Configuring Integrated Windows Authentication for IdentityIQ with SPNEGO

Integrating IdentityIQ with Active Directory for Single Sign On (SSO) via SPNEGO

This document provides instructions and best practices for integrating IdentityIQ with an Active Directory domain for Single-Sign-On (SSO) and Integrated Windows Authentication (IWA) using the SPNEGO package from SourceForge.

Document Revision History

Revision Date	Written/Edited By	Comments
May 18, 2014	Adam E. Hampton	Initial Creation
May 19, 2014	Adam E. Hampton	Merged in Christian Cairney's JBoss configuration details
May 20, 2014	Adam E. Hampton	Merged in Palle Hansen and Yann Kernin's review comments
May 23, 2014	Adam E. Hampton	Merged in Jennifer Mitchell's language recommendations
June 6, 2014	Adam E. Hampton	Merged in Paul Wheeler's changes

Table of Contents

Introduction

Microsoft's Windows and Active Directory domains represent a widely used form of user authentication used in enterprise environments today. Workers all over the world start their day by pressing "ctrl-alt-del" and logging into their Windows based workstations and networks. When they do this, they are often authenticating to an Active Directory domain and establishing a session with the domain controlling the network.

Enterprises commonly want to reduce the number of times users need to log in or authenticate to internal tools. When a user has already authenticated to their domain, there is little reason to have them "re-authenticate" to a specific internally hosted software application. Setting up software to rely on the Active Directory session for a logged-in user is called configuring "Integrated Windows Authentication", or IWA for short.

Some documents define IWA as a specific form of integration that leverages SSO though Microsoft's IIS Web Server. This narrower definition describes fronting the Java-based application with an IIS-based .NET or ASP application to handle the Active Directory integration. In this definition the Java-based application simply reads HTTP headers containing user principal information that are "injected" into the stream by the IIS application.

Other documents describe "Windows Native Authentication" or WNA for this integration. WNA is often described as the integration process that requires an application inside a Java-based servlet container to be declared in the Kerberos Domain Controller (KDC) as a "service principal" and the servlet container to communicate via the Kerberos protocol to the domain controller by itself. Under these definitions this document describes the latter case, "WNA using Kerberos", and not the former "IWA using .NET/ASP". This document uses the terms IWA and WNA interchangeably to mean the Kerberos based approach.

There are a few common strategies for configuring IdentityIQ for Single-Sign-On (SSO) in an enterprise environment. Those strategies include:

- Integrating with a pre-existing enterprise SSO tool (Tivoli Access Manager, Site Minder, IdentityNow, etc.) for HTTP header injection into IdentityIQ
- Integrating Microsoft IIS and ARR web technologies to "Front End" IdentityIQ and HTTP-inject the user's account or Identity details into IdentityIQ's HTTP sessions
- Integrating with a Kerberos integration layer like SPNEGO to provide the user's account details

All of these involve configuring Active Directory and at least one other component outside of IdentityIQ. Configuring IWA for IdentityIQ is always an integration exercise; almost all of the configuration takes place outside of the core IdentityIQ product - in either a 3rd party system or in the servlet container housing IdentityIQ.

The goal of this document is to communicate the procedure, details, and best practices of the third strategy, namely: integrating with SPNEGO to perform Integrated Windows Authentication (IWA). Simple Protected Negotiation (hence "SPNEGO") is the protocol for negotiating the exchange of a user's "principal" information (their domain and account name) through a series of trusted and encrypted transactions over HTTP, GSS-API, and LDAP. This document describes integrating with the SPNEGO package available on SourceForge.com; this package is a GNU LGPL licensed tool that is freely distributed and freely available for use to integrate Java and J2EE based applications with Active Directory domains.

This document provides examples for configuring SPNEGO for use with IdentityIQ under Apache Tomcat. SPNEGO can also be configured for other servlet containers like JBoss and WebLogic using similar procedures. See the JBoss specific section for details related to JBoss configuration.

This document assumes that IdentityIQ is the only web application running under its servlet container. Configuring SPNEGO requires configuration settings that impact the entire servlet container and all of the web applications running inside of the container's JVM. It is common for many applications to share a common SPNEGO configuration for Active Directory domain integration. Environments using a shared servlet container for multiple web services can use this guide but must be cautious about impacting other applications configured in the servlet container; coordinate with the administrators of other applications hosted under Tomcat alongside IdentityIQ when configuring SPNEGO for the first time.

This document assumes all browser interactions are performed with Internet Explorer, which negotiates with Active Directory domains natively for IWA. Alternative browsers, such as FireFox, can be configured to work with IWA as well. The specific configuration of these alternative browsers is beyond the scope of this document. The following site provides a good reference to start with for browser specific configuration of IWA:

http://www-01.ibm.com/support/knowledgecenter/SS7JFU_6.1.0/com.ibm.websphere.express.doc/info/exp/ae/tsec_SPNEGO_config_web.html?cp=SS7JFU_6.1.0%2F1-7-9-9-0-1-5-0-2&lang=en

Sequence Diagram

With SPNEGO configured for IWA and IdentityIQ configured with an appropriate SSO rule, an installation can recognize a user's already authenticated Windows domain login as authentication for accessing IdentityIQ. The following IETF style ASCII sequence diagram illustrates the approximate sequence of events when a domain user accesses IdentityIQ after logging in with this configuration.

```
User, Browser                  Tomcat + IdentityIQ                     Active Directory
========================================================================================
Login to Windows with
username, password
     ==================================================================>
                                                                     Validate Login
                                                                     return session token
     <==================================================================
...
Open Browser to
IdentityIQ dashboard
     ===============================>
                            No HTTP session, return
                            error 401, unauthorized
     <===============================
Browser offers more
details, including
GSS-API, SPNEGO tokens
     ===============================>
                            SPNEGO tokens received
                            and processed, check with
                            domain controller for
                            validation
                                       ===============================>
                                                                     Validate token
                                                                     return user account
                                                                     details, samAccount
                                       <===============================
                            AD samAccount details
                            received from domain controller
                            Correlate domain and samAccountName
                            to Identity cube via SSO rule, establish
                            HTTP session and process login
     <===============================
IdentityIQ dashboard presented to
user in browser
```

Those familiar with the full Kerberos sequence would expect more detail in this diagram, involving TGT and TGS tickets. This diagram is intended to simply illustrate the process here and not necessarily capture the detailed flow of packets across the network. Readers interested in more details about the Kerberos protocol are encouraged to review the following diagrams:

https://software.intel.com/sites/manageability/AMT_Implementation_and_Reference_Guide/default.htm?turl=WordDocuments%2Fintroductiontokerberosauthentication.htm

http://i.technet.microsoft.com/dynimg/IC195542.gif

Prerequisites

Before configuring an integration between a Windows domain and IdentityIQ via SPNEGO you will need the following resources to be successful. Review this checklist with your project team because coordination with servlet container administrators and Windows domain administrators is often required in large enterprise environments. Gathering the right personnel can be a challenge in large organizations.

- An IdentityIQ installation up and running with Identities populated from an authoritative source.
- The DNS name or virtual host name entry that end-users will use to browse to IdentityIQ.
- An Identity attribute configured and populated with the "samAccountName" or the domain and "samAccountName" of the group of users you want to enable for SSO and IWA. Sites integrating with multiple AD Domains will want to have "DOMAIN\samAccountName" as an Identity attribute.
- Administrative access to an Active Directory domain for the purposes of creating a service account for Tomcat (or the servlet container in use) to communicate with the Active Directory domain. For installations integrating with multiple AD Domains, all involved domains must be configured with a proper AD Forest Trust between them.
- Administrative access to an Active Directory domain controller host for the purposes installing the Windows Support Tools (suptools.msi, suptools.cab) if they are not installed already.
- Administrative access to an Active Directory domain controller host to execute command line utilities for configuring the Kerberos integration to the domain (ktpass.exe, setspn.exe).
- Access to a Windows workstation that is not running IdentityIQ and is configured as a member of the Active Directory domain for which SSO is being configured.
- A regular user account with credentials for the domain is required for testing the end-user SSO experience. The account must that correlate to a cube under IdentityIQ via some attribute.
- Administrative Access to the Tomcat installation to download and install the SPNEGO .jar file, configuration files and "tomcat.keytab" file into the Tomcat home directory.
- Administrative access to restart Tomcat on demand. The Tomcat server will be restarted several times during testing.
- Access to compile a test driver program in Eclipse (or any editor) and copy it over to the server running Tomcat.
- Access to install the "hello_spnego.jsp" test script and to execute it under Tomcat.
- System Administrator Access to IdentityIQ to install an SSO Rule in the system. NOTE: For environments integrating with multiple domains, the SSO Rule will require code to parse the domain component from the user name.
- Ability to change browser configurations; if using a browser other than Internet Explorer (e.g. Firefox) then there may be a piece of configuration required in the client to enable Kerberos

External Resources

Readers familiar with SPNEGO configuration from other software systems will find that this documentation closely follows the recommendations and examples published on SourceForge with the SPNEGO package and the Apache for Tomcat configuration for IWA. While progressing through this document readers are strongly encouraged to review the documentation and tutorial artifacts for SPNEGO available online.

This document attempts to expand on the examples provided in the online resources and provide an IdentityIQ centric level of additional detail where warranted. The following tutorials online should also be reviewed while working through this material:

- The SPNEGO overview:
    - http://spnego.sourceforge.net/
- The SPNEGO pre-flight checklist:
    - http://spnego.sourceforge.net/pre_flight.html
- The SPNEGO Tomcat installation guide:
    - http://spnego.sourceforge.net/spnego_tomcat.html
- The Apache Tomcat Windows Authentication How-To Documents:
    - http://tomcat.apache.org/tomcat-7.0-doc/windows-auth-howto.html

The SourceForge SPNEGO project's approaches differ slightly from the Tomcat examples. With the goal of portability to other servlet containers, this document uses the SPNEGO approach so that JBoss and other servlet containers can benefit from the best practices and examples shown here.

Setup Procedures

The procedure for configuring SPNEGO follows the outline below. The balance of this document follows the same outline and expands on each section in more detail.

1. Download the SPNEGO ".jar" file and install it into the …/[tomcat-home]/lib/ directory
2. Create the service account for Tomcat to access the domain. This is the "pre-authentication" account
3. Open up firewalls to allow Kerberos (UDP port 88) from IdentityIQ servers to Domain Controllers
4. Create the "tomcat.keytab" file for the Service Principal Name (SPN) for IdentityIQ's web service
5. Configure the service account as owner of the SPN for IdentityIQ's web service with "setspn.exe"
6. Copy the "tomcat.keytab" file to "…/[tomcat-home]/conf/ tomcat.keytab"
7. Create a "…/[tomcat-home]/conf/krb5.conf" file for your environment
8. Create a "…/[tomcat-home]/conf/login.conf" file for your environment
9. Create a HelloJDC.class test driver for your environment and execute it
10. Enable Kerberos debugging in your environment
11. Edit the "…/[tomcat-home]/conf/web.xml" file under Tomcat to enable SPNEGO
12. Create the "…/[tomcat-home]/webapps/spnegotest/hello_spnego.jsp" test application
13. Start Tomcat and test from the "spnegotest/hello_spnego.jsp" test application
14. Install the example SPNEGO SSO Rule in your IdentityIQ environment, extend as necessary
15. Test logging into Tomcat from a Windows machine on the domain as a regular user
16. Disable the debugging and logging options enabled during installation
17. Add the SPNEGO SSO Rule to your build system, optionally add the Kerberos artifacts as well

Downloading the SPNEGO Binaries

The SPNEGO binaries can be downloaded from the SPNEGO web site hosted on SourceForge:

http://spnego.sourceforge.net/

This file provides the compiled SPNEGO package, which integrates with a Kerberos domain controller. This library includes the code to encrypt and decrypt Kerberos tokens. It includes the code that inserts the user principal data into the HTTP session details of the servlet container.

At the time of writing, the current release of SPNEGO's .jar file package was "spnego-r7.jar". You are encouraged to use the latest release available from SPNEGO's download site. Using the latest release of Apache Tomcat and JDK/JVM are also recommended for general security and to benefit from the latest bug fixes in each package.

Once downloaded, place a copy of the .jar file in the "…/[tomcat-home]/lib/" directory under Tomcat. Note that this is *not* the "…/webapps/identityiq/WEB-INF/lib" directory of IdentityIQ.

Example path for Windows installations:

```
C:\SailPoint\tomcat\lib\spnego-r7.jar
```

Example path for Unix/Linux installations:

```
/opt/sailpoint/tomcat/lib/spnego-r7.jar
```

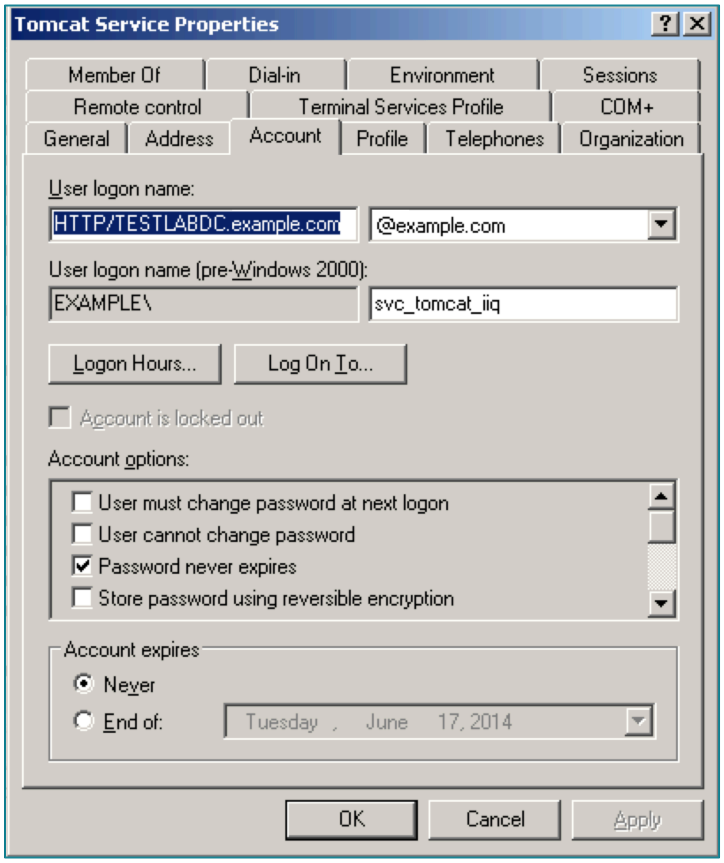
Configuring a Pre-Authorization Service Account

When SPNEGO negotiates with Active Directory to identify a user it must communicate with the Active Directory domain, and with the Domain Controller hosts. A service account is required to facilitate this communication. Conceptually this means Tomcat and the IdentityIQ instance under it need a service account in Active Directory to communicate with Active Directory to ask it to verify the details of the user establishing a session through their browser.

The Active Directory administrators for your site will need to be involved here. They will need to create a user account for Tomcat and IdentityIQ. This account is called the "pre-authentication" account in the SPNEGO documentation. Note the domain name, user name (samAccountName) and password for the service account.

Guidelines for Account Creation:

- This guide recommends a service account named "svc_tomcat_iiq" to clearly communicate to casual viewers that this service accounts supports Tomcat and IdentityIQ. The naming standard for service accounts in your enterprise may vary and you may need to change this for your particular installation site. Some sites use "_s" suffixes to denote service accounts others have more obscure patterns or no pattern at all. The remainder of this document will simply refer to the service account as "svc_tomcat_iiq". The example domain used in this document is simply a domain named "EXAMPLE". Often the ".com" is appended to domain names so "EXAMPLE.COM" can be seen in some configuration files. Using traditional Windows "DOMAIN\samAccountName" format the service account shown in this document is "EXAMPLE\svc_tomcat_iiq".
- The First Name, Last Name, and Display Name for this service account can be configured in any way you wish that conforms with naming standards for your site. The examples shown here use a first name of "Tomcat" and a last name of "Service" and a display name of "Tomcat Service".
- This account must be created with read access to the domain (LDAP tree) for users who will use IWA.
- This account must be created with a password set to never expire. Expiring passwords for service accounts are not recommended and changing this password requires a change to the "web.xml" file under Tomcat to reflect the new password. If an expiring password is required then negotiate for an exception to site policy.
- This account does not require RDP or direct login access to computers in the domain but it is recommended to start off with such access enabled so you can test and verify that the user name and password work by simply logging in with them to a workstation.
- The administrator will need to give you this account's password in clear text; this account's credentials will live in a configuration file under the Tomcat directory tree. There is no strategy for encrypting this account's password – encrypted passwords are not supported by the SPNEGO package's configuration. Given that this account is a read-only account with limited security rights (basically no security rights) little risk is introduced here.



Tomcat Service Properties

Member Of | Dial-in | Environment | Sessions | Remote control | Terminal Services Profile | COM+ | General | Address | Account | Profile | Telephones | Organization

User logon name:
HTTP/TESTLABDC.example.com @example.com

User logon name (pre-Windows 2000):
EXAMPLE\ svc_tomcat_iiq

Logon Hours... Log On To...

☐ Account is locked out

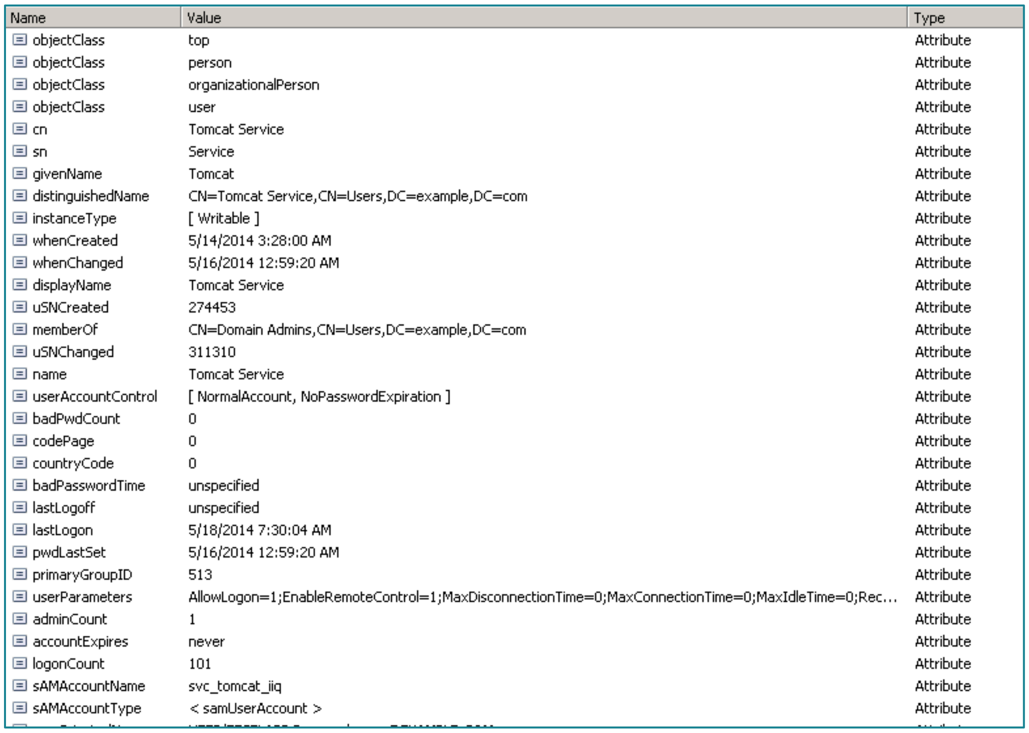
Account options:
- ☐ User must change password at next logon
- ☐ User cannot change password
- ☑ Password never expires
- ☐ Store password using reversible encryption

Account expires
- ● Never
- ○ End of: Tuesday , June 17, 2014

OK Cancel Apply

Figure 1: Example of service account create under the AD "Users and Computers" tool

After the account is created it can be seen via any LDAP browsing tool.



Name	Value	Type
objectClass	top	Attribute
objectClass	person	Attribute
objectClass	organizationalPerson	Attribute
objectClass	user	Attribute
cn	Tomcat Service	Attribute
sn	Service	Attribute
givenName	Tomcat	Attribute
distinguishedName	CN=Tomcat Service,CN=Users,DC=example,DC=com	Attribute
instanceType	[Writable]	Attribute
whenCreated	5/14/2014 3:28:00 AM	Attribute
whenChanged	5/16/2014 12:59:20 AM	Attribute
displayName	Tomcat Service	Attribute
uSNCreated	274453	Attribute
memberOf	CN=Domain Admins,CN=Users,DC=example,DC=com	Attribute
uSNChanged	311310	Attribute
name	Tomcat Service	Attribute
userAccountControl	[NormalAccount, NoPasswordExpiration]	Attribute
badPwdCount	0	Attribute
codePage	0	Attribute
countryCode	0	Attribute
badPasswordTime	unspecified	Attribute
lastLogoff	unspecified	Attribute
lastLogon	5/18/2014 7:30:04 AM	Attribute
pwdLastSet	5/16/2014 12:59:20 AM	Attribute
primaryGroupID	513	Attribute
userParameters	AllowLogon=1;EnableRemoteControl=1;MaxDisconnectionTime=0;MaxConnectionTime=0;MaxIdleTime=0;Rec...	Attribute
adminCount	1	Attribute
accountExpires	never	Attribute
logonCount	101	Attribute
sAMAccountName	svc_tomcat_iiq	Attribute
sAMAccountType	< samUserAccount >	Attribute

Figure 2: Example service account as seen in regular LDAP browsing tool

Creating a Key-Tab file and Service Principal Name

Next, create a "keytab" file for your Tomcat service in the domain. The "keytab" file associates your web service (like http://identityiq.example.com) with a Service Principal in your domain. This allows the domain to track permissions on which accounts can access this service and which service accounts can access the domain to verify access to this service. This file is generated by the Windows Support Tools package's "ktpass.exe" tool. If your server or domain controller does not have this package installed then you will have to download and install it from the Microsoft download site. This package is a freely available package for Windows servers. This step will involve the participation of your Active Directory administrator to run the "ktpass.exe" utility if you do not have the privileges to do so.

The "keytab" file in our example will be called "tomcat.keytab". Before you can create the "keytab" file for your installation you will need the following pieces of information for your installation:

- The DNS name for the IdentityIQ web service in your organization. This document uses "TESTLABDC.example.com" for its DNS examples.
- The name to give to your Service Principal. This is a name that describes the IdentityIQ web service in your enterprise. The example show in this document is "HTTP/TESTLABDC.example.com@EXAMPLE.COM". This is generally of the format: "HTTP/" as a leading string, followed by the DNS name of the IdentityIQ system with the fully qualified domain name, followed by "@" and the name of the domain in all capital letters. See "Identifying your Service Principal Name" in the Troubleshooting section for more details on this topic.

- The password you want to assign to the service principal. The recommended best practice is to set this to the same password as your service account.

Once you have these details assembled you can run the “ktpass.exe” command line utility to create your “keytab” file. Here is an example of invoking the command (this is one line of command):

```
ktpass /out c:\tomcat\conf\tomcat.keytab /mapuser svc_tomcat_iiq@EXAMPLE.COM /princ
HTTP/TESTLABDC.example.com@EXAMPLE.COM /pass Password1 /ptype krb5_nt_principal /kvno 0
```

Expanded for better readability, here is each argument separated out line by line:

```
ktpass /out c:\tomcat\conf\tomcat.keytab
        /mapuser svc_tomcat_iiq@EXAMPLE.COM
        /princ HTTP/TESTLABDC.example.com@EXAMPLE.COM
        /pass Password1
        /ptype krb5_nt_principal
        /kvno 0
```

Each of the arguments to this command are described below:

- The “/out” output argument specifies where the “tomcat.keytab” file is created. You may need to modify this path to match your Tomcat installation path. Or you can create the file on one server and then copy it to your Tomcat server later.
- The “/mapuser” argument tells the keytab file what domain user to map to the Service Principal. You want to specify your service account in [samAccountName@DOMAIN.SUFFIX] format here.
- The “/princ” is the name of the Service Principal for which you are creating the keytab file. The format is general [PROTOCOL/HOST.fully.qualified.domain@DOMAIN.SUFFIX]. The example here shows HTTP, HTTPS is also common and more useful in production environments. Use HTTPS if you already have SSL/TLS security certificates cut for your service name.
- The “/pass” argument assigns a password to the Service Principal.
- The “/ptype” and “/kvno” are static arguments for this document’s purposes and should be entered in the command as specified here.

This produces a configuration file called “tomcat.keytab”. If “ktpass.exe” is run from a different server from your Tomcat installation then you will need to copy the “tomcat.keytab” file your Tomcat server under /conf/.

For Unix installations make sure the “tomcat.keytab” is owned by your Tomcat service account user and is readable by the account in the directory. Make sure “root” isn’t the owner and the only account with access to this file.

It is recommended to keep a copy of this “tomcat.keytab” file in your source control system after creation.

Registering SPNs to the Service Account

The next step of the process is to configure the Tomcat service account (“svc_tomcat_iiq”) as the owner of the Service Principal Name (SPN) for which you have made a keytab file. To do this, use the “setspn.exe” utility from the Windows Support Tools. If your server or domain controller does not have this package installed then you will have to download and install it from the Microsoft download site. This package is a freely available package for Windows servers. This step will involve the participation of your Active Directory administrator to run the “setspn.exe” utility if you do not have the privileges to do so.

You can think of the “setspn.exe” utility as a tool to assign ownership of an SPN to a service account. This process of assigning ownership is called “registering” the SPN with the service account. This entitles the service account to verify the login sessions of end users requesting access to the SPNs that it owns. The command line arguments include the SPN and the service account to own the SPN.

The following example assigns ownership of the IdentityIQ instance running on the TESTLABDC host to the “svc_tomcat_iiq” service account. Note this section is case sensitive and may be in your environment as well. Two ownership calls are made in the example – one with a port number suffix and one without. Be careful to not assign ownership of an SPN to multiple service accounts as this creates conflicts. Be certain to modify this command with the name of your service account when executing:

```
setspn -a http/testlabdc                  svc_tomcat_iiq
setspn -a HTTP/TESTLABDC.example.com      svc_tomcat_iiq
setspn -a HTTP/TESTLABDC.example.com:8080 svc_tomcat_iiq
```

Once the SPN ownership is assigned, it can be reviewed and confirmed with a different argument:

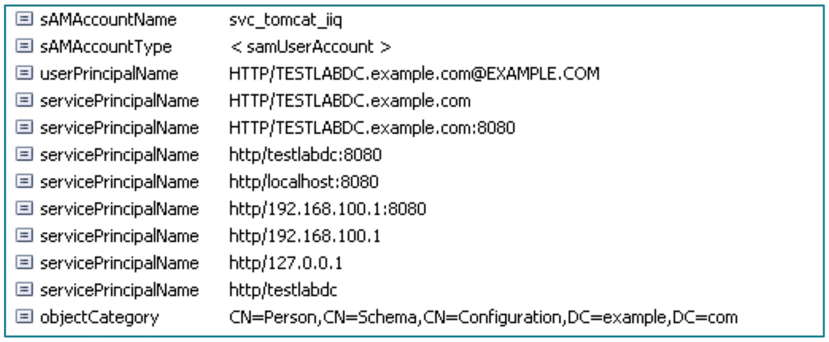
```
C:\tomcat\lib>setspn -L svc_tomcat_iiq
Registered ServicePrincipalNames for
        CN=Tomcat Service,CN=Users,DC=example,DC=com:

HTTP/TESTLABDC.example.com
HTTP/TESTLABDC.example.com:8080
http/testlabdc:8080
http/localhost:8080
http/192.168.100.1:8080
http/192.168.100.1
http/127.0.0.1
http/testlabdc
```

If you need to un-register an SPN to correct a typo or incorrect assignment you can use the “-D” option:

```
:: Example of un-registering an SPN:
setspn -D HTTP/TESTLABDC.example.com svc_tomcat_iiq
setspn -D HTTP/TESTLABDC.example.com:8080 svc_tomcat_iiq
```

The SPN registration/ownerships can also be viewed with an LDAP tool on the service account’s object in the LDAP directory:



sAMAccountName	svc_tomcat_iiq
sAMAccountType	< samUserAccount >
userPrincipalName	HTTP/TESTLABDC.example.com@EXAMPLE.COM
servicePrincipalName	HTTP/TESTLABDC.example.com
servicePrincipalName	HTTP/TESTLABDC.example.com:8080
servicePrincipalName	http/testlabdc:8080
servicePrincipalName	http/localhost:8080
servicePrincipalName	http/192.168.100.1:8080
servicePrincipalName	http/192.168.100.1
servicePrincipalName	http/127.0.0.1
servicePrincipalName	http/testlabdc
objectCategory	CN=Person,CN=Schema,CN=Configuration,DC=example,DC=com

Figure 3: LDAP tool showing SPN registrations

Creating a Kerberos Configuration File

The Kerberos configuration file defines the realms (AD domains), encryption schemes permitted, and host names and IP addresses of the Kerberos domain controller. This is a key configuration file for the integration with the Active Directory domain. This file should be called krb5.conf and should be stored in the /conf/directory of the Tomcat installation.". You will need to create this file from scratch in a text editor. Copy and paste this example or an example from the SPNEGO tutorial sites and then modify it to match your installation.

Here is an example configuration file. Each line will be covered in detail below.

```
[libdefaults]
  default_realm = EXAMPLE.COM
  default_keytab_name = FILE:c:/tomcat/conf/tomcat.keytab
  default_tkt_enctypes = aes256-cts-hmac-sha1-96,aes128-cts-hmac-sha1-96,aes128-cts,rc4-hmac
  default_tgs_enctypes = aes256-cts-hmac-sha1-96,aes128-cts-hmac-sha1-96,aes128-cts,rc4-hmac
  permitted_enctypes   = aes256-cts-hmac-sha1-96,aes128-cts-hmac-sha1-96,aes128-cts,rc4-hmac

[realms]
  EXAMPLE.COM  = {
     kdc =  192.168.56.101
     default_domain = EXAMPLE.COM
  }

[domain_realm]
  example.com = EXAMPLE.COM
  .EXAMPLE.COM = EXAMPLE.COM
```

The "krb5.conf" configuration file is broken into 3 sections: "libdefaults" which sets the defaults for the Kerberos java library, "realms" which defines the Kerberos realms and what domain controller host names or hosts and domains they map to, and the "domain_realm" map.

The "default_realm" section should name the default domain where you want to process authentication requests. You should be able to get this from your Active Directory configuration. The example above shows "EXAMPLE.COM" for this value.

The "default_keytab_name" specifies the path to the keytab file. This is the same keytab file you created earlier. This document recommends the keytab file live in the /conf/ directory under Tomcat's home directory. Note that Unix style slashes are still used in Windows context here for consistency. For a Linux/Unix example you might use "FILE:/opt/sailpoint/tomcat/conf/tomcat.keytab".

The next 3 lines ("*_enctypes") define the allowed and permitted encryption types when communicating with the Kerberos server. The examples here have higher-grade encryption specified than some of the examples on the SPNEGO tutorials. Notably the older and weaker "DES" (data encryption standard) ciphers have been omitted from the specified list. If you need to add them back then you can add: "des3-cbc-sha1,des-cbc-md5,des-cbc-crc". It is recommended that you configure this file with the strongest grade encryption supported by the target Active Directory domain. Check with the local Active Directory administrator for more details.

The "realms" section defines the mapping of domain controller IP addresses or host names and domain names to Kerberos realms. You will need to enter your DC's IP address (or the DNS round robin name for your DC hosts) and the name of the AD domain on these two lines. For sites configuring IWA with multiple domain controllers there will need to be separate items for each domain that SPNEGO will communicate with.

NOTE: Configuring SPNEGO to negotiate with multiple domains is as advanced topic not covered in this guide. Readers interested in this topic are encouraged to review the following online examples and discussion:

http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp?topic=%2Fcom.ibm.itame.doc_6.0%2Frev%2Fam60_webseal_admin209.htm

http://stackoverflow.com/questions/13746669/spnego-cross-domain-configuration

The final section, "domain_realm" defines mappings between host name suffixes and Kerberos domains. You will generally want 2 lines for each domain you integrate with. The left hand side of the equals signs are the host-name components, and the right hand sign of the equals sign is the Kerberos domain name, which maps to a KDC IP address and windows domain specified above in the "realms" section. You can use this for limited domain aliasing.

For Unix installations make sure the "krb5.conf" is owned by your Tomcat service account user and is readable by the account in the directory. Make sure "root" isn't the owner and the only account with access to this file.

After testing it is recommended that a copy of this file be stored in your project's source code management system.

Creating a SPNEGO Configuration File

The SPNEGO configuration file is called "login.conf", and like the other files in this guide, it is placed in the /conf/ directory of the Tomcat installation. This file tells the SPNEGO modules what Kerberos implementation to include in the JVM and what service account to authenticate with to lookup users. Here is an example "login.conf" file that you can use as a template.

```
// Notes:
// The login.conf file accepts comments with a double-slash prefix as shown.
// The syntax of this file can be confusing: each section separated by curly
// braces {} is delimited with a semicolon following the closing curly brace
// The list of properties inside curly braces is delimited with new lines
// and the _last_ item in the list needs to be terminated with a semi-colon.
// This is confusing to some engineers who's instincts are to terminate every
// line with a semi-colon like the C or Java programming languages expect.
// You can use the HelloKDC.java test program from the SPNEGO web site at
//    http://spnego.sourceforge.net/pre_flight.html
// to verify the syntax of your login.conf file.
// --Adam E. Hampton, SailPoint Technologies Inc., 20140518.

spnego-client {
        com.sun.security.auth.module.Krb5LoginModule required;
};

spnego-server {
        com.sun.security.auth.module.Krb5LoginModule required
        storeKey=true
        principal="HTTP/TESTLABDC.example.com@EXAMPLE.COM"
        isInitiator=true
        useKeyTab=true
        keyTab="c:/tomcat/conf/tomcat.keytab";
};
```

NOTE: The required placement of semicolon ";" characters in this configuration file is as shown above: semicolons should appear just before and after each close curly brace "}". Java developers will instinctively put semicolons at the end of every line of this file, which will result in confusing syntax errors that are hard to track down. Edit with caution.

The first section, "spnego-client", tells the SPNEGO client module what Kerberos implementation to use. This section can be copied verbatim into your config file.

The second section, "spnego-server", tells the SPNEGO server which SPN it should handle and where the keyTab file is. The "principal" and "keyTab" lines should be edited to match your environment.

NOTE: This example only applies to installations using Sun/Oracle's JVM implementation for Kerberos. Installations using IBM's or JRockit's JVM technology will need to consult other documentation for which Kerberos modules to include.

For Unix installations make sure the "login.conf" is owned by your Tomcat service account user and is readable by the account in the directory. Make sure "root" isn't the owner and the only account with access to this file.

After testing it is recommended that a copy of this file be stored in your project's source code management system.

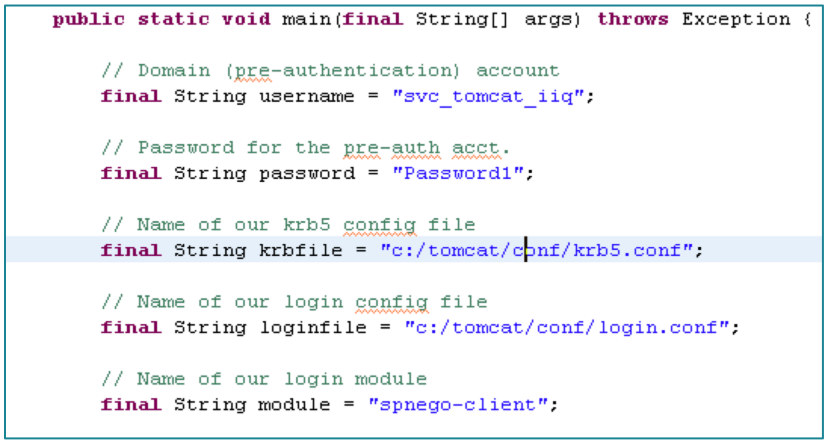
Testing the Configuration Files

At this point, you have created the "krb5.conf" and "login.conf" configuration files and placed them under our Tomcat's "/conf/" directory. These files house half of the configuration details needed to speak with the Active Directory domain controllers. The next step is to test these configuration files with an independent test driver.

Download the "HelloKDC.java" file from the SPNEGO tutorials site at:

http://spnego.sourceforge.net/HelloKDC.java

Edit the test driver program in your Java editor or IDE of choice (Eclipse, IntelliJ, Notepad++, vi, etc.). Modify the 4 variables **username**, **password**, **krbfile**, and **loginfile** to match your service account's user name, password, full path to krb5.conf and full path to login.conf, and compile the program into a .class file.



```
public static void main(final String[] args) throws Exception {

    // Domain (pre-authentication) account
    final String username = "svc_tomcat_iiq";

    // Password for the pre-auth acct.
    final String password = "Password1";

    // Name of our krb5 config file
    final String krbfile = "c:/tomcat/conf/krb5.conf";

    // Name of our login config file
    final String loginfile = "c:/tomcat/conf/login.conf";

    // Name of our login module
    final String module = "spnego-client";
```

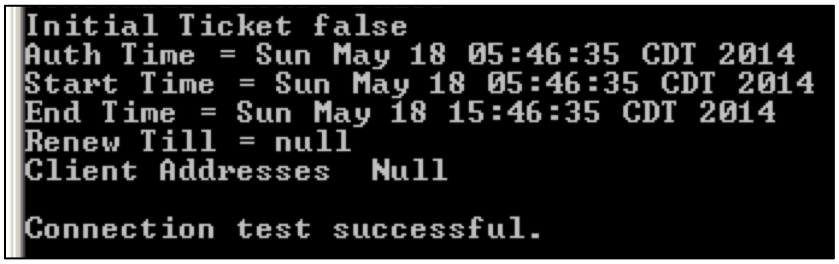
Figure 4: Example HelloKDC.java modifications

Note the use of Unix style slashes for windows path names in the code. This is required.

Transfer the compiled class file out to your Tomcat server and run the program using java from command line.

```
iiq-server-1:~ spadmin$ java HelloKDC.class
```

You should be presented with screens of Debugging text. If your screens end with “Connection test successful.” then the configuration files are valid. If there are errors then read through the debugging output and see if there are any syntax errors to correct.



```
Initial Ticket false
Auth Time = Sun May 18 05:46:35 CDT 2014
Start Time = Sun May 18 05:46:35 CDT 2014
End Time = Sun May 18 15:46:35 CDT 2014
Renew Till = null
Client Addresses  Null

Connection test successful.
```

Figure 5: Example output of a successful test

Editing “web.xml” to enable SPNEGO

The next step is to edit Tomcat’s web.xml file to enable SPNEGO. Open the “…/[tomcat-home]/conf/web.xml” file in a text editor. After the “Built In Filter Definitions” section of comments you will insert and configure a large block of XML text based on the example below.

```
<filter>
    <filter-name>SpnegoHttpFilter</filter-name>
    <filter-class>net.sourceforge.spnego.SpnegoHttpFilter</filter-class>

    <init-param>
        <param-name>spnego.allow.basic</param-name>
        <param-value>true</param-value>
    </init-param>
    <init-param>
        <param-name>spnego.allow.localhost</param-name>
        <param-value>false</param-value>
    </init-param>
```

```
        <init-param>
            <param-name>spnego.allow.unsecure.basic</param-name>
            <param-value>true</param-value>
        </init-param>
        <init-param>
            <param-name>spnego.login.client.module</param-name>
            <param-value>spnego-client</param-value>
        </init-param>

        <!-- EDIT ME! -->
        <init-param>
            <param-name>spnego.krb5.conf</param-name>
            <param-value>c:/tomcat/conf/krb5.conf</param-value>
        </init-param>
        <init-param>
            <param-name>spnego.login.conf</param-name>
            <param-value>c:/tomcat/conf/login.conf</param-value>
        </init-param>
        <init-param>
            <param-name>spnego.preauth.username</param-name>
            <param-value>svc_tomcat_iiq</param-value>
        </init-param>
        <init-param>
            <param-name>spnego.preauth.password</param-name>
            <param-value>Password1</param-value>
        </init-param>
        <!-- Done with EDIT! -->

        <init-param>
            <param-name>spnego.login.server.module</param-name>
            <param-value>spnego-server</param-value>
        </init-param>
        <init-param>
            <param-name>spnego.prompt.ntlm</param-name>
            <param-value>true</param-value>
        </init-param>
        <init-param>
            <param-name>spnego.logger.level</param-name>
            <param-value>1</param-value>
        </init-param>
    </filter>
    <filter-mapping>
        <filter-name>SpnegoHttpFilter</filter-name>
        <url-pattern>/dashboard.jsf</url-pattern>
        <url-pattern>*.jsp</url-pattern>
    </filter-mapping>
```

This entire block needs to be copied into the file, and the EDIT ME section needs to be modified to match your installation's details. Specify the path to your configuration files and your service account's user name and password there.

At the bottom of the block is a "filter-mapping" section. This tells Tomcat the web pages/resources where the SPNEGO authentication filter should be used. The examples shown enable SPNEGO for all JSP pages (for a test driver shown later) and for the main IdentityIQ dashboard, "dashboard.jsf". Note this does not include the login landing page, "login.jsf", which is left un-filtered so users can login with whatever account they wish if they go directly to the login page.

There may be circumstances where your installations wants direct SPNEGO authenticate for other pages, such as "workitem.jsf". The specific list of what filters match for SPNEGO is site-specific and can be configured as needed during a delivery project. After testing, it is recommended that a copy of this file be stored in your project's source code management system.

Some installations have had problems adding the filters to web.xml of the Tomcat server. An alternative approach is to add the filters to the IdentityIQ web.xml. This approach is discouraged because modifying web.xml is a change that has to be merged and reviewed before each patch or upgrade of IdentityIQ. If you must patch IdentityIQ's web.xml file then write the patch as a custom Ant script in your Services Standard Build to do the patching.

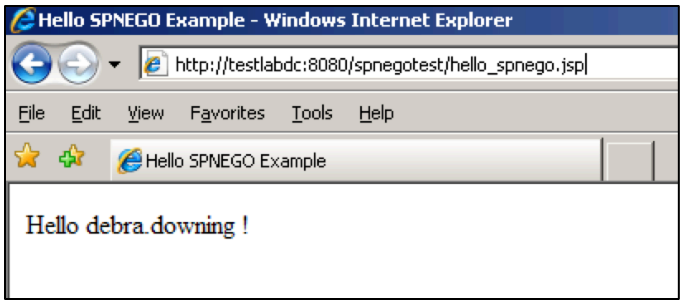
Creating the Hello SPNEGO Test Application

The next step is to create a test application to test the basic SPNEGO functionality under tomcat. This is a simple process:

- Create a sub directory under /webapps/ called "spnegotest".
- Copy the following example text to your Tomcat install as /webapps/spnegotest/hello_spnego.jsp

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
    <title>Hello SPNEGO Example</title>
</head>
<body>
Hello <%= request.getRemoteUser() %> !
</body>
</html>
```

- Start your Tomcat instance and tail the "catalina.out" log file. Look for errors indicating syntax errors from the "web.xml" file. Correct any syntax errors that prevent the web applications from starting.
- Log into a windows box with a regular user account that is a member of your windows domain.
- Open a browser and browse to your Tomcat server and the hello_spnego test application under it (for example: http://testlabdc:8080/spnegotest/hello_spnego.jsp
- You should see a success message as shown below or you should see an error message.


Hello SPNEGO Example - Windows Internet Explorer
http://testlabdc:8080/spnegotest/hello_spnego.jsp
File
Edit
View
Favorites
Tools
Help
Hello SPNEGO Example

Hello debra.downing !

Figure 6: Hello SPNEGO Test Application ran successfully

If you see a hello greeting then Tomcat was successfully able to authenticate you to the JSP page using SPNEGO. If you see an error message then check the Tomcat logs for information. Also turn on Kerberos debugging as shown in the Troubleshooting section below.

NOTE: in the above screen capture only the short hostname ("testlabdc") is being tested. Be sure to test both the long (FQDN) and short hostnames in your environment to see which works and/or which does not work. If one works and the other does not then there is a mis-configuration of your Service Principal Name (SPN).

Configuring the SPNEGO SSO Rule in IdentityIQ

Once the SPNEGO configuration items are in place and tested, the nest step is to configure IdentityIQ to support Single-Sign-On via the SSO rule. This section provides an example SSO rule to use as a starting base for your site-specific SSO rule.

The SPNEGO package places the details about the authenticated user in the HTTP header that is passed to the "javax.servlet.http.HttpServletRequest" instance used by IdentityIQ. This HTTPServletRequest is provided in context to the SSO rule in IdentityIQ so that bean shell code can be used to parse and process the user's details. In the event that SPNEGO could not authenticate a user these fields are null and IdentityIQ defaults to presenting the Login screen to collect the user details.

The example rule below looks for the user details and, for extra visibility, parses all of the fields of the HTTP header and displays them in the log for review when configuring and debugging SSO under IdentityIQ.

An example SPNEGO SSO Rule:

```
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;
import java.util.Enumeration;

import sailpoint.object.Application;
import sailpoint.object.Identity;
import sailpoint.object.Link;
import sailpoint.tools.GeneralException;
import sailpoint.api.Correlator;
import sailpoint.api.SailPointContext;

import org.apache.log4j.Logger;
import org.apache.log4j.Level;

Logger log = Logger.getLogger("sailpoint.services.poc.spnegoSSORule");
// Temporarily force all logging to DEBUG level.
log.setLevel(Level.DEBUG);

log.debug("Inside SPNEGO SSO rule executing...");

if (null == httpRequest) {
   log.error("httpRequest is null!");
   return null;
}

// SPNEGO will set this variable.
String remoteUser = (String) httpRequest.getRemoteUser();
String userPrincipal = null;
String userRealm = null;

// Check to make sure we have a SPNEGO user principal record.
// log.debug("getUserPrincipal(): " + httpRequest.getUserPrincipal());
// log.debug("getUserPrincipal() class: " + httpRequest.getUserPrincipal().getClass().getName());
if ((null != httpRequest.getUserPrincipal()) &amp;&amp;
    (httpRequest.getUserPrincipal() instanceof net.sourceforge.spnego.SpnegoPrincipal)) {
        userPrincipal = httpRequest.getUserPrincipal().getName();
        userRealm = httpRequest.getUserPrincipal().getRealm();
}

if (null != userPrincipal) {
   log.debug("userPrincipal: " + userPrincipal);
}

if (null != userRealm) {
   log.debug("userRealm: " + userRealm);
}

log.debug("requestURI: " + httpRequest.getRequestURI());
log.debug("getHeaderNames(): "   + httpRequest.getHeaderNames());
log.debug("getRemoteUser(): "    + (String)httpRequest.getRemoteUser());

Enumeration names = httpRequest.getHeaderNames();
while(names.hasMoreElements()) {
   String name = (String) names.nextElement();
   log.debug("requestHeader: " + name + " = " + httpRequest.getHeader(name));
}

Correlator correlator = new Correlator(context);

if (null != userPrincipal) {
   log.debug("TODO: Use some site-specific user + domain logic here.");
}

// This logic left in for demonstration and testing purposes:
log.debug("Remote User: " + remoteUser);
Identity id = correlator.findIdentityByAttribute("name", remoteUser);

return id;
```

This rule uses reads back the getUser() method from the SPNEGO principal class to find the user name attached with the domain name. The getRemoteUser() call, also shown in the example script, strips off the realm/domain, which is a problem for installations with multiple domains being managed by one IdentityIQ and SPNEGO integration. Implementers should extend this rule to parse the "userPrincipal" variable by it's component "@" symbol and then search across their Identity attributes as necessary for a correlated Identity.

Example output from this rule as written looks like this:

```
spnegoSSORule:? - Inside SPNEGO SSO rule executing...
spnegoSSORule:? - userPrincipal: chris.clay@EXAMPLE.COM
spnegoSSORule:? - userRealm: EXAMPLE.COM
spnegoSSORule:? - requestURI: /identityiq/dashboard.jsf
spnegoSSORule:? - getHeaderNames(): org.apache.tomcat.util.http.NamesEnumerator@6d1672f4
spnegoSSORule:? - getRemoteUser(): chris.clay
spnegoSSORule:? - requestHeader: accept = image/gif, image/x-xbitmap, image/jpeg, imag...
spnegoSSORule:? - requestHeader: accept-language = en-us
spnegoSSORule:? - requestHeader: ua-cpu = x86
spnegoSSORule:? - requestHeader: accept-encoding = gzip, deflate
spnegoSSORule:? - requestHeader: user-agent = Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1)
spnegoSSORule:? - requestHeader: host = testlabdc:8080
spnegoSSORule:? - requestHeader: connection = Keep-Alive
spnegoSSORule:? - requestHeader: authorization = Negotiate YIIE5QYGKwYBBQUCoIIE2TCCBNW...
spnegoSSORule:? - TODO: Use some site-specific user + domain logic here.
spnegoSSORule:? - Remote User: chris.clay
```

The "TODO" block is left as a hook for the implementer to insert site-specific logic. After development is done and tested and when promoting to production you should remove or comment out the "log.setLevel()" call in the code. This will allow administrators to manage the verbosity of logging at run time instead of forcing this to be printed on every authorization. After testing, it is recommended that a copy of this file be stored in your project's source code management system, under the "config/Rule/" path if using the Services Standard Build.

Multi-Server IdentityIQ Installations

The SPNEGO binaries and configuration must be copied out to each Tomcat instance in your IdentityIQ server cluster. The SailPoint Services Standard Build (SSB) will not take care of this automatically as it only prepares contents for the IdentityIQ web application directly and below. The library and configuration files will need to be manually copied out to each server in your cluster. Fortunately this is a one-time setup step done only the first time you configure SPNEGO or if you SPNEGO's service account password changes. (**NOTE**: the pre-authorization service account's password should be configured to never expire in Active Directory.) The same files can be used across each server in your cluster.

The following files will need to be copied out to each Tomcat instance in your cluster:

- …/[tomcat-home]/lib/spnego-r7.jar
- …/[tomcat-home]/conf/krb5.conf
- …/[tomcat-home]/conf/login.conf
- …/[tomcat-home]/conf/tomcat.keytab
- …/[tomcat-home]/conf/web.xml

Note that Tomcat's "web.xml" may be version specific so make certain that the same release and patch of Tomcat is in use across all of your application servers.

If you use an automated build system to deploy Tomcat for IdentityIQ or if you have extended the SSB to deploy Tomcat for you then you will want to add these 5 files to your source code control system and your deployment scripts. Your configuration and key-tab files should be stored in your source code control system for your IdentityIQ project in a sub directory named "spnego-config" directory, under the main project directory.

Configuring SPNEGO with JBoss

IdentityIQ can be configured with SPNEGO under the JBoss application server in a fashion very similar to the Tomcat configuration. One difference between the Tomcat and JBoss approaches is that JBoss ships with its own specific implementation of SPNEGO. Administrators have had challenges with the JBoss version of SPNEGO that ships with JBoss and recommend using the downloadable version in the same fashion shown in this document for Tomcat.

The steps below assume the open source SPNEGO SourceForge project is used as the login module for JBOSS 7.1, instead of the native SPNEGO login module for JBOSS 7.1. JBOSS AS 7.1.0 Final can be downloaded here:

http://www.jboss.org/jbossas/downloads/

The test implementation was completed with:

- Windows 2008, AD – Domain functional level: Windows Server 2003
- JDK 1.6.0.35
- JBOSS 7.1.0
- IIQ 6.1p3
- SPNEGO release 7 (as shown earlier in this document)

This guide assumes that JBoss is configured "standalone". While this example talks about JBoss on a Windows platform, the same principles apply to JBoss on a Unix/Linux environment. For JBoss instances running under Unix/Linux a ".keytab" file will be required; follow the Tomcat instructions shown in the "Creating a Key-Tab File" section above.

JBoss Configuration Steps

1. Review the pre-requisites section of this document for the Tomcat install. All access privileges and administrative rights requirements apply equally to JBoss as they do to Tomcat.
2. Follow the pre-flight documentation on the SPNEGO SourceForge site. Don't perform any JBOSS specific instructions, as they will be documented here.
3. Install JBOSS 7.1 (or newer) on the windows server.
4. Ensure that the IdentityIQ "6_1_IdentityIQ_Installation_Guide.pdf" (or guide appropriate for your version of IdentityIQ) install instructions for JBOSS 7.1 are complete.
5. Configure JBOSS to use an external public interface.
6. Copy the krb5.conf and login.conf from the SPNEGO pre-flight instructions into the {JBOSS_HOME}/bin directory
7. Download the "spnego-r7.jar" file from the SPNEGO download site.
8. Copy the spnego-r7.jar file into the IdentityIQ installs' ".../WEB-INF/lib" directory. If using the SailPoint Services Standard Build then also add the .jar file to the build's "./web/WEB-INF/lib" directory and check it into your source code management system.
9. Extract the ./WEB-INF/web.xml file from your IdentityIQ installation files.
10. Modify IdentityIQ's "web.xml" file, inserting the filter section from the "Edit "web.xml" to enable SPNEGO" of this document into the web.xml file. Note that this is different than the recommendation for Tomcat, which recommends putting the SPNEGO configuration inside Tomcat's "web.xml". During any subsequent upgrade of IdentityIQ, this modification of web.xml must be reapplied to the new

version's web.xml file. **Ensure this SPNEGO section comes before any other filter configuration in the file** as this needs to be executed first.

11. Copy the extracted and modified ./WEB-INF/web.xml file to the services builds /web/WEB-INF directory.
12. Edit the {jboss root}/standalone/configuration/standalone.xml file
    a. Locate the `<subsystem xmlns="urn:jboss:domain:security:1.1">` section.
    b. Insert into the top of this section the following block and save the "standalone.xml" file:

```
<security-domain name="spnego-client" cache-type="default">
  <authentication>
    <login-module code="com.sun.security.auth.module.Krb5LoginModule" flag="required"/>
  </authentication>
</security-domain>
<security-domain name="spnego-server" cache-type="default">
  <authentication>
    <login-module code="com.sun.security.auth.module.Krb5LoginModule" flag="required">
      <module-option name="storeKey" value="true"/>
    </login-module>
  </authentication>
</security-domain>
```

13. Launch the JBOSS instance using {JBOSS_HOME}/bin/standalone.bat file.

Troubleshooting

Configuring SPNEGO can be a very finicky and painstaking process. One single typo in a configuration file made several steps back can be hard to track down and identify. While this guide attempts to provide as much context and copy-paste based examples, errors and site-specific challenges will invariably occur. The goal of this section is cover common pitfalls found during configuration.

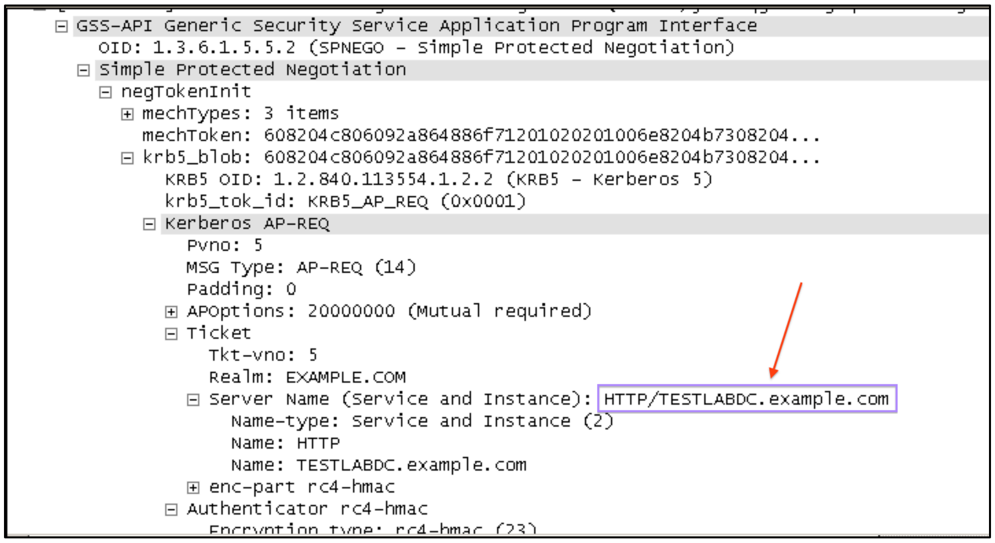
Identifying your Service Principal Name

When processing IWA and SSO transactions the end-user's browser is actually the piece of software that defines the Service Principal Name (SPN) to which the user is requesting access. In some environments it can be challenging to identify the SPN used by the end-user client-side browsers when connecting to IdentityIQ. This can lead to frustration when attempting to create configuration files and keytab files for your environment. Sometimes the SPN is case sensitive, sometimes it uses port numbers, other times not.

Fortunately there is a relatively simple and deterministic way to figure out what the SPN name for your IdentityIQ service should be. Here is the procedure for identifying what your SPN should be:

- Have the test .jsp script installed under Tomcat and have SPNEGO enabled in web.xml
- Download and install the WireShark network tracing tool into a workstation
- Trace the network HTTP and HTTPS network connections on the workstation
- Open a browser and attempt to connect to the test .jsp script on the IdentityIQ server
- Expand the HTTP transactions and you should see a "401 Unauthorized" response from the server
- In the next transaction you will see the browser respond with another GET attempt with negotiation
- Drill into the GSS-API and Simple Protected Negotiation portions of the packet and you will see the principal name for the service

The following illustration shows the identification of the Service Principal Name being requested from an Internet Explorer session as captured by WireShark.



```
⊟ GSS-API Generic Security Service Application Program Interface
    OID: 1.3.6.1.5.5.2 (SPNEGO - Simple Protected Negotiation)
  ⊟ Simple Protected Negotiation
    ⊟ negTokenInit
      ⊞ mechTypes: 3 items
        mechToken: 608204c806092a864886f71201020201006e8204b7308204...
      ⊟ krb5_blob: 608204c806092a864886f71201020201006e8204b7308204...
          KRB5 OID: 1.2.840.113554.1.2.2 (KRB5 - Kerberos 5)
          krb5_tok_id: KRB5_AP_REQ (0x0001)
        ⊟ Kerberos AP-REQ
            Pvno: 5
            MSG Type: AP-REQ (14)
            Padding: 0
          ⊞ APOptions: 20000000 (Mutual required)
          ⊟ Ticket
              Tkt-vno: 5
              Realm: EXAMPLE.COM
            ⊟ Server Name (Service and Instance): HTTP/TESTLABDC.example.com
                Name-type: Service and Instance (2)
                Name: HTTP
                Name: TESTLABDC.example.com
            ⊞ enc-part rc4-hmac
          ⊟ Authenticator rc4-hmac
              Encryption type: rc4-hmac (23)
```

Figure 7: Server Name identifies the Service Principal Name

Once you have identified this component remember the SPN has an @DOMAIN.SUFFIX extension to its full name. So if your domain is EXAMPLE.COM then the full SPN is:

"HTTP/TESTLABDC.example.com@EXAMPLE.COM"

Enabling Kerberos debugging

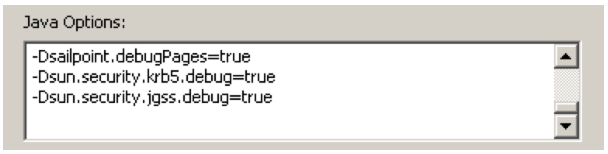
It is useful to turn on Kerberos library debugging in your JVM and Tomcat instance to see extra debugging messages about communication with the Active Directory domain controllers. To do this you enable the following JMV options in your environment:

```
-Dsun.security.krb5.debug=true
-Dsun.security.jgss.debug=true
```

On Linux installations of Tomcat, set these in your "…/[tomcat-home]/bin/catalina.sh" configuration file:

```
# Enable Kerberos/SPNEGO debugging:
JAVA_OPTS="-Dsun.security.krb5.debug=true $JAVA_OPTS"
JAVA_OPTS="-Dsun.security.jgss.debug=true $JAVA_OPTS"
```

For Windows based installations of Tomcat you can set these properties in the Windows Tomcat agent under the "Java" tab, adjacent to where the maximum memory and class paths are configured:



Java Options:

```
-Dsailpoint.debugPages=true
-Dsun.security.krb5.debug=true
-Dsun.security.jgss.debug=true
```

Figure 8: Java Options in Windows Tomcat agent

Once enabled you will see informative messages about the Kerberos transactions in the log files.

You can also enable Kerberos logging to log files by editing the krb5.conf file and including references for log files in the configuration (**NOTE**: these examples are given with Unix path names):

```
default = FILE:/var/log/krb5libs.log
kdc = FILE:/var/log/krb5kdc.log
admin_server = FILE:/var/log/kadmind.log
```

Syntax Errors in web.xml

It is common to introduce syntax errors while editing the …/tomcat/conf/web.xml file. Specifically editing the filter string section can be error prone if you are using a custom dashboard or landing page. The "catalina.out" log file under Tomcat contains the standard output details that can help you in this scenario. Look for log messages like the following that indicate a syntax error or other misconfiguration from "web.xml":

```
May 18, 2014 2:02:58 AM org.apache.catalina.startup.ContextConfig processDefaultWebConfig SEVERE:
Parse error in default web.xml org.xml.sax.SAXParseException; systemId:
file://C/tomcat/conf/web.xml; lineNumber: 520; columnNumber: 18;
Error at (520, 18: Invalid <url-pattern> dashboard.jsf in filter mapping   at
org.apache.tomcat.util.digester.Digester.createSAXException(Digester.java:2806)
```