



# Distributed Database Access and Data Stream Management

## Integration of Streaming and Persistent Data Management<sup>1</sup>

Deliverable	D4.7
Authors	Angelika Reiser, Tobias Scholl
Editors	Tobias Scholl
Date	August 18, 2008
Document Version	1.0.0
Current Version	1.0.0
Previous Versions	0.1.0,0.7.0

### **A: Status of this Document**

Deliverable D7 of working group 4.

### **B: Reference to project plan**

Seventh deliverable of working group *Distributed Database Access and Data Stream Management*.

---

<sup>1</sup>This work is part of the AstroGrid-D project and D-Grid. The project is funded by the German Federal Ministry of Education and Research (BMBF).

**C: Abstract**

This deliverable describes the integration of the AstroGrid-D data stream management with the services provided by OGSA-DAI to access persistent data.

We describe the functionality to retrieve data from a database management system into the data stream management network and to store streaming data into relational database systems via OGSA-DAI.

**D: Change History**

<b>Version</b>	<b>Date</b>	<b>Name</b>	<b>Brief summary</b>
1.0.0	18.08.2008	Tobias Scholl	First release.
0.7.0	05.08.2008	Tobias Scholl, Angelika Reiser	Major updates.
0.1.0	20.08.2007	Tobias Scholl	Initial draft.

E:

## Contents

<b>Abstract</b>	<b>2</b>
<b>Change History</b>	<b>3</b>
<b>1 Introduction and Motivation</b>	<b>5</b>
<b>2 A Relational Content Server (from DB to DSMS)</b>	<b>6</b>
2.1 Configuration . . . . .	6
2.2 Example . . . . .	7
<b>3 A Relational StreamIterator (from DSMS to DB)</b>	<b>8</b>
3.1 Configuration . . . . .	8
3.2 Example . . . . .	8
<b>4 Supporting OGSA-DAI 3.0</b>	<b>10</b>
<b>References</b>	<b>11</b>

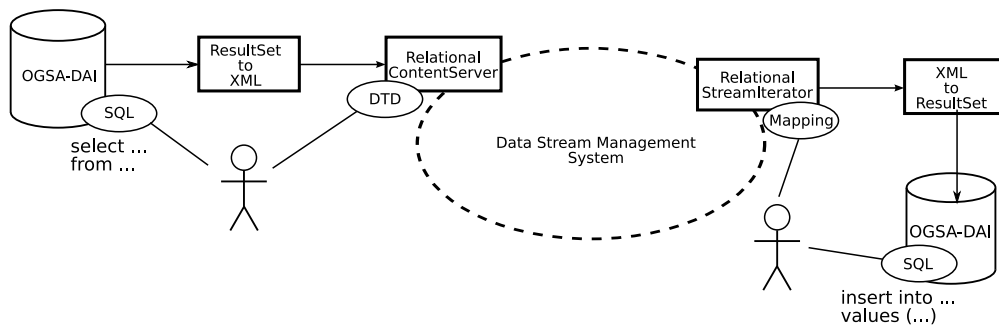


Figure 1: Overview of the integration of persistent data with the data stream management.

## 1 Introduction and Motivation

The data stream management system (DSMS) (see [4, 1, 6, 5]) for AstroGrid-D is designed to cope with continuous data streams. Users may publish and/or subscribe to streams. The DSMS takes care of efficiently processing (in general filtering) the streams, optimizing the operational flow within the network and sending the result streams to the subscribers. Much effort has been spent to tune the DSMS with respect to load balancing, performance optimization, multi-stream processing, and a lot more. On the other hand, persistent data stored in databases can also be processed in AstroGrid-D (see [4, 3, 7]) with the use of the Open Grid Services Architecture Data Access and Integration (OGSA-DAI) services framework. This deliverable describes how the two access and processing possibilities, via data streams and via databases can be combined. This leads to two interfaces:

- the Relational Content Server, which provides access to a relational database for streaming data queried from the database into the DSMS and
- the Relational StreamIterator, which inserts data streams into a database. 0

The Relational Content Server takes a query to a database and a *Document Type Definition (DTD)* to represent the result set of the query as an XML file. After transforming the result set to the specified XML format, this data serves as input for the DSMS. On the other hand, the Relational StreamIterator stores XML data into relational data tuples. Given a mapping between the XML and relational formats, the data elements are inserted into the database.

The possibility of processing database tuples via the DSMS and of storing elements of data streams persistently into databases has three main advantages:

1. Many eScience applications, especially in Astrophysics, process continuous data streams with combining them with persistent data, e.g. comparing observed objects with elements from persistent catalogs, existing of either other observations or simulations. Another example for combining streams with persistent data is calibration of scientific instruments.

2. Some workflows on persistent data are very complex and benefit from the sophisticated optimization and distribution of the work load into the network via the DSMS.
3. The user who subscribes to a data stream, usually with a strong filter so that only a small fraction of the whole stream fulfills the subscription predicate, wants to store this result persistently.

In the following we describe in detail the configuration of the Relational Content Server and the Relational StreamIterator. More details on the actual implementation can be found in the literature [8].

## 2 A Relational Content Server (from DB to DSMS)

The relational content server provides access to a relational database table for streaming it into the data stream management system.

Using the OGSA-DAI services allows us to get all the benefits described in our Deliverable on access to persistent data [3].

When accessing a relational table (or in general the result of an SQL query), we need to specify how the individual tuples and their fields are represented in an XML structure.

### 2.1 Configuration

Some parameters for the content server setup are the same as for publishing data files via the existing file-based ContentServers. These parameters are the *dtd*, which provides the DTD for the XML stream that the ContentServer will feed into the data stream network. Furthermore *stream.server.port* specifies the TCP-port used by the ContentServer to publish the data; *stream.sleep.time* configures the waiting time interval of the ContentServer before the next data block is transmitted; *stream.injection.mode*, finally, defines whether the content server should immediately start serving its contents or is started manually.

Besides these parameters, new configuration parameters have been introduced in order to support relational data sources for ContentServers. These are now described in more detail.

**serviceHandler** The URL of the OGSA-DAI service. The OGSA-DAI service running in at the specified URL has an exposed resource with the identifier *resourceId* and is accessible within your community Grid.

**resourceId** The exposed identifier of the OGSA-DAI resource. This resource contains the data (i. e., tables) to be accessed with the specified *sqlQuery*.

**sqlQuery** The SQL query which will be issued to the OGSA-DAI resource. Basically, any valid SQL query is permitted. OGSA-DAI on purpose does not provide translation techniques for the individual SQL dialects of its relational resources. However, it offers enough meta-data that an application can differentiate (if required) between the different database vendors.

**xmlHeader** The header of the XML data stream. Most common use will be a single XML element (together with an optional XML preamble).<sup>2</sup>

**xmlTemplate** The XML structure template that will be created for each database tuple. It contains placeholders for all database fields, that should be included into the data stream. The field is referenced by a *#fieldname#*. For example, the field *det-time* is reference with *#det-time#*.

**xmlFooter** Configures the closing XML tag for the data stream and is the corresponding closing tag for the tag defined in *xmlHeader*.

## 2.2 Example

The following XML fragment of a scenario configuration shows the definition of data streams that are provided into the data stream network. It describes the configuration parameters explained above. In order to show the similarities and differences between the various kinds of data streams, we included a file-based (*xmlfile*) and db-based (*db*) ContentServer.

```
<streams>
  <kindDefinition>
    <kind name="xmlfile" class="streamglobe.ws.cp.impl.XMLFileContentServer"/>
    <kind name="db" class="streamglobe.ogsadai.RelationalContentServer"/>
  </kindDefinition>
  <stream sid="stream-0" type="xmlfile">
    <dtd filename="/path/to/dtd/vela_nested.dtd"/>
    <param name="stream.filename">
      /path/to/xmlfile/vela_demo.xml
    </param>
    <param name="stream.server.port">9009</param>
    <param name="stream.sleep.time">1</param>
    <param name="stream.injection.mode">>manual</param>
  </stream>
  <stream sid="stream-1" type="db">
    <dtd filename="/path/to/dtd/vela_nested.dtd"/>
    <param name="serviceHandler">
      http://bladekemper21:8080/wsrf/services/ogsadai/DataService
    </param>
    <param name="resourceId">DataServiceResource</param>
    <param name="sqlQuery">select * from vela</param>
    <param name="xmlHeader"><![CDATA[<photons>]]></param>
```

<sup>2</sup>In order to directly specify XML tags within the element without the need of escaping `<` or `>`, `<![CDATA[...]]>` can be used.

```

        <param name="xmlTemplate"><![CDATA[
<photon>
  <coord>
    <cel><ra>#ra#</ra><dec>#dec#</dec></cel>
    <det><dx>#dx#</dx><dy>#dy#</dy></det>
  </coord>
  <phc>#phc#</phc><en>#en#</en><det-time>#det-time#</det-time>
</photon>]]></param>
        <param name="xmlFooter"><![CDATA[</photons>]]></param>
        <param name="stream.server.port">9090</param>
        <param name="stream.sleep.time">1</param>
        <param name="stream.injection.mode">manual</param>
    </stream>
</streams>

```

### 3 A Relational Streamlterator (from DSMS to DB)

The complement to the relational ContentServer is the relational Streamlterator. The Streamlterator provides means to store a data stream into relational database system.

#### 3.1 Configuration

The extraction parameters for the individual database columns are specified as *variables*. The specified *select*-paths are extracted and parsed as the specified types. These values are then inserted into the database.

Additionally, the following parameters are available for configuring the relational streamlterator.

**SERVER\_URL** The service URL of the OGSA-DAI service. As in Section 2.1, this is the OGSA-DAI services that provides access to a community data source.

**RESOURCE\_NAME** The name of the published OGSA-DAI resource, containing the *TABLE*, where the data will be stored in.

**TABLE** This parameter specifies the name of the database table.

**CREATE\_TABLE (optional)** If set to *TRUE*, the database table will be created if it does not yet exists.

#### 3.2 Example

In order to use the relational Streamlterator, you need to configure a *streamoperator* element in data stream query execution plan.

The following configuration basically stores the data stream created in Section 2.2 back into a relational database table called *vela*.

```

    <streamoperator id="sqlplan-0" codebase="file:/path/to/starglobe-operators.jar"
      name="streamglobe.ogsadai.RelationalStreamIterator22"
      xsi:type="externalStreamoperatorType">
      <dependencies>
        <stream id="stream-0"/>
      </dependencies>
      <authorizedby>Michael Zech</authorizedby>
      <inputstreamdata id="stream-0">
        <dtd><![CDATA[
<!ELEMENT photons (photon)* >
<!ELEMENT photon (coord, phc, en, det-time)>
<!ELEMENT coord (cel, det)>
<!ELEMENT cel (ra, dec)>
<!ELEMENT ra (#PCDATA)>
<!ELEMENT dec (#PCDATA)>
<!ELEMENT det (dx, dy)>
<!ELEMENT dx (#PCDATA)>
<!ELEMENT dy (#PCDATA)>
<!ELEMENT phc (#PCDATA)>
<!ELEMENT en (#PCDATA)>
<!ELEMENT det-time (#PCDATA)>
]]>
        </dtd>
        <variable name="ra" select="./coord/cel/ra" type="Double"/>
        <variable name="dec" select="./coord/cel/dec" type="Double"/>
        <variable name="dx" select="./det/dx" type="Integer"/>
        <variable name="dy" select="./det/dy" type="Integer"/>
        <variable name="phc" select="./phc" type="Integer"/>
        <variable name="en" select="./en" type="Double"/>
        <variable name="det-time" select="./det-time" type="Double"/>
      </inputstreamdata>
      <outputstreamdata>
        <dtd />
      </outputstreamdata>
      <parameter>
        <key>SERVER_URL</key>
        <value>http://127.0.0.1:8080/wsrf/services/ogsadai/DataService</value>
      </parameter>
      <parameter>
        <key>RESOURCE_NAME</key>
        <value>DataServiceResource</value>
      </parameter>
      <parameter>
        <key>TABLE</key>
        <value>vela</value>
      </parameter>
      <!--
      <parameter>
        <key>CREATE_TABLE</key>
        <value>TRUE</value>
      </parameter>
      -->
    </streamoperator>

```

## 4 Supporting OGSA-DAI 3.0

In September 2007, the OGSA-DAI team released OGSA-DAI 3.0. It is a complete redesign and rewrite of the OGSA-DAI core libraries. More details about the changes from OGSA-DAI 2.2 to OGSA-DAI 3.0 and the rationale behind the redesign are summarized in a paper by the OGSA-DAI team [2]. Members of the AstroGrid-D working group 4 contributed as beta testers in order to familiarize with the new interfaces, to learn the necessary migration steps, and to discuss with the OGSA-DAI developers. As the OGSA-DAI developers took great care in keeping the changes to the client interfaces as minimal as possible, it is feasible to provide the according ContentServers and StreamIterators in a future release.

## F: References / Bibliography

### References

- [1] A. Carlson and T. Scholl. Distributed Database Access and Data Stream Management: Prototype for Manual Query Execution Plans. Deliverable D4.2, AstroGrid-D project, August 2006. <http://www.gac-grid.de/project-documents/deliverables/wp4/wg4-d2-1.0.0.pdf>.
- [2] M. Antonioletti, N. P. Chue Hong, A. C. Hume, M. Jackson, K. Karasavvas, A. Krause, J. M. Schopf, M. P. Atkinson, B. Bobrzelecki, M. Illingworth, N. McDonnell, M. Parsons, and E. Theocharopoulos. OGSA-DAI 3.0 – The Whats and the Whys. In *Proceedings of the UK e-Science All Hands Meeting*, September 2007. [http://www.ogsadai.org.uk/documentation/publications/AHM2007\\_OGSA-DAI\\_WhatAndWhy.pdf](http://www.ogsadai.org.uk/documentation/publications/AHM2007_OGSA-DAI_WhatAndWhy.pdf).
- [3] T. Scholl. Distributed Database Access and Data Stream Management: Access to Persistent Data. Deliverable D4.3, AstroGrid-D project, November 2006. <http://www.gac-grid.de/project-documents/deliverables/wp4/wg4-d3-1.0.0.pdf>.
- [4] T. Scholl. Distributed Database Access and Data Stream Management: Requirements Specification and Architectural Design. Deliverable D4.1, AstroGrid-D project, May 2006. <http://www.gac-grid.de/project-documents/deliverables/wp4/wg4-d1-1.0.0.pdf>.
- [5] T. Scholl and A. Reiser. Distributed Database Access and Data Stream Management: Data Stream Management–Deployment Report. Deliverable D4.5, AstroGrid-D project, August 2007. <http://www.gac-grid.de/project-documents/deliverables/wp4/wg4-d5-1.0.0.pdf>.
- [6] T. Scholl and A. Reiser. Distributed Database Access and Data Stream Management: Distributed Function Providers. Deliverable D4.4, AstroGrid-D project, February 2007. <http://www.gac-grid.de/project-documents/deliverables/wp4/wg4-d4-1.0.0.pdf>.
- [7] T. Scholl and A. Reiser. Distributed Database Access and Data Stream Management: Optimization for Distributed Queries. Deliverable D4.6, AstroGrid-D project, May 2008. <http://www.gac-grid.de/project-documents/deliverables/wp4/wg4-d6-1.0.0.pdf>.
- [8] M. Zech. Design und Implementierung einer bidirektionalen Anbindung des StreamGlobe/StarGlobe Projekts an relationale Datenbanken unter Verwendung von OGSA-DAI. Diplomarbeit, Technische Universität München, Germany, 2008.