Novell Developer Kit

www.novell.com

February 28, 2007

NOVELL ODBC DRIVER FOR EDIRECTORY™ READ-WRITE

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.

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 developer products, and to get updates, see developer.novell.com/ndk. To access online documentation for Novell products, 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

ODBC Driver for eDirectory

The Novell® ODBC Driver for Novell eDirectory™ Read-Write provides an Open Database Connectivity (ODBC) driver specifically designed to query and retrieve eDirectory data. This eDirectory ODBC driver serves as an independent interface for extracting and reporting specified directory information for use in the applications that you use every day. It allows you to populate reports, import data into your custom programs, or view data within a spreadsheet. In addition, the Novell ODBC driver for eDirectory retains the standard ODBC qualities of simplicity, high performance, and independent interface programming to make the job of reporting and retrieving information quick and easy.

With the implementation of the write feature, you can now do simple update operations like insert, modify, and delete on eDirectory objects. Refer to Section 2.7, “Inserting, Updating or Deleting Entries from the eDirectory Data Source,” on page 15 for more details. The driver also comes with more advanced features like update enabled cursor and positioned update. Refer to Chapter 5, “ODBC Driver Advanced Write Capability,” on page 45 for more details.

The architecture behind the Novell ODBC driver for eDirectory consists of the application, the ODBC.DLL Driver Manager, the Novell ODBCNDS.DLL driver, the network, and eDirectory itself. The driver employs ODBCNDS.DLL to abstract 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 Structured Query Language (SQL) statements embedded into the application.

This guide consists of the following sections:

Audience

This guide is intended for developers who are not familiar with all of the components of the LDAP ODBC 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 Comment feature at the bottom of each page of the online documentation.

Additional Information

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

Documentation Updates

For the most recent version of this guide, see the ODBC Driver for eDirectory NDK page (http://developer.novell.com/ndk/odbc.htm).

Docuentation 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.

Requirements

1

The Novell® ODBC driver for Novell eDirectory™ 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:

- Section 1.1, “Resource Restrictions,” on page 9
- Section 1.2, “Hardware and Software Requirements,” on page 9

1.1 Resource Restrictions

Since the ODBC Driver for eDirectory 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.

eDirectory can contain a tremendous number of objects. For example, a container object can have

- Over 10,000 objects in NetWare 5
- Millions of objects in NDS 8

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

Also, each object in eDirectory contains multiple attributes. For example, User objects in NDS 8 had over 90 attributes. If you add other products which add attributes to User objects, such as ZENworks, the number is even greater. The Novell ODBC Driver for NDS reads the schema to determine the actual number of attributes an object 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 objects is not recommended. To keep a report to a reasonable size, you need to select specific attributes for the report. To determine which objects have tables and what attributes an object has, see Possible NDS Tables and Columns. 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. NDS 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 how the NDS database has been replicated and whether replicas of all partitions are stored locally or over WAN links.

1.2 Hardware and Software Requirements

To install the Novell ODBC Driver for eDirectory, you must have workstation with the following configuration:

- Windows 95*, Windows 98*, or Windows NT 3.51* or later
- A network card
- Novell Client32™ or NetWare Client™ for Windows software installed

In addition, you must have Novell Directory Services (eDirectory) installed and running on the network.

To use the driver, you need the following amounts of disk space. The target drive can be on the workstation or a server.

Table 1-1 *Hardware and Software Requirements*

Location	Size	Stores
Workstation	2 MB	`ODBCNDS.DLL`, `Help`, `Redist DLL`
Target Drive	200 KB	Example and information files

Getting Started

2

The following sections explain what you need to know and do to use the Novell® ODBC driver for Novell eDirectory™ for the first time:

2.1 Installing the ODBC Driver

To install the Novell ODBC Driver for eDirectory, launch the install program. The standard installation dialog boxes opens, prompting you to select a directory to load the programs. The default installation will place the files in the `C:\Novell\ODBCNDS` directory. The `license` and `Readme` files are placed here, along with a directory containing sample applications.

When you install the Novell ODBC Driver for eDirectory, the following components are loaded into their respective directories:

- `ODBCNDS.DLL` file
- Configured Default eDirectory Database data source
- Sample Crystal Report and Visual Basic reports
- Help documentation

After installation, you can open the Data Source Administrator, documentation, and Visual Basic and Crystal Report samples from the *Windows Start Menu*. During installation, you can click the *Configure Data Sources* button or select the 32bit ODBC file in the *Control Panel* to open the *Data Source Administrator*.

The last dialog box gives you options to Launch Release Notes, which opens the `Readme file`, or to `Configure Data Sources`, which opens the *Data Source Administrator* to specify the eDirectory tree and context that you want to configure as a data source for the ODBC driver.

To select the eDirectory tree and context that you want to abstract into an eDirectory data source, see Section 2.5, “Selecting the eDirectory Data Source,” on page 12.

To uninstall the driver, see Section 6.2, “Uninstalling the Novell ODBC Driver for eDirectory,” on page 50.

2.2 Installing an SQL Report Generating Tool

Many applications are ODBC-aware and allow you to connect to an NDS ODBC data source to report information, populate spreadsheets, and load data to external databases. You can also build custom applications that embed SQL statements in the program code. To name a few, the Novell ODBC Driver for NDS supports the following tools:

- Seagate Software Crystal Reports
- Microsoft Visual Basic
- Microsoft Access
- JDBC to ODBC Bridge

The Novell ODBC Driver for NDS allows you to read information and make it work for you in a variety of ways. With the popularity of the ODBC technology, most data applications provide an ODBC interface for connecting to an ODBC data source. And because NDS is a scalable, extensible network directory that contains valuable information and can be accessed easily through your network, the reports and applications that can be designed are limitless. Here are a few ideas for using the Novell ODBC Driver for NDS:

- Use a spreadsheet application to show the time used on a network account for each user.
- Use a report application mailing feature to run a mailing list from the users in the directory.
- Run a report that identifies volume information, including volume name, the host server, and status of the volume.
- Access Workstation information including entry name, operating system, CPU, and memory.

2.3 Logging In to the eDirectory Tree

Before using the Novell ODBC Driver for Novell eDirectory, log in to the eDirectory tree or trees that you will be using to generate reports. The driver does not prompt you to log in. It uses the credentials you establish before it is launched. If you are not logged in and authenticated, you have no rights to any eDirectory data and therefore your reports will be empty.

2.4 Check eDirectory Rights

The Novell ODBC Driver for NDS is constrained by NDS security. If the account you use to log in to the NDS tree does not have rights to the requested information, the information is not returned. Even if you have Supervisor rights to the root of the NDS tree, you may not have rights to all information in the NDS database because NDS allows rights to be revoked in branches of the tree so that these branches can be managed independently. Before using the driver to generate a report, make sure you have the NDS rights to the data you are requesting. If you do not have the rights to read an attribute, the driver returns the SQL NULL value in that attribute's column in the SQL table.

2.5 Selecting the eDirectory Data Source

To ODBC, the data source is the database that you are accessing. Since eDirectory is a hierarchical, distributed database, the Novell ODBC Driver for NDS allows you to select the NDS tree or trees you want to access, and within the tree, the NDS context or contexts you want to use as data sources. Each tree and each point within a tree are configured as a different data source.

This allows you great flexibility. You will want to configure at least one NDS data source for each NDS tree or specific context that you want to query. The data source determines from which NDS containers the driver can gather information. If the data source uses the root container as its context, information can be gathered from the entire NDS tree. If the data source uses a child container as its context, the driver can gather information only from that branch of the NDS tree.

You can have different data sources for different trees and contexts in the database. For example, you can specify a data source for your whole directory tree, another for a Organizational Unit object in the tree, and another for an Organization object in the tree. All three would use the Novell ODBC Driver for NDS, but each would have a unique data source name (DSN) and abstract a unique scope within the directory. If your organization has multiple NDS trees, you can specify multiple data sources for each NDS tree.

The ODBC Data Source Administrator requires four pieces of information to create a data source: name, description, NDS tree, and NDS context. The Novell Data Source Setup dialog box allows you to browse and select an NDS tree, and from the NDS tree, to browse and select an NDS context.

The Windows ODBC Data Source Administrator dialog box allows you to view a list of the configured data sources and add, delete, and modify the data source names. It allows you to select the Novell ODBC Driver for NDS, which opens the Novell Data Source Setup dialog box. Once this Novell dialog box is open, you can enter a DSN and configure the NDS context of the data source. When you install the Novell ODBC Driver for NDS, the installation program automatically installs a default data source to the NDS context that you were using during installation.

Once you have created a data source with a unique name, applications can view this as another repository of relational database information. The directory can now be viewed and used in standard reporting applications. To retrieve and order this information, you can embed SQL requests in your application, or use applications that automatically generate SQL statements for reporting directory information.

To select the default data source, complete the following steps:

1. Start your SQL application.
2. Connect to the Novell ODBC Driver for NDS.
3. From the data source window, select the "Default NDS Data Source".

To add, modify, or delete a data source, see one of the following:

- Section 2.5.1, “Adding an eDirectory Data Source,” on page 13
- Section 2.5.2, “Deleting an eDirectory Data Source,” on page 14
- Section 2.5.3, “Modifying an eDirectory Data Source,” on page 14

2.5.1 Adding an eDirectory Data Source

For a given driver you can create multiple data sources. For example, if you have several network trees, you can create one data source for each tree, or you can select multiple contexts within a tree and create a data source for each context. All data sources uses the same ODBC driver but will have a unique DSN (data source name).

Follow these steps to add a new data source, select a driver, and name the data source:

1 Select *Start* > *Settings* > *Control Panel* > *32BIT ODBC* (or just ODBC on Windows NT).

The ODBC Data Source Administrator dialog box in Windows opens.

2. Select the *USER DSN* tab. Click the *Add* button.

   The *Create New Data Source* dialog box opens, displaying a list of drivers.

3. Double-click the *Novell ODBC Driver* for NDS driver in the list.

   The *NDS ODBC Driver* dialog box will open.

4. Enter a Name and Description for the new data source.

5. To use your current NDS context for the data source, select the *Use Default NDS Context* button and skip to Step 7. To select the NDS context for the data source, deselect the *Use Default NDS Context* button.

   The NDS tree view window will activate.

6. Select the NDS tree and context from the tree view in the list box.

7. Select *OK*.

   The ODBC Data Source Administrator lists the new data source.

NOTE: These steps can be accomplished in a different order depending on the version of ODBC Administrator installed on your machine. This example is based on the 32-bit ODBC administrator supplied with ODBC version 3.0. In earlier versions, click the Add button to select a data source list.

2.5.2 Deleting an eDirectory Data Source

Follow these steps to remove a data source from the Data Source Administrator dialog box:

1. Select the data source to be deleted from the ODBC Data Source Administrator dialog.
2. Click the Remove button.

You will be prompted to verify the deletion before the DSN is removed from the list.

2.5.3 Modifying an eDirectory Data Source

You can change your mind and alter the configuration of a data source in the administrator dialog box. This allows you to reuse the DSN (Data Source Name) and redefine the tree or organization unit.

Follow these steps to modify a new data source for an existing Data Source Name.

1. Double-click an existing data source name in the Data Source Administrator dialog box.

   The NDS ODBC Driver dialog will open with Name, Description, and the current data source displayed.

2. Select a new tree and context data source from the Database Details list box.
3. Click OK.

The DSN remains in the list, but will now have a new configuration of entry class definitions.

2.6 Selecting the Data and Generating the Report

The following SQL statement uses two User class attributes (Given Name and Surname) and one special column name NDS_Tree. It gathers information about users so it uses the UserNDS table. These attributes are public read attributes, so any logged in NDS user can execute this statement and generate a report.

SQL Statement

Enter the following statement into the query window, execute the statement, and retrieve the data.

```
SELECT "Given Name", "Surname", "NDS_Tree"
FROM UserNDS
```

Result Table

The SQL statement generates a report similar to the following:

Table 2-1 *SQL Statement Result Table*

Given Name	Surname	NDS_Tree
Kris	Hammons	NOVELL_INC
Lynn	Thayn	NOVELL_INC
Jan	Aitchison	NOVELL_INC
Alex	Buckley	NOVELL_INC
Tom	Turner	NOVELL_INC
Cory	Petersen	NOVELL_INC

2.7 Inserting, Updating or Deleting Entries from the eDirectory Data Source

You can update the NDS Data Source using simple insert, update, and delete SQL queries.

Insert

The following SQL statement enables you to add an user object to the NDS Data Source:

```
INSERT into UserNDS (Surname, Nds_Name, UID) values ('Duke', 'Luke',
8)
```

Update

The following SQL statement enables you to modify an user object in the NDS Data Source:

```
UPDATE UserNDS SET Surname = 'Kuke', UID = UID + 20 WHERE Nds_Name LIKE
'%uke'
```

The following SQL escape syntax enables you to modify the timestamp:

```
UPDATE UserNDS SET "password expiration time" = {ts '2004-02-02
08:00:00.00'} WHERE Nds_Name = 'user'"
```

Delete

The following SQL statement enables you to delete an user object from the NDS Data Source:

```
DELETE FROM UserNDS WHERE Nds_Name = 'Luke' OR UID > 7
```

NOTE: While inserting an object, you need to specify the Nds_Name as an attribute (column in SQL).

The driver does not support positioned update, delete, or insertion.

If the attribute value is in decimals it will be treated as an Integer while adding to Novell eDirectory.

You cannot create or drop a table (create or delete an object class, which requires schema extension) using the ODBC driver.

For a single entry, SQL semantics does not permit using multiple values for a single column. Hence, this driver does not allow you to Insert/Update an entry with multiple values for one particular attribute. However, the driver allows you to put in just one value for a multi-valued attribute.

Novell eDirectory and SQL Integration

3

The Novell® ODBC driver for Novell eDirectory™ maps eDirectory information into an SQL relational table. In general, eDirectory objects become the rows in the table, and eDirectory attributes become the columns in the table. eDirectory also maintains information about objects other than attributes. This information is mapped to the table in special columns. For more information about these topics, see:

3.1 Mapping of eDirectory Data to Relational Tables

The object class and attribute definitions in the eDirectory schema are taken from their structure as a hierarchical X.500 directory and mapped to a flattened relational database table. eDirectory 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 eDirectory database, become the rows in the table. The eDirectory data in a relational table has the following format:

- eDirectory object classes correspond to the tables
- eDirectory class attributes correspond to columns of the table
- Entries correspond to rows of the table

For example, a table for the User object with four entries and with three attributes (Surname, Given Name, and Title) would look similar to the following diagram.

Figure 3-1 *Mapping of eDirectory Data to Relational Tables.*

Surname	Given Name	Title
Jones	Kim	Manager
Nelson	Chris	Engineer
Smith	Sam	Tester
Wilson	Lynn	Writer

The eDirectory class name corresponds with the table name, for example, the table for the Volume object is Volume. However, the eDirectory classes "User" and "Group" are exceptions to this convention. They are represented as "UserNDS" and "GroupNDS" because these class names are also SQL keywords. The following table describes in general the eDirectory elements and their SQL counterparts.

Table 3-1 *eDirectory elements and their SQL counterparts.*

SQL	eDirectory
Database	The selected tree and context represents the database. These are selected when the Data Source is configured. Refer to Section 2.5, “Selecting the eDirectory Data Source,” on page 12 for more information.
Tables	eDirectory classes are represented as database tables. The table name is the same as the class name. If the class name contains spaces, the table name must be enclosed in double quotes in an SQL query. Two of the eDirectory class names have been given special table names to avoid conflicts with the SQL query keywords. The table for the User class is UserNDS, and the table for Group class is GroupNDS. eDirectory has more than one type of class: effective and non-effective classes; super and auxiliary classes. Class type affects the entries that belong to the table. (For more information, see Section 3.7, “eDirectory Class Types,” on page 27.)
Columns	eDirectory class attributes represent the table columns. Each attribute represents one or more table columns. If the attribute type is a structure, it will result in more than one column in the database table. For example, the "Home directory" attribute will create the following three possible columns in the table: Home Directory_NameSpace Home Directory_Path Home Directory_VolName For more information, see Section 3.5, “Data Type Mappings,” on page 23.
Records	Each eDirectory object represents one or more rows in the database table. If an object has a multi-valued attribute and multiple values have been assigned to the attribute, the object can have multiple rows in the table. For more information, see Section 3.4, “Multi-Valued Attributes,” on page 20.
Special Columns	Special columns are used to present additional information items which are available from the eDirectory database but are not attributes. These items include the object’s eDirectory context, tree name, and full name. For more information, see Section 3.2, “Special Columns in Tables,” on page 18.

3.2 Special Columns in Tables

eDirectory does not use attributes to keep track of the following information about eDirectory objects:

- The tree in which the object resides.

- The fully distinguished name of the object.
- The fully distinguished name of the context that contains the object.
- The name value in the multi-valued naming attribute that is the name of the object.

Since these are not attributes, the ODBC Driver for eDirectory 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_Tree
- NDS_FullName
- NDS_Context
- NDS_Name

3.3 Composite Attributes

Some attributes use a syntax that contains multiple data fields. For example, the Home Directory attribute uses the Path syntax. This syntax has three fields: name space, volume, and path. The ODBC Driver for eDirectory 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 Home Directory attribute is split into the following columns:

- Home Directory_NameSpace
- Home Directory_VolName
- Home Directory_Path

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 eDirectory does not verify that the data contains a valid name space value. The volume field contains a distinguished name which eDirectory verifies. However, eDirectory does not verify that it contains a volume name, only that it contains a distinguished name of an object in the eDirectory tree. The path field contains a string which eDirectory stores but does not verify. Therefore, the path syntax can be used to store the distinguished name of any object in the eDirectory tree with a string value (perhaps a description) and 4-byte value (perhaps a integer level).

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 Home Directory attribute, you would use the following:

```
select "Home Directory_NameSpace" from UserNDS
```

To select all columns, you would use the following:

```
select "Home Directory_NameSpace", "Home Directory_VolName", "Home
Directory_Path"      from UserNDS
```

Notice, you cannot just specify the attribute name for composite attributes. You must specify each column.

For more information, see Section 3.5, “Data Type Mappings,” on page 23.

3.4 Multi-Valued Attributes

Most eDirectory 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. Since the default is to allow multiple values, an attribute must be defined with the DS_SINGLE_VALUED_ATTR flag to have eDirectory enforce the rule of only one value. For example, the Given Name attribute is multi-valued so that a User object 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 eDirectory Schema Reference for this information.) If your report selects multi-valued attributes, you need to select the method of reporting them:

- Section 3.4.1, “Multiple Rows,” on page 20
- Section 3.4.2, “Concatenating Rows,” on page 21

3.4.1 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 Telephone Number and Group Membership attributes, and each user has 2 telephone numbers and belongs to 3 groups. The table has six rows for each user. The figure below illustrates such a table.

Figure 3-2 *Multiple Rows*

	Telephone Number	Group Membership
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 Section 3.4.2, “Concatenating Rows,” on page 21.

To determine whether an attribute is multi-valued, see the eDirectory Schema Reference.

For information on how an eDirectory syntax is translated to an SQL data type, see Section 3.5, “Data Type Mappings,” on page 23.

3.4.2 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 multi-values, concatenating values into a single row reduces the size of the table (see Section 3.4.1, “Multiple Rows,” on page 20 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

Not all multi-valued attributes can be concatenated. The Novell ODBC Driver for eDirectory 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 (ObjName 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 eDirectory Schema Reference to verify which attributes support these conditions or use the ODBC catalog function of your SQL tool to query the eDirectory tree for a list of all columns. The tool returns the column names for all the eDirectory attributes defined in that eDirectory tree. 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 eDirectory attributes can store a variety of data types, a single separator cannot be selected that will be appropriate for all possible values. The Novell ODBC Driver for eDirectory 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 3-2 *Separator Characters Used by Concatenation*

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

Separator Character Functions

The Novell ODBC Driver for eDirectory supports two scalar functions for managing the separator character used when concatenating values.

Table 3-3 *Separator Character Functions*

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

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

Table 3-4 *SQL Escape Syntax for Scalar Functions*

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

For a list of characters that SetSeparator can use, see "Separator Characters Used by Concatenation" on page 22.

Concatenation Query

The following example illustrates how to use the SetSeparator function to set the separator to the plus (+) character and then concatenate the values for the Member attribute into one row for GroupNDS objects.

SQL Statement:

```
SELECT Member_S

FROM GroupNDS WHERE { fn SetSeparator('+') } <>','
```

Enter this statement into the query window, execute the statement, and retrieve the data.

Result Table:

The SQL statement will generate a report similar to the following:

Table 3-5 *Concatenation Query Result*

NDS_Name	Member_S
Top Dogs	FJohnson.Fred's Widgets+GSwift.Fred's Widgets+ SBrady.Fred's Widgets

3.5 Data Type Mappings

This section shows the mapping from eDirectory data types to SQL data types:

Table 3-6 *Details of Data Type Mappings*

eDirectory Attribute Syntax	Column Name	SQL Data Type
SYN_BACK_LINK	<attribute>_RemoteId	SQL_INTEGER
	<attribute>_ObjName	SQL_VARCHAR
SYN_BOOLEAN	<attribute>	SQL_BIT
SYN_CE_STRING	<attribute>	SQL_VARCHAR
SYN_CI_LIST*	<attribute>	SQL_VARCHAR
SYN_CI_STRING	<attribute>	SQL_VARCHAR
SYN_CLASS_NAME	<attribute>	SQL_VARCHAR
SYN_COUNTER	<attribute>	SQL_INTEGER
SYN_DIST_NAME	<attribute>	SQL_VARCHAR
SYN_EMAIL_ADDRESS	<attribute>_Type	SQL_INTEGER
	<attribute>_Addr	SQL_VARCHAR
SYN_FAX_NUMBER	<attribute>	SQL_VARCHAR

eDirectory Attribute Syntax	Column Name	SQL Data Type
SYN_HOLD	<attribute>_ObjName	SQL_VARCHAR
	<attribute>_Amount	SQL_INTEGER
SYN_INTEGER**	<attribute>	SQL_INTEGER
SYN_INTERVAL	<attribute>	SQL_INTEGER
SYN_NET_ADDRESS	<attribute>_AddrType	SQL_INTEGER
	<attribute>_AddrLength	SQL_INTEGER
	<attribute>_Addr	SQL_VARCHAR
SYN_NU_STRING	<attribute>	SQL_VARCHAR
SYN_OBJECT_ACL	<attribute>_ProtAttr	SQL_VARCHAR
	<attribute>_Privileges	SQL_INTEGER
	<attribute>_Supervisor	SQL_BIT
	<attribute>_Browse	SQL_BIT
	<attribute>_Create	SQL_BIT
	<attribute>_Delete	SQL_BIT
	<attribute>_Rename	SQL_BIT
	<attribute>_Compare	SQL_BIT
	<attribute>_Read	SQL_BIT
	<attribute>_Write	SQL_BIT
	<attribute>_Add Self	SQL_BIT
	<attribute>_Inheritable	SQL_BIT
SYN_OCTET_LIST	Not Supported	
SYN_OCTET_STRING**	<attribute>_Length	SQL_INTEGER
	<attribute>_Data***	SQL_VARCHAR
SYN_PATH**	<attribute>_NameSpace	SQL_INTEGER
	<attribute>_Path	SQL_VARCHAR
	<attribute>_VolName	SQL_VARCHAR
SYN_PO_ADDRESS	<attribute>_Name	SQL_VARCHAR
	<attribute>_Street	SQL_VARCHAR
	<attribute>_POBox	SQL_VARCHAR
	<attribute>_City	SQL_VARCHAR
	<attribute>_State	SQL_VARCHAR
	<attribute>_Zip	SQL_VARCHAR

eDirectory Attribute Syntax	Column Name	SQL Data Type
SYN_PR_STRING	<attribute>	SQL_VARCHAR
SYN_REPLICA_POINTER	<attribute>_Server	SQL_VARCHAR
	<attribute>_Type	SQL_INTEGER
	<attribute>_Number	SQL_INTEGER
	<attribute>_Count	SQL_INTEGER
	<attribute>_AddrType	SQL_INTEGER
	<attribute>_AddrLength	SQL_INTEGER
	<attribute>_Addr	SQL_VARCHAR
SYN_STREAM	<attribute>	SQL_VARCHAR
SYN_TEL_NUMBER	<attribute>	SQL_VARCHAR
SYN_TIME	<attribute>	SQL_TIMESTAMP
SYN_TIMESTAMP	<attribute>_Time	SQL_TIMESTAMP
	<attribute>_EventId	SQL_INTEGER
SYN_TYPED_NAME	<attribute>_ObjName	SQL_VARCHAR
	<attribute>_Level	SQL_INTEGER
	<attribute>_Interval	SQL_INTEGER
SYN_UNKNOWN	<attribute>_Name	SQL_VARCHAR
	<attribute>_SyntaxId	SQL_INTEGER
	<attribute>_Length	SQL_INTEGER
	<attribute>_Value	SQL_VARCHAR

NOTE: * The strings in the Case Ignore List are placed in a single string with a comma separating the individual strings.

**Certain attributes with this syntax are specially interpreted by the ODBC driver.

***The data is presented in hexadecimal format.

3.6 Effective Rights Table

The Effective Rights table is a special table. It does not represent an eDirectory object class. It represents the rights that the objects in the eDirectory tree have to the objects in the branch of the eDirectory tree specified by the data source. The rights are calculated by treating each object in the tree as a trustee of each object in the container and its subcontainers and as a trustee of each of those object's attributes. The following equation gives an estimate of the total number of rows in this table:

```
(# of objects in the data source context) x

(# of objects in the tree) x
```

```
(average number of mandatory & optional attributes per object class)
```

As the equation shows, the number of rows in the effective rights table is extremely large for even a small table. A table for a tree with 20 objects in the data source context, 100 objects in the tree and an average of 50 attributes per object class would have approximately 100,000 rows. You will almost always need to use a where clause to restrict the number of rows returned by your queries of the Effective Rights table.

The table below lists and defines each of the columns in the Effective Rights table. Notice that this table does not include the NDS_Tree, NDS_Context, or NDS_FullName.

Table 3-7 *Details of the Effective Rights Table*

Column Name	SQL Data Type	Description
Object Name	SQL_VARCHAR	The full name of the object for which the trustee has rights. Values for object name include all of the objects in the context specified by the data source.
Object Class	SQL_VARCHAR	The class of the object for which the trustee has rights.
Trustee Name	SQL_VARCHAR	The full name of the object that has rights to the object specified by the Object Name column. Values for trustee name include the names of all objects in the directory.
Trustee Class	SQL_VARCHAR	The class of the trustee object.
Attribute	SQL_VARCHAR	The name of the protected attribute. A value of [entry rights] indicates the rights are for the object itself. A value of [all attributes] indicates the rights are applied to all attributes.
Privileges	SQL_INTEGER	An integer whose value represents the combination of the individual privileges that are granted.
Add Self	SQL_BIT	The trustee has rights to add or remove itself as an attribute value. This right is used only for attributes that contain object names as values, such as lists of group members or mailing lists.
Browse	SQL_BIT	The trustee has the right to see object in the NDS tree.
Compare	SQL_BIT	The trustee has the the right to compare the values of an attribute.
Create	SQL_BIT	The trustee has the right to create a new object in the NDS tree. This right is available only for container objects.
Delete	SQL_BIT	The trustee has the right to delete the object from the NDS tree.
Read	SQL_BIT	The trustee has the the right to read and compare the values of an attribute. The Read right implies the Compare right.
Rename	SQL_BIT	The trustee has the right to change the name of the object.
Supervisor	SQL_BIT	The trustee has all rights to the object, all of the object's attributes, or a specific attribute.

Column Name	SQL Data Type	Description
Write	SQL_BIT	The trustee has the right to add, change, or remove any values of the attribute. The Write property right implies the Add Self property right.

For example query statements and their results, see

- Section 4.10, “Simple Effective Rights Query,” on page 36
- Section 4.9, “Restricted Effective Rights Query,” on page 35

3.7 eDirectory Class Types

eDirectory classes are either effective or non-effective. Non-effective classes can be either auxiliary classes or super classes, but they cannot be used to create entries in the database. Effective classes are the base classes that are used to create entries in the database. They can also be super classes. From super classes, classes inherit attributes and may inherit other structural features such as naming and containment .

3.7.1 Super Classes

Since a super class is a class definition, it has its own relational database table, and you can generate reports from these super class tables. However, to generate reports without surprises, you need to be aware of how eDirectory and the ODBC driver use super classes.

- eDirectory has a hierarchical schema, and more than one class can inherit from the same super class. For example, Top is a super class of all effective classes, and ndsLoginProperties is a super class of User, Person, Organizational Person, Organization, and Organizational Unit.
- The ODBC driver determines the base set of entries for a table by performing an eDirectory query for all entries whose Object Class attribute contains the class associated with the table. The Object Class attribute for each entry is multi-valued and contains the names of the entry’s base class, its super classes, and auxiliary classes. For example, the Object Class attribute for User entry would always include the following classes: Top, ndsLoginProperties, Person, Organizational Person, and User.

Thus, if you run a report using the ndsLoginProperties table, the table could contain entries from the following base classes: User, Person, Organizational Person, Organization, and Organizational Unit. In addition, if your schema has been extended to include classes that define ndsLoginProperties as a super class such as residentialPerson and inetOrgPerson, then the table could contain entries from these classes.

When using a table from a super class, you should use the special column, NDS_Name, for the entry’s name rather than an attribute such as CN. Not all classes use CN as their naming attribute, and some super classes do not define a naming attribute. The NDS_Name column retrieves the entry’s name regardless of the naming attribute.

For an example query using a super class table, see Section 4.14, “Super Class Query,” on page 41.

For more information about super classes, see the *Novell eDirectory Schema Reference* (http://developer.novell.com/ndk/doc/ndslib/schm_enu/data/h4q1mn1i.html).

3.7.2 Auxiliary Classes

Auxiliary classes also require some special handling. Since they are a class definition, they have their own relational database table, and you can generate reports from them. To generate a report, you should be aware of the following features of auxiliary classes.

- Auxiliary classes are assigned to individual entries in the directory rather than to a base class. Therefore, not all entries of a base class will have the auxiliary class value in their Object Class attributes and entries of other base classes may have the auxiliary class value in their Object Class attributes. An auxiliary class can be assigned to any entry in the directory. For example, Partition is an auxiliary class in NDS 8.xx and is assigned to any container entry such as a Country, Organization, or Orgranizational Unit entry that becomes the partition's root.
- The ODBC driver determines the base set of entries for an auxiliary class table by performing an eDirectory query for all entries whose Object Class attribute contains the auxiliary class associated with the table. The Object Class attribute for each entry is multi-valued and contains the names of the any auxiliary classes assigned to the entry.

You should use the special column, NDS_Name, to include the entry's name in the report because most auxiliary classes do not include naming attributes and because not all base classes use the same naming attribute. (For more information about auxiliary classes, see the eDirectory Schema Reference.)

For an example query using an auxiliary class table, see Section 4.2, "Auxiliary Class Queries," on page 30.

An auxiliary class table contains only the attributes defined for the auxiliary class. At times you may want to create a query that returns columns for the auxiliary class attributes and base class attributes. To do this, you must perform a join between the base class and the auxiliary class. The join should be performed on the NDS_FullName field. For an example, see Section 4.6, "Join Query," on page 33.

Tasks

4

The following sections provide examples of simple to complex SQL query statements and sample tables that result from such queries.

4.1 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 Section 3.5, “Data Type Mappings,” on page 23. A value of one in a privilege column such as ACL_Read or ACL_Write indicates the privilege is granted to the indicated trustee. A value of zero indicates the trustee does not have that privilege. The O (Organization Name) attribute is the naming attribute for the Organization class.

SQL Statement

```
SELECT O, ACL_Trustee, ACL_Attribute, ACL_Read, ACL_Write from
Organization
```

Result Table

Table 4-1 *ACL Attribute Query Result*

O	ACL_Trustee	ACL_Attribute	ACL_Read	ACL_Write
org1	org1	Login Script	1	0
org1	org1	Print Job Configuration	1	0

O	ACL_Trustee	ACL_Attribute	ACL_Read	ACL_Write
ManyUsers	ManyUsers	Login Script	1	0
ManyUsers	ManyUsers	Print Job Configuration	1	0
FewUsers	FewUsers	Login Script	1	0
FewUsers	FewUsers	Print Job Configuration	1	0

4.2 Auxiliary Class Queries

In NDS 8.xx, Partition is an auxiliary class. The example queries produce results from a Novell eDirectory tree with the following configuration.

The following query searches the eDirectory tree and creates a report that lists all the containers which are partition roots regardless of the container's base class. The containers which have not been assigned to Partition auxiliary class do not appear in the report.

SQL Statement

```
SELECT NDS_Name FROM Partition
```

Result Table

NDS_Name
Acme,Inc.
England
Operations
Engineering

The following query restricts the query to containers which belong to the Organization class.

SQL Statement

```
SELECT NDS_Name FROM Partition
WHERE "Object Class" = 'Organization'
```

Result Table

NDS_Name
Operations
Engineering

4.3 Concatenation Query

The following example illustrates how to use the SetSeparator function to set the separator to the plus (+) character and then concatenate the values for the Member attribute into one row for GroupNDS objects.

SQL Statement

```
SELECT Member_S

FROM GroupNDS WHERE { fn SetSeparator('+') } <>',
```

Enter this statement into the query window, execute the statement, and retrieve the data.

Result Table

The SQL statement will generate a report similar to the following:

Table 4-2 *Concatenation Query Result Report*

NDS_Name	Member_S
Top Dogs	FJohnson.Fred's Widgets+GSwift.Fred's Widgets+ SBrady.Fred's Widgets

4.4 Container Class Query

The following SQL code queries the O attribute, the Telephone Number attribute, and the L (local) attribute from all Organization objects. Note that the specified column names are the same as the names of the corresponding attributes. The name of the table is Organization which is the class name for all Organization objects. The Telephone Number attribute name must be enclosed in quotes since it contains a space. In the NetWare Administrator utility the O, L, and Telephone Number attribute values are labeled as Name, Location, and Telephone respectively. Notice that a <null> result is returned for the FewUsers organization's telephone number. This simply means that the attribute has not been assigned a value in eDirectory.

SQL Statement

```
SELECT O, L, "Telephone Number" from Organization
```

Result Table

Table 4-3 *Container Class Query Result Report*

O	L	Telephone Number
org1	Salt Lake City	801-837-9999
ManyUsers	Provo	801-123-4567
FewUsers	San Jose	<Null>

4.5 Creating Custom Tables

In order to provide further customization and avoid ODBC tool limitations, you can create custom user tables. User table names start with an underscore. For example, if a table name "_userReport" has been created by the user then a query using it could look like, "Select * from _userReport". All user-defined tables will appear in the lists of tables displayed by tools like MSQuery, Excel, Crystal Reports, etc.

4.5.1 Defining Tables

To define a table:

1. Open the ODBC Data Source Administrator by clicking *Start* > *Settings* > *Control Panel*, then double-clicking the *ODBC Data Source Administrator* icon.
2. Add a new Novell NDS source or configure an existing one. The driver should say "*Novell ODBC, Driver for NDS*".
3. Select a Tree and press the "*User Defined Tables*" button. User Defined tables are based on the schema of an existing tree. If "*Use default context*" is selected then the schema of your default tree will be used.
4. Add a new Table, or change and existing one. The default directory is "`c:\winnt\odbc`". All files with a ".out" extension (Odbc User Table) will listed in the dialog. The "*Browse*" button can change this directory. Double clicking a table in the list, or selecting a table and clicking "*Modify*", brings up a dialog to modify that table.
5. Select attributes for the new table. There are two types of Classes in NDS: Base and Auxiliary Classes. In a user-defined table you can have attributes from only one base class. You can have attributes from many Auxiliary classes. The top half of the dialog is used for Base class attributes, and the bottom for the Auxiliary Attributes.
6. The left column represents the classes and attributes in the schema and the right side represents the attributes in the user-defined table. All Base Attributes added to the user-defined table will be erased if a new Base Class is selected. However, all Auxiliary attributes remains if a new Auxiliary class is selected.

4.5.2 Where are the User-Defined Tables Stored?

By default User-defined tables are stored in ‘UserDefinedTables' under the installation directory. So a typical default directory would be “`c:\novell\odbcnds\userDefinedTables`”. The "*Browse*" button can change this directory. This directory is stored in the Registry under the key, `HKEY_CURRENT_USER\Software\Novell\ODBCnds\path`.

All files with a ".out" extension (Odbc User Table) will listed as user-defined tables.

4.5.3 Defining Tables from a Text File

The user table is a standard text file. A User defined table should always begin with an underscore and have either an out extension or no extension.

The file format is as follows:

Line 1: Ignored by driver for now. In the future this includes version numbers and tree name.

Line 2: Base Class name.

Line 3 - End: All attributes, auxiliary and base. Column names should not include composite attribute suffixes or concatenation suffixes. NDS_* columns are automatically included in all tables.

4.6 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 4-4 *Join Query Result Report*

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

4.7 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 June 10, 1999 at 8:00 am. 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 Statement

```
SELECT CN, “Last Login Time” FROM UserNDS

WHERE “Last Login Time” < {ts ’1999-06-10 08:00:00.00’}
```

Result Table

Table 4-5 *Last Login Time Query Reslut Report*

CN	Last Login Time
dward0	1999-06-04 07:05:36.000
dward1	1999-06-08 09:17:12.000

CN	Last Login Time
dward2	1999-06-06 10:15:47.000
dward3	1999-06-08 08:37:04.000
dward4	1999-06-08 10:45:16.000

4.8 Parent Container Queries

The first SQL query results in entries from the Fred's Widgets organization as well as its subordinate organizational units. The second query illustrates the use of the NDS_Context column to get information from only the parent context.

SQL Statement

```
select NDS_Context, Surname from UserNDS
```

Result Table

Table 4-6 *Parent Container Query Result Report*

NDS_Context	Surname
Sales.Fred’s Widgets	Purdy
Sales.Fred’s Widgets	Anderson
Sales.Fred’s Widgets	Desmond
Sales.Fred’s Widgets	Knight
Sales.Fred’s Widgets	Turnet
Marketing.Fred’s Widgets	Gilbert
Marketing.Fred’s Widgets	Bergman
Marketing.Fred’s Widgets	Fairbanks
Marketing.Fred’s Widgets	Metcalf
Product Develpment.Fred’s Widgets	Sanders
Product Develpment.Fred’s Widgets	Smith
Product Develpment.Fred’s Widgets	Weirsdorf
Product Develpment.Fred’s Widgets	Hancock
Operations.Fred’s Widgets	Brown
Operations.Fred’s Widgets	Jones
Operations.Fred’s Widgets	Madsen
Operations.Fred’s Widgets	Garcia
Fred’s Widgets	Johnson

NDS_Context	Surname
Fred’s Widgets	Brady
Fred’s Widgets	Swift

SQL Statement

```
select NDS_Context, Surname from UserNDS where NDS_Context = ’Fred’’s
Widgets’
```

Result Table

Table 4-7 *Result Table*

NDS_Context	Surname
Fred’s Widgets	Johnson
Fred’s Widgets	Brady
Fred’s Widgets	Swift

4.9 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.

SQL Statement

```
 SELECT "Object Name", "Trustee Name", "Attribute", "Browse", "Delete"

FROM "Effective Rights"

WHERE "Trustee Name" LIKE ’*.FewUsers’ AND "Object Name" =
’dward0.FewUsers’

AND "Attribute" = ’[Entry Rights]’
```

Result Table

The query produces a 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).

Table 4-8 *Restricted Effective Rights Query Result Report*

Object Name	Trustee Name	Attribute	Browse	Delete
dward0.FewUsers	cjohnson0.FewUsers	[Entry Rights]	1	0
dward0.FewUsers	cjohnson1.FewUsers	[Entry Rights]	1	0
dward0.FewUsers	cjohnson2.FewUsers	[Entry Rights]	1	0
dward0.FewUsers	cjohnson3.FewUsers	[Entry Rights]	1	0
dward0.FewUsers	cjohnson4.FewUsers	[Entry Rights]	1	0
dward0.FewUsers	cjohnson5.FewUsers	[Entry Rights]	1	0
dward0.FewUsers	cjohnson6.FewUsers	[Entry Rights]	1	0
dward0.FewUsers	cjohnson7.FewUsers	[Entry Rights]	1	0
dward0.FewUsers	cjohnson8.FewUsers	[Entry Rights]	1	0
dward0.FewUsers	cjohnson9.FewUsers	[Entry Rights]	1	0
dward0.FewUsers	dward0.FewUsers	[Entry Rights]	1	0
dward0.FewUsers	dward1.FewUsers	[Entry Rights]	1	0
dward0.FewUsers	dward2.FewUsers	[Entry Rights]	1	0
dward0.FewUsers	dward3.FewUsers	[Entry Rights]	1	0
dward0.FewUsers	dward4.FewUsers	[Entry Rights]	1	0
dward0.FewUsers	dward5.FewUsers	[Entry Rights]	1	0
dward0.FewUsers	dward6.FewUsers	[Entry Rights]	1	0
dward0.FewUsers	dward7.FewUsers	[Entry Rights]	1	0
dward0.FewUsers	dward8.FewUsers	[Entry Rights]	1	0
dward0.FewUsers	dward9.FewUsers	[Entry Rights]	1	0

4.10 Simple Effective Rights Query

The following query illustrates how to obtain the read, write, and compare rights of user Sjones on the attributes of user JDoe.

SQL Statement

```
SELECT "Object Name", "Trustee Name", "Attribute", "Compare", "Read",
"Write"

FROM "Effective Rights"

WHERE "Object Name" = 'JDoe.Sales' AND "Trustee Name" =
'SJones.Marketing'
```

Results Table

The query produces a 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).

Table 4-9 *Simple Effective Rights Query Result Report*

Object Name	Trustee Name	Attribute	Compare	Read	Write
JDoe.Sales	SJones.Marketing	[Entry Rights]	0	0	0
JDoe.Sales	SJones.Marketing	[All Attributes Rights]	0	0	0
JDoe.Sales	SJones.Marketing	ACL	1	1	0
JDoe.Sales	SJones.Marketing	Back Link	1	1	0
JDoe.Sales	SJones.Marketing	Bindery Property	1	1	0
JDoe.Sales	SJones.Marketing	Cross Certificate Pair	1	1	0
JDoe.Sales	SJones.Marketing	CA Public Key	1	1	0
JDoe.Sales	SJones.Marketing	CN	1	1	0
JDoe.Sales	SJones.Marketing	Description	1	1	0
JDoe.Sales	SJones.Marketing	Facsimile Telephone Number	1	1	0
JDoe.Sales	SJones.Marketing	Home Directory	1	1	0
JDoe.Sales	SJones.Marketing	L	1	1	0
JDoe.Sales	SJones.Marketing	Login Allowed Time Map	1	1	0
JDoe.Sales	SJones.Marketing	Login Disabled	1	0	1
JDoe.Sales	SJones.Marketing	Login Expiration Time	1	1	0
JDoe.Sales	SJones.Marketing	Login Grace Limit	1	1	0
JDoe.Sales	SJones.Marketing	Login Grace Remaining	1	1	0
JDoe.Sales	SJones.Marketing	Login Intruder Address	1	1	0
JDoe.Sales	SJones.Marketing	Login Intruder Attempts	1	1	0
JDoe.Sales	SJones.Marketing	Login Intruder Reset Time	1	1	0
JDoe.Sales	SJones.Marketing	Login Maximum Simultaneous	1	1	0
JDoe.Sales	SJones.Marketing	Login Script	1	1	0
JDoe.Sales	SJones.Marketing	Login Time	1	1	0
JDoe.Sales	SJones.Marketing	Minimum Account Balance	1	1	0
JDoe.Sales	SJones.Marketing	EMail Address	1	1	0

Object Name	Trustee Name	Attribute	Compare	Read	Write
JDoe.Sales	SJones.Marketing	Network Address	1	1	0
JDoe.Sales	SJones.Marketing	Network Address Restriction	1	1	0
JDoe.Sales	SJones.Marketing	Obituary	1	1	0
JDoe.Sales	SJones.Marketing	Object Class	1	1	0
JDoe.Sales	SJones.Marketing	OU	1	1	0
JDoe.Sales	SJones.Marketing	Password Allow Change	1	1	0
JDoe.Sales	SJones.Marketing	Password Expiration Interval	1	1	0
JDoe.Sales	SJones.Marketing	Password Expiration Time	1	1	0
JDoe.Sales	SJones.Marketing	Password Minimum Length	1	1	0
JDoe.Sales	SJones.Marketing	Password Required	1	1	0
JDoe.Sales	SJones.Marketing	Password Unique Required	1	1	0
JDoe.Sales	SJones.Marketing	Physical Delivery Office Name	1	1	0
JDoe.Sales	SJones.Marketing	Postal Address	1	1	0
JDoe.Sales	SJones.Marketing	Postal Code	1	1	0
JDoe.Sales	SJones.Marketing	Postal Office Box	1	1	0
JDoe.Sales	SJones.Marketing	Print Job Configuration	1	1	0
JDoe.Sales	SJones.Marketing	Printer Control	1	1	0
JDoe.Sales	SJones.Marketing	Profile	1	1	0
JDoe.Sales	SJones.Marketing	Public Key	1	1	0
JDoe.Sales	SJones.Marketing	Higher Privileges	1	1	0
JDoe.Sales	SJones.Marketing	Security Equals	1	1	0
JDoe.Sales	SJones.Marketing	See Also	1	1	0
JDoe.Sales	SJones.Marketing	S	1	1	0
JDoe.Sales	SJones.Marketing	SA	1	1	0
JDoe.Sales	SJones.Marketing	Surname	1	1	0
JDoe.Sales	SJones.Marketing	Telephone Number	1	1	0
JDoe.Sales	SJones.Marketing	Title	1	1	0
...	...	...	...	...	...

This example table is not complete. The User class has over 90 attributes in a default NetWare 5 installation. Since the Effective Rights table is returning rights to attributes, the query returns results for all attributes defined for the class unless restricted to a selected set. Also, the attributes come back in an unsorted order, unless you include an order statement.

4.11 Simple Search Query

The following SQL statement uses two User class attributes (Given Name and Surname) and one special column name NDS_Tree. It gathers information about users so it uses the UserNDS table. These attributes are public read attributes, so any logged in eDirectory user can execute this statement and generate a report.

SQL Statement

```
SELECT "Given Name", "Surname", "NDS_Tree"

FROM UserNDS
```

Enter this statement into the query window, execute the statement, and retrieve the data.

Result Table

The SQL statement will generate a report similar to the following:

Table 4-10 *Simple Search Query Result Report*

Given Name	Surname	NDS_Tree
Kris	Hammons	NOVELL_INC
Lynn	Thayn	NOVELL_INC
Jan	Aitchison	NOVELL_INC
Alex	Buckley	NOVELL_INC
Tom	Turner	NOVELL_INC
Cory	Petersen	NOVELL_INC

4.12 Simple Update Queries

You can update the NDS Data Source using simple insert, update and delete SQL queries. To update the Data Source, you should have rights to write onto Novell eDirectory.

Note that:

- While inserting an object, you need to specify the Nds_Name as an attribute (column in SQL).
- The driver does not support positioned update, delete, or insertion.
- If the attribute value is in decimals it will be treated as an Integer while adding to Novell eDirectory.
- You cannot create or drop a table (create or delete an object class, which requires schema extension) using the ODBC driver.

- For a single entry, SQL semantics does not permit using multiple values for a single column. Hence, this driver does not allow you to Insert/Update an entry with multiple values for one particular attribute. However, the driver will allow you to put in just one value for a multi-valued attribute.

4.12.1 Insert

The following SQL statement enables you to add an user object to the NDS Data Source:

```
INSERT into UserNDS (Surname, Nds_Name, UID) values (’Duke’, ’Luke’, 8)
```

4.12.2 Update

The following SQL statement enables you to modify an user object in the NDS Data Source:

```
UPDATE UserNDS SET Surname = ’Kuke’, UID = UID + 20 WHERE Nds_Name LIKE ’%uke’
```

4.12.3 Delete

The following SQL statement enables you to delete an user object from the NDS Data Source:

```
DELETE FROM UserNDS WHERE Nds_Name = ’Luke’ OR UID > 7
```

4.13 Sorting Query

This example illustrates the use of the “order by” clause to sort a table on multiple columns.

SQL Statement

```
SELECT NDS_Context, Surname

FROM UserNDS

ORDER BY NDS_Context, Surname
```

Result Table

Table 4-11 *Sorting Query Result Report*

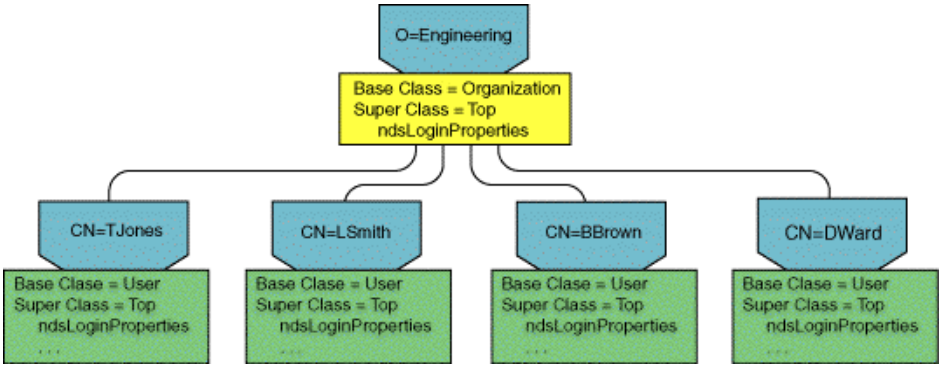
NDS_Context	Surname
Marketing.Fred’s Widgets	Bergman
Marketing.Fred’s Widgets	Fairbanks
Marketing.Fred’s Widgets	Gilbert
Marketing.Fred’s Widgets	Metcalf
Operations.Fred’s Widgets	Metcalf
Operations.Fred’s Widgets	Garcia

NDS_Context	Surname
Operations.Fred's Widgets	Garcia
Operations.Fred's Widgets	Madsen
Product Development.Fred's Widgets	Hancock
Product Development.Fred's Widgets	Sanders
Product Development.Fred's Widgets	Smith
Product Development.Fred's Widgets	Weirsdorf
Sales.Fred's Widgets	Anderson
Sales.Fred's Widgets	Desmond
Sales.Fred's Widgets	Knight
Sales.Fred's Widgets	Purdy
Sales.Fred's Widgets	Turnet

4.14 Super Class Query

The ndsLoginProperties class is a super class of Organization, Organizational Unit, Person, Organizational Person, and User. The following example query is based on an eDirectory tree with the following structure.

Figure 4-1 *Super Class Query*



The following query searches the eDirectory tree for all entries that have ndsLoginProperties as a super class. It returns entries from multiple base classes.

SQL Statement

```
SELECT NDS_Name FROM ndsLoginProperties
```

Result Table

NDS_Name
Engineering

NDS_Name
BBrown
TJones
LSmith
DWard

4.15 User Class Query

This example queries the attributes of a user. Notice that the table name, UserNDS, is not the same as the object class name, User. This is because user is a keyword in SQL. Also, the NDS_Context, NDS_FullName, and NDS_Tree columns do not correspond to eDirectory attributes but are special columns available for each entry.

SQL Statement

```
select CN, Surname, NDS_Context, NDS_FullName, NDS_Tree from UserNDS
```

Result Table

Table 4-12 *User Class Query Result Report*

CN	Surname	NDS_Context	NDS_FullName	NDS_FullName
admin	admin	Widgets	admin.Widgets	BGB
Cory	Jones	Sales.Widgets	Cory.Sales.Widgets	BGB
Kris	Smith	Operations.Widgets	Kris.Operations.Widgets	BGB
Lynn	Brown	Operations.Widgets	Lynn.Operations.Widgets	BGB
Michael	Anderson	Development.Widgets	Michael.Development.Widgets	BGB
Mike	Anderson	Development.Widgets	Michael.Development.Widgets	BGB
Kim	Wilson	Development.Widgets	Kim.Development.Widgets	BGB

Notice that Michael Anderson is listed twice in the table. CN is a multi-valued attribute and two values have been entered: Michael and Mike. Only the first value is used in the object's distinguished name, or as the column labels it, in the object's NDS full name.

4.16 Optimizing the ODBC Driver Performance

Following are the few optimization methods suggested when using the ODBC driver for eDirectory. Using the following SQL sample query, you can make use one or more of these methods to improve the ODBC driver performance.

```
SELECT "_GroupNDS"."CN", "_UserNDS"."CN", "_UserNDS"."employeeStatus",
"_OU"."OU" FROM   "_GroupNDS" "_GroupNDS", "_OU" _OU", "_UserNDS"
```

```
"_UserNDS" WHERE  ("_GroupNDS"."NDS_Context"="_OU"."NDS_FullName") AND
("_GroupNDS"."Member"="_UserNDS"."Group Membership") AND
"_GroupNDS"."CN"='FP2111' ORDER BY "_GroupNDS"."CN"
```

Method 1: Use one more static comparison in the query. In the above query, "_GroupNDS"."CN"='FP2111' implies a static comparison which makes the query perform faster than a query without it. We recommend you to use the static comparison for attributes which have unique value. For example, CN.

Method 2: Use static comparison in the largest table selected. For example, if you are reading from User and OU and if the number of Users are larger than OU, use the static comparison in the User. This considerably reduces the time taken for search.

Method 3: Order the tables with the slowest spin in the left, when selecting the table in the from clause. For example, from the previous query, order the tables in the following sequence:

```
SELECT "_GroupNDS"."CN", "_UserNDS"."CN",
"_UserNDS"."employeeStatus","_OU"."OU" FROM
"_UserNDS","_OU","_GroupNDS"
```

ODBC Driver Advanced Write Capability

5

The Novell® ODBC Driver for Novell eDirectory™ now has an advanced write capability specified by the ODBC standards. It supports positioned update and/or delete and allows you to open a writable cursor for the fields you wish to modify.

The positioned update and/or delete is done by positioning a writable cursor on a specific row in a result set which is obtained from a select request.

5.1 Cursor

A cursor is a tool that allows you to step through a result set row-by-row for row-conditional processing. Applications can perform multiple operations on each individual row in a given result set. A cursor is opened on the result set by execution of the query.

A cursor is used when the program needs to perform update or delete operations on specific rows in a result set. For example, the program might retrieve some rows from the query results, display them on the screen for the user, and then respond to a user's request to update or delete data.

For updating data using a cursor, the SELECT statements that are used to generate the result set must explicitly specify FOR UPDATE or FOR UPDATE OF column_list in the SELECT statement. This will open a cursor for that column_list that you can update.

For example, `SELECT * FROM account FOR UPDATE OF balance`

If the statement has been not declared with FOR UPDATE, it defaults to a read only cursor and you will not be allowed to do cursor updates or deletes.

5.1.1 Updating Using the Cursor

The syntax for update and delete statements using cursors is:

```
UPDATE tablename SET column = value [, SET column = value...]  WHERE
CURRENT OF cursorname
```

OR

```
DELETE FROM tablename WHERE CURRENT OF cursorname
```

This will update or delete the row on which the cursor is currently positioned. If the cursor is positioned on AFTER_LAST_ROW, the operation will throw an error ERR_CURSORSTATE.

5.1.2 Getting the Cursor Name

The ODBC driver automatically generates a cursor name when you call SQLAllocStmt to allocate a statement handle. You can use SQLGetCursorName to get the full name of the cursor associated with a specific statement handle.

The prototype for SQLGetCursorName is:

```
RETCODE SQLGetCursorName(HSTMT      hstmt,
                         UCHAR FAR *szCursor,
                         SWORD      cbCursorMax,
                         SWORD FAR *pcbCursor
                            );
```

You can use SQLSetCursorName to set the cursor name of an active statement handle. You have to use SQLSetCursorName to change the cursor name before executing the SELECT statement.

The prototype for SQLSetCursorName is:

```
RETCODE SQLSetCursorName( HSTMT   hstmt,
                         UCHAR FAR *szCursor,
                         SWORD      cbCursor
                            );
```

5.1.3 Excerpt of a Sample Program

```
/* Execute the select query */

retcode = SQLExecDirect(hstmtSlt, (unsigned char*)SELECT
      Nds_Name,SurName,\"Telephone number\" from UserNDS FOR UPDATE OF
\"Telephone
Number\",SQL_NTS);

if (retcode == SQL_SUCCESS || retcode == SQL_SUCCESS_WITH_INFO) {
    retcode = SQLBindCol(hstmtSlt, 1, SQL_C_CHAR, ndsname, C_LENGTH,
    &cbndsname);

    retcode = SQLBindCol(hstmtSlt, 2, SQL_C_CHAR, sname, C_LENGTH,
    &cbsname);

    retcode = SQLBindCol(hstmtSlt, 3, SQL_C_CHAR, telnum, C_LENGTH,
    &cbtelnum);

} else  {
     printf ("Error while executing the query!\n");
     exit(1);
}
/* Read through the resultset until the cursor points to the required
row */
do {
      retcode = SQLFetch(hstmtSlt);
}
while ((retcode == SQL_SUCCESS || retcode == SQL_SUCCESS_WITH_INFO)
&&
       (strcmp((const char*)ndsname, "tpeter") != 0 ));
/* Get the cursor name */
SQLGetCursorName(hstmtSlt, cursor, CURSOR_LEN, &cursorLen);

/* Allocate a statement for update */
SQLAllocStmt(hdbc, &hstmtUpdt);

sprintf(updsql,"UPDATE UserNDS SET \"Telephone Number\"='01-1022-1122'
```

```
WHERE current of %s", cursor);

retcode = SQLExecDirect(hstmtUpdt,(unsigned char*)updsql, SQL_NTS);

if(retcode == SQL_SUCCESS || retcode == SQL_SUCCESS_WITH_INFO) {
      // Successfully updated
 }
```

Troubleshooting

6

The following sections describe some of the common problems that you might encounter while using the Novell® ODBC driver for Novell eDirectory™ and provide instructions for uninstalling the driver.

6.1 Common Problems

There are three common problems that can cause the Novell ODBC Driver for eDirectory to fail in producing the report you requested.

Insufficient Resources

Some reports generate so much data that the workstation runs out of memory or disk space. An eDirectory Data Source can contain thousands and millions of objects, depending on the eDirectory versions that are running in the tree. If you are getting "out of resource" types of errors, restrict your query to select fewer items:

- Select fewer attributes or specify the attributes rather than using a wildcard to include all attributes.
- Examine the attributes you select to ensure that only a few of them are multi-valued.
- Restrict the number of objects selected by specifying only one container.

For more information on how eDirectory attributes and objects can cause a report to become extremely large, see:

- Section 3.4, “Multi-Valued Attributes,” on page 20
- Section 3.3, “Composite Attributes,” on page 19
- Section 3.6, “Effective Rights Table,” on page 25

eDirectory Rights

You must have eDirectory rights to the data you request in your report. If you do not have at least Browse rights to the objects and Read rights to the attributes, your report contains NULL in all the columns. You can use the Effective Rights Table to generate a report that indicates your rights to the objects and attributes for which you are interested in generating a report.

SQL Statement Errors

You must use the correct table and column names in SQL statements.

- If the table or column name has a space, the name must be enclosed in double quotes.
- If the attribute uses a syntax that creates multiple columns in the table, the column name for the attribute includes a field specifier.

For information on how attributes are split into multiple columns, see Section 3.3, “Composite Attributes,” on page 19.

For information on how to discover the tables and columns available in your eDirectory tree, see Possible eDirectory Tables and Columns.

Jet Engine Limit of 255 Columns

Many Microsoft* tools use the Jet Engine for ODBC access. Because Jet limits the number of columns to 255 you may not have access to all the columns you need. The following are workarounds for this limitation:

- Create a User-defined table to access the columns you need.
- Select ODBC options instead of Jet, if available. For example the ‘defaultType' property of Visual Basic data access objects can be switched from ‘useJet' to ‘useODBC'.

Run-time Error 3061 Too Few Parameters, While Using Microsoft Jet Database Engine

You might get this error if the Jet is parsing the query incorrectly. To resolve this, use the dbOpenSnapshot and dbSQLPassThrough flags with the OpenRecordset() or any equivalent function. For more information, see the sample code for Browse under vb.

While using DAO over ODBCDirect, if the time required to retrieve the result set exceeds the default or user specified query timeout interval, you will get a "Query Cancelled" message from the ODBC Driver, followed by a "Execution Cancelled" message.

To avoid this timeout problem, set the QueryTimeout value to 0 as follows:

```
Dim conODBCDirect As DAO.Connection
conODBCDirect.QueryTimeout = 0
```

6.2 Uninstalling the Novell ODBC Driver for eDirectory

To uninstall the Novell ODBC Driver for eDirectory, follow these steps.

1. Go to *Start Menu* > *Control Panel* > *Add/Remove Programs*.
2. Select *Novell ODBC Driver For eDirectory*.
3. Click the *Add/Remove* button.

This invokes the uninstallation program and removes the `ODBCNDS.DLL` file and the sample applications from your hard drive.

Revision History

A

The following table lists all changes made to the Novell® ODBC driver for Novell eDirectory™ documentation:

February 2007	Added addtional type for update syntax on Section 2.7, “Inserting Updating or Deleting Entries from the eDirectory Data Source“
March 2006	Fixed formatting issues.
October 2005	Added information on Section 4.16, “Optimizing the ODBC Driver Performance,” on page 42.
March 2005	Changed the component name ODBC Driver for NDS to ODBC Driver for eDirectory Read-Write in required instances.
October 2004	Made the following changes: • Added information on Chapter 5, “ODBC Driver Advanced Write Capability,” on page 45. • Organized the document to improve navigability.
February 2004	Made the following changes: • Added information on Simple Update Queries. • Renamed the product name from “NDS” to “Novell eDirectory” at relevant instances.
October 2003	Added troubleshooting tips.
June 2001	Added user defined tables.
May 2000	Added information about super classes and auxiliary classes. Added tasks for performing queries to the super and auxiliary classes and a task for a join query.