LDAPConnector

The `LDAPDirectory` data connector generates one or more attributes based on a query to an LDAP Directory.

The underlying library used for this is called ldaptive, and many of the more advanced options are usable only by referring to its documentation, but most deployers should be able to make to do with the information we supply.

Schema Name and Location

This `xsi:type` is defined in the `urn:mace:shibboleth:2.0:resolver` namespace [3.3], the schema for which is located at http://shibboleth.net/schema/idp/shibboleth-attribute-resolver.xsd

Prior to V3.3 supplied plugins were defined by a schema type in the `urn:mace:shibboleth:2.0:resolver:dc` namespace, the schema for which is located at http://shibboleth.net/schema/idp/shibboleth-attribute-resolver-dc.xsd. This is still supported, but every element or type in the old namespace has an equivalently named (but not necessarily identical) version in the `urn:mace:shibboleth:2.0:resolver` namespace. The use of the `urn:mace:shibboleth:2.0:resolver` namespace also allows a relaxation of the ordering requirements of child elements to reduce strictness.

Default Configuration

The IdP ships with an example configuration file, *attribute-resolver-ldap.xml*, that demonstrates property-driven attribute resolution, using the properties defined in *ldap.properties*, much of which can be shared with the LDAP Authentication Configuration for the common case of a single LDAP directory used for both authentication and attributes. You can freely mix use of properties with explicit settings.

If execution of the query returns multiple results, these are merged by default.

The following properties are demonstrated. They take their defaults from the values of the similarly-named authentication properties, but can be explicitly set to split them off as needed.

Property	Type	Default Authn Property	Function
idp.attribute.resolver.LDAP.ldapURL	URL	idp.authn.LDAP.ldapURL	Connection URL for the LDAP directory
idp.attribute.resolver.LDAP.baseDN	String	idp.authn.LDAP.baseDN	Base DN to search against,
idp.attribute.resolver.LDAP.bindDN	String	idp.authn.LDAP.bindDN	DN to bind as before performing the search
idp.attribute.resolver.LDAP. bindDNCredential	String	idp.authn.LDAP.bindDNCredential	Password to bind with before performing the search
idp.attribute.resolver.LDAP.useStartTLS	Boolean	idp.authn.LDAP.useStartTLS (defaults true)	Whether StartTLS should be used immediately after connecting to the LDAP
idp.attribute.resolver.LDAP. trustCertificates	Resour ce	idp.authn.LDAP.trustCertificates	A resource to load trust anchors from, usually a local file in %{idp. home}/credentials
idp.attribute.resolver.LDAP. connectTimeout	Integer	idp.authn.LDAP.connectTimeout (defaults PT3S)	Connection timeout in milliseconds
idp.attribute.resolver.LDAP. responseTimeout [3.3]	Duration	idp.authn.LDAP.responseTimeout (defaults PT3S)	Time to wait for response

TLS Validation Configuration

Assuming ldap-over-TLS (ldaps) or StartTLS is used, you SHOULD (and, in a future version of the software, MUST) configure the rules for validating the LDAP server's TLS key within the connector. It is possible to leave this to the Java runtime by relying on default behavior, but this will result in warnings as of V3.3.2 and will cease to function in V4.0, as this advisory outlines.

There are generally two ways to do this:

- Reference a CA (Certificate Authority) that has signed the certificate chain presented by the LDAP server.
- Reference the LDAP server's certificate explicitly.

The latter provides better security but likely will require constant updating and coordination with the LDAP server's operational staff unless it relies on a longer-lived certificate.

You may configure multiple certificates in whatever combination you choose (by concatenating PEM formatted certificates together).

You reference the certificate(s) you choose to trust by setting the `trustFile` XML attribute inside the `<DataConnector>` element, and the example below passes that through to a property you can configure separately, but that's purely a matter of style.

Note that this is distinct from configuring the IdP to authenticate itself **to** the LDAP server with a certificate. That relies on a different set of options (`authKey` , `authCert`).

Examples

Simple DataConnector entirely in custom syntax

```
<DataConnector id="myLDAP" xsi:type="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}"
        trustFile="%{idp.attribute.resolver.LDAP.trustCertificates}"
        useStartTLS="%{idp.attribute.resolver.LDAP.useStartTLS:true}"
    noResultIsError="%{idp.attribute.resolver.LDAP.noResultsIsError:false}"
    multipleResultsIsError="%{idp.attribute.resolver.LDAP.multipleResultsIsError:true}">
    <FilterTemplate>
        <![CDATA[
            %{idp.attribute.resolver.LDAP.searchFilter}
        ]]>
    </FilterTemplate>
    <LDAPProperty name="name1" value="%{idp.attribute.resolver.LDAP.prop1}"/>
    <LDAPProperty name="name2" value="%{idp.attribute.resolver.LDAP.prop2}"/>
    <StartTLSAuthenticationCredential xsi:type="security:X509Filesystem" xmlns:security="urn:mace:shibboleth:
2.0:security" id="IdPtoLDAPCredential">
        <security:PrivateKey>%{idp.attribute.resolver.LDAP.authenticationKey}</security:PrivateKey>
        <security:Certificate>%{idp.attribute.resolver.LDAP.authenticationCertificate}</security:Certificate>
    </StartTLSAuthenticationCredential>
    <ConnectionPool
        minPoolSize="%{idp.pool.LDAP.minSize}"
        maxPoolSize="%{idp.pool.LDAP.maxSize}"
        blockWaitTime="%{idp.pool.LDAP.blockWaitTime}"
        expirationTime="%{idp.pool.LDAP.expirationTime}"
        validatePeriodically="%{idp.pool.LDAP.validatePeriodically}"
        validateTimerPeriod="%{idp.pool.LDAP.validatePeriod}"
        validateDN="%{idp.pool.LDAP.validateDN}"
        validateFilter="%{idp.pool.LDAP.validateFilter}"
        failFastInitialize="%{idp.pool.LDAP.failFastInitialize}"/>
    <ResultCache
        elementTimeToLive="%{idp.cache.LDAP.timeToLive}"
        maximumCachedElements="%{idp.cache.LDAP.cacheSize}"/>
</DataConnector>
```

Example of a springResources file

```
<!-- In this case the definition would be <DataConnector" xsi:type="LDAPDatabase" springResources="....." /> --
>
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:p="http://www.springframework.org/schema/p"
       xmlns:context="http://www.springframework.org/schema/context"
       xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema
/beans/spring-beans-3.1.xsd
                           http://www.springframework.org/schema/context http://www.springframework.org/schema
/context/spring-context-3.1.xsd">

    <!-- each bean is set on LDAPDataConnector -->
    <bean class="org.ldaptive.pool.PooledConnectionFactory">
        <property name="connectionPool">
            <bean class="org.ldaptive.pool.BlockingConnectionPool" init-method="initialize" p:
blockWaitTime="${connectionPool.blockWaitTime}">
                <constructor-arg index="0">
```

```
                    <bean class="org.ldaptive.pool.PoolConfig"
                        p:minPoolSize="%{idp.pool.LDAP.minSize}"
                        p:maxPoolSize="%{idp.pool.LDAP.maxSize}"
                        p:validateOnCheckIn="%{idp.pool.LDAP.validateOnCheckin}"
                        p:validateOnCheckOut="%{idp.pool.LDAP.validateOnCheckout}"
                        p:validatePeriodically="%{idp.pool.LDAP.validatePeriodically}"
                        p:validatePeriod="%{idp.pool.LDAP.validatePeriod}" />
                </constructor-arg>
                <constructor-arg index="1">
                    <bean class="org.ldaptive.DefaultConnectionFactory">
                        <property name="connectionConfig">
                            <bean class="org.ldaptive.ConnectionConfig" p:ldapUrl="%{idp.attribute.resolver.
LDAP.ldapURL}"
                                p:connectTimeout="%{idp.attribute.resolver.LDAP.connectTimeout}"
                                p:responseTimeout="%{idp.attribute.resolver.LDAP.responseTimeout}"
                                p:useSSL="%{idp.attribute.resolver.LDAP.useSSL}"
                                p:useStartTLS="%{idp.attribute.resolver.LDAP.useStartTLS}">
                                <property name="connectionInitializer">
                                    <bean class="org.ldaptive.BindConnectionInitializer"
                                        p:bindDn="%{idp.attribute.resolver.LDAP.bindDN}"
                                        p:bindCredential="%{idp.attribute.resolver.LDAP.bindDNCredential}" />
                                </property>
                                <property name="sslConfig">
                                    <bean class="org.ldaptive.ssl.SslConfig">
                                        <property name="credentialConfig">
                                            <bean class="org.ldaptive.ssl.X509CredentialConfig"
                                                p:trustCertificates="%{idp.attribute.resolver.LDAP.
trustCertificates}"
                                                p:authenticationCertificate="%{idp.attribute.resolver.LDAP.
authCertificate}"
                                                p:authenticationKey="%{idp.attribute.resolver.LDAP.authKey}" />
                                        </property>
                                    </bean>
                                </property>
                            </bean>
                        </property>
                    </bean>
                </constructor-arg>
                <property name="validator">
                    <bean class="org.ldaptive.pool.SearchValidator">
                        <property name="searchRequest">
                            <bean class="org.ldaptive.SearchRequest">
                                <constructor-arg value="%{idp.pool.LDAP.validatorBaseDN}" />
                                <constructor-arg value="%{idp.pool.LDAP.validatorFilter}" />
                            </bean>
                        </property>
                    </bean>
                </property>
                <property name="pruneStrategy">
                    <bean class="org.ldaptive.pool.IdlePruneStrategy"
                        p:prunePeriod="%{idp.pool.LDAP.prunePeriod}"
                        p:idleTime="%{idp.pool.LDAP.idleTime}" />
                </property>
            </bean>
        </property>
    </bean>

    <bean class="org.ldaptive.SearchExecutor"
        p:baseDn="%{idp.attribute.resolver.LDAP.baseDN}"
        p:returnAttributes="%{idp.attribute.resolver.LDAP.returnAttributes}" />

    <bean id="cacheBuilder" class="com.google.common.cache.CacheBuilder" factory-method="from">
        <constructor-arg value="expireAfterAccess=10s,maximumSize=25" />
    </bean>

    <bean id="cache" class="com.google.common.cache.Cache" factory-bean="cacheBuilder" factory-method="build" />

    <bean class="net.shibboleth.idp.attribute.resolver.dc.ldap.impl.TemplatedExecutableSearchFilterBuilder"
        p:templateText="%{idp.attribute.resolver.LDAP.searchFilter}" p:velocityEngine-ref="shibboleth.
VelocityEngine"
        init-method="initialize" />
```

```
</beans>
```

Attributes

Any of the common attributes can be specified in addition to the following:

Name	Req?	Type	Default	Description
Commonly Used Attributes				
`ldapURL`	Y	Space-delimited list of URLs		URL(s) to the LDAP server. Each listed URL is tried according to the `connectionStrategy`.
`baseDN`	Y	String		Base DN from which the LDAP search will be executed.
`principal`	Y	String		User name (service DN) that the connector will use to bind to the LDAP directory
`principalCredential`	Y	String		Password used to authenticate as the principal (service DN)
`lowercaseAttributeNames`		Boolean	false	Whether all attribute IDs from the LDAP should be lower-cased. This can be important since Shibboleth attribute IDs are case-sensitive while LDAP attribute IDs are not
`trustFile` 3.3	N	String (filename)		Path to a file containing the X.509 trust information to use when connecting to the directory over LDAPS or startTLS. Replaces the deprecated use of `<StartTLSTrustCredential>`
Other Attributes				
`connectionStrategy`		One of `ROUND_ROBIN, DEFAULT, RANDOM, ACTIVE_PASSIVE`	`ACTIVE_PASSIVE`	If Multiple URLs were provided as the ldapURL this describes how each URL will be processed. • DEFAULT - Indicates that the default JNDI provider behavior will be used • ACTIVE_PASSIVE (default value) - Indicates that the first LDAP URL will be used for every request unless it fails and then the next LDAP URL will be used. • ROUND_ROBIN - Indicates that for each new connection the next LDAP url in the list (circling back to the start of the list when the end is reached) will be used • RANDOM - Indicates that for each new connection a random LDAP url will be selected
`authenticationType`		One of `DIGEST_MD5, CRAM_MD5, GSSAPI`		A SASL mechanism to be used for the Bind operation
`searchScope`		One of `SUBTREE, ONELEVEL, OBJECT`	SUBTREE	The scope of the search. • SUBTREE: The entire LDAP directory subtree below the search baseDN will be searched. • ONELEVEL: Only the immediate children of LDAP object corresponding to the search baseDN will be searched. • OBJECT: Only the LDAP object itself is searched.
`derefAliases` 3.4.5		One of NEVER, SEARCHING, FINDING, ALWAYS	NEVER	How aliases should be dereferenced. See the Oracle JNDI docs for more details on these options.
`useStartTLS`		Boolean	false	Whether to use startTLS when connecting to the LDAP
`searchTimeLimit`		number of milliseconds	3000	Length of time in milliseconds that a search operation should execute; a value of 0 means execute indefinitely; when time limit arrives the result will contain any entires returned up to that point
`maxResultSize`		Integer	1	Maximum number of entries to include in the search result; a value of 0 means includes all entries
`noResultIsError`		Boolean	false	Whether an empty result set is an error
`multipleResultsIsError`		Boolean	false	Whether a result set with more than one result is an error
`templateEngine`		Bean ID		The ID of a Spring bean defining a org.apache.velocity.app.VelocityEngine
`mappingStrategyRef`		Bean ID		The ID of a Spring bean defining a MappingStrategy<org.ldaptive.SearchResult>
`executableSearchBuilderRef` 3.4		Bean ID		The ID of a Spring bean defining an ExecutableSearchBuilder<ExecutableSearchFilter>
`validatorRef` 3.2		Bean ID		Bean ID of a Validator to control what constitutes an initialization failure (set this to "shibboleth.NonFailFastValidator" to bypass connection attempt at config load time)
`connectTimeout`		Integer	3000 (PT3S) 3.3	Connection timeout in milliseconds

`responseTimeout` 3.3		Duration	PT3S	Time to wait for response
`authCert` 3.4		String (filename)		Path to the file containing the X. 509 certificate to provide when connecting to the directory over LDAPS or startTLS
`authKey` 3.4		String (filename)		Path to the file containing the X. 509 key to provide when connecting to the directory over LDAPS or startTLS
`authKeyPassword` 3.4		String		Password to use on the `authKey` file
Deprecated Attributes				
`poolInitialSize`				Use the `<ConnectionPool>` element
`poolMaxIdleSize`				Use the `<ConnectionPool>` element

The deprecated attributes are not available in the `urn:mace:shibboleth:2.0:resolver` namespace'd type, but remain available in the deprecated `urn:mace:shibboleth:2.0:resolver:dc` namespace

Child Elements

Any of the common child elements can be specified, in addition to the following:

Name	Cardinality	Description
`<FilterTemplate>`	0 or 1	The template of the search filter to be sent to the LDAP directory server
`<ReturnAttributes>`	0 or 1	A list of attributes to be returned from the LDAP directory server; this may help the server respond more quickly
`<BinaryAttributes>` 3.4.5	0 or 1	A list of attributes whose values contain binary data and must be base64 encoded; format is a space delimited list of attribute names
`<LDAPProperty>`	0 or more	Custom LDAP configuration properties
`<StartTLSTrustCredential>`	0 or 1	X.509 trust information to use when connecting to the directory over LDAPS or startTLS, DEPRECATED in favor of the `trustFile` attribute
`<StartTLSAuthenticationCredential>`	0 or 1	X.509 client authentication information to provide when connecting to the directory over LDAPS or startTLS, DEPRECATED in favor of the `authCert, authKey` and `authKeyPassword` attributes
`<ConnectionPool>`	0 or 1	Describes how the LDAP connection may be pooled
`<Column>`	0 or more	Allows for remapping of LDAP Attributes into alternately named IdPAttributes within the resolver
`<ResultCache>`	0 or 1	The definition of how results should be cached
`<ResultCacheBean>`		The definition of how results should be cached as an externally defined com.google.common.cache.Cache<String,Map<String,IdPAttribute>>, the Spring bean ID of which is supplied as the content of the element

Externally (Spring) Defined Content

If the `springResource` or `springResourceRef` attributes are specified, then the configuration of the data connector bean is delegated to the supplied resources. The system will create a factory for an LDAPDataConnector object, and look for beans in the Spring resource(s) supplied that match the types of properties supported by that type and its parent classes. Note that since these are not public, but implementation classes, they are subject to change, which creates some risk during non-patch upgrades, so you must take additional precautions to use this feature.

In practice, the data connector may be supplied with Spring-defined beans of the following types:

- `org.ldaptive.ConnectionFactory`
- `org.ldaptive.SearchExecutor`
- `com.google.common.cache.Cache<String,Map<String,IdPAttribute>>`
- `Validator`
- `MappingStrategy<SearchResult>`
- `ExecutableSearchBuilder<ExecutableSearchFilter>`