JBoss Enterprise Application Platform 5

Negotiation ユーザーガイド

JBoss Enterprise Application Platform 5 向け
エディッション 5.1.2

Last Updated: 2017-10-26

JBoss Enterprise Application Platform 5 Negotiation ユーザーガイド

JBoss Enterprise Application Platform 5 向け
エディッション 5.1.2

Darran Lofthouse
darran.lofthouse@redhat.com

Eva Kopalová
ekopalova@redhat.com

法律上の通知

概要

JBoss Negotiation Guide は、JBoss Enterprise Application Platform に SPNEGO 認証を設定したいと考えるシステム管理者や開発者向けです。本ガイドは、SPNEGO 認証と LDAP サーバーを統合可能にする AdvancedLDAPLoginModule セットアップの追加情報や設定に関する説明をしています。

目次

第1章 はじめに

JBoss Negotiation は JBoss Enterprise Application Platform のコンポーネントで、SPNEGO ベース (Simple and protected Negotiation) の シングルサインオン (SSO : Single Sign on) メカニズムを提供します。

JBoss Negotiation は ***$JBOSS_HOME*/jboss-as/common/lib/jboss-negotiation.jar** に置かれています。

SPNEGO はクライアントとサーバー認証の Generic Security Services Application Program Interface (GSSAPI) メカニズムです。リモートシステムへのサイレント認証が可能、かつセキュリティサービスへアクセスできるようになります。また、ユーザーの認証情報をリモートシステムに委譲し、リモートシステムがユーザーの代わりに他のシステムへコンタクトすることもできます。

1.1. SPNEGO 認証プロセス

一般的に、サーバーでのユーザー認証時、クライアントは入力した認証情報をサーバーに送り、サーバーのログインモジュールが、保存されている認証情報とその認証情報を照合します。しかし、SPNEGO 認証はいくつかの点で異なります。

1. アプリケーションサーバーは KDC に対してサーバー自身を認証し、ユーザー認証できるようになる前にチケットを取得します。
2. その後にのみ、サーバーはクライアントに対し認証のプロンプトを出します。クライアントは、SPNEGO トークンに応答し、サーバーはサーバー自体のチケットを使いクライアントのチケットを解読し、クライアントへ応答します。
3. 必要であればクライアントは、サーバーにそれ自体を認証するようにリクエストすることができます。
4. また、呼び出しているクライアントの代わりにサーバーが他のシステムを呼び出すことができるよう、クライアントはサーバーへ認証情報を委譲することができます。

JBoss Negotiation は通常以下のようなシナリオで便利です。

- ユーザーが Active Directory ドメインや FreeIPA で統括されているログイン方法で、デスクトップコンピューターにログインする場合
- ユーザーが Web ブラウザーを開き、JBoss Negotiation を使う Web アプリケーションへアクセスする場合
- Web ブラウザーがデスクトップの認証情報を Web アプリケーションに転送する場合

重要

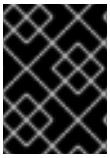
Active Directory および FreeIPA を設定し JBoss Negotiation を使うことができます (6章*FreeIPA の設定* および 5章*Microsoft Active Directory の設定* を参照)。

1.2. 設定概要

ご利用中の環境を JBoss Negotiation を使えるように設定するには、以下を行う必要があります。

- JBoss Negotiation を利用できるようアプリケーションサーバーを設定 (2章*設定* 参照)。

- オプションで、JBoss Negotiation を使えるよう Active Directory あるいは FreeIPA を設定 (5章*Microsoft Active Directory の設定* あるいは 6章*FreeIPA の設定* 参照)。
- JBoss Negotiation を使えるようクライアントの Web ブラウザーを設定 (7章*Web ブラウザーの設定* 参照)
- Negotiation Toolkit で設定をテスト (8章*Negotiation Toolkit*)
- JBoss Negotiation を利用できるよう Web アプリケーションを設定 (9章*Web アプリケーションの設定* 参照)。



重要

JBoss Negotiation を利用できるようご利用中のアプリケーションを設定する前に Negotiation Toolkit で設定のテストを行います (8章*Negotiation Toolkit* 参照)。

第2章 設定

JBoss Application Server で JBoss Negotiation 実行できるよう設定するには、以下を行う必要があります。

- コアとなる認証メカニズムを拡張し JBoss Negotiation に対応します (SPNEGO 認証システムの追加)。
- アプリケーションのセキュリティドメインを定義し、アプリケーションが SPNEGOLoginModule 経由でアプリケーションサーバーと通信できるようにします。
- サーバーのセキュリティドメインを定義し、アプリケーションサーバーが KDC に対して初回認証を行えるようにします。

サーバーが今までに realm 認証設定がされていない場合は、サーバーが認証レルム (Kerberos realm) を検索できるように、realm プロパティを設定する必要がある場合もあります。

JBoss Negotiation は Negotiation Toolkit という Web アプリケーションに同梱されており、SPNEGO 設定のテストが可能です。ご自身の Web アプリケーション上でテストする前にこのアプリケーションの利用を検討してみてください (8章*Negotiation Toolkit* 参照)。

2.1. SPNEGO 認証システムの追加

SPNEGO 認証システムをコアの認証メカニズムに追加するには、以下を行います。

1. ***$JBOSS_HOME*/server/*PROFILE*/deployers/jbossweb.deployer/META-INF/war-deployers-jboss-beans.xml** を開き編集します。
2. **authenticators** プロパティの場所を特定します。
3. 以下のエントリをこのプロパティに追加します。

```
<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>
```

キーの値は任意ですが、Negotiation Toolkit を使い利用中のサーバー設定をテストしたい場合は、このツールは、この名前を持つ SPNEGO 認証システムでしか機能しないため、**SPNEGO** 値を使うようにしてください。

2.2. サーバーのセキュリティドメインの定義

アプリケーションサーバーは、セキュリティドメインを定義し、KDC の初回認証ができるようにする必要があります。

重要

Krb5LoginModule は、ローカル認証情報のキャッシュを使うことができますが、このオプションは SPNEGO で必要な storekey オプションとの互換はありません。このモジュールがローカル認証キャッシュを使用していないか確認してください。

サーバーのセキュリティドメインを定義するには、以下を行います。

1. **$JBOSS_HOME/server/$PROFILE/conf/login-config.xml** ファイルを開き編集します。
2. アプリケーションポリシーの要素を authentication 要素で定義します。この要素のオプションは以下のとおりです。

   storeKey

   true の場合は、Subject に秘密鍵がキャッシュされます (**true** に設定)。

   useKeyTab

   true の場合は、キーが keyTab ファイルからロードされます (**true** に設定)。

   principal

   この属性は、プリンシパルのフルネームを keyTab ファイルから取得するよう記述する必要があります。

   keyTab

   この属性は、keyTab ファイルへの完全パスとサーバーキーを定義します (サーバーと KDC の情報を暗号化するキー)。

   doNotPrompt

   true の場合は、パスワードのプロンプトが出てこないようにします。 (これはサーバーであるため、**true** に設定)。

   debug

   true の場合、システムは追加のデバッグ情報を STDOUT にログします。

例2.1 サーバーのセキュリティドメイン

```
<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. アプリケーションのセキュリティドメインの定義

アプリケーションが SPNEGOLoginModule 経由でアプリケーションサーバーと通信できるよう、アプリケーションサーバー上でアプリケーションのセキュリティドメインを定義する必要があります。

アプリケーションのセキュリティドメインを定義するには、以下を行います。

1. **`$JBOSS_HOME/jboss-as/server/$PROFILE/conf/login-config.xml`** ファイルを開き編集します。
2. 新しいアプリケーションポリシーを以下のチェーン設定で定義します。
   - 以下のオプションを使った SPNEGOLoginModule とその設定

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

     パスワードスタッキングオプションは、他のログインモジュールでクライアント側の認証をアクティベートします。**`password-stacking`** オプションを**`useFirstPass`**に設定し、まず、共有されているユーザー名が**`javax.security.auth.login.name`**、パスワードが **`javax.security.auth.login.password`** のものをモジュールが検索するようにします (詳細情報は、Password Stacking chapter in the Security Guideを参照)。

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

     serverSecurityDomain オプションは、サーバーのセキュリティドメインを定義し、これにより、認証モジュール (Kerberos) とサーバー認証プロパティ (「サーバーのセキュリティドメインの定義」 参照) を定義します。

   - 認証ユーザーのロールと設定オプションを返すログインモジュール。プロパティファイルからユーザーロールを取得する UsersRolesLoginModule あるいは GSSAPI に従う LDAP サーバーからユーザーロールを取得する AdvancedLdapLoginModule を利用することができます。詳細情報は、「ロールマッピング」 を参照してください。

例2.2 アプリケーションのセキュリティドメイン

```
<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>
```

例2.2「アプリケーションのセキュリティドメイン」 では、２つのログインモジュールを持つ、**SPNEGO** と呼ばれるアプリケーションのセキュリティドメインを定義しました。

- **org.jboss.security.negotiation.spnego.SPNEGOLoginModule** は、SPNEGO ユーザー認証を提供します。
- **org.jboss.security.auth.spi.UsersRolesLoginModule** は、SPNEGOLoginModule で認証されたユーザーロールを返します (ロールはユーザーのプロパティファイルからフィルタリングされます)。

2.4. ロールマッピング

ユーザーが KDC で認証されると (org.jboss.security.negotiation.spnego.SPNEGOLoginModule で発生)、アプリケーションサーバーはユーザーロールを取得する必要があります。認証は、users.properties ファイルからユーザーロールを取得する org.jboss.security.auth.spi.UsersRolesLoginModule か、LDAP サーバーからユーザーロールを取得する org.jboss.security.negotiation.AdvancedLdapLoginModule を利用できます。

2.4.1. ユーザープロパティファイルでロールマッピングを設定

SPNEGO が認証済みのユーザーロールを users.properties ファイルから取得できるようにするには、以下を実行します。

1. アプリケーションのセキュリティドメインで、SPNEGO 認証の２番目のログインモジュールを**org.jboss.security.auth.spi.UsersRolesLoginModule** に設定 (例2.2「アプリケーションのセキュリティドメイン」 参照) し、モジュールオプションを渡します (セキュリティガイドの　UsersRolesLoginModule 参照)。
2. **$JBOSS_HOME/server/$PROFILE/conf/login-config.xml** ファイルでアプリケーションのセキュリティドメインを定義している場合、**$JBOSS_HOME/server/$PROFILE/conf/props/spnego-users.properties** ファイルでユーザーロールを定義します。以下のパターンを使ってください (***fullyQualifiedUserName=comma-separatedListOfRoles***)。

例2.3 users.properties file

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

2.4.2. LDAP サーバーでロールマッピングを設定

AdvancedLdapLoginModule では、SPNEGOLoginModule で KDC との認証を以前に済ませているユーザーのロールを取得できるようになります。AdvancedLdapLoginModule は、LdapExtLoginModule に基づき、GSSAPI に従います。

注記

本項では、SPNEGOLoginModule を使ったチェーン設定にあるモジュールについて説明していますが、LDAP サーバーからの認証とロール検索にこのモジュールを利用できます。このような設定に関する詳細は、付録A *高度な LDAP ログインモジュール：完全 LDAP 認証* を参照してください。

SPNEGOLoginModule を持つチェーン設定で AdvancedLdapLoginModule を使うには、SPNEGO アプリケーションのセキュリティドメインにある、SPNEGOLoginModule とチェーンする必要があります。SPNEGO 認証の 2 番目のログインモジュールを **`org.jboss.security.negotiation.AdvancedLdapLoginModule`** に設定します (例2.2「アプリケーションのセキュリティドメイン」 参照)。

LDAP サーバーのロールマッピングを設定するには、以下をする必要があります。

- InitialLdapContext プロパティを定義します。これらのプロパティを使い LDAP との関係を取得します (「初期の LDAP コンテキストを GSSAPI で定義」 を参照。Java API の詳細は、http://download.oracle.com/javase/6/docs/api/javax/naming/ldap/InitialLdapContext.html を参照)。
- 識別名 (DN: Distinguished Name) プロパティを定義します。これらのプロパティを使い、LDAP サーバーで認証済みユーザーを検索します (「DN 検索の定義」 参照)。
- ロールの検索プロパティを定義します。これらのプロパティは、LDAP サーバー上のロール検索を管理しています (「ロール検索の定義」)。

ログインモードで設定しているプロパティは、InitialLdapContext のコンストラクターに渡されます。つまり、LdapCtxFactory に対応しているオプションのいずれかを利用可能になります。

2.4.2.1. 初期の LDAP コンテキストを GSSAPI で定義

初期の LDAP コンテキストを取得するには、アプリケーションセキュリティドメインにある AdvancedLdapLoginModule に対し、以下のモジュールプロパティを定義します (「アプリケーションのセキュリティドメインの定義」)。

bindAuthentication

認証タイプを定義します (プロパティの値を **`GSSAPI`** に設定し GSSAPI ベースの認証を利用します)。

jaasSecurityDomain

接続に必要なサブジェクトを取得するのに利用するセキュリティドメインを定義します (必須の jaasSecurityDomainを定義する際の情報に関しては、「サーバーのセキュリティドメインの定義」を参照)。

2.4.2.2. DN 検索の定義

モジュールが LDAP 初期コンテキストを作成した後、提供済みのユーザー名をとり、ユーザー DN を検索します。検索プロパティを定義するには、以下のプロパティを提示します。

baseCtxDN

コンテキストの固定 DN を定義しユーザーを検索します。これは、ユーザーが実際に置かれている場所の識別名 (DN) ではなく、ユーザーが含まれているオブジェクトが置かれている場所の DN という点を考慮します (つまり、Active Directory では、ユーザーアカウントを持つ DN ということにな

ります)。

baseFilter

認証ユーザーのコンテキストの場所を突き止めるために使う検索フィルターを定義します。ログインモジュールのコールバックから取得した通りの、入力ユーザー名／ユーザー DN は、**{0}** 表現を代入します。この代入動作は、標準のDirContext?.search(Name, String, Object[], SearchControls?cons) メソッドから来ています。一般的な検索フィルターの例は、**(uid={0})**です。

searchTimeLimit

ユーザーとロール検索のタイムアウト時間をミリ秒で定義します (デフォルトは 10000 で、１０秒となっています)。

注記

ユーザー DN 検索を無効にするには、**baseCtxDN** プロパティを省略します。提示したユーザー名をログインモジュールで DN として使います。

2.4.2.2.1. ユーザー認証

AdvancedLDAPLoginModule が最初のログインモジュールでなく、以前のログインモジュールがすでにユーザー認証を済ましている場合、ユーザー認証は省略されます。

ユーザー認証は、以下のプロパティを定義してください。

allowEmptyPassword

空の (length==0) パスワードが LDAP サーバーに渡された場合。LDAP サーバーは空のパスワードを無名ログインとして処理します。このプロパティを **false** に設定し、空のパスワードを受け付けないようにするか、**true** に設定し、LDAP サーバーが空のパスワードを認証できるようにします (デフォルトは、**false**)。

2.4.2.3. ロール検索の定義

AdvancedLdapLoginModule は、LDAP サーバーに対する特定ユーザーやロール検索を定義するプロパティを渡します。

重要

以下のロール検索設定は、LdapExtLoginModule 設定に似ていますが、反復により DN 内にリストされているロールも検索します。

rolesCtxDN

コンテキストの固定 DN を定義しユーザーロールを検索します。これは、実際のロールが実際に置かれている場所の識別名 (DN) ではなく、ユーザーロールが含まれているオブジェクトが置かれている場所の DN という点を考慮します (つまり、Active Directory では、ユーザーアカウントを持つ DN ということになります)。

roleFilter

認証済みユーザーロールの場所をつきとめるために利用する検索フィルターを定義します。ログインモジュールのコールバックから取得した入力ユーザ名／userDN はフィルター定義で **{0}** 表現を

代入します。認証済み userDN は、フィルター定義で **{1}** を代入します。入力したユーザ名と一致する検索フィルターの例は、**(member={0})** です。もう 1 つは、認証済みの userDN に一致する場合で、 **(member={1})** となっています。

注記

roleFilter 属性を飛ばすと、ロール検索は UserDN を roleAttributeID 値を取得する DN として利用します。

roleAttributeID

ロール名に該当するコンテキストのロール属性名を定義します。roleAttributeIsDN プロパティが **true** に設定されている場合、このプロパティは、roleNameAttributeID 属性に対しクエリを行うコンテキストの DN となります。roleAttributeIsDN プロパティが **false** に設定されている場合、このプロパティは、ロール名の属性名となります。

roleAttributeIsDN

ロール属性がロールオブジェクトあるいはロール名の完全な識別名を含むか定義します。 **false**の場合、ロール名はユーザーのロール属性の値から取得します。**true**の場合、ロール属性はロールオブジェクトの識別名を表します。ロール名は、該当オブジェクトのroleNameAttributeId 属性の値から取得します。特定のディレクトリスキーマでは (Microsoft Active Directory など)、ユーザーオブジェクトのロール (グループ) 属性は、簡易名ではなく DN としてロールオブジェクトに保存されます。このような場合、このプロパティは **true** に設定します。このプロパティのデフォルト値は**false** です。

roleNameAttributeID

ロール名に該当するコンテキストのロール属性を定義します。roleAttributeIsDN プロパティが **true** に設定されている場合、このプロパティを使いロールオブジェクトの属性を検索します。また、roleAttributeIsDN property が **false** に設定されている場合、このプロパティは無視されます。

recurseRules

再帰ロール検索を有効にするか定義します。ログインモジュールは、すでに追加済みのロールをトラッキングし、循環参照を処理します。

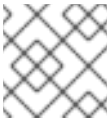
searchScope

検索範囲を以下のいずれかに制限することができます (デフォルト値は**SUBTREE_SCOPE**)。

- OBJECT_SCOPE - 指定のロールコンテキストのみを検索します。
- ONELEVEL_SCOPE - 指定のロールコンテキストを直接検索します。
- SUBTREE_SCOPE - ロールコンテキストが DirContext? でない場合、オブジェクトのみを検索します。ロールコンテキストが DirContext? の場合、サブツリーは、指定のオブジェクトにルートを置き、そのオブジェクト自体を検索します。

searchTimeLimit

ユーザーとロール検索のタイムアウト時間をミリ秒で定義します (デフォルトは 10000 で、１０秒となっています)。



注記

これらの検索は両方、同じ searchTimeLimit 設定を使います。

2.4.3. SPNEGO モジュールを使った LDAP 設定例

以下の SPNEGO 認証設定は、SPNEGOLoginModule と LDAP ログインモジュールを使います。このチェーン設定は、**`baseFilter`** 値以外は、FreeIPA と Active Directory の両方で同じです。この **`baseFilter`** は、SPNEGOLoginModule で識別される LDAP にて検索する名称を定義します (関連の ldiff dump については、「Active Directory 向けの完全 LDAP 認証」 と 「Free IPA 向けの完全 LDAP 認証」 を参照のこと)。

SPNEGOLoginModule が AdvancedLdapLoginModule に対して認証済みのユーザー名を渡せるよう、両ログインモジュールにて **`password-stacking`** プロパティが **`useFirstPass`** に設定されている点に注意してください。

2.4.3.1. FreeIPA でのチェーン設定

以下の設定では、FreeIPA の SPNEGOLoginModule の後に連結した AdvancedLdapLoginModule です。

```
<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. Active Directory でのチェーン設定

以下の設定では、Active Directory の SPNEGOLoginModule の後に連結した AdvancedLdapLoginModule です。

```
<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>
```

第3章 TRACE ロギング

JBoss Security および JBoss Negotiation の認証システムのロギングを有効にするには、以下を行います。

1. ***$JBOSS_HOME*/server/*$PROFILE*/conf/jboss-log4j.xml** を開きます。
2. 以下を追加し、 **org.jboss.security** の 完全 TRACE ロギングを有効にします。

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

3. オプションで、**com.sun.security.auth.module.Krb5LoginModule** ログインモジュール用に追加のロギングを許可します。方法は、**debug** オプションを**true** にします。

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

4. システムプロパティ **-Dsun.security.krb5.debug=true** を設定し GSSAPI ネゴシエーションプロセス全体の詳細出力を取得します。

3.1. メッセージ追跡の設定

TRACE レベルで交換メッセージを選択的にログに残すことができます。Request および Response 両方のメッセージを Hex あるいは Base 64 もしくは両方としてログに残すことができます。

メッセージ追跡の基本カテゴリーは、**org.jboss.security.negotiation.MessageTrace** です。TRACE ロギングをこのカテゴリーで有効にすると、全リクエストおよびレスポンスのメッセージが TRACE レベルで Hex および Base64 エンコーディングにてログに残されます。

例3.1 全メッセージの追跡設定

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

リクエストメッセージあるいはレスポンスメッセージのみにロギングを減らすには、**.Request** あるいは **.Response** を category 値に追加します。

例3.2 リクエストメッセージのみの追跡設定 (Hex および Base64 でメッセージをロギング)

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

例3.3 レスポンスメッセージのみの追跡設定 (Hex および Base 64 でメッセージをロギング)

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

特定のエンコーディングでメッセージのログを残すには、**.Hex** あるいは **.Base64** を category 値に追加します。

例3.4 定義したエンコーディングでメッセージを追跡

```
<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>
```

第4章 AUTHENTICATION プロパティをサーバーへ渡す手順

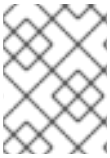
JBoss Negotiation 設定後、Kerberos realm プロパティを JBoss Application Server に渡したか確認する必要があります。

java.security.krb5.realm

サーバーが認証を行う kerberos realm

java.security.krb5.kdc

KDC ホスト名



注記

KDC の認証設定が済んでいるホストで JBoss 設定を行われている場合は、この手順を飛ばしてください。

このプロパティに関する詳細情報は、Java Generic Security Services (Java GSS) and Kerberos を参照してください。

コマンドラインから、あるいはサーバープロパティを追加することで、このプロパティをサーバーに渡すことができます。

4.1. コマンドラインからプロパティを渡す手順

コマンドラインからプロパティをサーバーに送るには、*KERBEROS.JBOSS.ORG* 部分をご利用中の realm で置き換え、該当の Java プロパティを使い **run** コマンドを実行します。

- Red Hat Enterprise Linux では、以下のコマンドを実行します。

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

- Windows では以下のコマンドを実行します。

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

これらのプロパティはサーバーがシャットダウンするまで有効で、サーバーが起動するたびにこれらのプロパティをサーバーに渡す必要があります。

4.2. システムプロパティにプロパティを追加

プロパティを永続化し、アプリケーションサーバーが SPNEGO メカニズムで起動するようにするには、***$JBOSS_HOME*/server/*$PROFILE*/deploy/properties-service.xml** 記述子のプロパティを定義します。これらのプロパティが最初の認証試行の前にロードされるようにしてください (サーバーが完全に起動する前は、JBoss は受信 HTTP 接続を許可しません)。

記述子を開き、以下の属性を **jboss:type=Service,name=SystemProperties** MBean に追加します。

```
<attribute name="Properties">
```

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

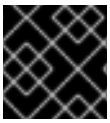
4.2.1. 複数の KDC

マスター KDC 以外に、スレーブ KDC を 1 つ以上使用する場合は、java.security.krb5.kdc システムプロパティの後に、KDC をコロンで区切りリストアップにしてください。このシステムでは、マスター KDC が利用できない場合、提示された代わりの KDC を使います。

例4.1 複数の KDC でサーバーを起動

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

第5章 MICROSOFT ACTIVE DIRECTORY の設定

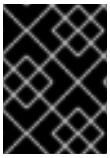


重要

本項のドメインの内容は、本ガイドの別の箇所で使われているものと異なります。

JBoss Negotiation を使いユーザー認証を行えるように Active Directory を設定するには、以下を行う必要があります。

- サーバーのユーザーアカウントを作成し、サービスプリンシパル名 (SPN: Service Principal Name) アカウントとして設定します。サービスプリンシパル名アカウント (SPN アカウント) のユーザーは、Kerberos サーバー、Active Directory、JBoss Web サーバーの間をつなげます。
- サーバーユーザー用に Keytab ファイルを生成し、アプリケーションサーバーにそのファイルをエクスポートします。アプリケーションサーバーは keytab を使い AD にて KDC を認証します。



重要

Active Directory ドメインコントローラーを利用しているようにしてください。ローカルで管理されているアカウントで Windows マシンを使うことはできません。

警告

本ガイドの説明は、Windows 2003 向けですので、お使いの Windows オペレーティングシステムに適した説明とは違ってくる場合があります。

5.1. アプリケーションサーバーのユーザーアカウント

AD ドメインコントローラーでアプリケーションサーバー向けの SPN アカウントを設定するには、**Setspn** と **Ktpass** が必要です。コマンドラインユーティリティーは、Windows Server 2003 Support Tools の一部で、アプリケーションサーバーとその HTTP サービスへサーバーのユーザー名をマッピングする際に便利です。

これらのユーティリティは、Microsoft web ページ から入手できます。

AD ドメインでサーバー向けに通常のユーザーアカウントを作成し (コンピューターのアカウントではなくユーザーアカウントであることを確認します)、そのアカウントをサービスアカウントにマッピングします。

5.1.1. サーバーユーザーの作成

サーバーの新規ユーザーを作成するには、以下を行なってください。

1. **スタート** → **管理ツール** → **Active Directory ユーザーとコンピューター**へ移動します。

2. **Active Directory ユーザーとコンピューター** のウィンドウで、**アクション** → **新規作成** → **ユーザー**へ移動します。

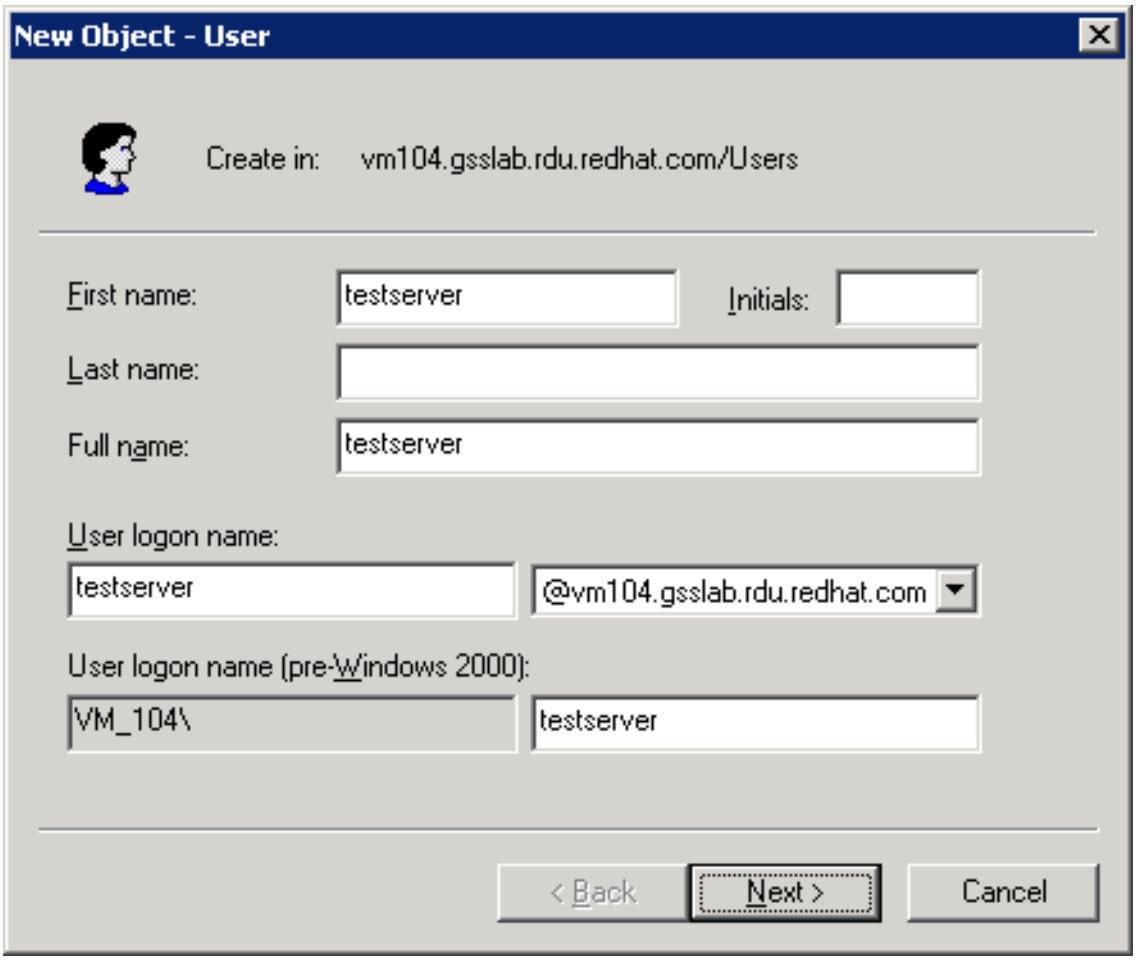
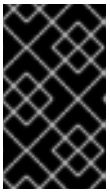


図5.1 新規ユーザー

3. **新規ユーザー** ウィンドウで、ユーザーの詳細を入力し、**Next** をクリックします。図5.1「新規ユーザー」 は**@vm104.gsslab.rdu.redhat.com** サーバーを使い、**testserver** と呼ばれるユーザーを定義します。

4. ユーザーのパスワードを入力し、**ユーザーはパスワードを変更できない** と **パスワードを無期限にする** を選択します。



重要

後にパスワードを変更すると keytab ファイルを無効にし、JBoss インストール設定が壊れてしまう可能性があるため、有効なパスワードを入力するようにします。

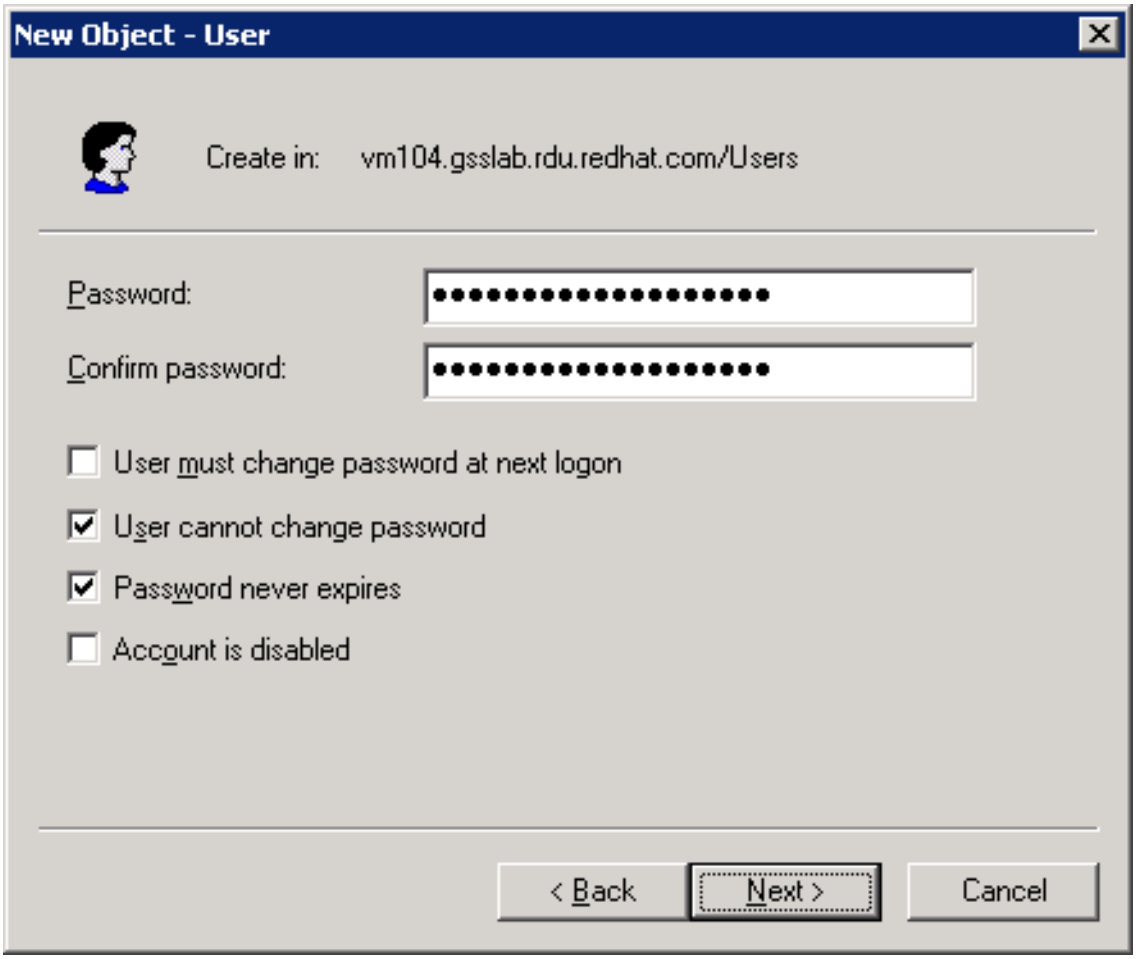


図5.2 新規ユーザーのパスワード

5. **Next** をクリックし、 **Finish** を押します。

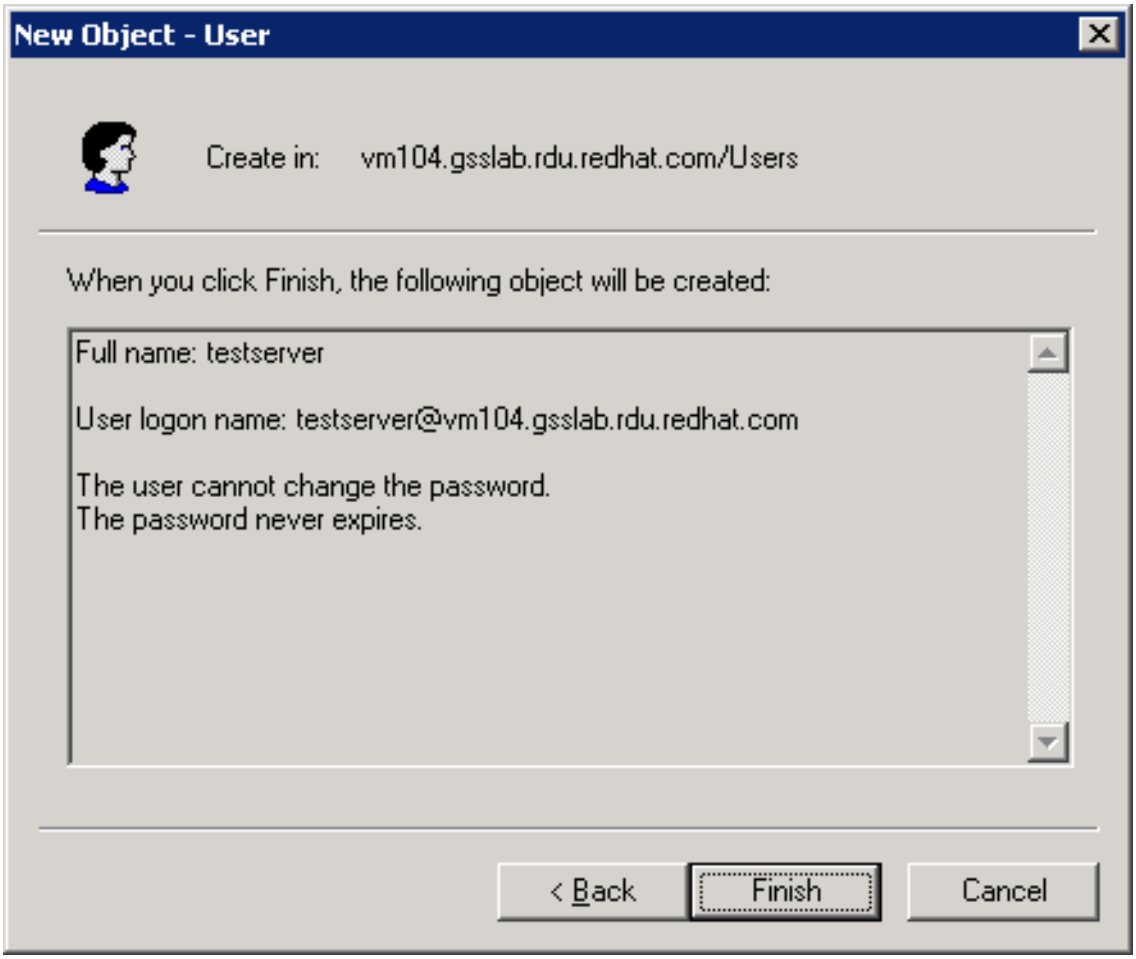


図5.3 新規ユーザーの終了

6. **Active Directory ユーザーとコンピューター** のウィンドウで、ユーザーを右クリックし、**Properties** をクリックします。

7. ユーザープロパティのウィンドウで、**アカウント**タブをクリックし、**アカウントオプション**の**Kerberos 事前認証を必要としない** と **このアカウントに DES 暗号化タイプを使う** を選択されているか確認します。

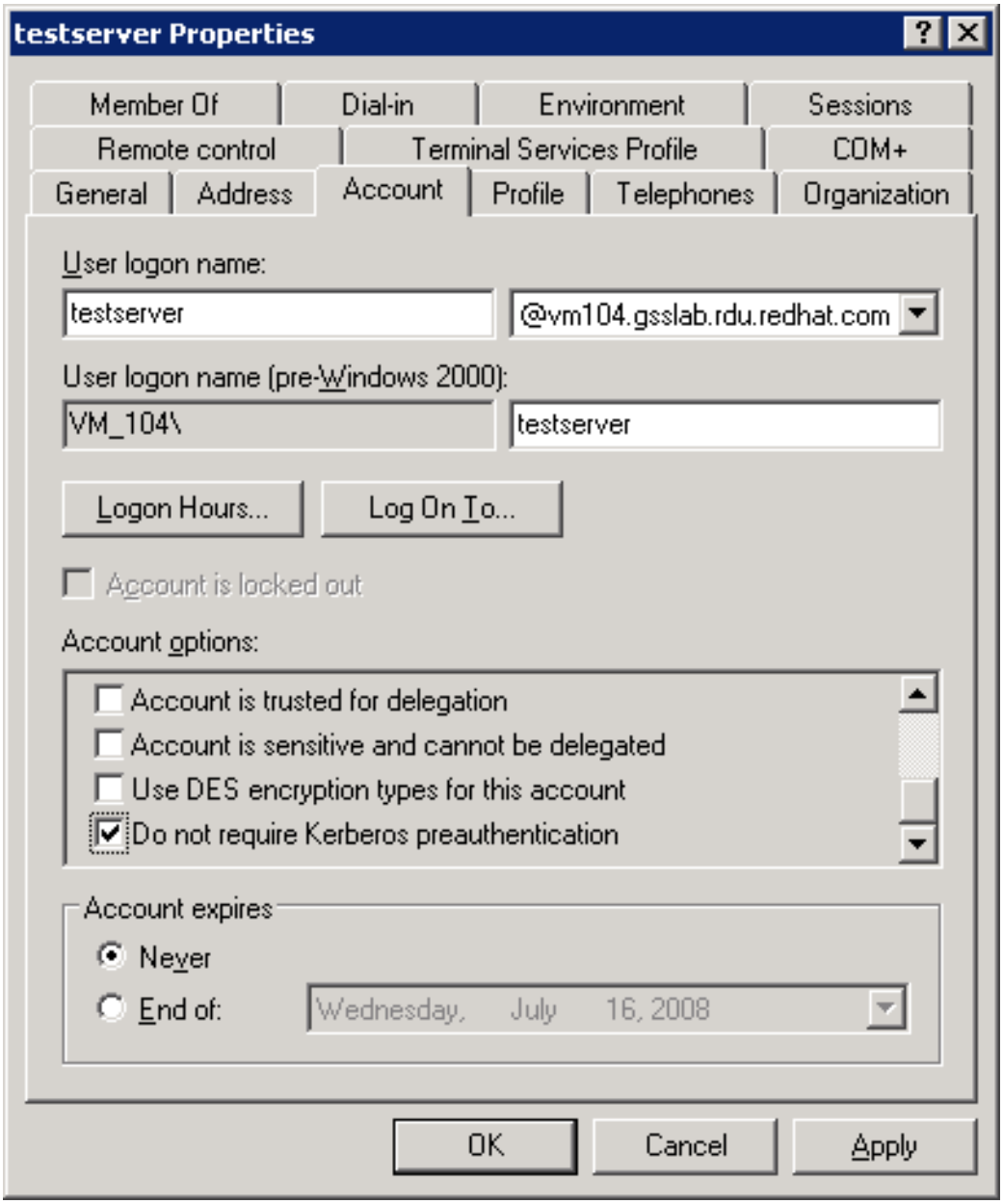


図5.4 ユーザープロパティ

ここで、作成したユーザーのKeytab ファイルを作成し、エクスポートする必要があります。

5.2. KEYTAB のエクスポート

アプリケーションサーバーのユーザーアカウントを作成しましたので、**Ktpass** ユーティリティを使い信頼済みホストとして SPN アカウントをマッピングし、サーバーの keytab をエクスポートします。

1. ktpass コマンドを実行し、信頼済みホストとして作成したユーザーをマッピングし keytab ファイルを生成します。**-princ** オプションは、マッピング先のサービスプリンシパルを定義し、**-mapuser** オプションは、マッピング先のユーザーアカウントを定義します。

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

例5.1 ktpass コマンド

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

2. プロンプトが出ると、ユーザーのパスワードを入力します。
3. 以下のコマンドを実行し利用可能なマッピングを表示し、新規のマッピングが登録されているか確認します。

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

例5.2 setspn コマンド

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

第6章 FREEIPA の設定

JBoss Negotiation を利用できるよう FreeIPA の設定する前に、FreeIPA が正しくインストール、設定されているか、またクライアントは Kerberos チケットを取得することができるか確認してください。FreeIPA の詳細文書は http://www.freeipa.org/ で入手できます。

警告

FreeIPA の対応暗号化タイプの性質上、JBoss アプリケーションサーバーを Java 6 JVM で暗号化の無制限強度が有効になった状態で実行する必要があります。

JBoss Negotiation を使いユーザー認証を行えるように FreeIPA を設定するには、以下を行う必要があります。

- サーバーのサービスプリンシパルを作成し、HTTP サービスをそのプリンシパルに追加します。サーバーユーザーが FreeIPA と JBoss Web サーバー間をつなぎます。
- サーバーユーザー用に Keytab ファイルを生成し、アプリケーションサーバーにそのファイルをエクスポートします。アプリケーションサーバーは keytab を使い FreeIPA にて KDC を認証します。

注記

以下の説明は、FreeIPA 1.1 に適用されます。

6.1. サービスプリンシパルの作成

JBoss Application Server の HTTP サービスを代表するサービスプリンシパルを作成し、クライアントがこのサービスのチケットをリクエストできるようにする必要があります。

注記

サービスプリンシパル作成に関する包括的な説明が http://freeipa.org/page/AdministratorsGuide#Managing_Service_Principals で確認できます。

1. 最も簡単にサービスプリンシパルを作成するには、FreeIPA WebUI を利用します。このツールへ管理者としてアクセスしてください。
2. **`Add Service Principal`** リンクをクリックします。

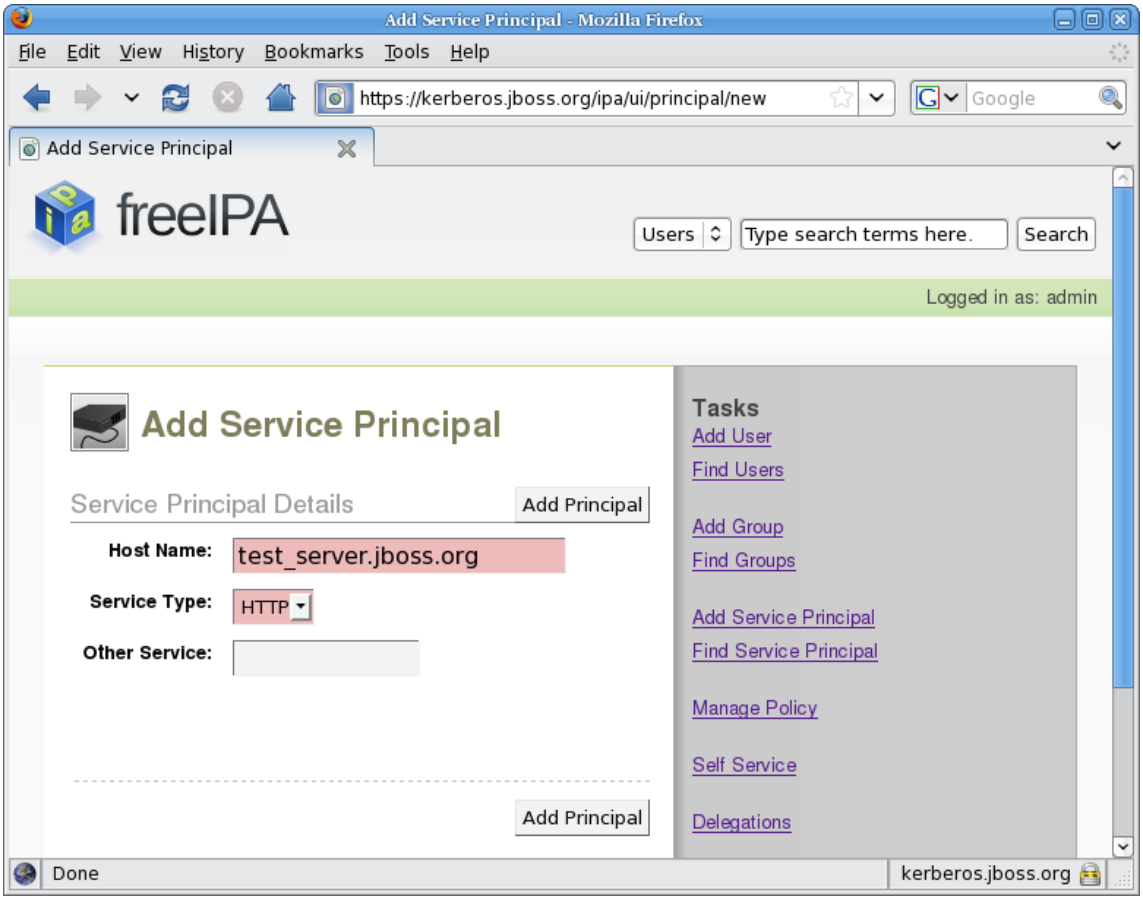


図6.1 サービスプリンシパルの追加

3. ホスト名をご利用中のサーバーのホスト名 (**`test_server.jboss.org`**) に、サービスタイプを **`HTTP`** に設定し **`Add Principal`** をクリックします。

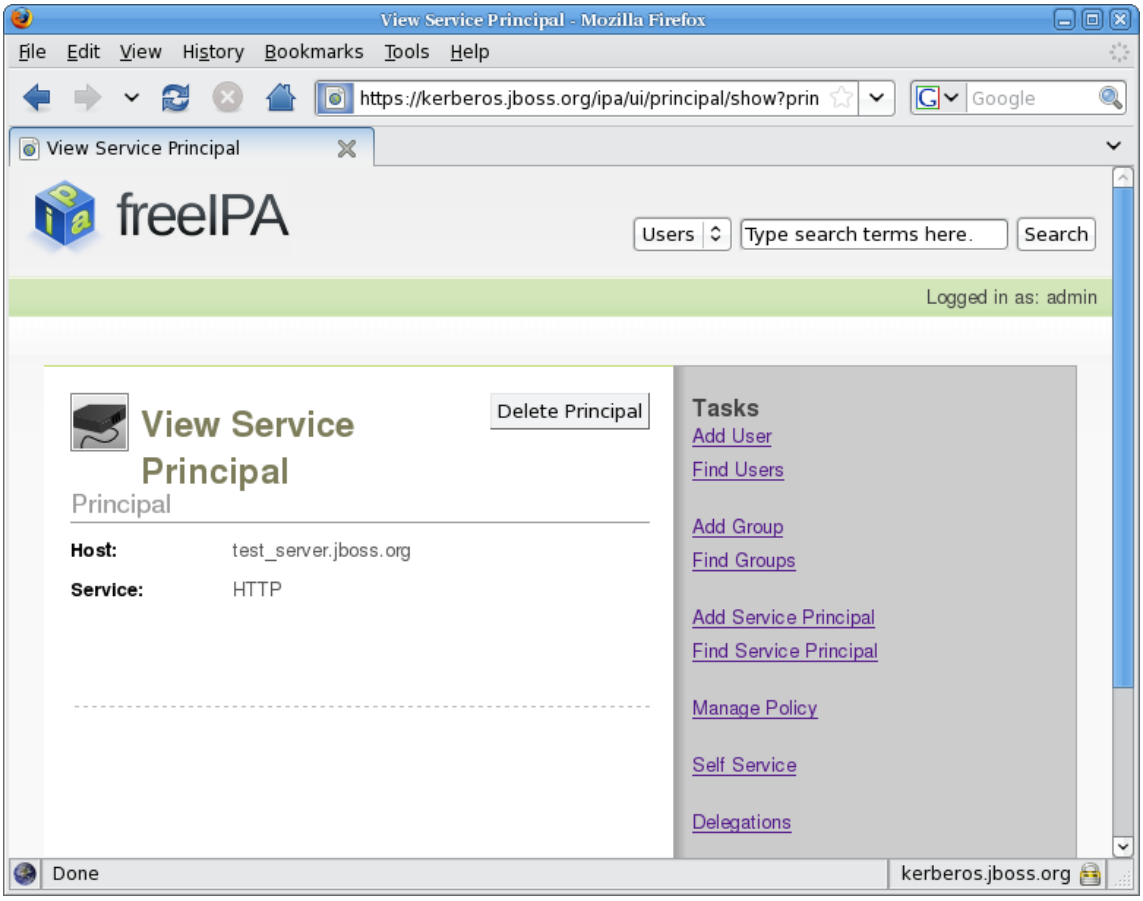


図6.2 サービスプリンシパルの表示

注記

サービスプリンシパルを作成するには、ホスト名を DNS とマッピングする必要があります。この手順に失敗した場合、コマンドラインから **ipa-addservice HTTP/test_server.jboss.org@JBOSS.ORG --force** コマンドを実行しプリンシパルを作成してください。

6.2. KEYTAB のエクスポート

警告

新規の keytab をエクスポートすると、サービスプリンシパルと紐付けられたシークレットをリセットし、そのプリンシパルで以前に作成された keytab を無効にします。

サーバーの keytab をエクスポートするには、以下を行なってください。

1. Kerberos Ticket-granting ticket (TGT) を管理者用に取得します。これには、**ipa-addservice HTTP/test_server.jboss.org@JBOSS.ORG --force** コマンドを実行します。

2. keytab を取得するには、以下のオプションをつけ、**ipa-getkeytab** コマンドを実行します。

-s

keytab 取得元の FreeIPA サーバー

-p

完全なプリンシパル名の非レルム部分

-k

keytab を添付するファイル

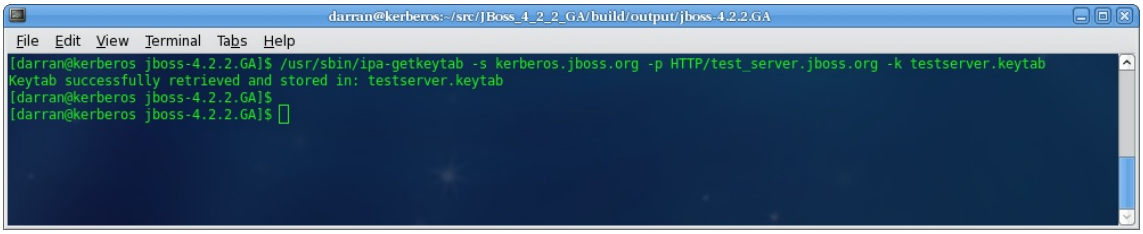


図6.3 Keytab の取得

サーバープリンシパルを設定し keytab をエクスポートした後、サーバーのセキュリティドメインが出力した keytab ファイルを使い (「サーバーのセキュリティドメインの定義」 参照)、２つ目のログインモジュールをクライアントが認証済みのユーザーに割り当てられたロールをロードできるよう設定しているか、確認してください (「ロールマッピング」 参照)。

第7章 WEB ブラウザーの設定

注記

Web ブラウザーは、通信を行うアプリケーションサーバーを信頼する必要があります。アプリケーションサーバーを信頼済みリソースに追加するには、JBoss アプリケーションサーバーの IP アドレスを信頼済みホストに追加します。Red Hat Enterprise Linux では、**`/etc/hosts`** ファイルを編集し、このファイルが確実にホスト名検索に利用されるようにしてください。Windows では、**`C:\windows\system32\drivers\etc\hosts`** を編集します。DNS サーバー上でもローカルのクライアントマシン上でも、この変更を加えることができます。

Kerberos realm が **`KERBEROS.JBOSS.ORG`** で、サーバーがホストする JBoss が**`testserver`** であれば、信頼済みホストとして追加する必要のある IP アドレスは **`testserver.kerberos.jboss.org`** です。

7.1. INTERNET EXPLORER の設定

Internet Explorer (IE) 上で JBoss Negotiation を有効にするというここでの説明は、Microsoft Windows 2003 上の Internet Explorer 6 が該当します。

デフォルトでは、Internet Explorer は **`Local intranet`** にあるサイトに対してのみSPNEGO 認証を実行します。SPNEGO ネゴシエーションを有効にするには、JBoss のサーバー URL をローカルのイントラネットサイトに追加します。

1. **ツール** メニューで、**インターネットオプション** をクリックします。

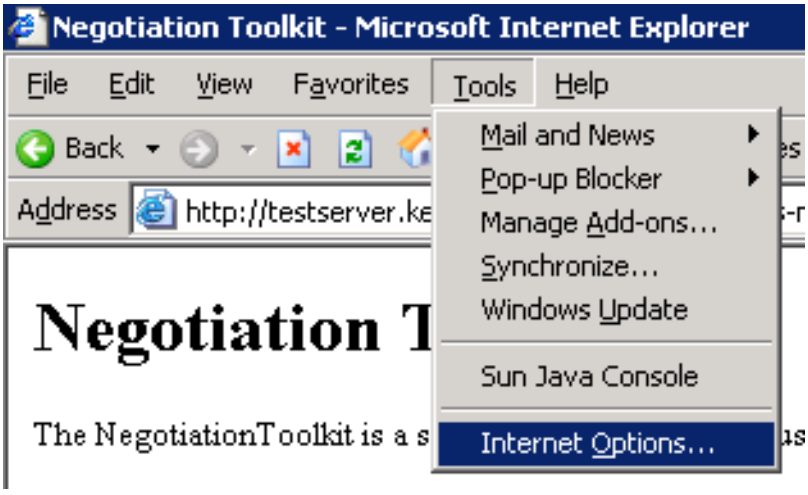


図7.1 ツール - インターネットオプション

2. **インターネットオプション** のダイアログで、**セキュリティ** タブをクリックします。
3. **セキュリティ** タブで、**ローカルイントラネット** アイコンが選択されているか確認し、**サイト**ボタンをクリックします。

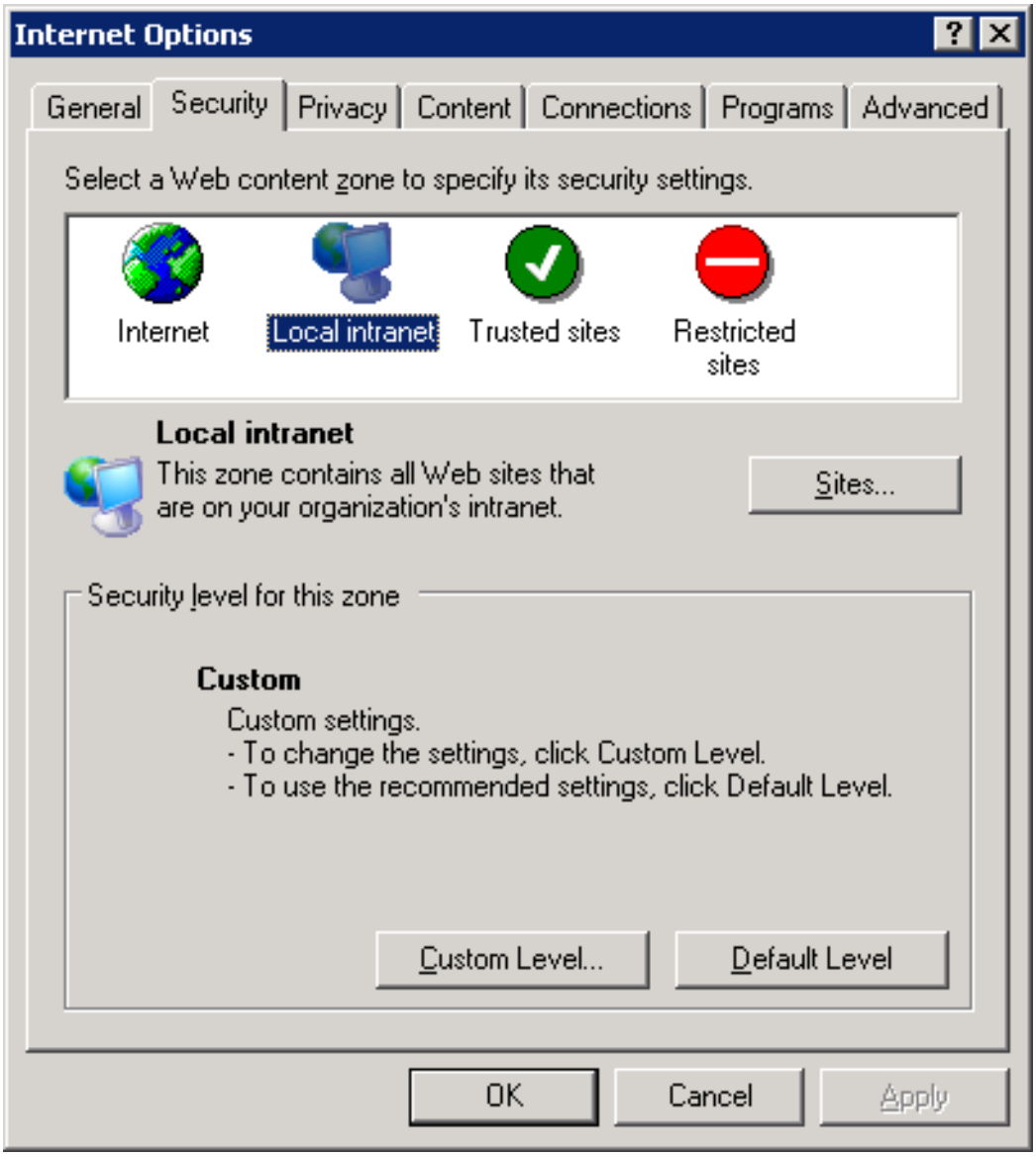


図7.2 インターネットオプション

4. **`Local intranet`** ダイアログで、JBoss がインストールされているサーバーの URL を入力し、**`Add`**をクリックします。

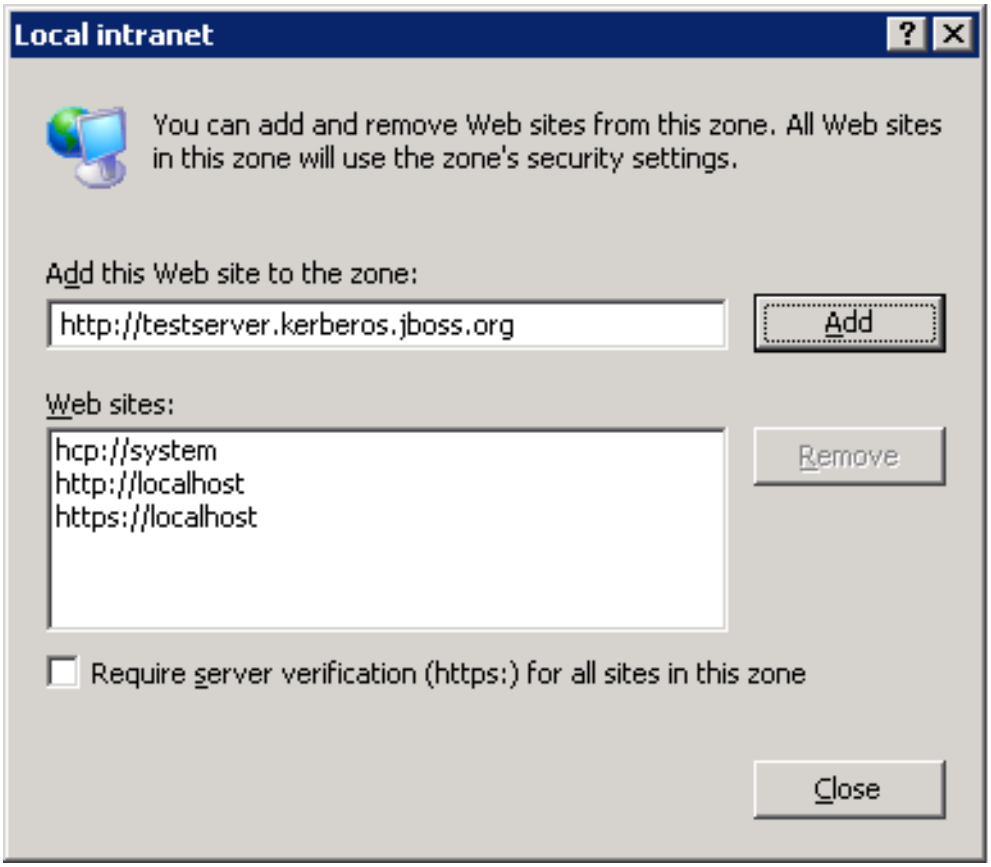


図7.3 ローカルイントラネット

サーバーが以下の**Web サイト** リストに表示されます。Internet Explorer は、JBoss インストール設定を信頼し、SPNEGO ネゴシエーションを実行します。**Basic Negotiation** サーブレットを使い Negotiation をテストするようにしてください (「Basic Negotiation」 参照)。

7.2. FIREFOX の設定

Mozilla Firefox で SPNEGO ネゴシエーションを有効にする方法の説明ですが、Microsoft Windows 2003 上の Mozilla Firefox 2.0.0.11、Fedora 9 上のFirefox 3.0.1 が該当します。

SPNEGO ネゴシエーションを有効にするには、Mozilla Firefox 設定を以下のように変更します。

1. Firefox の設定オプションで、about:config URL へ移動します。
2. フィルターを**network.negotiate** に設定し該当のオプションを表示します。

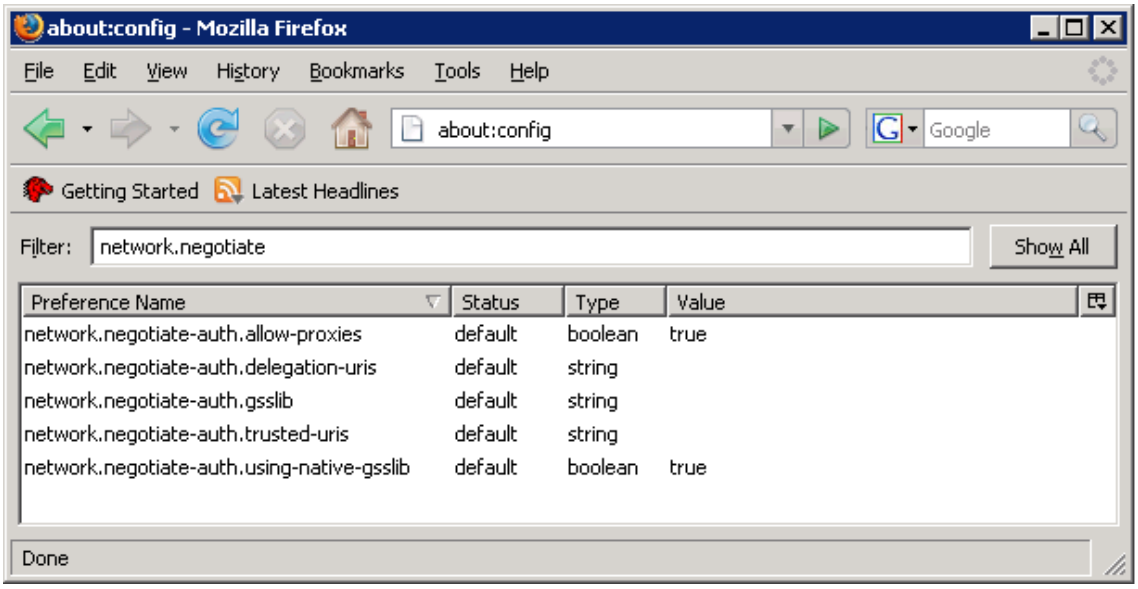


図7.4 Firefox の設定

3. **network.negotiate-auth.delegation-uris** をダブルクリックし、**Enter string value** ダイアログで、SPNEGO ネゴシエーションの URI を入力します。URI は URI の一部**http://** や **testserver** でも、全部 (例：**http://testserver.jboss.org**) でも入力することもできます。

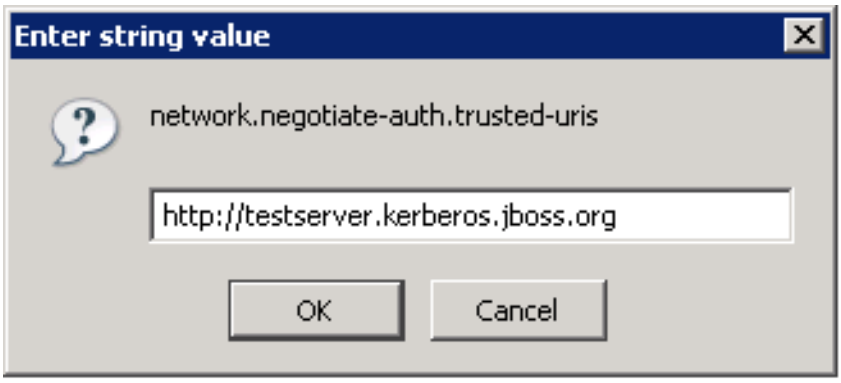
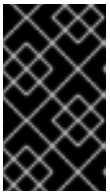


図7.5 Firefox の設定



重要

network.negotiate-auth.delegation-uris オプションは ユーザー認証の委譲先の URI を指定します。JBoss Negotiation のこのバージョンでは、委譲には対応していません。

URI が、**Value** カラムに表示され、Firefox が JBoss インストール設定を信頼し、SPNEGO ネゴシエーションを実施するようになります。**Basic Negotiation** サーブレットを使い Negotiation をテストするようにしてください (「Basic Negotiation」 参照)。

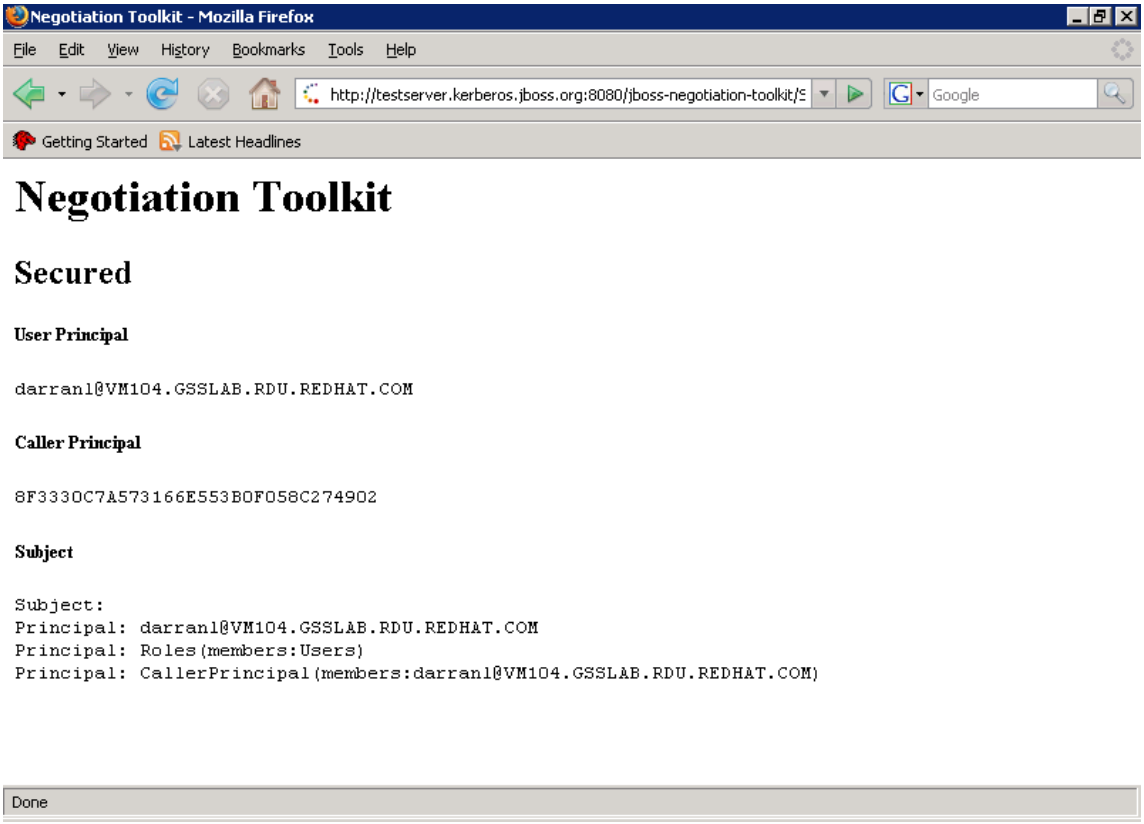


図7.6 Firefox Negotiation Toolkit

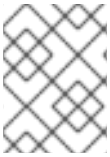
第8章 NEGOTIATION TOOLKIT

Negotiation Toolkit は、利用者のアプリケーションで設定のテストを行わないでいいように、SPNEGO 設定の検証を行う Web アプリケーションです。**jboss-negotiation-toolkit.war** ファイルは 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 から入手できます。ファイルを***$JBOSSHOME*/server/*$PROFILE*/deploy** ディレクトリにコピーし Negotiation Toolkit をデプロイします。

このツールキットは、認証システム名が**SPNEGO** で、アプリケーションのセキュリティドメインも **SPNEGO** と、仮定しています。いずれかが別の名前の場合、Web アプリケーションを展開アーカイブとしてデプロイし、 **web.xml** と **jboss-web.xml** を変更してください。

- **WEB-INF/web.xml** ファイルで、**auth-method** の認証システムキーをアップデートします (**<auth-method>SPNEGO</auth-method>**)。
- **WEB-INF/jboss-web.xml** ファイルで、**security-domain** のセキュリティドメイン名をアップデートします(**<security-domain>SPNEGO</security-domain>**)。

デプロイ後、http://testserver.kerberos.jboss.org:8080/jboss-negotiation-toolkit の Negotiation Toolkit の Web アプリケーションにアクセスします。



注記

DNS を「アプリケーションのセキュリティドメインの定義」 の 前提条件：DNS 設定 で説明されているように設定してください。

8.1. 最初のページ

Negotiation Toolkit のメインページには、SPNEGO 認証メカニズムのテストを行うツールキットユーティリティへのリンクが含まれています。上から下にリンクをしていくよう推奨しています。

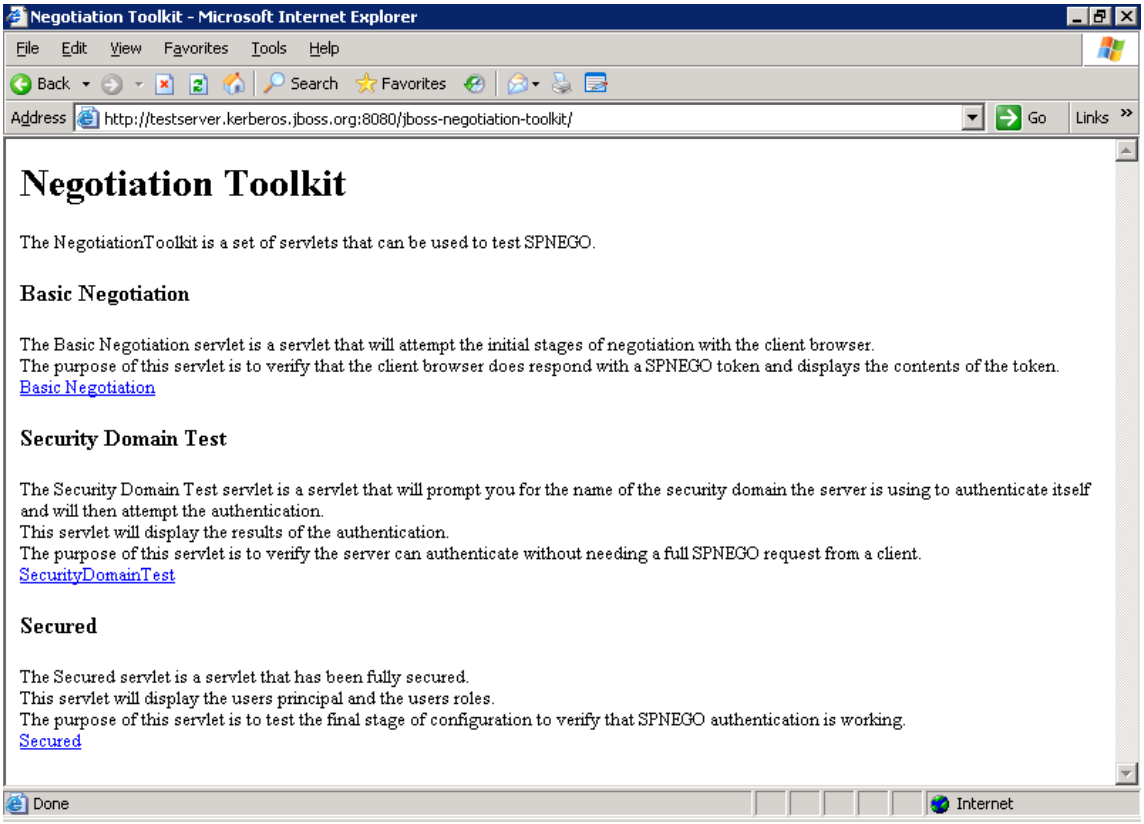


図8.1 Negotiation Toolkit の最初のページ

注記

ツールキットのテストにはアプリケーションサーバーや KDC との通信を行うため、Negotiation Toolkit を利用する前にアプリケーションサーバーのインストールが完了しているようにしてください。

8.2. BASIC NEGOTIATION

Basic Negotiation サーブレットは、Web ブラウザーがアプリケーションサーバーを信頼するかをテストします。Web ブラウザーにネゴシエーションのプロンプトを出し、アプリケーションサーバーがネゴシエーショントークンを受け取ったか確認します。

Web ブラウザーがネゴシエーショントークンの送信に失敗した場合、サーブレットは 図8.2「Basic Negotiation に失敗」 のような Web ページを表示します。

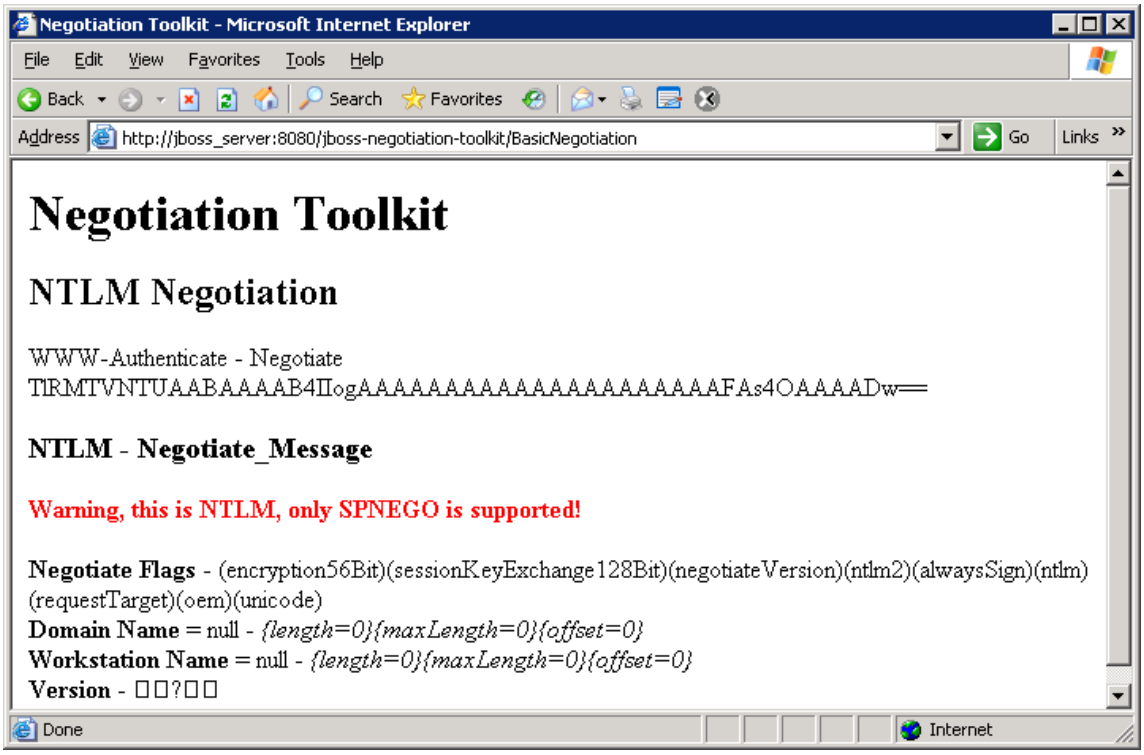


図8.2 Basic Negotiation に失敗

Web ブラウザーがネゴシエーショントークンを正常に送信した場合、サーブレットは 図8.3「Basic Negotiation に成功」のような Web ページを表示します。

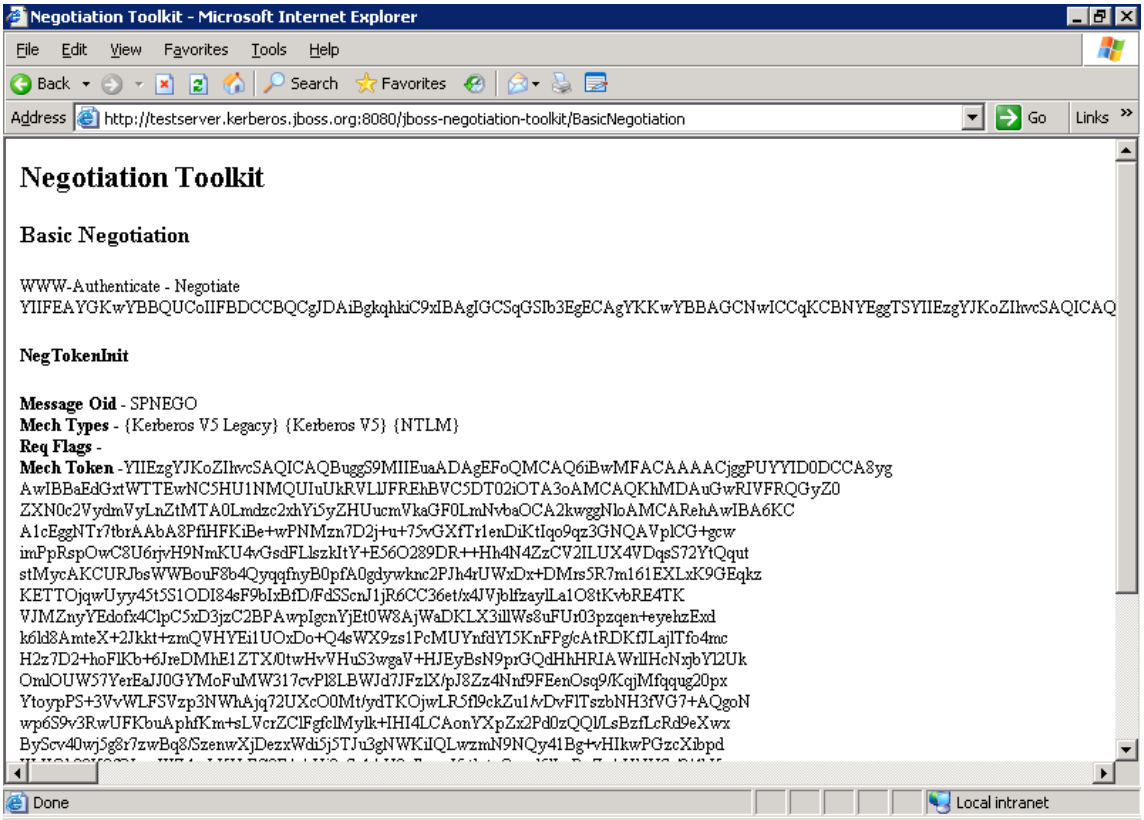


図8.3 Basic Negotiation に成功

Web ページは、ネゴシエーショントークン内に含まれる情報の一部を表示します。

8.3. SECURITY DOMAIN TEST

Security Domain Test は、アプリケーションサーバーがセキュリティドメインを使い KDC に対する認証が可能かテストします。

まず、サーブレットは、セキュリティドメイン名を入力するよう促します (本ガイド全体で、ドメインは **host** を使っており、ページは 図8.4「Security Domain Test」 で表示されているようになります)。

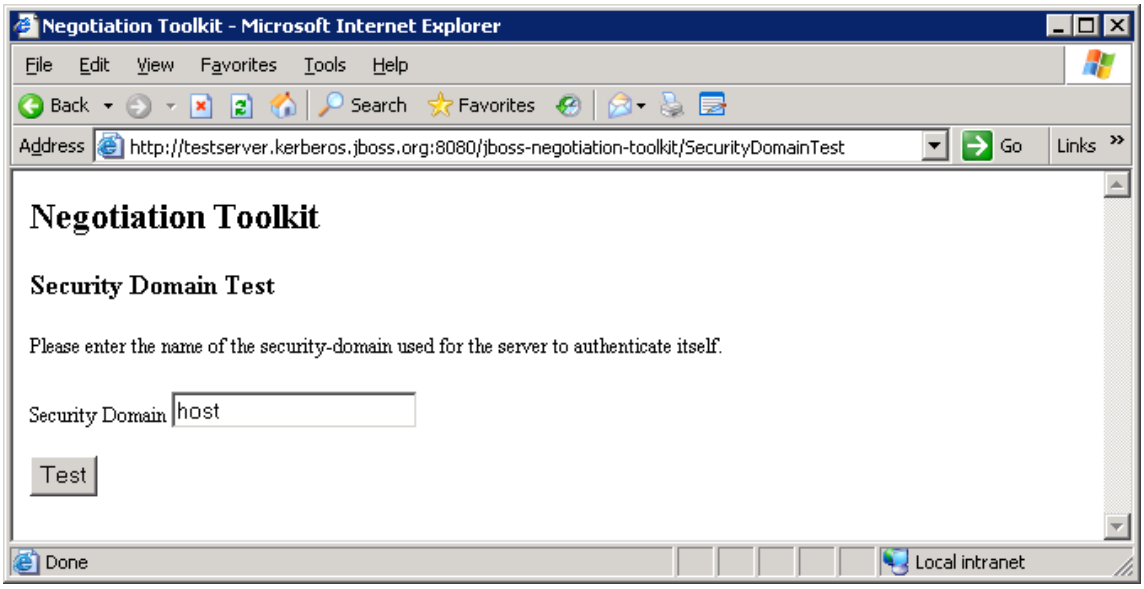


図8.4 Security Domain Test

サーブレットが正常に認証を確立すると、図8.5「Security Domain Test - Authenticated」 のようなページが表示されます。

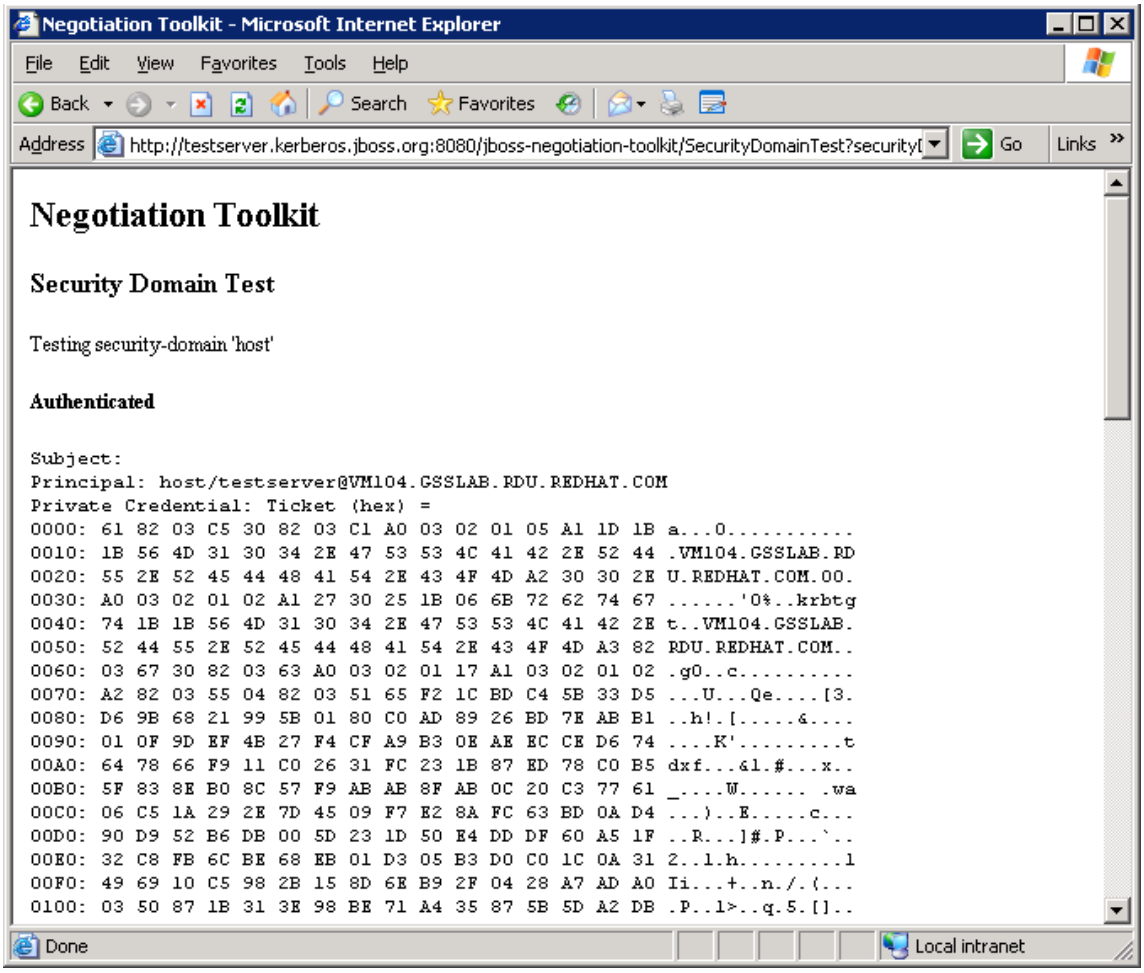


図8.5 Security Domain Test - Authenticated

8.4. SECURED

SPNEGO の完全認証を求めるように、**Secured** サーブレットを設定します。サーブレットが図 8.6「Secured」 のようなページを返した場合、実行は正常で SPNEGO 認証が正しく設定されていることになります。

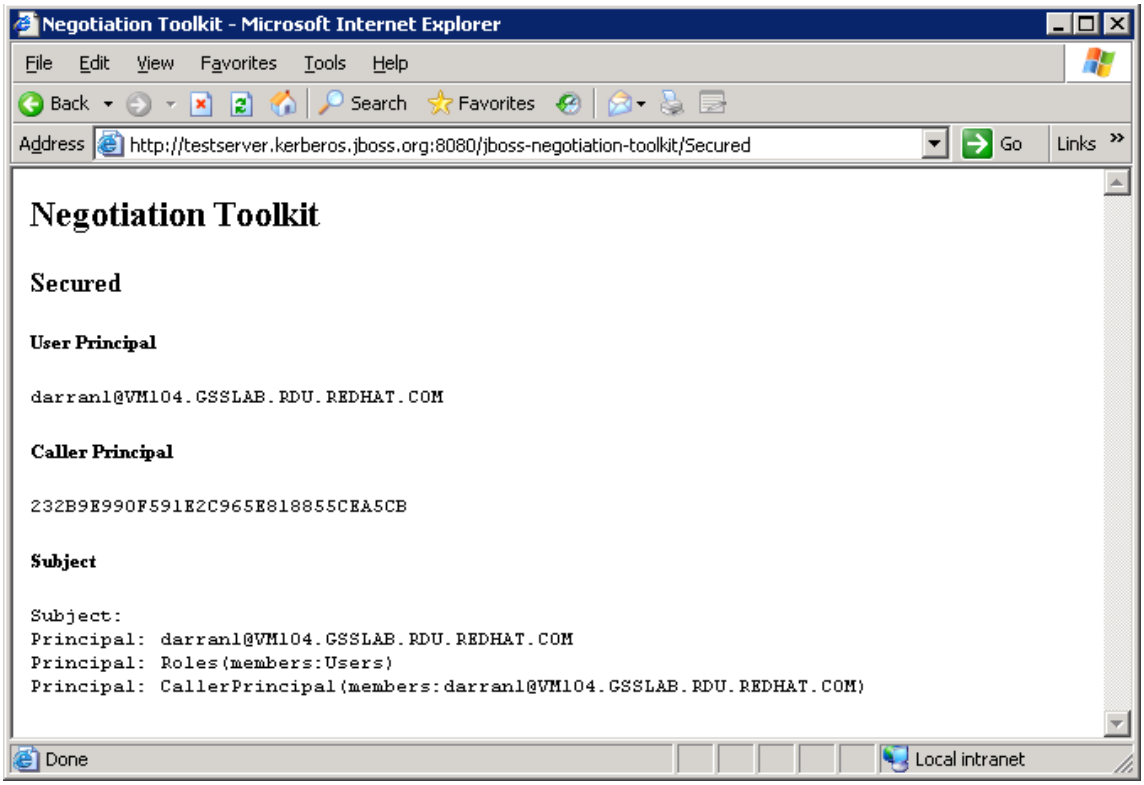

Negotiation Toolkit - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://testserver.kerberos.jboss.org:8080/jboss-negotiation-toolkit/Secured
Go Links
Negotiation Toolkit
Secured
User Principal
darran1@VM104.GSSLAB.RDU.REDHAT.COM
Caller Principal
232B9E990F591E2C965E818855CEA5CB
Subject
Subject:
Principal: darran1@VM104.GSSLAB.RDU.REDHAT.COM
Principal: Roles(members:Users)
Principal: CallerPrincipal(members:darran1@VM104.GSSLAB.RDU.REDHAT.COM)
Done
Local intranet

図8.6 Secured

第9章 WEB アプリケーションの設定

JBoss Negotiation をご利用中のサーバーに設定し、FreeIPA あるいは Active Directory への接続設定が終わると、JBoss Negotiation 認証を使うために Web アプリケーションの設定が必要になります。

JBoss Negotiation 認証を使うために Web アプリケーションを設定するには、以下を行います。

1. SPNEGO セキュリティドメインを **WEB-INF/jboss-web.xml** ファイルに追加します。

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

2. SPNEGO 認証を使うには、**login-config.xml** ファイルを設定します。

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

auth-method は、認証システムで使用するキーをマッピングします。

付録A 高度な LDAP ログインモジュール：完全 LDAP 認証

JBoss Negotiation プロジェクトには、LDAP ロール検索要件を処理する AdvancedLdapLoginModule が含まれています。

AdvancedLdapLoginModule は、LdapExtLoginModule に基づいています。SPNEGOLoginModule とともに、チェーン設定内でこのモジュールを利用することで、GSSAPI 認証が LDAP 経由で認証を行えるようになります (「ロールマッピング」 を参照)。あるいは、LDAP を使った完全認証にこのモジュールを利用可能です。また、必要であれば認証あるいはロール検索、ユーザー検索を省略するように設定することができます。

A.1. 設定

新規ログインモジュールの完全修飾クラス名は **`org.jboss.security.negotiation.AdvancedLdapLoginModule`** です。

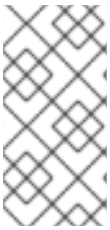
警告

ベータリリースでは、クラス名は **`org.jboss.security.negotiation.spnego.AdvancedLdapLoginModule`** でした。今でもログインモジュールは、この名前で利用できますが、廃止予定で今後のリリースでは削除されます。

AdvancedLDAPLoginModule はパスワードスタッキングに対応しています。このモジュールを他のログインモジュールと利用したい場合は、password-stacking プロパティが **`useFirstPass`** に設定されているようにしてください。

A.1.1. 初期の LDAP コンテキストを定義

まずユーザー認証情報を定義する必要があります。この情報はInitialLdapContextを取得し、ユーザーとユーザーロールの検索に利用されます。



注記

ログインモジュールがあれば、ユーザー名と認証情報を使うか、すでに認証済みのユーザーを GSSPI を使うことで、この InitialLdapContext を取得することができます。ここでは、ユーザー認証をつかっています。GSSAPI での設定については、「ロールマッピング」 を参照してください。

ユーザー名とパスワードで認証する場合、これから提示する内容により、以下の設定を定義します。

bindDN

LDAP サーバーにバインドするために利用する DN を定義します。定義済みの baseCtxDN および rolesCtxDN への読み取り／検索の権限を持つ DN です。

bindCredential

bindDN パスワードを定義します。jaasSecurityDomain を指定している場合は、このパスワードは暗号化可能です。

jaasSecurityDomain

jaasSecurityDomain の JMX ObjectNameを定義します。これは、java.naming.security.principal を復号化する際に利用する jaasSecurityDomain です。このドメインの JaasSecurityDomain#encrypt64(byte[]) メソッドは、暗号化されたパスワードを返します。org.jboss.security.plugins.PBEUtils を使い、暗号化形式を生成することも可能です。

A.1.2. DN 検索の定義

モジュールが LDAP 初期コンテキストを作成した後、提供済みのユーザー名をとり、ユーザー DN を検索します。検索プロパティを定義するには、以下のプロパティを提示します。

baseCtxDN

コンテキストの固定 DN を定義しユーザーロールを検索します。これは、実際のロールが実際に置かれている場所の識別名 (DN) ではなく、ユーザーロールが含まれているオブジェクトが置かれている場所の DN という点を考慮します (つまり、Active Directory では、ユーザーアカウントを持つ DN ということになります)。

baseFilter

認証ユーザーのコンテキストの場所を突き止めるために使う検索フィルターを定義します。ログインモジュールのコールバックから取得した通りの、入力ユーザー名／ユーザー DN は、**{0}** 表現を代入します。この代入動作は、標準のDirContext?.search(Name, String, Object[],SearchControls?cons) メソッドから来ています。一般的な検索フィルターの例は、**(uid={0})**です。

searchTimeLimit

ユーザーとロール検索のタイムアウト時間をミリ秒で定義します (デフォルトは 10000 で、10 秒となっています)。

注記

ユーザー DN 検索を無効にするには、**baseCtxDN** プロパティを省略します。提示したユーザー名をこのログインモジュールで DN として使います。

A.1.3. ユーザー認証

注記

LDAP ログインモジュールが最初のログインモジュールでなく、以前のログインモジュールがすでにユーザー認証を済ましている場合、ユーザー認証は省略されます。

今までにログインモジュールでユーザー認証を行なっていない場合、この手順で、ユーザー DN 検索と提示した認証情報からのユーザー DN を使い、新たな InitialLdapContext を作成しユーザー DN と認証情報の組み合わせが有効でなるか確認します。

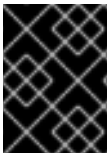
ユーザー認証は、以下のプロパティを定義してください。

allowEmptyPassword

空の (length==0) パスワードが LDAP サーバーに渡された場合。LDAP サーバーは空のパスワードを無名ログインとして処理します。このプロパティを **false** に設定し、空のパスワードを受け付けないようにするか、**true** に設定し、LDAP サーバーが空のパスワードを認証できるようにします (デフォルトは、**false**)。

A.1.4. ロール検索の定義

LDAP ログインモジュールは、LDAP サーバーに対する特定ユーザーやロール検索を定義するプロパティを渡します。



重要

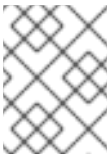
以下のロール検索設定は、LdapExtLoginModule 設定に似ていますが、反復により DN 内にリストされているロールも検索します。

rolesCtxDN

コンテキストの固定 DN でユーザーロールを検索します。これは、実際のロールが実際に置かれている場所の識別名 (DN) ではなく、ユーザーロールが含まれているオブジェクトが置かれている場所の DN という点を考慮します (例：Active Directory では、ユーザーアカウントを持つ DN ということになります)。

roleFilter

認証済みユーザーに紐付けられたロールの場所をつきとめるために利用する検索フィルターを定義します。ログインモジュールのコールバックから取得した入力ユーザ名／userDN はフィルター定義で **{0}** 表現を代入します。認証済み userDN は、フィルター定義で **{1}** を代入します。入力したユーザ名と一致する検索フィルターの例は、**(member={0})** です。もう 1 つは、認証済みの userDN に一致する場合で、 **(member={1})** となっています。



注記

roleFilter 属性を飛ばすと、ロール検索は UserDN を roleAttributeID 値を取得する DN として利用します。

roleAttributeID

ロール名に該当するコンテキストのロール属性を定義します。roleAttributeIsDN プロパティが **true** に設定されている場合、このプロパティは、roleNameAttributeID 属性に対しクエリを行うコンテキストの DN となります。roleAttributeIsDN プロパティが **false** に設定されている場合、このプロパティは、ロール名の属性名となります。

roleAttributeIsDN

ロール属性がロールオブジェクトあるいはロール名の完全な識別名を含むか定義します。 **false**の場合、ロール名はユーザーのロール属性の値から取得します。**true**の場合、ロール属性はロールオブジェクトの識別名を表します。ロール名は、該当オブジェクトのroleNameAttributeId 属性の値から取得します。特定のディレクトリスキーマでは (Microsoft Active Directory など)、ユーザーオブジェクトのロール (グループ) 属性は、簡易名ではなく DN としてロールオブジェクトに保存されます。このような場合、このプロパティは **true** に設定します。このプロパティのデフォルト値は**false** です。

roleNameAttributeID

ロール名に該当するコンテキストのロール属性を定義します。roleAttributeIsDN プロパティが **`true`** に設定されている場合、このプロパティを使いロールオブジェクトの属性を検索します。また、roleAttributeIsDN property が **`false`** に設定されている場合、このプロパティは無視されます。

recurseRules

再帰ロール検索を有効にします。ログインモジュールは、すでに追加済みのロールをトラッキングし、循環参照を処理します。

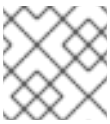
searchScope

検索範囲を以下のいずれかに設定することができます (デフォルト値は**SUBTREE_SCOPE**)。

- OBJECT_SCOPE - 指定のロールコンテキストのみを検索します。
- ONELEVEL_SCOPE - 指定のロールコンテキストを直接検索します。
- SUBTREE_SCOPE - ロールコンテキストが DirContext? でない場合、オブジェクトのみを検索します。ロールコンテキストが DirContext? の場合、サブツリーは、指定のオブジェクトにルートを置き、そのオブジェクト自体を検索します。

searchTimeLimit

ユーザーとロール検索のタイムアウト時間をミリ秒で定義します (デフォルトは 10000 で、10 秒となっています)。



注記

これらの検索は両方、同じ searchTimeLimit 設定を使います。

A.2. 完全 LDAP 認証の例

以下の設定例は、Active Directory および FreeIPA 向けに AdvancedLdapLoginModule を使った完全 LDAP 認証について示しています。この設定は、SPNEGOLoginModule で識別される名称であるため、baseFilter 属性が違います。

bindAuthentication、jaasSecurityDomain、java.naming.provider.url オプションでは、このログインモジュールが LDAP へ接続する方法、認証が行われる方法を設定します。

baseCtxDN オプションは、ユーザーを検索開始する DN で、この例にある baseFilter 属性は Active Directory 上では**`sAMAccountName`** 属性を使い、FreeIPA 上では **`uid`** 属性を使い、ユーザーを検索します。

memberOf 属性は、ユーザーから直接読み取るため、rolesCtxDN あるいは roleFilter プロパティを指定する必要はありません。roleAttributeID オプションに定義している属性は、ユーザーから直接読み取ります。

roleAttributeIsDN オプションは、この値が DN であることを指定するため、グループオブジェクトがリトリーブされます。roleNameAttributeID オプションは、**`cn`** 属性をグループから読み取るよう指定します。ログインモジュールは、このロールを返します。

recurseRoles が **`true`** に設定されているため、検索で発見したグループからの DN を使いこのプロセスを繰り返します。そのため、グループが **`memberOf`** 属性で設定されている場合、これは再帰的に利用されすべてのロールを検索していきます。

A.2.1. Active Directory 向けの完全 LDAP 認証

以下は、Active Directory ドメインからダンプした ldiff を抜粋したものです。

```
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
```

以下の設定では、ユーザー名とパスワードを認証プロセスに提示する必要があります。

```
<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. Free IPA 向けの完全 LDAP 認証

以下は、FreeIPA ドメインからダンプした ldiff を抜粋したものです。

```
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
```

以下の設定では、ユーザー名とパスワードを認証プロセスに提示する必要があります。

```
<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>
```

付録B 改訂履歴

改訂 5.1.2-2.400 **2013-10-31** **Rüdiger Landmann**
Rebuild with publican 4.0.0

改訂 5.1.2-2 **2012-07-18** **Anthony Towns**
Rebuild for Publican 3.0

改訂 5.1.2-100 **Thu 8 December 2011** **Russell Dickenson**
JBoss Enterprise Application Platform 5.1.2 GAに対する変更を追加。本ガイド文書の変更に関する情報は、『 リリースノート 5.1.2』を参照してください。