



A WS-DAI implementation using OGSA-DAI

Elias Theodoropoulos and Mike Jackson
DAIS Working Group Session – OGF22
Wednesday, 27th February 2008

Outline

- Implementing WS-DAI with OGSA-DAI
- General issues and ambiguities about WS-DAI
- OGSA-DAI related issues
- Conclusions

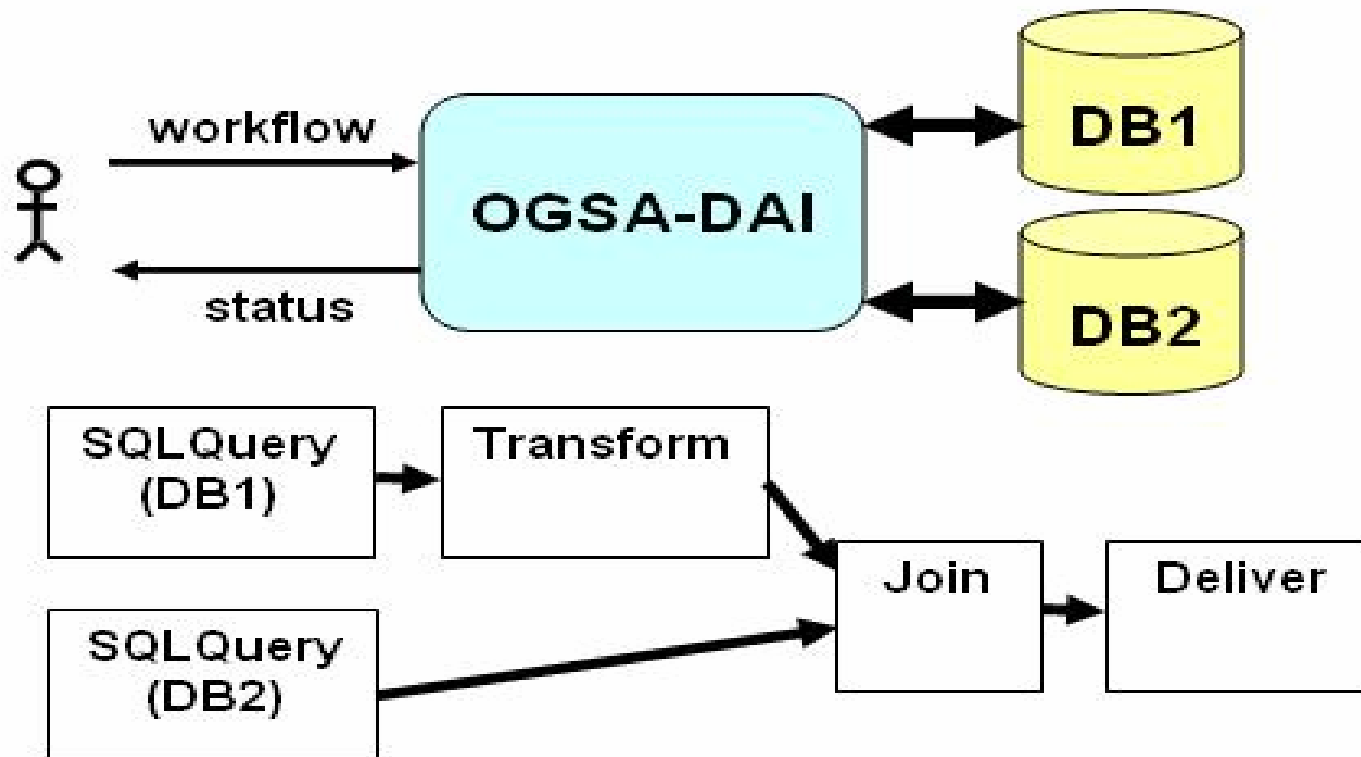


Implementing WS-DAI with OGSA-DAI

Roadmap

- Started work on October 2007.
 - Currently 1.5 FTEs working on the WS-DAI implementation.
- Started by designing common components for DAIX and DAIR
- Now focused on a DAIX implementation.
 - Expect a working version of DAIX in March 2008.
 - Expect a working version of DAIR in June 2008.

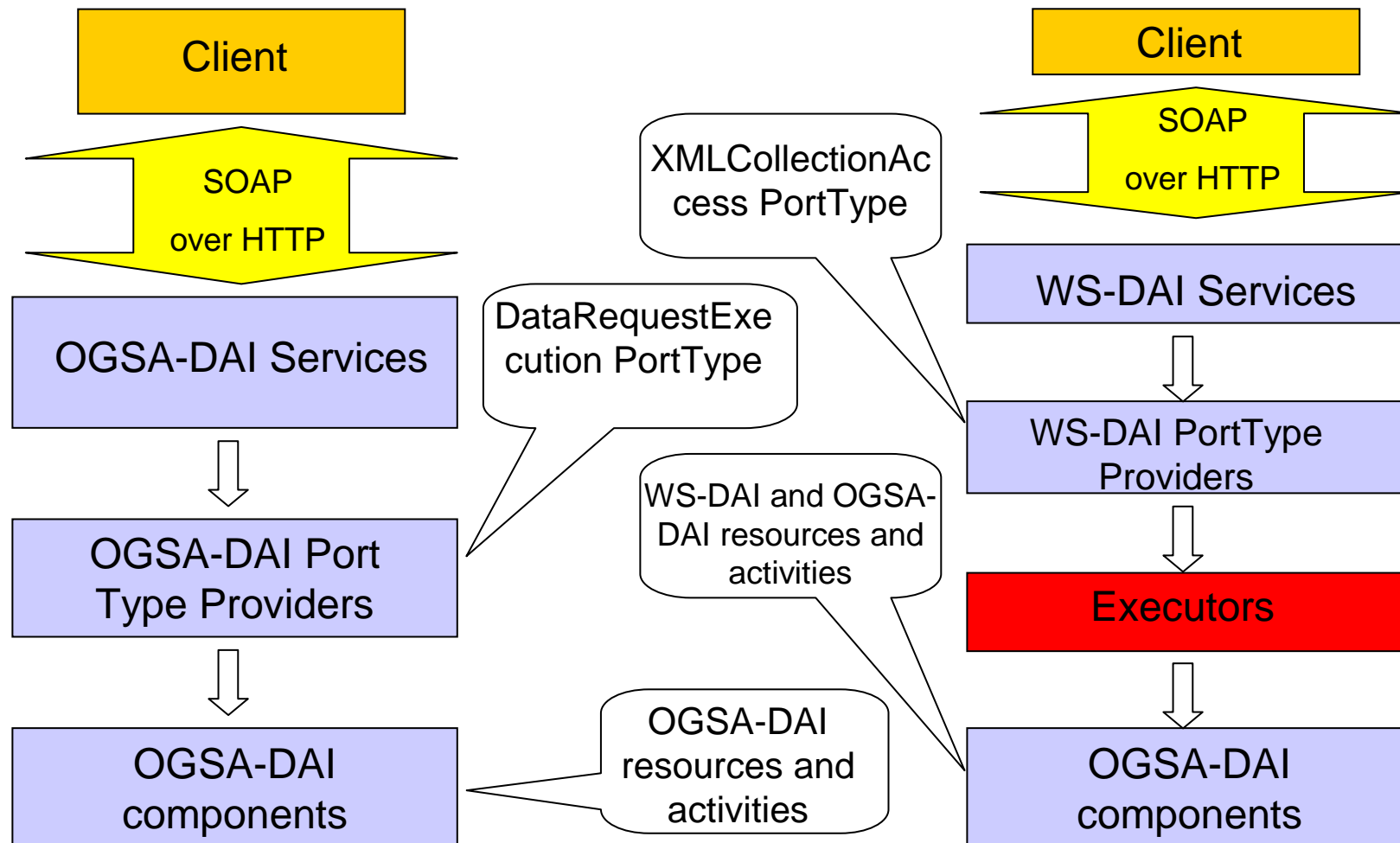
OGSA-DAI in Operation



WS-DAI operations

- A web service offers a number of operations. Operations of similar functionality are part of the same portType.
 - XMLCollectionAccess: AddDocuments, RemoveCollection, CollectionSelectionFactory, ...
 - SQLResponse: GetSQLResponseItem, GetSQLRowset, GetSQLUpdateCount, ...

OGSA-DAI 3.0 vs WS-DAI architecture

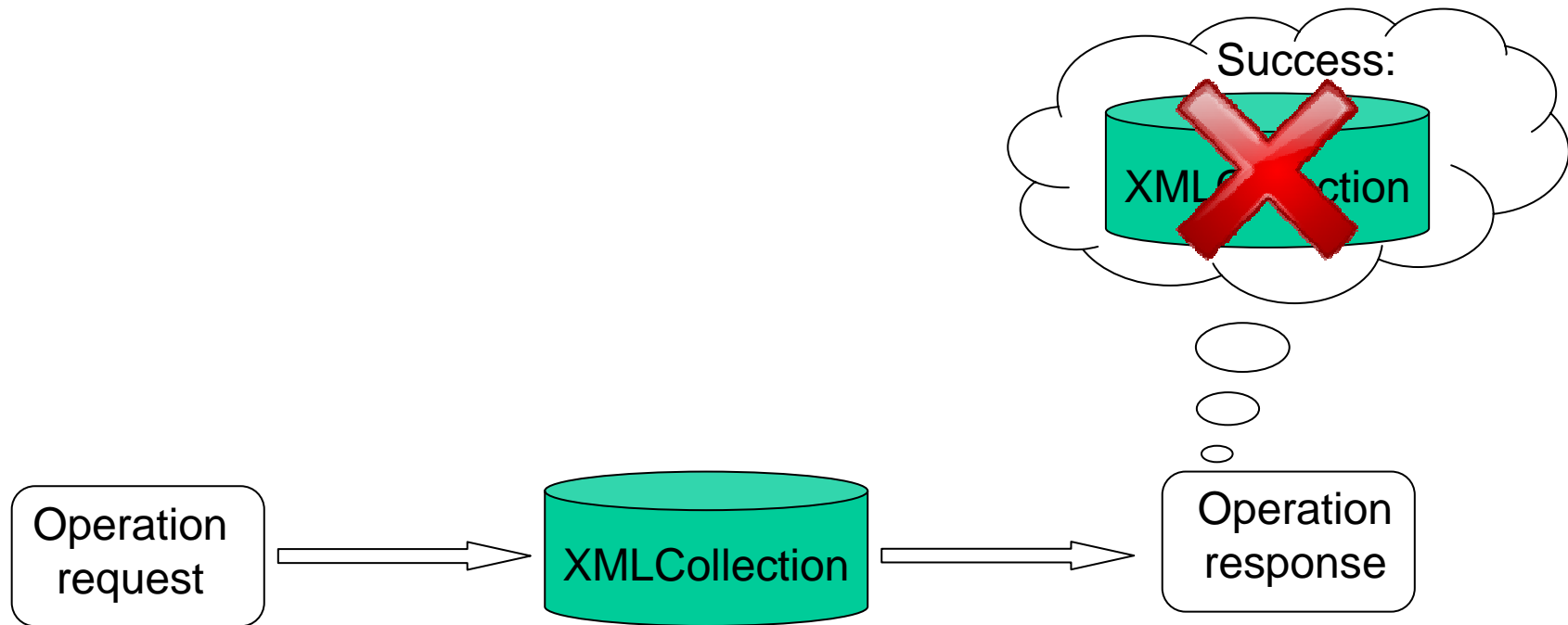


Executors

- Executors implement WS-DAI specification operations.
- The executors:
 - Either delegate the execution to resources
 - Or formulate OGSA-DAI workflows to be executed that involve resources.

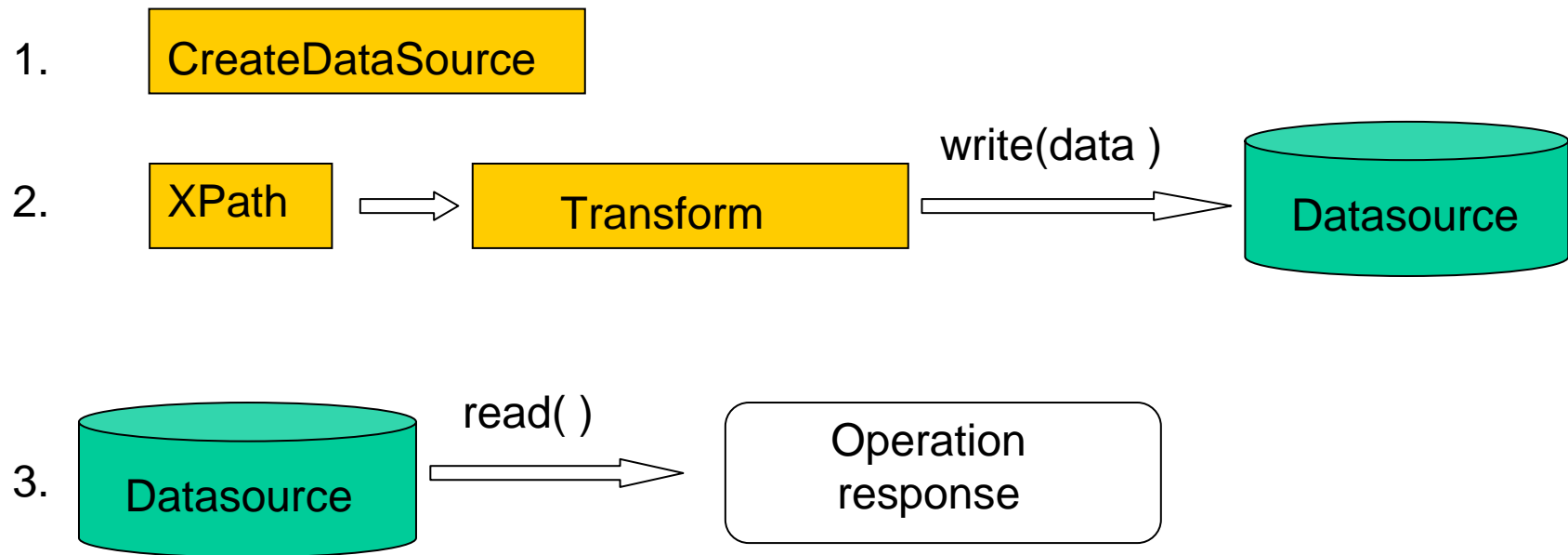
Delegation to resources : An example

- CoreDataAccess:Destroy



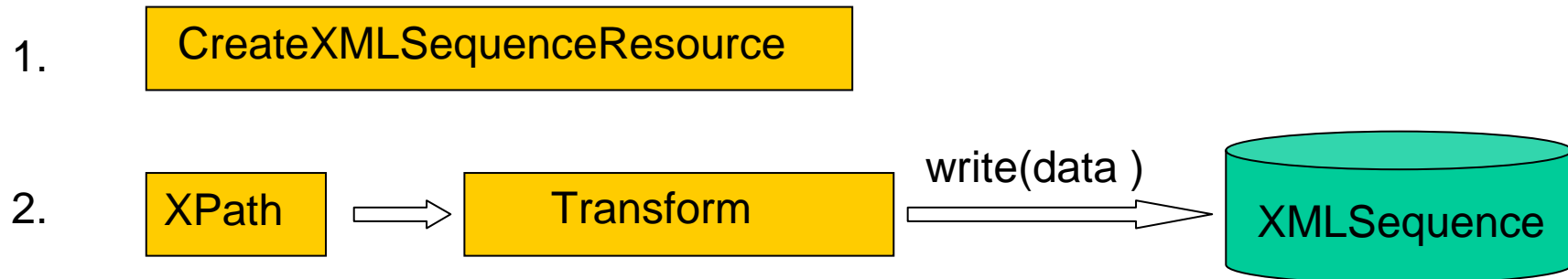
Involving resources: Workflow example(1/2)

- **XPathAccess:XPathExecute**

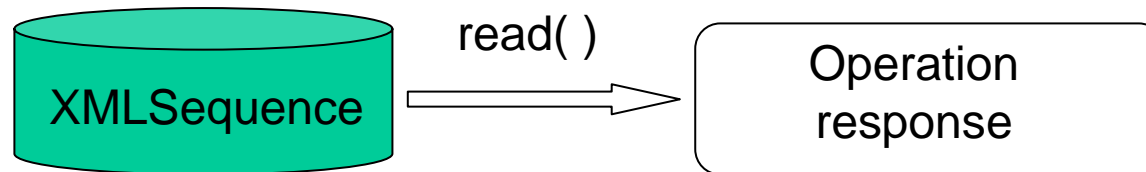


Involving resources: Workflow example(2/2)

- **XPathFactory:XPathExecuteFactory**



- **XMLSequenceAccess:GetItems**





General issues and ambiguities about WS- DAI

Memory overhead(1/2)

- DAIR:SQLRowSetAccess::GetTuples requires all tuples to be retrieved and the number of tuples exposed as a property.
- DAIX:XMLSequenceAccess:GetItems requires all data to be retrieved before being exposed by the XMLSequence resource.

Memory overhead(2/2)

- In both cases there is a performance overhead to retrieve all data.
- More importantly, all the requested data must be streamed out of the DB and saved elsewhere. This means:
 - Difficult to exploit streaming functionality of DB
 - Intermediate storage becomes necessary and incurs memory and storage overheads.
 - This can be in-memory storage or in files or in a temporary DB.
 - Scalability becomes an issue if large amounts of data need to be retrieved.

JDBC Challenges(1/2)

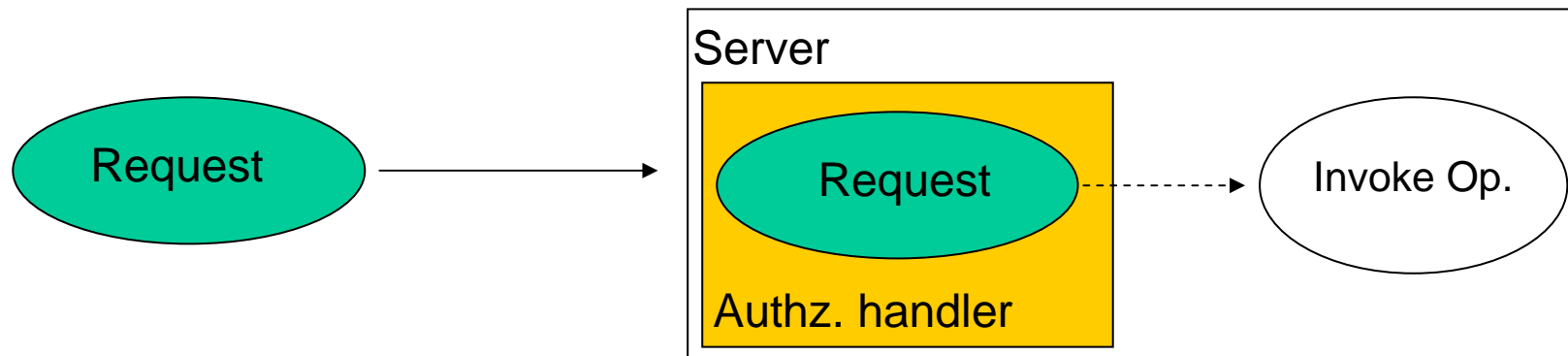
- JDBC ResultSet object does not provide directly the number of rows.
- DAIR:
 - requires the number of rows as part of the SQLRowset property document.
 - permits getting intermediate rows from the Resultset.
- It becomes a necessity to retrieve all the rows. This can pose a scalability issue.

JDBC Challenges(2/2)

- JDBC Statement object by default supports having only one ResultSet open at a time.
- DAIR effectively allows access
 - in multiple ResultSets of the same statement execution,
 - interchangeably,
 - through the creation of two or more SQLRowSet resources.
- It becomes a necessity to stream all data from the DB and store them temporarily elsewhere

Authorisation

- “NotAuthorizedFault” assumes that authorisation takes place during the operation invocation.
- Some WS containers and some security protocol implementations support authorisation before the operation invocation.



- The fault becomes obsolete.

Resources Opacity

- WS-Addressing recommends the opacity of the reference parameters of a WS-EPR.
- However, “when a data resource address is returned ..., the EPR Reference parameter **MUST** contain the DataResourceAbstractName that identifies the data resource...”

CreateSubcollection operation(1/3)

- SubcollectionName URI: Should it be an absolute URI or a relative URI to be appended to the collection path of the resource?
 - E.g. Collection book2 should be provided as daix:/db/littleblackbook/book2 or daix:book2?
- We opted for the latter.

CreateSubcollection operation(2/3)

- SubcollectionName URI: What if the URI contains intermediate collections that do not exist
 - e.g. Client provides the following:
daix:book1/book2 in order to add book2 but book1 doesn't exist.
- Do we throw a fault or create the intermediate collections?
- XMLDB API supports incremental creation of collections so we do too.

CreateSubcollection operation(3/3)

- CollectionName URI: Should it be an absolute URI or not?
 - We opted for absolute URI since this is the way it is exposed in the property document.
 - Furthermore, if they were relative, the client would have to parse them.

RemoveSubcollection operation

- SubcollectionName URI: What happens if the sub-collection to be removed doesn't exist?
- Is it a fault or a successful response?
 - We opted for a successful response since the InvalidCollectionFaultName doesn't provide an argument for the invalid collection name – so as to differentiate between invalid collection and subcollection name.

AddDocuments operation

- XMLCollectionAccess:AddDocuments response may include a value indicating whether an
 - “Existing document of same name is overwritten”
- Does this mean?
 - The existing document WAS overwritten?
 - The existing document WOULD BE overwritten so the new document was not added?
 - We haven't implemented this yet.

Factory operations

- Possibly rename factory operations to be verbs (as this is the convention in, for example, Java):
- For example:
 - SQLExecuteFactory => CreateSQLResponse
 - SQLRowsetFactory => CreateSQLRowset
 - CollectionSelectionFactory => CreateXMLCollection
 - DocumentSelectionFactory => CreateXMLSequence
 - XPathExecuteFactory => CreateXMLSequenceFromXPath





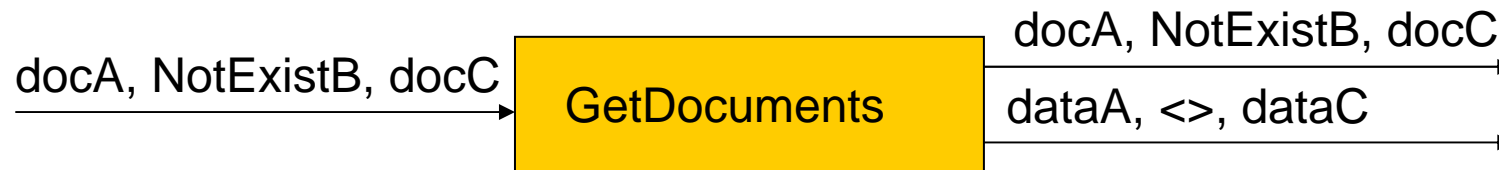
OGSA-DAI related issues

Exploiting OGSA-DAI's streaming model

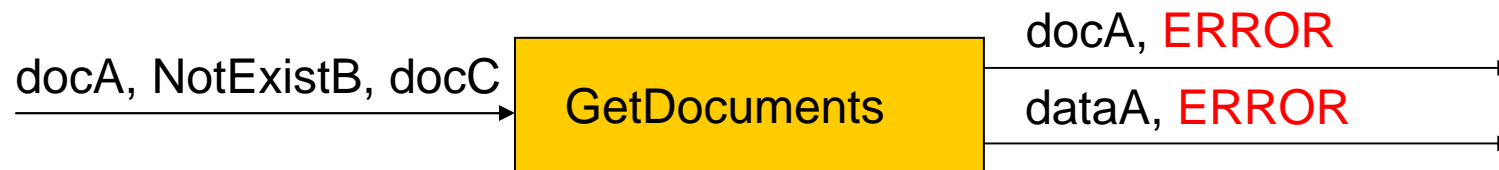
- WSDAI allows to access part or all of the data multiple times
- OGSA-DAI streams the data:
 - From the DB through the OGSA-DAI server to the client - if the DB supports streaming.
 - When the clients request the data
 - As much as the client asks for.
- Data cannot be re-streamed within OGSA-DAI, so all data need to be retrieved and stored.

Failures' handling

- XMLCollectionAccess::GetDocuments operation:
 - WS-DAI approach:

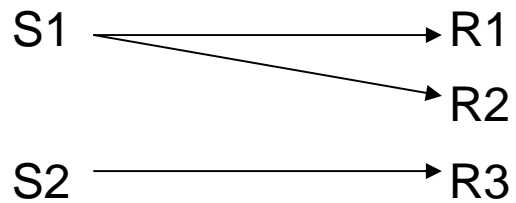


- OGSA-DAI approach:

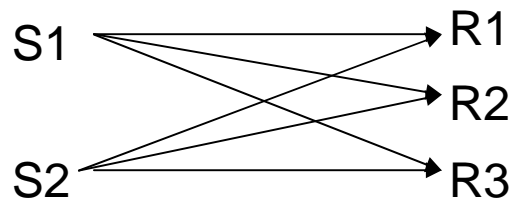


Service-resource coupling

- WS-DAI approach:



- OGSA-DAI approach:



Conclusions

- A number of specification ambiguities were identified during the implementation.
- A challenge to address: Scalability due to memory overhead and JDBC constraints.
- Implementation proceeds smoothly to date without major obstacles.