



HDF5 ODBC Connector – User's Guide

Release 1.0.1b1

Gerd Heber, The HDF Group

March 08, 2017

CONTENTS

1	About the HDF5 ODBC Connector	3
1.1	Key Features	3
1.2	ODBC Driver Library	3
1.3	HDF5 Library References	3
1.4	Coexistence of 32-Bit and 64-Bit Drivers on the Same System	4
2	Configuring ODBC Data Sources	5
2.1	Configuration for Windows Systems	5
2.2	Configuration for Linux Systems	8
3	Connecting to HDF5 Files via ODBC	11
3.1	Standalone Mode	11
3.2	Client-Server Mode	16
4	The <code>SELECT</code> Statement	19
4.1	Catalogs, Schemas, and Tables	19
4.2	HDF5 Datatypes and SQL Data Types	21
4.3	The <code>ROWID</code> Pseudo-Column	21
4.4	2D-Mode	23
4.5	The <code>HDF5.VRTL.Attributes</code> Table	24
4.6	The <code>HDF5.VRTL.Shapes</code> Table	25
5	Appendix	27
5.1	Platforms and System Requirements	27
5.2	ODBC Conformance Level	27
5.3	Supported ODBC/SQL Functions	27
5.4	Error Messages and Logging	30
	Bibliography	35

The product or products described in this book are licensed products of The HDF Group or its affiliates.

Linux is a registered trademark of Linus Torvalds.

Microsoft, Active Directory, Excel, Windows, Windows NT, and Windows Server are registered trademarks of Microsoft Corporation in the United States and other countries.

Simba, the Simba logo, SimbaEngine, SimbaEngine C/S, SimbaExpress and SimbaLib are registered trademarks of Simba Technologies Inc.

Unicode is a registered trademark of Unicode, Inc. in the United States and other countries.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other product and company names mentioned herein may be the trademarks of their respective owners.

THE INFORMATION CONTAINED IN THIS DOCUMENT IS PROVIDED ON AN “AS-IS” BASIS, WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO THE ABOVE EXCLUSION MAY NOT APPLY TO YOU. IN NO EVENT WILL THE HDF GROUP BE LIABLE FOR ANY INDIRECT, DIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, INCLUDING LOST PROFITS OR LOST SAVINGS, EVEN IF EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Information contained in this document may contain technical inaccuracies or typographical errors. Information may be changed or updated without notice. The HDF Group may also make improvements or changes in the products or services described in this information at any time without notice. To maintain the quality of our products and services, we would like your comments on the accuracy, clarity, organization, and value of this document. Please email: help@hdfgroup.org. Any comments or materials (collectively referred to as “Feedback”) sent to The HDF Group will be deemed non-confidential. The HDF Group will have no obligation of any kind with respect to Feedback and will be free to use, reproduce, disclose, exhibit, display, transform, create derivative works of, and distribute the Feedback and derivative works thereof without limitation on a royalty-free basis. Further, The HDF Group will be free to use any ideas, concepts, know-how, or techniques contained in such Feedback for any purpose whatsoever, including developing, manufacturing, or marketing products or services incorporating Feedback.

Copyright © 2015-2017 by The HDF Group. All Rights Reserved.

ABOUT THE HDF5 ODBC CONNECTOR

The HDF5 ODBC Connector lets applications access data stored in HDF5 files through an ODBC or JDBC ¹ interface. It supports the full core-level ODBC 3.80 specification and most of the Level 1 and Level 2 APIs. A list of supported SQL functions can be found in the appendix *Supported ODBC/SQL Functions*.

1.1 Key Features

- Standalone or client-server mode deployment
- Read-only access
- Mapping of HDF5 groups as relational schemata
- Support for pre-defined scalar types, such as integers, floating-point numbers, and strings
- Support for compounds of pre-defined scalar types
- A ROWID pseudo-column
- A 2D-mode for rendering two-dimensional datasets
- Virtual tables for HDF5 attributes and dataset shapes

1.2 ODBC Driver Library

The HDF5 ODBC Connector contains licensed components of the SimbaEngine SDK version 9.5.15 of Simba Technologies Inc.

1.3 HDF5 Library References

The HDF5 ODBC Connector contains binaries of the following Open Source Software:

- HDF5 version 1.10.0-patch1 [[HDF5](#)]
- Zlib version 1.2.11 [[Zlib](#)]
- Szzip version 2.1.1 [[Szzip](#)]

¹ The JDBC interface is available only in the client-server deployment mode. See *Client-Server Mode*.

1.4 Coexistence of 32-Bit and 64-Bit Drivers on the Same System

The 32-bit version of the connector is for 32-bit applications running on 32-bit or 64-bit versions of an OS. The 64-bit version of the connector is for 64-bit applications running on a 64-bit version of an OS.

It is very likely that you are running a 64-bit version of your OS and that you have a mixture of 32-bit and 64-bit applications. In this case, it is recommended to install both connector versions.

If your OS version is a 32-bit version then you cannot run 64-bit applications, and you need only the 32-bit version of the HDF5 ODBC connector.

If your OS version is a 64-bit version and you plan on using only 64-bit applications with the HDF5 ODBC connector, then you need only the 64-bit version of the connector. In this case, you *will not* be able to connect to HDF5-backed ODBC resources from 32-bit applications.

CONFIGURING ODBC DATA SOURCES

2.1 Configuration for Windows Systems

In this section, the installation and configuration of the HDF5 ODBC Connector are described, as well as the creation and configuration of ODBC data sources backed by HDF5 files. (See the section *Error Messages and Logging* in the appendix for how to enable and configure logging.)

2.1.1 Determining the Current Connector Version

To determine the currently installed version of HDF5 ODBC Connector, select **Start > Control Panel > Add or Remove Programs**. From the *Currently Installed Programs* list, find HDF5 ODBC Connector BASIC. The release number can be found in the *Version* column.

2.1.2 Windows ODBC Driver Directories

The *default* installation path is different for different processor architectures.

Windows (32-bit)

C:\Program Files\HDF5 ODBC Connector BASIC

Windows (64-bit)

The HDF5 ODBC Connector for 32-bit applications is installed in:

C:\Program Files (x86)\HDF5 ODBC Connector BASIC

The HDF5 ODBC Connector for 64-bit applications is installed in:

C:\Program Files\HDF5 ODBC Connector BASIC

Let <INSTALL_DIR> denote the installation directory where the software was installed. In each installation directory, there are three subdirectories

```
<INSTALL_DIR>\Bin
<INSTALL_DIR>\HDF5
<INSTALL_DIR>\SSLCertificates
```

The Bin directory contains the installation binaries and an error messages configuration file. The HDF5 directory contains a few sample HDF5 files. These files are referenced by the *HDF5Test* ODBC data source. The SSLCertificates directory is only used for connections to HDF5 ODBC servers over a TCP connection and can be ignored.

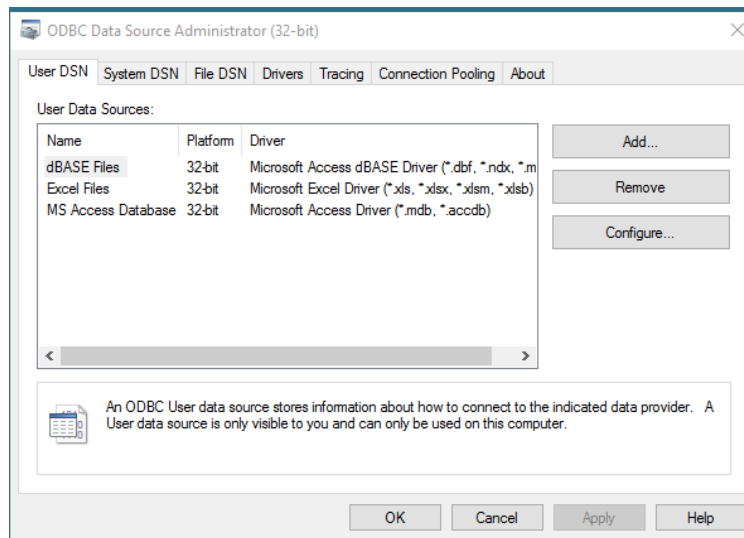
2.1.3 Creating ODBC Data Sources Backed by HDF5 Files

In this section, the creation and configuration of an HDF5-backed ODBC data source is described. To be specific one of the sample files included in the HDF5 ODBC Connector installation, `tickdata.h5` is used. The reader should feel free to use a different HDF5 file.

The creation is a simple three-step process:

1. Selecting a name (and optional description) for the new ODBC data source.
2. Selecting an HDF5 file backing the ODBC data source.
3. Testing the ODBC data source.

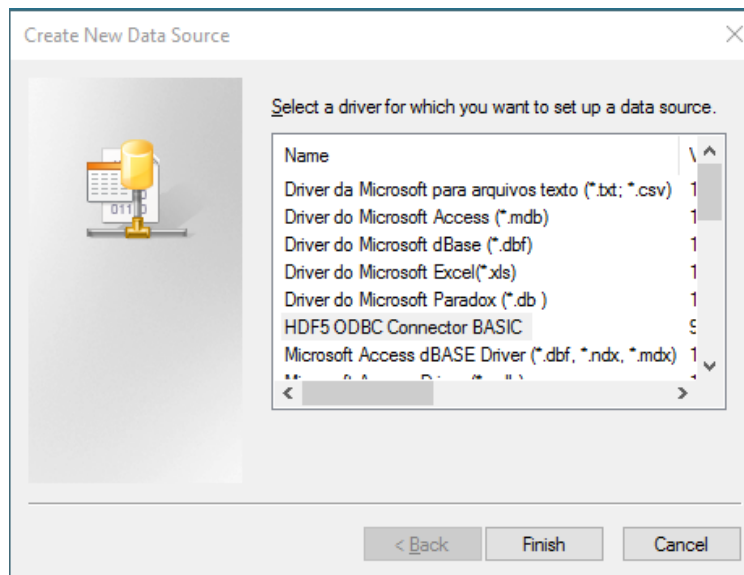
Open the *ODBC Data Source Administrator* and select the *User DSN* tab.



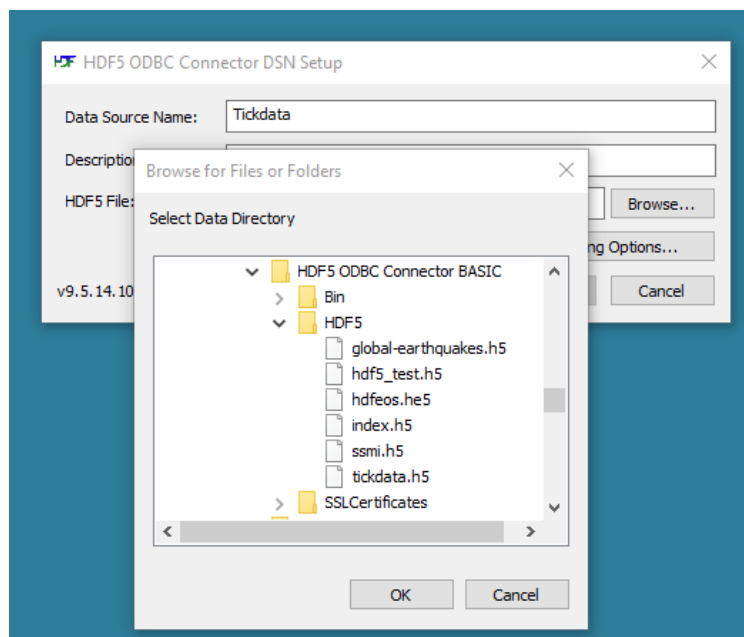
Note: There are two kinds of *data source names* (DSN), system-wide DSN (System DSN) and DSN only visible to a particular user (User DSN). As the name suggests, the creation of System DSN requires administrative privileges.

Caution: Some applications support System DSN only. Please consult your documentation for details.

Click **Add** and the *Create New Data Source* dialog will open.



Select the driver named *HDF5 ODBC Connector BASIC* and click **Finish**. This will open the *HDF5 ODBC Connector DSN Setup* main dialog. Enter *Tickdata* as the name for the new ODBC data source and click the **Browse** button.



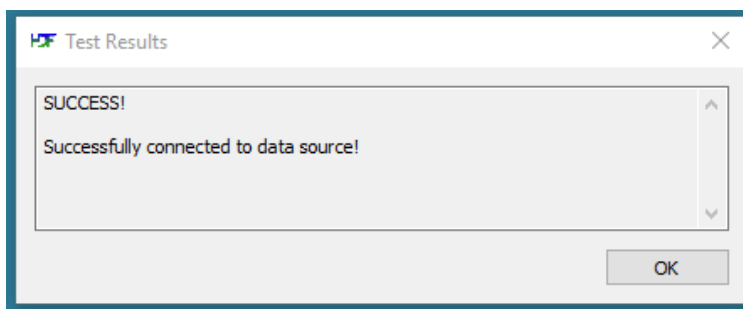
Navigate to and select the *tickdata.h5* file, which is located in the *HDF5* subdirectory of the *HDF5 ODBC Connector*’s installation directory, such as

```
C:\Program Files (x86)\HDF5 ODBC Connector BASIC
```

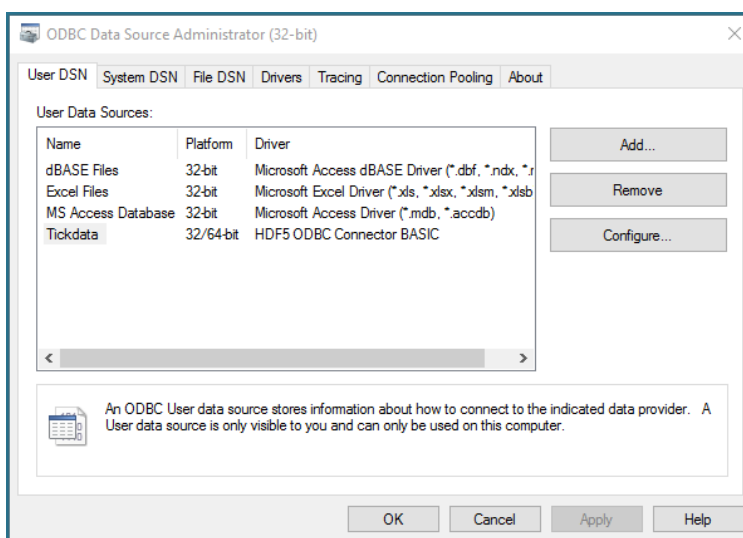
or

```
C:\Program Files\HDF5 ODBC Connector BASIC
```

Click **OK** to close the *Browse for File or Folders* dialog, and click **Test** to establish a test connection to the new data source. If everything went according to plan a pop-up similar to the one shown below will appear.



Click **OK** once to dismiss the pop-up followed by **OK** a second time to complete the setup. You will be returned to the *ODBC Data Source Administrator* main window. Notice the new *Tickdata* data source in the *Name* column.



In the next chapter, different applications will be used to connect to ODBC data sources backed by HDF5 files.

2.2 Configuration for Linux Systems

This section contains basic information about the installation layout and sufficient information to help you get started and create ODBC data sources backed by HDF5 files.

2.2.1 Linux ODBC Driver Directories

Let `<INSTALL_DIR>` denote the installation directory where the software was installed. The default installation directory is `$HOME/HDF5_ODBC_Basic`. In each installation directory, there are three subdirectories

```
<INSTALL_DIR>/Bin
<INSTALL_DIR>/HDF5
<INSTALL_DIR>/SSLCertificates
```

The `Bin` directory contains the installation binaries and an error messages configuration file. There are separate directories for 32-bit and 64-bit binaries:

```
<INSTALL_DIR>/Bin/Linux_x86
<INSTALL_DIR>/Bin/Linux_x86_64
```

The HDF5 directory contains a few sample HDF5 files. These files are referenced by the *HDF5Test* ODBC data source. The *SSLCertificates* directory is only used for connections to HDF5 ODBC servers over a TCP connection and can be ignored.

2.2.2 Creating ODBC Data Sources Backed by HDF5 Files

The standard *unixODBC* driver manager installation includes two simple tools, *odbcinst* and *isql*, for managing ODBC data sources and running simple queries, respectively. In this example, we will use one of the sample HDF5 files included in the HDF5 ODBC connector installation. The file is called *tickdata.h5* and located in *<INSTALL_DIR>/HDF5*. We will use *odbcinst* to create a new ODBC data source called *ticks*. Using your favorite text editor, create a template file called *dsn_ticks* with the following content:

```
[ticks]
Driver = HDF5ODBCDSIIDriver
Description = My ticks datasource
DBF=/home/user/HDF5_ODBC_Basic/HDF5/tickdata.h5
Locale=en-US
```

Please make sure that the path under the *DBF* key matches your installation, i.e., contains a valid path to *tickdata.h5*! Install a user DSN with:

```
odbcinst -i -s -h -f dsn_ticks
```

Verify the installation by querying the installed ODBC data sources:

```
odbcinst -q -s
```

The output should contain *[ticks]*:

```
...
[HDF5Test]
[HDF5ServerTest]
[ticks]
...
```

Finally, run a sample query with *isql*:

```
user@host:~$ isql ticks
+-----+
| Connected!                                |
|                                           |
| sql-statement                            |
| help [tablename]                        |
| quit                                    |
|                                           |
+-----+
SQL> SELECT COUNT(*) FROM HDF5."/". "22-09-2011"
+-----+
| EXPR_1                                |
+-----+
| 23536                                |
+-----+
SQLRowCount returns -1
1 rows fetched
SQL>
```

Congratulations! You’ve just created your first HDF5-backed ODBC data source.

In the following sections, we describe how to access ODBC data sources from certain client applications. Not all of them might be available for your particular operating system. If your application is not among the examples, please check its documentation for information on how to configure its ODBC client with existing ODBC data sources.

CONNECTING TO HDF5 FILES VIA ODBC

The HDF5 ODBC connector supports two distinct access modes. We refer to them as *standalone* and *client-server* mode, respectively.

In standalone mode, the connector supports only HDF5 files which are accessible via a file system mounted on the installation host. In other words, an ODBC client application must run on a machine on which the connector is installed and that has direct access to the HDF5 files.

In client-server mode, the client application talks to an HDF5/ODBC server over a network connection. This mode is similar to accessing a traditional RDBMS with an ODBC or JDBC client. In this mode, the client application cannot directly access the HDF5 files, which are located elsewhere. See section *Client-Server Mode* for an example on how to load data from HDF5 into an Apache Spark DataFrame.

3.1 Standalone Mode

In this chapter, several methods of connecting to HDF5 files via ODBC are discussed. Examples for three popular ODBC clients, Excel, `pyodbc` [*pyodbc*], and R [*R*] are shown.

3.1.1 Connecting to a Data Source

The HDF5 ODBC connector installation includes a sample ODBC data source called *HDF5Test*. In this section, Excel is used to read the HDF5 dataset with path name

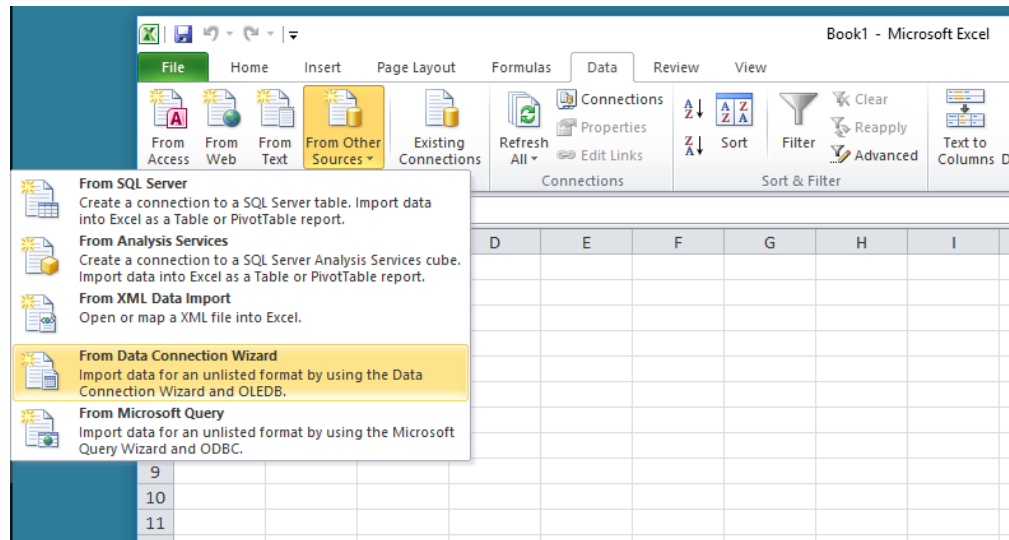
```
/tickdata.h5/22-09-2011
```

or table identifier (see *Catalogs, Schemas, and Tables*)

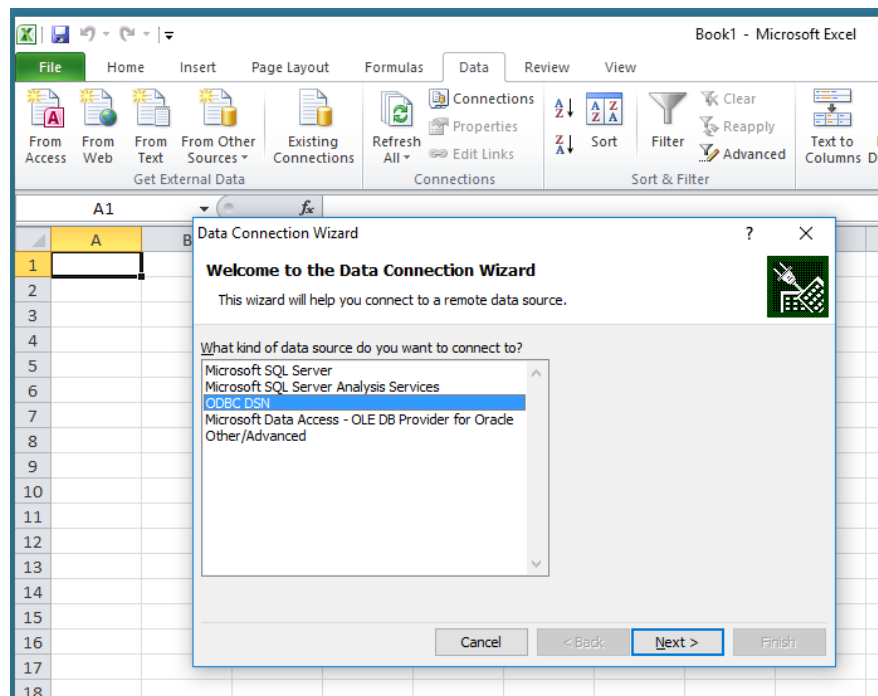
```
HDF5."/tickdata.h5/". "22-09-2011"
```

Example: Microsoft Excel 2010

On the ribbon, select the *Data* tab, click the *From Other Sources*, and pick the *From Data Connection Wizard* menu item.



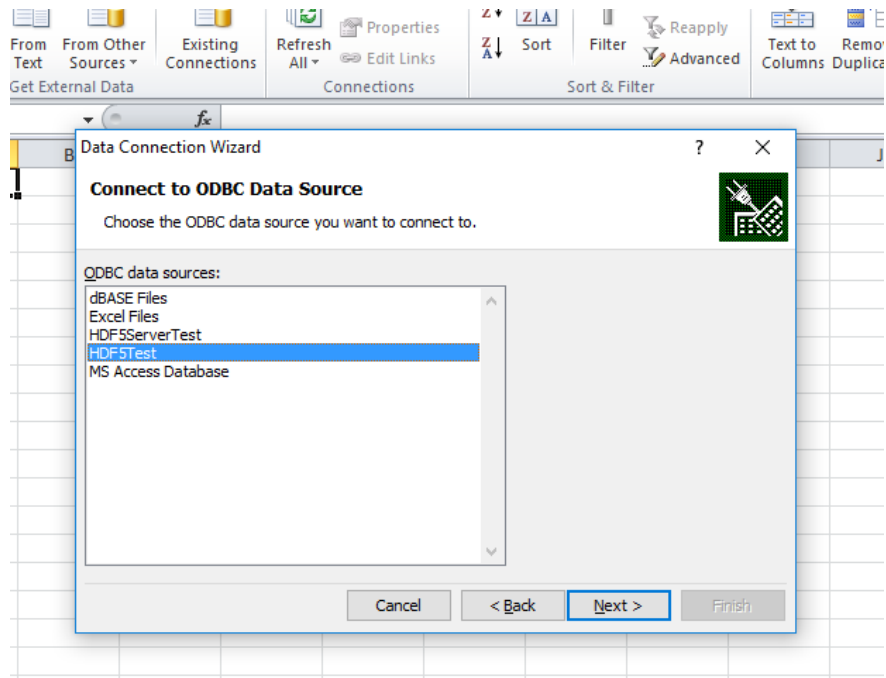
This will open the *Data Connection Wizard* dialog. Select the *ODBC DSN* data source and click **Next**.



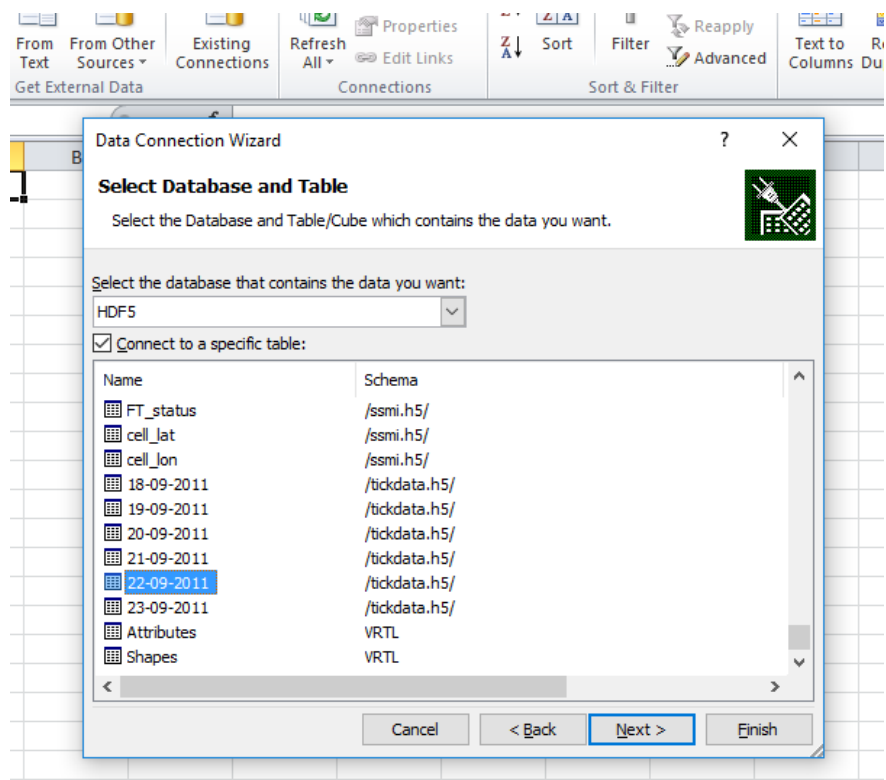
You will be offered a list of available ODBC data sources.

Note: The list on your system might be different from the list shown below. If you can’t see the *HDF5Test* data source listed, please contact your administrator and confirm that the HDF5 ODBC connector was installed and tested.

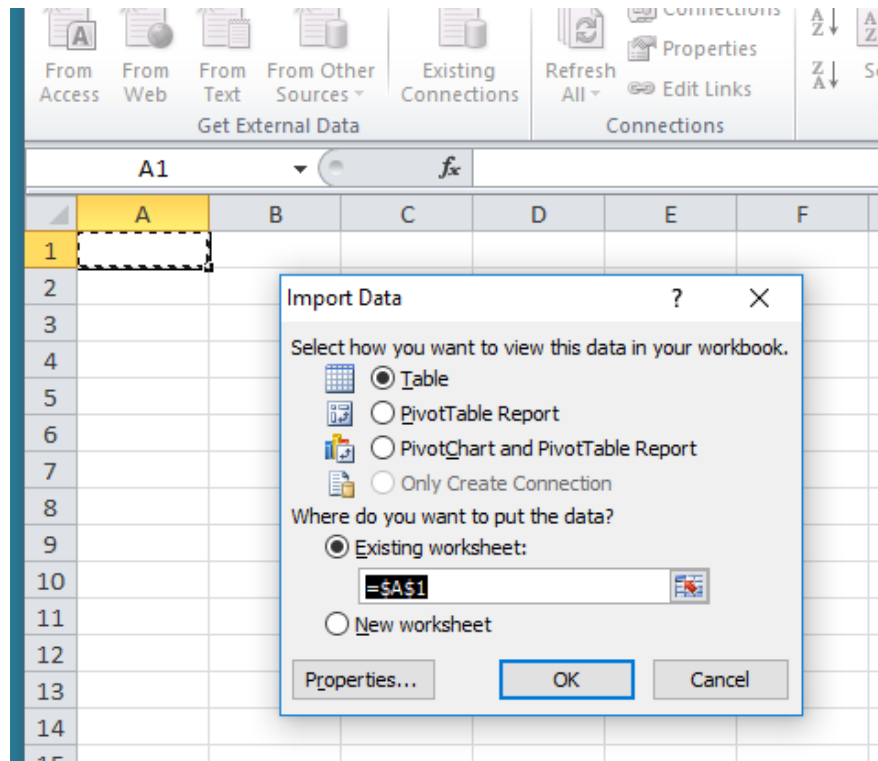
Select *HDF5Test* and click **Next**.



You will be presented with a list of tables. Select the table named **22-09-2011** and click **Finish**.



The wizard will prompt you for a cell destination on the worksheet. Accept the default (\$A\$1) and click **OK**.



After a few moments, the data will be loaded into the first four columns of the work sheet.

ROWID	Time	Bid	Ask
1	0 00:00:02:200000	0.888899982	0.889599979
2	1 00:00:04:300000	0.888800025	0.889500022
3	2 00:00:22:500000	0.888899982	0.889599979
4	3 00:00:23:200000	0.888999999	0.889699996
5	4 00:00:33:900000	0.888800025	0.889500022
6	5 00:00:39:400000	0.888899982	0.889599979
7	6 00:00:42:800000	0.888999999	0.889699996
8	7 00:00:47:300000	0.888899982	0.889599979
9	8 00:00:59:300000	0.888999999	0.889699996
10	9 00:01:01:400000	0.889100015	0.889800012
11	10 00:01:18:600000	0.888999999	0.889699996
12	11 00:02:24:600000	0.889100015	0.889800012
13	12 00:02:25:300000	0.889199972	0.889900029
14	13 00:02:25:600000	0.889100015	0.889800012
15	14 00:02:27:600000	0.889199972	0.889900029
16	15 00:02:44:200000	0.889299989	0.889999986
17	16 00:02:45:800000	0.889400005	0.890100002
18	17 00:03:15:400000	0.889299989	0.889999986
19	18 00:03:26:400000	0.889400005	0.890100002
20	19 00:04:27:600000	0.889299989	0.889999986
21	20 00:04:29:000000	0.889199972	0.889900029
22	21 00:04:34:200000	0.889100015	0.889800012
23	22 00:04:34:500000	0.889199972	0.889900029
24	23 00:05:05:800000	0.889299989	0.889999986
25	24 00:05:14:700000	0.889199972	0.889900029

Try loading a few more tables onto the same or other worksheets!

3.1.2 Connecting to a Data Source Using a Connection String

Many BI tools offer data import capabilities similar to Excel’s *Data Connection Wizard*. Other environments, such as scripting languages, come equipped with ODBC clients packaged in modules that offer greater programmatic control over the import process. In this section, Python’s `pyodbc` module and R’s `RODBC` module are used to connect to the

same ODBC data source.

Example pyodbc

The `pyodbc` module can be used with existing ODBC resources, such as *HDF5Test*, or dynamically establish connections. The general workflow is as follows:

1. Import the module
2. Create a connection
3. Create a cursor
4. Execute a query against that cursor
5. Fetch all or part of the results (rows)
6. Process the results
7. Close the connection

When executed, the script shown below returns the exact same data as the Excel example shown earlier.

```
import pyodbc

with pyodbc.connect('DSN=HDF5Test') as conn:

    cursor = conn.cursor()
    cursor.execute('''
SELECT * FROM HDF5."/tickdata.h5/"."22-09-2011"
''')
    print(cursor.fetchall())
```

To establish a connection *dynamically* (i.e., without an existing ODBC data source) the driver and the target HDF5 file need to be specified in the connection string. Assuming that the HDF5 file name is supplied as the first argument of a Python script, the file name can be interpolated into the connection string as follows:

```
import pyodbc, sys

with pyodbc.connect( \
    'DRIVER={HDF5 ODBC Connector BASIC};2D_MODE=OFF;DBF={}'.format(sys.argv[1])) \
    as conn:

    cursor = conn.cursor()
    query = '''
SELECT AVG(Value)
FROM HDF5."/HDFEOS/SWATHS/HIRDLS/Data Fields/"."03
WHERE Value > -999.0
'''
    cursor.execute(query)
    print(cursor.fetchall()[0])
```

When saved as `ask_odbc.py`, we can run a query directly against one of the HDF5 files from the collection of HDF5 samples.

```
python ask_odbc.py "C:\Program Files\HDF5 ODBC Connector BASIC\HDF5\hdfeos.he5"
(1.3718425016092453e-05, )
```

Example: R

The ODBC workflow in R is similar to `pyodbc`. When running the R-script shown below, the names of all the tables in the HDF5. `"/tickdata.h5"` schema are listed, followed by a plot of the ask and bid prices from the table

```
HDF5."/tickdata.h5/". "22-09-2011"
```

```
# load the RODBC module and establish a connection
library(RODBC)
con<-odbcConnect("HDF5Test",believeNRows=TRUE, readOnlyOptimize=TRUE)

# print the table names in schema "/tickdata.h5/"

tbls<-sqlTables(con, catalog="HDF5",schema="/tickdata.h5/")
print(tbls$TABLE_NAME)
readline("Press <return to continue")

# read and plot a time series
data<- sqlQuery(con, "SELECT * from HDF5.\"/tickdata.h5/\".\"22-09-2011\"")
plot(data[[2]],data[[3]])

# close the connection
odbcClose(con)
```

3.2 Client-Server Mode

In client-server mode, applications connect through ODBC or JDBC clients to an HDF5/ODBC server over a network connection.

The following clients are available:

ODBC

- *SimbaClient_MTDLL.dll* (Windows)
- *libSimbaClient.so* (Linux)
- *libSimbaClient.dylib* (MacOS X)

JDBC

- *SimbaJDBCClient41.jar* (Java archive)

3.2.1 Example: Apache Spark

Apache Spark [*Spark*] can load data from a variety of external sources, including JDBC data sources. In this example, we use the JDBC client to load the table *HDF5.VRTL.Attributes* from an HDF5/ODBC server at IP address *10.10.10.120* into an *org.apache.spark.sql.DataFrame*. When invoking the the Spark (Scala) shell, we need to pass the JDBC connector JAR and class path as follows:

```
bin/spark-shell --driver-class-path SimbaJDBCClient41.jar --jars SimbaJDBCClient41.jar
```

The Java class implementing the JDBC driver is called:

```
jdbc:simba://10.10.10.120:12345
```

```
val jdbcDF = spark.read.format("jdbc").option("driver", \
"com.simba.client.core.jdbc41.JDBC41Driver").option("url", \
"jdbc:simba://10.10.10.120:12345").option("dbtable", \
"HDF5.VRTIL.Attributes").load()
```

```
bin/spark-shell --driver-class-path SimbaJDBCClient41.jar \
  --jars SimbaJDBCClient41.jar
```

...
Welcome to

[illegible]

```
Using Scala version 2.11.8 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_121)
Type in expressions to have them evaluated.
Type :help for more information.
```

```
scala> val jdbcDF = spark.read.format("jdbc").option("driver", \
  "com.simba.client.core.jdbc41.JDBC41Driver").option("url", \
  "jdbc:simba://10.10.10.120:12345").option("dbtable", \
  "HDF5.VRTL.Attributes").load()
jdbcDF: org.apache.spark.sql.DataFrame = [ObjectPath: string, \
  AttributeName: string ... 1 more field]
```

```
scala> jdbcDF.show()
```

ObjectPath	AttributeName	AttributeValue
/hdfeos.he5/HDFEO...	InstrumentName	HIRDLIS
/hdfeos.he5/HDFEO...	ProcessLevel	L2
/hdfeos.he5/HDFEO...	PGEVersion	V1.0.0
/hdfeos.he5/HDFEO...	HIRDLISFileType	HIRDLIS2
/hdfeos.he5/HDFEO...	GranuleMonth	1
/hdfeos.he5/HDFEO...	GranuleDay	1
/hdfeos.he5/HDFEO...	GranuleYear	2008
/hdfeos.he5/HDFEO...	TAI93At0zOfGranule	473299206
/hdfeos.he5/HDFEO...	VerticalCoordinate	Pressure
/hdfeos.he5/HDFEO...	_FillValue	-99
/hdfeos.he5/HDFEO...	MissingValue	-99
/hdfeos.he5/HDFEO...	Title	10.8 Micron Cloud...
/hdfeos.he5/HDFEO...	Units	0=no contaminatio...
/hdfeos.he5/HDFEO...	UniqueFieldDefini...	HIRDLIS-Specific
/hdfeos.he5/HDFEO...	_FillValue	-999
/hdfeos.he5/HDFEO...	MissingValue	-999
/hdfeos.he5/HDFEO...	Title	10.8 Micron Exti...
/hdfeos.he5/HDFEO...	Units	1/km

```
|/hdfeos.he5/HDFEO...|UniqueFieldDefini...|      HIRDLIS-Specific|
|/hdfeos.he5/HDFEO...|      Units|      NoUnits|
+-----+-----+-----+
only showing top 20 rows

scala> jdbcDF.count()
res0: Long = 574
```

A more complex query can be “dressed” as a subquery:

```
(SELECT * FROM HDF5.\"/tickdata.h5/\".\"22-09-2011\" WHERE Bid > 0.89) ticks
```

```
scala> val query = "(SELECT * FROM HDF5.\"/tickdata.h5/\".\"22-09-2011\" \
WHERE Bid > 0.89) ticks"

query: String = (SELECT * FROM HDF5.\"/tickdata.h5/\".\"22-09-2011\" \
WHERE Bid > 0.89) ticks

scala> val jdbcDF = spark.read.format("jdbc").option("driver", \
"com.simba.client.core.jdbc41.JDBC41Driver").option("url", \
"jdbc:simba://10.10.10.120:12345").option("dbtable", query).load()
jdbcDF: org.apache.spark.sql.DataFrame = [ROWID: decimal(20,0), \
Time: string ... 2 more fields]

scala> jdbcDF.show()
+-----+-----+-----+-----+
|ROWID|      Time|      Bid|      Ask|
+-----+-----+-----+-----+
|  47|00:12:38:100000|0.8901000022888184|0.8907999992370605|
|  49|00:13:12:900000|0.8901000022888184|0.8907999992370605|
|  51|00:13:34:200000|0.8901000022888184|0.8907999992370605|
|  52|00:13:38:300000|0.8902000188827515|0.8909000158309937|
|  53|00:13:38:700000|0.8901000022888184|0.8907999992370605|
|  54|00:13:39:700000|0.8902000188827515|0.8909000158309937|
|  55|00:13:41:400000|0.8902999758720398| 0.890999972820282|
|  56|00:13:46:200000|0.8903999924659729|0.8910999894142151|
|  57|00:13:47:200000|0.8902999758720398| 0.890999972820282|
|  58|00:14:22:300000|0.8902000188827515|0.8909000158309937|
|  59|00:14:23:000000|0.8901000022888184|0.8907999992370605|
|  60|00:14:24:700000|0.8902000188827515|0.8909000158309937|
|  61|00:14:40:500000|0.8901000022888184|0.8907999992370605|
|  62|00:15:07:000000|0.8902000188827515|0.8909000158309937|
|  63|00:15:20:800000|0.8902999758720398| 0.890999972820282|
|  64|00:15:52:700000|0.8902000188827515|0.8909000158309937|
|  65|00:15:53:100000|0.8902999758720398| 0.890999972820282|
|  66|00:15:53:800000|0.8902000188827515|0.8909000158309937|
|  67|00:15:55:500000|0.8902999758720398| 0.890999972820282|
|  68|00:16:07:100000|0.8902000188827515|0.8909000158309937|
+-----+-----+-----+-----+
only showing top 20 rows

scala> jdbcDF.count()
res4: Long = 6030
```

THE SELECT STATEMENT

The `SELECT` statements supported by the HDF5 ODBC connector follow the same SQL grammar as the statements supported by other connectors for other ODBC data sources. For example, the HDF5 ODBC connector recognizes the following `SELECT` statement clauses:

- `FROM`
- `WHERE`
- `GROUP BY`
- `HAVING`
- `UNION`
- `ORDER BY`

The `WHERE` clause predicates supported by ODBC SQL are the “usual suspects”:

- Relational predicates (comparisons `<`, `>`, `<=`, `>=`, `=`, `<>`)
- `BETWEEN`
- `LIKE`
- `IN`
- `EXISTS`
- `NULL`

However, to make the most effective use of the HDF5 ODBC connector, it is important to understand how the data stored in HDF5 files is mapped onto relational tables, and how certain HDF5 constructs that do not have a direct counterpart in the relational model are “translated.” In this chapter, the structure of table identifiers is described, followed by a discussion of the relationship between HDF5 datatypes and SQL data types. Finally, the representation of certain metadata is described.

4.1 Catalogs, Schemas, and Tables

While there seems to be a natural correspondence between tables and HDF5 datasets, that’s about where the similarity between databases and HDF5 files ends. The HDF5 data model is more complex than the relational model, and there are at least two categories of approaches to the problem of mapping between the models.

1. *Dogmatic approach*: Prescribe a mapping that will satisfy the needs of a majority of users.
2. *Pragmatic approach*: Create facilities that allow the user to control the mapping.

In this version of the connector, the first approach was adopted and a mapping prescribed. In future releases, other options might be available.

An HDF5 dataset is usually referred to by an HDF5 *path name*, e.g.,

```
/hdfeos.he5/HDFEOS/SWATHS/HIRDLS/Data Fields/O3
```

A *table identifier* is a three part construct, which consists of the catalog, schema, and table names (separated by dots).

```
CATALOG.SCHEMA.TABLE
```

The mapping implemented in this version of the connector can be summarized as follows:

SQL	HDF5
Catalog name	HDF5
Schema name	HDF5 “parent” group path name
Table name	HDF5 dataset (link) name

In (over-simplified) words:

- The catalog name is always HDF5.
- The schema name is the group path name of the dataset’s “parent group.”
- The table name is the “dataset name.”

In many cases, this mental model is sufficient. What’s problematic is that the terms in quotes are not as well-defined as they might appear. Before getting into the fine-print, let’s see an example!

Example

The table identifier of the dataset with path name

```
/hdfeos.he5/HDFEOS/SWATHS/HIRDLS/Data Fields/O3
```

is

```
HDF5."/hdfeos.he5/HDFEOS/SWATHS/HIRDLS/Data Fields/"O3
```

Note: Since the schema name in the previous example contains non-alphanumeric characters, such as slashes (/) and spaces, it must be included in quotation marks (").

More precisely, the mapping between HDF5 datasets and tables is as follows:

- For all HDF5-backed ODBC data sources, there is only one *catalog name*: HDF5.
- A *schema name* of a table is an absolute HDF5 path name of an HDF5 group that contains a link to the corresponding dataset, followed by a trailing slash.
- A *table name* is a link name of the dataset in the group that can be reached by following the HDF5 path name in the table’s schema name.

The previous formulations sound so vague because of the heavy use of indefinite articles. *Confusing things can happen*: the same HDF5 dataset can be a member of multiple groups, under different names, and it can even be a member of the same group appearing multiple times (under different names). Under those circumstances neither the concept of ‘parent’, nor the concept of a ‘dataset name’ are very intuitive.

4.2 HDF5 Datatypes and SQL Data Types

HDF5 supports a great variety of pre- and user-defined scalar and non-scalar types. The HDF5 ODBC connector supports three families of scalar types (integers, floating-point numbers, strings) and one class of non-scalar types, so-called *compounds*. The supported *scalar* HDF5 datatypes and their SQL data type counterparts are summarized in the table below:

HDF5 Datatype	SQL Data Type
8-bit (un)signed	SQL_TINYINT (un)signed
16-bit (un)signed	SQL_SMALLINT (un)signed
32-bit (un)signed	SQL_INTEGER (un)signed
64-bit (un)signed	SQL_BIGINT (un)signed
32-bit IEEE floating-point	SQL_REAL
64-bit IEEE floating-point	SQL_DOUBLE
Fixed-length ASCII strings	SQL_CHAR (N)
Variable-length ASCII strings	SQL_VARCHAR
Fixed-length Unicode strings	SQL_NCHAR (N)
Variable-length Unicode strings	SQL_NVARCHAR

Note: HDF5 stores Unicode strings as UTF-8 encoded byte sequences. A fixed-length HDF5 string datatype has a finite byte capacity *N*. The UTF-8 encoding of a single Unicode code point requires between 1 and 4 bytes. Hence *N* is an upper bound on the number of Unicode code points. Generally, there will be less than *N* Unicode code points in an HDF5 fixed-length Unicode string of *byte-capacity* *N*.

In addition, the connector supports HDF5 compound datatypes whose fields are among the supported scalar HDF5 datatypes. The column names and types of a table backed by an HDF5 dataset of a supported compound datatype are inferred from the fields of the compound datatype.

Note: The connector currently does *not* support the following HDF5 datatypes:

- User-defined HDF5 atomic datatypes
- HDF5 array datatypes
- HDF5 bfields
- HDF5 compound datatypes with fields of an unsupported datatype
- HDF5 enumerated datatypes
- HDF5 opaque datatypes
- HDF5 reference datatypes
- HDF5 variable-length sequence types

If the HDF5 ODBC connector encounters a dataset in an HDF5 file with an unsupported element type it is simply skipped and excluded from the table listing.

4.3 The ROWID Pseudo-Column

HDF5 datasets are (logically) dense, multi-dimensional arrays of a fixed element type. Each dataset element has a logical position or multi-dimensional index in the underlying rectilinear lattice. HDF5 supports several methods to access individual elements or subsets at random. Neither array indexes nor random access are relational concepts. A row in a table is referred to by its primary or candidate key, which consists of a subset of row attributes (columns). This, in turn, is an “alien concept” to HDF5, where there is no concept of a “dataset element key” or a default *data*

access path via the value of dataset elements. The array index might be viewed as an implicit primary key, but its usefulness is limited.

Many database systems provide a proprietary ROWID or internal ID extension. The current version of the HDF5 ODBC connector offers a similar capability. When querying an HDF5 dataset-backed table, the connector returns a column named ROWID, which corresponds to the *C-linearized* position of a dataset element. For a one-dimensional dataset this is just the array index of the dataset element. For an r -dimensional ($r \geq 2$) dataset with dimensions D_0, D_1, \dots, D_{r-1} , the ROWID of an r -index $(i_0, i_1, \dots, i_{r-1})$, $0 \leq i_k < D_k$, $0 \leq k < r < 32$ is obtained as follows:

$$(\dots((i_0 \cdot D_1 + i_1) \cdot D_2 + i_2) \dots + i_{r-2}) \cdot D_{r-1} + i_{r-1}$$

For two-dimensional datasets, this order is the familiar *row-major* or C-order. For example, in a 5×7 dataset, an element (i_0, i_1) will be assigned index

$$i_0 \cdot 7 + i_1$$

Hence, the elements of the first row will have indices $0, \dots, 6$. The elements of the last row will have indices $28, \dots, 34$.

This mapping (and its inverse) can be used, for example, to JOIN datasets along different dimensions.

Example

The ROWID can be used to join the three datasets

```
/global-earthquakes.h5/data/double1 [4, 59209]
/global-earthquakes.h5/data/int0      [3, 59209]
/global-earthquakes.h5/data/int2      [59209]
```

from the *HDF5Test* data source as follows:

```
SELECT A.Value AS Year, B.Value AS Month, C.Value AS Day,
       D.Value AS UTC, E.Value AS Lat, F.Value AS Lon,
       G.Value AS Magnitude, H.Value AS Depth
FROM   HDF5."/global-earthquakes.h5/data/"..int0 AS A,
       HDF5."/global-earthquakes.h5/data/"..int0 AS B,
       HDF5."/global-earthquakes.h5/data/"..int0 AS C,
       HDF5."/global-earthquakes.h5/data/"..double1 AS D,
       HDF5."/global-earthquakes.h5/data/"..double1 AS E,
       HDF5."/global-earthquakes.h5/data/"..double1 AS F,
       HDF5."/global-earthquakes.h5/data/"..double1 AS G,
       HDF5."/global-earthquakes.h5/data/"..int2 AS H
WHERE  (A.ROWID-MOD(A.ROWID,59209)) = 0
AND    (B.ROWID-MOD(B.ROWID,59209)) = 59209
AND    (C.ROWID-MOD(C.ROWID,59209)) = 2*59209
AND    (D.ROWID-MOD(D.ROWID,59209)) = 0
AND    (E.ROWID-MOD(E.ROWID,59209)) = 59209
AND    (F.ROWID-MOD(F.ROWID,59209)) = 2*59209
AND    (G.ROWID-MOD(G.ROWID,59209)) = 3*59209
AND    MOD(A.ROWID,59209) = MOD(B.ROWID,59209)
AND    MOD(A.ROWID,59209) = MOD(C.ROWID,59209)
AND    MOD(A.ROWID,59209) = MOD(D.ROWID,59209)
AND    MOD(A.ROWID,59209) = MOD(E.ROWID,59209)
AND    MOD(A.ROWID,59209) = MOD(F.ROWID,59209)
AND    MOD(A.ROWID,59209) = MOD(G.ROWID,59209)
AND    MOD(A.ROWID,59209) = H.ROWID
```

Warning: The previous example works correctly when the so-called *2D-mode*, described in the next section, is disabled in the connection string.

4.4 2D-Mode

The usability of many ODBC clients can be improved greatly for *two-dimensional* datasets by treating the second dimension as “columnar dimension.” This behavior is referred to as the *2D-MODE* of connector operation. In this mode, a table backed by a 2D dataset has as many columns as the extent of the dataset’s second dimension. The column names are of the form Column N where N is the index of the 1-based second dimension.

Attention: 2D-Mode is enabled by default. To disable, pass the 2D_MODE=OFF property in the connection string.

Note: The maximum number of columns supported in 2D-mode is 16,384.

Note: The ROWID pseudo-column is *suppressed* in 2D-mode.

Below a screenshot of the HDF5 dataset

/hdfEOS.he5/HDFEOS/SWATHS/HIRDLS/Data Fields/O3

from the *HDF5Test* data source, loaded into an Excel workbook, is shown.

	Column 7	Column 8	Column 9	Column 10	Column 11	Column 12	Column 13	Column 14	Column 15	Column 16
1	-999	-999	6.19468E-08	6.84297E-08	7.70715E-08	8.69493E-08	9.83223E-08	1.07933E-07	1.16848E-07	1.21313
2	-999	-999	6.39962E-08	7.1321E-08	8.08336E-08	9.20513E-08	1.05064E-07	1.1735E-07	1.28913E-07	1.35603
3	-999	-999	6.0418E-08	6.69974E-08	7.57095E-08	8.55871E-08	9.65807E-08	1.05413E-07	1.13007E-07	1.15318
4	-999	-999	5.40266E-08	5.93242E-08	6.598E-08	7.36879E-08	8.16957E-08	8.82014E-08	9.29075E-08	9.25468
5	-999	-999	5.38155E-08	5.90769E-08	6.58222E-08	7.328E-08	8.0851E-08	8.61163E-08	8.9909E-08	8.79555
6	-999	-999	6.75516E-08	7.69609E-08	8.87949E-08	1.03613E-07	1.20328E-07	1.37812E-07	1.52966E-07	1.63916
7	-999	-999	-999	-999	-999	-999	-999	-999	-999	-999
8	-999	-999	8.68752E-08	1.02414E-07	1.22972E-07	1.49278E-07	1.81062E-07	2.16497E-07	2.52848E-07	2.85925
9	-999	-999	7.37395E-08	8.51512E-08	9.98841E-08	1.18188E-07	1.38751E-07	1.59852E-07	1.79169E-07	1.92485
10	-999	-999	7.45622E-08	8.64498E-08	1.01476E-07	1.20441E-07	1.41818E-07	1.6485E-07	1.85324E-07	2.01606
11	-999	-999	7.76911E-08	9.03537E-08	1.06765E-07	1.26637E-07	1.48873E-07	1.7116E-07	1.92253E-07	2.06373
12	-999	-999	9.81663E-08	1.18479E-07	1.45143E-07	1.79872E-07	2.22378E-07	2.72068E-07	3.24776E-07	3.77498
13	-999	-999	-999	-999	-999	-999	-999	-999	-999	-999
14	-999	-999	9.69679E-08	1.17198E-07	1.4341E-07	1.77216E-07	2.17837E-07	2.64937E-07	3.14004E-07	3.62106
15	-999	-999	1.04336E-07	1.27275E-07	1.57708E-07	1.96774E-07	2.4415E-07	2.98538E-07	3.57986E-07	4.16049
16	-999	-999	1.09083E-07	1.34435E-07	1.67832E-07	2.11793E-07	2.65899E-07	3.30841E-07	4.02297E-07	4.78779
17	-999	-999	1.03898E-07	1.26737E-07	1.56864E-07	1.95509E-07	2.41615E-07	2.94559E-07	3.51716E-07	4.08038
18	-999	-999	9.01177E-08	1.06347E-07	1.31144E-07	1.63253E-07	2.06093E-07	2.58705E-07	3.22911E-07	3.92213E-07
19	-999	-999	9.87108E-08	1.1962E-07	1.46874E-07	1.81166E-07	2.2067E-07	2.64541E-07	3.10439E-07	3.53156
20	-999	-999	1.11043E-07	1.36438E-07	1.7014E-07	2.1315E-07	2.6436E-07	3.23306E-07	3.87974E-07	4.53924
21	-999	-999	1.08768E-07	1.33251E-07	1.65647E-07	2.06294E-07	2.54065E-07	3.07462E-07	3.6577E-07	4.22752
22	-999	-999	1.15149E-07	1.42193E-07	1.78138E-07	2.2391E-07	2.78442E-07	3.41195E-07	4.10797E-07	4.83
23	-999	-999	1.06693E-07	1.3051E-07	1.61686E-07	2.00454E-07	2.45209E-07	2.94448E-07	3.47435E-07	3.98406
24	-999	-999	1.19224E-07	1.48143E-07	1.86393E-07	2.35935E-07	2.9485E-07	3.64669E-07	4.41954E-07	5.2661
25	-999	-999	1.08566E-07	1.32017E-07	1.62368E-07	1.97639E-07	2.38117E-07	2.77898E-07	3.20595E-07	3.59163
26	-999	-999	9.92676E-08	1.19946E-07	1.46515E-07	1.7839E-07	2.13337E-07	2.49772E-07	2.87094E-07	3.2021
27	-999	-999	1.08073E-07	1.32161E-07	1.63729E-07	2.02216E-07	2.45669E-07	2.92443E-07	3.426E-07	3.90576

If 2D-mode is disabled (via 2D_MODE=OFF in the connection string), the same dataset appears as follows:

ROWID	Value
0	-999
1	-999
2	-999
3	-999
4	-999
5	-999
6	-999
7	-999
8	6.19468E-08
9	6.84297E-08
10	7.70715E-08
11	8.69493E-08
12	9.83223E-08
13	1.07933E-07
14	1.16848E-07
15	1.21313E-07
16	1.36788E-07
17	1.71215E-07
18	3.0801E-07
19	5.88886E-07
20	5.83084E-07
21	5.43952E-07
22	4.66643E-07
23	4.70833E-07

Notice the return of the ROWID pseudo-column.

4.5 The HDF5.VRTL.Attributes Table

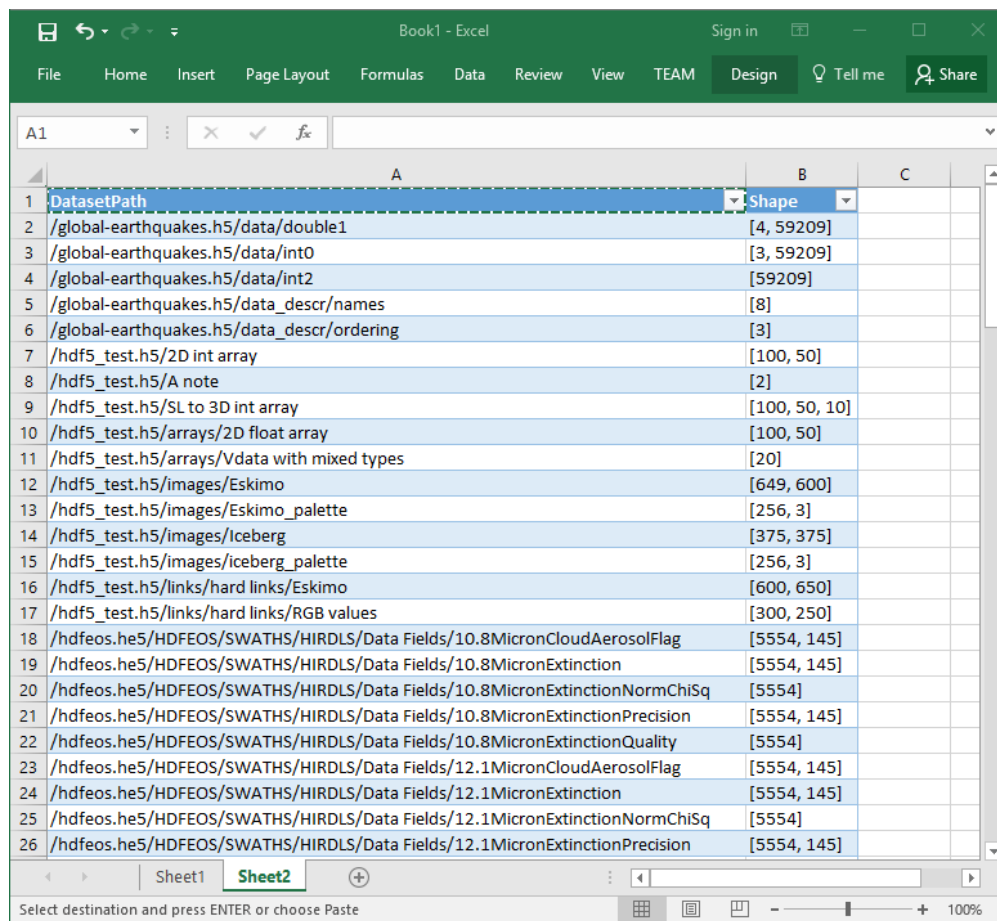
HDF5 attributes have no counterpart in the relational model. For convenience, the HDF5 ODBC connector exposes the attributes of all objects through a “virtual” table with identifier `HDF5.VRTL.Attributes`. It has three string columns, *ObjectPath*, *AttributeName*, and *AttributeValue*.

ObjectPath	AttributeName	AttributeValue
/global-earthquakes.h5	name	global-earthquakes
/global-earthquakes.h5	mldata	0
/global-earthquakes.h5	comment	CSV
/hdf5_test.h5/arrays/vdata with mixed types	HDF4_OBJECT_TYPE	Vdata
/hdf5_test.h5/arrays/vdata with mixed types	HDF4_OBJECT_NAME	Vdata with mixed types
/hdf5_test.h5/arrays/vdata with mixed types	HDF4_REF_NUM	46
/hdf5_test.h5/images/Eskimo	CLASS	IMAGE
/hdf5_test.h5/images/Eskimo	IMAGE_VERSION	1
/hdf5_test.h5/images/Eskimo	IMAGE_SUBCLASS	IMAGE_INDEXED
/hdf5_test.h5/images/Eskimo	IMAGE_COLORMODEL	RGB
/hdf5_test.h5/images/Eskimo	IMAGE_TRANSPARENCY	255
/hdf5_test.h5/images/Eskimo	PALETTE	NIL
/hdf5_test.h5/images/Eskimo_palette	CLASS	PALETTE
/hdf5_test.h5/images/Eskimo_palette	PAL_VERSION	1
/hdf5_test.h5/images/Eskimo_palette	PAL_COLORMODEL	RGB
/hdf5_test.h5/images/Eskimo_palette	PAL_TYPE	DIRECTINDEX
/hdf5_test.h5/images/Iceberg	CLASS	IMAGE
/hdf5_test.h5/images/Iceberg	IMAGE_VERSION	1
/hdf5_test.h5/images/Iceberg	IMAGE_SUBCLASS	IMAGE_INDEXED
/hdf5_test.h5/images/Iceberg	IMAGE_COLORMODEL	RGB
/hdf5_test.h5/images/Iceberg	PALETTE	NIL
/hdf5_test.h5/images/Iceberg_palette	CLASS	PALETTE
/hdf5_test.h5/images/Iceberg_palette	PAL_VERSION	1
/hdf5_test.h5/images/Iceberg_palette	PAL_COLORMODEL	RGB
/hdf5_test.h5/images/Iceberg_palette	PAL_TYPE	DIRECTINDEX

Note: The attributes table is created dynamically by visiting all HDF5 objects in a data source (which might be backed by one or more HDF5 files). Each object is visited exactly once, cycles and multiple paths are ignored, and the object appears in the table exactly once, under a single HDF5 path name. If there are multiple paths to an object, no guarantees can be given as to which particular path will be reported.

4.6 The HDF5.VRTL.Shapes Table

With the exception of two-dimensional datasets and 2D-mode, the shape information for all higher-dimensional datasets would be lost under the C-linearization model. To retain the information about the shape of an HDF5 dataset the HDF5 ODBC connector exposes a second “virtual” table with the identifier `HDF5.VRTL.Shapes`. It has two string columns, *DatasetPath* and *Shape*, where *Shape* is a rendering of a dataset’s shape as a JSON array (or Python list). For scalar and null datasets, the *Shape* is reported as `SCALAR` or `NULL`, respectively.



DatasetPath	Shape
/global-earthquakes.h5/data/double1	[4, 59209]
/global-earthquakes.h5/data/int0	[3, 59209]
/global-earthquakes.h5/data/int2	[59209]
/global-earthquakes.h5/data_descr/names	[8]
/global-earthquakes.h5/data_descr/ordering	[3]
/hdf5_test.h5/2D int array	[100, 50]
/hdf5_test.h5/A note	[2]
/hdf5_test.h5/SL to 3D int array	[100, 50, 10]
/hdf5_test.h5/arrays/2D float array	[100, 50]
/hdf5_test.h5/arrays/Vdata with mixed types	[20]
/hdf5_test.h5/images/Eskimo	[649, 600]
/hdf5_test.h5/images/Eskimo_palette	[256, 3]
/hdf5_test.h5/images/Iceberg	[375, 375]
/hdf5_test.h5/images/iceberg_palette	[256, 3]
/hdf5_test.h5/links/hard links/Eskimo	[600, 650]
/hdf5_test.h5/links/hard links/RGB values	[300, 250]
/hdfeos.he5/HDFEOS/SWATHS/HIRDLS/Data Fields/10.8MicronCloudAerosolFlag	[5554, 145]
/hdfeos.he5/HDFEOS/SWATHS/HIRDLS/Data Fields/10.8MicronExtinction	[5554, 145]
/hdfeos.he5/HDFEOS/SWATHS/HIRDLS/Data Fields/10.8MicronExtinctionNormChiSq	[5554]
/hdfeos.he5/HDFEOS/SWATHS/HIRDLS/Data Fields/10.8MicronExtinctionPrecision	[5554, 145]
/hdfeos.he5/HDFEOS/SWATHS/HIRDLS/Data Fields/10.8MicronExtinctionQuality	[5554]
/hdfeos.he5/HDFEOS/SWATHS/HIRDLS/Data Fields/12.1MicronCloudAerosolFlag	[5554, 145]
/hdfeos.he5/HDFEOS/SWATHS/HIRDLS/Data Fields/12.1MicronExtinction	[5554, 145]
/hdfeos.he5/HDFEOS/SWATHS/HIRDLS/Data Fields/12.1MicronExtinctionNormChiSq	[5554]
/hdfeos.he5/HDFEOS/SWATHS/HIRDLS/Data Fields/12.1MicronExtinctionPrecision	[5554, 145]

Note: The shapes table is created dynamically by visiting all HDF5 datasets in a data source (which might be backed by one or more HDF5 files). Each dataset is visited exactly once, cycles and multiple paths are ignored, and the dataset appears in the table exactly once, under a single HDF5 path name. If there are multiple paths to a dataset, no guarantees can be given as to which particular path will be reported.

5.1 Platforms and System Requirements

Minimum Hardware Requirements

- 1 GB of free disk space
- 1 GB of RAM

Software Requirements

Platform	Versions	Bits
Windows	7 SP1, 8, 8.1, 10; Server 2008 R2 SP1, 2012, 2012 R2	32, 64
Linux	CentOS/RHEL 5, 6, 7; SLES 11, 12; Ubuntu 12.04, 14.04, 16.04; Debian 7, 8	32, 64

5.2 ODBC Conformance Level

The HDF5 ODBC Connector supports the full core-level ODBC 3.80. It supports most of the Level 1 and Level 2 API.

5.3 Supported ODBC/SQL Functions

5.3.1 Explicit Conversion Functions

- CONVERT
- CAST

5.3.2 String Functions

- ASCII
- CHAR
- CONCAT
- INSERT

- LCASE
- LEFT
- LENGTH
- LOCATE
- LTRIM
- REPEAT
- REPLACE
- RIGHT
- RTRIM
- SOUNDEX
- SPACE
- SUBSTRING
- UCASE

5.3.3 Numeric Functions

- ABS
- ACOS
- ASIN
- ATAN
- ATAN2
- CEILING
- COS
- COT
- DEGREES
- EXP
- FLOOR
- LOG
- LOG10
- MOD
- PI
- POWER
- RADIANS
- RAND
- ROUND
- SIGN
- SIN

- SQRT
- TAN
- TRUNCATE

5.3.4 Time, Date, and Interval Functions

- CURDATE
- CURTIME
- CURRENT_DATE
- CURRENT_TIME
- CURRENT_TIME (time precision)
- CURRENT_TIMESTAMP
- CURRENT_TIMESTAMP (time precision)
- DAYNAME
- DAYOFMONTH
- DAYOFWEEK
- DAYOFYEAR
- HOUR
- MINUTE
- MONTH
- MONTHNAME
- NOW
- QUARTER
- SECOND
- TIMESTAMPADD
- TIMESTAMPDIFF
- WEEK
- YEAR

5.3.5 System Functions

- DATABASE
- IFNULL
- USER

5.3.6 Aggregate Functions

- AVG
- COUNT
- MAX
- MIN
- STDDEV
- STDDEV_POP
- SUM
- VAR
- VAR_POP

5.4 Error Messages and Logging

Although no effort was spared to make the software as robust as possible, users might experience unexpected behavior. Contacting support at help@hdfgroup.org is the first step toward a resolution, but additional information might be required to diagnose the problem. The HDF5 ODBC Connector can be configured to produce additional diagnostic information, however, this feature is disabled by default. In this section, the process of how to enable and configure this capability is described.

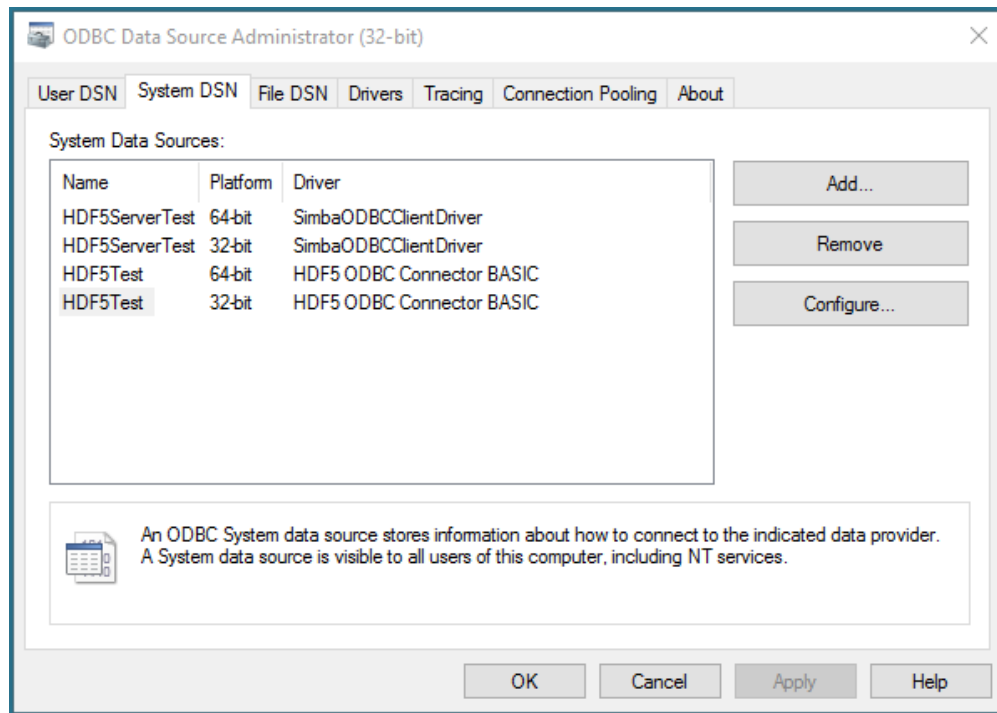
There are two configurable parameters:

1. The *destination directory* for log files.
2. The *verbosity level* of the log information.

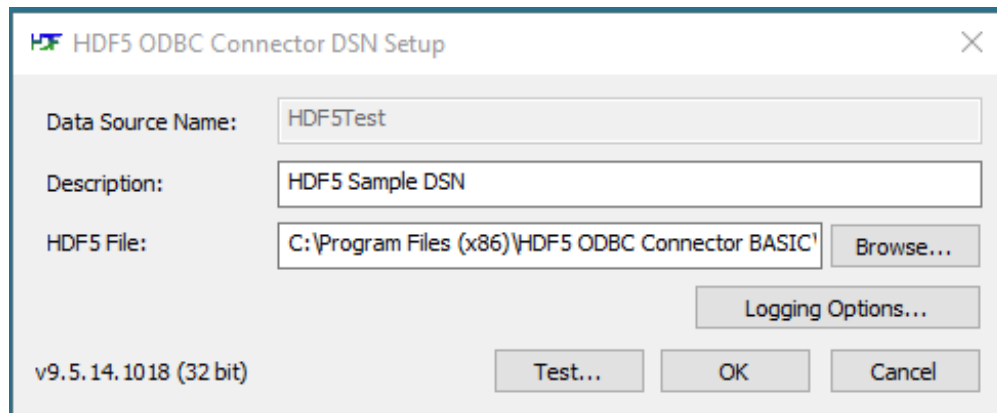
Both can be configured through the *ODBC Data Source Administrator*.

Note: The procedure is the same for System and User Data Sources, however, administrative privileges might be required to configure System Data Sources.

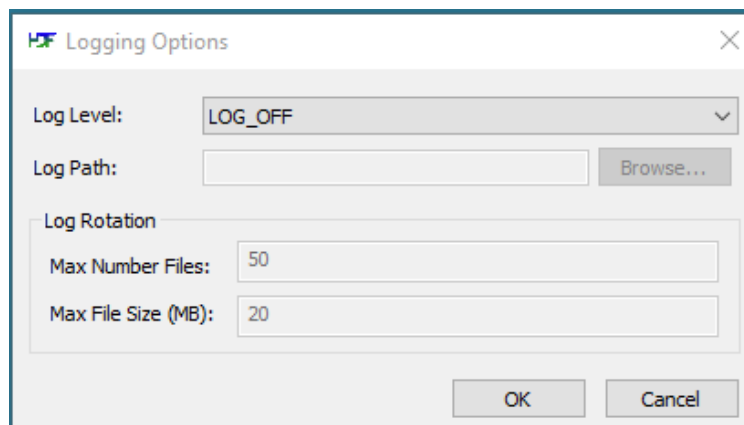
In this example, the *HDF5Test* data source is used to illustrate the steps.



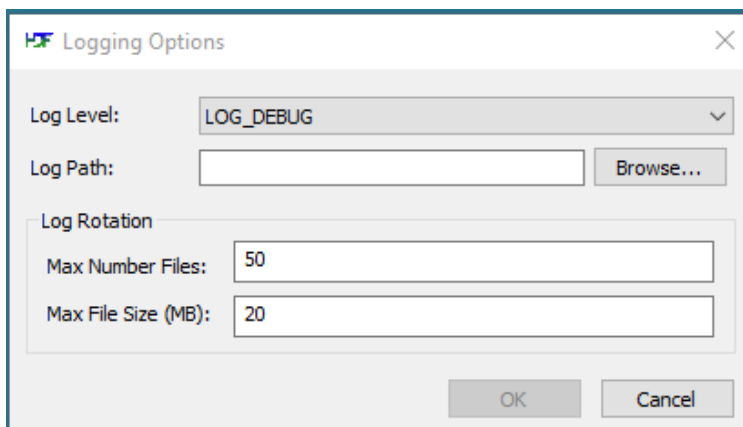
Select *HDF5Test* from the list of data sources and click **Configure**. This will open the *HDF5 ODBC Connector DSN Setup* dialog.



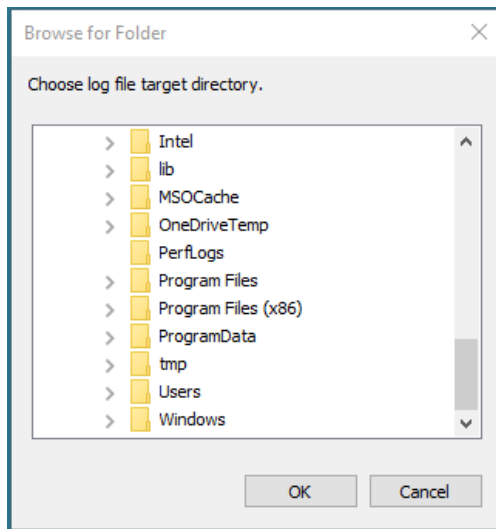
Click **Logging Options**, which will open the *Logging Options* dialog.



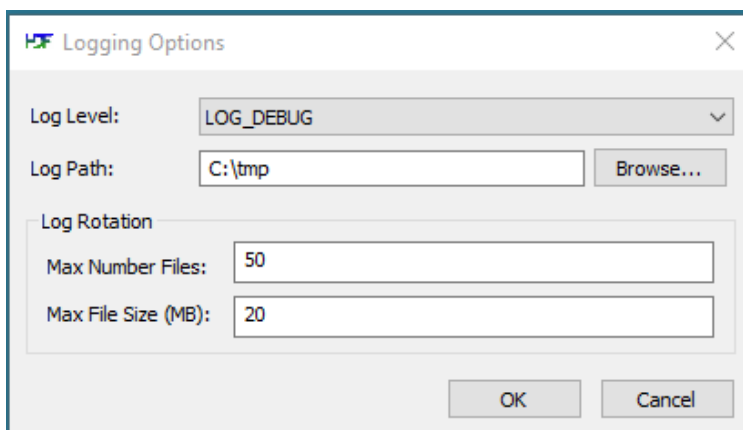
The default *Log Level* is LOG_OFF with an empty *Log Path*. Unless otherwise instructed by our support staff, please change the logging level by selecting the LOG_DEBUG item from the *Log Level* drop-down list.



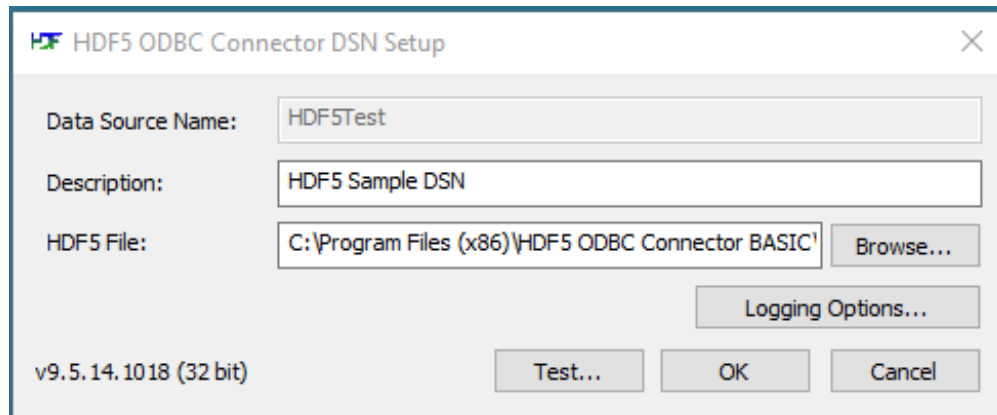
Click the **Browse** button to open a directory selection dialog. Please select a directory on the system to which you have *write access* and click **OK**.



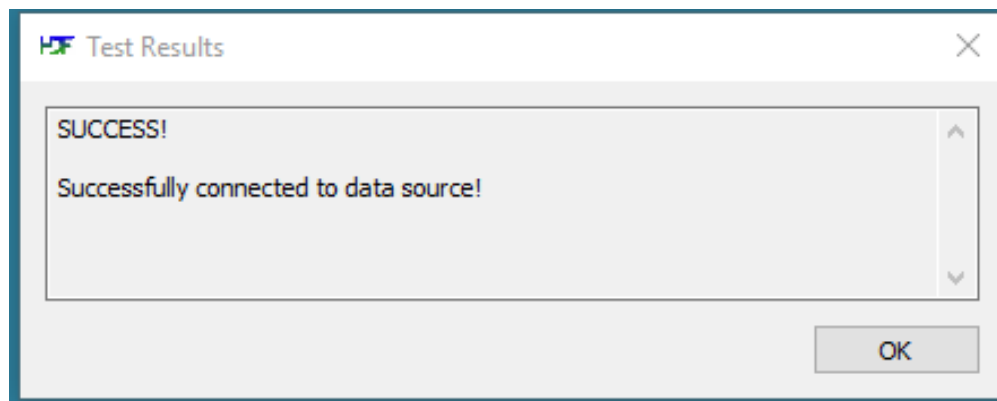
The path you’ve selected in the previous step should show up in the *Log Path* text box.



Click **OK** to return to the *HDF5 ODBC Connector DSN Setup* dialog.



Optionally, click **Test** to verify the configuration.



Important: The name of the active log file is `hdf5_odbc_client.log`. If you are trying to enable logging for different data sources at the same time, make sure you choose different logging paths.

BIBLIOGRAPHY

- [HDF5] <https://www.hdfgroup.org/HDF5/>
- [pyodbc] Python ODBC bridge. <http://mkleehammer.github.io/pyodbc/>
- [R] The R Project for Statistical Computing. <https://www.r-project.org/>
- [Spark] Apache Spark. <http://spark.apache.org/>
- [Szip] https://www.hdfgroup.org/doc_resource/SZIP/
- [Zlib] <http://www.zlib.net/>