IBM® DB2® Life Sciences Data Connect

Planning, Installation, and Configuration Guide

Version 7

IBM® DB2® Life Sciences Data Connect

Planning, Installation, and Configuration Guide

Version 7

Before using this information and the product it supports, be sure to read the general information under “Notices” on page 71.

Contents

About this book

This book contains:

- An introduction to DB2 Life Sciences Data Connect and how it fits into the IBM Life Sciences DiscoveryLink offering, a comprehensive set of software and services tailored to the life sciences
- Installation instructions for DB2 Life Sciences Data Connect
- Instructions for adding data sources to a federated system by registering wrappers. Wrappers are modules that enable you or an application to communicate with a data source using SQL statements.

Technical changes to the text are indicated by a vertical line to the left of the change.

Who should read this book

This book is for administrators who are setting up a federated database environment for life sciences research and development data, and for application programmers who are developing applications for such an environment.

Online information

This section provides Web addresses and e-mail addresses related to this product.

http://www.ibm.com/software/data/db2/lifesciencesdataconnect/
DB2 Life Sciences Data Connect product website

http://www.ibm.com/solutions/lifesciences/discoverylink.html
DiscoveryLink website

http://www.ibm.com/solutions/lifesciences/
IBM Life Sciences website

ls@us.ibm.com
IBM Life Sciences email address

Conventions

This book uses these highlighting conventions:

Boldface type
Indicates commands and graphical user interface (GUI) controls (for example, names of fields, names of folders, menu choices).

`Monospace type`
Indicates examples of coding or of text that you type.

Italic type
Indicates variables that you should replace with a value. Italic type also indicates book titles and emphasizes words.

UPPERCASE TYPE
Indicates SQL keywords and names of objects (for example, tables, views, and servers).

How to read the syntax diagrams

Throughout this book, syntax is described using the structure defined as follows:

Read the syntax diagrams from left to right and top to bottom, following the path of the line.

The ▸▸— symbol indicates the beginning of a statement.

The —▸ symbol indicates that the statement syntax is continued on the next line.

The ▸— symbol indicates that a statement is continued from the previous line.

The —▸◂ symbol indicates the end of a statement.

Required items appear on the horizontal line (the main path).

Optional items appear below the main path.

If an optional item appears above the main path, that item has no effect on the execution of the statement and is used only for readability.

If you can choose from two or more items, they appear in a stack.

If you *must* choose one of the items, one item of the stack appears on the main path.

If choosing none of the items is an option, the entire stack appears below the main path.

If one of the items is the default, it will appear above the main path and the remaining choices will be shown below.

An arrow returning to the left, above the main line, indicates an item that can be repeated. In this case, repeated items must be separated by one or more blanks.

If the repeat arrow contains a comma, you must separate repeated items with a comma.

A repeat arrow above a stack indicates that you can make more than one choice from the stacked items or repeat a single choice.

Keywords appear in uppercase (for example, `FROM`). They must be spelled exactly as shown. Variables appear in lowercase (for example, `column-name`). They represent user-supplied names or values in the syntax.

If punctuation marks, parentheses, arithmetic operators, or other such symbols are shown, you must enter them as part of the syntax.

Sometimes a single variable represents a set of several parameters. For example, in the following diagram, the variable `parameter-block` can be replaced by any of the interpretations of the diagram that is headed **parameter-block**:

parameter-block:

Adjacent segments occurring between "large bullets" (●) may be specified in any sequence.

The above diagram shows that item2 and item3 may be specified in either order. Both of the following are valid:

```
STATEMENT item1 item2 item3 item4
STATEMENT item1 item3 item2 item4
```

How to send your comments

Your feedback helps IBM to provide quality information. Please send any comments that you have about this book or other DB2 documentation. You can use any of the following methods to provide comments:

- Send your comments from the Web. You can access the IBM Data Management online readers' comment form at http://www.ibm.com/software/data/rcf
- Send your comments by e-mail to comments@vnet.ibm.com. Be sure to include the name of the product, the version number of the product, and the name and part number of the book (if applicable). If you are commenting on specific text, please include the location of the text (for example, a chapter and section title, a table number, a page number, or a help topic title).

Chapter 1. What is DB2 Life Sciences Data Connect?

This chapter introduces you to the DB2 Life Sciences Data Connect product, the IBM Life Sciences DiscoveryLink offering, and the general steps involved in setting up a system to query life sciences data.

DB2 Life Sciences Data Connect

IBM DB2 Life Sciences Data Connect enables a DB2 federated system to integrate genetic, chemical, biological, and other research data from distributed sources. A DB2 federated system is a distributed computing system that consists of a DB2 Universal Database server and multiple data sources from which the DB2 Universal Database server retrieves data.

With a federated system, you or an application can use SQL statements to query, retrieve, and join data that can be located in several heterogeneous data sources, such as relational databases from IBM, Oracle, Sybase, and Microsoft, and non-relational data sources, such as table-structured files. Figure 1 illustrates a federated system using DB2 Life Sciences Data Connect to access multiple sources of research data.

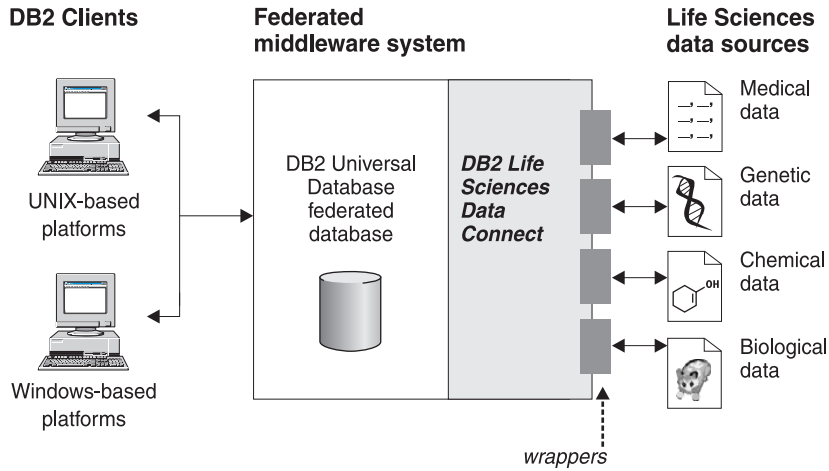


Figure 1. Accessing life sciences data with DB2 Life Sciences Data Connect

A DB2 federated system includes clients, a database to which the clients submit queries (called a federated database), an interface through which the federated database communicates with data sources, and the data sources themselves.

The mechanism by which a federated server communicates with a data source is called a *wrapper*. To implement a wrapper, the server uses routines stored in a library called a *wrapper module*. These routines allow the server to perform operations such as connecting to a data source and retrieving data from it iteratively.

After a federated system is set up, the information in data sources can be accessed as though it is in one large database. Users and applications send queries to one federated database, which retrieves data from multiple data sources. Applications work with the federated database just like with any other DB2 database.

For more information on federated systems, see the *DB2 SQL Reference*.

IBM Life Sciences DiscoveryLink

The DiscoveryLink offering is a set of middleware software and services tailored specifically to life sciences research and development requirements for integrating data from multiple heterogeneous data sources.

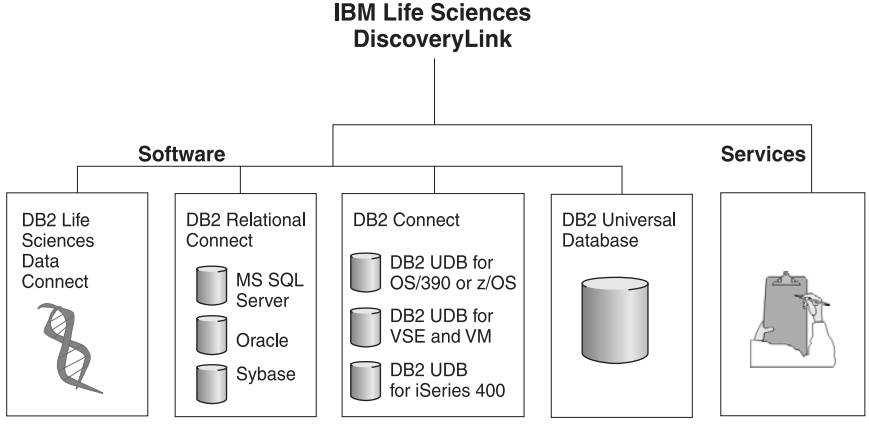


Figure 2. IBM Life Sciences DiscoveryLink

For example, with DiscoveryLink, you can use a single SQL statement to integrate protein sequence data from an Oracle database in Switzerland, chemical structure data from a Sybase database in Japan, and spectroscopic

data stored in table-structured flat files on your local area network. The data appears as if it is in one virtual database.

Software components include:

DB2 Life Sciences Data Connect
For accessing life sciences data.

DB2 Relational Connect
For accessing Oracle, Sybase, and Microsoft relational databases. For more information on DB2 Relational Connect, see the *IBM DB2 Universal Database Release Notes Version 7.2/Version 7.1 FixPak 4*.

DB2 Connect
For accessing DB2 database servers on host systems. For more information on DB2 Connect, see the *DB2 Connect User's Guide*.

DB2 Universal Database
To optimize queries and integrate results across multiple heterogeneous data sources. For more information on DB2 Universal Database, see the *DB2 Administration Guide*.

For more information on DiscoveryLink software and services, see "Online information" on page v.

Querying life sciences data

To query and retrieve data located in life sciences data sources, you must first install DB2 Life Sciences Data Connect.

After you install DB2 Life Sciences Data Connect, configure the wrapper to the data source. This process is known as registering the wrapper.

Chapter 2. Installing DB2 Life Sciences Data Connect

This chapter describes how to install DB2 Life Sciences Data Connect to query and retrieve life sciences data on Windows NT, Windows 2000, AIX, HP-UX, Linux, and Solaris Operating Environment. Table 1 shows the DB2 Life Sciences Data Connect wrappers on each platform.

Table 1. DB2 Life Sciences Data Connect wrappers by platform

Wrapper	Windows NT / Windows 2000	AIX	HP-UX	Linux	Solaris Operating Environment
Table-structured files	X	X	X	X	X
Documentum		X			
Excel	X				

After DB2 Life Sciences Data Connect is installed, you must register the wrappers for the various data sources to add them to your federated system. Instructions for registering each life sciences wrapper are provided in the chapters listed in Table 2.

Table 2. Where to find information on each life sciences wrapper

Wrapper	Chapter
Table-structured files	"Chapter 3. Using table-structured files as data sources" on page 9
Documentum	"Chapter 4. Using Documentum as a data source" on page 23
Excel	"Chapter 5. Using Excel as a data source" on page 55

Before installing

Before you install DB2 Life Sciences Data Connect on your federated server:

- Confirm that you have one of the following products installed on the federated server:
  - DB2 Universal Database Enterprise Edition
  - DB2 Universal Database Enterprise - Extended Edition
- Make sure the database has Federated Database System Support turned on. To check this setting, run the following command from the DB2 Command Line Processor:

  ```
  GET DATABASE MANAGER CONFIGURATION
  ```

This command displays all of the database parameters and their current settings. Confirm that the FEDERATED parameter is set to YES.

If the FEDERATED parameter is set to NO, run the following command from the DB2 Command Line Processor:

```
UPDATE DATABASE MANAGER CONFIGURATION USING FEDERATED YES
```

Installing DB2 Life Sciences Data Connect on AIX, HP-UX servers, Linux, and Solaris Operating Environment

To install DB2 Life Sciences Data Connect on an AIX, HP-UX, Linux, and Solaris Operating Environment federated server, use the db2setup utility.

Note: The screens that are displayed when you use the db2setup utility depend on what software products are installed on the federated server. These steps assume that DB2 Life Sciences Data Connect is not installed.

1. Log in as a user with root authority.
2. Insert and mount your DB2 Life Sciences Data Connect CD-ROM. For information on how to mount a CD-ROM, see the *DB2 for UNIX Quick Beginnings* manual.
3. Change to the directory where the CD-ROM is mounted by entering the **cd** */cdrom* command, where *cdrom* is the mount point for your product CD-ROM.
4. Type the following command:

   ```
   ./db2setup
   ```

   The DB2 Setup Utility window opens.
5. Press the space bar to select Distributed Access for Life Sciences data sources. An asterisk is displayed next to the option when it is selected.
6. Select **OK** and press the Enter key. The DB2 Services window opens.
7. In the DB2 Services window, you can choose to set up an existing DB2 instance for DB2 Life Sciences Data Connect. Use the db2setup installation help to guide you through setup, and through the remaining installation steps.

   When the installation is complete, DB2 Life Sciences Data Connect will be installed in the directory along with your other DB2 products.

   - On DB2 for AIX servers, the directory is /usr/lpp/db2_07_01
   - On DB2 for Solaris Operating Environment servers, the directory is /opt/IBMdb2/V7.1
   - On DB2 for HP-UX servers, the directory is /opt/IBMdb2/V7.1
   - On DB2 for Linux servers, the directory is /usr/IBMdb2/V7.1

Installing DB2 Life Sciences Data Connect on Windows NT and Windows 2000 servers

1. Log on to the federated server with the user account you created to perform the DB2 Universal Database installation.
2. Shut down any programs that are running so that the setup program can update files as required.
3. Invoke the setup program. You can invoke the setup program either automatically or manually. If the setup program fails to start automatically, or if you want to run the setup in a different language, invoke the setup program manually.
   - To automatically invoke the setup program, insert the DB2 Life Sciences Data Connect CD into the drive. The auto-run feature automatically starts the setup program. The system language is determined, and the setup program for that language is launched.
   - To manually invoke the setup program:
     a. Click **Start**, then click **Run**.
     b. In the **Open** field, type the following command:

        ```
        x:\setup /i language
        ```

        where:

        x: Represents your CD-ROM drive.

        language
        Represents the code for your language (for example, EN for English).

     c. Click **OK**.

   The installation launchpad opens.
4. Click **Install** to begin the installation process.
5. Follow the prompts in the setup program.

   When the installation is complete, DB2 Life Sciences Data Connect is installed in the install directory with other DB2 products. The default install directory is \sqllib.

After installing

After installation, wrapper library files are placed on your system. These libraries are used during the wrapper registration process. The default filename for each library, by supported platform, is listed in Table 3.

Table 3. Default wrapper library names by platform

Wrapper	Windows NT / Windows 2000	AIX	HP-UX	Linux	Solaris Operating Environment
Table-structured files	liblsfile.dll	liblsfile.a	liblsfile.sl	liblsfile.so	liblsfile.so
Documentum		liblsdctm.a			
Excel97 and Excel98	liblsexcel97.dll				
Excel2000	liblsexcel2k.dll				

Chapter 3. Using table-structured files as data sources

This chapter describes:

- Table-structured files
- How to add a table-structured file data source to a federated system
- Limitations and considerations
- The file access control model used
- Optimization tips
- Messages you might encounter when working with table-structured files

What are table-structured files?

A table-structured file has a regular structure consisting of a series of records, where each record contains the same number of fields, separated by an arbitrary delimiter. Null values are represented by two delimiters next to each other.

The following example shows the contents of a file called DRUGDATA1.TXT. It contains three records, each with three fields, separated by commas:

```
234,DrugnameA,Manufacturer1
332,DrugnameB,Manufacturer2
333,DrugnameC,Manufacturer2
```

The first field is the drug's unique ID number. The second field is the name of the drug. The third field is the name of the manufacturer who produces the drug.

Types of table-structured files

Table-structured files can be sorted or unsorted.

Sorted files

DRUGDATA1.TXT contains sorted records. The file is sorted by the first field, the drug's unique ID number. This field is the primary key because it is unique for each drug. Sorted files must be sorted in ascending order.

```
234,DrugnameA,Manufacturer1
332,DrugnameB,Manufacturer2
333,DrugnameC,Manufacturer2
```

Unsorted files

DRUGDATA2.TXT contains unsorted records. There is no order to the way the records are listed in the file.

```
332,DrugnameB,Manufacturer2
234,DrugnameA,Manufacturer1
333,DrugnameC,Manufacturer2
```

The wrapper can search sorted data files much more efficiently than non-sorted files.

How DB2 Life Sciences Data Connect works with table-structured files

Using a module called a wrapper, DB2 Life Sciences Data Connect can process SQL statements that query data in a table-structured file as if it were contained in an ordinary relational table or view. This enables data in a table-structured file to be joined with relational data or data in other table-structured files.

For example, suppose that the table-structured file DRUGDATA1.TXT is located on your computer in your laboratory. Trying to query this data and match it up with other tables from other data sources that you use can be tedious.

After you register DRUGDATA1.TXT with DB2 Life Sciences Data Connect, the file behaves as if it is a relational data source. You can now query the file together with other relational and non-relational data sources and analyze the data together.

For example, you could run the following query:

```
SELECT * FROM DRUGDATA1 ORDER BY DCODE
```

This query produces the following results.

Dcode	Drug	Manufacturer
234	DrugnameA	Manufacturer1
332	DrugnameB	Manufacturer2
333	DrugnameC	Manufacturer2

Adding table-structured files to a federated system

To add a data source for a table-structured file to a federated server:

1. Register the wrapper using the CREATE WRAPPER command.
2. Optional: Set the DB2_DJ_COMM environment variable to improve query performance.
3. Register the server using the CREATE SERVER command.
4. Register nicknames using the CREATE NICKNAME command for all table-structured files.

These steps are explained in detail in this section. The commands can be run from the DB2 Command Line Processor.

Step 1: Registering the wrapper

Use the CREATE WRAPPER statement to specify which wrapper will be used to access table-structured files. Wrappers are mechanisms that federated servers use to communicate with and retrieve data from data sources.

Wrappers are installed on your system as library files. Table 3 on page 8 lists the default library names by platform for the table-structured file wrapper. These library files must be registered using the CREATE WRAPPER statement before the wrapper can be used.

For example, to register a wrapper on AIX, run the following statement:

```
CREATE WRAPPER laboratory_flat_files LIBRARY 'liblsfile.a'
```

In this example, `laboratory_flat_files` is the name chosen for the wrapper. This name must be unique within the database in which the wrapper is being registered. The required library name for the table-structured file wrapper on AIX is liblsfile.a.

The library name is installed as liblsfile.a by default, but it might have been customized during installation. Check with your system administrator for the correct name.

For more information on the CREATE WRAPPER statement, see the *DB2 SQL Reference*.

Step 2: Optional: Setting the DB2_DJ_COMM environment variable

To improve performance when table-structured files are accessed, set the DB2_DJ_COMM environment variable. This variable determines whether the federated server loads the wrapper upon initialization. Set the DB2_DJ_COMM environment variable to include the wrapper library that corresponds to the wrapper that you specified in "Step 1: Registering the wrapper". For example:

```
export DB2_DJ_COMM='liblsfile.a'
```

Ensure that there are no spaces on either side of the equal sign (=).

For more information about the DB2_DJ_COMM environment variable, see the *DB2 Administration Guide*.

Step 3: Registering the server

Use the CREATE SERVER statement to define servers that will access table-structured files. A server can be configured to access either sorted or unsorted table-structured files, but a single server cannot be used for both.

However, you can define one server to manage sorted table-structured files, and another server to manage unsorted table-structured files. For example:

```
CREATE SERVER biochem_lab TYPE SORTED VERSION 1.0 WRAPPER laboratory_flat_files
OPTIONS (NODE 'biochem_node1')
```

In this example, `biochem_lab` is the name assigned to the table-structured file server. The name must be unique to the database in which the server is being registered.

The TYPE keyword is required. It is used to determine which search algorithm is used by the server. Specify SORTED or UNSORTED, depending on the data source. For more information on sorted and unsorted file types, see "Types of table-structured files" on page 9.

VERSION is required. It must be set to 1.0.

The wrapper name, in this example, is `laboratory_flat_files`. This is the name previously specified on the CREATE WRAPPER statement.

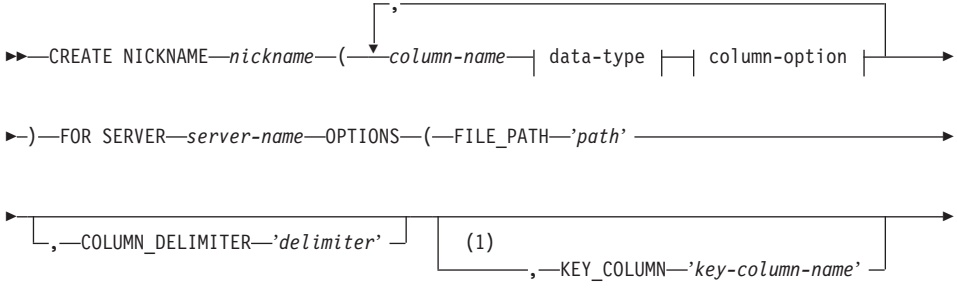
The NODE option is required. It is the name given to the local node. It can contain any text string you like.

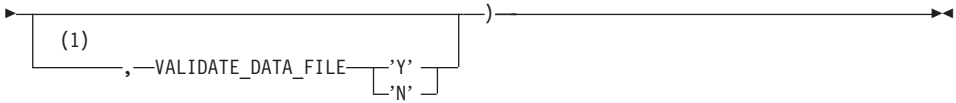
For more information about the CREATE SERVER statement, see the *DB2 SQL Reference*.

Step 4: Registering nicknames

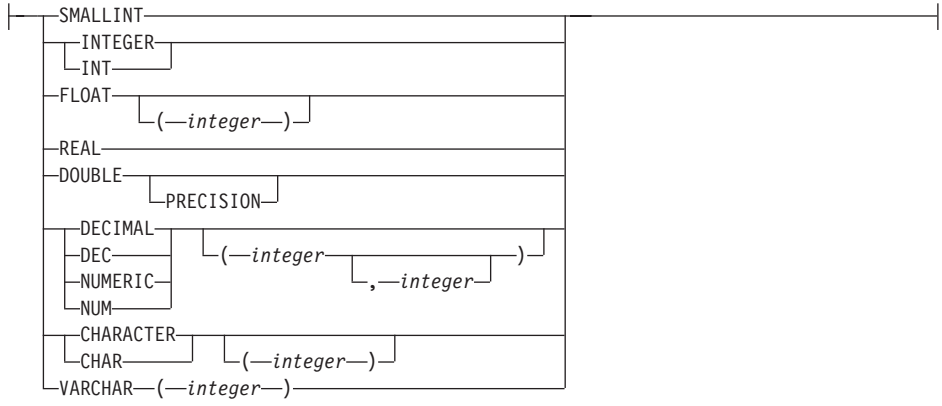
Use the CREATE NICKNAME statement to register a nickname for each table-structured file that you want to access using any of the servers that you registered in "Step 3: Registering the server" on page 11. Nicknames are used when you refer to a table-structured file in a query.

The syntax for the CREATE NICKNAME statement is:





data-type:



column-option:

Notes:

1 Optional for sorted files only.

nickname
A unique nickname for the table-structured file to be accessed. It must be distinct from all other nicknames, tables, and views in the schema in which it is being registered.

column-name
A unique name given to each field in the table-structured file. Follow each column name with its data type. Only columns of type CHAR, VARCHAR, SMALLINT, INTEGER, FLOAT, DOUBLE, REAL, and DECIMAL are supported.

SMALLINT
For a small integer.

INTEGER or INT
For a large integer.

FLOAT(*integer***)**
For a single or double precision floating-point number, depending on the value of *integer*. The value of *integer* must be in the range 1 through 53. The values 1 through 24 indicate single precision and the values 25 through 53 indicate double precision.

REAL For single precision floating-point.

DOUBLE or DOUBLE PRECISION
For double precision floating-point.

FLOAT
For double precision floating-point.

DECIMAL(*precision-integer, scale-integer***) or DEC(***precision-integer, scale-integer***)**
For a decimal number.

The first integer is the precision of the number; that is, the total number of digits. This value can range from 1 to 31.

The second integer is the scale of the number; that is, the number of digits to the right of the decimal point. This value can range from 0 to the precision of the number.

If precision and scale are not specified, the default values of 5,0 are used.

The words **NUMERIC** and **NUM** can be used as synonyms for **DECIMAL** and **DEC**.

CHARACTER(*integer***) or CHAR(***integer***) or CHARACTER or CHAR**
For a fixed-length character string of length *integer*, which can range from 1 to 254. If the length specification is omitted, a length of 1 character is assumed.

VARCHAR(*integer***)**
For a varying-length character string of maximum length *integer*, which can range from 1 to 32672.

NOT NULL
Prevents the column from containing null values.

server-name
Identifies the server you registered in "Step 3: Registering the server" on page 11. This server will be used to access the table-structured file. If the file is sorted, the server specified should be of type SORTED; otherwise specify a server of type UNSORTED.

'path' The fully qualified path to the table-structured file to be accessed, enclosed in single quotation marks. The data file must be a standard file or a symbolic link, rather then a pipe or another non-standard file

type. Data files must be readable by the DB2 instance owner. For more information on instance owners, see the *DB2 Administration Guide*.

'delimiter'
The delimiter used to separate columns of the table-structured file, enclosed in single quotation marks. If no column delimiter is defined, the column delimiter defaults to the comma. The column delimiter cannot exist as valid data for a column. For example, a column delimiter of a comma cannot be used if one of the columns contains data with embedded commas.

'key-column-name'
The name of the column in the file that forms the key on which the file is sorted, enclosed in single quotation marks. Use this option for sorted files only. It is case insensitive.

Only single-column keys are supported. The value must be the name of a column defined in the CREATE NICKNAME statement. The column must be sorted in ascending order. If the value is not specified for a sorted server, it defaults to the first column in the nicknamed file.

VALIDATE_DATA_FILE
For sorted files, this option specifies whether the wrapper verifies that the key column is sorted in ascending order. The only valid values for this option are 'Y' or 'N', enclosed in single quotation marks. The check is done once at registration time. If this option is not specified, then no validation takes place.

The following example shows a CREATE NICKNAME statement for the table-structured file DRUGDATA1.TXT described in "What are table-structured files?" on page 9:

```
CREATE NICKNAME DRUGDATA1(Dcode Integer, Drug CHAR(20), Manufacturer CHAR(20))
FOR SERVER biochem_lab OPTIONS(FILE_PATH '/usr/pat/DRUGDATA1.TXT',
COLUMN_DELIMITER ',', KEY_COLUMN 'Dcode', VALIDATE_DATA_FILE 'Y')
```

See the *DB2 SQL Reference* for more information about the CREATE NICKNAME statement. For more information about nicknames, see the *DB2 Administration Guide*.

Wrapper limitations and considerations

- Passthru sessions are not allowed when using the wrapper.
- Multi-column keys are not allowed.
- Sorted files must be in ascending order only. Sorting in descending order is not supported.

- The wrapper does not enforce the NOT NULL constraint, but DB2 does. If you create a nickname and attach a NOT NULL constraint on a column and then select a row containing a null value for the column, DB2 will issue a SQL0407N error stating that you can't assign a NULL value to a NOT NULL column.

  The exception to this rule is for sorted servers. The key column for nicknames defined for sorted servers cannot be NULL. If a NULL key column is found for a nickname using a sorted server, the SQL1822N error is issued, stating that the key column is missing.
- On DB2 Universal Database Enterprise-Extended Edition, any table-structured file for which a nickname has been created must be accessible with the same path name from each node. The file does not have to be on a DB2 Universal Database node as long as it can be accessed from any node with a common path.

File limitations and considerations

- Files are limited to one record per line.
- Each record must have an equal number of delimited columns.
- Each record must be terminated by the standard line-termination character(s) for the platform on which the wrapper is installed.
- The column delimiter must be consistent throughout the file.
- A null value is represented by two delimiters next to each other or a delimiter followed by a line terminator, if the NULL field is the last one on the line.
- The radix character is determined by the RADIXCHAR item of the LC_NUMERIC National Language Support category.
- Sorted data sources must be sorted in ascending order according to the collation sequence for the current locale, as defined by the settings in the LC_COLLATE National Language Support category.
- The database codepage must match the file's character set; otherwise, you could get unexpected results.
- Files containing multibyte characters are not supported.
- If a non-numeric field is too long for its column type, the excess data is truncated.
- If a decimal field in the file has more digits after the radix char than are allowed by the scale parameter of its column type, the excess data is truncated.
- The maximum line length is 32768.

File access control model

The database management system will access table-structured files with the authority of the DB2 instance owner. The wrapper can only access files that can be read by this user ID (or group ID). The authorization ID of the application (the ID that establishes the connection to the federated database) is not relevant.

Optimization tips and considerations

- The system can search sorted data files much more efficiently than non-sorted files.
- For sorted files, you can improve performance by specifying a value or range for the key column.
- Statistics for nicknames of table-structured files must be updated manually by updating the SYSTAT views. For more information on manually updating SYSTAT views, see the *DB2 Administration Guide*.

Messages

This section lists and describes messages you might encounter while working with the wrapper for table-structured files. For more information on messages, see the *DB2 Message Reference*.

Table 4. Messages issued by the wrapper for table-structured files

Error Code	Message	Explanation
SQL0405N	The numeric literal "<literal>" is not valid because its value is out of range.	A column in the data file, or a predicate value in an SQL statement, contains a value that is out of the possible range for that data type. Correct the data file or redefine the column to a more appropriate type.
SQL0408N	A value is not compatible with the data type of it's assignment target. Target name is "<column_name>".	A column in the data file contains characters that are invalid for that data type. Correct the data file or redefine the column to a more appropriate type.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason "Data source path is NULL".)	Contact IBM Software Support.

Table 4. Messages issued by the wrapper for table-structured files (continued)

Error Code	Message	Explanation
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″Key Column retrieval failure″.)	Contact IBM Software Support.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″STAT failed on data source. ERRNO = <error_number>″.)	Contact IBM Software Support.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″No column info found″.)	Contact IBM Software Support.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″Server parser failed, RC = <parser_return_code>″.)	Contact IBM Software Support.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″Unsupported operator″.)	Contact IBM Software Support.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″Cannot identify bind variable″.)	Contact IBM Software Support.

Table 4. Messages issued by the wrapper for table-structured files (continued)

Error Code	Message	Explanation
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″Unable to identify query components″.)	Contact IBM Software Support.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent statements can be processed. (Reason ″Could not access data when converting values″.)	Contact IBM Software Support.
SQL1816N	Wrapper ″<wrapper_name>″ cannot be used to access the ″type″ of data source (″<type>″ ″″) that you are trying to define to the federated database.	The server type was invalid. The only server types allowed are SORTED or UNSORTED. Change the SQL statement and rerun it.
SQL1822N	Unexpected error code ″ERRNO = <error_number>″ received from data source ″<server_name>″. Associated text and tokens are ″Unable to read file″.	Check the value of the error number. Make sure that the file can be read by the DB2 instance owner. Then rerun the SQL command.
SQL1822N	Unexpected error code ″Data Error″ received from data source ″<server_name>″. Associated text and tokens are ″Data source is a non-standard file″.	The data source file is a directory, socket, or FIFO. Only standard files can be accessed as data source. Change the FILE_PATH option to point to a valid file and reissue the SQL command.
SQL1822N	Unexpected error code ″ERRNO = <error_number>″ received from data source ″<server_name>″. Associated text and tokens are ″File open error″.	The wrapper was unable to open the file. Check the error number to determine why the error occurred. Correct the problem with the data source and reissue the SQL command.

Table 4. Messages issued by the wrapper for table-structured files (continued)

Error Code	Message	Explanation
SQL1822N	Unexpected error code ″Data Error″ received from data source ″<server_name>″. Associated text and tokens are ″Key column missing″.	A record retrieved from the data source was missing the key field. The key column must not be null. Correct the data, or register the file with an unsorted server.
SQL1822N	Unexpected error code ″Data Error″ received from data source ″<server_name>″. Associated text and tokens are ″File not sorted″.	The file was not sorted on the key column. Do one of the following: change the KEY_COLUMN option to point to the correct column; resort the data file; or register the nickname with an unsorted server.
SQL1822N	Unexpected error code ″Data Error″ received from data source ″<server_name>″. Associated text and tokens are ″Key exceeds definition size″.	The key column field read from the data source was larger than the DB2 column definition which could cause the wrapper search routines to function incorrectly. Correct the data or correct the nickname definition, and reregister the nickname.
SQL1822N	Unexpected error code ″Data Error″ received from data source ″<server_name>″. Associated text and tokens are ″Line in data file exceeds 32k″.	A line in the data file exceeded the maximum line length allowed by the wrapper. The line length cannot be greater than 32768. Shorten the length of the line in the data file.
SQL1823N	No data type mapping exists for data type ″<data_type>″ from server ″<server_name>″.	The nickname was defined with an unsupported data type. Redefine the nickname using only supported data types.
SQL1881N	″<option_name>″ is not a valid ″<component>″ option for ″<object_name>″.	The listed value is not a valid option for the listed object. Remove or change the invalid option then resubmit the SQL statement.
SQL1882N	The ″Nickname″ option ″COLUMN_DELIMITER″ cannot be set to ″<delimiter>″ for ″<nickname_name>″.	The column delimiter was more than one character long. Redefine the option with a single character. Then rerun the SQL statement command.

Table 4. Messages issued by the wrapper for table-structured files (continued)

Error Code	Message	Explanation
SQL1882N	The ″Nickname″ option ″KEY_COLUMN″ cannot be set to ″<column_name>″ for ″<nickname_name>″.	The column selected as the key column is not defined for this nickname. Correct the KEY_COLUMN option to be one of the sorted columns for this nickname, then reissue the SQL command.
SQL1882N	The ″Nickname″ option ″VALIDATE_DATA_FILE″ cannot be set to ″<option_value>″ for ″<nickname_name>″.	The option value was invalid. Valid values are ″Y″ or ″N″. Correct the option and register the nickname again.
SQL1883N	″<option_name>″ is a required ″<component>″ option for ″<object_name>″.	A required option for the wrapper was missing from the SQL statement. Add the required option and resubmit the SQL statement.
SQL30090N	Operation invalid for application execution environment. Reason code = ″21″.	You attempted a passthru session. The table-structured file wrapper does not support passthru sessions.

Chapter 4. Using Documentum as a data source

This chapter describes:

- The Documentum data source
- How to add a Documentum data source to a federated system
- How to run queries against a Documentum data source
- How to dual define repeating attributes
- How to create nicknames using the CreateNicknameFile utility
- Limitations and considerations
- The access control model used
- Messages you might encounter when working with the Documentum wrapper

What is Documentum?

Documentum is document management software that provides management of document content and attributes such as check-in, check-out, workflow, and version management. The Documentum product is a three-tier, client-server system built on top of a relational database.

A Docbase is a Documentum repository that stores document content, attributes, relationships, versions, renditions, formats, workflow, and security. Documentum Query Language (DQL), an extended SQL dialect, is used to query Documentum data. A Docbase is the equivalent of an Oracle instance or a DB2 database plus document content files. The metadata is stored in the underlying relational database management system (RDBMS), and the content is stored as binary large objects (BLOBs) in the database or as files stored within the file-system of the server system. For more information on Documentum, refer to the Documentum manuals.

The wrapper for Documentum allows you to add a Documentum data source to a DB2 federated system. By adding the Documentum data source to a federated system, you can use SQL statements to access and query objects and registered tables in a Documentum Docbase. You can then integrate this data with other data sources in your federated system without having to move the data out of the native data source. The Documentum wrapper uses a client library to interface with the Documentum server. The Documentum wrapper provides access to two versions of the Documentum server: EDMS 98 (also referred to as version 3) and 4i. Figure 3 on page 24 illustrates how the Documentum wrapper works.

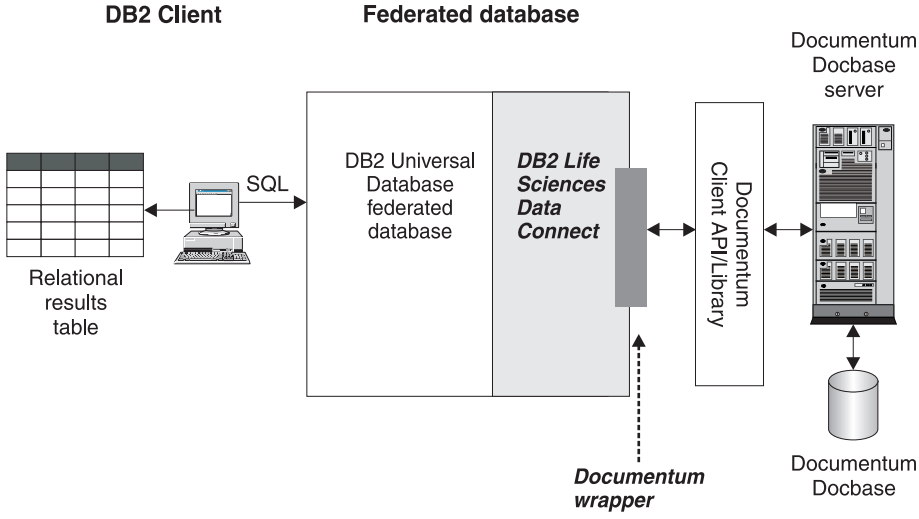


Figure 3. How the Documentum wrapper works

After the Documentum wrapper is registered, you can map Documentum Docbase objects and registered tables as relational tables. This is done by mapping Docbase attributes to column names in a DB2 relational table.

For example, Table 5 lists a subset of attributes for the Documentum Docbase default document type, dm_document, along with the associated data. You have determined that this attribute subset is important to you, and you would like to connect these attributes into your federated database system. You named this subset of data DrugAB_data.

Table 5. DrugAB_data

Title	Subject	Authors	Keywords
The effect of drug A on rabbits	Drug A	Curran, L.	rabbits, drug A
Toxicity results for drug A	Drug A	Abelite, P., McMurtrey, K.	toxicity, drug A
Drug B interactions	Drug B	DeNiro, R., Stone, S.	interactions, drug B
Chemical structure of drug B	Drug B	Boyslim, F.	structure, drug B

After you register the Documentum wrapper, the data can be queried using SQL statements.

The following query displays the titles and authors whose subject is Drug A. The result table is shown in Table 6 on page 25.

```
SELECT title, authors
FROM drugAB_data
WHERE subject = 'Drug A'
```

Table 6. Query results

Title	Authors
The effect of drug A on rabbits	Curran, L.
Toxicity results for drug A	Abelite, P., McMurtrey, K.

Adding Documentum to a federated system

To add the Documentum data source to a federated server:

1. Set the environment variables.
2. Link to the Documentum client libraries.
3. Recycle the DB2 instance.
4. Register the wrapper using the CREATE WRAPPER statement.
5. Optional: Set the DB2_DJ_COMM environment variable to improve query performance.
6. Register the server using the CREATE SERVER statement.
7. Give users access to the data source by using the CREATE USER MAPPING statement.
8. Register nicknames using the CREATE NICKNAME statement.
9. Create custom functions using the CREATE FUNCTION statement.

These steps are explained in detail in this section. The statements can be run from the DB2 Command Line Processor. Once registered, you can run queries against the data source.

Step 1: Setting environment variables

Access to Documentum Docbases are controlled through the Documentum client's dmcl.ini file. A DB2 instance must have its environment variables set to the Documentum client's dmcl.ini file in order to gain access to a Documentum Docbase.

To set the environment variables, edit the db2dj.ini file located in $HOME/sqllib/cfg/, and set one of the following environment variables:

```
DOCUMENT=<path to location of dmcl.ini>
DMCL_CONFIG=<path to location of dmcl.ini>/dmcl.ini
```

The default path to the location of Documentum's dmcl.ini file is /pkgs/documentum. If both lines are included, DMCL_CONFIG is used.

Note: Ensure that the name of a docbroker, to which all accessible Docbases for the DB2 instance report, is specified in the dmcl.ini file.

Step 2: Linking to Documentum client libraries

To enable access to Documentum data sources, the DB2 federated system must be link-edited to the client libraries. The link-edit process creates a wrapper library for each data source with which the federated server communicates. When you run the djxlinkDctm script you create the Documentum wrapper library.

To run the djxlinkDctm script:

1. Set the DM_HOME environment variable to point to the Documentum client library. For example:

   ```
   export DM_HOME=/pkgs/documentum/product/current
   ```

2. Type the following command:

   ```
   djxlinkDctm
   ```

Step 3: Recycling the DB2 instance

To ensure that the environment variables are set in the program, recycle the DB2 instance. When you recycle the instance, you refresh the DB2 instance to accept the changes that you made. Recycle the DB2 instance by issuing the following commands:

```
db2stop
db2start
```

Step 4: Registering the wrapper

To register the Documentum wrapper, submit the CREATE WRAPPER statement.

For example, to create a Documentum wrapper called Dctm_Wrapper from the default library file, liblsdctm.a, submit the following statement:

```
CREATE WRAPPER Dctm_Wrapper LIBRARY 'liblsdctm.a';
```

For more information on the CREATE WRAPPER statement, see the *DB2 SQL Reference*.

Step 5: Optional: Setting the DB2_DJ_COMM environment variable

To improve performance, set the DB2_DJ_COMM environment variable. This variable determines whether the federated server loads the wrapper upon initialization. Set the DB2_DJ_COMM environment variable to include the wrapper library that corresponds to the wrapper that you specified in "Step 4: Registering the wrapper". For example:

```
export DB2_DJ_COMM='liblsdctm.a'
```

Ensure that there are no spaces on either side of the equal sign (=).

For more information about the DB2_DJ_COMM environment variable, see the *DB2 Administration Guide*.

Step 6: Registering the server

Register the Documentum server to the federated system using the CREATE SERVER statement.

For example, suppose there is a server called Dctm_Server1 for the Dctm_Wrapper wrapper created in "Step 4: Registering the wrapper" on page 26. Suppose that server contains a Docbase that runs on AIX and uses Oracle to store data. To register the server, submit the following statement:

```
CREATE SERVER Dctm_Server1
TYPE DCTM
VERSION '3'
WRAPPER Dctm_Wrapper
OPTIONS(
NODE 'Dctm_Docbase',
OS_TYPE 'AIX',
RDBMS 'ORACLE');
```

Arguments

TYPE Specifies the type of the data source. For Documentum, the type is DCTM. This argument is required.

VERSION
Specifies the version of the data source. For EDMS98, the value is '3'. For 4i, the value is '4'. This argument is required.

WRAPPER
Specifies the name of the wrapper associated with this server. This argument is required.

Options

CONTENT_DIR
Specifies the name of the locally-accessible root directory for storing content files retrieved by the GET_FILE(), GET_FILE_DEL(), GET_RENDITION(), and GET_RENDITION_DEL() custom functions. It must be writable by all users who can use these functions. Its default value is /tmp. This option is optional. For more information on custom functions, see "Step 9: Registering custom functions" on page 34.

NODE
Specifies the actual name of the Documentum Docbase. This option is required.

OS_TYPE
Specifies the Docbase server's operating system. The only valid value is AIX. This option is required.

RDBMS_TYPE
Specifies the RDBMS used by the Docbase. The only valid value is ORACLE. This option is required.

TRANSACTIONS
Specifies the server transaction mode. The valid values are:

- NONE — no transactions are enabled.
- QUERY — transactions are enabled only for Dctm_Query methods.
- ALL — transactions are enabled for the Dctm_Query method. ALL has the same function as QUERY in this release.

The default is QUERY. This option is optional.

DEBUG_FILE
Specifies the fully-qualified name of a file to receive wrapper activity messages. It must be a file writable by the instance owner. The local name of the file should be <server_name>.log. The default is "" which means not to log wrapper activity messages. This option is optional.

DEBUG_LEVEL
Specifies the level of debug messages to be logged. This option is ignored if DEBUG_FILE is not specified. The valid values are:

- DEBUG_ALL
- DEBUG_INFO
- DEBUG_WARN
- DEBUG_ERROR - Error messages are logged to the wrapper activity file specified in the DEBUG_FILE option. Error messages are always logged to the DB2 error log regardless of the DEBUG_LEVEL option's setting.
- DEBUG_CRITICAL - Critical messages are logged to the wrapper activity file specified in the DEBUG_FILE option. Critical messages are always logged to the DB2 error log regardless of the DEBUG_LEVEL option's setting.

The default is DEBUG_WARN. This option is optional.

For more information on the CREATE SERVER statement, see the *DB2 SQL Reference*.

Step 7: Mapping users

You must map users to the previously defined servers using the CREATE USER MAPPING statement.

The following example maps user 'Chuck' to user 'Charles' on Dctm_Server1 created in "Step 6: Registering the server" on page 27.

```
CREATE USER MAPPING FOR Chuck SERVER Dctm_Server1
OPTIONS(REMOTE_AUTHID 'Charles', REMOTE_PASSWORD 'Charles_pw');
```

You can also define your own user mapping. In the following example, USER is a keyword meaning the current user, not a user named "USER".

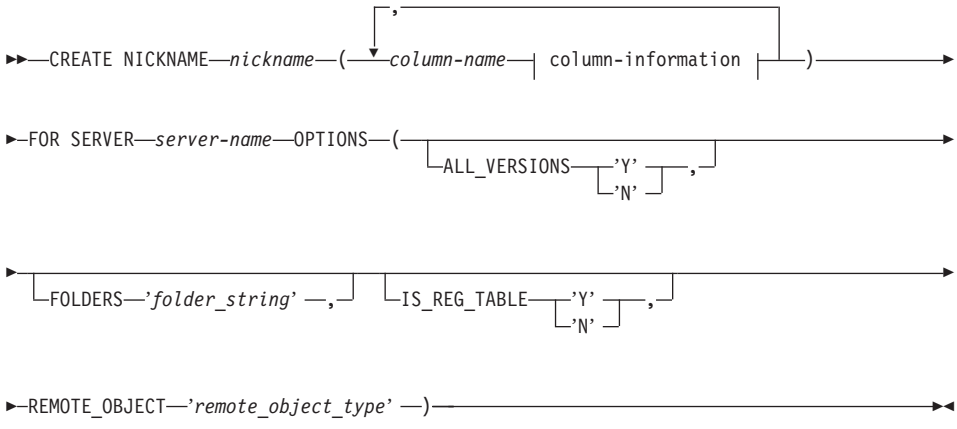
```
CREATE USER MAPPING FOR USER SERVER Dctm_Server1
OPTIONS(REMOTE_AUTHID 'Lisa', REMOTE_PASSWORD 'Lisa_pw)
```

For more information on the CREATE USER MAPPING statement, see the *DB2 SQL Reference*.

Step 8: Registering nicknames

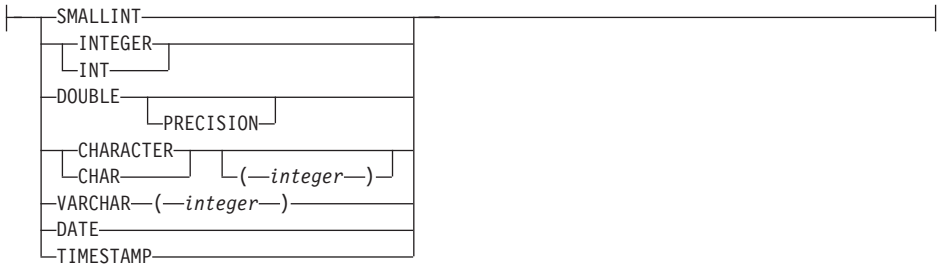
You must create a nickname for each Docbase for each object type or registered table of interest using the CREATE NICKNAME statement to map attribute names to DB2 relational column names.

The syntax for the CREATE NICKNAME statement for Documentum is:



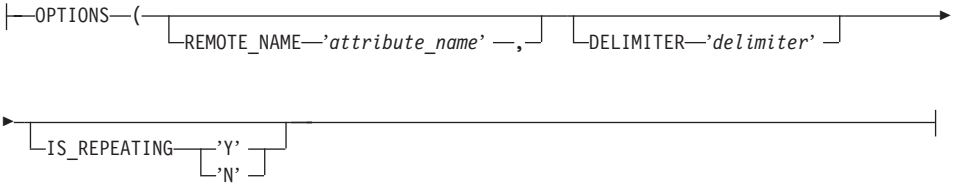
column-information:

data-type:



column-option:

wrapper-column-options:



For more information on the CREATE NICKNAME statement, see the *DB2 SQL Reference*.

Column options

NOT NULL

All single-valued columns except those defined as TIMESTAMP must be defined as NOT NULL. Repeating attributes must not be defined as NOT NULL in nicknames.

Wrapper column options

Wrapper column option values must be enclosed in single quotation marks.

DELIMITER

Specifies the delimiter string to be used when concatenating multiple values of a repeating attribute. The delimiter can be one or more characters. The default delimiter is a comma. This option is only valid for columns where the IS_REPEATING option is set to 'Y'. This option is optional.

IS_REPEATING
Indicates if the column is multi-valued. Valid values are 'Y' and 'N'. The default is 'N'.

REMOTE_NAME
Specifies the name of the corresponding Documentum attribute or column. This option maps remote attribute or column names to local DB2 column names. It defaults to the DB2 column name. This option is optional.

Nickname options

Nickname option values must be enclosed in single quotation marks.

ALL_VERSIONS
Specifies whether all object versions will be searched. The valid values are 'y', 'Y', 'n', and 'N'. The default value of 'N' means that only the current object versions are included in query processing. This option is invalid when IS_REG_TABLE = 'Y'. This option is optional.

FOLDERS
Specifies a string that contains one or more logically-combined and syntactically-correct Documentum FOLDER predicates. Specifying FOLDER predicates restricts the set of documents represented by this nickname to those in the designated folders.

When you specify this option, enclose the entire value of the FOLDERS option in single quotes and use double quotes in place of the single quotes within the string.

For example, if you want to insert:

```
FOLDER('/Tools',DESCEND) OR FOLDER('/Cars')
```

Specify the following FOLDERS option:

```
FOLDERS 'FOLDER("/Tools",DESCEND) OR FOLDER("/Cars")'
```

This option is invalid when IS_REG_TABLE = 'Y'. This option is optional.

IS_REG_TABLE
Specifies whether the object specified by the REMOTE_OBJECT option is a Documentum registered table. The valid values are 'y', 'Y', 'n', and 'N'. The default value is 'N'. This option is optional.

REMOTE_OBJECT
Specifies the name of the Documentum object type associated with the nickname. The name can be any Documentum object type or registered table. In the case of a registered table, it should be prefixed

by the table owner's name. If the registered table belongs to the Docbase owner, dm_dbo can be used for the owner name. This option is required.

CREATE NICKNAME example

The following CREATE NICKNAME statement defines the nickname std_doc. Std_doc is associated with a Documentum Docbase with an object type of dm_document. Table 7 maps the Documentum attributes and data types to DB2 relational column names and data types that are then used to construct the CREATE NICKNAME statement.

Table 7. Mapping of Documentum attributes to DB2 columns for the std_doc nickname

Documentum attribute name	Documentum data type	DB2 column name	DB2 data type	Repeats?	Description
object_name	string(255)	object_name	varchar	No	The user-defined name of the object.
r_object_id	ID	object_id	char(16)	No	The unique object identifier for this object, set at creation time.
r_object_type	string(32)	object_type	varchar	No	The object's type, set when the object is created.
title	string(255)	title	varchar	No	The user-defined title of the object.
subject	string(128)	subject	varchar	No	The user-defined subject of the object.
authors	string(32)	author	varchar	Yes	The user-defined list of the authors for the object.
keywords	string(32)	keyword	varchar	Yes	The list of user-defined keywords for the object.
r_creation_date	time	creation_date	timestamp	No	The date and time that the object was created.
r_modify_date	time	modified_date	timestamp	No	The date and time that the object was last modified.

Table 7. Mapping of Documentum attributes to DB2 columns for the std_doc nickname (continued)

Documentum attribute name	Documentum data type	DB2 column name	DB2 data type	Repeats?	Description
a_status	string(16)	status	varchar	No	Set by server when a router task is forwarded. The value is taken from the values assigned to attached_task_status in the router object.
a_content_type	string(32)	content_type	varchar	No	The file format of the object's content.
r_content_size	double	content_size	integer	No	The number of bytes in the content. For multi-page documents, this attribute records the size of the first content associated with the document.
owner_name	string(32)	owner_name	varchar	No	The name of the object's owner (the user who created the object).

Table 7 on page 32 translates into the following CREATE NICKNAME statement.

```
CREATE NICKNAME std_doc (
  object_name varchar(255) not null,
  object_id char(16) not null OPTIONS(REMOTE_NAME 'r_object_id'),
  object_type varchar(32) not null OPTIONS(REMOTE_NAME 'r_object_type'),
  title varchar(255) not null,
  subject varchar(128) not null,
  author varchar(32) OPTIONS(REMOTE_NAME 'authors', IS_REPEATING 'Y'),
  keyword varchar(32) OPTIONS(REMOTE_NAME 'keywords', IS_REPEATING 'Y'),
  creation_date timestamp OPTIONS(REMOTE_NAME 'r_creation_date'),
  modifed_date timestamp OPTIONS(REMOTE_NAME 'r_modify_date'),
  status varchar(16) not null OPTIONS(REMOTE_NAME 'a_status'),
  content_type varchar(32) not null OPTIONS(REMOTE_NAME 'a_content_type'),
```

```
  content_size integer not null OPTIONS(REMOTE_NAME 'r_content_size'),
  owner_name varchar(32))
FOR SERVER Dctm_Server2 OPTIONS (REMOTE_OBJECT 'dm_document', IS_REG_TABLE 'N')
```

After you submit the CREATE NICKNAME statement, you can use the nickname std_doc to query your federated system. You can also join the std_doc nickname with other nicknames and tables in the federated system.

You can use the CreateNicknameFile utility to automatically map Documentum types to DB2 types and to create an initial CREATE NICKNAME statement. For more information on the CreateNicknameFile utility, see "CreateNicknameFile utility" on page 44.

Step 9: Registering custom functions

You must use the CREATE FUNCTION statement to register several custom functions. You can use these functions to access some of the unique capabilities of Documentum, such as full-text searching and retrieving document content within queries.

Custom functions for predicates are listed in Table 8. Custom functions that can be specified only in a SELECT clause are listed in Table 9 on page 40. Custom functions for the SELECT statement when the statement contains a search clause are listed in Table 10 on page 42.

DB2 does not support a BOOLEAN data type. Therefore, to create valid SQL statements, the value of each custom function must be explicitly tested. The wrapper implementation only supports the semantics for "<function>() = 1" regardless of the test comparison operator specified.

Note: References to the TOPIC function are to Documentum function provided as part of its third-party full-text indexing system from Verity, Inc.

Table 8. Custom functions for predicates

Function name	Description
ANY_EQ(arg1, arg2)	Tests a repeating attribute for any value equal to the specified value. Takes two required arguments: **arg1** Specifies the name of a column that represents a repeating attribute. **arg2** Specifies the value to be compared. For example: `... WHERE DCTM.ANY_EQ(authors,'Dave Winters')=1`

Table 8. Custom functions for predicates (continued)

Function name	Description
ANY_NE(arg1, arg2)	Tests a repeating attribute for any value not equal to the specified value. Takes two required arguments: **arg1** Specifies the name of a column that represents a repeating attribute. **arg2** Specifies the value to be compared. For example: `... WHERE DCTM.ANY_NE(authors,'Dave Winters')=1`
ANY_LT(arg1, arg2)	Tests a repeating attribute for any value less than the specified value. Takes two required arguments: **arg1** Specifies the name of a column that represents a repeating attribute. **arg2** Specifies the value to be compared. For example: `... WHERE DCTM.ANY_LT(num_approvers,4)=1`
ANY_GT(arg1, arg2)	Tests a repeating attribute for any value greater than the specified value. Takes two required arguments: **arg1** Specifies the name of a column that represents a repeating attribute. **arg2** Specifies the value to be compared. For example: `... WHERE DCTM.ANY_GT(num_approvers,3)=1`
ANY_LE(arg1, arg2)	Tests a repeating attribute for any value less than or equal to the specified value. Takes two required arguments: **arg1** Specifies the name of a column that represents a repeating attribute. **arg2** Specifies the value to be compared. For example: `... WHERE DCTM.ANY_LE(num_approvers,2)=1`

Table 8. Custom functions for predicates (continued)

Function name	Description
ANY_GE(arg1, arg2)	Tests a repeating attribute for any value greater than or equal to the specified value. Takes two required arguments: **arg1** Specifies the name of a column that represents a repeating attribute. **arg2** Specifies the value to be compared. For example: `... WHERE DCTM.ANY_GE(num_approvers,1)=1`
ANY_IN(arg1, arg2 – arg11)	Tests a repeating attribute for any of ten values in a specified list of values. Takes 3–11 arguments of the same data type: **arg1** Specifies the name of a column that represents a repeating attribute. **arg2–arg11** Specifies a comma-separated list of values to be compared. For example: `... WHERE DCTM.ANY_IN(authors,'Crick','Watson')=1`

ANY_LIKE(arg1, arg2)	Tests a repeating attribute for any value like the specified value. Takes two required arguments:
	arg1 Specifies the name of a column that represents a repeating attribute.
	arg2 Specifies the pattern being compared with sub-strings in single quotes.
	For example:
	`... WHERE DCTM.ANY_LIKE(authors,'Dave Win%')=1`
	`OR DCTM.ANY_LIKE(keywords,'%\_%')=1`
	Note: The escape clause is not supported in ANY_LIKE() predicates.

Table 8. Custom functions for predicates (continued)

Function name	Description
ANY_NOT_LIKE(arg1, arg2)	Tests a repeating attribute for any value not like the specified value. Takes two required arguments: **arg1** Specifies the name of a column that represents a repeating attribute. **arg2** Specifies the pattern being compared with sub-strings in single quotes. For example: `... WHERE DCTM.ANY_NOT_LIKE(authors,'Dave Win%')=1 OR DCTM.ANY_NOT_LIKE(keywords,'%\_%')=1` **Note:** The escape clause is not supported in ANY_NOT_LIKE() predicates.
ANY_NULL(arg)	Tests a repeating attribute for IS NULL. Takes one required argument that is the name of the repeating attribute or single-valued DATE or TIMESTAMP attribute. For example: `... WHERE DCTM.ANY_NULL(authors)=1`
ANY_NOT_NULL(arg)	Tests a repeating attribute for IS NOT NULL. Takes one required argument that is the name of the repeating attribute. For example: `... WHERE DCTM.ANY_NOT_NULL(authors)=1`
ANY_SAME_INDEX(arg1 – arg10)	Tests repeating attributes for values at the same index of each attribute. Takes two to ten of the other ANY_xx() functions. The following example checks whether a document has at least one author named Ken who is not affiliated with UCD. `... WHERE DCTM.ANY_SAME_INDEX( ANY_EQ(author_name,'Ken'), DCTM.ANY_NE(author_affiliation,'UCD'))`

Table 8. Custom functions for predicates (continued)

Function name	Description

CABINET(arg) and CABINET_TREE(arg)	Takes one required argument that is the fully-qualified name of a Docbase cabinet.
	
	For example:
	`... WHERE DCTM.CABINET('/Tools')=1`
	`... WHERE DCTM.CABINET_TREE('/MyDocs')=1`
	
	Use multiple instances of CABINET and CABINET_TREE to specify multiple cabinets.
	
	For example:
	`... WHERE DCTM.CABINET('/Tools')=1`
	`OR DCTM.CABINET_TREE('/Parts')=1`

FOLDER(arg) and FOLDER_TREE(arg)	Takes one required argument that is the fully-qualified name of a Docbase folder or cabinet.
	
	For example:
	`... DCTM.FOLDER('/Tools/Drills')=1`
	`... DCTM.FOLDER_TREE('/MyDocs/WhitePapers')=1`
	
	Use multiple instances of FOLDER and FOLDER_TREE to specify multiple folders.
	
	For example:
	`... DCTM.FOLDER('/Tools/Drills')=1`
	`OR DCTM.FOLDER_TREE('/Animals/Horses')=1`

USER(1)	Compares a value to the Documentum author ID of the current user. Takes a dummy argument that must be 1. For example: `... WHERE approver = DCTM.USER(1)` **Note:** To make the Documentum author ID correspond to the DB2 author ID, use the CREATE USER MAPPING statement. For more information on user mapping, see "Step 7: Mapping users" on page 28.

Table 8. Custom functions for predicates (continued)

Function name	Description

SEARCH_WORDS(arg)	Takes one required string argument that is a list of individual words enclosed in single quotes, separated by AND, OR, or NOT, and using parentheses to control precedence. Words cannot contain white space and must be enclosed in single quotes.
	
	For example:
	`... DCTM.SEARCH_WORDS('''yeast''`
	`AND (''bread'' OR ''cake'')`
	`AND NOT ''wedding''' )=1`

SEARCH_TOPIC(arg)	Takes one required string argument which is a Verity TOPIC query statement that is to be passed to Documentum and Verity verbatim. For example: `... WHERE DCTM.SEARCH_TOPIC('"quick"')=1`

Table 9 lists custom functions for SELECT clauses.

Table 9. Custom functions for SELECT clauses

Function name	Description
GET_FILE(1)	Retrieves the content file for the current row in addition to the column values. Takes a dummy argument that must be 1. r_object_id and object_name must also be specified in the SELECT list because the content file for that object ID will be retrieved for the row and given its object name in local store. The extension for the content file is its Documentum format name. If a file of the same name exists, it will be overwritten. GET_FILE(1) attempts to get the object's base format. Its value in the row is the object's a_content_type. Its value is the string ″no_content″ if the object has no content file. For example: `SELECT object_name, r_object_id, DCTM.GET_FILE(1) FROM ...` The content file is placed in the server directory that is specified by the Server's CONTENT_DIR option. It is also placed in a subdirectory named with the user's DB2 local name. The subdirectory will be created if it doesn't exist. It's extension will be its DOS extension defined in the Docbase for the document's format type. For example, ″.doc″, for MS Word documents. Returns the string ″no_content″ or the fully-qualified name of the file.
GET_FILE_DEL(1)	This function is the same as GET_FILE(1) except GET_FILE_DEL(1) first deletes the file retrieved for the previous row, if any, in that query. Takes a dummy argument that must be 1. Returns the string ″no_content″ or the fully-qualified name of the file.

Table 9. Custom functions for SELECT clauses (continued)

Function name	Description

GET_RENDITION(arg)	Retrieves the content file of that rendition, a copy of the original document in a different format, for the current row in addition to the column values. Takes one argument which is the name of the desired rendition.
	
	r_object_id and object_name must also be specified in the SELECT list because the content file for that object ID will be retrieved for the row and given its object name in local store. The extension for the content file is its Documentum format name. If a file of the same name exists, it will be overwritten.
	
	GET_RENDITION() attempts to get the named rendition of the object. Its value in the row is the object's a_content_type, except that its value is the string ″no_content″ if the object has no content file, or the string ″not_found″ if the rendition does not exist.
	
	For example:
	`SELECT object_name, r_object_id,`
	`DCTM.GET_RENDITION('pdf')`
	`FROM ...`
	
	The content file is placed in the server directory that is specified by the Server's CONTENT_DIR option. It is also placed in a subdirectory named with the user's DB2 local name. The subdirectory will be created if it doesn't exist.
	
	It's extension will be its DOS extension defined in the Docbase for the document's format type. For example, ″.doc″, for MS Word documents.
	
	Returns the string ″no_content″ or the fully-qualified name of the file.

GET_RENDITION_DEL(arg)	This function is the same as GET_RENDITION() except GET_RENDITION_DEL() first deletes the file retrieved for the previous row, if any, in that query. Returns the string ″no_content″ or the fully-qualified name of the file.

Table 10 on page 42 lists custom functions for SELECT clauses in queries that contain search clauses.

Table 10. Custom functions for SELECT clauses in queries that contain search clauses

Function name	Description

HITS(1)	Returns an integer number that represents the number of places in the document in which the search criteria was matched.
	
	For example:
	`SELECT r_object_id, object_name, DCTM.HITS(1)`
	`FROM std_doc DCTM.SEARCH WORDS('workflow' OR 'flowchart')`
	
	For each document returned, the number of occurrences of the words ″workflow″ and ″flowchart″ within the document's content are summed and returned as the HITS value.
	
	The HITS function is appropriate when the documents have only one content file. This is the typical case. This keyword can be used in a WHERE clause qualification for a SELECT statement. However, it must also be specified in the SELECT clause.

SCORE(1)	Returns a document's relevance ranking.
	
	Use this custom function in conjunction with the Documentum's ACCRUE concept operator. Both return a number that indicates how many of the specified words were found in each returned document.
	
	For example:
	`SELECT object_name, DCTM.SCORE(1) FROM std_doc`
	`DCTM.SEARCH_TOPIC('<ACCRUE>("document","management","workflow")')`
	`WHERE DCTM.SCORE(1) >=75`
	
	The statement returns all documents that have either two or three of the specified words in their content. If a document has only one of the words, it is assigned a score of 50 and therefore fails the WHERE clause criteria and is not returned. If two of the three words are found, a document is assigned a score of 75. If all three words are found, the document's score is 88.
	
	The SCORE(1) function is used for documents that have one content file. This is the typical case.
	
	SCORE(1) can be in a SELECT clause only if the WHERE contains a SEARCH_WORDS() or SEARCH_TOPIC() function. In a WHERE clause, it is used in conjunction with the ACCRUE concept operator.
	
	For information on the ACCRUE concept operator, see the Documentum documentation.

Custom function string argument rules

All arguments passed as strings must adhere to the following rules:

- Each string is enclosed in single quotes.
- Single quotes within strings are expressed by two single quotes.

Specifying custom functions with the CREATE FUNCTION statement

All custom functions must be registered with the schema name DCTM. The fully-qualified name of each function is DCTM.<function_name>.

The following example registers the ANY_EQ custom function.

```
CREATE FUNCTION DCTM.ANY_EQ (CHAR(), CHAR()) RETURNS INTEGER AS TEMPLATE
```

You must register each custom function one time for each DB2 database that has the Documentum wrapper installed.

To assist you in registering custom functions, the sample file, create_fuction_mappings.ddl, is provided in the sqllib/samples/lifesci directory. This file contains definitions for each custom function. You can run this ddl file to register the custom functions for each DB2 database that has the Documentum wrapper installed.

Using custom functions in queries

The following examples illustrate the use of the custom functions in queries.

To display the object name and author from the std_doc nickname for documents that have one or more authors named 'Dave Winters':

```
SELECT object_name,authors FROM std_doc
WHERE DCTM.ANY_EQ(authors,'Dave Winters')=1
```

To display the object name and author from the std_doc nickname for documents that have one or more authors named 'Dave Winters' or 'Jon Doe':

```
SELECT object_name,authors FROM std_doc
WHERE DCTM.ANY_IN(authors,'Dave Winters','John Doe')=1
```

To display the object name and r_object_id, and to retrieve the content file, from the std_doc nickname for documents containing strings like 'Dave Win%' in the authors column:

```
SELECT object_name,r_object_id,DCTM.GET_FILE(1) FROM std_doc
WHERE DCTM.ANY_LIKE(authors,'Dave Win%')=1
```

To display the object name and r_object_id, and to retrieve the content file of the 'pdf' rendition, from the std_doc nickname for documents containing strings like 'IBM DiscoveryLink%' in the title column:

```
SELECT object_name,r_object_id,DCTM.GET_RENDITION('pdf') FROM std_doc
WHERE title like 'IBM DiscoveryLink%'
```

For more information on the CREATE FUNCTION statement, see the *DB2 SQL Reference*.

Running queries

After you register the wrapper, you can run SQL queries against the Documentum data source. This section provides several example queries.

The following query displays all of the Docbase documents for documents named 'Test Document':

```
SELECT object_name
FROM std_doc
WHERE object_name='Test Document';
```

The following query uses the custom function ANY_EQ to display all the documents where one of the authors is 'Joe Doe'.

```
SELECT *
FROM std_doc
WHERE DCTM.ANY_EQ(author,'Joe Doe')=1
```

The following query uses the FOLDER_TREE function and the SEARCH_WORDS function to find all documents in the Approved cabinet that contain the text "protein".

```
SELECT object_name
FROM std_doc
WHERE DCTM.FOLDER_TREE('/Approved')=1
      AND DCTM.SEARCH_WORDS('protein')=1
```

The following query uses the GET_FILE, FOLDER_TREE and ANY_IN custom functions to retrieve the name of the files, on the DB2 server, into which the content has been placed for all documents in the Approved cabinet that have any of the authors listed.

```
SELECT object_name, object_id, DCTM.GET_FILE(1)
FROM std_doc
WHERE DCTM.FOLDER_TREE('/Approved')=1
      AND DCTM.ANY_IN(author, 'Mary Black', 'Joe Carson', 'Peter Miller')=1
```

CreateNicknameFile utility

You can use a Docbasic utility named CreateNicknameFile, available for free download, to create an ASCII file that contains a complete definition of any Docbase object or registered table. You can edit the output file to:

- Define custom local names for columns and attributes. The local and remote names are initially the names as they are known in the Docbase.
- Delete unwanted columns and attributes. The only predefined Documentum document type (dm_document) has 59 attributes in EDMS98

and 76 attributes in 4i. Most of these contain metadata for low-level document management and application development. Deleting the attributes that are not of interest can make SELECT * SQL statements more useful without impacting performance.

- Add a value for the FOLDERS option to restrict searches against this nickname to particular Documentum folders.
- Change DATE mappings to TIMESTAMP if that is desired. The utility generates a mapping from DQL DATE to DB2 DATE because that seems the most useful.
- Change CHAR mappings to VARCHAR or vice-versa depending on application insight.

You must install the utility in a Docbase and be run it from a Documentum Windows graphical user interface. The files that the utility generates are specific to the Docbase in which it is installed.

Installing the CreateNicknameFile utility

To install the utility:

1. Download the CreateNicknameFile utility from the download section of the DB2 Life Sciences Data Connect product website at: http://www.ibm.com/software/data/db2/lifesciencesdataconnect/
2. Use the EDMS98 Workspace graphical user interface or the 4i Desktop Client to import the utility, named CreateNicknameFile.txt. You can import the utility as a procedure type into any Docbase cabinet or folder, and you can give it any name you want.
3. Check the **Can be run by user** box on the properties dialog for the newly imported CreateNicknameFile.txt object.

Configuring the CreateNicknameFile utility

To configure the utility after you install it:

1. Double-click on the utility's icon to run it.
2. Type the Documentum Document/object-type name. The default is dm_document.

   Note: Specify dm_registered as the name if you need to create a nickname file for a registered table. If you specify dm_registered, you will also be prompted for the fully-qualified table name in <owner>.<table_name> format. You can use dm_dbo for the owner name if the table is owned by the Docbase owner (the typical case).

   The utility assumes a naming convention for the names of nicknames for registered tables. The convention is to prefix the table

name with ″rt_″ to indicate ″registered table″. You can change the nickname proposed by the utility if you don't want to use this convention.

3. Type the server name associated with the nickname you are creating.
4. Type the name of the nickname.

   The names of nickname should be self-explanatory and must be unique within the DB2 instance. The utility assumes a naming convention of <server_name>.<object_type> because the same <object_type> might need to be defined to multiple servers. You can change the nickname proposed by the utility if you don't want to follow this convention.
5. Type the name of the output file.

   The default is C:\Temp\nickname.txt. The directory to receive the output file must already exist and be writeable by the user running the utility.

After you answer the prompts, the nickname file is created and opens in a text editor.

Mapping the DM_ID object type in Documentum registered tables

The column definitions created by the utility are compliant with the requirements of the Documentum wrapper, including the correct mapping of each data type to the corresponding DB2 data type. The only exception is that Documentum does not support the DM_ID data type in registered tables.

The utility assumes that a column in a registered table is used to contain an object ID if it is defined as a string, is 16 characters long, and has a name ending with ″_id″. In the case of the DM_ID data type, the utility maps the column to the DB2 CHAR(16) data type. In all other cases, all string/varchar columns are mapped to the DB2 VARCHAR data type. If the utility maps the column incorrectly, change the DB2 data type before using the file to define the nickname to DB2.

Dual defining repeating attributes

To maximize the query capabilities of the wrapper, each attribute must be defined as its true equivalent DB2 data type. That is, Documentum integers must be defined as DB2 integers and so forth. However, these definitions prevent the return of multiple values for non-VARCHAR repeating attributes. For such columns, only the value at index[0] is returned.

This restriction exists because, whenever possible, the wrapper returns only one row of results per Docbase object. This restriction is an issue only when repeating attributes are selected. However, you can define a second column for the same remote repeating attribute but with a data type of VARCHAR.

This column name would be used in the SELECT list to return all values as a delimiter-separated list of all its values. (Each column's DELIMITER option specifies the delimiter to be used.)

You should standardize the local names of the multi-value columns. You can standardize the local names of each multi-value column by adding a prefix of ″m_″ to the local name of the column that is defined as its true data type.

For example, suppose you have a nickname column of a Documentum repeating attribute called approval_dates defined with the data type TIMESTAMP. You can create a second nickname column called m_approval_dates and define it as a VARCHAR data type. You can then use m_approval_dates in a SELECT list to return all approval dates in a delimiter-separated list.

You do not need to use dual definitions for repeating attributes whose true data type is VARCHAR.

Limitations and considerations

This section contains a list of limitations and considerations associated with the use of the Documentum wrapper.

- Limitations concerning returning repeating attributes values: Only the value at index[0] is returned for non-VARCHAR repeating attributes. To overcome this limitation, you can create a dual definition for the repeating attribute column. For more information on creating dual definitions for repeating attributes, see "Dual defining repeating attributes" on page 46.

  Also, multiple values of repeating attributes defined as VARCHAR are returned as one delimiter—separated string. The delimiter depends on the setting of the DELIMITER nickname option described in "Wrapper column options" on page 30.
- The Passthru capability is not supported.
- For each connection to a DB2 database made by a DB2 application, the Documentum wrapper can support a maximum of 10 simultaneous Documentum sessions, and each such session can simultaneously manage up to 10 Documentum queries. A single DB2 application can have several queries in progrss simultaneously; the lifetime of a query begins when it is submitted to DB2 and ends when the corresponding cursor over the result set is closed. At any given time, across the entire set of queries in progress at that time, the following restrictions must hold:
  - All the nicknames referenced by all the queries must reside at no more than 10 different Documentum servers.
  - No more than 10 nicknames from one Documentum server may be referenced.

Nicknames mentioned in more than one query, or referenced multiple times in a single query, must be counted once for each time they appear.

- The Documentum wrapper uses Version 3.1.7a for AIX of the client library. If you are using Documentum 4i , you will need to acquire the older version of the client library from Documentum (if it is not already installed).
- Because DB2 does not support the Boolean type, most of the custom functions (except for USER, HITS & SCORE) used in the WHERE clause must do a check for ″=1″ because these functions are defined to return an integer.

  For example,

  ```
  "... WHERE DCTM.ANY_EQ(authors,'Dave Winters')=1"
  ```

- Due to a limitation of DB2, custom functions are not defined without arguments. Instead, these functions are defined with an integer argument that is not used. These functions are USER, GET_FILE, GET_FILE_DEL, HITS, and SCORE.
- All servers running against the same instance of DB2 must share the same Documentum dmcl.ini configuration parameters.
- The maximum number of values in an ANY_IN custom function for repeating attributes is 10 for a single statement. However, multiple statements can be OR'd.
- For the ANY_SAME_INDEX custom function the maximum number of tests for values at the same index of repeating attributes is 10. The tests must be AND tests that are evaluated left to right.
- The wrapper has no capabilities that are specific to a particular code page.

Access control

Queries are subject to the user's permissions in the Docbase. Only those documents to which the user has at least read access are included in query results.

Messages

This section lists and describes messages you might encounter while working with the wrapper for Documentum. For more information on messages, see the *DB2 Message Reference*.

Table 11. Messages issued by the wrapper for Documentum

Error Code	Message	Explanation
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″dmAPI exec failed: [DM_QUERY_E_BAD_QUAL] error: ″The attribute qualifier, A0, for attribute <column_name>, is not a valid qualifier.″″.)	An incorrect Documentum type or registered table was entered for the REMOTE_OBJECT nickname option. Change the nickname to use the correct Documentum object type or registered table.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″Invalid null column specified″.)	Internal programming error. Contact IBM Software Support.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″Nickname specification is empty″.)	Internal programming error. Contact IBM Software Support.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″The Output object is empty or incomplete″.)	Internal programming error. Contact IBM Software Support.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″Unexpected number of columns requested″.)	Internal programming error. Contact IBM Software Support.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″No column information found″.)	Internal programming error. Contact IBM Software Support.

Table 11. Messages issued by the wrapper for Documentum (continued)

Error Code	Message	Explanation
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″Unsupported column type requested″.)	Internal programming error. Contact IBM Software Support.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″Incorrect Column definition″.)	Internal programming error. Contact IBM Software Support.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″Inconsistent type; DB2 request != nickname type″.)	Internal programming error. Contact IBM Software Support.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″Output parameter is not NULL″.)	Internal programming error. Contact IBM Software Support.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″Query output variable is not NULL″.)	Internal programming error. Contact IBM Software Support.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″Invalid timestamp length″.)	Internal programming error. Contact IBM Software Support.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″Inconsistent number of columns″.)	Internal programming error. Contact IBM Software Support.

Table 11. Messages issued by the wrapper for Documentum (continued)

Error Code	Message	Explanation
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″Error in recursive function build_predicate_string″.)	Internal programming error. Contact IBM Software Support.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″Error in function build_function_string″.)	Internal programming error. Contact IBM Software Support.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″Could not access data when converting values″.)	Internal programming error. Contact IBM Software Support.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″Invalid RDBMS_TYPE, should be DB2, INFORMIX, ORACLE, SQLSERVER or SYBASE″.)	An invalid value was specified for the RDBMS_TYPE server option. Specify a valid value.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″Invalid TRANSACTIONS, should be NONE, QUERY, PASSTHRU or ALL″.)	An invalid value was specified for the TRANSACTIONS server option. Specify a valid value.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″Failed to initialize the DMCL client″.)	The Documentum client cannot initialize. Contact your system administrator.

Table 11. Messages issued by the wrapper for Documentum (continued)

Error Code	Message	Explanation
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″Get_User returned NULL″.)	Internal programming error. Contact IBM Software Support.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″Get_Local_User returned NULL″.)	Internal programming error. Contact IBM Software Support.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″Begin Transaction failed″.)	Documentum reported that begintrans failed. Contact your system administrator.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″Input parameter was not NULL″.)	Internal programming error. Contact IBM Software Support.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″Invalid column number requested″.)	Internal programming error. Contact IBM Software Support.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″Invalid value for IS_REG_TABLE, must be 'Y' or 'N'″.)	An invalid value for IS_REG_TABLE nickname option was specified in the CREATE NICKNAME statement. Specify a valid value.
SQL0901N	The SQL statement failed because of a non-severe system error. Subsequent SQL statements can be processed. (Reason ″Invalid value for ALL_VERSIONS, must be 'Y' or 'N'″.)	An invalid value for ALL_VERSIONS nickname option was specified in the CREATE NICKNAME statement. Specify a valid value.

Table 11. Messages issued by the wrapper for Documentum (continued)

Error Code	Message	Explanation
SQL30090N	Operation invalid for application execution environment. Reason code = ″Invalid column name, IS_REG_TABLE, or IS_REPEATING specified in nickname″	Check the nickname statement for the correct specification of the IS_REG_TABLE, IS_REPEATING, REMOTE_NAME options, and column names.
SQL30090N	Operation invalid for application execution environment. Reason code = ″Versions are not supported on registered tables.″	The VERSIONS option was specified for a registered table nickname. Remove the VERSIONS option from the Nickname definition.
SQL30090N	Operation invalid for application execution environment. Reason code = ″An invalid function has been included in the select list.″	Only GET_FILE, GET_FILE_DELETE, GET_RENDITION, GET_RENDITION_DEL, HITS, and SCORE are permitted in the SELECT list. Remove the invalid function from the SELECT list.
SQL30090N	Operation invalid for application execution environment. Reason code = ″Dctm functions must be like DCTM.function(..)=1″	The user did not use =1 as the RHS of the predicate for the Dctm function. Correct the syntax and run the query again.
SQL30090N	Operation invalid for application execution environment. Reason code = ″Invalid constant in select clause.″	Selecting a constant from a table is not allowed. Remove the constant from the SELECT list and try again.
SQL30090N	Operation invalid for application execution environment. Reason code = ″db2dj.ini missing DOCUMENTUM or DMCL_CONFIG env var″	The required environment variables are not set. Set them in the db2dj.ini file.
SQL30090N	Operation invalid for application execution environment. Reason code = ″Failed to open log file for debugging″	The log file used for troubleshooting is not accessible. Contact your system administrator.
SQL30090N	Operation invalid for application execution environment. Reason code = ″Invalid debug level found″	An invalid debug level was specified as a server option. Specify a valid debug level.

Table 11. Messages issued by the wrapper for Documentum (continued)

Error Code	Message	Explanation
SQL30090N	Operation invalid for application execution environment. Reason code = ″Invalid OS_TYPE, should be AIX,HPUX,SOLARIS or WINDOWS″	An invalid value was specified for the OS_TYPE server option. Specify a valid value.
SQL30090N	Operation invalid for application execution environment. Reason code = ″FOLDERS clause is not valid for registered tables″	The IS_REG_TABLE option is set to ″Y″ and the FOLDERS option is set. FOLDERS is not used for registered table. Update the CREATE NICKNAME statement options.
SQL30090N	Operation invalid for application execution environment. Reason code = ″Only one search condition may be specified″	Only one custom search function may be specified per query.
SQL30090N	Operation invalid for application execution environment. Reason code = ″Failed to create content directory″	Make sure the destination directory is writable by the DB2 agent.
SQL30090N	Operation invalid for application execution environment. Reason code = ″Failed to change permissions on content file″	Make sure the target content directory is writable by the db2 agent.
SQL30090N	Operation invalid for application execution environment. Reason code = ″DELIMITER is not valid unless IS_REPEATING = 'Y'″	Check the nickname values of IS_REPEATING and DELIMITER.

Chapter 5. Using Excel as a data source

This chapter describes:

- The Excel spreadsheet data source
- How to add an Excel data source to a federated system
- How to run queries against an Excel data source
- A sample user scenario
- Limitations and considerations
- The file access control model that is used
- Messages you might encounter when working with the Excel wrapper

What is Excel?

An Excel spreadsheet or workbook is a file created using the Microsoft (MS) Excel application and has a file extension of xls. DB2 Life Sciences Data Connect supports spreadsheets from Excel 97, Excel 98, and Excel 2000. Figure 4 illustrates how the Excel wrapper connects your spreadsheets to your federated system.

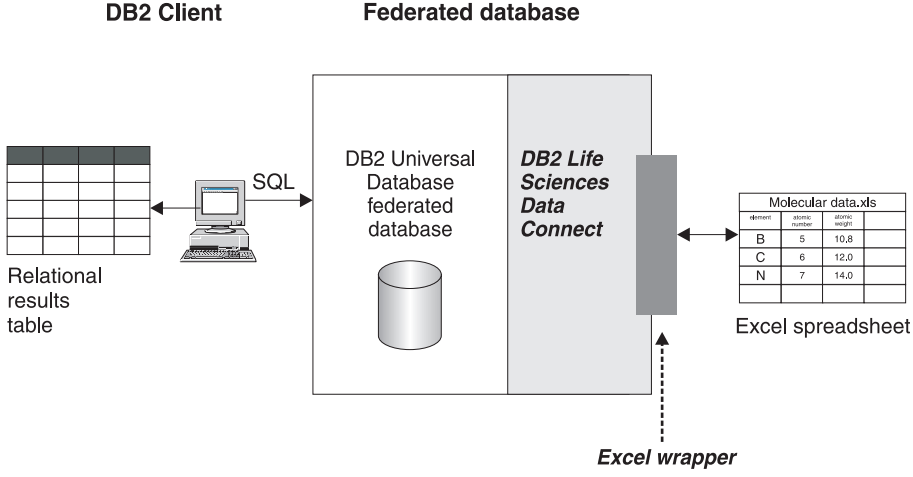


Figure 4. How the Excel wrapper works

The Excel wrapper uses the CREATE NICKNAME statement to map the columns in your Excel spreadsheet to columns in your DB2 federated system. Table 12 on page 56 shows sample spreadsheet data that is stored in a file called Compound_Master.xls.

Table 12. Sample spreadsheet for Compound_Master.xls

	A	B	C	D
1	compound_A	1.23	367	tested
2	compound_G		210	
3	compound_F	0.000425536	174	tested
4	compound_Y	1.00256		tested
5	compound_Q		1024	
6	compound_B	33.5362		
7	compound_S	0.96723	67	tested
8				
9	compound_O	1.2		tested

This information is usually not available to you through standard SQL commands. When the Excel wrapper is installed and registered, you can access this information as if it were a standard relational data source. For example, if you wanted to know all the compound data where the molecular count is greater than 100, you would run the following SQL query:

```
SELECT * FROM compound_master WHERE mol_count > 100
```

The results of the query are shown in Table 13.

Table 13. Query results

COMPOUND_NAME	WEIGHT	MOL_COUNT	WAS_TESTED
compound_A	1.23	367	tested
compound_G		210	
compound_F	0.000425536	174	tested
compound_Q		1024	

Prerequisites

The prerequisites for utilizing the the Excel data source wrappers are:

- The Excel application must be installed on the server where DB2 Life Sciences Data Connect is installed before an Excel wrapper can be utilized.
- If Excel 97 or 98 is installed, then the Excel_9x wrapper should be registered. If Excel 2000 is installed, then the Excel_2000 wrapper should be registered.

Adding Excel to a federated system

To add the Excel data source to a federated system:

1. Register the wrapper using the CREATE WRAPPER statement.
2. Register the server using the CREATE SERVER statement.
3. Register nicknames using the CREATE NICKNAME statement for each Excel spreadsheet you want to access.

These steps are explained in detail in this section. The commands can be run from the DB2 Command Line Processor.

Step 1: Registering the wrapper

To register the Excel data source wrapper, submit a CREATE WRAPPER statement.

To create an Excel wrapper for Excel 97 or 98 called Excel_9x_Wrapper using the library file liblsexcel97.dll, submit the following statement:

```
CREATE WRAPPER Excel_9x_Wrapper LIBRARY 'liblsexcel97.dll';
```

To create an Excel wrapper for Excel 2000 called Excel_2000_Wrapper using the library file liblsexcel2k.dll, submit the following statement:

```
CREATE WRAPPER Excel_2000_Wrapper LIBRARY 'liblsexcel2k.dll';
```

For more information on the CREATE WRAPPER statement, see the *DB2 SQL Reference*.

Step 2: Registering the server

Register the Excel server to the federated system using a CREATE SERVER statement.

For example, to create a server called biochem_lab, with a node name of biochem_node1 that registers the server for the Excel_2000_Wrapper wrapper created in "Step 1: Registering the wrapper", submit the following statement:

```
CREATE SERVER biochem_lab
TYPE Excel_2000
VERSION '2000'
WRAPPER Excel_2000_Wrapper
OPTIONS(
NODE 'biochem_node1');
```

Argument definitions

TYPE Specifies the server type, either Excel_9x or Excel_2000. This argument is required.

VERSION
Specifies the Excel version installed on the server, either '9x' or '2000'. This argument is required.

WRAPPER
Specifies the name of the wrapper that you registered in "Step 1: Registering the wrapper" on page 57. This argument is required.

Option definitions

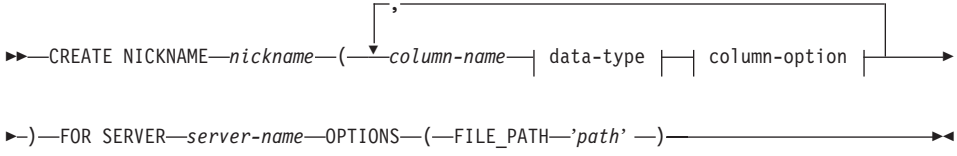
NODE
Specifies the node name that you choose. It is the name you give to the local node. It can contain any text string you like. This option is required.

For more information on the CREATE SERVER statement, see the *DB2 SQL Reference*.

Step 3: Registering nicknames

To map the Excel data source to relational tables, create a nickname using the CREATE NICKNAME statement.

CREATE NICKNAME syntax (for Excel)



data-type:

column-option:

For more information on the CREATE NICKNAME statement, see the *DB2 SQL Reference*.

FOR SERVER
Identifies the server that you registered in "Step 2: Registering the server" on page 57. This server is used to access the Excel spreadsheet.

Specify the server name followed by a period, then the base filename. For example, if biochem_lab server accesses the CompoundMaster.xls file, you would specify biochem_lab.CompoundMaster.

Option definitions

FILE_PATH

Specifies the fully qualified directory path and file name of the Excel spreadsheet that you want to access.

The statement in the following example creates the Compounds nickname from the Excel spreadsheet file named CompoundMaster.xls. The file contains three columns of data that are being defined to the federated system as Compound_ID, CompoundName, and MolWeight.

```
CREATE NICKNAME Compounds (
Compound_ID INTEGER,
CompoundName VARCHAR(50),
MolWeight FLOAT)
FOR SERVER biochem_lab.CompoundMaster
OPTIONS(PATH 'C:\My Documents\CompoundMaster.xls');
```

Running queries

This section lists several sample Excel spreadsheet queries using the example nickname Compounds from "Step 3: Registering nicknames" on page 58.

The following query displays all compound_ID's where the molecular weight is greater than 2000:

```
SELECT compound_ID
FROM Compounds
WHERE MolWeight > 200;
```

The following query displays all records where the compound name or molecular weight is null:

```
SELECT *
FROM Compounds
WHERE CompoundName IS NULL
OR MolWeight IS NULL;
```

The following query displays all records where the compound name contains the string 'ase' and the molecular weight is greater than or equal to 300:

```
SELECT *
FROM Compounds
WHERE CompoundName LIKE '%ase%
AND MolWeight >=300;
```

Sample scenario

This section demonstrates a sample implementation of the Excel_2000 wrapper accessing an Excel 2000 spreadsheet located in the C:\Data directory. The scenario registers the wrapper, registers a server and registers one nickname, that will be used to access the spreadsheet. The statements shown in the scenario are entered using the DB2 Command Line Processor. After the wrapper is registered, you can run queries against the spreadsheet.

The scenario starts with a compound spreadsheet, called Compund_Master.xls, with 4 columns and 9 rows. The fully-qualified path name to the file is C:\Data\Compound_Master.xls. The contents are show in Table 14.

Table 14. Sample spreadsheet Compound_Master.xls

	A	B	C	D
1	compound_A	1.23	367	tested
2	compound_G		210	
3	compound_F	0.000425536	174	tested
4	compound_Y	1.00256		tested
5	compound_Q		1024	
6	compound_B	33.5362		
7	compound_S	0.96723	67	tested
8				
9	compound_O	1.2		tested

1. Register the Excel_2000 wrapper:

```
db2 => CREATE WRAPPER Excel_2000 LIBRARY 'liblsexcel2k.dll'
```

2. Register the server:

```
db2 => CREATE SERVER biochem_lab TYPE Excel2000 VERSION '2000'
WRAPPER Excel_2000 OPTIONS(NODE 'biochem_node1')
```

3. Register a nickname that refers to the Excel spreadsheet:

```
db2 => CREATE NICKNAME Compound_Master (compound_name VARCHAR(40), weight FLOAT,
mol_count INTEGER, was_tested VARCHAR(20)) FOR biochem_lab.compound_master
OPTIONS ( PATH 'C:\Data\Compound_Master.xls')
```

The registration process is complete. The Excel data source is now part of the federated system, and can be used in SQL queries.

The following examples show sample SQL queries and results obtained using the Excel data source.

- Sample SQL query: ″Give me all the compound data where mol_count is greater than 100″

```
SELECT * FROM compound_master WHERE mol_count > 100
```

Result: All fields for rows 1, 2, 3, 5, and 7.

- Sample SQL query: ″Give me the compound_name and mol_count for all compounds where the mol_count has not yet been determined.

```
SELECT compound_name, mol_count FROM compound_master
WHERE mol_count IS NULL
```

Result: Fields compound_name & mol_count of rows 4, 6, 8 and 9 from the spreadsheet.

- Sample SQL query: ″Count the number of compounds that have not been tested and the weight is greater than 1.″

```
SELECT count(*) FROM compound_master
WHERE was_tested IS NULL AND weight > 1
```

Result: The record count of 1 which represents the single row 6 from the spreadsheet that meets the criteria.

- Sample SQL query: ″Give me the compound_name and mol_count for all compounds where the mol_count has been determined and is less than the average mol_count.″

```
SELECT compound_name, mol_count
FROM compound_master
WHERE mol_count IS NOT NULL
AND mol_count < (SELECT AVG(mol_count) FROM compound_master
                 WHERE mol_count IS NOT NULL AND was_tested IS NOT NULL)
```

The sub-query returns the average 368 to the main query which then returns Table 15:

Table 15. Query results

COMPOUND_NAME	MOL_COUNT
compound_A	367
compound_G	210
compound_F	174
compound_S	67

Limitations and considerations

This section contains a list of limitations and considerations associated with the use of the Excel wrapper.

Wrapper considerations

- Use the Excel_9x wrapper when using the MS Excel 97 or 98 application.
- Use the Excel_2000 wrapper when using the MS Excel 2000 application.

Wrapper limitations

- The Excel wrappers are only available for Microsoft Windows operating systems that support DB2 Universal Database Enterprise Edition and DB2 Universal Database Enterprise-Extended Edition.
- Passthru sessions are not allowed with the Excel wrappers.
- Excel spreadsheet data can only be read not written.
- The Excel wrapper does not allow the altering of column names using the ALTER NICKNAME statement.

Excel file limitations

- Data types must be consistent within each column and the column data types must be described correctly during the register nickname process.
- The Excel wrappers can only access the primary spreadsheet within an Excel workbook.
- The Excel_2000 wrapper can access Excel 97, 98, and 2000 spreadsheets.
- The Excel_9x wrapper can access Excel 97 and 98 spreadsheets.
- Blank cells in the spreadsheet are interpreted as NULL.
- Up to 10 consecutive blank rows can exist in the spreadsheet and be included in the data set. More than 10 consecutive blank rows are interpreted as the end of the data set.
- Blank columns can exist in the spreadsheet. However, these columns must be registered and described as valid fields even if they will not be used.

File access control model

The database management system accesses Excel files with the authority of the LOG ON AS property of the DB2 database service. This setting can be viewed in the LOG ON properties page for the DB2 instance. The properties page is accessed through the Windows NT Services control panel.

Messages

This section lists and describes messages you might encounter while working with the wrapper for Excel. For more information on messages, see the *DB2 Message Reference*.

Table 16. Messages issued by the wrapper for Excel

Error Code	Message	Explanation
SQL1817N	The CREATE SERVER statement does not identify the "VERSION" of data source that you want defined to the federated database.	The VERSION parameter was not specified during the CREATE SERVER statement. Correct the SQL statement and run it again.

Table 16. Messages issued by the wrapper for Excel (continued)

Error Code	Message	Explanation
SQL1822N	Unexpected error code ″-1000.<internal program code>″ received from data source ″Excel Wrapper″. Associated text and tokens are ″Memory allocation error″	Contact IBM Software Support.
SQL1822N	Unexpected error code ″-1001.<internal program code>″ received from data source ″Excel Wrapper″. Associated text and tokens are ″Unknown option″.	The option specified in the DDL statement is not supported. Correct the SQL statement and run it again.
SQL1822N	Unexpected error code ″-1002.<internal program code>″ received from data source ″Excel Wrapper″. Associated text and tokens are ″Creation of DELTA object failed″.	An internal program error has occurred. Contact IBM Software Support.
SQL1822N	Unexpected error code ″-1100.<internal program code>″ received from data source ″Excel Wrapper″. Associated text and tokens are ″Wrapper options are not supported″	Wrapper OPTIONS are not supported by this wrapper. Correct the SQL statement and run it again.
SQL1822N	Unexpected error code ″-1200.<internal program code>″ received from data source ″Excel Wrapper″. Associated text and tokens are ″<option> is an unsupported Server option″.	The specified option is not supported by this wrapper. Correct the SQL statement and run it again.
SQL1822N	Unexpected error code ″-1201.<internal program code>″ received from data source ″Excel Wrapper″. Associated text and tokens are ″Error obtaining server name″	An internal program error has occurred. Contact IBM Software Support.
SQL1822N	Unexpected error code ″-1209. <internal program code>″ received from data source ″Excel Wrapper″. Associated text and tokens are ″ Error converting VARCHAR data″	An internal program error has occurred. Contact IBM Software Support.

Table 16. Messages issued by the wrapper for Excel (continued)

Error Code	Message	Explanation
SQL1822N	Unexpected error code ″-1211.<internal program code>″ received from data source ″Excel Wrapper″. Associated text and tokens are ″ Error converting INTEGER data″	An internal program error has occurred. Contact IBM Software Support.
SQL1822N	Unexpected error code ″-1212.<internal program code>″ received from data source ″Excel Wrapper″. Associated text and tokens are ″ Error converting FLOAT data″	An internal program error has occurred. Contact IBM Software Support.
SQL1822N	Unexpected error code ″-1400.<internal program code>″ received from data source ″Excel Wrapper″. Associated text and tokens are ″<option> is an unsupported User option″	The specified option is not supported by this wrapper. Correct the SQL statement and run it again.
SQL1822N	Unexpected error code ″-1401.<internal program code>″ received from data source ″Excel Wrapper″. Associated text and tokens are ″Creation of USER Delta object failed″	An internal program error has occurred. Contact IBM Software Support.
SQL1822N	Unexpected error code ″-1500.<internal program code>″ received from data source ″Excel Wrapper″. Associated text and tokens are ″<option> is an unsupported Nickname option″	The specified option is not supported by this wrapper. Correct the SQL statement and run it again.
SQL1822N	Unexpected error code ″-1501.<internal program code>″ received from data source ″Excel Wrapper″. Associated text and tokens are ″Required option PATH not specified″	The PATH option is required to register the NICKNAME. Correct the SQL statement and run it again.
SQL1822N	Unexpected error code ″-1502.<internal program code>″ received from data source ″Excel Wrapper″. Associated text and tokens are ″Creation of NICKNAME Delta object failed″	An internal program error has occurred. Contact IBM Software Support.

Table 16. Messages issued by the wrapper for Excel (continued)

Error Code	Message	Explanation
SQL1822N	Unexpected error code ″-1503.<internal program code>″ received from data source ″Excel Wrapper″. Associated text and tokens are ″Error obtaining Nickname column type″	An internal program error has occurred. Contact IBM Software Support.
SQL1822N	Unexpected error code ″-1504.<internal program code>″ received from data source ″Excel Wrapper″. Associated text and tokens are ″Error obtaining Nickname column type name″	An internal program error has occurred. Contact IBM Software Support.
SQL1822N	Unexpected error code ″-1505.<internal program code>″ received from data source ″Excel Wrapper″. Associated text and tokens are ″received from data source ″Excel Wrapper″.	The specified <data type> is not supported by this wrapper. Correct the SQL statement and run it again.
SQL1822N	Unexpected error code ″-1506.<internal program code>″ received from data source ″Excel Wrapper″. Associated text and tokens are ″Error obtaining Nickname column info″	An internal program error has occurred. Contact IBM Software Support.
SQL1822N	Unexpected error code ″-1507.<internal program code>″ received from data source ″Excel Wrapper″. Associated text and tokens are ″<option> option cannot be dropped″	The specified option cannot be dropped because it is a required option.
SQL1822N	Unexpected error code ″-1508.VANI″ received from data source ″Excel Wrapper″. Associated text and tokens are ″Column names cannot be altered″	The altering of column names is not permitted by the Excel wrapper.
SQL1822N	Unexpected error code ″-1701.<internal program code>″ received from data source ″Excel Wrapper″. Associated text and tokens are ″Error parsing SQL″	An internal program error has occurred. Contact IBM Software Support.

Table 16. Messages issued by the wrapper for Excel (continued)

Error Code	Message	Explanation
SQL1822N	Unexpected error code ″-1702.<internal program code>″ received from data source ″Excel Wrapper″. Associated text and tokens are ″Error accessing NICKNAME object″	An internal program error has occurred. Contact IBM Software Support.
SQL1822N	Unexpected error code ″-1703.<internal program code>″ received from data source ″Excel Wrapper″. Associated text and tokens are ″Error building data storage area″	An internal program error has occurred. Contact IBM Software Support.
SQL1822N	Unexpected error code ″-1704.<internal program code>″ received from data source ″Excel Wrapper″. Associated text and tokens are ″Error linking SQL to Nickname Data″	An internal program error has occurred. Contact IBM Software Support.
SQL1822N	Unexpected error code ″-1705.<internal program code>″ received from data source ″Excel Wrapper″. Associated text and tokens are ″Excel application startup failed″	The startup of the Excel application failed. Confirm that Excel is installed on the system and has been registered with the correct version of the wrapper. Check the LOG ON AS property for the DB2 instance in the Windows NT Services control panel. The Excel application will be accessed using this authority. Confirm that this user has appropriate rights or change this property to an authorized account, then restart DB2 and run the SQL query again.
SQL1822N	Unexpected error code ″-1706.<internal program code>″ received from data source ″Excel Wrapper″. Associated text and tokens are ″Error opening source spreadsheet″	A problem occurred while opening the spreadsheet referenced by the nickname in the SQL query. Ensure that the file exists in the PATH specified during the CREATE NICKNAME statement during registration.

Table 16. Messages issued by the wrapper for Excel (continued)

Error Code	Message	Explanation
SQL1822N	Unexpected error code ″-1707.<internal program code>″ received from data source ″Excel Wrapper″. Associated text and tokens are ″Error accessing DL output storage area″	An internal program error occurred. Contact IBM Software Support.
SQL1822N	Unexpected error code ″-1708.<internal program code>″ received from data source ″Excel Wrapper″. Associated text and tokens are ″Excel application end failed″	An internal program error occurred. If this error persists after repeated queries, contact IBM Software Support.
SQL1822N	Unexpected error code ″-1711.<internal program code>″ received from data source ″Excel Wrapper″. Associated text and tokens are ″Error during fetch, possible data/col type mismatch″	The data fetched during the SQL query was of a different data type than the data type specified during the registration of the nickname. Correct the data in the source spreadsheet or correct the registered data type in the nickname. If this does not correct the problem, contact IBM Software Support.
SQL1822N	Unexpected error code ″-1900.<internal program code>″ received from data source ″Excel Wrapper″. Associated text and tokens are ″Memory allocation error″	An internal program error has occurred. Contact IBM Software Support.

Chapter 6. Altering nicknames

You can use the ALTER NICKNAME statement to modify the federated database's representation of a data source or view. You can:

- Change the local name of the columns in a table or view
- Change the local data types of these columns
- Add, change, or delete options for these columns

For more information on the ALTER NICKNAME statement, see the *DB2 SQL Reference*.

Changing the column name

The SQL statement in the following example changes the local column name from DCODE to DRUGCODE. The nickname DRUGDATA1 refers to a local table-structured file called drugdata1.txt. The DCODE column is the local column name that references the first field in the file.

```
ALTER NICKNAME DRUGDATA1
   ALTER COLUMN DCODE
   LOCAL NAME DRUGCODE
```

Changing the data type

The SQL statement in the following example changes the local data type of the DRUG column to CHAR(30). The DRUG column was originally defined as a CHAR(20) using a CREATE NICKNAME statement. The nickname DRUGDATA1 refers to a local table-structured file called drugdata1.txt.

```
ALTER NICKNAME DRUGDATA1
   ALTER COLUMN DRUG
   LOCAL TYPE CHAR(30)
```

Changing the file path

The SQL statement in the following example changes the fully qualified path for the table-structured file, drugdata1.txt. The path was originally defined as '/user/pat/drugdata1.txt' using a CREATE NICKNAME statement. The nickname DRUGDATA1 refers to a local table-structured file called drugdata1.txt.

```
ALTER NICKNAME DRUGDATA1
   OPTIONS (SET FILE_PATH '/usr/kelly/data/drugdata1.txt')
```

Notices

IBM may not offer the products, services, or features discussed in this document in all countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make

improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Canada Limited
Office of the Lab Director
1150 Eglinton Ave. East
North York, Ontario
M3C 1H7
CANADA

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information may contain examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:

This information may contain sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Each copy or any portion of these sample programs or any derivative work must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

Trademarks

The following terms, which may be denoted by an asterisk(*), are trademarks of International Business Machines Corporation in the United States, other countries, or both.

ACF/VTAM
AISPO
AIX
AIX/6000
AIXwindows
AnyNet
APPN
AS/400
BookManager
CICS
C Set++
C/370
DATABASE 2
DataHub
DataJoiner
DataPropagator
DataRefresher
DB2
DB2 Connect
DB2 Extenders
DB2 OLAP Server
DB2 Universal Database
Distributed Relational Database Architecture
DRDA
eNetwork
Extended Services
FFST
First Failure Support Technology
IBM
IMS
IMS/ESA
LAN DistanceMVS
MVS/ESA
MVS/XA
Net.Data
OS/2
OS/390
OS/400
PowerPC
QBIC
QMF
RACF
RISC System/6000
RS/6000
S/370
SP
SQL/DS
SQL/400
System/370
System/390
SystemView
VisualAge
VM/ESA
VSE/ESA
VTAM
WebExplorer
WIN-OS/2

The following terms are trademarks or registered trademarks of other companies:

Microsoft, Windows, and Windows NT are trademarks or registered trademarks of Microsoft Corporation.

Java or all Java-based trademarks and logos, and Solaris are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Tivoli and NetView are trademarks of Tivoli Systems Inc. in the United States, other countries, or both.

UNIX is a registered trademark in the United States, other countries or both and is licensed exclusively through X/Open Company Limited.

Other company, product, or service names, which may be denoted by a double asterisk(**) may be trademarks or service marks of others.

Bibliography

This bibliography contains DB2 Universal Database publications that you might find useful while working with DB2 Life Sciences Data Connect.

- *DB2 Connect User's Guide* (SC09-2954)
- *DB2 for UNIX Quick Beginnings* (GC09-2970)
- *DB2 SQL Reference* (SC09-2974)
- *DB2 Administration Guide: Planning* (SC09-2946)
- *DB2 Administration Guide: Implementation* (SC09-2944)
- *DB2 Administration Guide: Performance* (SC09-2945)
- *DB2 Message Reference* (GC09-2978)
- *IBM DB2 Universal Database Release Notes Version 7.2/Version 7.1 FixPak 4*

Index

Contacting IBM

If you have a technical problem, please review and carry out the actions suggested by the *Troubleshooting Guide* before contacting DB2 Customer Support. This guide suggests information that you can gather to help DB2 Customer Support to serve you better.

For information or to order any of the DB2 Universal Database products contact an IBM representative at a local branch office or contact any authorized IBM software remarketer.

If you live in the U.S.A., then you can call one of the following numbers:

- 1-800-237-5511 for customer support
- 1-888-426-4343 to learn about available service options

Product Information

If you live in the U.S.A., then you can call one of the following numbers:

- 1-800-IBM-CALL (1-800-426-2255) or 1-800-3IBM-OS2 (1-800-342-6672) to order products or get general information.
- 1-800-879-2755 to order publications.

http://www.ibm.com/software/data/
The DB2 World Wide Web pages provide current DB2 information about news, product descriptions, education schedules, and more.

http://www.ibm.com/software/data/db2/library/
The DB2 Product and Service Technical Library provides access to frequently asked questions, fixes, books, and up-to-date DB2 technical information.

Note: This information may be in English only.

http://www.elink.ibmlink.ibm.com/pbl/pbl/
The International Publications ordering Web site provides information on how to order books.

http://www.ibm.com/education/certify/
The Professional Certification Program from the IBM Web site provides certification test information for a variety of IBM products, including DB2.

ftp.software.ibm.com
Log on as anonymous. In the directory /ps/products/db2, you can find demos, fixes, information, and tools relating to DB2 and many other products.

comp.databases.ibm-db2, bit.listserv.db2-l
These Internet newsgroups are available for users to discuss their experiences with DB2 products.

On Compuserve: GO IBMDB2
Enter this command to access the IBM DB2 Family forums. All DB2 products are supported through these forums.

For information on how to contact IBM outside of the United States, refer to Appendix A of the *IBM Software Support Handbook*. To access this document, go to the following Web page: http://www.ibm.com/support/, and then select the IBM Software Support Handbook link near the bottom of the page.

Note: In some countries, IBM-authorized dealers should contact their dealer support structure instead of the IBM Support Center.

Part Number: CT0M8NA

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

(1P) P/N: CT0M8NA