

# Enabling Access to Federated Grid Databases: An OGSA-DAI ODBC Driver

Michael J Jackson<sup>1</sup>, Ashley D Lloyd<sup>2,3</sup> and Terence M Sloan<sup>1</sup>

<sup>1</sup>EPCC, The University of Edinburgh, James Clerk Maxwell Building,  
Mayfield Road, Edinburgh EH9 3JZ.

<sup>2</sup>Management School, The University of Edinburgh, William Robertson Building, 50 George Square,  
Edinburgh EH8 9JY.

<sup>3</sup>Curtin Business School, Curtin University of Technology, GPO Box U1987, Perth, Western Australia  
6845.

## Abstract

Access to distributed heterogeneous data resources is possible through Grid middleware such as OGSA-DAI (Open Grid Services Architecture – Data Access and Integration) which is designed to assist with the integration of data from separate resources via Web services. This paper reports on the INWA project’s investigations into whether OGSA-DAI can be exposed as an ODBC data source and used as a back-end to existing, popular data extraction and processing programs that are ODBC-compliant. This capability would provide additional location and product transparency for the data resources used by these programs. It also opens up the possibility of allowing these programs to extract data from federated data resources or “virtual databases”.

## 1. Introduction

The INWA project [1] is funded by the UK Economic and Social Research Council, as part of their programme on “Pilot Projects in e-Social Science”. The remit of the project is “Informing Business and Regional Policy: Grid-enabled fusion of global data and local knowledge”. To achieve data fusion and the delivery of required analytical tools and processing power, a capability for the secure data mining of corporate data was developed using existing Grid technologies [2].

During the initial phase of the INWA project, when data was being accessed and analysed between Curtin Business School (Perth, Western Australia) and EPCC (Edinburgh, UK), it quickly became evident that the most immediate realisation of the Grid vision of focusing knowledge within a globally distributed team would be achieved if researchers were able to seamlessly use their existing, familiar, applications to access and analyse data located at remote sites. If this were possible then researchers would not even need to know that these data Grids existed during the execution of their data access tasks. Removing this learning curve might be expected to improve the rate of adoption of the underlying

Grid technologies and improve the transfer of skills into a Grid environment.

A question therefore arises – how to enable standard data analysis tools used by researchers, such as SPSS (Statistical Package for the Social Sciences) [3] or SAS (Statistical Analysis System) [4], to access Grid-enabled data resources and so migrate their existing skills into a Grid environment.

One possible solution is via the development of an ODBC driver for OGSA-DAI. ODBC [5] is a standard API for interacting with data resources and is supported by SPSS and SAS, amongst others. OGSA-DAI [6] supports the exposure of data resources onto a Grid. Exposing OGSA-DAI via ODBC would therefore allow SPSS, SAS, or indeed any ODBC-compliant tools, to access data in a Grid environment. From the perspective of these tools, OGSA-DAI is, for all intents, their back-end data resource. OGSA-DAI however, manages communications with the actual resources which store the required data.

The format of this paper is as follows. In sections 2 and 3 we provide brief overviews of ODBC and OGSA-DAI respectively. In section 4 we outline the constraints under which this work was undertaken and justify the solution adopted. Section 5 presents the design of our prototype ODBC driver for OGSA-DAI, developed using the Open Access ODBC SDK

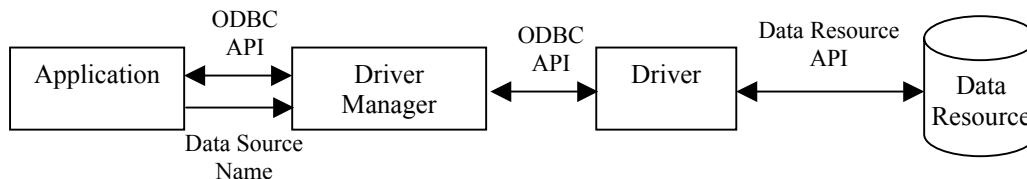
[7]. In section 6, we describe the potential benefits that such a driver could yield. In section 7 we outline some of the issues and concerns that arise from this work. Finally, section 8 provides a summary of the paper.

## 2. ODBC

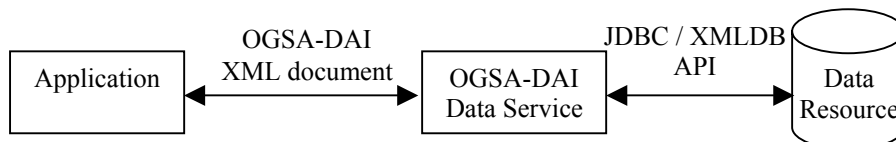
*Open Database Connectivity (ODBC)* is a specification from Microsoft that is intended to support access to any data from any application irrespective of the data resource, or database management system, within which the data resides. It is a common language through which applications can access and update data.

An application interacts with a data resource via an *ODBC driver*. An ODBC driver converts commands expressed in ODBC into data resource-specific commands and responses from a data resource back into ODBC for return to the application.

The association of a data resource plus a compatible driver is termed an *ODBC data source*. Each ODBC data source has a name



**Figure 1: Interacting with data resources via ODBC**



**Figure 2: Interacting with data resources via OGSA-DAI**

unique to a specific host. ODBC data sources are created via the use of an *ODBC administrator program*. These are typically available on PC desktops and allow a user to select an ODBC driver, from those available on the host, and provide information on the data resource to be accessed via that driver (for example its connection URL, user name and password). The user also specifies the name of the ODBC data source.

To access a data resource via ODBC, an application provides the name of the relevant ODBC data source to an *ODBC driver manager*. The driver manager loads the appropriate driver based upon the data source configuration. The application then interacts with the driver, and so the data resource, via the

driver manager. Figure 1 shows typical ODBC components and their interactions.

Under this architecture, changing a data resource with which an application interacts requires only changing the driver used and the ODBC data source configuration rather than recoding significant parts of the application code or recompiling or relinking this code.

In addition, ODBC supports a call-level interface to data resources. SQL statements are generated at run-time, they do not have to be embedded within the application and pre-compiled. In addition, ODBC does not require that applications be compiled against data resource-specific libraries either.

## 3. OGSA-DAI

*Open Grid Services Architecture – Data Access and Integration (OGSA-DAI)* provides access to data resources via WSI-, WSRF- or OGS-compliant Web services. Via OGSA-DAI data services, applications can execute database access and update operations, data

transformations and asynchronous third-party deliveries.

Applications forward XML documents – *Perform documents* – to OGSA-DAI services which express the data-related activities to be executed. The service executes the activities, returning the results to the application in XML documents – *Response documents*. These results may include data itself. Figure 2 shows typical OGSA-DAI components and their interactions.

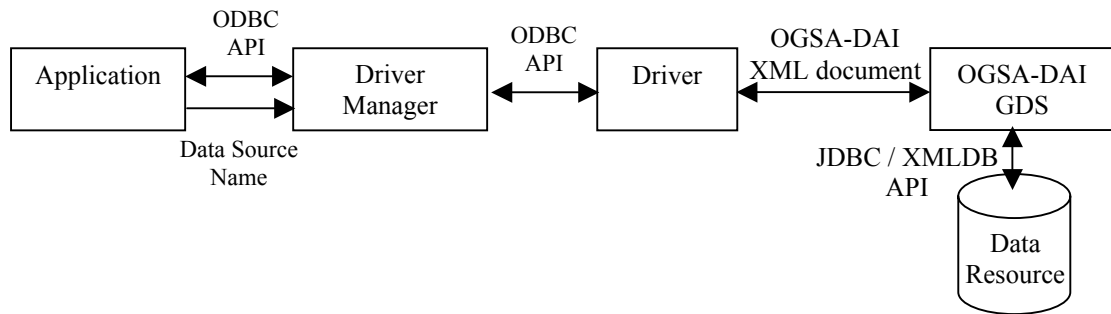
The OGS-compliant OGSA-DAI release 5 was used in this work. An application calls a *Grid Data Service Factory (GDSF)* service and requests the creation of a *Grid Data Service (GDS)*, which can be viewed as an embodiment of a session. The application then forwards *Perform documents* to and receives *Response documents* from the GDS.

OGSA-DAI also provides a client toolkit which supports access to OGSA-DAI services from within Java [8]. The benefit of this toolkit is that it removes from application programmers the need to construct or parse Perform or Response documents. Rather, application programmers can express requests by building objects representing OGSA-DAI-supported activities. The toolkit manages XML manipulation and client-service

independent skeleton and fill in with OGSA-DAI-specific detail.

3. Use an ODBC SDK to develop a driver. A number of commercial SDKs are available, for example Open Access [7], Simba [9] and Syware [10] and.

Option 1 was the most attractive option. However, it was discovered that there is a lack of ODBC-related development resources



**Figure 3: Exposing OGSA-DAI via ODBC**

communications. The toolkit also protects developers from changes in APIs and specifications.

#### 4. Constraints and Options

The focus of researchers using tools like SAS and SPSS is typically on data analysis rather than data management. Consequently, the project decided to develop a prototype OGSA-DAI ODBC driver that would support queries and query submission. In particular, the project focused on developing a prototype that could support the following data access scenario:

1. Connect to OGSA-DAI ODBC data source.
2. Submit a `SELECT * FROM table` query.
3. Get back the result set.
4. Disconnect from the data source.

In OGSA-DAI terms this translates into:

1. Connect to a GDSF and request creation of a GDS.
2. Construct a Perform document and submit it to the GDS.
3. Receive a Response document from the GDS and parse it.
4. Destroy the GDS.

There were three development approaches available:

1. Implement an OGSA-DAI ODBC driver from scratch with reference to the ODBC API.
2. Download an existing open source ODBC driver, extract a data resource-

available – tutorials and code skeletons for example. In addition, ODBC drivers are typically developed in C/C++ and this, consequently, would have required developing C/C++ client-side support for OGSA-DAI. Together, these factors would have increased the development time, even for a prototype, to beyond the time available.

Open source drivers were surveyed but were found to be typically C/C++-based, tightly-coupled to their specific data resource and lacking in design overviews and documentation. These problems made the extraction of data resource-independent skeletons impractical.

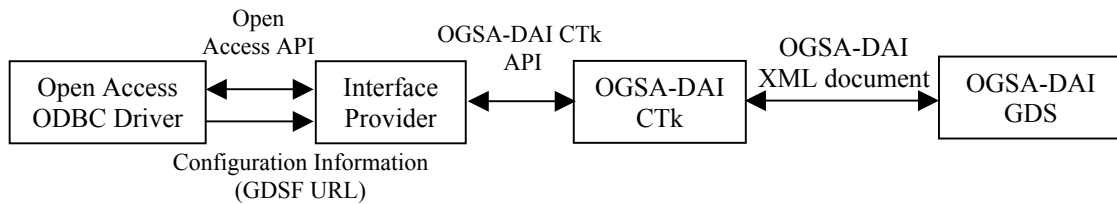
It was decided therefore to use an SDK. Though these are commercial products, and thus would require the purchase of a licence to use in a real deployment situation, it was felt that this decision was justifiable since:

- It would allow a proof-of-concept to be prototyped in a short period of time.
- If the proof-of-concept was successful a decision as to whether to request additional resources (time and effort) to undertake the development of an ODBC driver – i.e. pursue option 1 – could be made.

The Open Access SDK for Java was selected due to the availability of a 30 day evaluation licence and the fact that Java development was supported. This was beneficial since it allowed exploitation of OGSA-DAI's client toolkit.

## 5. Design and Development

The first stage in the design of an ODBC driver is to define an OGSA-DAI ODBC data source. This has a unique name and represents the association of a specific GDSF URL with the OGSA-DAI ODBC driver. An application provides the name of this data source to an ODBC driver manager. The driver manager loads the driver and provides the configuration information (i.e. the GDSF URL) to the driver. Upon receipt of a request to connect to a data resource the driver requests that the GDSF create a GDS.



**Figure 4: An OGSA-DAI ODBC driver implemented using Open Access**

The application can then interact with the GDS, and consequently with a data resource, via ODBC, as shown in Figure 3.

It is important to note that it is an OGSA-DAI service and not a data resource that is represented by an OGSA-DAI data source. This is an important distinction which we return to in section 6.

### 5.1. Open Access Development

The Open Access SDK provides an ODBC driver ‘out-of-the-box’. The driver interacts with what is termed an *Interface Provider*. This is an ODBC-independent interface which is completed by developers with code specific to their data resource, in our case OGSA-DAI. For OGSA-DAI the Interface Provider operates as follows:

- Configuration information from the OGSA-DAI ODBC data source is received. From this the GDSF URL is extracted.
- A request to connect to a data resource is converted into a request, via the OGSA-DAI client toolkit, for the GDSF to create a GDS.
- A request to query a data resource is converted into the submission of a Perform document containing a query request, via the client toolkit, to a GDS. The Response document from the GDS is parsed by the

client toolkit and the results extracted.

- A request to disconnect from a data resource is converted into a request, via the client toolkit, that the GDS terminate itself.

The relationship between the various Open Access and OGSA-DAI components is shown in Figure 4.

### 5.2. Testing

An ODBC-SQL query tool, provided by Open

Access, was used to test the OGSA-DAI ODBC driver. This tool supports the submission of SQL queries to any ODBC data source. An OGSA-DAI ODBC data source was created on an EPCC PC, using the URL of a GDSF hosted by an OGSA-DAI server at Curtin Business School, Western Australia. The SQL-ODBC query tool was given the name of the OGSA-DAI ODBC data source and a connection with this data source established. SQL queries were submitted and the results successfully displayed. Behind the scenes the Open Access ODBC driver and OGSA-DAI Interface Provider component contacted the remote GDSF and a GDS was created, the GDS serviced the query requests and was then destroyed when the connection was closed using the SQL-ODBC query tool.

## 6. Implications for Practice

The work undertaken revealed that developing an ODBC driver for OGSA-DAI was possible. However, for wider adoption of Grid technologies this capability needs to be related to practical benefits of accessing a data resource through ODBC and OGSA-DAI rather than just, for example, developing a lightweight driver that forwards ODBC calls to data resource-specific Web services. Methods for doing this already exist, for example DB2 [11] can expose Web services to support remote database query, update and management.

## 6.1. Transparency

A typical ODBC data source needs to record the location of the associated data resource so that an ODBC driver can communicate with this data resource. This is in addition to other connection information, for example a username and password. This means that every client which is to access the data resource needs to record this information. Consequently, a change in data resource location or other connection information requires each of these clients to be updated with the new information.

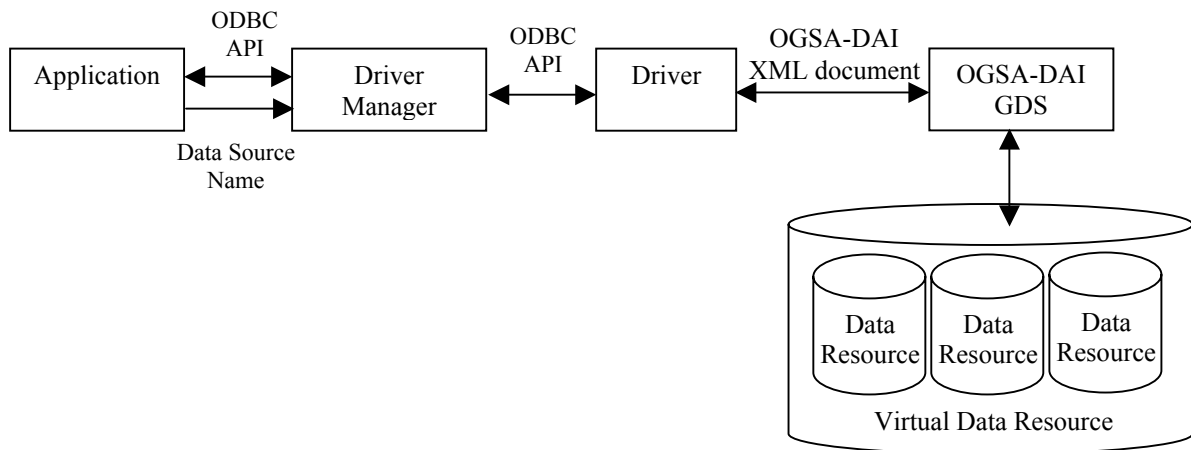
OGSA-DAI provides for a level of location transparency. The location of a data resource and other connection information is located on the OGSA-DAI server. Any changes to this information therefore affects only the server, clients are unaffected.

This transparency gives rise to the possibility of easing the exposure of data resources on a global scale. A data provider can publicise a GDSF URL plus associated information describing their data. Any client could then access this information providing

The use of ODBC and OGSA-DAI also provides for transparency of a data resource product. If a data provider decides to change their data resource product then, under the typical ODBC model, all clients would need to download and install ODBC drivers compatible with the new product. Using OGSA-DAI as an intermediary again limits the effect of product and driver updates to just the OGSA-DAI server, clients remaining unaffected.

## 6.2. Data Federation

Typically an OGSA-DAI data service allows interaction with a single resource. The OGSA-DQP project [12] are investigating the development of OGSA-DAI services which expose “virtual data resources”, a virtual data resource being a federation of a number of actual data resources. Under OGSA-DQP a OGSA-DAI-style data service (a data query service) exposes a federation of data resources – represented as GDSs – and allows clients to run SQL queries over this federation using an SQL-like language, OQL. Each data query service



**Figure 5: An OGSA-DAI ODBC driver accessing a federation of data resources via OGSA-DAI**

they have access to the OGSA-DAI client toolkit and an OGSA-DAI ODBC driver – this plus the GDSF URL allows them to configure an ODBC data source on their local host. There would be no need for clients to acquire data resource usernames and passwords for example.

This, of course, does not preclude the securing of the data for access only by specific clients – this is still achievable and can be enforced by the OGSA-DAI server in conjunction with information from a client (e.g. a certificate which could be provided as part of the OGSA-DAI data source configuration to the OGSA-DAI ODBC driver).

provides operations allowing clients to specify the GDSFs used to create the GDSs which will each provide access to the individual data resources that make up the federation.

One limitation of DQP is that the data query service is a non-standard OGSA-DAI service. However, work is currently underway to allow DQP functionality to be accessed via standard OGSA-DAI Perform documents and services. An OGSA-DAI deployer could then expose a standard OGSA-DAI service which exposes a federation of data resources, rather than a single data resource, and allows queries over this federation.

These federated resources could then be exposed as ODBC data sources as already described in previous sections (see Figure 5). Users of SAS, SPSS and other ODBC-compliant data analysis applications would then be able to extract and analyse data from these federations. These federations could be set up by users themselves or exposed by third-parties.

Such a scenario provides scope for data access and analysis scenarios which traditional ODBC usage scenarios – connecting applications directly to data resources via ODBC – do not support.

## **7. Development Issues and Concerns**

The demonstration of core functionality required by an OGSA-DAI ODBC driver raises a number of issues with regard to compatibility and compliance, and hence concerns about how this work should be taken forward.

### **7.1. OGSA-DAI WSI and WSRF Compliance**

The work described was completed using OGSI-compliant OGSA-DAI release 5. There would be minimal changes if utilising OGSA-DAI WSI- or WSRF-compliant data services instead. OGSA-DAI WSI and WSRF have no notion of a factory, rather they are based around the notion of a data service which exposes one or more data service resources. Under such a system an OGSA-DAI ODBC data source, instead of including a GDSF URL, would include a data service URL plus the name of a data service resource exposed by that data service. Or, alternatively, the name of a data service resource plus a registry which could be searched for data services exposing those data service resources. Connection would just involve contacting the data service – ideally a session would be created which could be destroyed when the OGSA-DAI ODBC driver receives a disconnection request. Session management functionality is planned for the next release of OGSA-DAI.

### **7.2. Driver Development**

One option for continued driver development is to purchase the Open Access SDK and to complete the Interface Provider development. However, the use of Open Access would require anyone who wishes to use OGSA-DAI via the OGSA-DAI ODBC driver to purchase this SDK. This is not consistent with the aims of OGSA-DAI to provide a freely-available data access and integration solution.

The alternative is to design and write a pure OGSA-DAI ODBC driver. This will have a much higher overhead in terms of time and effort but would yield a stand-alone deliverable which is not dependent upon third-party tooling. This could then be made freely available to users.

### **7.3. ODBC Conformance**

ODBC is defined in terms of three conformance levels – Core, Level 1 and Level 2. An interface conformance level determines the features provided by any ODBC driver satisfying that conformance level. A driver conforming to a given level must support all the features specified by that level plus all those of the lower levels – it can optionally provide selected features from higher levels also.

To allow for conformance to even the Core level would require extending OGSA-DAI's functionality to support cursor management for example. Higher levels include requirements for transactions and query timeouts. Support for transactions is planned for the next release of OGSA-DAI.

Another important issue in developing an OGSA-DAI ODBC driver is the nature of meta data about data resources must be provided via the ODBC API. OGSA-DAI only provides limited meta-data at present (for example product name, vendor, version and database schema). This provision may need to be extended depending upon the detailed requirements of the ODBC specification.

Another requirement of an OGSA-DAI ODBC driver is that OGSA-DAI's SQL statement handling needs to be made more flexible and transparent to clients. At present OGSA-DAI offers support for SQL queries, updates and database management operations but these must be explicitly marked as such in Perform documents. Such distinctions between classes of SQL statement are also reflected in the client toolkit. The ODBC API makes no such distinction and so, at present, the OGSA-DAI ODBC driver must determine the type of statement so that the appropriate client toolkit components can be used. Ideally, determining the class of an SQL statement would be the responsibility of the OGSA-DAI server.

Analysis of the use of ODBC by SAS, SPSS or other ODBC-compliant data analysis applications might assist in addressing these requirements. From this analysis the most frequently used ODBC functions and types of queries submitted could be identified and, instead of developing a generic OGSA-DAI

ODBC driver, a driver specific to an application could be developed. This, of course, would be complicated by the fact that the use of ODBC by these applications might, in turn, be affected by the application domains in which they are used.

#### 7.4. Efficiency

Efficiency is a prime concern in OGSA-DAI. By providing an ODBC driver for OGSA-DAI, an additional level of indirection between applications and data is introduced as data access requests go via the ODBC driver manager and the driver itself. Using an SDK such as Open Access introduces further indirection as communications also go through SDK-specific layers. The efficiency impact of any ODBC drivers developed for OGSA-DAI on the round-trip data access time for applications must therefore be evaluated and decisions made as to whether the performance impact upon communications, which can take place on a global scale, are outweighed by the benefits of using OGSA-DAI at the back-end which were discussed in section 6.

### 8. Conclusions

This paper has described work into developing an ODBC driver for OGSA-DAI. We described the successful development of a proof-of-concept using a commercially-available SDK. We outlined the advantages that utilising an ODBC driver for OGSA-DAI may provide over direct ODBC-data resource connections including transparency of data resource location and product and access to federated data resources. We also highlighted issues of concern from the minor (WSI and WSRF compliance) to the major (full ODBC driver development, ODBC compliance and concerns about efficiency), that will impact upon how easy such a technology will be to adopt, and how well it scales up to global Grid-supported research collaborations.

### References

- [1] INWA Project, <http://www.epcc.ed.ac.uk/inwa>. ESRC award name *Informing Business/Regional Policy: Grid Fusion of Global Data & Local Knowledge*, number RES-149-25-005, investigators Lloyd, A.D., Parsons, M.I., Sloan, T.M. and Fildes, R.
- [2] Hume, A.C., Lloyd, A.D., Sloan, T.M. and Carter, A.C. Applying Grid Technologies to Distributed Data Mining. *Proceedings Grid and Cooperative Computing – GCC 2004: Third International Conference*, Wuhan, China, October 21<sup>st</sup>-24<sup>th</sup>, 2004, Springer-Verlag Heidelberg, LNCS Vol 3251/2004, pp 696-703.
- [3] SPSS, <http://www.spss.com>.
- [4] SAS, <http://www.sas.com>.
- [5] Microsoft ODBC, <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/odbc/hm/dasdkodbcoverview.asp>.
- [6] OGSA-DAI Project, <http://www.ogsadai.org.uk>.
- [7] Open Access ODBC SDK, <http://www.odbcsdk.com>.
- [8] Sugden, T., Antonioletti, M., Chue Hong, N., Hume, A., Jackson, M., Krause, A. and Westhead, M. Protecting Application Developers – a Client Toolkit for OGSA-DAI. *Proceedings of UK e-Science All Hands Meeting* (ed. Cox, S), Nottingham, 31<sup>st</sup> August-3<sup>rd</sup> September 2004, ISBN 1-904425-21-6.
- [9] Simba, <http://www.simba.com/technologies/odbc.htm>.
- [10] Syware, <http://www.syware.com/prodlib/odbc>.
- [11] IBM DB2, <http://www-306.ibm.com/software/data/db2>.
- [12] DQP Project, <http://www.ogsadai.org.uk/dqp>.