INTRODUCTION TO SERVICE DATA CONCEPTS

A grid service is a stateful Web service. Because of this architecture model design fact, the service data concept requires the OGSI to identify a common mechanism to expose the state data of the service instance to the service requestor of the query, the update action itself, and finally enable the change notification to occur. In this case, the OGSI utilized the "service data declaration" as a mechanism for publicly expressing the available state information of a service. This concept, however, is not limited to grid services.

The service data concept can be extended to any stateful Web service for declaring its publicly available state information through the service data concepts. Therefore, developers that were exposed to some of the more traditional distributed technologies, and their interface declaration (IDL) approaches, will be familiar with this somewhat similar concept. Some of the object-oriented distributed language interfaces use attributes declaration to indicate the exposed state/properties of the services they describe.

The following describes the service data concepts introduced by the OGSI specification:

- Service data declaration (SDD) is a mechanism to expose a publicly available state of a service.
- Service data elements (SDE) are accessible through the common grid service interfaces ("findServiceData" and "setServiceData").
- The internal state of a service should not be a part of the service data declaration.

Provided that we have now discussed the usability of service data, let us now explore the concepts, semantics, and usage model of service data in the context of a grid service.

Service Data Structure

Service data is clearly modeled in the OGSI-defined namespace attribute.This new OGSI schema type for service data ("sd:serviceData") contains seven predefined attributes, including name, type, minOccurs, maxOccurs, modifiable, mutability, and nilable. Most of these attributes are standard XSD types, with the exception of the "mutability" attribute. This is further defined by OGSI as an enumerated type, with values of "static," "constant," "extendable," and "mutable."
Note that this schema allows us to always add additional attributes of choice. There is another notable, yet often underutilized feature in this instance provided by this type of definition. It is the open content model related to content from any other namespace. This feature may be utilized in the future to expose some policies or meta-data about the service data, including security profiles.

This attribute set is extensible through the open attribute declaration of the schema's SDE. Therefore, the service can add more semantic information about a service data through attribute extensibility. An example of this extensibility is presented later with life cycle attributes for a service data element.
Remembering the default values of these attributes will help us to understand and define a good state management framework for grid services. Based on the SD definition, and as shown in the above table, the required attributes for an SDE are "name" and "type" attributes and the service developer is expected to provide them. The other attributes have default values assigned to them.

Table 6.1 lists these attributes of service data and their default values

SDE Attributes	Description and Default Values
Name	This is a required attribute with a uniquely identifiable name of the service data element in the target namespace.
Type	This is the other required attribute, which defines the XML schema type of the service data value, the SD value. The SD value is based upon this schema and can be defined as simple or complex in a manner related to XSD schema types, and/or one may define this in terms of other complex types.
maxOccurs	This indicates the maximum number of SDE values that can appear in the service instance's SDE value set, or the portType staticServiceData Values. Default value = 1
minOccurs	This indicates the minimum number of SDE values that can appear in the service instance's SDE value set or the portType staticServiceDataValues. If minOccurs = 0, then this SDE is optional. Default value = 1

nilable	This indicates whether an SD value can have a nil value. One can declare this SDE as:
	
	`<sd:serviceData name="lifecycleModel"`
	
	`type="crm:lifecycleModelType" nillable="true"/>`
	
	Another valid SDE value is:
	
	`<sd: lifecycleModel xsd:nil="true" />.`
	
	Default value = false

modifiable	This is a mechanism to specify a read-only and changeable service data element. If changeable, you can use "setServiceData" operation to change its SDE value based on the other attribute (mutability, min, and max) constraints. This modifiable attribute is applicable to the service requestor only. Internally a service can change its SDE values if other constraints are met. Default value = false (all SDEs are by default "read only")
mutability	This is an indication of whether and how the values of a service data element can change.

Table 6.2. Service data element mutability attributes

SDE Mutability Attribute Value	Description of SDE Value	How to Define and Initialize This SDE Value
Static	Analogous to a language class member variable. All portType declarations carry this service data value.	Inside GWSDL portType using <staticServiceDataValues>. We can see this example in the previous listing.
Constant	This SDE value is constant and must not change.	This SDE value is assigned on the creation of grid service (runtime behavior).
Extendable	Similar to the notion of appending values. Once added, these values remain with the SDE, while new values are appended.	Programmatically speaking, we can append new SDE values. The new values are appended while the old ones remain.
Mutable	The SDE values can be removed and others can be added.	Programmatically speaking, we can change these SDE values and add new ones.

Types of Service Data Elements and Service Data Values

Every service instance has a collection of service data elements it exposes through public interfaces. These service data elements can be classified into two types of attributes, based upon the creation semantics. These are:

1. Static. Declared as part of the service's interface definition (GWSDL portType definition).
2. Dynamic. Added to a service instance dynamically. This behavior is implementation specific. The client may know the semantics (type and meaning) of the service data, or can acquire that information from somewhere (service or third party) through meta-data exchange.

For example, in order to process the dynamic SDE values, you may need to get the schema type information for the SDE values from a remote location

Qualifying Service Data Element with Lifetime Attributes

In addition to the expressed features of the service data as previously discussed, there is also a "hidden" concept in the specification with respect to the lifetime properties associated with the service data elements. The concept is hidden because it is just a recommendation that the service/client implementation could possibly ignore. However, good designs, programs, and tools should be aware of this feature.

The service data element represents the real-time observations of the dynamic state of a service instance. This real-time observation forces the clients to understand the validity and availability of the state representation. That is, certain service data elements, especially within dynamic SDEs, may have a limited lifetime. If there is lifetime information associated with the SDE, it can help the client to make decisions on whether an SDE is available, has validity, and when it is to revalidate the SDE. The most helpful development implementation of this concept may be the client-side service data cache, and an associated revalidation mechanism.

Based on the preceding requirements, the specification provides three kinds of lifetime properties:

- The time from which the contents of this element are valid (ogsi:goodFrom)
- The time until which the contents of this element are valid (ogsi:goodUntil)
- The time until which this element itself is available (ogsi:availableUntil)

The first two properties are related to the lifetime of the contents, while the third, the availableUntil attribute, defines the availability of the element itself. For example, we may see a dynamic SDE with availability until a specific time, and thereafter, it ceases to exist. This is a good indication for the users of this specific service data element not to use that SDE after that specified time.

According to the specification, these values are optional attributes of the SDE element and the SDE values; however, it is always recommended to include the optional attributes in the XML schema design of the types for the service data elements.