

Data Service Specification: The XML Realisation

Status of This Memo

This memo provides information to the Grid community regarding the specification of Grid Database Services. The specification is presently a draft for discussion. It does not define any standards or technical recommendations. Distribution is unlimited.

Copyright Notice

Copyright © Global Grid Forum (2004). All Rights Reserved.

Abstract

Data management systems are central to many applications across multiple domains, and play a significant role in many others. Web services provide implementation neutral facilities for describing, invoking and orchestrating collections of networked resources. The Open Grid Services Architecture (OGSA) provides consistent interfaces for creating, managing and exchanging information among Grid services, which are dynamic computational artifacts cast as Web services. Both Web and Grid service communities stand to benefit from the provision of consistent, agreed service interfaces to data management systems. Such interfaces must support the description and use of data management systems using Web service standards, taking account of the design conventions and mandatory features of Grid services. This document presents a specification for a collection of data access interfaces for XML Data Resources, which extends interfaces defined in the Grid Data Service Specification [GDSS], which in turn is based on the OGSA Data Services proposal [Data Services].

This document is presented for discussion within the Global Grid Forum (GGF) Database Access and Integration Services (DAIS) Working Group, with a view to the document evolving to become a proposed recommendation. There are several respects in which the current proposal is incomplete, but it is hoped that the material included is sufficient to allow an informed discussion to take place concerning both its form and substance. Future versions of this document will be revised to remove the dependency on the Open Grid Services Infrastructure (OGSI).

Contents

Abstract.....	1
1. Introduction	2
2. Notational Conventions	2
3. Specification Overview	3
3.1 Scope of specification	3
3.2 Mapping to the Data Service model.....	3
3.3 Relationships with other specifications	4
4. DataDescription Port Types.....	5
4.1 XMLCollectionDescription.....	5
4.2 XMLDocumentDescription	5
5. DataAccess PortTypes	5
5.1 XMLCollectionAccess	56
5.2 XPathAccess.....	6
5.3 XUpdateAccess	7
5.4 XQueryAccess	78
6. DataFactory PortTypes.....	8
6.1 XMLCollectionFactory.....	8
6.2 XMLDocumentFactory	89
6.3 XPathFactory	9
6.4 XQueryXFactory	940
7. DataManagement portTypes	10
7.1 XMLCollectionManagement.....	10
8. Security Considerations.....	11
9. Conclusion	11
Editor Information	11
Contributors	12
Acknowledgements	12
Intellectual Property Statement	12
Full Copyright Notice	13
References	13

1. Introduction

This document presents a specification for a collection of data access interfaces for XML Data Resources. An XML Data Resource is taken to mean a data source/sink, together with any associated management infrastructure, that exhibits capabilities that are characteristic of XML repositories, e.g., can be queried using XPath or updated using XUpdate or any another suitable XML query/update language. The interfaces adopt the framework provided by the OGSA Data Services proposal [Data Services], in that interfaces are categorised according to the support they provide for data description, data access, Data Service creation and data management. As such, this document should be read in conjunction with the OGSA Data Services proposal and the generic Grid Data Service Specification [GDSS], which defines the base portTypes that are extended in this specification. All of these documents assume some familiarity with the Open Grid Services Infrastructure (OGSI) [OGSI]. The specification does not mandate how the interfaces are composed into services. The proposed interfaces may be used in isolation or in conjunction with others.

2. Notational Conventions

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

This specification generally adopts the terminology currently defined in the *OGSA Data Services* document [Data Services]. The OGSA Data Services document is evolving and this terminology is expected to change in future versions of the DAIS Working Group specifications.

This specification uses namespace prefixes throughout; these are listed in the table below. Note that the choice of any namespace prefix is arbitrary and is not semantically significant.

Prefix	Namespace
dais	http://www.ggf.org/namespaces/2004/03/DAIS
http	http://www.w3.org/2002/06/wsd/http
xqx	http://www.w3.org/2003/12/XQueryX
xsd	http://www.w3.org/2001/XMLSchema
xsi	http://www.w3.org/2001/XMLSchema-instance

3. Specification Overview

3.1 Scope of specification

This document extends the interfaces presented in the *Grid Data Services Specification* [GDSS] to allow access to and description of XML Data Resources. The proposed interfaces are aligned with the base types provided in the *OGSA Data Services* [Data Services] document. The XML Data Resources are assumed to be composed of a hierarchy of collections, which store XML resources such as XML document descriptions and the documents themselves.

3.2 Mapping to the Data Service model

3.2.1 DataDescription portTypes

DataDescription portTypes allow a description of data represented by a Data Service to be provided via Service Data Elements (SDEs). No operations are defined within these interfaces. The model independent specification for these is given in the *Grid Data Service Specification* [GDSS] document. Here they are extended to provide a description of XML based Data Resources. There are two main points of extension for XML Data Resources:

- *XMLCollectionDescription*: provides information about an XML collection that a Data Service may represent.
- *XMLDocumentDescription*: provides information about a particular instance of a document that a Data Service may represent. This portType will make available information about the structure representing an XML instance document as well as any other relevant data.

These interfaces are described in Section 4.

3.2.2 DataAccess portTypes

DataAccess portTypes allow XML Data Resources to be modified through insertion or updates, or queried through an appropriate language. When a Data Service is created the supporting *DataAccess* interfaces may be specified in the parameters passed to the createService operation.

- *XMLCollectionAccess*: provides access to subcollections and documents in a collection.

- *XQueryAccess*: allows the evaluation of XQuery requests across a collection of XML documents.
- *XUpdateAccess*: allows XML documents to be updated using XUpdate.
- *XPathAccess*: allows the evaluation of XPath requests across a collection of XML documents.

These are covered in more detail in Section 5.

3.2.3 DataFactory portTypes

The *DataFactory* portTypes allow data represented in XML Data Resources, usually as the result of a query, to be instantiated as Data Services. The specialisations in this instance thus deal with the type of expression that can be passed to a *DataFactory* to expose the results in a meaningful fashion. The properties and interfaces that are supported by these Data Services are specified in the schema for the creation parameters. *DataFactory* specialisations are:

- *XMLCollectionFactory*.
- *XMLDocumentFactory*.
- *XPathFactory*.
- *XQueryFactory*.

These are covered in more detail in Section 6.

3.2.4 DataManagement portTypes

The *DataManagement* portTypes allow XML Data Resources to be managed, such as modifying the collection tree or adding or removing schemas.

- *XMLCollectionManagement*

These are covered in more detail in Section 7.

3.3 Relationships with other specifications

DAIS does not propose to provide its own query/update languages for XML based Data Resources. Instead, it acts as a conduit for existing XML based query and update languages to be conveyed to the appropriate Data Resources, in this instance XML based Data Resources or a relational Data Resource that supports XML type queries. As such DAIS relies on existing XML based query and update languages. In this document, interface support is explicitly provided for languages based on the following standards:

- **XPath**: version 1.0 is a W3C recommendation defining a language for addressing parts of an XML document [XPath]. There is work in progress to define a second version of XPath that is closely aligned with XQuery.
- **XUpdate**: is a language for updating XML documents [XUpdate]. XUpdate is a de facto standard not standardised by any of the main standardisation bodies. It is still a working draft. Nevertheless it is supported by several of the XML DBMS products in the market hence DAIS supports interfaces for XUpdate.
- **XQueryX**: currently a W3C working draft [XQueryX] proposes an XML representation for the XQuery language. [XQuery] proposes to provide a query language for XML Data Resources. Although XQuery is not yet a standard DAIS proposes to describe interfaces for it.

The DAIS framework could be extended to encompass any new or emerging XML query/update standards by employing the patterns established in this document.

4. DataDescription Port Types

The DataDescription portTypes allow metadata to be made available[m1]. DataDescription portTypes are provided for use with XML collections and for XML documents.

4.1 XMLCollectionDescription[m2][AK3]

4.1.1 Service Data Declarations

The service data elements (SDEs) described in this section are associated with a Data Resource that has been represented as an XML collection.

- *collectionStructure*: Describes the sub-collections of an XML database collection[m4].
- *collectionSchema*: a list of XML Schemas associated with a collection of XML documents[m5]. Each [m6]document in the collection must conform to one or more of the XML Schemas contained in this SDE if there are any schemas present.
- *documentNames*[MA7]: the set of names that uniquely identify each XML document belonging to the collection being described.



XMLDocumentDescription

4.2.1 Service Data Declarations

Service data, at the document level, works at a finer granularity. It provides information about a single or set of documents represented by a Data Service.

- *documentName*: single or set of document names that are available through the Data Service.
- *documentSchema*: XML Schema that all documents conform to if the schema name is not null.



5. DataAccess PortTypes



5.1 XMLCollectionAccess

The *XMLCollectionAccess* portType provides access to a collection of XML documents, providing operations for adding, updating and removing documents. If there are schemas associated with this collection, added or updated documents must validate with one of the schemas, otherwise the operation will throw a fault.

5.1.1 Service Data Declarations

No SDEs are defined in addition to those inherited from *DataAccess*.

5.1.2 Operations

5.1.2.1 XMLCollectionAccess::addDocuments

Create new XML documents in the current collection. If the new documents do not validate against any schema that is present in the collection, the operation must throw a fault.[AK8]

Input

- *documentNames*: the names of the new resources.



- *Data[MA9]*: the content of the documents

Output

- *None*.

Fault(s)

- *PermissionDenied*
- *DocumentAlreadyExists*
- *DocumentDoesNotValidate*
- *Fault*: any other fault.

5.1.2.2 XMLCollectionAccess::removeDocuments

Remove a resource from this collection.

Input

Names: names of the documents in this collection to be removed.

- *Status*: A booleans for reach resource indicating whether it was successfully removed.

Fault(s)

- *PermissionDenied*
- *NoSuchDocument*:
- *Fault*: any other fault.

5.2 XPathAccess

This portType facilitates the evaluation of XPath queries across an XML resource or a collection of resources. The response document will contain the results of the query.

5.2.1 Service Data Declarations

- *xPathVersion*: The XPath version that is supported
- *querySchema*: The XML schema for the query parameters

5.2.2 Operations**5.2.2.1 XPathAccess::query**

Query an XML resource or a collection of resources and return the result immediately.

Input

- *Expression*: An XPath request wrapped in XML, including namespaces.
- *Collection* (optional): The subcollection where the query is run. If omitted, the query is run against the default collection. Note that the *resourceID* element is ignored if the *collection* argument is not present.
- *ResourceID* (optional): An ID of a document in the collection. If omitted, the query is run across the entire collection. Ignored if the *collection* element is not specified.

```
<xsd:complexType name="XPathExpressionType">
  <xsd:sequence>
    <xsd:element name="expression" type="xsd:string"/>
    <xsd:element name="collection" type="xsd:string" minOccurs="0"/>
    <xsd:element name="resourceId" type="xsd:string" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>
```

Output

- *Response*: An XML document, the results of the XPath query.

Fault(s)

- *InvalidExpression*: The query is not a valid XPath expression.
- *Fault*: Any other fault.

5.3 XUpdateAccess

A service implementing XUpdateAccess will typically be associated with one or more XML resources (or a collection of XML resources) allowing the resources to be updated using XUpdate.

5.3.1 Service Data Declarations

- *xUpdateVersion*: The version of XUpdate that is supported (current version based on the working draft document is 1.0)
- *requestSchema*: The XML Schema for the update parameters

5.3.2 Operations**5.3.2.1 XUpdateAccess::update**

Update an XML resource using XUpdate.

Input

- *Expression*: An XUpdate request
- *Collection*: The name of the collection
- *ResourceID* (optional): The resources to be updated. If omitted the update is run against the specified collection.

Output

- *Response*: The number of modified nodes.

Fault(s)

- *PermissionDenied*.
- *InvalidExpression*.
- *Fault*: Any other fault.

5.4 XQueryAccess

This portType provides an interface for XQuery requests across a collection of XML resources.

5.4.1 Service Data Declarations

The following metadata are associated to the XQueryAccess port type:

- *xQueryVersion*: The XQuery version that is supported
- *querySchema*: The XML schema for the query parameters (e.g. XQueryX)

5.4.2 Operations**5.4.2.1 XQueryAccess::execute****Input**

- *Expression*: An XML document containing the XQuery request. The request must conform to the XML schema as defined in the *querySchema*.

Output

- *Response*: The results of the request.

Fault(s)

- *PermissionDenied.*
- *InvalidExpression.*
- *Fault:* Any other fault.

6. DataFactory PortTypes

6.1 XMLCollectionFactory

6.1.1 Service Data Declarations

No SDEs are defined aside from those inherited from DataFactory.

6.1.2 Operations

The *createService* operation inherited from the OGSi defined Factory portType is used to create a Data Service representing an existing data resource. For example, by specifying the name of a subcollection, the *createService* operation may create a service which implements the XMLCollectionAccess, XPathAccess and XPathFactory interfaces and binds to the selected subcollection of the current collection. The interfaces that the factory can create are published as SDEs. The client specifies the creation of the required interfaces in the parameters passed to *createService*.

6.1.2.1 XMLCollectionFactory::createService

Create a new Data Service handle, which represents an XML collection. The arguments are the same as those defined in the OGSi specification [OGSi] for the Factory portType.

Input

- *collectionNames* or *documentNames*: a set of collection names or a set of document names the new service should represent
- *createdPortTypes*: the QNames of the portTypes the newly created service should support

Output

- *CreatedService*: a handle to the new service

Fault(s)

- *PortTypeDoesNotExist*: The specified port types are not supported.
- *CollectionDoesNotExist*: A specified collection does not exist.
- *DocumentDoesNotExist*: A specified document does not exist.
- *Fault*: any other fault

6.2 XMLDocumentFactory

6.2.1 Service Data Declarations

No SDEs are defined in addition to those inherited from DataFactory.

6.2.2 Operations

6.2.2.1 XMLDocumentFactory::createService

Create a new Data Service representing an existing XML document in a collection.

Input

- *documentName*: the document identifier
- *CreatedPortTypes*: the QNames of the port types the newly created service should support

Output

- *CreatedService*: a handle to the new service

Fault(s)

- *PermissionDenied*

- *PortTypeDoesNotExist*: The specified port types are not supported.
- *DocumentDoesNotExist*: The specified document could not be found.
- *Fault*: any other fault.

6.3 XPathFactory

This portType allows the result of XPath queries across one or more XML resources to be represented as a Data Service.

6.3.1 Service Data Declarations

The following metadata elements are associated with the XPathFactory port type:

- *xPathVersion*: The XPath version that is supported.
- *querySchema*: The XML schema for the query parameters.

6.3.2 Operations

6.3.2.1 XPathFactory::createService

Create a new Data Service instance, which represents the results of an XPath query. A document[MA10] holding an XPath request and the interfaces to be created is passed to Factory::createService. The factory will create a Data Service holding the result of the query.

Input

- *Request*: an XPath expression, including namespace definitions.
- *Collection name* or *document names*: the names of the XML documents.
- *CreatedPortTypes*: the QNames of the interfaces (port types) that the new service should support.

Output

- *CreatedService*: a handle to the new service.

Fault(s)

- *PermissionDenied*: The client does not have sufficient access rights.
- *InvalidRequest*: The XPath request could not be parsed.
- *Fault*: any other fault.

6.4 XQueryXFactory



6.4.1 Service Data Declarations

To be determined once XQueryX 1.0 becomes a W3C recommendation. The behaviour of this portType is similar to XPathFactory.

6.4.2 Operations

6.4.2.1 XQueryXFactory::createService.

Create a new Data Service that corresponds to the results of an XQueryX request.

7. DataManagement portTypes

7.1 XMLCollectionManagement

7.1.1 Service Data Declarations

No SDEs are defined in addition to those inherited from DataManagement.

7.1.2 Operations

7.1.2.1 XMLCollectionManagement::createSubcollection

Create a new subcollection of the current collection. This creates the named collection or throws a fault if an error occurred and the collection could not be created. It does not, however, create a service that is attached to the newly created resource.

Input

- *Name*: Name of the new subcollection.
- *XMLCollectionDescription*: the metadata description of this collection.

Output

- *None*.

Fault(s)

- *PermissionDenied*: The client does not have sufficient access rights.
- *CollectionAlreadyExists*: a collection with the given description already exists.
- *Fault*: any other fault.

7.1.2.2 XMLCollectionManagement::removeSubcollection

Remove a subcollection of the collection that is being represented. If the subcollection cannot be removed the operation must throw a fault.

Input

- *Name*: the name the collection to be removed.

Output

- *None*.

Fault(s)

- *PermissionDenied*
- *NoSuchCollection*: the collection could not be found.
- *Fault*: any other fault.

7.1.2.3 XMLCollectionManagement::addSchema[MA11]S[AK12][AK13]

Associate a set of XML schemas to this collection. If a schema cannot be associated with the collection the operation must throw a fault.

Input

- *Schemas*: a set of XML Schema documents.

Output

- *None*.

Faults(s)

- *PermissionDenied*
- *SchemaAlreadyExists*.
- *Fault*: any other fault.



7.1.2.4 XMLCollectionManagement::removeSchemas

Remove a set of XML schemas from this collection.

Input

- *Names*: a set of XML Schema names.

Output

- *None*.

Faults(s)

- *PermissionDenied*
- *SchemaDoesNotExist*: the specified schemas do not exist.
- *Fault*: any other fault.

7.1.2.5 XMLCollectionManagement::bulkLoad

Load structured data into a collection (including subcollections, documents and schemas)

Input

- *Data*: data including collection structure and resources.

Output

- *Report*: .

Fault(s)

- *PermissionDenied*
- *InvalidDataFormat*: The format of the input data is not valid.
- *Fault*: any other fault.

8. Security Considerations

The XML Realisation of a Grid Data Service will use standard Grid Security mechanisms as specified by OGSA Security working group combined with standard ways of relating Grid credentials and authorities to resource access rights. The assumption is that these standards will also indicate how to make information related to authentication, authorization security etc available.

9. Conclusion

This document has discussed a specialization of the portTypes defined in the *Grid Data Service Services* [GDSS] document providing the additional capabilities required to address XML based Data Resources. This is a work in progress and feedback is welcomed on this document.

Editor Information

Mario Antonioletti,
EPCC,
The University of Edinburgh,
James Clerk Maxwell Building,
Mayfield Road,
Edinburgh EH9 3JZ,
United Kingdom.

Shannon Hastings,
Ohio State University,
333 W. Tenth Ave.,
Columbus OH, 43210,
USA.

Amy Krause,
EPCC,
The University of Edinburgh,
James Clerk Maxwell Building,
Mayfield Road,
Edinburgh EH9 3JZ,
United Kingdom.

Stephen Langella,
Ohio State University,
333 W. Tenth Ave.,
Columbus OH, 43210,
USA.

Simon Laws,
IBM United Kingdom Limited,
Hursley Park,
Winchester,
Hampshire, SO21 2JN,
United Kingdom.

Susan Malaika,
IBM Corporation,
Silicon Valley Laboratory,
555 Bailey Avenue,
San Jose, CA 95141,
USA.

Norman W. Paton,
Department of Computer Science,
University of Manchester,
Oxford Road,
Manchester M13 9PL,
United Kingdom.

Contributors

Malcolm Atkinson, NESC.
Dave Pearson, Oracle.
Greg Riccardi, Florida State University.

Acknowledgements

The DAIS Working Group of the Global Grid Forum is active, and many people have contributed to discussions within the group in recent months, including but not limited to: Bill Allcock, Vijay Dialani, Dieter Gawlick, Allen Luniewski, Sastry Malladi, Inderpal Narang, Steve Tuecke, Jay Unger, Paul Watson and Martin Westhead.

Intellectual Property Statement

The GGF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Copies of claims of rights made available for publication and any assurances of licenses to be made

available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the GGF Secretariat.

The GGF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this recommendation. Please address the information to the GGF Executive Director.

Full Copyright Notice

Copyright (C) Global Grid Forum (2003). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the GGF or other organizations, except as needed for the purpose of developing Grid Recommendations in which case the procedures for copyrights defined in the GGF Document process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the GGF or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and THE GLOBAL GRID FORUM DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE."

References

[Data Services]

I.Foster, S.Tuecke, J.Unger. *OGSA Data Services*, August 14, 2003. See: https://forge.gridforum.org/docman2/ViewProperties.php?group_id=42&document_content_id=733.

[GDSS]

M. Antonioletti, M. Atkinson, S. Malaika, S. Laws, N. W. Paton D. Pearson and G. Riccardi. *Grid Data Service Specification*. DAIS-WG Informational Draft, 9th Global Grid Forum, 19th September 2003.

[OGSI]

S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, C. Kesselman, D. Snelling, P. Vanderpilt, Open Grid Services Infrastructure, Version 1.0, <http://www.gridforum.org/ogsi-wg>, March 13, 2003.

[RFC2119]

S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*, Internet Engineering Task Force, RFC 2119, <http://www.ietf.org/rfc/rfc2119.txt>, March 1997.

[WS-Agreement]

K.Czajkowski, A.Dan, J.Rofrano, S.Tuecke, M.Xu. *Agreement-based Grid Service Management*, Version 0, June 12, 2003.

[XPath]

J. Clark and S. DeRose. *XML Path Language (XPath)*, Version 1.0 W3C Recommendation 16 November 1999. See: <http://www.w3.org/TR/xpath>.

[XQuery]

S. Boag, D. Chamberlin, M. F. Fernández, D. Florescu, J. Robie and J. Siméon. *XQuery 1.0: An XML Query Language*, W3C Working. See: <http://www.w3.org/TR/xquery/>.

[XQueryX]

A. Malhotra, J. Robie and M. Rys. *XML Syntax for XQuery 1.0 (XQueryX)*. W3C Working, See: <http://www.w3.org/TR/xqueryx>.

[XUpdate]

A. Laux and L. Martin. *XUpdate Working Draft*, last release September 14, 2000. See: <http://www.xmldb.org/xupdate/xupdate-wd.html>.

Page: 5

[m1] Need to check with the OGSA Data Service document whether the data description refers to the data service itself as well as the data resource.

Page: 5

[m2] Need to be explicit as to whether native DBMS capabilities are being exposed or whether the service can layer additional functionality which is not native to the DBMS. If, for example, XML Schemas were to be associated with a collection this would have to be done at the service layer as Xindice does not support this capability. If an XML Schema is then associated with a collection against which some documents do not validate then what is the implication for the service? Throw a fault and terminate? Not represent the documents that do not validate? Is this an implementation issue? I think the spec needs to give some guidance if the interfaces are to behave in a uniform manner.

[AK3]Referring to [m3]: That's an implementation detail which the capabilities the data service exposes and how they are implemented.

Mario: this statement worries me. Potentially the semantics/capabilities obeyed by a data resource and by the service that exposes it may be different. This could lead to all sorts of inconsistencies where the expectation is that a service extension point is expected to be persistent, e.g. association of an XML schema with a collection. On the other hand service extensibilities of the underlying data resource may be necessary.

Page: 5

[m4] How are these descriptions provided? Are they xml documents that follow a particular XML schema(s)? In order to have a uniform interface should we not give the form that this should be in?

Page: 5

[m5] Again need to be more precise? Does it give a URI pointing to where the schema may be dereferenced? The actual content of the Schemas themselves? The name of the schema file? If I were implementing I would like to know what this means.

Page: 5

[m6] I think there should ONLY BE ONE schema that can be associated with a collection unless someone can think of a counter example as to why there might be more than one schema ... I've thought of one – a collection which is composed of sub-collections and each sub-collection may be associated with a different XML schema. We then have a one-to-one relationship between collection and XML Schema BUT not one document being associated with more than one XML Schema.

[MA7] What happens if a collection contains sub-collections? Are these included in the documentNames or should there also be collectionNames for any sub collections?

[AK8]Need to be specific when the DocumentDoesNotValidate fault is thrown. When one of the documents does not validate, are the other documents added to the collection? Is this a "transaction" which only succeeds when all of the documents validate?

[MA9]If more than one document how is each document separated?

[MA10]Need to find an alternative. Agreement will not come out before DAIS has to produce a standard. Current consensus is to parameterise the terms and pass these as arguments to the factory with the view that these will become terms in an agreement once WS-Agreement comes out.

[MA11] need to specify what happens if documents in the collection do not validate with the new schema – presumably the operation fails. Also, is this something that is persistent? If a schema is associated with a Xindice collection does that only apply for the duration of the session? Does this make sense or is there an expectation that the service will remember the schema?

[AK12]This is a persistent operation. It is performed on the underlying data resource.

Mario: this worries me then. It means that for Xindice you would have to store the schema somewhere (possibly in the collection) but then you are imposing a constraint on the usage of that data resource that is not intrinsic to the data resource. I do not think that it can be a persistent modification unless it is supported by the underlying data resource.

[AK13]We have to specify whether the operation fails if one schema cannot be added.