An Outline of the Global Grid Forum Data Access and Integration Service Specifications

Mario Antonioletti[1], Amy Krause[1], and Norman W. Paton[2]

[1] EPCC, University of Edinburgh, JCMB, The King's Buildings, Mayfield Road, Edinburgh EH9 3JZ, UK
[2] School of Computer Science, University of Manchester, Oxford Road, Manchester M13 9PL, UK

Abstract. Grid computing concerns itself with building the infrastructure to facilitate the sharing of computational and data resources to enable collaboration within virtual organisations. The Global Grid Forum (GGF) provides a framework for users, developers and vendors to come together to develop standards to ensure interoperability between middleware from different service providers. Central to this effort is the Open Grid Services Architecture (OGSA), and its associated specifications. These define consistent interfaces, generally couched as web services, and the components required to construct grid infrastructures. Both the web service and grid communities stand to benefit from the provision of consistent and agreed web service interfaces for data resources and the systems that manage them. This paper describes, motivates and presents the context for the work that has been undertaken by the GGF Data Access and Integration Services Working Group (DAIS-WG). The group has defined a set of data access and integration interfaces that are consistent with the OGSA vision. A brief overview of the current family of DAIS specifications is given: WS-DAI specifies a collection of generic data resource properties and messages that are specialised by WS-DAIR and WS-DAIX for use with relational and XML data resources, respectively. The WS-DAI specifications can be applied in regular web services environments or as part of a grid fabric.

Keywords: Data, Databases, Grid, DAIS, OGSA-DAI.

1 Introduction

The *Database Access and Integration Services* Working Group (DAIS-WG) was formed within the *Global Grid Forum* (GGF) to standardise service types and interfaces to allow databases to be seamlessly integrated into grids. From the very beginning the DAIS-WG has aligned itself with the GGF's *Open Grid Services Architecture* (OGSA)[OGSA] vision. The DAIS specifications would then be consistent with and be able to interoperate with the other services and interfaces being proposed for OGSA based grids. The group has in addition been in communication with other standardisation groups, both inside and outside the GGF, to ensure consistency with adjacent standardisation activities. For

J.-M. Pierson (Ed.): VLDB DMG 2005, LNCS 3836, pp. 71–84, 2005.

example, DAIS members are active in the refinement of the GGF OGSA data architecture[1], and outside the GGF, the group has provided use cases for the OASIS *Web Services Resource Framework* (WSRF) technical committee that is producing standards for identifying and interacting with resources in web services. The group has also worked with the Distributed Management Task Force (DMTF) to extend its *Common Information Model* (CIM) with an XML rendering of the CIM model that includes relational metadata.

The primary outcome of the DAIS-WG has been a collection of specifications:

1. *WS-DAI,* which defines properties and message patterns that are independent of the type of data resource that is being accessed [WS-DAI].
2. *WS-DAIR,* which extends WS-DAI with properties and messages for accessing relational data resources [WS-DAIR].
3. *WS-DAIX,* which extends WS-DAI with properties and messages for accessing XML data resources [WS-DAIX].

This paper describes these services, outlining design decisions that have influenced their scope and relationships to existing and emerging standards. The paper is structured as follows. Section 2 describes and motivates the scope of the specifications, and describes how the specifications relate to other web service standards. Section 3 provides an overview of the specifications, which is expanded on in Sections 4. Section 5 describes how the specifications make use of the Web Services Resource Framework, a family of specifications for representing resources in web services. Section 6 presents some conclusions.

2 Scope and Context

In common with most other standardisation activities, the DAIS-WG has iterated towards stable positions on *what* should be included in the standards and *how* these capabilities should be supported. This section reviews several design decisions, with a view to clarifying the role of the DAIS specifications in relation to other web and grid service standards.

2.1 Transparency

Distributed data management is associated with various forms of transparency, which may or may not be supported by an infrastructure. For example, [Ozsu-99] includes the provision of *language*, *fragmentation* and *replication* transparencies as important functionalities that a data management infrastructure may support. The key design feature behind the DAIS specifications that affects their relationship to such transparencies is that they are designed to provide access to *existing* data management systems. As such, the DAIS specifications are silent with respect to both *fragmentation* and *replication* transparencies; the specifications can be used to access database management systems that support such

[1] See http://forge.gridforum.org/projects/ogsa-d-wg for more details.

transparencies or not, but this need not be the concern of the implementer of the specification, and does not surface in the specifications themselves.

A similar position holds with respect to *language* transparency. Many operations in the DAIS specifications take query language statements as parameters. Such operations are generally explicit about the language that is to be used, but DAIS does not require that service implementers parse such language expressions. As such, the DAIS specifications essentially provide web service wrappers for databases; such wrappers will typically pass query language statements directly to an underlying database management system, but are at liberty to intercept, parse, translate or redirect such language statements – DAIS compliant services may implement thin or thick wrappers. As such, the specifications have dependencies on existing query language standards, but are not prescriptive with respect to how a service processes statements provided in such standards.

2.2 Request Composition

Requirements analyses conducted by the DAIS-WG [Atkinson-03] indicated that there was significant demand for services that not only accessed data resources, but which supported flexible data movement and transformation capabilities. For example, there was a widespread need for the ability to express a request that could retrieve data from a database, transform the data using XSLT, and deliver the result to a third party. The DAIS-WG designed simple language interfaces to support such requirements, which formed the basis for the activity model in the widely-used OGSA-DAI system [Antonioletti-05]. However, defining the scope and role of such a language in relation to emerging workflow specifications proved problematic, and the current DAIS specifications support a more limited collection of access patterns that provide extensibility points for more sophisticated data transformation or movement functionalities.

2.3 Metadata

Data access services may have to be able to be discovered and used on the basis of the metadata provided by the services. As such, the DAIS specifications provide a wide range of properties that can be used to describe the behaviour of a service to its consumers. One significant issue for the group has been the provision of an XML Schema for describing the structure of a relational database; such a schema is a complex artifact, which is potentially of use in settings other than DAIS services. As such, the DAIS-WG is working with the Database Working Group of the DMTF to extend the coverage of the CIM database model to include relational metadata from the SQL standard. In parallel with this modelling activity, the DMTF is working to support an XML representation of the complete CIM model, which should be usable by several GGF working groups to describe the properties of resources on a grid.

2.4 Transactions and Security

Web services specifications, such as *web services security* [WS-Security] and *web services atomic transaction* [WS-AtomicTransaction] can be used to specify the

security and transaction contexts for a DAIS message. As a result, the DAIS specifications do not provide messages or properties addressing such capabilities. Unfortunately, at the time of writing there are competing proposals for web service transaction standards, and thus the DAIS specifications are likely to complete their standardisation process before there is a widely accepted standard for combining DAIS messages into transactions.

3 Specifications Overview

Before proceeding some DAIS terminology used in the remainder of this paper is outlined. A *data resource* is any entity that can act as a source or sink of data. Data resources may be further sub-classified into: *externally managed data resources* which exist independently of DAIS services and have their lifetime managed by means outside the control of the service, and *service managed data resources* which do not normally exist outside the service-oriented middleware and whose lifetime is controlled by the service. Thus, a database in a DBMS system will normally be in the externally managed data resource category while data stored in memory by a service, which may have been derived from an externally managed data resource, and is accessed via a service will generally be in the service managed data resource category.

A data resource must always have an identifier, an *abstract name*, which is unique and persistent. There is currently an on-going effort to standardise naming of entities within OGSA; for now DAIS uses a URI to represent data resource's abstract names. A DAIS service that provides access to a data resource is called a *data service* – a data service may represent zero or more data resource. The data resource to which a message is targeted at, through a data service, is specified by the provision of the data resource's abstract name in the body of the SOAP message, optionally also including a data resource address in the header of the SOAP message. A *data resource address* is an *End Point Reference* (EPR) as defined in WS-Addressing [WS-Addressing] which also contains the abstract name of the data resource in its *reference parameters*. DAIS mandates the inclusion of the data resource's abstract name in the body of the message so that the messaging framework is the same regardless of whether WSRF is used or not. A *consumer* is an application that exploits a data service to access a data resource.

Two main types of access pattern have been proposed within DAIS. These are represented schematically in Figure 1. *Direct access* mimics the standard request-response pattern currently employed in most web service interactions.

Indirect access uses the factory pattern to create a derived data resource located at the service end. Thus, any data resulting from a consumer-service interaction is not returned to the consumer in the response, as is the case for direct access. Instead, the consumer receives an EPR which can then be used to access the data via a data service. This data service could support a different service interface from the service that created the data resource. This avoids unnecessary data movement and could, in effect, be used as an indirect form of

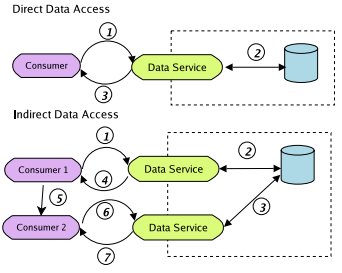

Fig. 1. Direct and Indirect access – the numbering indicates the temporal ordering of the interactions with the corresponding consumers

third party delivery as is illustrated in the picture where the EPR is passed to a second consumer which then pulls the data from the second data service.

The DAIS specifications classifies its interfaces into types originally proposed in the *OGSA Data Services* [OGSA-Data] document:

Data description contains a set of properties – XML elements collected together in a property document – that provide metadata about the underlying data resource and the relationship between the data resource and the data service with which it is associated. Some of these properties are static and are thus informational while others may be changed and may thus affect the behaviour of the service.

Data access collects a set of operations that provide access to a data resource through a data service. These operations implement the direct data access pattern.

Data factory collects together a set of operations that can be used to create derived data resources. These also provide mechanisms to specify the data service interfaces which are to be used to access the data. These operations implement the indirect data access pattern.

Data management was originally also included in this interface classification. However, a management interface could be used to manage: the web service, the data resource through the web service or the relationship between the web service and its associated data resource. The first two types of management were deemed to be out of scope for DAIS as the general management of or through web services is of wider interest than just to the DAIS community. Moreover, the OASIS DMTF TC has provided a set of standards to manage web services and manage entities through web services [MOWS, MUWS]. The DAIS specifications then only provide a limited means for managing the relationship between a data service and its data resource. The next sections consider the specifications in more detail, concentrating on the core messages and properties in the WS-DAI specification together with its WS-DAIR extensions. The principles employed for extending the core interfaces and properties to cater for XML based data resources are very similar and are not covered in this paper – for details see [WS-DAIX].

4 WS-DAI and WS-DAIR

4.1 Message Patterns

The WS-DAI specification defines a set of core properties and operations that are independent of any particular data model used by a data resource. These are then extended by realisations to cater for particular types of data resource. The WS-DAIR and WS-DAIX specifications extend the operations and properties defined in the core document to provide access to XML and relational data resources respectively. The core specifications also provides a set of message patterns that must be observed by realisations. This ensures that DAIS as a whole has a coherent framework. To date most of the effort has been spent in producing realisation for XML and relational data resources although there are preliminary drafts of documents that aim to extend the base DAIS interfaces to deal with object databases and files.

Figure 2 illustrates the message pattern prescribed by the WS-DAI document that is to be used for direct data access interfaces. Note that this is only an illustrative example, for the actual details you should refer to the corresponding specifications. For each of these templates a relational example of its implementation is also shown.

The contents shown in this figure are intended to go in the body of a SOAP message. The `DataResourceAbstractName` identifies the data resource the message is targeting and the `DataFormatURI` specifies the format in which the data should be returned to the consumer. Valid return formats are specified in one or more `DatasetMap` properties set by the service (see later). The query expression is found at the bottom of the message. It is also possible to include parameterised queries with a list of parameters contained in the same request message though this is not shown in this figure. If the query is successful the data is returned to the consumer in the response message. Note that the SQL realisation extends

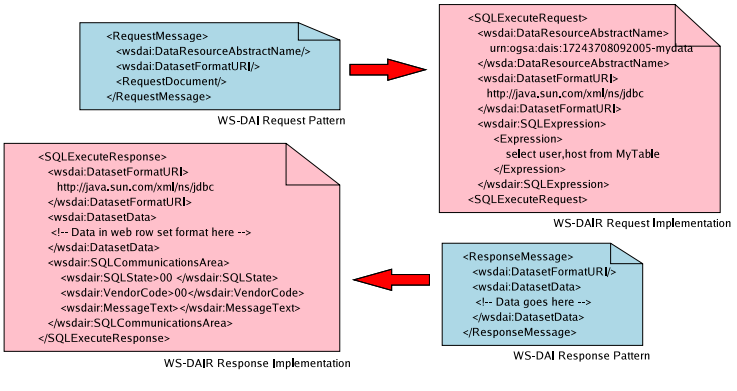

Fig. 2. The DAIS direct data access pattern specified in the core spec and its implementation in the relational specification

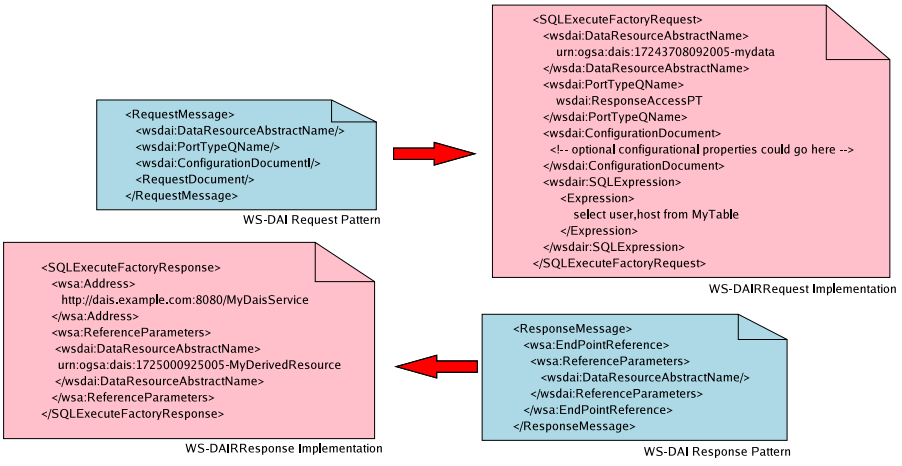

Fig. 3. The DAIS indirect data access pattern specified in the core spec and its implementation in the relational specification

the message pattern to also include information from the SQL communication area.

Figure 3 contains the corresponding message pattern set by the WS-DAI specification for indirect data access together with a WS-DAIR realisation implementation of this pattern.

The request message contains the mandatory data resource abstract name and an optional element containing the QName of the port type with which a data service will provide access to the resulting data. A configuration document allows default values to be set for some of the properties of that data service (values for this are not shown in the figure but the corresponding properties are described in the next Section). The request message contains the query that will populate the resulting data resource. If the query is successful the consumer gets the EPR from which the data may be retrieved.

4.2 Service Properties

Figure 4 shows the properties defined in the WS-DAI specification and the extensions made in the WS-DAIR specification. The different SQL extension groupings reflect the possible service interfaces that can be used to access different types of relational data. The names used for the WS-DAI properties largely describe their purpose. A cursory review is given here but details are left to the WS-DAI specification. Properties can be divided into two general classes: *static properties* which are largely defined by the implementation and cannot be modified and *configurable properties* that may be set by the consumer when they create a data resource using the indirect data access pattern.

The static properties shown in Figure 4: the `DataResourceAbstractName` property provides a place holder for the unique and persistent name of the

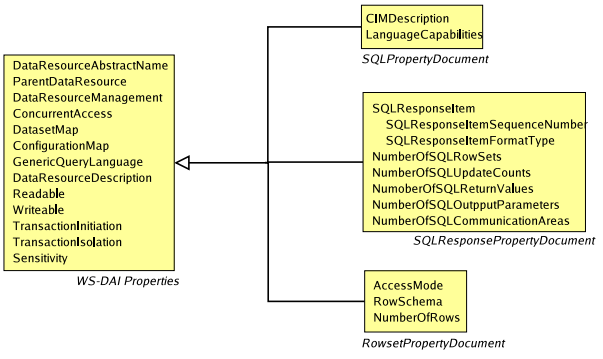

Fig. 4. The core properties defined in the WS-DAI specification and the different extensions to these in the WS-DAIR specification

data resource; the `ParentDataResource` property contains the parent's data resource abstract name if this is a derived data resource otherwise it is empty; the `DataResourceManagement` property indicates whether the data resource is externally or service managed; the `ConcurrentAccess` property is a boolean indicating whether the data service supports concurrent access or not; the `DatasetMap` property provides a means of specifying the valid return formats supported by a data service, there will be one of these elements for each possible supported return type; the `ConfigurationMap` property associates an incoming message type with a valid requested access interface type and a default set of values for the configuration property document; finally the `GenericQueryLanguage` property specifies the valid query languages that can be used with the generic query operation defined in the core spec (see below).

The configurable properties can be set when a new data service-data resource relationship is established: the `DataResourceDescription` allows a human readable description of the data resource to be provided; `Readable` is a boolean indicating whether the data resource can be read by the consumer; the `Writeable` property is another boolean indicating whether the data resource can be written to; the `TransactionInitiation` property enumerates the possible transactional support provided by the service on the arrival of a message – possibilities are: there is no transactional support, an atomic transaction is initiated on the arrival of each message or the message corresponds to a transactional context which is under the control of the consumer; the `TransactionIsolation` property enumerates behaviour of how transactions behave in relation to other on-going transactions, details are left to the specification; finally the `Sensitivity` property describes how sensitive the derived data is to changes in the values of the parent data resource, i.e. whether changes in the parent data resource will be reflected in the derived data or not.

The relational extensions to these base properties are largely self explanatory and are not described here other than for the `CIMDescription` property which is a content holder for an XML rendering of CIM for relational database that is

being produced by the DMTF. This will be used by DAIS to provide metadata about the relational data resource. The details of these properties are available in the WS-DAIR specification.

It is perhaps more illustrative to examine the use case represented in Figure 5.

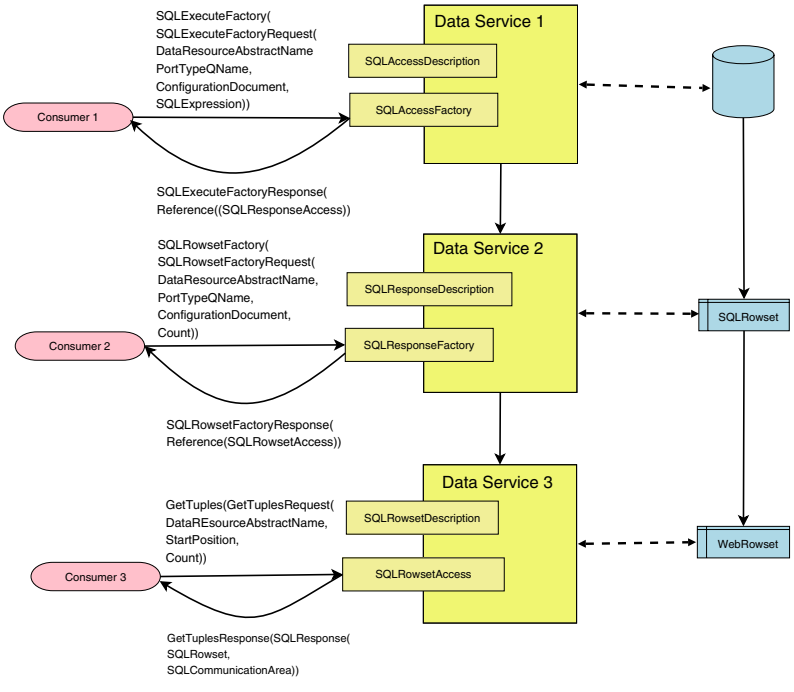

Fig. 5. Example of relational data services

In this example *Data Service 1* is associated with a relational data resource. *Consumer 1* sends a message to the service's `SQLExecuteFactory` operation to create a data resource, populate it with the result set returned from the query sent to the relational data resource and associate this data resource with a data service, *Data Service 2*, that supports an `SQLResponseFactory` interface. The response returned to *Consumer 1* contains the EPR required to access this data. *Consumer 1* passes the EPR to *Consumer 2* which then sends a message to this data service's `SQLResponseFactory` interface to create another data resource which uses a web row set format and is associated with a data service, *Data Service 3*, that supports an `SQLRowsetAccess` interface. *Consumer 2* then gets an EPR which he passes on to *Consumer 3*. *Consumer 3* then finally uses the `SQLRowAccess` to pull the data off the service. This example shows how the different hierarchies of services can be used to access different types of relational data and the context in which the relational property extensions may be used although this has not been shown in this example. Different consumers have

been used to indicate the versatility of the pattern though in practise these will generally be the same entity. Clearly it is not necessary to go through all the steps to get a web row set data resource – all that would be required is for *Data Service 1* to support the `SQLResponseFactory` interface for this to happen. Finally, in this instance the first data resource would correspond to an externally managed data resource while the other two derived data resources would be service managed data resources.

4.3 Operations

To conclude this section a brief overview of the operations defined in the WS-DAI specification and the extensions made in the WS-DAIR specification is given. These are illustrated in Figure 6. The WS-DAI specification only defines three core data access operations. There is a `DestroyDataResource` operation that destroys the relationship between the data service and the data resource. Once this is done the service will no longer have any knowledge of that data resource and will not be able to provide access to it. What this entails for the data held in that data resource is dependent on whether it is an externally managed data resource in which case the data will probably remain in place or, if it is a service managed data resource, in which case the data should be removed once the relationship is terminated. The `GenericQuery` allows a query expression to be submitted to the underlying data resource without having to use one of the specialised interfaces. Valid query languages are advertised in the `GenericQueryLanguage` property previously described. A `GetDataResourcePropertyDocument` allows the *whole* property document for WS-DAI defined properties in that data service to be retrieved. Properties within this property document cannot be obtained at a lower level of granularity unless WSRF is used (see the next section).

The `CoreResourceList` is an optional set of operations that may be implemented by a DAIS service. If this is implemented, the list of data resources known to a data service may be retrieved using the `GetResourceList` operation. Also, the EPR corresponding to a data resource's abstract name may be retrieved using the `Resolve` operation.

The relational extensions to the core defined operations have a base `SQLAccess` interface that allows SQL expressions to be submitted to a relational data resource, the results of which will be returned in the response, and an operation to retrieve the `SQLPropertyDocument` which contains metadata about the relational data resource, the data service, and the data service-data resource relationship. A `SQLFactory` interface allows a service managed data resource to be created and populated by the response of a SQL query. The data resource is then associated with an appropriate data service supporting an access interface requested by the consumer. This could use the `ResponseAccess` collection of operations which allow the data to be retrieved as well as finding out about the nature of the response. Likewise, a `ResponseFactory` allows a rowset based data resource to be created that can, in turn, be accessed by using the `RowsetAccess` set of operations. These then are the extensions to the core operations defined in the WS-DAIR specification to cater for relational data resources.

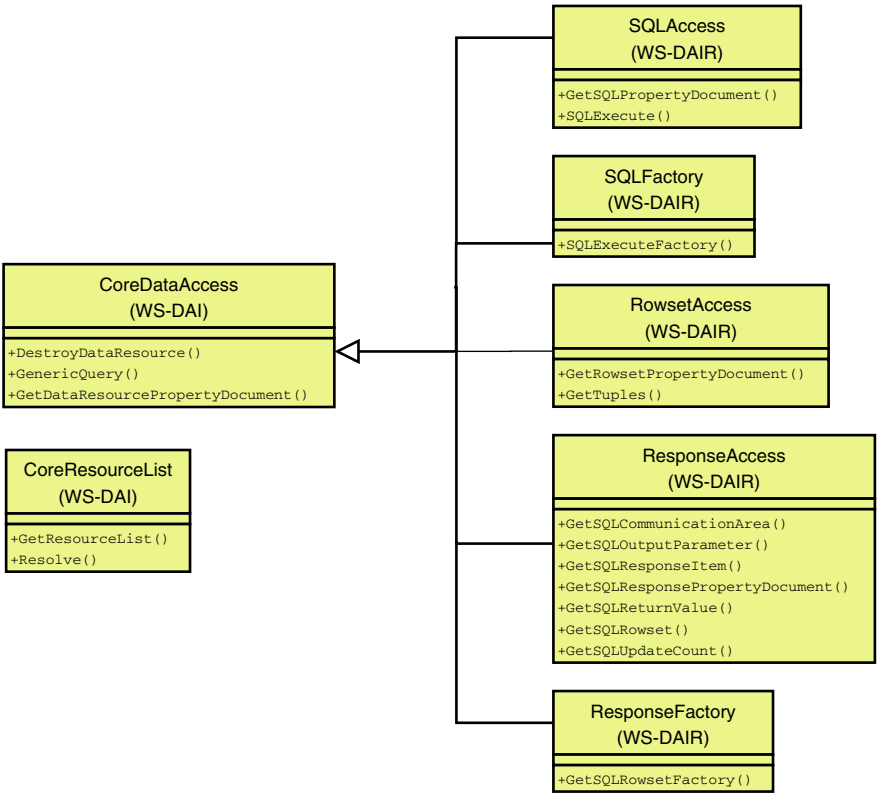

Fig. 6. Operations defined in the WS-DAI specifications and the extensions made in the WS-DAIR specification

Only the core and relational properties and interfaces have been outlined here. The XML extensions follow the same principles and provide support for querying XML data resources using XQuery, XPath, XUpdate as well as operations that manipulate collections and others that provide access to service managed data resources, details are in the WS-DAI specification [WS-DAIX]. It is worth noting that DAIS does not prescribe how these operations are to be combined to form services; the proposed interfaces may be used in isolation or in conjunction with others and, in time, viable compositions of interfaces will follow established patterns for data access.

5 DAIS and WSRF

The core functionality of DAIS has no reliance on WSRF. One can use the base interfaces that have already been outlined in this paper and access the service properties without requiring WSRF, albeit if you do not use WSRF you can only retrieve the whole property document. There are no means provided for getting

these properties at a finer level of granularity unless WSRF is used. The soft state lifetime management without WSRF has to be explicit, i.e. the consumer has to send a destroy operation to the data service or the data resource will be accessible for as long as the data service is there. Using WSRF allows one to have fine grain access to the service properties [WS-ResourceProperties] and also use the WSRF soft state lifetime management [WS-ResourceLifetime] but there is a caveat: you still require the data resource abstract name to be included in the message body even if it is only for a WSRF implementation to ignore it. This was done to preserve the message structure regardless of whether WSRF is used or not.

Figure 7 schematically represents how a WSRF service infrastructure is layered over the core DAIS functionality.

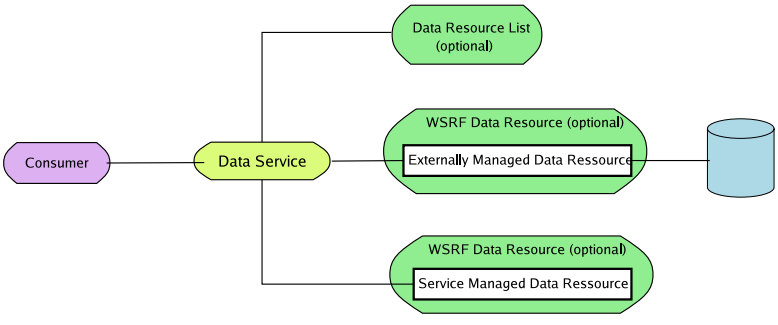

Fig. 7. WSRF DAIS extensions

A data service may represent more than one data resource. If WSRF is used each data resource must have a corresponding WSRF data resource. These are used for both externally and service managed data resources. In order to allow a data resource's abstract name to be mapped to an EPR the optional *data resource list* interface may be implemented which provides such functionality as well as allowing a consumer to obtain a list of all the data resources known to a data service.

This ability to use DAIS without requiring an explicit dependency on WSRF has been undertaken in order to provide a means for those who would like to use DAIS but are not ready to adopt WSRF. The ability to use DAIS without WSRF, in essence, provides a potential upgrade path by allowing service providers to start off with a non-WSRF solution and then, as confidence in WSRF grows, move on to exploit the additional capabilities provided by WSRF.

6 Conclusions

The DAIS specifications only provide web service interfaces to relation and XML data resources. However, the overall framework is extensible, and different groups

are exploring the development of additional realisations for object databases, ontologies and files.

The DAIS specifications are intended to provide useful functionalities for describing and accessing data resources, and can be used to support common use cases directly. However, they have really been designed to form part of a wider service-based architecture. As such, standards from the web services community for authentication and transaction management, and from the grid community for data movement and negotiation, can be seen as complementing the DAIS specifications, and facilitating their use in an increasing range of applications.

Acknowledgements. We gratefully acknowledge the input and contributions that have been made to the production of these standards, in particular: Bill Allcock, Malcolm Atkinson, N.Chue Hong, Brian Collins, Patrick Dantressangle, Vijay Dialani, Dieter Gawlick,Shannon Hastings, Allen Luniewski, James Magowan, Sastry Malladi, Inderpal Narang, Savas Parastatidis, Greg Riccardi, Steve Tuecke, Jay Unger, Paul Watson, Martin Westhead, Simon Laws, Susan Malaika, Dave Pearson and the many others who are not explicitly mentioned here.

References

[Antonioletti-05] M. Antonioletti, M.P. Atkinson, R. Baxter, A. Borley, N.P. Chue Hong, B. Collins, N. Hardman, A. Hume, A. Knox, M. Jackson, A. Krause, S. Laws, J. Magowan, N.W. Paton, D. Pearson, T. Sugden, P. Watson, and M. Westhead. The Design and Implementation of Grid Database Services in OGSA-DAI.Concurrency and Computation: Practice and Experience, 2005, Volume 17, Issue 2-4 , Pages 357-376.

[Atkinson-03] M.P. Atkinson, V. Dialani, L. Guy, I. Narang, N.W. Paton, D. Pearson, T. Storey and P. Watson. Grid Database Access and Integration: Requirements and Functionalities. GFD.13. March 13th 2003. http://www.ggf.org/documents/GFD.13.pdf.

[DAIS-Mappings] S. Laws, S. Malladi, S. Parastatidis. Scenarios for Mapping DAIS Concepts. September 1, 2004. http://forge.gridforum.org/projects/dais-wg/document/Scenarios_for_Mapping_DAIS_Concepts/en/3

[MOWS] I. Sedukhin (Ed). Web Services Distributed Management: Management of Web Services (WSDM-MOWS) 1.0. OASIS-Standard, 9 March 2005. http://www.oasis-open.org/apps/org/workgroup/wsdm/download.php/11761/wsdm-mows-1.0.pdf

[MUWS] W. Vambenepe (Ed).Management: Management Using Web Services (MUWS 1.0) Part 1. OASIS Standard, 9 March 2005. http://www.oasis-open.org/apps/org/workgroup/wsdm/download.php/11734/wsdm-muws-part1-1.0.pdf

[OGSA] I. Foster (Ed), H. Kishimoto (Ed), A. Savva (Ed), D. Berry, A. Djaoui, A. Grimshaw, B. Horn, F. Maciel, R. Subramaniam, J. Treadwell, J. Von Reich. The Open Grid Services Architecture, Version 1.0. Global Grid Forum. GFD-I.030. 29 January 2005. http://www.ggf.org/documents/GFD.30.pdf

[OGSA-Data] I. Foster, S. Tuecke, J. Unger, A. Luniewski. OGSA Data Services. February 24, 2004 http://forge.gridforum.org/projects/dais-wg/document/OGSA_Data_Services-ggf10/en/1

[Ozsu-99] T. Ozsu and P. Valduriez, Principles of Distributed Database Systems, 2nd Edition, Prentice-Hall, 1999.

[WS-Addressing] M. Gudgin (Ed), M. Hadley (Ed). Web Services Addressing 1.0 - Core. W3C Candidate Recommendation 17 August 2005. http://www.w3.org/TR/ws-addr-core.

[WS-AtomicTransaction] L. F. Cabrera, G. Copeland, M. Max Feingold (Editor) R. W. Freund, T. Freund, J. Johnson, S. Joyce, C. Kaler, J. Klein, D. Langworthy, M. Little, A. Nadalin, E. Newcomer, D. Orchard, I. Robinson, T. Storey, S. Thatte, Web Services Atomic Transaction (WS-AtomicTransaction). Version 1.0. August 2005. ftp://www6.software.ibm.com/software/developer/library /WS-AtomicTransaction.pdf

[WS-DAI] M. Antonioletti, M. Atkinson, S. Malaika, S. Laws, N. W. Paton D. Pearson and G. Riccardi. Web Services Data Access and Integration (WS-DAI). DAIS-WG Specification Version 1.0. Draft, Global Grid Forum. 2005.

[WS-DAIR] M. Antonioletti, B. Collins, A. Krause, S. Malaika, J. Magowan, S. Laws, N. W. Paton. Web Services Data Access and Integration - The Relational Realisation (WS-DAIR) Specification Version 1.0. Draft, Global Grid Forum. 2005.

[WS-DAIX] M. Antonioletti, A. Krause, S. Hastings, S. Langella, S. Malaika, S. Laws, N. W. Paton. Web Service Data Access and Integration - The XML Realisation (WS-DAIX) Specification Version 1.0. Draft. Global Grid Forum. 2005.

[WS-ResourceProperties] S. Graham (Ed), J. Treadwell (Ed). Web Services Resource Properties 1.2 (WS-ResourceProperties), Version 1.2, Committee Draft 01, 19 May 2005. http://docs.oasis-open.org/wsrf/wsrf-ws_resource_properties-1.2-spec-cd-01.pd

[WS-ResourceLifetime] L. Srinivasan (Ed), T. Banks (Ed). Web Services Resource Lifetime 1.2 (WS-ResourceLifetime), Version 1.2, Committee Draft 01, 19 May 2005. http://docs.oasis-open.org/wsrf/wsrf-ws_resource_lifetime-1.2-spec-cd-01.pdf

[WS-Security] Web Services Security v1.0 (WS-Security 2004) [OASIS 200401] http://www.oasis-open.org/committees/ tc_home.php?wg_abbrev=wss