



Understanding the LDAP Binding Component



Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 820-7855
June 2009

Copyright 2009 Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, CA 95054 U.S.A. All rights reserved.

Sun Microsystems, Inc. has intellectual property rights relating to technology embodied in the product that is described in this document. In particular, and without limitation, these intellectual property rights may include one or more U.S. patents or pending patent applications in the U.S. and in other countries.

U.S. Government Rights – Commercial software. Government users are subject to the Sun Microsystems, Inc. standard license agreement and applicable provisions of the FAR and its supplements.

This distribution may include materials developed by third parties.

Parts of the product may be derived from Berkeley BSD systems, licensed from the University of California. UNIX is a registered trademark in the U.S. and other countries, exclusively licensed through X/Open Company, Ltd.

Sun, Sun Microsystems, the Sun logo, the Solaris logo, the Java Coffee Cup logo, docs.sun.com, Java, and Solaris are trademarks or registered trademarks of Sun Microsystems, Inc. or its subsidiaries in the U.S. and other countries. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

The OPEN LOOK and SunTM Graphical User Interface was developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

Products covered by and information contained in this publication are controlled by U.S. Export Control laws and may be subject to the export or import laws in other countries. Nuclear, missile, chemical or biological weapons or nuclear maritime end uses or end users, whether direct or indirect, are strictly prohibited. Export or reexport to countries subject to U.S. embargo or to entities identified on U.S. export exclusion lists, including, but not limited to, the denied persons and specially designated nationals lists is strictly prohibited.

DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

Contents

| | |
|---------------------------------------------------------------|----------|
| Understanding the LDAP Binding Component | 5 |
| About the LDAP Binding Component | 5 |
| Functional Architecture of the LDAP Binding Component | 6 |
| Supported LDAP Servers | 6 |
| About LDAP | 7 |
| Entries, Attributes and Values | 7 |
| LDAP Directory Structure | 8 |
| Distinguished Names and Relative Distinguished Names | 8 |
| LDAP Service and LDAP Client | 9 |
| Terms and Definitions | 10 |
| Supported Features in the LDAP Binding Component | 11 |
| Supported LDAP Functions | 12 |
| Searching the LDAP Directory | 12 |
| OBJECT_SCOPE Search Method | 13 |
| ONELEVEL_SCOPE Search Method | 14 |
| SUBTREE_SCOPE Search Method | 15 |
| Security for LDAP Transactions | 16 |
| LDAP BC WSDL Configuration | 17 |
| Viewing the LDAP WSDL Document | 17 |
| Service Level WSDL Elements | 18 |
| Binding Level WSDL Elements | 20 |
| Runtime Configuration | 21 |
| Accessing the LDAP Binding Component Runtime Properties | 21 |
| LDAP Binding Component Runtime Properties | 22 |
| Application Variables | 25 |
| Application Configurations | 25 |

Understanding the LDAP Binding Component

The topics in this document provides information about the LDAP Binding Component.

What You Need to Know

These topics provide information about the functional behavior of LDAP Binding Component.

- [“About the LDAP Binding Component” on page 5](#)
- [“About LDAP” on page 7](#)
- [“Supported Features in the LDAP Binding Component” on page 11](#)
- [“Supported LDAP Functions” on page 12](#)
- [“Searching the LDAP Directory” on page 12](#)
- [“Security for LDAP Transactions” on page 16](#)
- [“LDAP BC WSDL Configuration” on page 17](#)
- [“Runtime Configuration” on page 21](#)
- [“Application Variables” on page 25](#)
- [“Application Configurations” on page 25](#)

About the LDAP Binding Component

The LDAP BC enables data exchange with an LDAP directory on an LDAP server. By providing the connection to the LDAP server, the LDAP BC enables processes to search, compare, and modify an LDAP directory using the LDAP protocol. The Binding Component wraps the JBI interface to enable access to operations provided by the LDAP service.

The LDAP BC conforms to the JBI specification and enables LDAP server integration in a JBI environment. The binding component consists of two modules: the runtime JBI module and the design-time NetBeans plug-in. The LDAP BC runtime module implements interfaces defined in the JBI framework and interacts with the external LDAP server for managing the directory services. The runtime uses LDAP v3 to interact with the LDAP directory server.

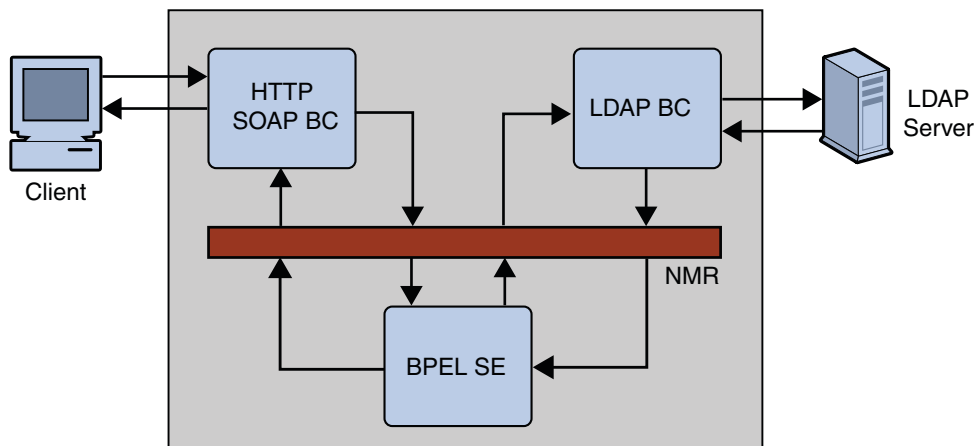
The design-time module provides visual configuration of the binding component. It also defines how other components can integrate with the LDAP BC inside the JBI framework.

Users can update properties for the component and incorporate LDAP components with others in service assemblies that are then deployed to an application server.

Functional Architecture of the LDAP Binding Component

The LDAP BC provides a comprehensive solution for configuring and connecting to the LDAP server within a JBI environment. The runtime component provides the physical connectivity between the Normalized Message Router in the JBI framework and the external LDAP server. The runtime component can act as a service provider, supporting Outbound services sending requests to the LDAP server from the JBI framework.

This component implements all the required interfaces from the JBI 1.0 specification. The diagram below illustrates the architecture of the LDAP BC.



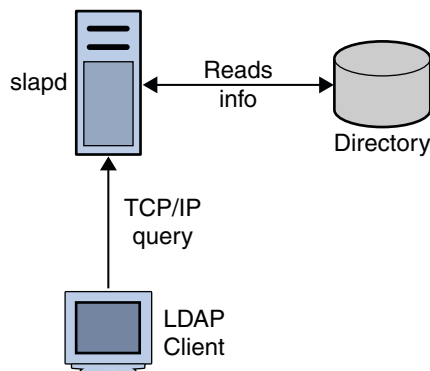
Supported LDAP Servers

LDAP Binding Component is certified to work with the following directory servers:

- Windows Server 2003 Active Directory
- Sun Java System Directory Server v6.3
- OpenLDAP 2.3.39
- OpenDS 1.0.0.

About LDAP

Lightweight Directory Access Protocol (LDAP) is an Internet protocol used to access information directories. A directory service is a distributed database application designed to manage the entries and attributes in a directory. LDAP allows clients to access different directory services based on entries. These LDAP entries are available to users and other applications based on access controls. LDAP runs over TCP/IP.



The Lightweight Directory Access Protocol (LDAP) Binding Component (BC) is a comprehensive solution for interacting with an LDAP Directory running on an LDAP server. The design time component of the LDAP BC is a NetBeans module that allows configuration of the Binding Component. The runtime is based on Java EE and JBI. It implements all the necessary interfaces available in the JBI specification.

The following topics provide information about LDAP and the directory structure:

- [“Entries, Attributes and Values” on page 7](#)
- [“LDAP Directory Structure” on page 8](#)
- [“Distinguished Names and Relative Distinguished Names” on page 8](#)
- [“LDAP Service and LDAP Client” on page 9](#)
- [“Terms and Definitions” on page 10](#)

Entries, Attributes and Values

An LDAP directory has entries that contain information pertaining to entities. Each attribute has a name and one or more values. The names of the attributes are mnemonic strings, such as `cn` for common name, or `mail` for email address.

For example, a company may have an employee directory. Each entry in the employee directory represents an employee. The employee entry contains such information as the name, email address, and phone number, as shown in the following example:

cn: John Doe

mail: johndoe@sun.com

mail: jdoe@stc.com

telephoneNumber: 471-6000 x.1234

Each part of the descriptive information, such as an employee's name, is known as an attribute. In the example above, the Common Name (cn) attribute, represents the name of the employee. The other attributes are mail and telephoneNumber. Each attribute can have one or more values. For example, an employee entry might contain a mail attribute whose values are johndoe@sun.com and jdoe@stc.com. In the example above, the mail attribute contains two mail values.

LDAP Directory Structure

The organization of a directory is a tree structure. The topmost entry in a directory is known as the root entry. This entry normally represents the organization that owns the directory. Entries at the higher level of hierarchy represent larger groupings or organizations. Entries under the larger organizations represent smaller organizations that make up the larger ones. The leaf nodes (or entries) of the tree structure represent the individuals or resources.

Distinguished Names and Relative Distinguished Names

An entry is made up of a collection of attributes that have a unique identifier called a Distinguished Name (DN). A DN has a unique name that identifies the entry at the respective hierarchy. In the example above, John Doe and Jane Doe are different common names (cn) that identify different entries at that same level.

A DN is also a fully qualified path of names that trace the entry back to the root of the tree. For example, the distinguished name of the John Doe entry is:

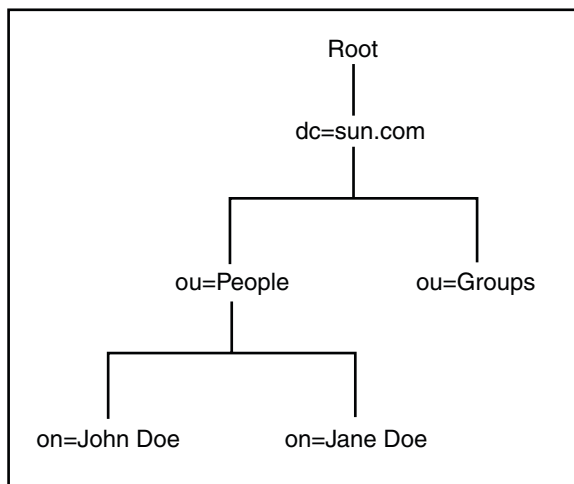
cn=John Doe, ou=People, dc=sun.com

A Relative Distinguished Name (RDN) is a component of the distinguished name. DNs describe the fully qualified path to an entry; RDNs describe the partial path to the entry relative to another entry in the tree.

For example, cn=John Doe, ou=People is a RDN relative to the root RDN dc=sun.com.

The following figure illustrates an example of an LDAP directory structure with distinguished names and relative distinguished names.

LDAP Directory Structure



LDAP Service and LDAP Client

A Directory Service is a distributed database application designed to manage the entries and attributes in a directory. A directory service also makes the entries and attributes available to users and other applications. OpenLDAP server is an example of a directory service. Other directory services include Sun Active Directory Service and Microsoft Active Directory.

A directory client uses the LDAP protocol to access a directory service. A directory client may use one of several client APIs available in order to access the directory service.

Terms and Definitions

Schema

A set of rules that describes the nature of data is stored. Schemas helps maintain consistency and quality data, and reduces duplication of data. The object class attribute determines the schema rules an entry must follow. Schemas define the following:

- Required attributes
- Allowed attributes
- The method to compare attributes
- Limits to what the attribute can store (for example, restricting the attribute to an integer)
- Restrictions on what information is stored (prevents duplication)

Attribute Abbreviation

The following are common attribute abbreviations used in LDAP:

- User id : uid
- Common Name ; cn
- Surname : sn
- Location : l
- Organizational Unit : ou
- Organization : o
- Domain Component : dc
- State : st
- Country : c
- Street address : street

Search Filters

Criteria for attributes that must satisfy for an entry to be returned. Search filters typically use a base DN, which is the base object entry the search is relative to. They also use prefix notations. LDAP uses the following standards:

- LDAP String Representation of Search Filters
- LDAPv3 Search Filters

The following search operators are supported:

- AND : &
- OR : |
- NOT : !
- Approximately equal : ~=
- Greater than or equal : >=
- Less than or equal : <=
- Any : *

Below are some examples of search filters:

```
(objectclass=posixAccount)
(cn=Mickey M*)
(|(uid=fred)(uid=bill))
(&(|(uid=jack)(uid=jill)(objectclass=posixAccount))
```

Supported Features in the LDAP Binding Component

The LDAP BC allows anonymous and authenticated connections. Users can perform numerous tasks when connected to an external LDAP system, including adding an entry, attribute, or value. Users can also modify, delete and search for a value or attribute entry.

The following features are supported in the LDAP Binding Component.

- Standard JBI Binding Component functions, including component and service unit lifecycle management, configuration at both installation and runtime, service provisioning and consumption, and so on.
- Standard LDAP functions, such as anonymous connections, adding and modifying entries, adding and removing values, searching for entries, and so on.
- Multiple security protocols, including SSL, TLS, TLS on Demand, keystore and truststore, and credential files.
- Multiple operations in the same WSDL file. The LDIF parser implementation allows you to create a WSDL from an LDAP LDIF file. The WSDL can be created with all four operations in the same WSDL.
- Automatic WSDL file generation using an LDAP WSDL Wizard to help you through the setup steps.
- Design-time WSDL validations.

- LDAP WSDL extensibility elements to further configure the WSDL document.
- Application configurations and variables in the runtime environment.
- Thread management.
- Connection management, including connection pools.
- Fault handling.
- Component logging and monitoring.
- Message exchange redelivery capability and graceful recovery.
- Dynamic addressing.

Supported LDAP Functions

The LDAP BC supports standard LDAP functions, including the following:

- **Anonymous Connection** - Connections that are not categorized as authenticated, trusted, suspect, or blocked.
- **Add Entry Node** - Adds entries to a directory using a variety of available options. Specify the name of the entry to add (RDN relative to the initial context), an entry with its attributes, and values for each attribute.
- **Add Attribute Value** - Modifies the values or attributes.
- **Add Value** - Adds any value to any attribute of an entry.
- **Remove Attribute** - Removes attributes from an entry.
- **Remove Value** - Removes values from any attribute of an entry.
- **Replace Value** - Replaces all the existing values of any attribute with any new value for an entry.
- **Search** - Searches for an entry or multiple entries in multiple levels of the LDAP directory.
- **Search Referral** - Searches for an attribute among many LDAP servers.

Searching the LDAP Directory

You can search for an entry or multiple entries of the LDAP directory. The LDAP BC supports the following search attributes:

- Use a *search filter* to specify the context or the first entry for the search, the scope of the search, or any other search criteria and the boundaries to which the search is limited.
- Use *page control* to specify a collection of set controls.
- Use *sort* to request for the search values to be sorted as per the specified attributes. Set the Sort Attributes field with a pipe (|) separated character string consisting of attributes to use sort control.

For example, set `SortAttributes` with the string `cn|mail` to sort entries first by `cn` and second by `mail`.

You can search for an attribute among many LDAP servers by using the search referral attribute to set up an LDAP referral to another LDAP server. This means after the search fails to locate the search string on one server, it automatically searches over the referred server. Use the RCF command line utility to generate the credential file that contains the appropriate referral credentials. Search referral supports the following attributes:

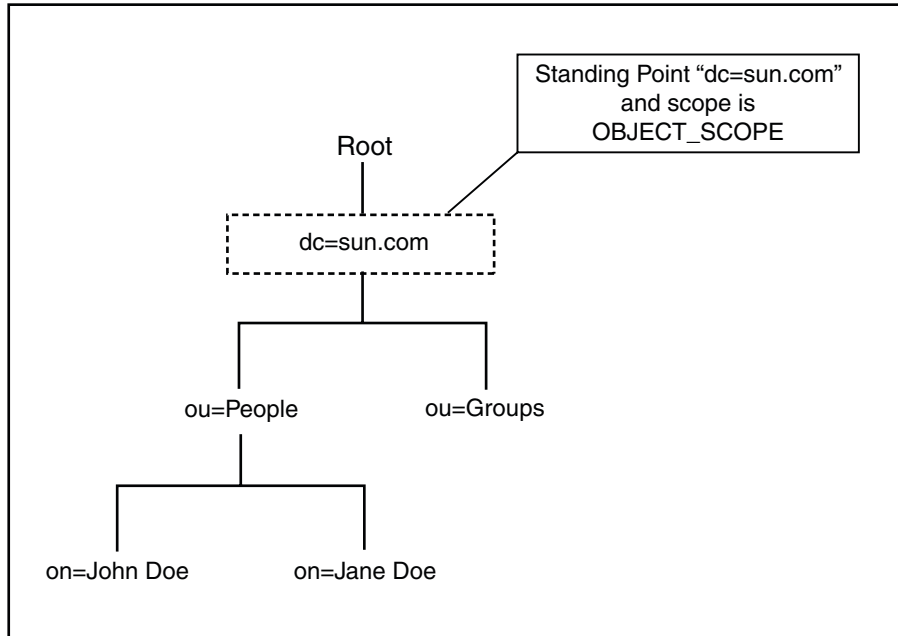
- `Ignore` – Ignores the referral server.
- `Follow` – Connects to the referred system and continues the search operation.
- `Throw` – Generates a referral exception, which the client can catch and initiate any action.

You can use any of the following LDAP search methods:

- [“OBJECT_SCOPE Search Method” on page 13](#)
- [“ONELEVEL_SCOPE Search Method” on page 14](#)
- [“SUBTREE_SCOPE Search Method” on page 15](#)

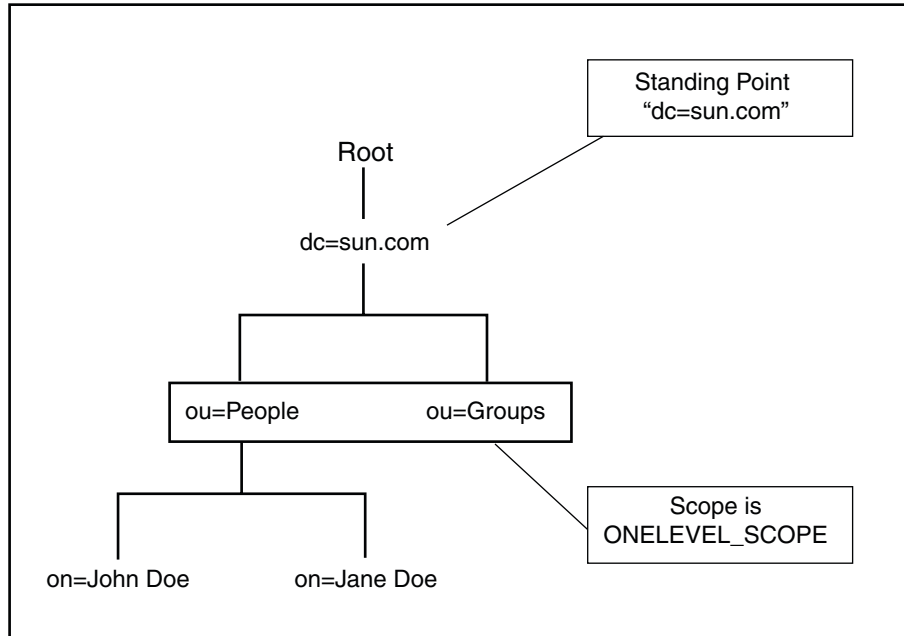
OBJECT_SCOPE Search Method

The `OBJECT_SCOPE` method defines the search method only within the named object that is defined with `ContextName`. The object scope compares the named object for some particular attribute or value.



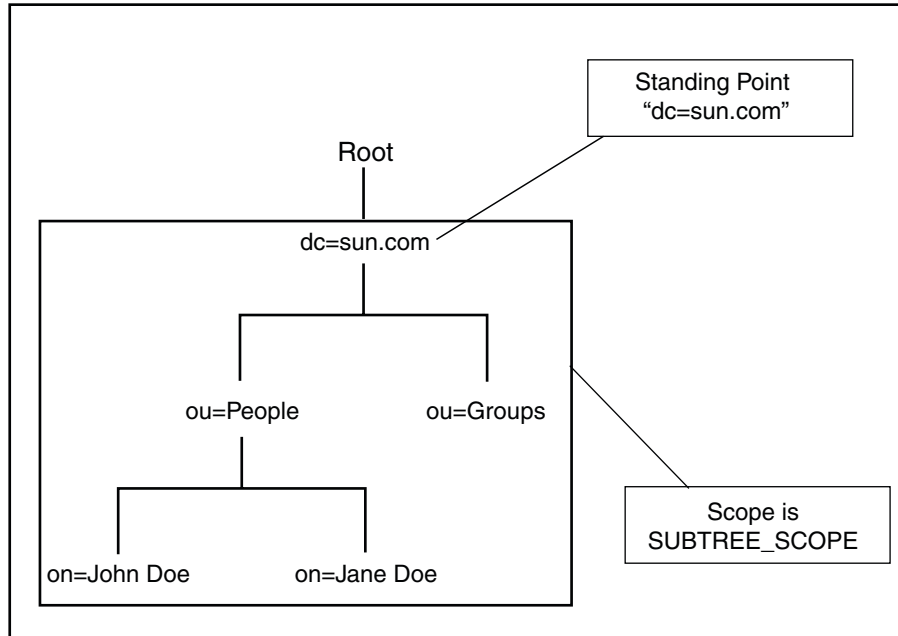
ONELEVEL_SCOPE Search Method

The ONELEVEL_SCOPE method defines the search method for entries that are one level below the named object.



SUBTREE_SCOPE Search Method

The SUBTREE_SCOPE method defines the search method for all entries starting from the named object and all descendants below the named object.



Security for LDAP Transactions

The LDAP server stores user names and passwords, so all transactions have to be secure. You can configure the following security options using the LDAP Binding Component:

- Secure Socket Layer (SSL)
Secure Socket Layer (SSL) is a cryptographic protocol that provides privacy and data integrity for communications over TCP/IP networks such as the Internet.
- Transport Layer Security (TLS)
Transport Layer Security (TLS) is a cryptographic protocol that provides privacy and data integrity for communications over TCP/IP networks such as the Internet.
- TLS on Demand
Selecting this option allows users to establish an SSL connection on demand.
 - Use the `startTLS` function to initiate a secure SMTP connection between two servers using the Secure Sockets Layer (SSL) (also known as TLS). Once the connection is established all future communication between the two servers is encrypted.
 - Use the `stopTLS` function to stop an SMTP connection between two servers using the Secure Sockets Layer (SSL) (also known as TLS).
- KeyStore and TrustStore Management

The Keystore is used for key or certificate management when establishing SSL connections. The TrustStore is used for CA certificate management when establishing SSL connections.

- **Credential File Management**

The Credential File Management feature allows users the credentials needed when authenticating logins other than anonymous login.

LDAP BC WSDL Configuration

When you use the New WSDL Document Wizard for the LDAP BC, it generates a WSDL document that includes all of the required elements and the configuration information you entered in the wizard. You can use the WSDL Editor to view and modify the configuration you defined. The elements you are most likely to configure are the *service level WSDL elements* and *binding level WSDL elements*. The following topics describe working with the WSDL document and the elements specific to the LDAP BC.

- [“Viewing the LDAP WSDL Document” on page 17](#)
- [“Service Level WSDL Elements” on page 18](#)
- [“Binding Level WSDL Elements” on page 20](#)

For an example of how to use the New WSDL Document Wizard for LDAP, see [“Creating a WSDL Document” in *Using the LDAP Binding Component in a Project*](#).

Viewing the LDAP WSDL Document

In the WSDL view of the WSDL Editor, the WSDL file appears as a tree component or a series of columns. The WSDL view has two subviews: tree view and column view. To switch between the subviews, use the buttons in the WSDL Editor toolbar. The main nodes in the WSDL view correspond to the major elements in a WSDL file.

- **Types:** This node enables you to import XML schemas and to add inline schemas.
- **Imports:** This node enables you to import WSDL files.
- **Messages:** This node enables you to create, edit, and delete messages.
- **Port Types:** This node enables you to create, edit, and delete port types.
- **Bindings:** This node enables you to create, edit, and delete bindings.
- **Services:** This node enables you to create, edit, and delete services.
- **Extensibility Elements:** This node enables you to add the following extensibility elements: partner link types, properties, and property aliases.

▼ **To Add Extension Attributes**

Some of the nodes in the WSDL view allow you to add extension attributes.

- 1 **Right-click the node and choose Add Extension Attribute.**
- 2 **Specify the name and namespace in the Add Extension Attribute dialog box.**
- 3 **Specify the value from the Properties window after adding the attribute.**
- 4 **Right-click the node and choose Remove Attributes to delete the attribute.**

Service Level WSDL Elements

The attributes of the service WSDL element configure LDAP directory security information for the LDAP BC. The attributes specify the connectivity to the LDAP directory *address* element *ldap:address*. These attributes specify login information, security protocols, authentication type, and so on.

When you create a WSDL file in the NetBeans IDE, the New WSDL Document Wizard generates the address service definition. You can then edit the attributes of the address service. The following table describes the attributes available for this service.

TABLE 1 LDAP Address Element Attributes

| Attribute | Description |
|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| location | The connection URL for the LDAP server in the format <code>ldap://hostname:port</code> . |
| principal | The LDAP principal (user name) needed when using an authentication method other than anonymous login. Use the fully qualified DN (Distinguished Name) of the user; for example, <code>CN=Administrator,CN=Users,DC=sun,dc=com</code> . |
| credential | The credentials (password) needed when using an authentication method other than anonymous login. |
| ssltype | <p>The type of SSL connection to use. Enter one of the following:</p> <ul style="list-style-type: none">■ None: A simple plain connection that does not use SSL.■ Enable SSL: Communication to the LDAP server uses an SSL secure communication channel.■ TLS On Demand: Communication to the LDAP server uses TLS on demand. <p>Note – If you use Enable SSL, the URL must point to a secure LDAP port.</p> |

TABLE 1 LDAP Address Element Attributes *(Continued)*

| Attribute | Description |
|--------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| authentication | The authentication method to be used. Enter one of the following: <ul style="list-style-type: none"> ■ None: Authentication is simple or not required. Make sure the LDAP server supports anonymous logins. ■ Simple: Authentication is based on a user name or password. If you select this option, you need to enter the user name in the principal property and the password in the credentials property. |
| protocol | The SSL protocol to use when establishing an SSL connection with the LDAP server. Enter TLS, TLSv1, SSLv3, SSLv2, or SSL |
| truststore | The path and name of the TrustStore file, which is used for CA certificate management when establishing SSL connections. |
| truststorepassword | The TrustStore password for accessing the TrustStore used for CA certificate management when establishing SSL connections. |
| truststoretype | The TrustStore type used for CA certificate management. If no type is specified, the applications uses JKS as the default type. |
| keystore | The path and name of the KeyStore file. The KeyStore is used for key/certificate management when establishing SSL connections. |
| keystorepassword | The KeyStore password for accessing the KeyStore used for key/certificate management when establishing SSL connections. |
| keystoreusername | The user name for accessing the keystore when establishing SSL connections. |
| keystoretype | The default keystore type, which is used for key/certificate management when establishing SSL connections. If no type is specified, the application uses JKS as the default type. |
| tlssecurity | An indicator of whether TLS security is enabled. Enter NO if TLS security is not used; enter YES if it is used. |

The following example illustrates the LDAP service element:

```
<service name="LDAPService"
  <wsdl:port name="LDAPPort" binding="tns:LDAPBinding">
    <ldap:address location="ldap://ldapServer1:389"
      principal = "cn=Manager,dc=sun,dc=com"
      credential = "admin"
      truststorepassword = "trustadmin"
      truststoretype = "JKS"
      keystore = "C:\security\ldap\keystore.jks"
      keystorepassword = "keystoreadmin"
      keystoreusername = "keystore"
      keystoretype = "JKS"
      tlssecurity = "NO"
```

```
        ssltype = "Enable SSL"
        authentication = "Simple"
        truststore = "C:\security\ldap\trust.jks"
        protocol = "SSL"
    />
</wsdl:port>
</service>
```

Binding Level WSDL Elements

The LDAP Binding Component binding level WSDL elements include the *binding*, *operation*, and *message* extensibility elements.

Binding elements define the file transport-specific information for operations and messages.

LDAP Binding Element

The LDAP binding extensibility element allows the association of a binding to be LDAP protocol specific. When you create a WSDL file for a BPEL project in the NetBeans IDE, the New WSDL Document Wizard generates the LDAP binding definition, which includes a name you specify and a type that is generated by the wizard.

The following example illustrates the LDAP binding element:

```
<binding name="LDAPBinding" type="tns:LDAPPortType">
    <ldap:binding/>
    ...
</binding>
```

LDAP Operation Element

The LDAP operation element defines the supported operations. For the LDAP Binding Component the operations that can be supported include the following:

- searchRequest
- updateRequest
- compareRequest
- insertRequest
- deleteRequest
- addConnectionRequest

The following example illustrates the LDAP operation element:

```
<binding name="LDAPBinding" type="tns:LDAPPortType">
    <ldap:binding/>
    <wsdl:operation name="LDAPSearchOperation">
        <ldap:operation type="searchRequest"/>
        ...
    </wsdl:operation>
</binding>
```

LDAP Output Element

The LDAP output element extends the binding element to specify properties associated with writing output messages. In the NetBeans IDE, select a `ldap:output()` element to view and modify the output properties. The following table describes the available output properties.

TABLE 2 LDAP Output Element Properties

| Property | Description |
|-----------------------------|----------------------------------------------------------------------------|
| <code>returnPartName</code> | The message part name that is returned. This is used in search operations. |
| <code>attributes</code> | A list of attributes to be retrieved. |

Runtime Configuration

The LDAP Binding Component's runtime properties can be configured from the NetBeans IDE, or from a command prompt (command line interface) during installation. The LDAP BC runtime properties apply to all instances of the binding component across the domain.

The following topics describe the runtime properties of the LDAP BC and how to access them:

- [“Accessing the LDAP Binding Component Runtime Properties” on page 21](#)
- [“LDAP Binding Component Runtime Properties” on page 22](#)

Accessing the LDAP Binding Component Runtime Properties

You can view and modify the runtime properties from the Service window of the NetBeans IDE. The LDAP BC must be started before you can modify any of the properties.

▼ To Access the LDAP Binding Component Runtime Properties

- 1 From the Services tab of the NetBeans IDE, expand the Servers node.
- 2 If the application server is not started, right-click the server name and select Start.
- 3 Under the application server, expand the JBI→ Binding Components nodes, and then right-click `sun-ldap-binding`.
- 4 Click Start.

5 Right-click `sun-ldap-binding` again, and then click **Properties.**

The LDAP BC Properties window appears. You can also select the binding component to display the Properties panel in the right side of the IDE.

Edit the properties as needed. To apply any changes you make to the Application Configuration and Application Variables properties, shut down and restart the affected composite applications. Any changes in the Number of Threads property and to the Logger settings are applied dynamically without restarting the LDAP BC.

LDAP Binding Component Runtime Properties

The LDAP Binding Component runtime properties are categorized into the following types:

- [“General Properties” on page 22](#)
- [“Identification Properties” on page 22](#)
- [“Configuration Properties” on page 23](#)
- [“Statistics Properties” on page 24](#)
- [“Loggers Properties” on page 24](#)

General Properties

The following table lists and describes each general property for the LDAP BC. These properties are for reference purpose only and cannot be modified.

| Name | Description |
|-------------|-------------------------------------------------------------------------------|
| Description | The purpose of the LDAP Binding Component. |
| Name | The name of the LDAP Binding Component. |
| State | Indicates the state of the LDAP Binding Component, either Started or Stopped. |
| Type | Indicates the type of component (service-engine or binding-component). |

Identification Properties

The following table lists and describes each identification property for the LDAP BC. These properties are for reference purpose only and cannot be modified.

| Name | Description |
|--------------|-------------------------------------------------------------------------|
| Version | The LDAP Binding Component specification fully supported by this build. |
| Build Number | Date and time stamp for the current build. |

Configuration Properties

The following table lists and describes each configuration property for the LDAP BC. These properties are for reference purpose only and cannot be modified.

| Name | Description |
|--------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Number of Outbound Processor Threads | Specifies the maximum number of threads to process outbound LDAP client invocations concurrently. The value can be any integer from 1 to 10,000. |
| Number of Retries | The number of times the LDAP BC will attempt to connect to the LDAP server. |
| Time Between Retries | The period of time in milliseconds that the LDAP BC will wait between attempts to connect to the LDAP server. |
| Recovery Type | An action to take on recovery. The available options are ERROR, DELETE, and SUSPEND. |
| Allow Connection Pooling | An indicator of whether connection pooling is enabled. Select the check box to enable connection pooling for connecting to the LDAP server. |
| Connection Pool Preferred Size | The preferred number of connections to maintain concurrently for each connection identity. |
| Connection Pool Maximum Size | The maximum number of connections that will be maintained concurrently for each connection identity. |
| Maximum Idle Timeout for Pooled Connection | The maximum period of time in milliseconds that a connection can remain idle in the pool. |
| Connection Pool Protocol | <p>The type of protocol to use for the pooled connections. Choose from the following options:</p> <ul style="list-style-type: none"> ■ plain ssl: Pools connections that use simple or no authentication and pools that use SSL. ■ plain: Pools connections that use simple or no authentication. ■ ssl: Pools connections that use SSL connections. |
| Connection Pool Authentication | <p>The type of authentication to use for the pooled connections. Choose from the following options:</p> <ul style="list-style-type: none"> ■ none: No authentication is required. Use this for anonymous access. ■ simple: Authentication requires a user name and password. ■ none simple: Authentication can be simple or not required. ■ strong: A stronger authentication is required. |
| Allow Dynamic Endpoint | An indicator of whether dynamic endpoints are allowed. Select the check box to allow them. |

| Name | Description |
|---------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Application Configuration | Specifies the values for a Composite Application's endpoint connectivity parameters (normally defined in the WSDL service extensibility elements), and applies these values to a user-named endpoint ConfigExtension property. The Application Configuration property editor includes fields for all of the connectivity parameters that apply to that component's binding protocol. |
| Application Variables | <p>Specifies a list of name and value pairs for a given type. The application variable name can be used as a token for a WSDL extensibility element attribute in a corresponding binding.</p> <p>The Application Variables configuration property offers four variable types:</p> <ul style="list-style-type: none">■ String: Specifies a string value, such as a path or directory■ Number: Specifies a number value■ Boolean: Specifies a Boolean value■ Password: Specifies a password value |

Statistics Properties

Statistics properties include several different component activities including exchanges, errors, requests, replies, and so on. It lists component statistics that are collected for actions such as completed exchanges, activated endpoints, and up time. Running statistics are automatically collected and displayed

Loggers Properties

Loggers properties include several different component activities that can be recorded by the server .log. The Logger properties specify the user-designated level of logging for an event.

Each logger can be set to record information at any of the following levels:

- FINEST: messages provide highly detailed tracing
- FINER: messages provide more detailed tracing
- FINE: messages provide basic tracing
- CONFIG: provides static configuration messages
- INFO: provides informative messages
- WARNING: messages indicate a warning
- SEVERE: messages indicate a severe failure
- OFF: no logging messages

Application Variables

The binding component Application Variables property allows you to define a list of `name:value` pairs for a given stated type. The application variable name can be used as a token for a WSDL extensibility element attribute in a corresponding binding. For example, if you were defining an application variable for the hostname as `atlas.domain.com`, then the WSDL attribute would be `${atlas.domain.com}`. In the Application Variables property you would enter a String value of `atlas.domain.com` for the name, and the desired attribute as the value. When you deploy an application that uses application variables, any variable that is referenced in the application's WSDL is loaded automatically. If you attempt to start an application and an Application Variables value is not defined (no value is specified for the Application Variable) an exception is thrown.

The Application Variables configuration property supports four variable types:

- **String:** Specifies a string value, such as a path or directory.
- **Number:** Specifies a number value.
- **Boolean:** Specifies a Boolean value. The VALUE field is a checkbox (checked indicates true).
- **Password:** Specifies a password value. The password is masked and displays only asterisks.

Variables also allow greater flexibility for your WSDL files. For example, you can use the same WSDL file for different runtime environments by using application variables to specify system specific information. These values can then be changed from the binding component runtime properties as needed, for any specific environment.

To change a property when the application is running, change your Application Variable property value, then right-click your application in the Services window under Servers > GlassFish V2 > JBI > Service Assemblies, and click Stop in the popup menu. When you restart your project, your new settings will take effect.

Application Configurations

An Application Configuration Object (ACO) defines a set of values that can be used to override `ldap:address` attributes that are defined in the WSDL, such as `credentials`, `protocol`, `authentication`, and so on.

Using the Application Configuration property, you can configure the external connectivity parameters for applications you create, such as a service assembly, and without changing or rebuilding the application you can deploy the same application into a different system. For example, you could take an application that is running in a test environment and deploy it to a production environment without rebuilding the application.

The properties you can define for the application configuration are normally defined in the WSDL service extensibility elements. You then apply these values to a user-named endpoint ConfigExtension Property. The Application Configuration property editor includes fields for all

of the parameters that apply to the LDAP service element in the WSDL. These values override the WSDL defined connectivity attributes when your project is deployed. To change these connectivity parameters again, you simply change the values in the Application Configuration editor, then shutdown and start your Service Assembly to apply the new values.

To change a property when the application is running, change your Application Configuration property value, then right-click your application in the Services window under Servers > GlassFishV2 > JBI > Service Assemblies, and click Stop in the popup menu. When you restart your project, your new settings will take effect.