

## Data Distribution in the Grid Environment

### Status of This Memo

This memo provides information to the Grid Data Access and the Grid Data Replication communities. It does not define any standards or technical recommendations. Distribution is unlimited.

### Copyright Notice

Copyright © Global Grid Forum (2002). All Rights Reserved.

### **Abstract**

Data Grids [DATA] are distributed environments where applications access, manage and distribute data and events in a timely fashion at a very large scale and potentially across organizational boundaries. Challenges arise from the size of the data, the geographical distance, and the number of involved parties (such publishers and consumers). Data Grids need to protect the critical data (security, authorization, auditing, etc.) and need to guarantee transparent and efficient access to and distribution of data with the required Quality Of Service (QOS).

"Grid Data Distribution" - here after known as (**GDD**) model - answers these requirements. The GDD model supports dynamic and efficient data distribution of customized information mainly based on consumer's needs.

The outline of the paper is as follows. First we provide a definition of the GDD components. Second we position GDD functionality with respect to Grid Data Access and Notification. Then we describe in detail the operations that are introduced to support the new functionality, which are defined as extensions of the existing Data Access specifications and we describe the use case scenarios of GDD for Data Replication and 3<sup>rd</sup> party Data delivery. Finally, we outline the remaining work and open issues in this area.

### Contents

Data Distribution in the Grid Environment .....	1
Status of This Memo .....	1
Copyright Notice .....	1
Abstract.....	1
1.    Introduction.....	3
1.1    The Grid Data Distribution model.....	3
1.2    Grid Data Distribution and Grid Data Service Specification (DAIS) .....	4
1.3    Grid Data Distribution and OGSI Notification.....	4
2.    Definition.....	4

Dieter Gawlick, Oracle Corporation  
 Vitthal Gogate, IBM Almaden Research Center  
 Cecile Madsen, IBM Silicon Valley Laboratory  
 Shailendra Mishra, Oracle Corporation  
 Inderpal Narang, IBM Almaden Research Center

Category: Informational  
 OGSA-DAI

Mahadevan Subramanian, IBM Almaden Research Center  
 10/02/2003

2.1	Data Publication .....	4
2.2	Data Subscription .....	4
2.3	Event and Data Propagation .....	5
3.	Overview of the GDD Interface .....	5
3.1	Terminology .....	5
3.2	The GDD Interface for Administrative and Operational Tasks .....	6
3.3	Bulk Data Transfer and Data Distribution .....	7
4.	The GDD PortType, Service Data Elements and Operations .....	7
4.1	GDD PortType .....	7
4.2	GDD Service Data Declarations .....	8
4.3	GDD Operations .....	10
5.	Scenarios .....	22
5.1	Asynchronous SQL Result Set Delivery to 3rd party consumers .....	22
5.2	Publication On Request: Asynchronous SQL Result Set Delivery .....	24
5.3	Workflow .....	25
5.4	Replication .....	28
6.	Open Issues .....	29
7.	Security Considerations .....	30
	Author Information .....	30
	Glossary .....	<b>Error! Bookmark not defined.</b>
	Intellectual Property Statement .....	31
	Full Copyright Notice .....	31
	References .....	31

## 1. Introduction

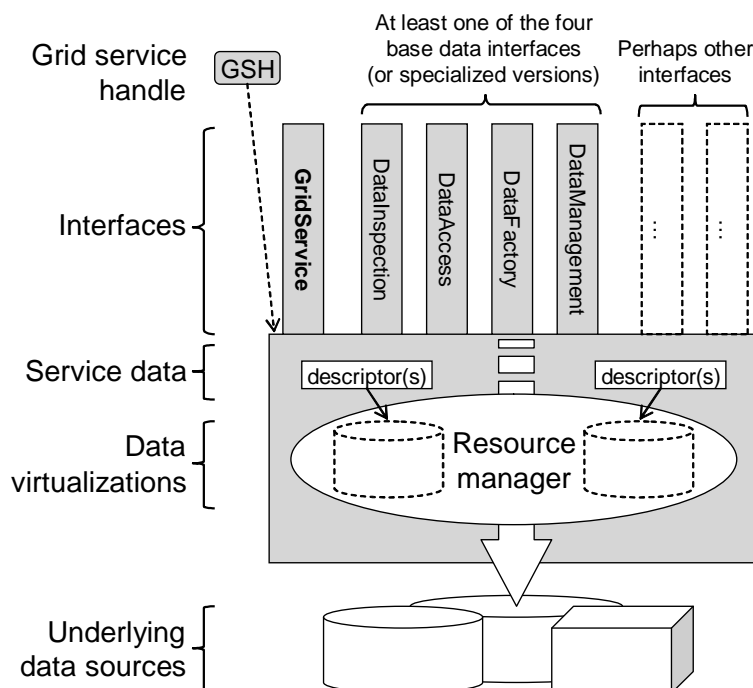
The Grid Data Distribution model allows users to share data between publishers and subscribers in a timely fashion. Publishers are considered to be the sources of data (information) while subscribers are considered to be the consumers of data (information). Data can be pushed or pulled from the publisher to the subscriber. The publisher can also alert subscribers of the existence of data.

In this model, publishers and subscribers can control how information is published, distributed, and consumed. The model supports efficient asynchronous distribution of data, so publishers don't necessarily need to know of the target recipients and vice-versa.

### 1.1 The Grid Data Distribution model

To support the Grid Data Distribution model, the Grid Data Distribution interface (GDD interface) is introduced.

The GDD interface is defined as an extension of the Data Management interface of the OGSA Data service [Data Services] as shown in the figure below.



**Figure 1: The OGSA Data service.**

The GDD interface operations include both *administrative* tasks (e.g. define rules for Data Publication and Data Subscription) and *operational* tasks. In essence, this new functionality

provides an extension to Data Access and OGSi Notification interfaces by introducing a new interface, the GDD interface.

### 1.2 Grid Data Distribution and Grid Data Service Specification (DAIS)

We propose the following scoping of Data Distribution with respect to DAIS functionality: define both synchronous and asynchronous data access “*operations by reference*” as mentioned in [DAIS], to be derived as a special case of Data Distribution.

For example, in DAIS specifications, the “sqlQueryByRefASync” and “sqlQueryByRefSync” operations can now be covered through data distribution model.

### 1.3 Grid Data Distribution and OGSi Notification

The Data Distribution model is a subscription-based model that extends the existing OGSi Notification specification [OGSi] for Data service to support:

- Event-based, schedule-based or continuous Data Distribution.
- Efficient scaling to a very large number of subscribers
- Propagation and consumption of data with transactional consistency
- Efficient proprietary data distribution mechanisms, as provided by existing vendors. This may imply dedicated data channels between data source and target
- Existing distribution topologies, eg publish/subscribe, brokering, pull models, etc.
- Controlled security, authorization and auditing of the distributed data.

## 2. Definition

The Grid Data Distribution interface allows users to distribute data between *publishers* and *subscribers*. Publishers are sources of data. Subscribers are typically consumers of data.

Data can be pushed or pulled from the publisher to the subscriber. Also, the publisher can alert subscribers of the existence of data of interest.

The proposed framework allows for data movement between Data service to Data service, client to Data service and Data service to client; its components include data publication, data subscription, event and data propagation and finally event and data consumption, as described below.

### 2.1 Data Publication

*Data Publication* occurs at the Data service which acts as the logical source of data. The Data publication rules may include whether and what data is to be published and information about the *publisher*. Data publication specifications includes QoX (Quality of Service, Quality of Protection, etc.), auditing and non-repudiation requirements.

### 2.2 Data Subscription

*Data Subscription* occurs at the Data service which acts as the logical data source. The Data subscription specifies a rule or a set of rules seeking matching data of interest to the subscriber and information about the *subscriber*.

Data subscription specifications include QoX, auditing and non-repudiation requirements.

## 2.3 Event and Data Propagation

Event and Data propagation rules define how to propagate events and data to a *consumer*. The consumer may be a client, or another Data service. These rules include propagation protocols, auditing and retention requirements and may further include to whom published data is to be delivered and how.

## 2.4 Event and Data Consumption

Event and Data Consumption rules define how the consumer consumes events and data. These rules include for example auditing and retention requirements.

# 3. Overview of the GDD Interface

This section defines the terms used throughout the Interface and Operations sections of this document, introduces the GDD interface and the various operations it supports and finishes with a quick discussion on direct bulk transfer.

The GDD interface supports both administrative and operational tasks. Administrative tasks are invoked by GDD Administrators who setup and maintain GDD metadata (rules). Operational tasks by contrast do not manipulate GDD metadata; rather they are invoked by publishers, subscribers or consumers to publish, retrieve or consume GDD Data or GDD Events.

## 3.1 Terminology

The following terminology is used throughout this document and more specifically in the Operations section:

GDD Data is the result data of publication, subscription, propagation, or consumption operations. It may be an encoded dataset, as defined in OGSA Data service.

GDD Event is an event resulting from a publication, subscription, propagation, or consumption operations. It may be an encoded dataset, as defined in OGSA Data service.

A publication rule is an XML element that determines which data can be published by a publisher to a Data service instance by a client or by a Data service.

A subscription rule is an XML element that determines which data is of interest to a subscriber.

A propagation rule is an XML element that determines how the data or event of interest is to be propagated from a Data service to a consumer. The propagation rules may also determine when, where and how (protocol) data should be sent based on what gets subscribed at the Data service.

A consumption rule is an XML element that determines how events and data are to be consumed by a consumer.

The specification of the publication, subscription, propagation and consumption rules will be defined at a later time.

### 3.2 The GDD Interface for Administrative and Operational Tasks

The GDD interface supports both *Administrative* and *Operational* tasks.

The administrative tasks for publishing data consist of:

- the *createPublication* operation, which creates publication rules for a publisher of GDD data and GDD events at a Data service.
- the *alterPublication* operation, which alters publication rules for a publisher of GDD data at a Data service. This operation returns a new publication ID.
- the *startPublication* operation, which enables publication of data to/at a Data service and defines rules for starting the publication.
- the *stopPublication* operation, which disables publication of data and/or event to/at a data source. A publication can be re-enabled by invoking *startPublication*.
- the *dropPublication* operation, which releases resources associated with a given publication.

The administrative tasks for subscribing to data and events consist of:

- the *createSubscription* operation, which creates subscription rules for a subscriber of GDD data and GDD events at a Data service.
- the *alterSubscription* operation, which alters subscription rules for a subscriber of GDD data or GDD events at a Data service. This operation returns a new subscription ID.
- the *startSubscription* operation, which enables the processing of a given subscription of GDD data and GDD events to/at a Data service and defines rules for starting the subscription.
- the *stopSubscription* operation, which disables the processing of a given subscription. A subscription can be re-enabled by invoking *startSubscription*.
- the *dropSubscription* operation, which releases resources associated with a given subscription.

The administrative tasks for propagating events and data consist of:

- the *createPropagation* operation defines propagation rules for propagating GDD data and events from a Data service to a consumer.
- the *alterPropagation* operation, which alters propagation rules for a given propagation of GDD data and events from a Data service to a consumer. This operation returns a new propagation ID.
- the *startPropagation* operation, which enables delivery of GDD data and GDD event to a consumer and defines rules for starting the propagation .
- the *stopPropagation* operation, which disables propagation of GDD data and GDD events to a consumer. A propagation can be re-enabled by invoking the *startPropagation* operation.

- the *dropPropagation* operation, which releases resources associated with a given propagation.

The operational tasks for consumption of events and data consist of:

- the *createConsumption* operation, which defines rules for Data consumption. This is a required operation for the consumer to receive data.
- the *alterConsumption* operation, which alters consumption rules for a given consumption. This operation returns a new consumption ID.
- the *startConsumption* operation, which enables consumption of GDD data and GDD events and defines rules for starting the consumption.
- the *stopConsumption* operation, which disables consumption of GDD data and GDD events a target. A consumption can be re-enabled by invoking the *startConsumption* operation.
- the *dropConsumption* operation, which releases resources associated with a given consumption.

The operational tasks also include the following operations:

- the *publishData* operation, which publishes GDD data to/at a Data service for a given publisher.
- the *deliverData* operation, which is an operation on the target (Data service or client) that is invoked by the source Data service to deliver data to the target consumer.
- the *deliverEvent* operation, which is an operation on the target (Data service or client) that is invoked by the source Data service to deliver an event to the target consumer.
- the *getData* operation, which retrieves data (if any) from a source Data service and is invoked by a consumer.

Some of the tasks above may be defined implicitly. For example, a publication may be derived from the definition of a subscription. At this time, all tasks are defined explicitly; however, in future versions of this document, the implicit tasks will be clearly identified.

### 3.3 Bulk Data Transfer and Data Distribution

In some cases, bulk data transfer is a pre-condition for starting a publication, e.g in Data Replication. This could be achieved through a data access operation (as opposed to data distribution operation).

## 4. The GDD PortType, Service Data Elements and Operations

The components of the Grid Data Distribution model include its GDD portType, SDEs and operations. They are described in this section.

### 4.1 GDD PortType

The GDD portType extends the Data Management portType of a Data service.

## 4.2 GDD Service Data Declarations

The GDD portType includes a set of Service Data Elements.

- GDDServiceDataName

A set of service data elements QNames to which a requestor MAY publish to, subscribe from, manage propagation and consumption for sending or receiving data and/or events.

```
<sd:serviceData name="GDDServiceDataName"
type="xsd:QName"
minOccurs="0" maxOccurs="unbounded"
mutability="mutable"
modifiable="false"
nillable="false"/>
```

A set of operation extensibility declarations for the publication, subscription, propagation and consumption operations. Any conforming inputElement declared by values of this SDE MAY be used by the client as a publicationRule, subscriptionRule, propagation Rule or consumption Rule parameter to the publication, subscription, propagation and consumption operations respectively and implies the publication, subscription, propagation and consumption semantics and the GDD data and GDD event that results from the same.

- publication Extensibility:

```
<sd:serviceData name="publicationExtensibility"
type="ogsi:OperationExtensibilityType"
minOccurs="1" maxOccurs="unbounded"
mutability="static"
modifiable="false"
nillable="false"/>
```

- startPublication Extensibility:

```
<sd:serviceData name="startPublicationExtensibility"
type="ogsi:OperationExtensibilityType"
minOccurs="1" maxOccurs="unbounded"
mutability="static"
modifiable="false"
nillable="false"/>
```

- subscription Extensibility:

```
<sd:serviceData name="subscriptionExtensibility"
type="ogsi:OperationExtensibilityType"
minOccurs="1" maxOccurs="unbounded"
mutability="static"
modifiable="false"
nillable="false"/>
```

- startSubscription Extensibility:

```
<sd:serviceData name="startSubscriptionExtensibility"
```



```

type="ogsi:OperationExtensibilityType"
minOccurs="1" maxOccurs="unbounded"
mutability="static"
modifiable="false"
nillable="false"/>

```

- propagation Extensibility:

```

<sd:serviceData name="propagationExtensibility"
type="ogsi:OperationExtensibilityType"
minOccurs="1" maxOccurs="unbounded"
mutability="static"
modifiable="false"
nillable="false"/>

```

- startPropagation Extensibility:

```

<sd:serviceData name="startPropagationExtensibility"
type="ogsi:OperationExtensibilityType"
minOccurs="1" maxOccurs="unbounded"
mutability="static"
modifiable="false"
nillable="false"/>

```

- consumption Extensibility:

```

<sd:serviceData name="consumptionExtensibility"
type="ogsi:OperationExtensibilityType"
minOccurs="1" maxOccurs="unbounded"
mutability="static"
modifiable="false"
nillable="false"/>

```

- startConsumption Extensibility:

```

<sd:serviceData name="startConsumptionExtensibility"
type="ogsi:OperationExtensibilityType"
minOccurs="1" maxOccurs="unbounded"
mutability="static"
modifiable="false"
nillable="false"/>

```

The GDD portType includes the following initial service data value elements:

```

<sd:staticServiceDataValues>
<ogsi:publicationExtensibility
inputElement="ogsi:publicationByServiceDataNames" />
<ogsi:startPublicationExtensibility
inputElement="ogsi:startPublicationByServiceDataNames" />
<ogsi:subscriptionExtensibility
inputElement="ogsi:subscriptionByServiceDataNames" />
<ogsi:startSubscriptionExtensibility
inputElement="ogsi:startSubscriptionByServiceDataNames" />

```

```

<ogsi:propagationExtensibility
inputElement="ogsi:propagationByServiceDataNames" />
<ogsi:startPropagationExtensibility
inputElement="ogsi:startPropagationByServiceDataNames" />
<ogsi:consumptionExtensibility
inputElement="ogsi:consumptionByServiceDataNames" />
<ogsi:startConsumptionExtensibility
inputElement="ogsi:startConsumptionByServiceDataNames" />
</sd:staticServiceDataValues>

```

### 4.3 GDD Operations

The Data Distribution tasks are divided into administrative tasks (invoked by an Administrator) and operational tasks (invoked by publishers, subscribers or consumers).

The assumption is that all operations defined below are invoked against a Data service which implements the GDD port type (and its relevant SDEs as discussed above) and which has already been instantiated.

#### 4.3.1 Operations for Administrative Tasks

The operations below are invoked by Data Distribution administrators.

##### 4.3.1.1 createPublication

The GDD::createPublication operation creates publication rules for a publisher of GDD data and GDD events at a Data service.

Input:

*publicationRule*: This controls the data and/or event that is allowed to be published to/at the source. This extensible parameter MUST conform to an inputElement declaration denoted by one of the publishExtensibility SDE values.

*publisherName*: This refers to an admissible publisher by name.

Output:

*GDDpublicationId*: An identifier for the publication that is created through this operation to manage this publication. If createPublication is called again with the same arguments it just returns the GDDpublicationId.

Fault(s):

*ExtensibilityNotSupportedFault*: Indicates that the service cannot evaluate the publicationRule because its type is not supported by this service.

*ExtensibilityTypeFault*: Indicates that the value passed as the publicationRule violates that value's type.

*TargetInvalidFault*: Indicates that one or more of the SDEs that the publicationRule requires does not exist in this service.

*Fault*: Any other fault.

## Usage Note:

- a createPublication does not necessarily starts the publication (see startPublication below).
- the concept of topics may be used to tie publications to subscriptions.

## 4.3.1.2 alterPublication

The GDD::alterPublication operation alters publication rules for a publisher of GDD data at a Data service. This operation returns a new publication ID.

## Input:

*GDDpublicationId*: An identifier for the publication that was created through a createPublication operation.

*publicationRule*: This controls the data that is allowed to be published to/at the source. This extensible parameter MUST conform to an inputElement declaration denoted by one of the publishExtensibility SDE values.

## Output:

*newPublicationId*: An identifier for the new publication that is created to manage the altered publication.

## Fault(s):

*ExtensibilityNotSupportedFault*: Indicates that the service cannot evaluate the publicationRule because its type is not supported by this service.

*ExtensibilityTypeFault*: Indicates that the value passed as the publicationRule violates that value's type.

*TargetInvalidFault*: Indicates that one or more of the SDEs that the publicationRule requires does not exist in this service.

*Fault*: Any other fault.

## 4.3.1.3 startPublication

The GDD::startPublication enables publication of data to/at a Data service and defines rules for starting the publication.

## Input:

*GDDpublicationId*: An identifier for the publication that was created through the createPublication operation to manage this publication.

*startPublicationRule*: This specifies the criteria for data to be published at start/restart. This extensible parameter MUST conform to an inputElement declaration denoted by one of the startPublicationExtensibility SDE values.

## Output:

*Nil*.

## Fault(s):

*TargetInvalidFault:* Indicates that the GDDpublicationId provided as input argument is invalid.

*Fault:* Any other fault.

Usage Note:

- a startPublication rule may include scheduling, restart points, etc.

#### 4.3.1.4 stopPublication

The GDD::stopPublication disables publication of data and/or event to/at a data source. A publication can be re-enabled by invoking the startPublication operation.

Input:

*GDDpublicationId:* An identifier for the publication that was created through createPublication operation to manage this publication.

Output:

*Nil.*

Fault(s):

*TargetInvalidFault:* Indicates that the GDDpublicationId provided as input argument is invalid.

*Fault:* Any other fault.

Usage Note:

- the stopPublication and startPublication operations form a pausing and restart capability respectively.

#### 4.3.1.5 dropPublication

The GDD::dropPublication releases resources associated with a given publication.

Input:

*GDDpublicationId:* An identifier for the publication that was created through createPublication operation to manage this publication.

Output:

*Nil.*

Fault(s):

*TargetInvalidFault:* Indicates that the GDDpublicationId provided as input argument is invalid.

*Fault:* Any other fault.

#### 4.3.1.6 createSubscription

The GDD::createSubscription operation creates subscription rules for a subscriber of GDD data and GDD events at a Data service.

**Input:**

*subscriptionRule*: The subscription to be performed. This extensible parameter **MUST** conform to an inputElement declaration denoted by one of the subscribeExtensibility SDE values.

*subscriberName*: This refers to a subscriber by name.

**Output:**

*GDDsubscriptionId*: An identifier to the subscription that is created through this operation to manage this publication. If createSubscription is called again with the same arguments it just returns the GDDsubscriptionId.

**Fault(s):**

*ExtensibilityNotSupportedFault*: Indicates that the service cannot evaluate the subscriptionRule because its type is not supported by this service.

*ExtensibilityTypeFault*: Indicates that the value passed as the subscriptionRule violates that value's type.

*TargetInvalidFault*: Indicates that one or more of the SDEs that the subscriptionRule requires does not exist in this service.

*Fault*: Any other fault.

**Usage Note:**

- a createSubscription does not necessarily starts the subscription (see startSubscription below).
- the concept of topics may be used to tie publications to subscriptions.

#### 4.3.1.7 alterSubscription

The GDD::alterSubscription operation alters subscription rules for a subscriber of GDD data or GDD events at a Data service. This operation returns a new subscription ID.

**Input:**

*GDDsubscriptionId*: An identifier to the subscription that was created through createSubscription operation to manage this subscription.

*subscriptionRule*: The subscription to be performed. This extensible parameter **MUST** conform to an inputElement declaration denoted by one of the subscribeExtensibility SDE values.

**Output:**

*newGDDSubscriptionId*: An identifier to the new subscription that is created to manage the altered publication.

**Fault(s):**

*ExtensibilityNotSupportedFault*: Indicates that the service cannot evaluate the subscriptionRule because its type is not supported by this service.

*ExtensibilityTypeFault*: Indicates that the value passed as the subscriptionRule violates that value's type.

*TargetInvalidFault*: Indicates that one or more of the SDEs that the subscriptionRule requires does not exist in this service.

*Fault*: Any other fault.

#### 4.3.1.8 startSubscription

The GDD::startSubscription enables the processing of a given subscription of GDD data and GDD events to/at a Data service and defines rules for starting the subscription.

Input:

*GDDsubscriptionId*: An identifier for the subscription that was created through the createSubscription operation to manage this subscription.

*startsubscriptionRule*: This specifies the criteria for data to be subscribed at start/restart. This extensible parameter MUST conform to an inputElement declaration denoted by one of the startSubscriptionExtensibility SDE values.

Output:

*Nil*.

Fault(s):

*TargetInvalidFault*: Indicates that the GDDsubscriptionId provided as input argument is invalid.

*Fault*: Any other fault.

#### 4.3.1.9 stopSubscription

The GDD::stopSubscription disables the processing of a given subscription. A subscription can be re-enabled by invoking the startSubscription operation.

Input:

*GDDsubscriptionId*: An identifier to the subscription that was created through createSubscription operation to manage this subscription.

Output:

*Nil*.

Fault(s):

*TargetInvalidFault*: Indicates that the GDDsubscriptionId provided as input argument is invalid.

*Fault*: Any other fault.

## Usage Note:

- the stopSubscription and startSubscription operations form a pausing and restart capability respectively.

## 4.3.1.10 dropSubscription

The GDD::dropSubscription releases resources associated with a given subscription.

## Input:

*GDDsubscriptionId*: An identifier to the subscription that was created through createSubscription operation to manage this subscription.

## Output:

*Nil*.

## Fault(s):

*TargetInvalidFault*: Indicates that the GDDsubscriptionId provided as input argument is invalid.

*Fault*: Any other fault.

## 4.3.1.11 createPropagation

The GDD::createPropagation defines propagation rules for propagating GDD data and events from a Data service to a consumer.

## Input:

*targetLocation(s)*: Data service locator(s), the locator is created during the instantiation of the Data service or target(s) information.

*propagationRule*: The propagation rules to be performed. This extensible parameter MUST conform to an inputElement declaration denoted by one of the propagationExtensibility SDE values.

## Output:

*GDDpropagationId*: An identifier of the new propagation that is created. This identifier may have to exist at both the source and target Data services.

## Fault(s):

*ExtensibilityNotSupportedFault*: Indicates that the service cannot evaluate the propagationRule because its type is not supported by this service.

*ExtensibilityTypeFault*: Indicates that the value passed as the propagationRule violates that value's type.

*TargetInvalidFault*: Indicates that one or more of the SDEs that the propagationRule requires does not exist in this service.

*Fault*: Any other fault.

#### Usage Note:

- a createPropagation does not necessarily starts the propagation (see startPropagation below).
- the publication ID or subscription ID may be provided as part of the propagation rule.

#### 4.3.1.12 alterPropagation

The GDD::alterPropagation alters propagation rules for a given propagation of GDD data and events from a Data service to a consumer. This operation returns a new propagation ID.

##### Input:

*GDDpropagationId*: An identifier for the propagation that was created through createPropagation operation to manage this propagation.

*propagationRule*: propagation rules to be performed. This extensible parameter MUST conform to an inputElement declaration denoted by one of the propagationExtensibility SDE values.

##### Output:

*newPropagationId*: An identifier for the new propagation that is created to manage the altered propagation.

##### Fault(s):

*ExtensibilityNotSupportedFault*: Indicates that the service cannot evaluate the propagationRule because its type is not supported by this service.

*ExtensibilityTypeFault*: Indicates that the value passed as the propagationRule violates that value's type.

*TargetInvalidFault*: Indicates that one or more of the SDEs that the propagationRule requires does not exist in this service.

*Fault*: Any other fault.

#### 4.3.1.13 startPropagation

The GDD::startPropagation enables delivery of GDD data and GDD event to a consumer and defines rules for starting the propagation.

##### Input:

*GDDpropagationId*: An identifier for the propagation that was created through the createPropagation operation to manage this propagation.

*startPropagationRule*: This specifies the criteria for data to be propagated at start/restart. This extensible parameter MUST conform to an inputElement declaration denoted by one of the startPropagationExtensibility SDE values.



Output:

*Nil.*

Fault(s):

*TargetInvalidFault:* Indicates that the GDDpropagationId provided as input argument is invalid.

*Fault:* Any other fault.

#### 4.3.1.14 stopPropagation

The GDD::stopPropagation operation disables propagation of GDD data and GDD events to a consumer. A propagation can be re-enabled by invoking the startPropagation operation.

Input:

*GDDpropagationId:* An identifier to the propagation that was created through createPropagation operation to manage this propagation.

Output:

*Nil.*

Fault(s):

*TargetInvalidFault:* Indicates that the GDDpropagationId provided as input argument is invalid.

*Fault:* Any other fault.

#### 4.3.1.15 dropPropagation

The GDD::dropPropagation releases resources associated with a given propagation.

Input:

*GDDpropagationId:* Identifier to the propagation to be dropped.

Output:

*Nil.*

Fault(s):

*TargetInvalidFault:* Indicates that the GDDpropagationId provided as input argument is invalid.

*Fault:* Any other fault.

### 4.3.2 Operations for Operational Tasks

The operations below are invoked by publishers, subscribers or consumers. Some of the operations may also be invoked by Administrators (consumption operations in particular).

#### 4.3.2.1 createConsumption

The GDD::createConsumption defines rules for Data consumption. This is a required operation for the consumer to receive data.

Input:

*consumptionRule*: The consumption rules to be performed. This extensible parameter **MUST** conform to an inputElement declaration denoted by one of the consumeExtensibility SDE values.

*consumerName*: This refers to a consumer by name.

Output:

*GDDConsumptionId*: An identifier of the consumption that is created.

Fault(s):

*ExtensibilityNotSupportedFault*: Indicates that the service cannot evaluate the consumptionRule because its type is not supported by this service.

*ExtensibilityTypeFault*: Indicates that the value passed as the consumptionRule violates that value's type.

*TargetInvalidFault*: Indicates that one or more of the SDEs that the consumptionRule requires does not exist in this service.

*Fault*: Any other fault.

Usage Note:

- a createConsumption does not necessarily start the consumption (see startConsumption below).
- The publication ID, subscription ID or propagation ID may be provided as part of the consumption rule.
- For the pull model, the consumption rules have to be specified at the source.
- The consumption rules may be evaluated at a source Data service.

#### 4.3.2.2 alterConsumption

The GDD::alterConsumption alters consumption rules for a given consumption. This operation returns a new consumption ID.

Input:

*GDDconsumptionId*: An identifier for the consumption that was created through createConsumption operation to manage this consumption.

*consumptionRule*: This controls the data that is allowed to be consumed at the target. This extensible parameter **MUST** conform to an inputElement declaration denoted by one of the consumeExtensibility SDE values.

Output:

*newConsumptionId*: An identifier for the new consumption that is created to manage the altered consumption.

## Fault(s):

*ExtensibilityNotSupportedFault*: Indicates that the service cannot evaluate the consumptionRule because its type is not supported by this service.

*ExtensibilityTypeFault*: Indicates that the value passed as the consumptionRule violates that value's type.

*TargetInvalidFault*: Indicates that one or more of the SDEs that the consumptionRule requires does not exist in this service.

*Fault*: Any other fault.

## 4.3.2.3 startConsumption

The GDD::startConsumption enables consumption of GDD data and GDD events and defines rules for starting the consumption.

## Input:

*GDDconsumptionId*: An identifier for the consumption that was created through the createConsumption operation to manage this consumption.

*startConsumptionRule*: This specifies the criteria for data to be consumed at start/restart. This extensible parameter MUST conform to an inputElement declaration denoted by one of the startConsumptionExtensibility SDE values.

## Output:

*Nil*.

## Fault(s):

*TargetInvalidFault*: Indicates that the GDDconsumptionId provided as input argument is invalid.

*Fault*: Any other fault.

## 4.3.2.4 stopConsumption

The GDD::stopConsumption disables consumption of GDD data and GDD events at a target. A consumption can be re-enabled by invoking the startConsumption operation.

## Input:

*GDDConsumptionId*: An identifier for the consumption that was created through the createConsumption operation to manage this consumption.

## Output:

*Nil*.

## Fault(s):

*TargetInvalidFault*: Indicates that the GDDconsumptionId provided as input argument is invalid.

*Fault*: Any other fault.

#### 4.3.2.5 dropConsumption

The GDD::dropConsumption releases resources associated with a given consumption.

Input:

*GDDconsumptionId*: Identifier of the consumption to be dropped.

Output:

*Nil*.

Fault(s):

*TargetInvalidFault*: Indicates that the GDDconsumptionId provided as input argument is invalid.

*Fault*: Any other fault.

#### 4.3.2.6 publishData

The GDD::publishData operation publishes GDD data to/at a Data service for a given publisher.

Input:

*GDDpublicationId*: An identifier to the publication that was created through createPublication operation to manage this publication.

*GDD data* – if null, this implies implicit publishing through pre-defined publication rules.

Output:

*Nil*.

Fault(s):

*TargetInvalidFault*: Indicates that the GDDpublicationId provided as input argument is invalid.

*Fault*: Any other fault.

Usage Note:

This operation should only be used for data publication.

#### 4.3.2.7 deliverData

The GDD::deliverData operation is an operation on the target (Data service or client) that is invoked by the source Data service to deliver data to the target consumer.

Input:

*GDDpropagationId*: An identifier to the propagation that was created through createPropagation operation to manage this propagation.

*GDDconsumerID*: An identifier to the consumer that is receiving the data.

*GDD Data*

## Output:

Nil.

## Fault(s):

*TargetInvalidFault:* Indicates that the GDDpropagationId provided as input argument is invalid.

*Fault:* Any other fault.

## Usage Note:

- only one of the propagation ID or consumer ID can be specified
- the consumer ID needs to be provided if the target is not a service
- if the propagation ID is provided, then the consumption rules are defined and applied at the target
- if the consumer ID is provided, then createConsumption has to be specified at the source Data service, where the consumption rules are also evaluated.

## 4.3.2.8 deliverEvent

The GDD::deliverEvent operation is an operation on the target (Data service or client) that is invoked by the source Data service to deliver an event to the target consumer.

## Input:

*GDDpropagationId:* An identifier to the propagation that was created through createPropagation operation to manage this propagation.

*GDDconsumerID:* An identifier to the consumer that is receiving the data.

*GDD Event*

## Output:

Nil.

## Fault(s):

*TargetInvalidFault:* Indicates that the GDDpropagationId provided as input argument is invalid.

*Fault:* Any other fault.

## Usage Note:

- only one of the propagation ID or consumer ID can be specified
- the consumer ID needs to be provided if the target is not a service
- if the propagation ID is provided, then the consumption rules are defined and applied at the target
- if the consumer ID is provided, then createConsumption has to be specified at the source Data service, where the consumption rules are also evaluated.

## 4.3.2.9 getData

The GDD::getData operation retrieves data (if any) from a source Data service and is invoked by a consumer.

Input:

*GDDconsumptionId*: An identifier of the consumption that was created through createConsumption.

Output:

*GDD Data*

Fault(s):

*TargetInvalidFault*: Indicates that the GDDpropagationId provided as input argument is invalid.

*Fault*: Any other fault.

Usage Note:

- this operation represents the pull model.
- createConsumption has to be specified at the source Data service, where the consumption rules are also evaluated.

## 5. Scenarios

This section presents a list of data distribution scenarios that rely on publications, subscriptions, propagations and consumptions to demonstrate how the proposed GDD interface is flexible enough to meet all of the data distribution requirements mentioned in the above sections.

The scenarios include:

1. Asynchronous SQL Result Set Delivery to 3<sup>rd</sup> party consumers
2. Publication on Request: Asynchronous SQL Result Set Delivery
3. Workflow
4. Continuous Replication

Note:

Since this document does not provide specifications for publication, subscription, propagation and consumption rules, the rules provided below are for illustration purposes only. They are enclosed in brackets.

### 5.1 Asynchronous SQL Result Set Delivery to 3rd party consumers

This scenario includes both push and pull models. It consists of an analyst who works at a company's headquarter and generates complex queries against Customer headquarter data at database DB1 in New York to find potential trends in the customers' purchasing habits.

The analyst runs the queries at 3PM. Consumers for these results consist of 2 teams:

- a Los Angeles team, who receives the data at 9PM every day and is divided into 2 groups :

Dieter.gawlick@oracle.com, gogate@almaden.ibm.com, madsen@us.ibm.com, shailendra.mishra@oracle.com, narang@almaden.ibm.com, maha@almaden.ibm.com

- remote database users, who maintain that data in Los Angeles and wait for the data to be replicated (service-based consumers)
- the sales team in Los Angeles which expects summary data after 9PM every day via email (non-service based consumers)
- a San Francisco team, who doesn't want the data pushed to San Francisco and thus gets the data directly from DB1 after 3PM every day. Again, this team consists of 2 groups: service-based consumers and non-service based consumers.

The tasks needed to setup and execute this scenario for the Los Angeles team are provided below.

- a. The analyst in New York ("NYAnalyst") locates the Data service for DB1 by looking up a global registry that lists such Data services. The analyst receives in return the DB1 GSH handle (DB1GSH).
- b. The analyst in New York defines a subscription at DB1GSH to express interest in the appropriate Customer data:

```
DB1GSH::createSubscription([implicitname=CustQueryPub, aggregation
                           and selection predicates, schedule = 3PM], NYAnalyst)
returns SubsID.
```

The subscription generates the publication implicitly.

- c. For Service-based consumer in Los Angeles, the analyst in New York defines the propagation rules at DB1GSH to send the result of the query to the Data service at DB3 in Los Angeles for auditing purposes:

```
DB1GSH::createPropagation(DB3GSH, [subscriber = SubsID, target = service,
                                   schedule = 9PM])
returns propagationId1.
```

```
DB3GSH::createConsumption([propagationId1], LosAngelesConsumer1)
returns consumptionId1.
```

- d. For non-Service based consumer in Los Angeles, the analyst in New York also defines the propagation rules at DB1GSH to send the result of the query to the appropriate sales team member in Los Angeles, specifying the 3rd party consumer information (target address), protocols to be used and time when the delivery has to happen:

```
DB1GSH::createPropagation(LosAngelesURI, [scheduleat = 9PM])
returns propagationId2.
```

```
DB1GSH::createConsumption([propagationId2, <format for delivery>],
                           LosAngelesConsumer2)
returns consumptionId2.
```

- e. The tasks needed to actually deliver and push the data to Los Angeles are:

For Service based consumer: the source service issues:

```
DB3GSH::deliverData(propagationId1, null, <Data>)
```

For non-Service based consumer, the source service uses the protocol mentioned for

propagationId2 to determine how to send the data to the consumer LosAngelesConsumer2 at the LosAngelesURI.

The tasks needed to setup and execute this scenario for the San Francisco team are provided below.

The scenario is essentially the same as the one for Los Angeles, except that consumers are pulling the data themselves so no propagation rules are required. Steps a and b are exactly the same as the steps a and b in the Los Angeles scenario described above. Steps c to e are modified as follows,

- c. For Service-based consumer in San Francisco, the analyst in New York defines the consumption rules at DB1GSH to retrieve results:

```
DB1GSH::createConsumption([], San FranciscoConsumer1)
returns consumptionId1.
```

- d. For non-Service based consumer in San Francisco, the step is the same, simply define the consumption rules at DB1GSH to retrieve results:

```
DB1GSH::createConsumption([], San FranciscoConsumer2)
Returns consumptionId2.
```

- e. The tasks needed for the San Francisco consumers to actually get the data from the DB1 Data service are:

The San Francisco service-based consumer1 issues:

```
DB1GSH::getData(consumptionId1)
```

The San Francisco non-service-based consumer2 issues:

```
DB1GSH::getData(consumptionId2)
```

## 5.2 Publication On Request: Asynchronous SQL Result Set Delivery

This scenario consists a delivering asynchronous SQL Result sets to 3rd party consumers, with the added requirement that the publication be on request. It consists of an analyst who publishes a complex dynamic query (aggregation, etc.) against scientific data with variables (parameter markers). Interested subscribers have to provide the values for the variables that are of interest to them. For example, assuming analysis of astronomical data is offered, the subscribers can then specify the parameters for regions and spectral lines and request the data to be delivered to the subscriber and/or other interested parties.

The tasks needed to setup and execute this scenario are provided below:

- a. The analyst locates the Data service for astronomical data by looking up a global registry that lists such Data service. The analyst receives in return the GSH handle (ASTGSH).
- b. The analyst defines a publication at ASTGSH:



ASTGSH::createPublication([Query(region=?P1, spectrum=?P2)], Analyst1)  
Returns PubID1.

- c. The subscriber defines a subscription for astronomical data:

ASTGSH::createSubscription([name=PubID1, P1='East', P2='ALL',  
scheduleat='daily'], Subscriber1)  
returns SubsID1.

- d. The analyst publishes data:

ASTGSH::publishData(PubID1, null)

This will result in the evaluation of the publication/subscription rules, the execution of the query and the creation of the result set (a data set DS1).

- e. For service-based consumer, the analyst defines the propagation rules at ASTGSH to send the results of the subscription (data set DS1) to the Data service at DB3 for further analysis:

ASTGSH::createPropagation(DB3GSH, [subscriber = SubsID1, target = service])  
returns propagationId1.

ASTGSH::createConsumption([propagationId1], consumer1)  
returns consumptionId1.

- f. For non-Service based consumer, the analyst also defines the propagation rules at ASTGSH to send the result of the subscription (data set DS1) to another consumer consumer2, for example specifying a schedule:

ASTGSH::createPropagation(TargetURI, [schedule = 9PM])  
returns propagationId2.

ASTGSH::createConsumption([propagationId2, <format for delivery>],  
Consumer2)  
returns consumptionId2.

- g. The tasks needed to actually deliver the data in Los Angeles are:

For service-based consumer, the source service issues:

DB3GSH::deliverData(null, consumerId1, <Data>)

For non-Service based consumer, the source service uses the protocol mentioned for propagationId2 to determine how to send the data to consumer2 at TargetURI.

Similarly, another consumer could pull the data directly from ASTGSH, by issuing a  
getData(consumptionID).

### 5.3 Workflow

This scenario consists of an Order execution module of a Books online system. When an end-user places an order for a book through the Web, the user submits the order with the following

arguments: Book Name, Book Author, Payment method (if by credit card, then Credit card information), Billing information, Shipping information, Shipping mode and Coupon if any.

The workflow defines the interaction between the various steps that are necessary before flagging the order as complete. The steps are briefly described below.

- on submission of this order, the Order execution system publishes the new order data to a Data service (D1). The order is classified into two categories:
  - credit card orders, which are further divided by credit card type (VISA, AMEX etc.).
  - non-credit card orders
- at Data service D1, an analyst (DBA1) subscribes to credit card orders. Non-credit card orders are processed by some other analyst.
- at Data service D1, the DBA1 analyst creates a consumption for processing orders for credit card type = MC, with a consumption rule specifying [credit card type = MC].
- at Data service D1, the DBA1 analyst requests Data from the data source (D1) to validate the MC data; flags it as per the rules below and re-publishes the data back onto D1 with the appropriate flag:
  - for orders that are valid, set flag = valid and re-publish the data to D1
  - for orders that are incomplete, set flag = incomplete and re-publish the data to D1
  - for orders that include identity fraud etc., set flag = invalid and republish the data to D1
- at Data service D1, the DBA1 analyst creates a propagation to propagate valid orders from Data service D1 to a new Data service, D2 where further processing of the valid data will occur.

The incomplete data is propagated to the customer support Data service (D3) for further Processing (that step will not be described in details below as it is very similar to the valid order case).

The invalid data is removed from D1 and sent to authorities for appropriate action (similarly, that step will not be described in details below because it is also very similar to the valid order case).

- at Data service D2, once the data has been published, the Analyst requests the data to determine whether the book is available (may require some more queries), flags the data as per the rules below and re-publishes the data back onto D2 with the appropriate flag:
  - if the book is available, set flag = available and re-publish data to D2
  - if the book is unavailable, set flag = unavailable and re-publish data to D2
- at Data service D2, the DBA1 analyst creates a propagation to move orders for books that are unavailable from Data service D2 to data Service D3 and to move orders for books that are available to another Data service D4.
- at Data service D4, where the valid and available book orders are propagated, the analyst retrieves the data to fulfill the order (generate invoice, delivery documentation, etc.).
- at Data service D3, a notification is sent to the end-user customer to alert him/her of the status of the order and requesting further input.

The tasks needed to setup and execute this scenario at Data service D1 are provided below. The remaining parts of the scenario, specifically tasks needed to setup and execute this scenario at D2, D3 or D4 are basically a repeat of the tasks for D1 so will not be repeated below.

The tasks needed to setup and execute this scenario at Data service D1 are provided below:

- a. the DBA1 analyst creates a publication rule to verify the input values of the data being published (the new order) at D1 after locating the Data service for D1 (D1GSH):

```
D1GSH::createPublication([validate input], DBA1)
returns PubID1.
```

- b. the DBA1 analyst creates a subscription for a subscriber named "CreditCardOrders" to express interest in books for which the user intends to pay by credit card:

```
D1GSH::createSubscription([name=PubID1,creditcard=Y], DBA1)
returns subsID1.
```

- c. the DBA1 analyst creates a consumption for the CreditCardOrders subscription for MC cards:

```
D1GSH::createConsumption([subscriber=subsID1, credit_card_type = "MC"],
DBA1)
returns consumptionId1.
```

- d. DBA1 is both the subscriber and the consumer. It then calls the start consumption to enable that consumption:

```
D1GSH::startConsumption(consumptionId1, [])
```

- e. once this is done, the user processing the credit card validation retrieves the data pertaining to orders which are being paid by credit card and the card type = MC:

```
D1GSH::getData(consumptionId1)
```

and tags the data with the appropriate status (valid, incomplete or invalid)

- f. for all orders, the user processing the credit card paid orders re-publishes the data to D1 with status by creating a new publication, starting it and publishing to it:

```
D1GSH::createPublication([], DBA1)
returns PubID2.
```

```
D1GSH::startPublication(PubID2, [])
```

```
D1GSH::publishData(PubID2, <data with valid, incomplete or invalid status>)
```

- g. the DBA1 analyst sets up propagation from D1 to D2 for all valid orders, first creating a subscription for the valid orders and then starting the propagation:

```
D1GSH::createSubscription([name=PubID2,[status=valid], DBA1)
returns SubsID2.
```

```
D1GSH::createPropagation(D2GSH, [subscription=SubsID2])
returns PropagationID2
```

```
D1GSH::startPropagation(PropagationID2, [])
```

Note that deliverData is implicitly done through the start propagation and that the consumption at D2 is also implicit.

#### 5.4 Replication

This scenario consists of a replication scenario for which only changes that occur at a source site are to be replicated at target sites.

A company's global "Customer" table data resides at the headquarter (DB1 in New York) and is distributed across branch offices (DB3 in Los Angeles, DB4 in Dallas and DB5 in San Francisco for example). On a daily basis at the headquarter office, customer data is analyzed for the whole company and whenever appropriate, the data is updated to reflect the new customer status/rating/outstanding offers information. Every evening, the Los Angeles and Dallas branch offices replicate the changes that occurred for the customers who live in their area, only if such changes exist.

Note:

It is assumed that the 2 branch office sites are already synchronized with the headquarter site so that the branch offices are ready to receive the change data (no need to refresh the branch office tables).

The tasks needed to setup and execute this scenario are provided below:

- a. The administrator in New York ("NYAdmin") locates the Data service for DB1 by looking up a global registry that lists such Data services. The administrator receives in return the DB1 GSH handle (DB1GSH).
- b. The administrator in New York defines publication rules that will capture changes from the database log of DB1 and specifies to only publish GDD data that relates to changes associated with the Customer table:

```
DB1GSH::createPublication([name=CustomerPublication, table=customer,
                        startPublication = N, log based=Y], NYAdmin)
returns NYPubId.
```

The Capture process at the source may be automatically started as a result of a start publication operation (startPublication()). In this case, a publishData is implied and thus not required as an explicit operation.

- c. The subscriber in Los Angeles ("LASubs") defines a subscription for GDD data at DB1GSH to carry over DB1 change data to Los Angeles and specifies a rule to express interest in the Los Angeles area data only:

```
DB1GSH::createSubscription([name=CustomerPublication, area=Los Angeles],
LASubs)
returns LASubsID.
```

- d. The subscriber in Dallas performs the same task as the Los Angeles Subscriber, expressing interest for the Dallas area data only:

```
DB1GSH::createSubscription([name=CustomerPublication, area=Dallas],
DallasSubs)
```

returns DallasSubsID.

- e. The administrator at DB1 defines the propagation mechanism for moving the data from the headquarter center to the branch offices, after locating the publication “CustomerPublication” and the Data services for DB3 and DB4 (DB3GSH and DB4GSH respectively):

```
DB1GSH::createPropagation(DB3GSH, [subs=LASubsID, DeliverOnce,
    TransactionConsistency])
DB1GSH::createPropagation(DB4GSH, [subs=DallsSubsID, DeliverOnce,
    TransactionConsistency])
```

returns 2 propagationIds (DB3propID, DB4propID respectively), which are also implicitly defined at their respective target Data services.

- f. The administrators at DB3 and DB4 define the consumption rules as follows:

```
DB3GSH::createConsumption([propagation=DB3propID], DB3Admin)
DB4GSH::createConsumption([propagation=DB4propID], DB4Admin)
```

- g. The Apply processes at the respective targets are automatically started as a result of a start consumption operation (startConsumption()).

Note that this example also illustrates that fact that some operations like publishData and deliverData may be implicitly performed.

### Remaining Work and **Open Issues**

The following items are remaining work to do:

- Rule specifications for Publication, Subscription, Propagation and Consumption
- Use of the DAIS Transformation specifications
- Description of GDD Data (especially for changed data case) and GDD Event format; this will enable the support for heterogeneous database data transfer
- Support for listing publications/subscriptions/propagation/consumption identifiers
- Support for monitoring functions
- Understand how having restart points (rules) specified at different stages (publication/subscription/propagation/consumption level) impact the distribution scenario
- Understand how the different rules tie together and what happens if some rules are omitted
- All operations are currently explicit; investigate how to support some implicit operations
- Best practices’ document (usage case scenarios, File Replication, Extract/Transform/Load support)
- Security considerations (Administrative versus Operational tasks)

The following items are open issues:

- GDD portType positioning vis-à-vis Data service Data Management portType
- GDD portType positioning vis-à-vis OGSi Notification portTypes
- Finalize terminology for start/stop operations (they behave as allow/disallow operations)

## 7. Security Considerations

This section will be described in detail at a later time.

### Author Information

Dieter Gawlick  
Oracle Corporation  
500 Oracle Parkway  
Redwood Shores  
CA 94065  
(650) 506 8706  
[dieter.gawlick@oracle.com](mailto:dieter.gawlick@oracle.com)

Vitthal Gogate  
IBM Almaden Research Center  
650 Harry Road  
San Jose, CA 95120-6099  
(408) 927 1799  
[gogate@almaden.ibm.com](mailto:gogate@almaden.ibm.com)

Cecile Madsen  
IBM Silicon Valley Laboratory  
555 Bailey Avenue  
San Jose, CA 95141  
(408) 463 2578  
[madsen@us.ibm.com](mailto:madsen@us.ibm.com)

Shailendra Mishra  
Oracle Corporation  
500 Oracle Parkway  
Redwood Shores  
CA 94065  
(650) 506 9123  
[shailendra.mishra@oracle.com](mailto:shailendra.mishra@oracle.com)

Inderpal Narang  
IBM Almaden Research Center  
650 Harry Road  
San Jose, CA 95120-6099  
(408) 927 1743  
[narang@almaden.ibm.com](mailto:narang@almaden.ibm.com)

Mahadevan Subramanian  
IBM Almaden Research Center  
650 Harry Road  
San Jose, CA 95120-6099  
(408) 927 1777  
[maha@almaden.ibm.com](mailto:maha@almaden.ibm.com)

### 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 © Global Grid Forum (date). 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]

A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, S. Tuecke, *The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets*, Journal of Network and Computer Applications, 23:187-200, 2001 (based on conference publication from Proceedings of NetStore Conference 1999).

#### [Data Services]

I. Foster, S. Tuecke, J. Unger, *OGSA Data Services*, DAIS-WG Informational Draft, 9th Global Grid Forum, 14th August 2003

#### [DAIS]

M. Antonioletti, M.P. Atkinson, N. P. Chue Hong, A. Krause, S. Malaika, G. McCance, S. Laws, J. Magowan, N.W. Paton, G. Riccardi, *Grid Data Service Specification*, 6<sup>th</sup> June 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