat.com

User logon name (pre-Windows 2000): VM_104\ testserver

< Back Next > Cancel

Figure 5.1. New User

- In the **New User** window, enter the user details and click **Next**. [Figure 5.1, “New User”](#) uses the server `@vm104.gsslab.rdu.redhat.com` and defines a user called `testserver`.
- Enter the password for the user and select the **User cannot change password** and **Password never expires**.



IMPORTANT

Make sure you have entered a valid password as changing the password later can invalidate the keytab file and break your JBoss installations.

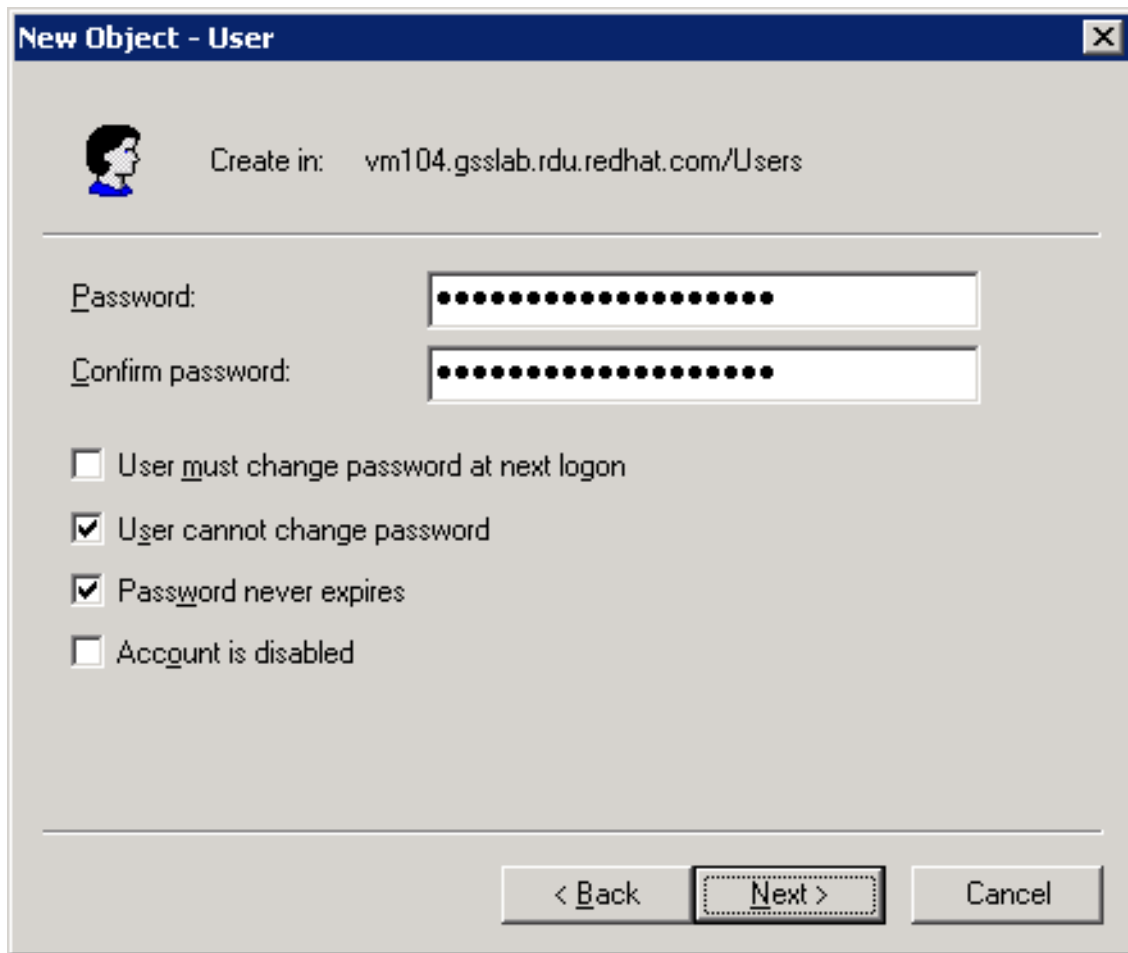


Figure 5.2. New User Password

5. Click **Next** and **Finish**.

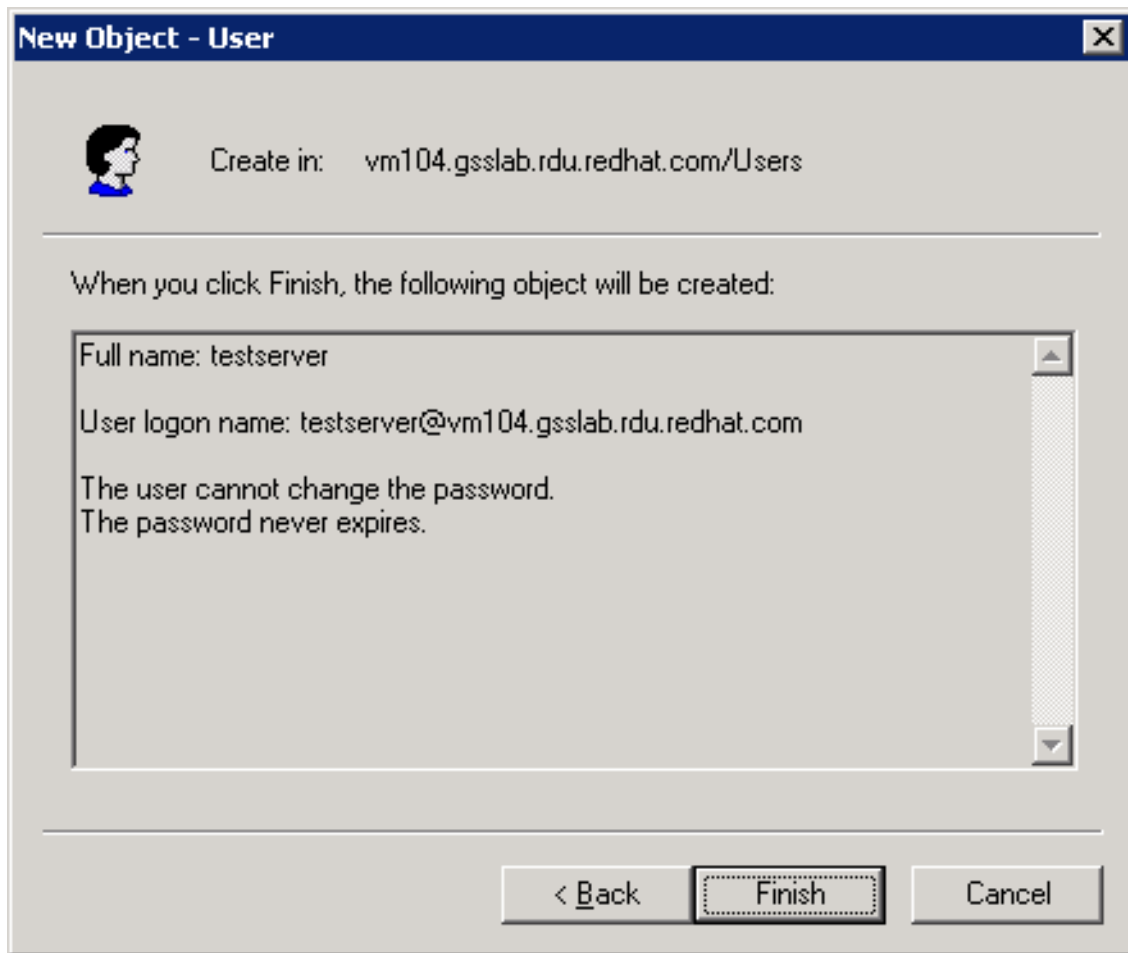


Figure 5.3. New User Finish

6. In the **Active Directory Users and Computers** window, right-click the user and click **Properties**.
7. In the user properties window, click the **Account** tab and make sure the **Do not require Kerberos preauthentication** and **Use DES encryption types for this account** are selected under **Account Options**.

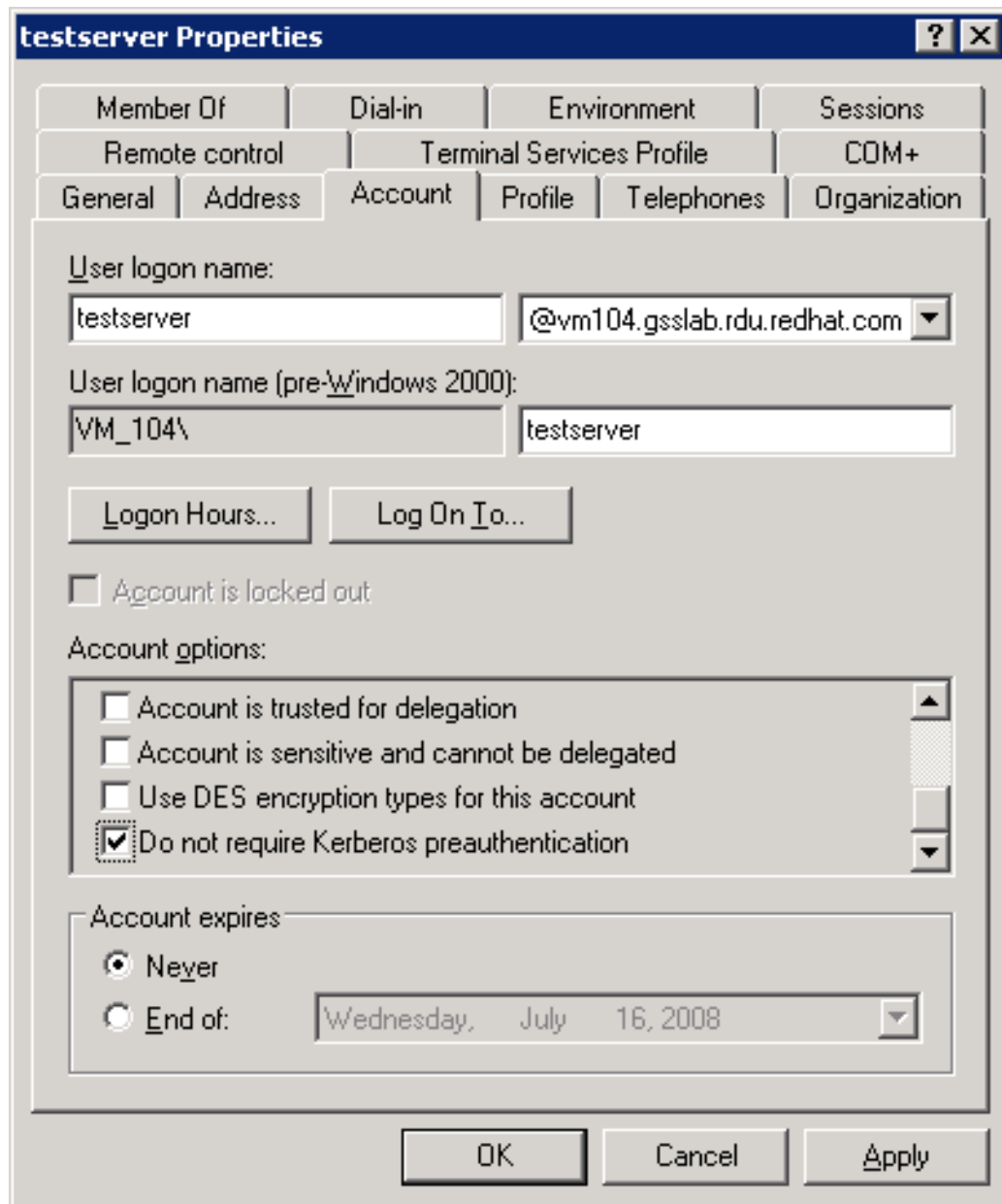


Figure 5.4. User Properties

Now you need to create and export the keytab file for the created user.

5.2. EXPORTING KEYTAB

Once you have created the user account for the application server, use the **Ktpass** utility to map the SPN account as a trusted host and export the keytab for the server:

1. Issue the `ktpass` command to map the created user as a trusted host and generate the keytab file. The `-princ` option defines the service principal that is being mapped to and the `-mapuser` option defines the user account being mapped to.

```
ktpass -princ <service principal mapping> -out <target keytab file>
-pass * -mapuser <user mapping>
```

Example 5.1. `ktpass` command

```
ktpass -princ host/testserver@kerberos.jboss.org -out  
C:\testeserver.host.keytab -pass * -mapuser KERBEROS\testserver
```

2. When prompted, enter the user password.
3. Issue the following command to display the available mappings and check if the new mapping is enlisted:

```
setspn.exe -l <user mapping>
```

Example 5.2. setspn command

```
setspn.exe -l testserver
```

CHAPTER 6. CONFIGURING FREEIPA

Before you configure FreeIPA to use JBoss Negotiation, make sure you have FreeIPA installed and configured correctly, and the clients are able to obtain Kerberos tickets. Detailed FreeIPA documentation is available on <http://www.freeipa.org/>.

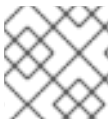


WARNING

Due to the supported encryption types of FreeIPA, the JBoss application server must run on a Java 6 JVM with unlimited cryptography enabled.

To configure FreeIPA to authenticate users through JBoss Negotiation you need to do the following:

- Create a service principal for the server and add the HTTP service to it. The server user acts as a connection between FreeIPA and the JBoss web server.
- Generate a keytab file for the server user and export it to the application server. The application server uses the keytab to authenticate to KDC in FreeIPA.



NOTE

These instructions apply to FreeIPA 1.1.

6.1. CREATING SERVICE PRINCIPAL

You need to create the service principal, which represents the HTTP service of your JBoss Application Server to allow the clients to request the ticket for this service.



NOTE

Full information on service principal creation is available on http://freeipa.org/page/AdministratorsGuide#Managing_Service_Principals.

1. The simplest way to create a service principal is through the FreeIPA WebUI: access the tool as an administrator.
2. Click the **Add Service Principal** link.

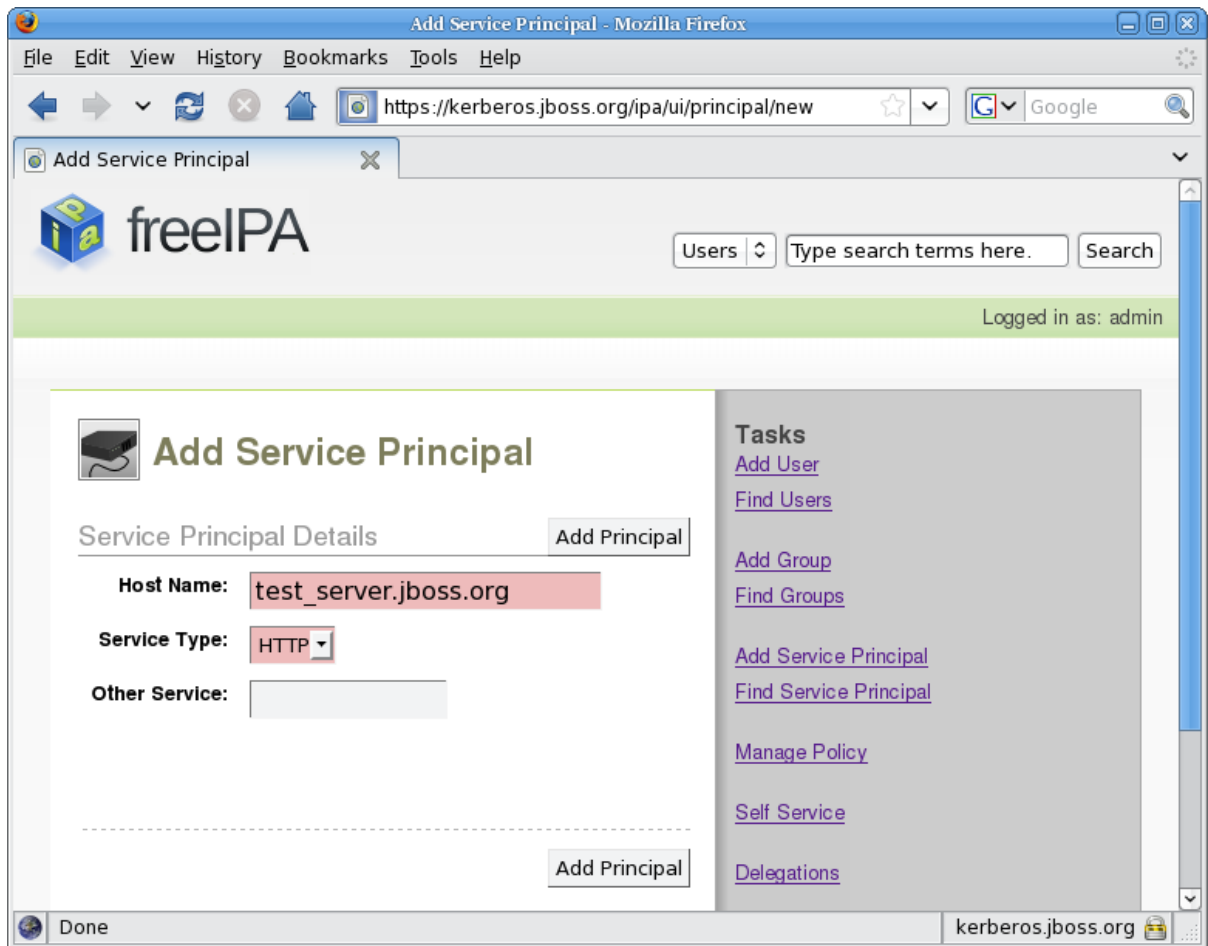


Figure 6.1. Adding Service Principal

3. Set the hostname to the host name of your server (`test_server.jboss.org`) and the service type to HTTP, and click **Add Principal**.

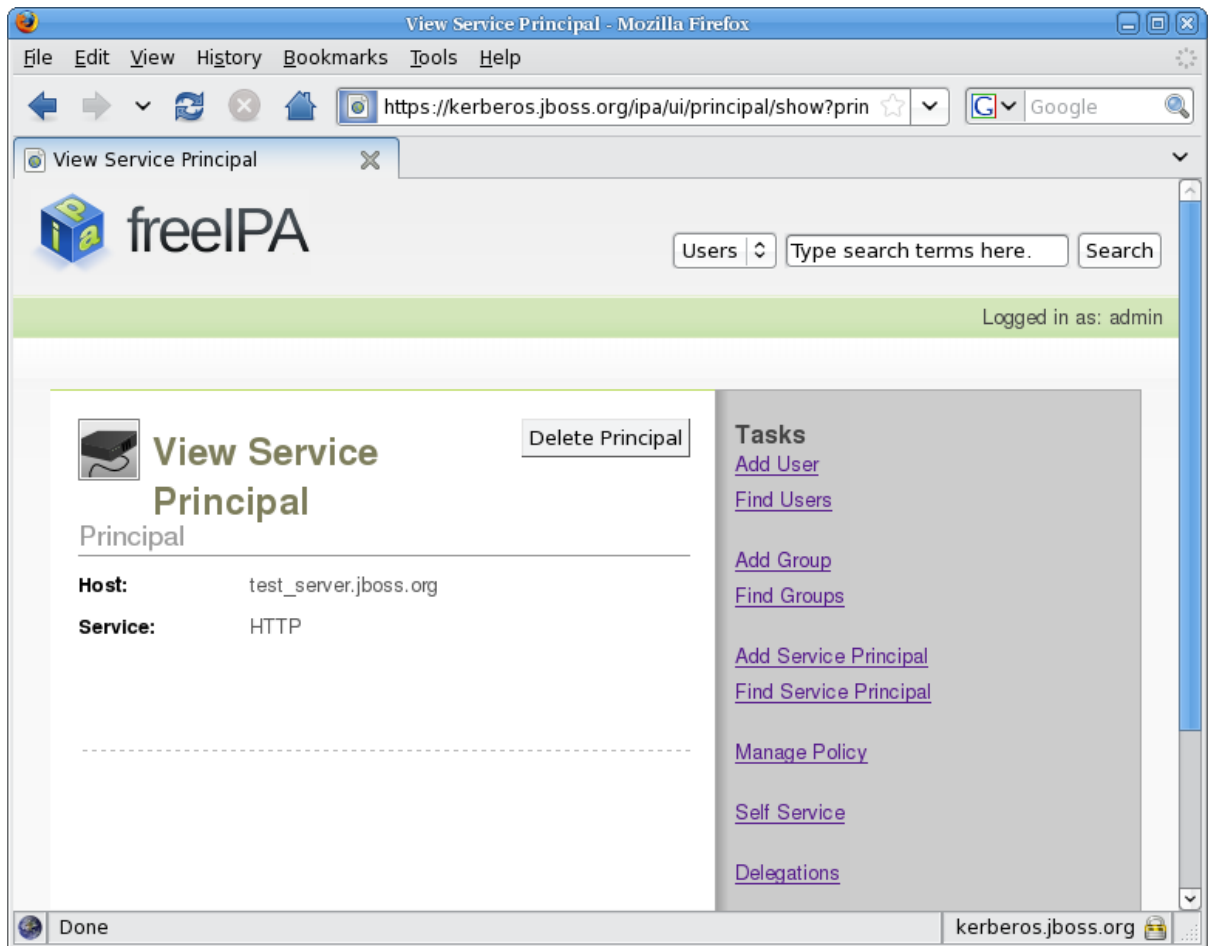


Figure 6.2. View Service Principal



NOTE

Creating the service principal requires the host name to be mapped with DNS. If this procedure fails, on the command line, issue the following command to create the principal: `ipa-addservice HTTP/test_server.jboss.org@JBASS.ORG --force`

6.2. EXPORTING KEYTAB



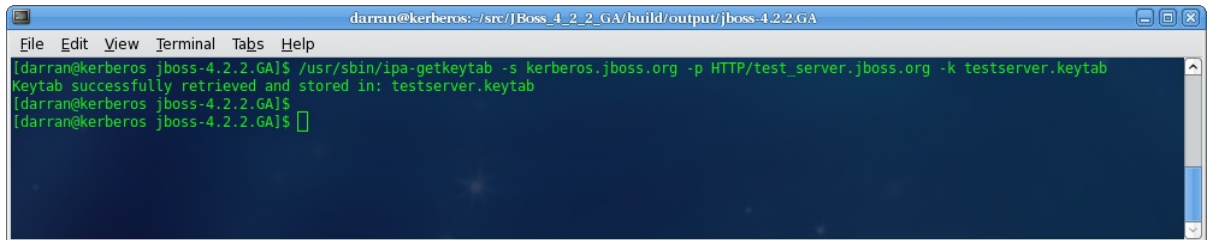
WARNING

Exporting a new keytab resets the secret associated with the service principal and invalidates any keytabs created previously for the principal.

To export a keytab for the server, do the following:

1. Obtain the Kerberos ticket-granting ticket for an administrator: issue the command `kinit <admin>`.
2. To obtain the keytab, issue the command `ipa-getkeytab` with the options:

- s**
FreeIPA server to obtain the keytab from
- p**
Non-realm part of the full principal name
- k**
File to append the keytab

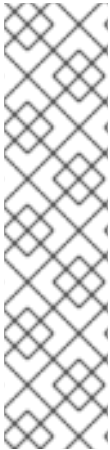
A terminal window titled 'darran@kerberos:~/src/JBoss_4.2.2_GA/build/output/jboss-4.2.2.GA' with a menu bar (File, Edit, View, Terminal, Tabs, Help). The terminal shows the command `/usr/sbin/ipa-getkeytab -s kerberos.jboss.org -p HTTP/test_server.jboss.org -k testserver.keytab` being executed. The output is 'Keytab successfully retrieved and stored in: testserver.keytab'. The prompt changes from `[darran@kerberos jboss-4.2.2.GA]` to `[darran@kerberos jboss-4.2.2.GA]$` after the command is run.

```
darran@kerberos jboss-4.2.2.GA] /usr/sbin/ipa-getkeytab -s kerberos.jboss.org -p HTTP/test_server.jboss.org -k testserver.keytab
Keytab successfully retrieved and stored in: testserver.keytab
[darran@kerberos jboss-4.2.2.GA]$
[darran@kerberos jboss-4.2.2.GA]$
```

Figure 6.3. Get Keytab

Once you have set up the service principal and exported the keytab, make sure your server security domain uses the output keytab file (refer to [Section 2.2, “Defining Server Security Domain”](#)) and configure the second login module to allow the client to load the roles assigned to the authenticated user (refer to [Section 2.4, “Role Mapping”](#)).

CHAPTER 7. CONFIGURING WEB BROWSERS



NOTE

Web browsers need to trust the application server they communicate with. To add the application server to trusted resources, add the IP address of the JBoss application server to trusted hosts: on Red Hat Enterprise Linux, edit the `/etc/hosts` file and make sure the file is used for host name lookups; on Windows edit `C:\windows\system32\drivers\etc\hosts`). You can make this change either on your DNS server or locally on the client machine.

If the Kerberos realm is `KERBEROS.JBOSS.ORG` and the server hosting JBoss is `testserver` then the IP address you need to add as a trusted host is `testserver.kerberos.jboss.org`.

7.1. CONFIGURING INTERNET EXPLORER

The instructions on how to enable JBoss Negotiation in Internet Explorer (IE) apply to Internet Explorer 6 on Microsoft Windows 2003.

By default Internet Explorer only performs SPNEGO authentication against sites in the `Local intranet` zone. To enable the SPNEGO negotiation, add the server URL to the Local intranet sites:

1. On the **Tools** menu, click **Internet Options**.

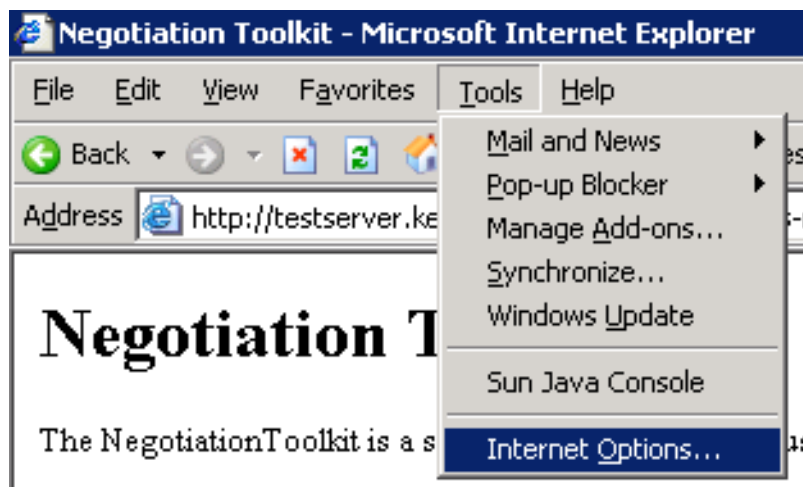


Figure 7.1. Tools - Internet Options

2. In the **Internet Options** dialog, click the **Security** tab label.
3. In the **Security** tab, make sure the `Local intranet` icon is selected and click the **Sites** button.

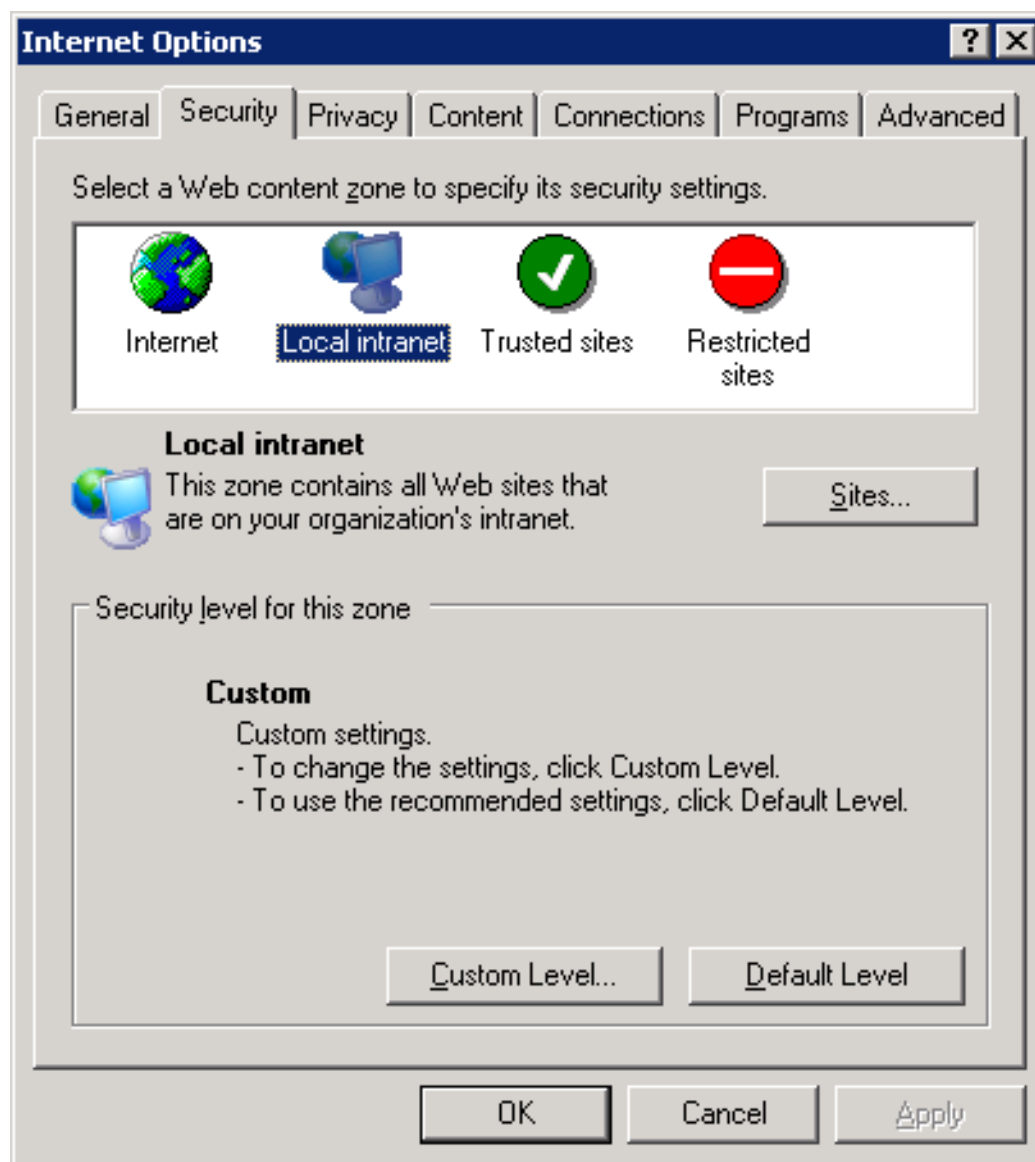


Figure 7.2. Internet Options

4. In the **Local intranet** dialog, enter the URL of the server with the JBoss installation and click **Add**.

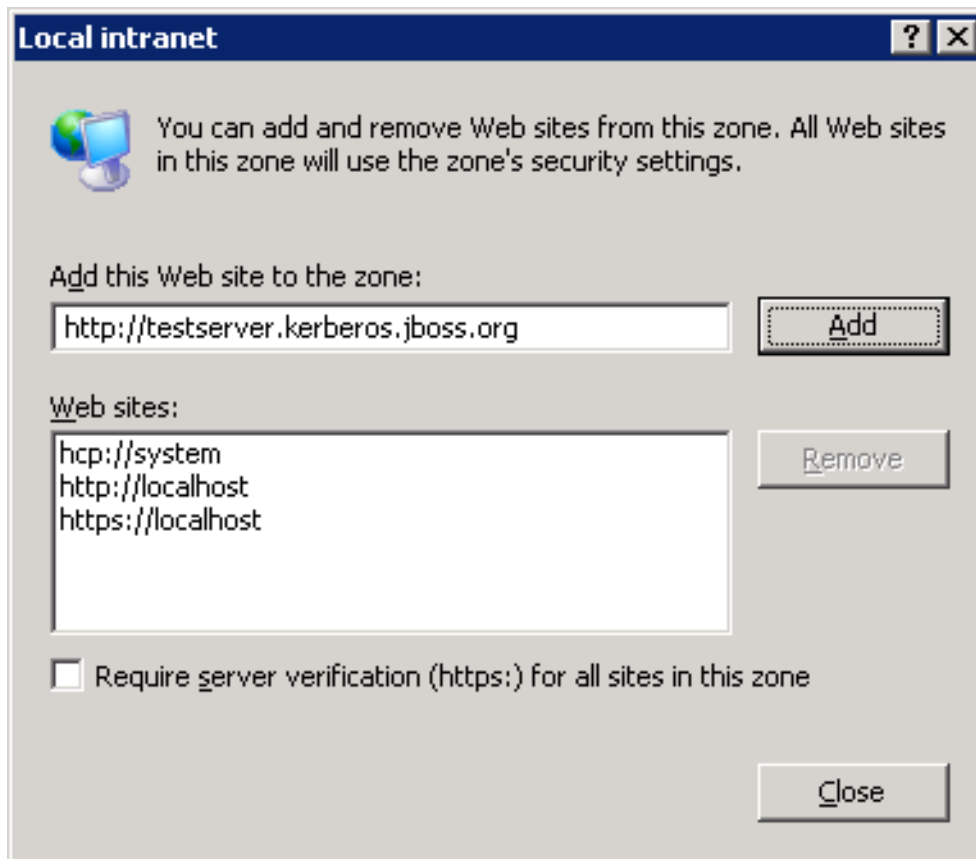


Figure 7.3. Local Intranet

The server appears in the **Web sites** list below. Internet Explorer now trusts the JBoss installation and performs the SPNEGO negotiation. Make sure to test the Negotiation with the **Basic Negotiation** servlet (refer to [Section 8.2, “Basic Negotiation”](#)).

7.2. CONFIGURING FIREFOX

The instructions on how to enable SPNEGO negotiation in Mozilla Firefox apply to Mozilla Firefox 2.0.0.11 on Microsoft Windows 2003 and to Firefox 3.0.1 on Fedora 9.

To enable the SPNEGO negotiation, change the Mozilla Firefox configuration as follows:

1. Navigate to the [about:config](#) URL with the configuration options for Firefox.
2. Set the filter to `network.negotiate` to display the relevant options.

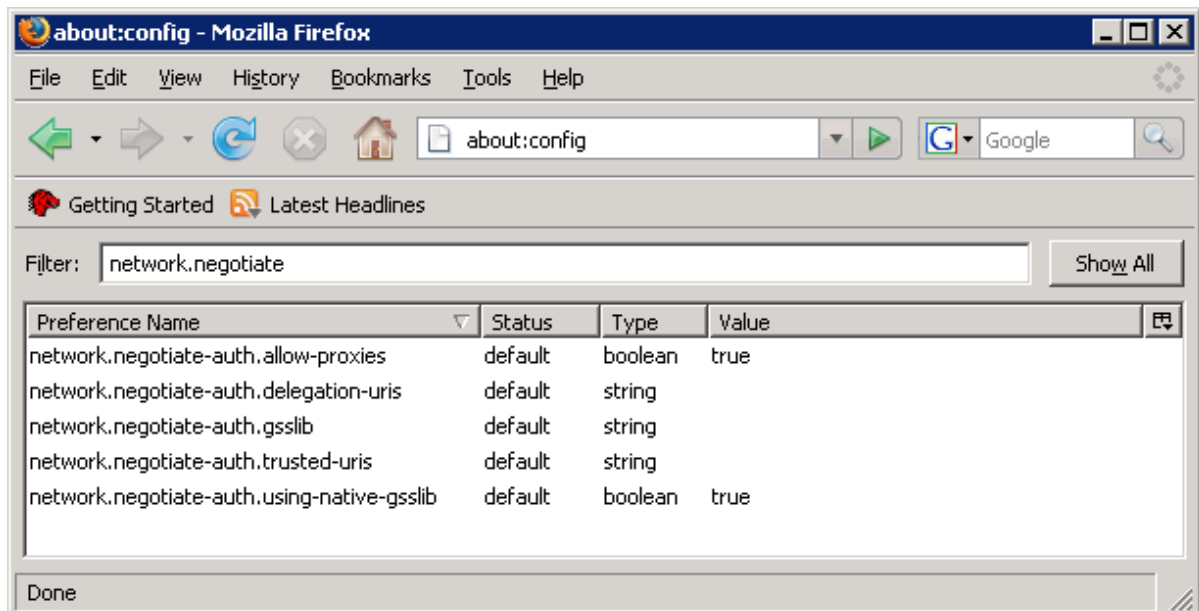


Figure 7.4. Firefox Configuration

3. Double-click the `network.negotiate-auth.delegation-uris` and in the **Enter string value** dialog, enter the URI for SPNEGO negotiation. The URI can be entered as a partial URI, for example `http://` or `testserver` or a full URI, for example `http://testserver.jboss.org`.

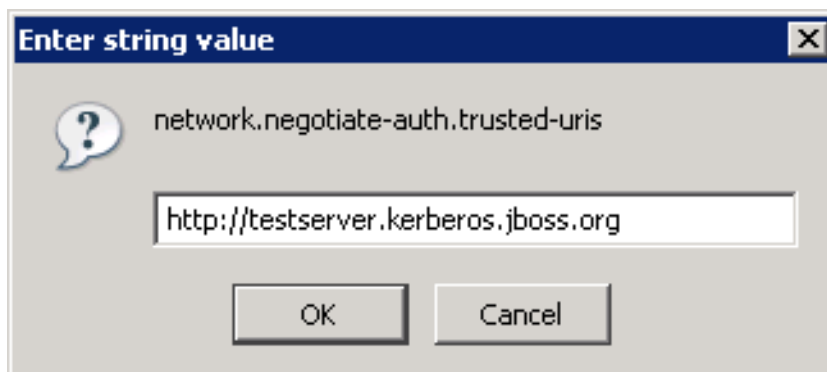


Figure 7.5. Firefox Configuration



IMPORTANT

The `network.negotiate-auth.delegation-uris` option specifies the URI the users credentials will be delegated to. In the JBoss Negotiation version, delegation is not yet supported.

The URI appears in the **Value** column. Firefox now trusts the JBoss installation and performs the SPNEGO negotiation. Make sure to test the Negotiation with the **Basic Negotiation** servlet (refer to [Section 8.2, “Basic Negotiation”](#)).

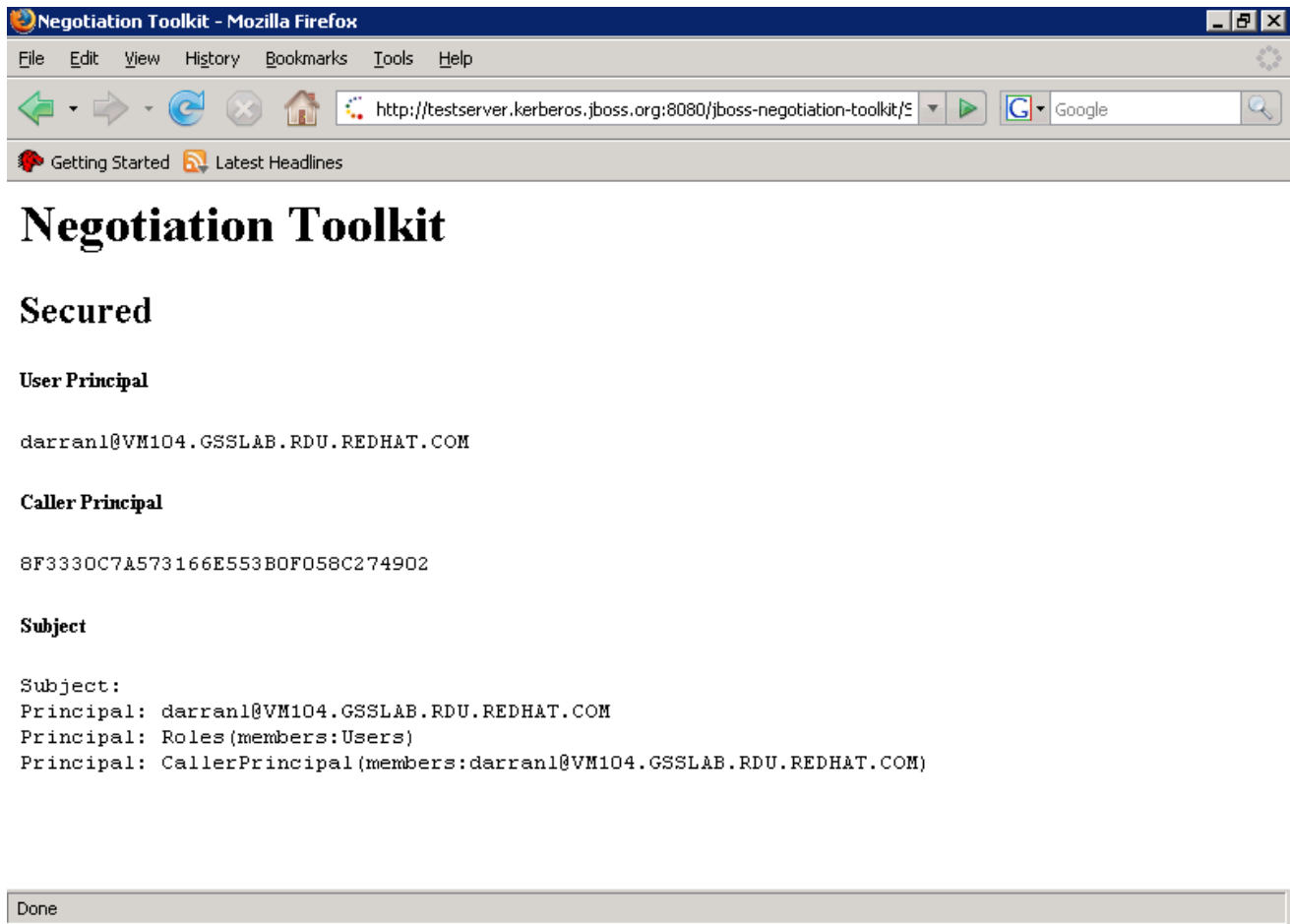


Figure 7.6. Firefox Negotiation Toolkit

CHAPTER 8. NEGOTIATION TOOLKIT

The Negotiation Toolkit is a web application for testing of the SPNEGO configuration so you do not need to test your configuration with your application. The `jboss-negotiation-toolkit.war` file is available at <https://repository.jboss.org/nexus/content/groups/public/org/jboss/security/jboss-negotiation-toolkit/2.0.3.SP1/jboss-negotiation-toolkit-2.0.3.SP1.war>. Copy the file to the `$JBOSSHOME/server/$PROFILE/deploy` directory to have the Negotiation Toolkit deployed.

The toolkit assumes that the authenticator has the name `SPNEGO` and that the application security domain is named `SPNEGO`. If either of these have other names, deploy the web application as an exploded archive and modify `web.xml` and `jboss-web.xml`:

- In the `WEB-INF/web.xml` file, update the authenticator key in `auth-method` (`<auth-method>SPNEGO</auth-method>`).
- In the `WEB-INF/jboss-web.xml` file, update the name of the security domain in `security-domain` (`<security-domain>SPNEGO</security-domain>`).

Once deployed, access the Negotiation Toolkit web application at <http://testserver.kerberos.jboss.org:8080/jboss-negotiation-toolkit>



NOTE

Make sure you have set the DNS entry as described in [Prerequisite: DNS Setting](#) in [Section 2.3, “Defining Application Security Domain”](#).

8.1. FRONT PAGE

The main page for the Negotiation Toolkit contains links to the toolkit utilities, which test the mechanisms of SPNEGO authentication. It is recommended that you follow the links from top to bottom.

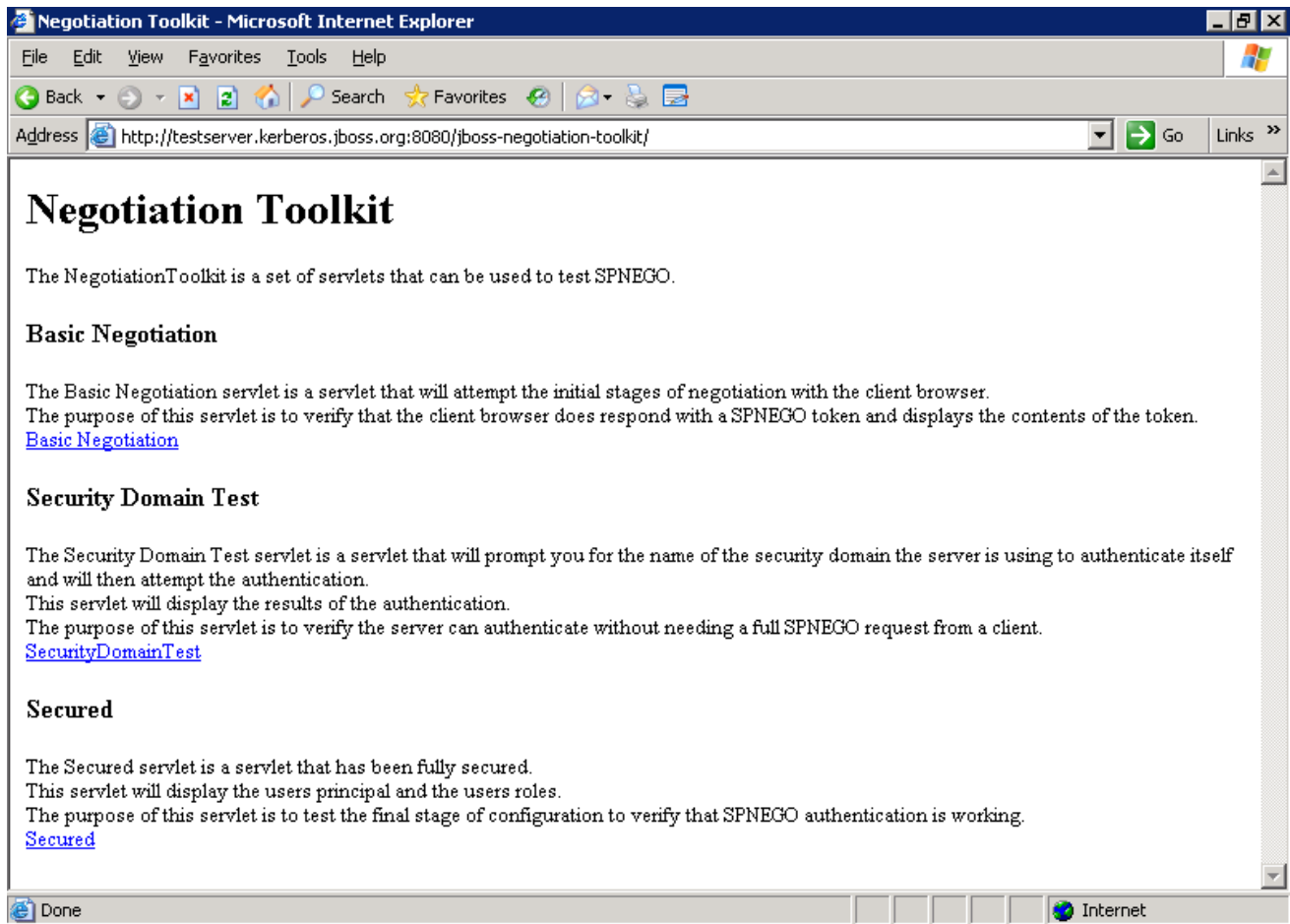
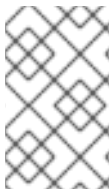


Figure 8.1. Negotiation Toolkit Front Page



NOTE

Make sure you have completed the installation before you use the Negotiation Toolkit as the toolkit tests involve communication with the application server, web browser, and the KDC.

8.2. BASIC NEGOTIATION

The **Basic Negotiation** servlet tests if the web browser trusts the application server: it prompts the web browser to negotiation and checks if the application server received a negotiation token.

If the web browser failed to send a negotiation token, the servlet displays a web page similar to [Figure 8.2, “Basic Negotiation Failure”](#)

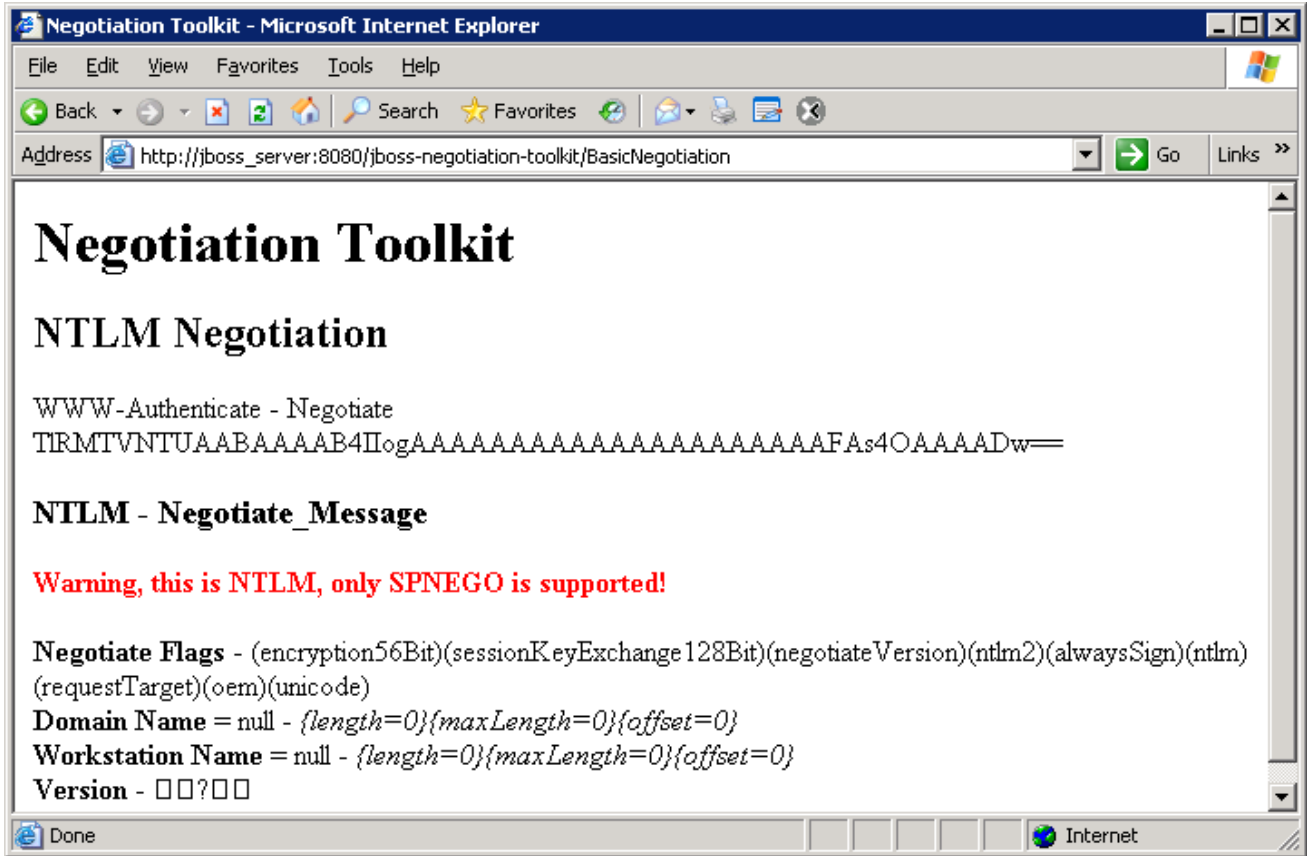


Figure 8.2. Basic Negotiation Failure

If the web browser successfully sends a negotiation token, the servlet displays a web page similar to Figure 8.3, “Basic Negotiation Success”

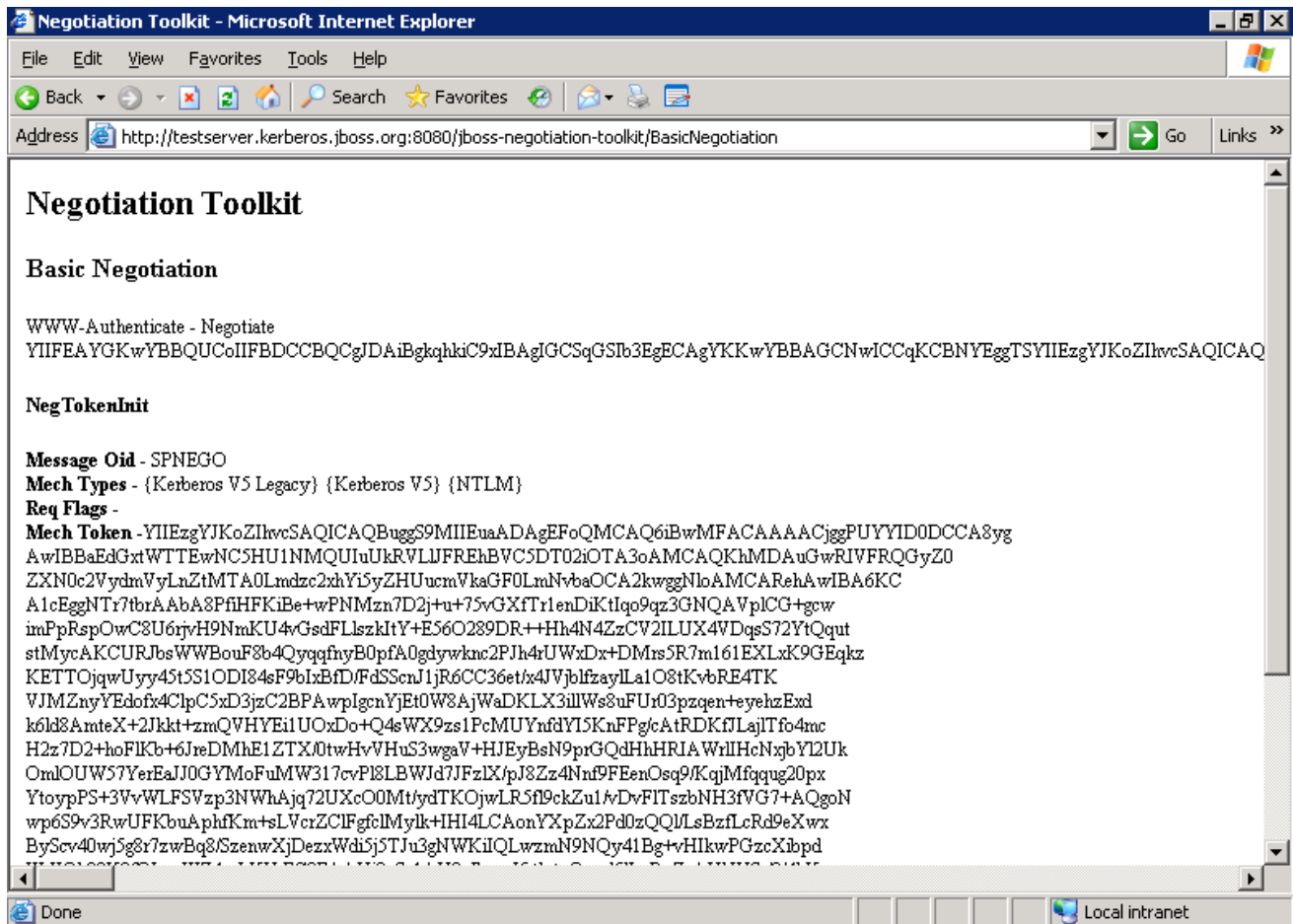


Figure 8.3. Basic Negotiation Success

The web page shows some of the information contained within the negotiation token.

8.3. SECURITY DOMAIN TEST

The **Security Domain Test** tests if the application server can authenticate against the KDC through its security domain.

First, the servlet prompts you to enter the name of the security domain (we have been using the domain `host` throughout this guide; the page is shown in [Figure 8.4, “Security Domain Test”](#)).

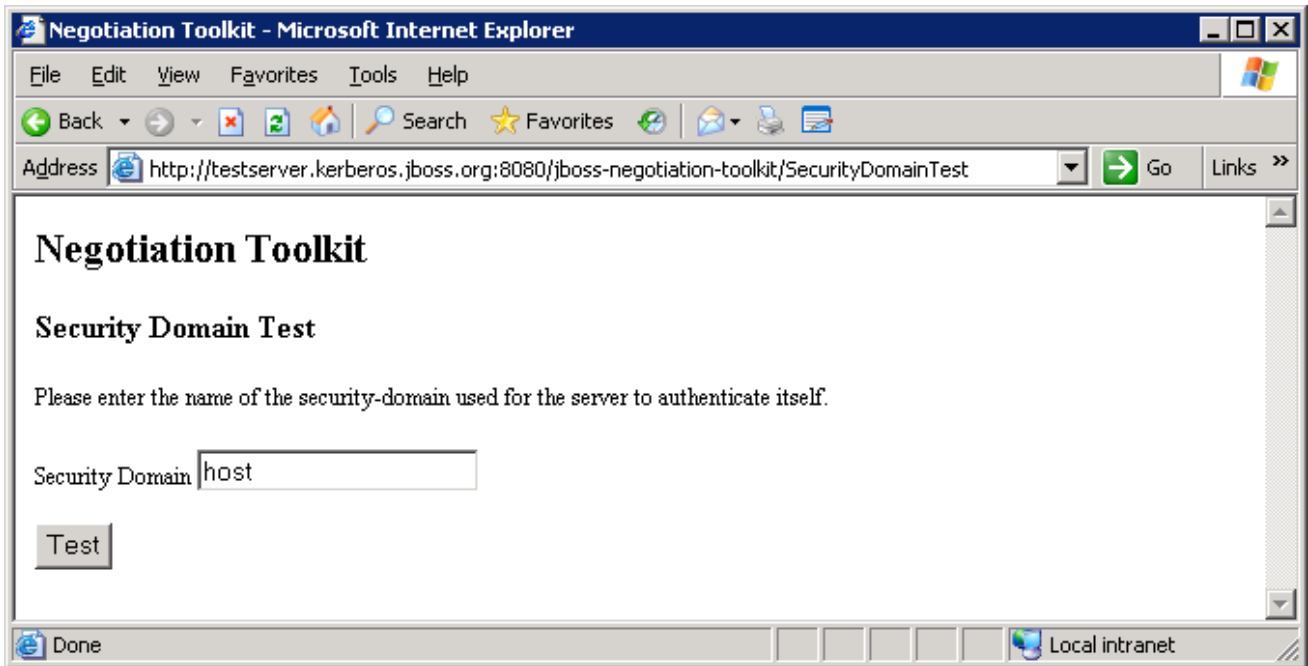


Figure 8.4. Security Domain Test

If the servlet establishes the authentication successfully, it displays a page similar to [Figure 8.5](#), "Security Domain Test - Authenticated".

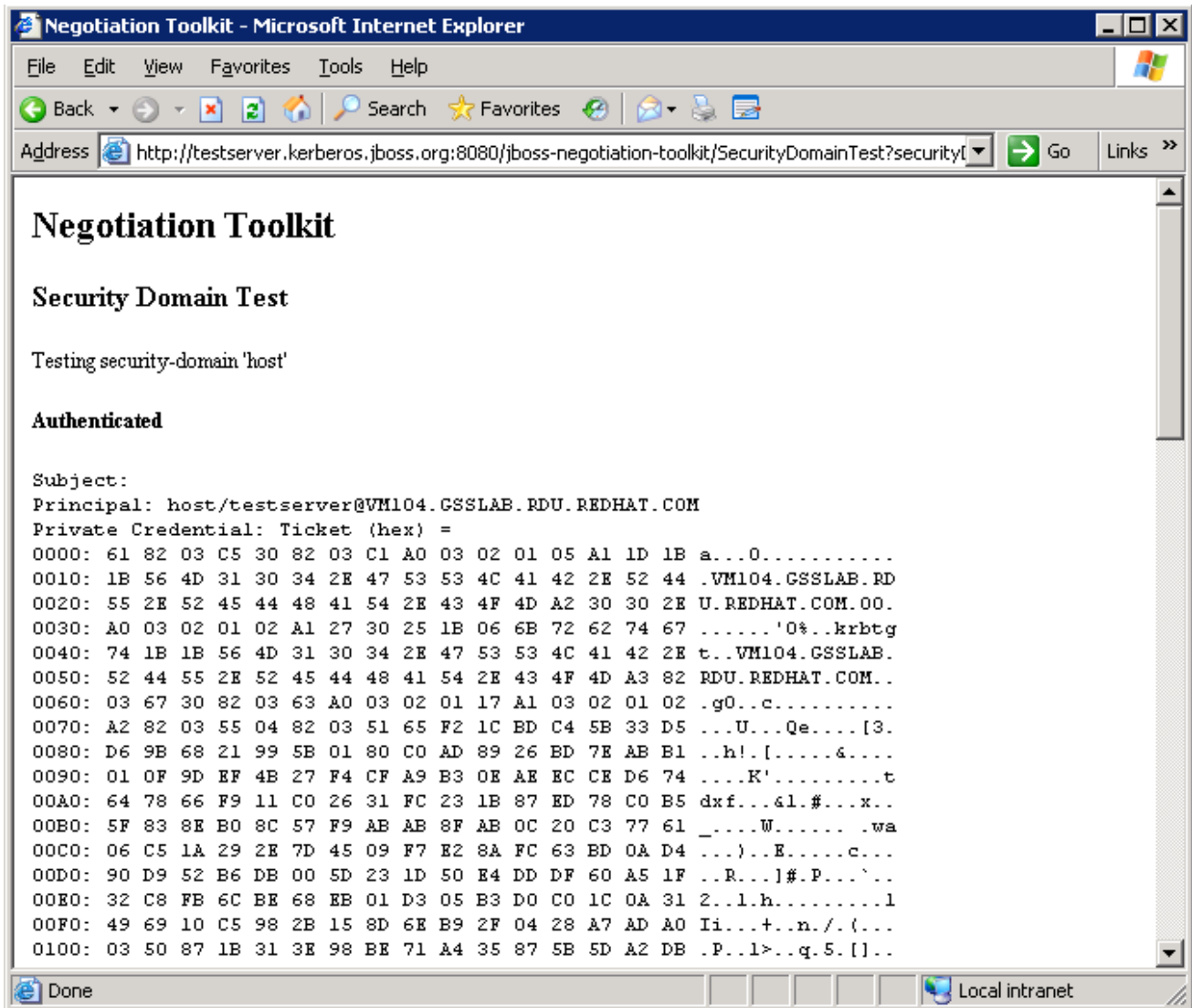


Figure 8.5. Security Domain Test - Authenticated

8.4. SECURED

The **Secured** servlet is configured to require full SPNEGO authentication. If the servlet returns a page similar to [Figure 8.6, “Secured”](#), its run was successful and your SPNEGO authentication is configured correctly.

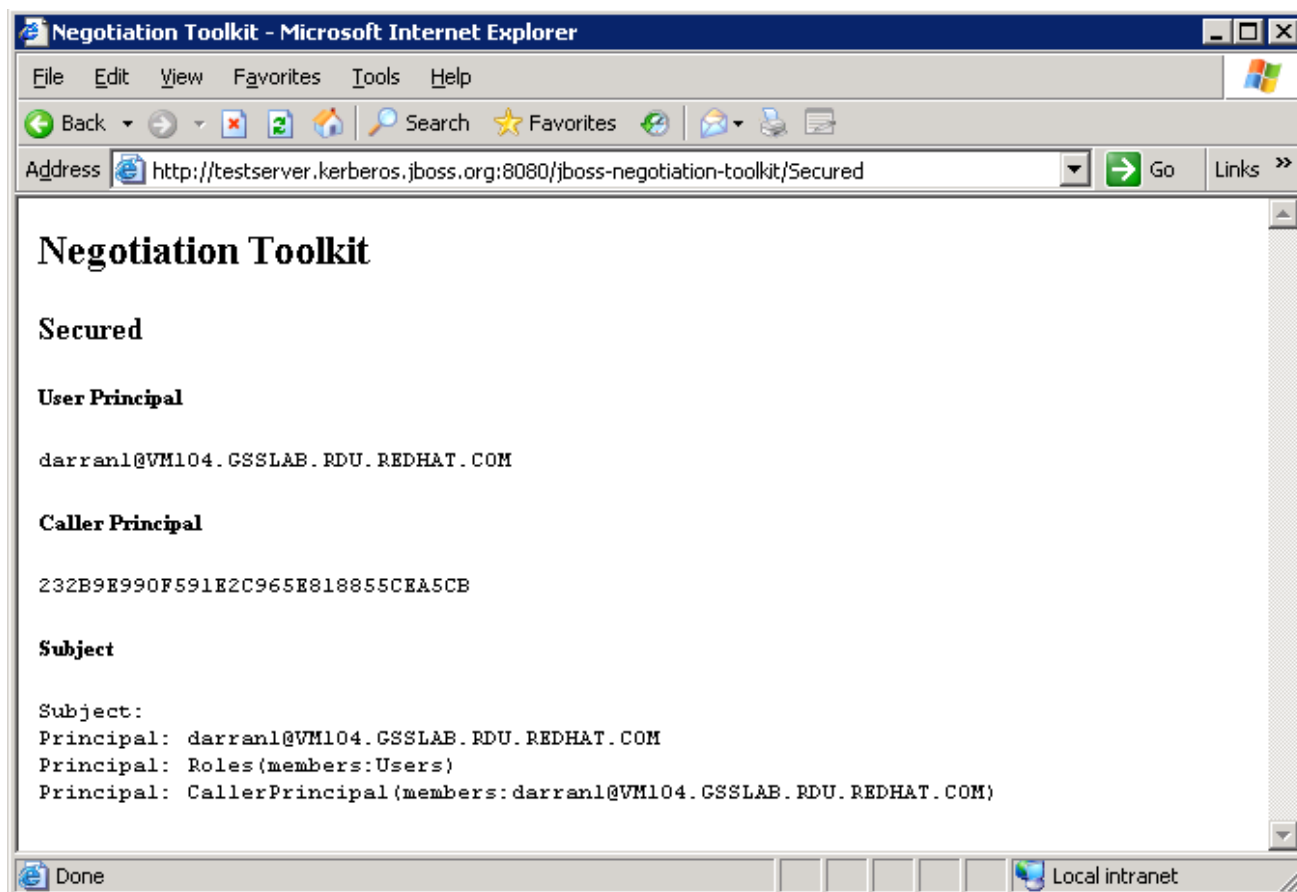


Figure 8.6. Secured

CHAPTER 9. CONFIGURING WEB APPLICATIONS

Once you have configured JBoss Negotiation on your server and the connection to FreeIPA or Active Directory, you need to configure your web application to use JBoss Negotiation authentication.

To configure your web application to use JBoss Negotiation authentication, do the following:

1. Add the SPNEGO security domain to the **WEB-INF/jboss-web.xml** file:

```
<jboss-web>
  <security-domain>java:/jaas/SPNEGO</security-domain>
</jboss-web>
```

2. Configure the **login-config.xml** file to use the SPNEGO authenticator:

```
<login-config>
  <auth-method>SPNEGO</auth-method>
  <realm-name>SPNEGO</realm-name>
</login-config>
```

The `auth-method` maps the key used for the authenticator.

APPENDIX A. ADVANCED LDAP LOGIN MODULE: FULL LDAP AUTHENTICATION

The JBoss Negotiation project includes the `AdvancedLdapLoginModule` to handle the LDAP role searching requirements.

The `AdvancedLdapLoginModule` is based on the `LdapExtLoginModule`; however, `AdvancedLdapLoginModule` differs in the following aspects:

- The accumulated subject roles do not include the role name of the first matching context.
- When the `roleAttributesDN` module property is set to `false`, the recursive role search is disabled even if the `recurseRoles` module option is set to `true`.

You can use the `AdvancedLdapLoginModule` module in a chained configuration with the `SPNEGOLdapLoginModule` to allow a GSSAPI authentication to allow authentication through LDAP (refer to [Section 2.4, “Role Mapping”](#)) or use the module for a full authentication through LDAP. You can also configure it to skip the user search, the authentication, or the role search if required.

A.1. CONFIGURATION

The fully qualified classname of the new login module is `org.jboss.security.negotiation.AdvancedLdapLoginModule`.



WARNING

In Beta releases the class name was `org.jboss.security.negotiation.spnego.AdvancedLdapLoginModule`. The login module is still available under this name; however, it has been deprecated and will be removed in a future release.

The `AdvancedLdapLoginModule` supports password-stacking: if you want to use the module in conjunction with other login modules, make sure the password-stacking property is set to `useFirstPass`.

A.1.1. Defining Initial LDAP Context

First, you need to define the user credentials, which are used to obtain the `InitialLdapContext` and then used to search for the user and the user roles.



NOTE

The login module supports obtaining this `InitialLdapContext` using a username and credential or using GSSAPI for a previously authenticated user. Here we use the user credentials. For configuration with GSSAPI, refer to [Section 2.4, “Role Mapping”](#).

To authenticate with a username and password the following define the following settings:

bindDN

defines the DN used to bind to the LDAP server. This is a DN with read/search permissions to the defined `baseCtxDN` and `rolesCtxDN`.

bindCredential

defines the `bindDN` password. The password can be encrypted if the `jaasSecurityDomain` is specified.

jaasSecurityDomain

defines the JMX ObjectName of the `jaasSecurityDomain`. This is the `jaasSecurityDomain` used to decrypt the `java.naming.security.principal`. The `JaasSecurityDomain#encrypt64(byte[])` method of the domain returns the encrypted form of the password. You can use also `org.jboss.security.plugins.PBEUtils` to generate the encrypted form.

A.1.2. Defining DN Search

After the module has created the LDAP initial context, it takes the provided username and searches for the user DN. To define the properties of the search, provide the following properties:

baseCtxDN

defines the fixed DN of the context to search for user roles. Consider that this is not the Distinguished Name of where the actual roles are located but the DN of where the objects containing the user roles are located (that is, for active directory, this is the DN with the user account).

baseFilter

defines the search filter used to locate the context of the user to authenticate. The input username/userDN as obtained from the login module callback substitutes the `{0}` expression. This substitution behavior comes from the standard `DirContext?.search(Name, String, Object[], SearchControls? cons)` method. An common example search filter is `"(uid={0})"`

searchTimeLimit

defines the timeout for the user and role searches in milliseconds (defaults to 10000, that is 10 seconds).

**NOTE**

To disable the user DN search omit the `baseCtxDN` property: the provided username will be used as the DN in this login module.

A.1.3. User Authentication**NOTE**

If the LDAP login module is not the first login module and a previous login module has already authenticated the user, the user authentication is skipped.

If no previous login module has authenticated the user this step takes the User DN from the User DN search and their provided credential and attempts to create a new InitialLdapContext and verify that the User DN and credential combination is valid.

For user authentication, you can define the following property:

allowEmptyPassword

If empty (length==0) passwords are passed to the LDAP server. An empty password is treated as an anonymous log in by an LDAP servers. Set the property to `false` to reject empty passwords or to `true` to allow the LDAP server to validate an empty password (the default is `false`).

A.1.4. Defining Role Search

The LDAP login module passes the properties to define the search for a particular user and its roles to the LDAP server.



IMPORTANT

The following role search settings are similar to the LdapExtLoginModule settings; however, the recursion now finds the roles listed within a DN.

rolesCtxDN

The fixed DN of the context to search for user roles. Consider that this is not the Distinguished Name of where the actual roles are; rather, this is the DN of where the objects containing the user roles are (e.g. for active directory, this is the DN where the user account is)

roleFilter

defines a search filter used to locate the roles associated with the authenticated user. The input username/userDN as obtained from the login module callback substitutes the `{0}` expression in the filter definition. The authenticated userDN substitutes the `{1}` in the filter definition. An example search filter that matches the input username is `(member={0})`. An alternative that matches the authenticated userDN is `(member={1})`.



NOTE

If you omit the `roleFilter` attribute, the role search will use the UserDN as the DN to obtain the `roleAttributeID` value.

roleAttributeID

defines the role attribute of the context that corresponds to the name of the role. If the `roleAttributesDN` property is set to `true`, this property is the DN of the context to query for the `roleNameAttributeID` attribute. If the `roleAttributesDN` property is set to `false`, this property is the attribute name of the role name.

roleAttributesDN

defines if the role attribute contains the fully distinguished name of a role object or the role name. If `false`, the role name is taken from the value of the user's role attribute. If `true`, the role attribute represents the distinguished name of a role object. The role name is taken from the value of the `roleNameAttributeID` attribute of the corresponding object. In certain directory schemas (for

example, Microsoft Active Directory), role (group)attributes in the user object are stored as DNs to role objects and not as simple names. In such case, set this property to `true`. The default value of this property is `false`.

roleNameAttributeID

defines the role attribute of the context which corresponds to the name of the role. If the `roleAttributesDN` property is set to `true`, this property is used to find the name attribute of the role object. If the `roleAttributesDN` property is set to `false`, this property is ignored.

recurseRoles

Enables a recursive role search. The login module tracks already added roles to handle cyclic references.

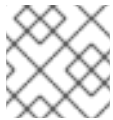
searchScope

sets the search scope to one of the following (the default value is `SUBTREE_SCOPE`):

- `OBJECT_SCOPE` - searches the named roles context only.
- `ONELEVEL_SCOPE` - searches directly in the named roles context.
- `SUBTREE_SCOPE` - searches only the object if the role context is not a `DirContext?`. If the roles context is a `DirContext?`, the subtree rooted at the named object and the named object itself are searched.

searchTimeLimit

defines the timeout for the user and role searches in milliseconds (defaults to 10000, that is 10 seconds).



NOTE

Both searches use the same `searchTimeLimit` setting.

A.2. EXAMPLES OF FULL LDAP AUTHENTICATION

The following example configurations show the full LDAP authentication with `AdvancedLdapLoginModule` for Active Directory and FreeIPA. The configuration differ in the `baseFilter` attribute as this is the name identified by the `SPNEGOLoginModule`.

The options `bindAuthentication`, `jaasSecurityDomain`, and `java.naming.provider.url` configure how the login module connects to LDAP and how the authentication occurs.

The `baseCtxDN` option is the DN to start the search for the user and the `baseFilter` attribute in these examples searches for the user using the `sAMAccountName` attribute on Active Directory and `uid` attribute on FreeIPA.

The `memberOf` attribute is read directly from the user, therefore there is no need to specify the `rolesCtxDN` or `roleFilter` property: the attribute defined for the `roleAttributeID` option is read directly from the user.

The `roleAttributesDN` option specifies that this value is a DN so the group object is retrieved and the `roleNameAttributeID` option specifies that the attribute `cn` is read from the group. The login module returns this role.

The `recurseRoles` is set to `true` so the DN from the located group is used to repeat the process so if a group is configured with the `memberOf` attribute then this is recursively used to locate all the roles.

A.2.1. Full LDAP Authentication for Active Directory

The following is an extract of the dumped `ldiff` from the example Active Directory domain:

```
dn: CN=Darran
Lofthouse, CN=Users, DC=vm104, DC=gsslab, DC=rdu, DC=redhat, DC=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: user
cn: Darran Lofthouse
distinguishedName:
  CN=Darran Lofthouse, CN=Users, DC=vm104, DC=gsslab, DC=rdu, DC=redhat, DC=com
memberOf: CN=Banker, CN=Users, DC=vm104, DC=gsslab, DC=rdu, DC=redhat, DC=com
name: Darran Lofthouse
sAMAccountName: darranl
userPrincipalName: darranl@vm104.gsslab.rdu.redhat.com

dn: CN=Banker, CN=Users, DC=vm104, DC=gsslab, DC=rdu, DC=redhat, DC=com
objectClass: top
objectClass: group
cn: Banker
member:
  CN=Darran Lofthouse, CN=Users, DC=vm104, DC=gsslab, DC=rdu, DC=redhat, DC=com
distinguishedName:
  CN=Banker, CN=Users, DC=vm104, DC=gsslab, DC=rdu, DC=redhat, DC=com
memberOf: CN=Trader, CN=Users, DC=vm104, DC=gsslab, DC=rdu, DC=redhat, DC=com
name: Banker
sAMAccountName: Banker

dn: CN=Trader, CN=Users, DC=vm104, DC=gsslab, DC=rdu, DC=redhat, DC=com
objectClass: top
objectClass: group
cn: Trader
member: CN=Banker, CN=Users, DC=vm104, DC=gsslab, DC=rdu, DC=redhat, DC=com
distinguishedName:
  CN=Trader, CN=Users, DC=vm104, DC=gsslab, DC=rdu, DC=redhat, DC=com
name: Trader
sAMAccountName: Trader
```

The following configuration requires a username and password to be provided for the authentication process:

```
<application-policy name="SPNEGO">
  <authentication>
    <login-module
      code="org.jboss.security.negotiation.spnego.AdvancedLdapLoginModule"
      flag="required">
```

```

    <module-option name="bindAuthentication">GSSAPI</module-option>
    <module-option name="jaasSecurityDomain">host</module-option>
    <module-option
name="java.naming.provider.url">ldap://VM104:3268</module-option>

    <module-option
name="baseCtxDN">CN=Users,DC=vm104,DC=gsslab,DC=rdu,DC=redhat,DC=com</modu
le-option>
    <module-option name="baseFilter">(sAMAccountName={0})</module-
option>

    <module-option name="roleAttributeID">memberOf</module-option>
    <module-option name="roleAttributeIsDN">>true</module-option>
    <module-option name="roleNameAttributeID">cn</module-option>

    <module-option name="recurseRoles">>true</module-option>
</login-module>
</authentication>
</application-policy>

```

A.2.2. Full LDAP Authentication for Free IPA

The following is an extract of the dumped ldif from the example FreeIPA domain:

```

dn: uid=darranl,cn=users,cn=accounts,dc=jboss,dc=org
displayName: Darran Lofthouse
uid: darranl
title: Mr
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
objectClass: inetUser
objectClass: posixAccount
objectClass: krbPrincipalAux
objectClass: radiusprofile
sn: Lofthouse
mail: darran.lofthouse@jboss.com
krbPrincipalName: darranl@JBOSS.ORG
givenName: Darran
cn: Darran Lofthouse
initials: DL
memberOf: cn=banker,cn=groups,cn=accounts,dc=jboss,dc=org
memberOf: cn=Trader,cn=groups,cn=accounts,dc=jboss,dc=org

dn: cn=Banker,cn=groups,cn=accounts,dc=jboss,dc=org
objectClass: top
objectClass: groupofnames
objectClass: posixGroup
objectClass: inetUser
cn: Banker
memberOf: cn=trader,cn=groups,cn=accounts,dc=jboss,dc=org
member: uid=darranl,cn=users,cn=accounts,dc=jboss,dc=org

```

```
dn: cn=Trader,cn=groups,cn=accounts,dc=jboss,dc=org
objectClass: top
objectClass: groupofnames
objectClass: posixGroup
objectClass: inetUser
cn: Trader
member: cn=Banker,cn=groups,cn=accounts,dc=jboss,dc=org
```

The following configuration requires a username and password to be provided for the authentication process:

```
<application-policy name="SPNEGO">
  <authentication>
    <login-module
      code="org.jboss.security.negotiation.spnego.AdvancedLdapLoginModule"
      flag="required">
      <module-option name="bindAuthentication">GSSAPI</module-option>
      <module-option name="jaasSecurityDomain">host</module-option>
      <module-option
name="java.naming.provider.url">ldap://kerberos.jboss.org:389</module-
option>

      <module-option
name="baseCtxDN">cn=users,cn=accounts,dc=jboss,dc=org</module-option>
      <module-option name="baseFilter">(uid={0})</module-option>

      <module-option name="roleAttributeID">memberOf</module-option>
      <module-option name="roleAttributeIsDN">>true</module-option>
      <module-option name="roleNameAttributeID">cn</module-option>

      <module-option name="recurseRoles">>true</module-option>
    </login-module>
  </authentication>
</application-policy>
```

APPENDIX B. REVISION HISTORY

Revision 5.2.0-100.400 Rebuild with publican 4.0.0	2013-10-31	Rüdiger Landmann
Revision 5.2.0-100 Incorporated changes for JBoss Enterprise Application Platform 5.2.0 GA. For information about documentation changes to this guide, refer to <i>Release Notes 5.2.0</i> .	Wed 23 Jan 2013	Russell Dickenson
Revision 5.1.2-109 Rebuild for Publican 3.	Wed 18 Jul 2012	Anthony Towns
Revision 5.1.2-100 Incorporated changes for JBoss Enterprise Application Platform 5.1.2 GA. For information about documentation changes to this guide, refer to <i>Release Notes 5.1.2</i> .	Thu 8 Dec 2011	Russell Dickenson