Independent Submission
Internet-Draft
Intended status: Standards Track
Expires: April 11, 2015

P. Lambert, Ed.
Dendory Networks

October 8, 2014

ODBC URI Scheme
draft 00

Abstract

This Internet-Draft document specifies a new URI scheme [RFC2396] for data access using the Open Database Connectivity [ODBC] standard API, in order to access data in databases from a web browser or any other client choosing to implement this method of showing data.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Uniform Resource Identifiers (URI) have become the standard way to access resources over a network, and as such the general public knows how to access web pages, email addresses and more through this standard scheme. However, there is no standard way to access data, typically kept inside of databases, via a standard client such as a web browser. As a result, we have many different ways of accessing data from a database, including server processes exporting the information to a web format, browser plugins that access the database through a secondary channel (such as ODBC) and then display the results, dedicated client software which may also have an export function to HTML format, and so on.

The Open Database Connectivity (ODBC) is already a standard API used by many database engines and available on most if not all operating systems and most programming languages. As a result, many client software access databases through it, but the scheme (or connection string) used varies by application, programming language and operating system. A standard URI can leverage this existing API and provide a common method to access this type of data.

1.1. Why ODBC?

ODBC is a format introduced by Microsoft and others in the early 1990s and that has become the de facto standard way to access databases. It is available on most systems and through most programming and scripting languages. As such, it makes sense to utilize it in order to allow other types of clients (such as web browsers, email clients, etc) to access data in databases using existing API calls.

ODBC provides independent database engine access through the use of Drivers (sometimes called Connectors) and Data Source Names (DSN). A driver is installed on the local machine to access a particular type of engine, and the DSN specifies things such as the server hostname, port, login credential, database name, and so on. An operating system level driver can be installed for most common database engines such as Microsoft SQL Server, MySQL, SQLite, Oracle Server, and many more. The DSN varies by driver, but is typically a single string of text such as:

```
SERVER=db.example.com;UID=MyUser;PWD=MyPassword;ENCRYPT=1
```

Since ODBC is so common and already defines a set of standards, it makes sense to leverage it in order to create a standard URI scheme.

1.2. Benefits

Having a standard URI for database connections would bring several benefits, above and beyond what ODBC itself provides to programmers and other IT professionals:

- o Simple data viewing interfaces could be built into web browsers to be accessed directly by users, without the need of a plugin or server process converting the data.
- o Web applications and scripts would be easier to write, able to leverage the data inside a database through JavaScript and other client side scripts, again without the need for third party applications.
- o Future client applications could be constructed based on this standard way to access data, such as adding the ability to edit or insert data through the location bar in a browser, without the need for a web server or other server process to be involved and interpret the data.
- o It would reduce the ever-growing amount of different ways data is accessed online, making it easier to have a common interface to data.

1.3. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2. Syntax

The basic syntax for the ODBC URI scheme must be one of the following:

```
odbc:[driver]//[DSN]/[database]/[tables]?[query]

odbc:[driver]//[user]:[password]@[host]:[port]/[database]/[tables]?[query]
```

In this syntax, odbc is a constant string starting the URI. [driver] is the name of the installed ODBC driver. This must be the exact name defined by the driver installed on the local system, such as "Microsoft ODBC Driver 11 for SQL Server", "MySQL ODBC 5.1 Driver", or a shortened name such as "SQL Server" and "mysql", without quotes. The [DSN] part is a freeform text string in the format shown in Section 1, as expected by the driver. Alternatively, a standard way to specify [user], [password], [host] and [port] can be used. Of those four options, only [host] is mandatory.

The next part specifies the [database] and [tables] in a way similar to how users access web pages. The first text after the slash is the database name, and any furter slashes specify tables. Finally, a [query] can optionally be specified in order to pass on queries to the database, such as SELECT, ALTER or INPUT, as specified in the ISO/IEC 9075-3:2003 [CLI] specification.

For example, to access the dbo.default table in the master database on the db.example.com host as user root using the SQL Server driver, the URI should be:

```
odbc:SQL Server//root@db.example.com/master/dbo/default
```

2.1. Implementation

This URI can be implemented by any type of client or server software, but is particularly suited to web browsers, or other software able to display formatted text. Implementation should be simple and straightforward, following the ODBC API specifications, documentation for which is available in many locations online. Once a user types in (or script specifies) a URI location, the client should use the speficied ODBC Driver to establish a connection to a database as specified in the rest of the URI, fetch data from the database and tables, optionally affecting the data using the optional query parameter. Then, the result can be displayed in a window for the user to view, in a very similar way that a user can access an FTP server using the ftp:// URI through a web browser, and the results are displayed as a formatted HTML page.

Implementers should avoid relying on third party resources other than the ODBC Driver in order to implement any part of this process. The idea of using ODBC is to access a common API interface in order to communicate directly to a database. Several systems already exist to export data from databases and other sources. The goal of this URI is to have direct access to the data source, without the need for a web server, server process or client plugin. ODBC is easy to implement in most languages, and should be implemented in the client software directly. For example, a web browser should be able to use any installed ODBC driver on the local machine to access an appropriate database engine, and display fields of data as formatted text in a similar way that it can connect to an FTP server and display a directory tree.

The alternate way to specify a [host], [user], [password] and [port] in the URI may require the software to create a custom DSN based on those values, before passing the results to the ODBC Driver. This alternate way is provided to make it easier for users to type in the URI, without having to remember specific DSN syntaxes. Implementers are also free to add various options as they see fit, such as ENCRYPT=1 to force SSL encryption. Software should try to support every possible ODBC Drivers, but constraints may require them to limit support to a specific list of drivers.

The CLI specification has commands to do a large amount of actions on data in a database, such as making new tables, inserting data, and altering it. At a bare minimum, any client software implementing this URI scheme must be able to display data through the common SELECT keyword. Any other action is optional. Any software should also be able to deal with any properly implemented ODBC Driver, including those dealing with non-standard database engines, like CSV files. It should also be able to connect to both local and remote databases, as specified by the DSN. Finally, it should accept any valid parameter as part of the DSN. Many ODBC drivers have options that can be added to the DSN string, and those should be passed on to the driver properly.

2.2. Security Considerations

The use of a standard URI format that can be entered by users presents obvious security concerns. Any software implementing this URI scheme must take care to sanitize user input. In particular, optional queries should be validated before being passed to the database engines. Furthermore, data transiting over networks can be intercepted if in clear text. Most database engines offer encryption, sometimes through optional parameters passed in the DSN. Finally, for clients that can access code from remote sources such as a web browser, care should be taken when allowing access to local databases through a script.

3. Acknowledgements

This document was based on a template derived from an initial version written by Pekka Savola and contributed by him to the xml2rfc project.

4. IANA Considerations

This memo includes no request to IANA.

5. References

5.1. Normative References

[RFC2396] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax", RFC 2396, August 1998.

5.2. Informative References

[CLI]   International Organization for Standardization / International Electrotechnical Commission, "Call Level Interface (ISO/IEC 9075-3:2003)", 2003, <https://www2.opengroup.org/ogsys/catalog/c451>.

[ODBC]   Microsoft, "Open Database Connectivity (ODBC)", 2010, <http://support.microsoft.com/kb/110093>.

Author’s Address

Patrick Lambert (editor)
Dendory Networks
Montreal
CA

Email: dendory@live.ca
URI: http://dendory.net