複数台のLDAPサーバを参照するための方法

ここでは2つの方法をご紹介します。

1. 複数台LDAPサーバ向けのLDAPプロキシサーバを使用する

複数台のLDAPサーバ向けにLDAPプロキシサーバを設置し、IdPにはLDAPプロキシサーバを参照させます。
使用するLDAPプロキシサーバの設定方法について、まとめられた資料があります。資料はCentOS 5系で記載されたものであるため、利用するバージョンに合わせて適宜読み替える必要があります。

> ⚠ プロキシサーバを構築する際、ログイン画面で入力するID（Shibboleth内部ではprincipalと表現されます）について、同一のIDが複数のLDAPツリー上に存在しないことを確認してください。同一のIDが存在する場合には、属性取得で問題が発生します。uidがこの条件を満たさない場合は、メールアドレスや学籍番号・教職員番号等、他のLDAP属性を使うことを検討してください。

2. IdPの設定ファイルで複数台LDAPサーバを指定する

IdPの設定で複数のLDAPツリー或いは複数のLDAPサーバを参照する例がShibboleth Wiki:LDAPAuthnConfigurationの"DNResolution"の項にあります。
最初の例（"Single Directory with multiple branches"の"Extensible Matching"）はLDAPサーバが一台のみで検索すべき複数のLDAPツリーがサブツリーの関係にある環境向けで、`ldap.properties`のみの変更で対応が可能です。`idp.authn.LDAP.baseDN`のサブツリーを検索するよう`idp.authn.LDAP.subtreeSearch`と`idp.authn.LDAP.userFilter`を設定します。

- LDAPサーバが一台であり複数のLDAPツリーがサブツリーの関係にある場合の例

/opt/shibboleth-idp/conf/ldap.properties の設定例

```
# Search DN resolution, used by anonSearchAuthenticator, bindSearchAuthenticator
# for AD: CN=Users,DC=example,DC=org
idp.authn.LDAP.baseDN = o=test_o,dc=ac,c=JP
idp.authn.LDAP.subtreeSearch = true ← trueを設定
idp.authn.LDAP.userFilter = (&(|(ou:dn:=Test Unit1)(ou:dn:=technology))(uid={user})) ← o=test_o,dc=ac,c=JPのサブツリー、
ou=Test Unit1とou=technologyを検索するよう設定
```

次の2つの例（"Single Directory with multiple branches - Aggregate DN Resolver"および"Multiple Directories"）は複数のLDAPツリーがサブツリーとして扱えない環境、或いはLDAPサーバが複数の環境向けで、`ldap.properties`と`ldap-authn-config.xml`を変更します。
LDAPサーバ/LDAPツリーの数に応じて、`ldap.properties`内の項目を追加して`authn-ldap-config.xml`でそれらを参照するようにします。
また、これらの例は認証処理のための設定なので、SPへ送出する属性をLDAPから取得している場合は`attribute-resolver.xml`を変更し、`ldap.properties`に追加した項目を参照するLDAP DataConnectorを追加します。

- LDAPサーバが一台であり複数のLDAPツリーがサブツリーとして扱えない場合の例

/opt/shibboleth-idp/conf/authn/ldap-authn-config.xml の設定例

```
    <bean name="aggregateAuthenticator" class="org.ldaptive.auth.Authenticator">
        <constructor-arg index="0" ref="aggregateDnResolver" />
        <constructor-arg index="1" ref="aggregateAuthHandler" />
    </bean>
    <bean id="aggregateDnResolver" class="org.ldaptive.auth.AggregateDnResolver">
        <constructor-arg index="0" ref="dnResolvers" />
    </bean>
    <bean id="aggregateAuthHandler" class="org.ldaptive.auth.AggregateDnResolver$AuthenticationHandler" p:
authenticationHandlers-ref="authHandlers" />
    <util:map id="dnResolvers">
        <entry key="filter1" value-ref="dnResolver1" />
        <entry key="filter2" value-ref="dnResolver2" />
    </util:map>
    <!-- Define two DN resolvers that use anonymous search against the same directory -->
    <bean id="dnResolver1" class="org.ldaptive.auth.PooledSearchDnResolver" p:baseDn="%{idp.authn.LDAP.baseDN}"
          p:subtreeSearch="%{idp.authn.LDAP.subtreeSearch:false}" p:userFilter="%{idp.authn.LDAP.userFilter}"
          p:connectionFactory-ref="anonSearchPooledConnectionFactory" />
    <bean id="dnResolver2" class="org.ldaptive.auth.PooledSearchDnResolver" p:baseDn="%{idp.authn.LDAP.baseDN2}"
          p:subtreeSearch="%{idp.authn.LDAP.subtreeSearch:false}" p:userFilter="%{idp.authn.LDAP.userFilter2}"
          p:connectionFactory-ref="anonSearchPooledConnectionFactory" />
    <!-- Use the same authentication handler for both DN resolvers -->
    <util:map id="authHandlers">
        <entry key="filter1" value-ref="authHandler" />
        <entry key="filter2" value-ref="authHandler" />
    </util:map> ← </beans>の前に追加

</beans>
```

/opt/shibboleth-idp/conf/ldap.properties の設定例

```
idp.authn.LDAP.authenticator = aggregateAuthenticator ← ldap-authn-config.xmlに追加したclass="org.ldaptive.auth.
Authenticator"のbeanのnameに変更
(省略)
# Search DN resolution, used by anonSearchAuthenticator, bindSearchAuthenticator
# for AD: CN=Users,DC=example,DC=org
idp.authn.LDAP.baseDN = LDAPツリー1のBaseDN ← 変更
idp.authn.LDAP.baseDN2 = LDAPツリー2のBaseDN ← 追加
idp.authn.LDAP.subtreeSearch = true
idp.authn.LDAP.userFilter = (LDAPツリー1の検索キー={user}) ← 変更
idp.authn.LDAP.userFilter2 = (LDAPツリー2の検索キー={user}) ← 追加
(省略)
# LDAP attribute configuration, see attribute-resolver.xml
# Note, this likely won't apply to the use of legacy V2 resolver configurations
idp.attribute.resolver.LDAP.ldapURL = %{idp.authn.LDAP.ldapURL}
idp.attribute.resolver.LDAP.baseDN = %{idp.authn.LDAP.baseDN:undefined}
idp.attribute.resolver.LDAP.baseDN2 = %{idp.authn.LDAP.baseDN2:undefined} ← 追加
idp.attribute.resolver.LDAP.bindDN = %{idp.authn.LDAP.bindDN:undefined}
```

/opt/shibboleth-idp/conf/attribute-resolver.xml の設定例

```
    <resolver:DataConnector id="myLDAP" xsi:type="dc:LDAPDirectory"
        ldapURL="%{idp.attribute.resolver.LDAP.ldapURL}"
        baseDN="%{idp.attribute.resolver.LDAP.baseDN}"
        principal="%{idp.attribute.resolver.LDAP.bindDN}"
        principalCredential="%{idp.attribute.resolver.LDAP.bindDNCredential}"
        noResultIsError="True" ← 追加
        useStartTLS="%{idp.attribute.resolver.LDAP.useStartTLS:true}">
        <resolver:FailoverDataConnector ref="myLDAP2" /> ← 追加(dcより前に追加する必要があります)
        <dc:FilterTemplate>
            <![CDATA[
                %{idp.attribute.resolver.LDAP.searchFilter}
            ]]>
        </dc:FilterTemplate>
        <dc:StartTLSTrustCredential id="LDAPtoIdPCredential" xsi:type="sec:X509ResourceBacked">
            <sec:Certificate>%{idp.attribute.resolver.LDAP.trustCertificates}</sec:Certificate>
        </dc:StartTLSTrustCredential>
    </resolver:DataConnector>
    ↓以下の行を追加
    <resolver:DataConnector id="myLDAP2" xsi:type="dc:LDAPDirectory"
        ldapURL="%{idp.attribute.resolver.LDAP.ldapURL}"
        baseDN="%{idp.attribute.resolver.LDAP.baseDN2}"
        principal="%{idp.attribute.resolver.LDAP.bindDN}"
        principalCredential="%{idp.attribute.resolver.LDAP.bindDNCredential}"
        noResultIsError="True"
        useStartTLS="%{idp.attribute.resolver.LDAP.useStartTLS:true}">
        <dc:FilterTemplate>
            <![CDATA[
                %{idp.attribute.resolver.LDAP.searchFilter}
            ]]>
        </dc:FilterTemplate>
        <dc:StartTLSTrustCredential id="LDAPtoIdPCredential" xsi:type="sec:X509ResourceBacked">
            <sec:Certificate>%{idp.attribute.resolver.LDAP.trustCertificates}</sec:Certificate>
        </dc:StartTLSTrustCredential>
    </resolver:DataConnector>
```

- LDAPサーバが複数である場合の例

/opt/shibboleth-idp/conf/authn/ldap-authn-config.xml の設定例

```
    <alias name="%{idp.authn.LDAP.sslConfig2:certificateTrust2}" alias="sslConfig2" />
    <bean id="certificateTrust2" class="org.ldaptive.ssl.SslConfig">
        <property name="credentialConfig">
            <bean parent="shibboleth.X509ResourceCredentialConfig" p:trustCertificates="%{idp.authn.LDAP.trustCertificates2:
undefined}" />
        </property>
    </bean>
    <bean id="keyStoreTrust2" class="org.ldaptive.ssl.SslConfig">
        <property name="credentialConfig">
            <bean parent="shibboleth.KeystoreResourceCredentialConfig" p:truststore="%{idp.authn.LDAP.trustStore2:undefined}"
/>
        </property>
    </bean>
    <bean name="aggregateAuthenticator" class="org.ldaptive.auth.Authenticator"
          c:resolver-ref="aggregateDnResolver"
          c:handler-ref="aggregateAuthHandler" />

    <!-- Aggregate DN resolution -->
    <bean id="aggregateDnResolver" class="org.ldaptive.auth.AggregateDnResolver"
          c:resolvers-ref="dnResolvers"
          p:allowMultipleDns="true" />
    <util:map id="dnResolvers">
        <entry key="directory1" value-ref="bindSearchDnResolver1" />
        <entry key="directory2" value-ref="bindSearchDnResolver2" />
    </util:map>

    <!-- DN resolver 1 -->
    <bean id="bindSearchDnResolver1" class="org.ldaptive.auth.PooledSearchDnResolver"
          p:baseDn="#{'%{idp.authn.LDAP.baseDN:undefined}'.trim()}"
          p:subtreeSearch="%{idp.authn.LDAP.subtreeSearch:false}"
          p:userFilter="#{'%{idp.authn.LDAP.userFilter:undefined}'.trim()}"
```

```
      p:connectionFactory-ref="bindSearchPooledConnectionFactory" />
<bean id="bindSearchPooledConnectionFactory1" class="org.ldaptive.pool.PooledConnectionFactory"
      p:connectionPool-ref="bindSearchConnectionPool1" />
<bean id="bindSearchConnectionPool1" class="org.ldaptive.pool.BlockingConnectionPool" parent="connectionPool"
      p:connectionFactory-ref="bindSearchConnectionFactory1"
      p:name="search-pool1" />
<bean id="bindSearchConnectionFactory1" class="org.ldaptive.DefaultConnectionFactory"
      p:connectionConfig-ref="bindSearchConnectionConfig1" />
<bean id="bindSearchConnectionConfig1" parent="connectionConfig"
      p:connectionInitializer-ref="bindConnectionInitializer1"
      p:ldapUrl="%{idp.authn.LDAP.ldapURL}" />
<bean id="bindConnectionInitializer1" class="org.ldaptive.BindConnectionInitializer"
      p:bindDn="#{'%{idp.authn.LDAP.bindDN:undefined}'.trim()}">
    <property name="bindCredential">
        <bean class="org.ldaptive.Credential" c:password="%{idp.authn.LDAP.bindDNCredential:undefined}" />
    </property>
</bean>
<!-- DN resolver 2 -->
<bean id="bindSearchDnResolver2" class="org.ldaptive.auth.PooledSearchDnResolver"
      p:baseDn="#{'%{idp.authn.LDAP.baseDN2:undefined}'.trim()}"
      p:subtreeSearch="%{idp.authn.LDAP.subtreeSearch:false}"
      p:userFilter="#{'%{idp.authn.LDAP.userFilter2:undefined}'.trim()}"
      p:connectionFactory-ref="bindSearchPooledConnectionFactory" />
<bean id="bindSearchPooledConnectionFactory2" class="org.ldaptive.pool.PooledConnectionFactory"
      p:connectionPool-ref="bindSearchConnectionPool2" />
<bean id="bindSearchConnectionPool2" class="org.ldaptive.pool.BlockingConnectionPool" parent="connectionPool"
      p:connectionFactory-ref="bindSearchConnectionFactory2"
      p:name="search-pool2" />
<bean id="bindSearchConnectionFactory2" class="org.ldaptive.DefaultConnectionFactory"
      p:connectionConfig-ref="bindSearchConnectionConfig2" />
<bean id="bindSearchConnectionConfig2" parent="connectionConfig"
      p:connectionInitializer-ref="bindConnectionInitializer2"
      p:ldapUrl="%{idp.authn.LDAP.ldapURL2}"
      p:useStartTLS="%{idp.authn.LDAP.useStartTLS2:true}"
      p:useSSL="%{idp.authn.LDAP.useSSL2:false}"
      p:connectTimeout="%{idp.authn.LDAP.connectTimeout2:3000}"
      p:sslConfig-ref="sslConfig2" />
<bean id="bindConnectionInitializer2" class="org.ldaptive.BindConnectionInitializer"
      p:bindDn="#{'%{idp.authn.LDAP.bindDN2:undefined}'.trim()}">
    <property name="bindCredential">
        <bean class="org.ldaptive.Credential" c:password="%{idp.authn.LDAP.bindDNCredential2:undefined}" />
    </property>
</bean>

<!-- Aggregate authentication -->
<bean id="aggregateAuthHandler" class="org.ldaptive.auth.AggregateDnResolver$AuthenticationHandler"
      p:authenticationHandlers-ref="authHandlers" />
<util:map id="authHandlers">
    <entry key="directory1" value-ref="authHandler1" />
    <entry key="directory2" value-ref="authHandler2" />
</util:map>

<!-- Authentication handler 1 -->
<bean id="authHandler1" class="org.ldaptive.auth.PooledBindAuthenticationHandler"
      p:connectionFactory-ref="bindPooledConnectionFactory1" />
<bean id="bindPooledConnectionFactory1" class="org.ldaptive.pool.PooledConnectionFactory"
      p:connectionPool-ref="bindConnectionPool1" />
<bean id="bindConnectionPool1" class="org.ldaptive.pool.BlockingConnectionPool" parent="connectionPool"
      p:connectionFactory-ref="bindConnectionFactory1"
      p:name="bind-pool1" />
<bean id="bindConnectionFactory1" class="org.ldaptive.DefaultConnectionFactory"
      p:connectionConfig-ref="bindConnectionConfig1" />
<bean id="bindConnectionConfig1" parent="connectionConfig"
      p:ldapUrl="%{idp.authn.LDAP.ldapURL}" />
<!-- Authentication handler 2 -->
<bean id="authHandler2" class="org.ldaptive.auth.PooledBindAuthenticationHandler"
      p:connectionFactory-ref="bindPooledConnectionFactory2" />
<bean id="bindPooledConnectionFactory2" class="org.ldaptive.pool.PooledConnectionFactory"
      p:connectionPool-ref="bindConnectionPool2" />
<bean id="bindConnectionPool2" class="org.ldaptive.pool.BlockingConnectionPool" parent="connectionPool"
      p:connectionFactory-ref="bindConnectionFactory2"
      p:name="bind-pool2" />
<bean id="bindConnectionFactory2" class="org.ldaptive.DefaultConnectionFactory"
      p:connectionConfig-ref="bindConnectionConfig2" />
<bean id="bindConnectionConfig2" parent="connectionConfig"
      p:ldapUrl="%{idp.authn.LDAP.ldapURL2}"
```

```
        p:useStartTLS="%{idp.authn.LDAP.useStartTLS2:true}"
        p:useSSL="%{idp.authn.LDAP.useSSL2:false}"
        p:connectTimeout="%{idp.authn.LDAP.connectTimeout2:3000}"
        p:sslConfig-ref="sslConfig2" /> ← </beans>の前に追加

</beans>
```

/opt/shibboleth-idp/conf/ldap.properties の設定例

```
idp.authn.LDAP.authenticator = aggregateAuthenticator ← ldap-authn-config.xmlに追加したclass="org.ldaptive.auth.
Authenticator"のbeanのnameに変更

## Connection properties ##
idp.authn.LDAP.ldapURL                          = LDAPサーバ1のURL ← 変更
idp.authn.LDAP.ldapURL2                          = LDAPサーバ2のURL ← 追加
idp.authn.LDAP.useStartTLS                      = true
idp.authn.LDAP.useStartTLS2                      = true
#idp.authn.LDAP.useSSL                          = false
#idp.authn.LDAP.connectTimeout                  = 3000

## SSL configuration, either jvmTrust, certificateTrust, or keyStoreTrust
idp.authn.LDAP.sslConfig                       = certificateTrust
↓ 追加(certificateTrustかkeyStoreTrustを指定する際はldap-authn-config.xmlに追加したLDAPサーバ2用のclass="org.ldaptive.ssl.
SslConfig"のbeanのidを指定)
idp.authn.LDAP.sslConfig2                        = certificateTrust2 ← 追加
## If using certificateTrust above, set to the trusted certificate's path
idp.authn.LDAP.trustCertificates                = LDAPサーバ1の証明書 ← 変更
idp.authn.LDAP.trustCertificates2                = LDAPサーバ2の証明書 ← 追加
## If using keyStoreTrust above, set to the truststore path
idp.authn.LDAP.trustStore                       = LDAPサーバ1のキーストア ← 変更
idp.authn.LDAP.trustStore2                       = LDAPサーバ2のキーストア ← 追加
(省略)
# Search DN resolution, used by anonSearchAuthenticator, bindSearchAuthenticator
# for AD: CN=Users,DC=example,DC=org
idp.authn.LDAP.baseDN                           = LDAPサーバ1のBaseDN ← 変更
idp.authn.LDAP.baseDN2                           = LDAPサーバ2のBaseDN ← 追加
idp.authn.LDAP.subtreeSearch                   = true
idp.authn.LDAP.userFilter                       = (LDAPサーバ1の検索キー={user}) ← 変更
idp.authn.LDAP.userFilter2                       = (LDAPサーバ2の検索キー={user}) ← 追加
# bind search configuration
# for AD: idp.authn.LDAP.bindDN=adminuser@domain.com
idp.authn.LDAP.bindDN                           = LDAPサーバ1のBindDN ← 変更
idp.authn.LDAP.bindDN2                           = LDAPサーバ2のBindDN ← 追加
idp.authn.LDAP.bindDNCredential                 = LDAPサーバ1のBindDNパスワード ← 変更
idp.authn.LDAP.bindDNCredential2                 = LDAPサーバ2のBindDNパスワード ← 追加
(省略)
# LDAP attribute configuration, see attribute-resolver.xml
# Note, this likely won't apply to the use of legacy V2 resolver configurations
idp.attribute.resolver.LDAP.ldapURL             = %{idp.authn.LDAP.ldapURL}
idp.attribute.resolver.LDAP.ldapURL2             = %{idp.authn.LDAP.ldapURL2} ← 追加
idp.attribute.resolver.LDAP.baseDN              = %{idp.authn.LDAP.baseDN:undefined}
idp.attribute.resolver.LDAP.baseDN2              = %{idp.authn.LDAP.baseDN2:undefined} ← 追加
idp.attribute.resolver.LDAP.bindDN              = %{idp.authn.LDAP.bindDN:undefined}
idp.attribute.resolver.LDAP.bindDN2              = %{idp.authn.LDAP.bindDN2:undefined} ← 追加
idp.attribute.resolver.LDAP.bindDNCredential    = %{idp.authn.LDAP.bindDNCredential:undefined}
idp.attribute.resolver.LDAP.bindDNCredential2    = %{idp.authn.LDAP.bindDNCredential2:undefined} ← 追加
idp.attribute.resolver.LDAP.useStartTLS         = %{idp.authn.LDAP.useStartTLS:true}
idp.attribute.resolver.LDAP.useStartTLS2         = %{idp.authn.LDAP.useStartTLS2:true} ← 追加
idp.attribute.resolver.LDAP.trustCertificates   = %{idp.authn.LDAP.trustCertificates:undefined}
idp.attribute.resolver.LDAP.trustCertificates2   = %{idp.authn.LDAP.trustCertificates2:undefined} ← 追加
idp.attribute.resolver.LDAP.searchFilter        = (LDAPサーバ1の検索キー=$resolutionContext.principal) ← 変更
idp.attribute.resolver.LDAP.searchFilter2        = (LDAPサーバ2の検索キー=$resolutionContext.principal) ← 追加
idp.attribute.resolver.LDAP.returnAttributes    = cn,homephone,mail
```

/opt/shibboleth-idp/conf/attribute-resolver.xml の設定例

```
    <resolver:DataConnector id="myLDAP" xsi:type="dc:LDAPDirectory"
        ldapURL="%{idp.attribute.resolver.LDAP.ldapURL}"
        baseDN="%{idp.attribute.resolver.LDAP.baseDN}"
        principal="%{idp.attribute.resolver.LDAP.bindDN}"
        principalCredential="%{idp.attribute.resolver.LDAP.bindDNCredential}"
        noResultIsError="True" ← 追加
        useStartTLS="%{idp.attribute.resolver.LDAP.useStartTLS:true}">
        <resolver:FailoverDataConnector ref="myLDAP2" /> ← 追加(dcより前に追加する必要があります)
        <dc:FilterTemplate>
            <![CDATA[
                %{idp.attribute.resolver.LDAP.searchFilter}
            ]]>
        </dc:FilterTemplate>
        <dc:StartTLSTrustCredential id="LDAPtoIdPCredential" xsi:type="sec:X509ResourceBacked">
            <sec:Certificate>%{idp.attribute.resolver.LDAP.trustCertificates}</sec:Certificate>
        </dc:StartTLSTrustCredential>
    </resolver:DataConnector>
    ↓以下の行を追加
    <resolver:DataConnector id="myLDAP2" xsi:type="dc:LDAPDirectory"
        ldapURL="%{idp.attribute.resolver.LDAP.ldapURL2}"
        baseDN="%{idp.attribute.resolver.LDAP.baseDN2}"
        principal="%{idp.attribute.resolver.LDAP.bindDN2}"
        principalCredential="%{idp.attribute.resolver.LDAP.bindDNCredential2}"
        noResultIsError="True"
        useStartTLS="%{idp.attribute.resolver.LDAP.useStartTLS2:true}">
        <dc:FilterTemplate>
            <![CDATA[
                %{idp.attribute.resolver.LDAP.searchFilter2}
            ]]>
        </dc:FilterTemplate>
        <dc:StartTLSTrustCredential id="LDAPtoIdPCredential" xsi:type="sec:X509ResourceBacked">
            <sec:Certificate>%{idp.attribute.resolver.LDAP.trustCertificates2}</sec:Certificate>
        </dc:StartTLSTrustCredential>
    </resolver:DataConnector>
```

attribute-filter.xmlの変更はDataConnector "myLDAP" にデータが見つからない場合エラーとして扱うようにし、フェイルオーバー処理としてDataConnector "myLDAP2" から改めてデータを検索する設定になります。

1つ目の方法と同様、同一のIDが複数のLDAPツリー上に存在すると問題になりますので、uidがこの条件を満たさない場合は他のLDAP属性をID(principal)として使うようにしてください。