Novell Developer Kit

www.novell.com

LDAP JDBC DRIVER

June 21, 2006

Novell®

Legal Notices

Novell, Inc. makes no representations or warranties with respect to the contents or use of this documentation, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Novell, Inc. reserves the right to revise this publication and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes.

Further, Novell, Inc. makes no representations or warranties with respect to any software, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Novell, Inc. reserves the right to make changes to any and all parts of Novell software, at any time, without any obligation to notify any person or entity of such changes.

Any products or technical information provided under this Agreement may be subject to U.S. export controls and the trade laws of other countries. You agree to comply with all export control regulations and to obtain any required licenses or classification to export, re-export or import deliverables. You agree not to export or re-export to entities on the current U.S. export exclusion lists or to any embargoed or terrorist countries as specified in the U.S. export laws. You agree to not use deliverables for prohibited nuclear, missile, or chemical biological weaponry end uses. Please refer to www.novell.com/info/exports/ for more information on exporting Novell software. Novell assumes no responsibility for your failure to obtain any necessary export approvals.

Copyright © 2006 Novell, Inc. All rights reserved. No part of this publication may be reproduced, photocopied, stored on a retrieval system, or transmitted without the express written consent of the publisher.

Novell, 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 of the U.S. patents listed at http://www.novell.com/company/legal/patents/ and one or more additional patents or pending patent applications in the U.S. and in other countries.

Novell, Inc.
404 Wyman Street, Suite 500
Waltham, MA 02451
U.S.A.
www.novell.com

Online Documentation: To access the online documentation for this and other Novell products, and to get updates, see www.novell.com/documentation.

Novell Trademarks

For Novell trademarks, see the Novell Trademark and Service Mark list (http://www.novell.com/company/legal/trademarks/tmlist.html)

Third-Party Materials

All third-party trademarks are the property of their respective owners

Contents

Preface

The LDAP Java Database Connectivity (JDBC) driver is specifically designed to query and retrieve LDAP data via the SQL (Structured Query Language) language. This is a Type 4 JDBC driver which provides SQL access to LDAP enabled directories. The Type 4 indicates that the driver is written in Pure Java, and communicates in the database system's own network protocol. Because of this, the driver is platform independent; once compiled, the driver can be used on any system.

This guide contains the following sections:

- LDAP JDBC Driver
- Revision History

Audience

This guide is intended for developers who are not familiar with all of the components of the LDAP JDBC SDK.

Feedback

We want to hear your comments and suggestions about this manual and the other documentation included with this product. Please use the User Comments feature at the bottom of each page of the online documentation.

Additional Information

For the related developer support postings for LDAP JDBC Driver, see the Developer Support Forums. (http://developer.novell.com/ndk/devforums.htm)

Documentation Updates

For the most recent version of this guide, see the LDAP JDBC Driver NDK page (http://developer.novell.com/ndk/ldapjdbc.htm).

Documentation Conventions

In Novell documentation, a greater-than symbol (>) is used to separate actions within a step and items in a cross-reference path.

A trademark symbol (®, ™, etc.) denotes a Novell trademark. An asterisk (*) denotes a third-party trademark.

When a single pathname can be written with a backslash for some platforms or a forward slash for other platforms, the pathname is presented with a backslash. Users of platforms that require a forward slash, such as Linux* or UNIX*, should use forward slashes as required by your software.

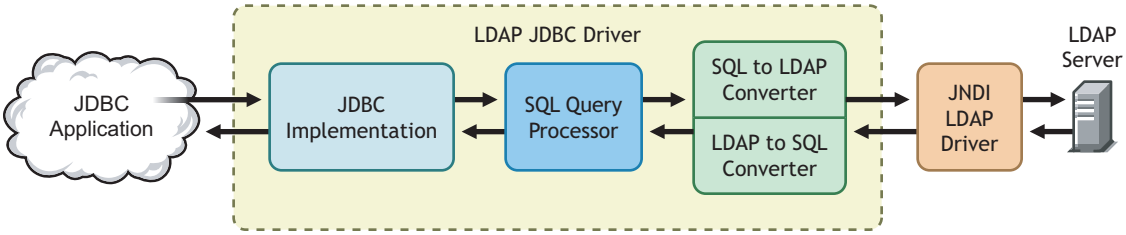
LDAP JDBC Driver

1

The LDAP JDBC Driver serves as an independent interface for extracting and reporting specified directory information for use in the Java applications that you use every day. It allows you to populate reports or import data into your custom programs.

The architecture behind the LDAP JDBC Driver consists of the Java application, the JVM, the LDAP JDBC Driver, the network, and the LDAP directory itself. The driver abstracts the directory tree into accessible relational database tables, which hides the complexity of the underlying directory syntax. Information is selected and ordered from the relational tables using standard SQL statements embedded into the application.

Figure 1-1 *LDAP JDBC Driver Architecture*



1.1 Requirements

The LDAP JDBC Driver has hardware and software requirements for the workstation you will run it from. Beyond the basic needs to run the driver, you should impose some resource restrictions because of the nature of the eDirectory database. See the following sections:

- “Resource Restrictions” on page 9
- “Hardware and Software Requirements” on page 10

1.1.1 Resource Restrictions

Since the LDAP JDBC Driver uses the resources of the workstation for memory and hard disk storage, you can ask for a report that generates more data than your workstation can handle in memory or store on the target drive.

An eDirectory database can contain a tremendous number of entries. For example, a container entry can have millions of entries.

Before requesting data on all entries of a certain class definition from the top of the tree or from a branch, you should have a rough estimate of how many entries you are requesting data about. If you are generating a report for thousands of entries, you may want to generate multiple reports rather than a single report.

Also, each entry in eDirectory contains multiple attributes. For example, the User class definition in eDirectory now has over 90 attributes. If you add other products which add attributes to the User definition, such as ZENworks or GroupWise, the number is even greater. The LDAP JDBC Driver reads the schema to determine the actual number of attributes a definition contains, and an SQL query for information on all attributes builds a table with all attributes, even those attributes which

have no values. Therefore, an SQL query for all attribute information about entries is not recommended. To keep a report to a reasonable size, you need to select specific attributes for the report. To determine which class definitions have tables and what attributes a class definition has, see “Mapping of LDAP Data to Relational Tables” on page 18.

You should also be aware that you can request a report that will take a tremendous amount of time to generate. The obvious cause is to request too much data. However, there are other causes. eDirectory allows data to be replicated and stored in multiple locations which makes it possible to have some data available only from a WAN link. Therefore, to generate reports quickly, you need to have some knowledge of the following:

- Whether replicas of all partitions are stored locally or over WAN links.
- Whether the LDAP server has been configured to automatically follow referrals. (The default setting in eDirectory 8.7 and higher is to return referrals rather than follow them.)

1.1.2 Hardware and Software Requirements

To install the LDAP JDBC Driver, you must have a workstation with the following configuration:

- JVM
- jre 1.4.2 and higher
- Java 1.4.2 and higher
- LDAP Extensions and Controls for JNDI (http://developer.novell.com/ndk/extjndi.htm)

In addition, you must have Novell eDirectory or Novell Directory Services (NDS) installed and running on the network (or another LDAP-aware directory). The eDirectory tree must have an LDAP server running eDirectory 8.7 or higher. If you want to generate effective rights reports, you need an LDAP server running on eDirectory 8.7. (For information on obtaining an evaluation copy, see Novell's download site (http://download.novell.com/).

1.2 Getting Started

The following sections explain what you need to do to use the LDAP JDBC Driver for the first time.

1.2.1 Accessing the Required Java Classes

A JDBC driver is a collection of Java classes that conform to the JDBC specification. For the LDAP JDBC Driver, these classes are contained in the jar file, `ldapjdbc.jar`. This jar file must be in the Java class path in order to use the LDAP JDBC Driver. The installation program installs the `ldapjdbc.jar` file on your machine in the `c:\Novell\Java\lib` directory unless you specify a path.

The LDAP JDBC Driver is implemented on top of the LDAP Extensions and Controls for JNDI. The LDAP Extensions and Controls for JNDI require the following jar files:

Table 1-1 *Jar File Description*

Jar File	Description
`jndi.jar`	JNDI 1.2 (Java Naming and Directory Interface) is a Java standard extension API. This is included with JDK 1.4
`ldap.jar`	LDAP (Light-weight Directory Access Protocol) provider allows JNDI to access eDirectory and other directories over LDAP. This is included with JDK 1.4.
`providerUtil.jar`	Provides utility functions for JNDI providers. This is included with JDK 1.4.
`novbp.jar`	Provides access to the eDirectory specific features through LDAP extensions. This is available by downloading "NDK: LDAP Extensions and Controls for JNDI".

1.2.2 Loading the Driver

One of the classes in a JDBC driver is designated as the main driver class. This main driver class must be loaded by your program or by the database tool you are using. The name of the main driver class for the LDAP JDBC Driver is com.novell.sql.LDAPDriver. This class is locate in the `ldapjdbc.jar` file.

Database tools provide a mechanism for you to specify this class name, and then the tool loads the driver. If you are programming directly to the LDAP JDBC Driver, use the following line of code to load the driver.

```
Class.forName("com.novell.sql.LDAPDriver");
```

This line of code causes the com.novell.sqo.LDAPDriver class to be loaded. When this class is loaded, it registers itself with the JDBC DriverManager class which makes it available for use.

1.2.3 Establishing a Connection

The JDBC protocol uses a URL for the following:

- Identify the JDBC driver
- Specify the database to connect to
- Pass property values to the driver

The URL is always used when programming directly to the driver using the JAVA language and is generally used by most database access tools. The URL for the LDAP JDBC Driver has the following format:

```
JDBC:LDAP:<ldap server>[;<property name>=<attribute name>]*
[;baseDN=<base DN>]
```

The following table describes each part of the URL.

Table 1-2 *Field Description of an LDAP URL*

URL Segment	Description
JDBC:LDAP	Specifies that you are using the JDBC protocol and a subprotocol, LDAP. In other words, the URL says you want a JDBC driver that can communicate with an LDAP server.
:<ldap server>	Identifies the LDAP server the driver should connect to. Its value should contain two forward slashes followed by either the server's IP address or the domain name.
[;<property name> = <property value>]*	Identifies an optional list of property name and value pairs with an equals sign separating the name and the value and a semicolon delimiting multiple name and value pairs.

The JDBC driver supports the following properties.

Table 1-3 *Properties Supported by the JDBC Driver*

Property Name	Description
user	The distinguished name of the user entry that will authenticate to the directory. The rights of this user will determine the access rights the driver has to the directory. If the user property is omitted an anonymous bind will be attempted.
password	The password for the user specified by the user property
baseDN	Specifies which container in the tree the driver will report from. Only entries below this container will be contained in the report.
useCleartext	By default the JDBC driver attempts to use an SSL connection to eDirectory. If an SSL connection cannot be established, the driver returns an error. If this property is specified with a value of "true," a non-secure, non-SSL connection will be established. Since such a connection will pass data and the user password on the network in clear text (not encrypted), this property should only be used in testing environments where the secrecy of passwords in not critical.
derefAlias	An Alias object in a directory points to another object in the directory, which can be a container, user object, or any other object in the tree.When you perform a search , the 'derefAlias' property specifies whether or not dreference(resolve) the Alias object to the object it points to. The 'derefAlias' property specifies how Aliases objects are resolved in any of the following four ways : • always - Always dereference aliases • never - Never dereferences aliases • finding - Dereferences aliases only during the name resolution • searching - Dereferences aliases only after the name resolution The default is "always".
saslExternal	The Simple Authentication and Security Layer (SASL) is a method to add authentication support to connection-based protocols. JDBC supports SASL EXTERNAL mechanism for authentication. To enable this, set the property "saslExternal=true" .

When programming in Java, a connection is created by passing the URL to the getConnection method of the DriverManager Class. Several examples are shown below. Note that there are three versions of the getConnection method.

Example 1

```
String url = "JDBC:LDAP://137.64.215.139" +
             ";user=cn=jbrown,o=acme" +
             ";password=myPassword" +
             ";baseDN=ou=enginerring,o=ACME";

Connection conn = DriverManager.getConnection(url);
```

Example 2

```
String url = "JDBC:LDAP://137.64.215.139" +
             ";baseDN=ou=enginerring,o=ACME";

Connection conn = DriverManager.getConnection(url,
                                              "cn=jbrown,o=acme",
                                               "myPassword");
```

Example 3

```
String url = "JDBC:LDAP://ldapserver.ACME.com" +

Properties props = new Properties();
props.put("user", "cn=jbrown,o=acme");
props.put("password", "myPassword");
props.put("baseDN", "ou=engineering,o=ACME");

Connection conn = DriverManager.getConnection(url, props);
```

Once a connection has been established you can use the connection object to execute one or more queries.

1.2.4 Obtaining More Information

For more information on using the JDBC interface, see the following sites:

- Sun's JDBC web site (http://java.sun.com/products/jdbc/). It contains specifications, white papers, and tutorials.
- Novell's NDK web site (http://developer.novell.com/ndk/doc/samplecode/ldapjdbc_sample/index.htm). It contains sample programs available for this driver.

1.3 eDirectory and LDAP Integration

The LDAP JDBC Driver uses the LDAP protocol to communicate with the directory. The LDAP names for object class and attribute definitions are not always the same as the corresponding eDirectory names. For example, the default in eDirectory 8.7 is to map the LDAP inetOrgPerson class to the eDirectory User class. Since the driver maps class definition names to SQL table names, and attribute names to SQL column names, the table and column names used by the driver reflect

LDAP names. If you are only familiar with the eDirectory names for the classes and attributes, see the *Developer Kit* for the corresponding LDAP names.

Name variations are just one of the differences you will need to be aware of when using JDBC with LDAP. You also need to understand:7

1.3.1 LDAP Authentication

To create meaningful reports with JDBC, you must have Read rights to the information you want to included in the report. This generally means that you need to log in as the supervisor of the subtree or container on which you are going to run reports. Even if you log in as the supervisor of the root container, remember the supervisor of the root container can be denied rights to portions of the eDirectory tree.

An LDAP v3 client can be set up to log in to eDirectory three ways: anonymous bind, clear-text passwords, and SSL.

Anonymous Bind. This method allows for authentication without requiring a username or password. In eDirectory, such a user has the equivalent rights of the eDirectory [Public] user. By default, the rights are granted to [Public] are only those attributes that a user requires to log in with a username and password.

You can use the JDBC driver with an anonymous bind; however, your access to eDirectory information is severely restricted.

Clear-Text Password Bind. This method allows the user to authenticate with a username and password, but the LDAP server must be configured to accept clear-text passwords. (By default, the Novell LDAP server accepts only encrypted passwords.) You must also set the useCleartext property of the JDBC driver to true.

SSL Bind. This method requires the LDAP client be set up to use SSL, uses a username and password, and encrypts all data, including the password. In eDirectory, you will have access to the eDirectory information to which the user has been made a trustee. To use SSL binds, the LDAP Extensions and Controls for JNDI must be configured for SSL. For instructions, see *NDK: LDAP Extensions and Controls for JNDI*.

1.3.2 LDAP Naming Rules and Conventions

eDirectory and LDAP have had numerous differences in naming conventions, structural rules for containment, leaf versus containment classes, and supported syntaxes. Each release of eDirectory has extended its support of the LDAP schema.

For example, eDirectory and LDAP naming conventions for attribute and class definitions are quite different:

- eDirectory allows spaces in its schema definition names, usually capitalizes the initial letter of each word in the name, and supports a number of non-alphanumeric characters such as periods and colons.

- LDAP supports alphanumeric characters for schema definition names, does not support spaces in a name, requires an alpha character to begin a name, and supports only one non-alphanumeric character, the dash (-). By convention, LDAP does not capitalize the initial letter of a name, but in multi-word names, it capitalizes the first letter of subsequent words.

Each release of the LDAP server has made these differences less significant. The first release of the LDAP server mapped the LDAP class and attribute names to their corresponding eDirectory class and attribute definitions. In eDirectory 8.7 version, if the schema name is a valid LDAP name, mapping is no longer required. Missing attributes or classes have been added to the eDirectory schema using LDAP naming conventions, and classes have been modified to include the new LDAP attributes. eDirectory 8.7 automatically maps all standard LDAP attributes and classes to eDirectory attributes and classes. For a list, see the "LDAP Classes and Attributes" index in the "Developer Kit".

If an attribute does not appear as a column or a class does not appear as a table because of an incompatible eDirectory name, use ConsoleOne to create a mapping to a valid LDAP name. The Attribute Map option of the LDAP Group entry contains the mappings. With correct mapping to LDAP names, all eDirectory classes are available as tables. After mapping attribute and class names, the LDAP server must be stopped and started for the new configuration to take affect.

Since a network administrator can configure the mapping of eDirectory schema names to LDAP names, you will need to read the schema to determine the LDAP names.

However, even with correctly mapped names, not all eDirectory attributes will be available as columns. The LDAP server does not currently support all eDirectory syntaxes for attributes (see "LDAP Support for eDirectory Syntaxes" on page 15).

1.3.3 LDAP Support for eDirectory Syntaxes

In eDirectory 8.7, the LDAP server supports the following eDirectory syntaxes:

SYN_BOOLEAN
SYN_DIST_NAME
SYN_CE_STRING
SYN_CI_STRING
SYN_PR_STRING
SYN_NU_STRING
SYN_INTEGER
SYN_OCTET_STRING
SYN_TEL_NUMBER
SYN_FAX_NUMBER
SYN_NET_ADDRESS
SYN_PATH
SYN_PO_ADDRESS
SYN_CLASS_NAME
SYN_STREAM
SYN_COUNTER
SYN_TIME
SYN_TYPED_NAME
SYN_INTERVAL

Four of these syntaxes are composite syntaxes: Path, Postal Address, Net Address, and Typed Name. The LDAP server converts these syntaxes to case ignore strings, using the dollar ($) sign to separate fields of similar data and the pound (#) sign to separate fields of dissimilar data.

The LDAP server currently does not support the following eDirectory syntaxes:

SYN_CI_LIST
SYN_OBJECT_ACL
SYN_OCTET_LIST
SYN_EMAIL_ADDRESS
SYN_REPLICA_POINTER
SYN_TIMESTAMP
SYN_BACK_LINK
SYN_HOLD

Attributes which use unsupported syntaxes are not visible from LDAP even when the attribute has a valid LDAP name.

1.3.4 Auxiliary Classes

eDirectory 8.7 and higher support auxiliary classes.

The following sample demonstrates using the LDAP JDBC driver to retrieve information from an auxiliary class:

```
SELECT AuxClassAttribute, BaseClassAttribute
FROM AuxClass a, BaseClass b
WHERE a.NDS_FULLNAME=b.NDS_FULLNAME
```

1.4 LDAP and SQL Integration

The LDAP JDBC Driver maps LDAP information into an SQL relational table. In general, LDAP class definitions become the table name, LDAP entries become the rows in the table, and LDAP attributes become the columns in the table. LDAP also maintains information about entries other than attributes. This information is mapped to the table in special columns. For more information about these topics, see

1.4.1 Supported SQL Syntaxes

The JDBC driver is read only, that is, it supports only select queries. The driver must implement all the JDBC interfaces for supporting the select operations. Following are the SQL syntaxes that are supported by this driver:

Predicates

The following predicates forms are supported:

Table 1-4 *Predicates and its Forms*

Predicates	Forms	Description
Basic		
	x=y	x is equal to y.
	x <> y	x is not equal to y
	x < y	x is less than y
	x > y	x is greater than y
	x >= y	x is greater than or equal to y
	x <= y	x is less than or equal to y
Quantified	ANY	
IN		
NULL		
EXISTS		

Alias

AS

Grouping

ORDER BY

Aggregate

- AVG
- COUNT - COUNT(*) only
- MAX
- MIN
- SUM

Logical Operators

- AND
- OR
- NOT

Join queries

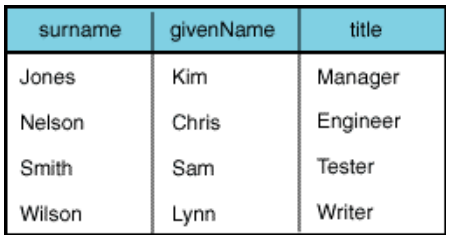
- Inner join

1.4.2 Mapping of LDAP Data to Relational Tables

The object class and attribute definitions in the LDAP schema are taken from their structure as a hierarchical X.500 directory and mapped to a flattened relational database table. Directory concepts such as object class inheritance, naming attributes, and attribute syntax give way to the relational database features of tables and columns. Actual entries, or objects created in the LDAP database, become the rows in the table.

For example, a table for the user object class with four entries and with three attributes (surname, givenName, and title) would look similar to the following diagram.

Figure 1-2 *Mapping of LDAP Data to Retional Tables*



surname	givenName	title
Jones	Kim	Manager
Nelson	Chris	Engineer
Smith	Sam	Tester
Wilson	Lynn	Writer

The following table describes in general the LDAP elements and their SQL counterparts.

Table 1-5 *LDAP Elements and their SQL Counterparts*

SQL	LDAP
Database	The selected tree and base domain name represent the database. These are selected when the connection is established with the LDAP server.
Tables	LDAP classes are represented as database tables. The table name is the same as the class name. Two of the LDAP class names have been given special table names to avoid conflicts with the SQL query keywords. The table for the User class is inetOrgPerson, and the table for group class is group eDirectory.
Columns	LDAP class attributes represent the table columns. Each attribute represents one or more table columns. If the attribute type has multiple data fields, it will result in more than one column in the database table. For example, the homeDirectory attribute will create the following three columns in the table: homeDirectory_NameSpace homeDirectory_Path homeDirectory_VolName For more information, see “Composite Attributes” on page 19.
Records	Each LDAP entry represents one or more rows in the database table. If an entry has a multi-valued attribute and multiple values have been assigned to the attribute, the entry can have multiple rows in the table (see “Multi-Valued Attributes” on page 21).
Special Columns	Special columns are used to present additional information items which are available from the LDAP database but are not LDAP attributes. These items include the entry's context, tree name, and distinguished name. For more information, see “Special Columns in Tables” on page 19.

1.4.3 Special Columns in Tables

LDAP does not use attributes to keep track of the following information about entries:

- The tree in which the entry resides
- The fully distinguished name of the entry
- The fully distinguished name of the context that contains the entry
- The relative name of the entry

Since these are not attributes, the LDAP JDBC Driver provides column names for this information, and these names can be used just like attribute names in SQL statements. All tables can contain the following columns:

- NDS_FullName
- NDS_Context
- NDS_Name

1.4.4 Composite Attributes

Some attributes use a syntax that contains multiple data fields. For example, the homeDirectory attribute uses the Path syntax. This syntax has three fields: name space, volume, and path. The JDBC driver splits such attributes into multiple columns, one for each data field in the syntax. The name of each column consists of the attribute name followed by an underscore and the field name. The homeDirectory attribute is split into the following columns:

- homeDirectory_NameSpace
- homeDirectory_VolName
- homeDirectory_Path

Column Names and Data. The column names may not be completely applicable to attributes which extend the schema. A syntax can be used to store data other than data specified by the label as long as the data fits the data type. For example, the name space field can contain 4 bytes of data, but LDAP does not verify that the data contains a valid name space value. The volume field contains a distinguished name which LDAP verifies. However, LDAP does not verify that it contains a volume name, only that it contains a distinguished name of an entry in the directory. The path field contains a string which LDAP stores but does not verify. Therefore, the path syntax can be used to store the distinguished name of any entry in the directory with a string value (perhaps a description) and 4-byte value (perhaps a integer level).

SQL Statements. When specifying a composite attribute in an SQL select statement, each column that you want must be included in the statement. For example, to select only one of the columns for the homeDirectory attribute, you would use the following:

```
select homeDirectory_NameSpace from inetOrgPerson
```

To select all columns, you would use the following:

```
select homeDirectory from inetOrgPerson
```

Notice, you can specify the attribute name for composite attributes when you want all fields.The multiple fields are put in the same column, and the fields are separated with either a dollar ($) sign for similar data or a pound (#) sign for dissimilar data.

Columns for Composite Attributes. The following table lists the possible columns for each syntax that has multiple fields. The NDS/eDirectory column lists the minimum version required for LDAP support for the syntax. The JDBC data types are VARCHAR unless noted.

Table 1-6 *Syntax with Multiple Fields*

Syntax—LDAP Name (NDS Name)	OID	Columns	NDS/ eDirectory
Tagged Data (Network Address)	2.16.840.1.113719.1.1.5.1.12	_Type	8.3.x
		_Length	
		_Address	
		_Number	
		_String	
Tagged Name and String (Path)	2.16.840.1.113719.1.1.5.1.15	_NameSpace	8.3.x
		_Path	
		_VolName	
Postal Address	1.3.6.1.4.1.1466.115.121.1.41	_Name	8.3.x
		_Street	
		_POBox	
		_City	
		_State	
		_ZipCode	
Typed Name	2.16.840.1.113719.1.1.5.1.25	_ObjectName	8.3.x
		_Level	
		_Interval	
Unknown	2.16.840.1.113719.1.1.5.1.0	_Name	8.5x
		_SyntaxID	
		_Length	
		_Value	
Tagged String (Email Address)	2.16.840.1.113719.1.1.5.1.14	_Type	8.5x
		_Address	
		_Number	
		_String	

Syntax—LDAP Name (NDS Name)	OID	Columns	NDS/ eDirectory
Replica Pointer	2.16.840.1.113719.1.1.5.1.16	_Server	8.5x
		_Type	
		_Number	
		_Count	
		_AddrType	
		_AddrLength	
		_Addr (VARBINARY)	
NDS Timestamp (Time Stamp)	2.16.840.1.113719.1.1.5.1.19	_Seconds	8.5x
		_ReplicaNumber	
		_Event	
Tagged Name (Back Link)	2.16.840.1.113719.1.1.5.1.23	_Name	8.5x
		_Number	

1.4.5 Multi-Valued Attributes

Most attributes are multi-valued, meaning that the attribute can contain more than one value. The order of the values is not guaranteed, and the order can vary from replica to replica. For example, the givenName attribute is multi-valued so that a User entry can have his or her legal given name and multiple nick names, for example, Elizabeth, Liz, and Beth.

When including attributes in a report, you need to know whether the attributes are multi-valued. (Consult the attribute definition in the *Developer Kit* for this information.) If your report selects multi-valued attributes, you need to select one of the following methods of reporting them:

- “Multiple Rows” on page 21
- “Concatenating Rows” on page 22

Multiple Rows

The standard method of reporting multiple values in an SQL report is to put each value in a separate row. If you include two or more multi-valued attributes in a report and select to report each value on a separate row, the size of the table grows by multiples of the number of values. For example, suppose you request a report that includes a user’s telephoneNumber and groupMembership

attributes, and each user has 2 telephone numbers and belongs to 3 groups. The table will have six rows for each user. The figure below illustrates such a table.

Figure 1-3 *Reporting Multiple Values in an SQL Report*

	telephoneNumber	groupMembership
User 1	801-999-9990	Word Processing
	801-999-9990	Email
	801-999-9990	News
	801-999-9991	Word Processing
	801-999-9991	Email
	801-999-9991	News
User 2	801-999-9992	Word Processing
	801-999-9992	Email
	801-999-9992	News
	801-999-9993	Word Processing
	801-999-9993	Email
	801-999-9993	News

If your report is gathering information for 100 users, instead of a 100 row table your report generates a 600 row table. Depending on available workstation resources and the number of multi-valued attributes you have included in the report, you could run out of memory or hard disk space before the report is completed. For a solution to this problem, see “Concatenating Rows” on page 22.

To determine whether an attribute is multi-valued, see the *Developer Kit*.

For information on how an eDirectory syntax is translated to an SQL data type, see “Data Type Mappings” on page 24.

Concatenating Rows

Concatenation allows multiple values of an attribute to be written to a single row in a column, with values separated by a specified delimiter. If your report has selected two or more attributes with multiple values, concatenating values into a single row reduces the size of the table (see “Multiple Rows” for information on how a table can increase in size with multi-valued attributes). The disadvantage of concatenation is that the values can get very long. The advantage is that you can create a report that contains many multi-valued attributes without increasing the number of rows in the report.

Multi-valued attributes have two column names to select from when creating a report. The column name that ends with an _S suffix is the name for concatenating values. For example, the column name for the cn attribute, which is multi-valued, is cn. When cn is selected, the driver produces a row for each value. When the cn_s column is selected, the driver concatenates the values into a single row in the column.

The following sections supply the information you need to concatenate values in your reports:

- “Attributes Eligible for Concatenation” on page 23

Attributes Eligible for Concatenation

Not all multi-valued attributes can be concatenated. The LDAP JDBC Driver supports concatenation of attributes only if they use one of the following syntaxes:

Case Exact String
Case Ignore String
Class Name
Distinguished Name
Facsimile Telephone Number
Network Address (Address column only)
Numeric String
Printable String
Telephone Number
Typed Name (ObjectName column only)

Thus, an attribute must meet two conditions for concatenation: be multi-valued and use one of the supported syntaxes. You can use the *Developer Kit* to verify which attributes support these conditions or use the JDBC catalog function of your SQL tool to query the schema for a list of all columns. The tool returns the column names for all the attributes defined in that directory. Attributes which can be concatenated have a name with an _S suffix.

Separator Characters Used by Concatenation

Value concatenation requires the use of a separator character to indicate the end of one value and the beginning of another. Since attributes can store a variety of data types, a single separator cannot be selected that will be appropriate for all possible values. The LDAP JDBC Driver allows you to set the separator for each report. The default separator is a comma. You can use any single character string as the separator or one of the following escaped values for non-printing characters.

Table 1-7 *Escaped Values for Non-Printing Characters*

Value	Description
'\n'	Line feed
'\r'	Carriage return
'\t'	Tab
'\f '	Form feed
'\b'	Back space
'\\' or '\'	Back slash

Separator Character Functions

The LDAP JDBC Driver supports two scalar functions for managing the separator character used when concatenating values.

Table 1-8 *Scalar Functions*

Function	Description
SetSeparator	Sets the character value of the separator. Accepts a string argument .
GetSeparator	Returns the value of the current separator. Accepts no arguments.

These functions use the standard SQL escape syntax for scalar functions.

Table 1-9 *Standard SQL Escape Syntax*

Function Syntax	Description
{fn SetSeparator(':')}	Sets the separator to a colon
{fn SetSeparator('\t')}	Sets the separator to a tab character
{fn SetSeparator(' ')}	Sets the separator to a line break
{fn GetSeparator()}	Gets the current separator character

Example

```
SELECT cn_S FROM inetorgperson WHERE {fn SetSeparator('+')} = '+'
```

1.4.6 Data Type Mappings

Each column of a table must have an associated data type. The allowed data types are defined by the JDBC protocol. By default, the LDAP protocol returns all attribute values as strings. Even values that contain numeric data are converted to their string representation. Since the JDBC driver is primarily intended for report generation, this default behavior is perfectly acceptable. Except for the exceptions noted below, all columns are of the JDBC type VARCHAR or LONGVARCHAR, which are variable length character strings.

Data Type Mappings for Non-Character Strings. The following table lists the data type mappings for syntaxes that do not map to VARCHAR or LONGVARCHAR.

Table 1-10 *Data Type Mapping for Non-Character Strings*

Syntax Name	Syntax OID	JDBC Data Type
Octet String	1.3.6.1.4.1.1466.115.121.1.40	LONGVARBINARY
Octet List	2.16.840.1.113719.1.1.5.1.13	LONGVARBINARY
Integer	1.3.6.1.4.1.1466.115.121.1.27	INTEGER
Counter	2.16.840.1.113719.1.1.5.1.22	INTEGER

1.4.7 Effective Rights Table

Most of the tables available to the JDBC driver are created from the object class definitions in the schema. The one exception is the effective rights table. This table is not part of the schema and is created by the JDBC driver. The four object columns contain information about the entry to which the trustee has rights. The four trustee columns contain information about the entry who has been granted rights to the entry specified by the object columns.

The table uses the baseDN property to determine where to start reading ACL attributes. The table generates a large report unless restricted by a WHERE clause. It includes the following:

- Rows for all entries below the context specified by the baseDN property
- A row for every entry in the eDirectory tree as a trustee for each ObjectName included in the report

If you have 1,000 entries in the eDirectory tree, each entry will have rows for the 1,000 potential trustees. In addition, if you specify all attributes, the report will have the 1,000 rows for each attribute.

To restrict the size of the report to the information you really want, use a WHERE clause with values for the following columns:

- ObjectName
- ObjectContext
- ObjectClass
- TrusteeName
- TrusteeContext
- TrusteeClass
- Attribute

Do not use ObjectFullname or TrusteeFullname to restrict the size. These two columns, when used in a WHERE clause, noticeably degrade performance.

For a sample SQL command, see "Restricted Effective Rights Query" on page 29.

The following table does not contain columns for the Attribute Inheritance Control or the Entry Inheritance Control rights because these rights are used to calculate effective rights, and once used to calculate rights, the other rights specify what has been granted.

Table 1-11 *Effective Rights Table*

Column Name	JDBC Data Type	Description
ObjectName	VARCHAR	Contains the relative distinguished name of the entry for which the trustee has rights.
ObjectFullname	VARCHAR	Contains the distinguished name of the entry for which the trustee has rights.
ObjectContext	VARCHAR	Contains the distinguished name of the container in which the entry is located.
ObjectClass	VARCHAR	Contains the entry's base class.

Column Name	JDBC Data Type	Description
TrusteeName	VARCHAR	Contains the relative distinguished name of the entry that is the trustee.
TrusteeFullname	VARCHAR	Contains the distinguished name of the entry that is the trustee.
TrusteeContext	VARCHAR	Contains the distinguished name of the container in which the trustee is located.
TrusteeClass	VARCHAR	Contains the trustee's base class.
Attribute	VARCHAR	Contains one of the following: • The name of the attribute to which the trustee has rights • [entry rights] which indicates the trustee assignment is to the entry itself • [all attributes] which indicates the trustee assignment is applied to all attributes
Privileges	INTEGER	Contains an integer whose value represents the rights that have been granted.
Add Self	BIT	Indicates that the trustee has the rights to add or remove itself as an attribute value. This right is only used for attributes whose values are distinguished names such as group members and mailing lists.
Browse	BIT	Indicates that the trustee has the rights to see the entry in the eDirectory tree.
Compare	BIT	Indicates that the trustee has the rights to compare values of an attribute.
Create	BIT	Indicates that the trustee has the rights to create new entries in the eDirectory tree. This right is available only for container objects.
Delete	BIT	Indicates that the trustee has the rights to delete entries from the eDirectory tree.
Read	BIT	Indicates that the trustee has the rights to read and compare attribute values. The Read right implies the Compare right.
Rename	BIT	Indicates that the trustee has the rights to change the name of an entry.
Supervisor	BIT	Indicates that the trustee has the rights to the entry and all of its attributes.
Write	BIT	Indicates that the trustee has the rights to add, change, or remove any values of the attribute. This right implies the Add Self right.

1.5 Samples

The following samples illustrate how to formulate basic SQL statements to generate reports.

1.5.1 Last Login Time Query

This example illustrates how to filter results using a comparison. The query below returns all users who have not logged in since July 8, 1999 at 6:45 pm. Notice how the literal timestamp value is expressed. It is contained within braces which serve as escape sequence delimiters. The ts indicates the value is a timestamp. The actual string representing the timestamp is contained within the single quotes.

SQL Command

```
SELECT CN, lastLoginTime FROM inetOrgPerson WHERE
lastLoginTime < {ts '1999-07-08 18:45:00.00'}
```

Report

Table 1-12 *Last Login Time Query Report*

CN	LastLoginTime
fsmith	1999-06-25 09:30:24.0
bbrown	1999-07-08 18:42:36.0
jdoe	1999-07-04 14:10:54.0

1.5.2 Sorting Query

This example illustrates the use of the ORDER BY clause to sort a table on multiple columns.

SQL Command

```
SELECT NDS_Context, SN FROM inetOrgPerson ORDER BY
NDS_Context, SN
```

Report

Table 1-13 *Sorting Query Report*

NDS_Context	SN
o=Fred's Widgets	Brady
o=Fred's Widgets	Johnson
o=Fred's Widgets	Smart
o=Fred's Widgets	Swift

NDS_Context	SN
ou=Marketing,o=Fred’s Widgets	Bergman
ou=Marketing,o=Fred’s Widgets	Fairbanks
ou=Marketing,o=Fred’s Widgets	Gilbert
ou=Product Develpment,o=Fred’s Widgets	Sanders
ou=Product Develpment,o=Fred’s Widgets	Smith
ou=Product Develpment,o=Fred’s Widgets	Weirsdorf
ou=retail,o=Fred’s Widgets	Newton
ou=retail,o=Fred’s Widgets	Stapley
ou=Sales,o=Fred’s Widgets	Anderson
ou=Sales,o=Fred’s Widgets	Desmond
ou=Sales,o=Fred’s Widgets	Knight

1.5.3 ACL Attribute Query

This sample illustrates the break up of the ACL attribute into multiple columns. The ACL_Trustee, ACL_Attribute, ACL_Read, ACL_Write columns all come from the value of the ACL attribute. For a complete list of all ACL columns see the Data Type Mappings table. A value of true (one) in a privilege column such as ACL_Read or ACL_Write indicates the privilege is granted to the indicated trustee. A value of false (zero) indicates the trustee does not have that privilege. The OU (Organization Unit) attribute is the naming attribute for the Organization class.

An ACL attribute query requires the 8.5x version of eDirectory. To obtain a beta copy, see the Novell's public beta site:

```
http://qasupport.provo.novell.com/beta/public/
```

SQL Command

```
SELECT OU, ACL_Trustee, ACL_Attribute, ACL_Read, ACL_Write
FROM OrganizationalUnit
```

Report

Table 1-14 *ACL Attribute Query Report*

OU	ACL_Trustee	ACL_Attribute	ACL_Read	ACL_Write
Sales	ou=Sales,o=Fred’s Widgets	loginScript	true	false
Sales	ou=Sales,o=Fred’s Widgets	printJobConfiguration	true	false
Marketing	ou=Marketing,o=Fred’s Widgets	loginScript	true	false
Marketing	ou=Marketing,o=Fred’s Widgets	printJobConfiguration	true	false
Operations	ou=Operations,o=Fred’s Widgets	loginScript	true	false

OU	ACL_Trustee	ACL_Attribute	ACL_Read	ACL_Write
Operations	ou=Operations,o=Fred's Widgets	printJobConfiguration	true	false

1.5.4 Restricted Effective Rights Query

The query below illustrates how to perform several restrictions when querying the Effective Rights table. First, the like operator is used to restrict the trustees to those in the FewUsers Organization. The Object is restricted to dward0.FewUsers by a simple equality comparison. Finally, the attributes are limited to [Entry Rights]. This query lists the rights of the trustee to operate on the dward0.FewUsers entry as a whole. The query is further restricted to retrieve the value of only the Browse and Delete privileges. The result table illustrates the typical situation where all trustees have browse privileges but none have delete privileges.

The Effective Rights table requires the 8.7 version of eDirectory. To obtain a Beta copy, see the Novell web site (http://www.novell.com/).

SQL Command

```
SELECT ObjectName, TrusteeName, Attribute, Browse, "Delete"
FROM EffectiveRights
WHERE TrusteeContext = 'ou=Sales,o=Fred''s Widgets'
AND objectName = 'cn=bjones'
AND Attribute = '[Entry Rights]'
```

The query produces a result table similar to the following. The zeros in the rights columns mean FALSE (the trustee does not have this right), and the ones, TRUE (the trustee does have this right).

Report

Table 1-15 *Restricted Effective Rights Query Report*

ObjectName	TrusteeName	Attribute	Browse	Delete
cn=bjones	cn=KTurner	[Entry Rights]	1	0
cn=bjones	cn=LKnight	[Entry Rights]	1	0
cn=bjones	cn=MDesmond	[Entry Rights]	1	0
cn=bjones	cn=SPurdy	[Entry Rights]	1	0
cn=bjones	cn=TAnderson	[Entry Rights]	1	0

1.5.5 Join Query

The following query joins a base class table with an auxiliary class table. It reads the O column which is part of the Organization table and the Synchronized Up To_Time column which is part of the Partition table.

SQL Statement

```
SELECT O, "Synchroinzed Up To_Time"
```

```
FROM Organization, Partition
WHERE Organization.NDS_FullName=Partition.NDS_FullName
```

Result Table

Table 1-16 *Result of the Join Query*

O	Synchronized Up To_Time
Operations	2000-02-18 13:06:41.000
Enginerring	2000-02-19 09:03:23.000

Revision History

A

The following table lists all changes made to the Novell LDAP JDBC Driver documentation:

June 2006	• Added new rows derefAlias and saslExternal toTable 1-3 on page 12 • Extended multicharacter support for SetSeparator • Added the escape syntax {fn SetSeparator(' ')} to Table 1-9, “Standard SQL Escape Syntax,” on page 24
March 2006	Added the following: • Figure 1-1, “LDAP JDBC Driver Architecture,” on page 9 • Section 1.4.1, “Supported SQL Syntaxes,” on page 16 Fixed formatting issues.
March 2005	Changed development name Tao to eDirectory 8.5.x.
September 2001	Changed information stating that UserNDS is the main user table, to state that inetOrgPerson is the main user table.
June 2001	Added information on using auxiliary classes with the JDBC driver, and updated references to LDAP JNDI providers.
May 2000	Added sample SQL commands and reports. Added the column names for the structured attributes supported in the 8.5x version of NDS.