Enabling Grids for E-science in Europe

www.eu-egee.org

WSDL Extensions

Grid Service Description Language

Section Overview

- Introduction
- WSDL Conventions
- GSDL Extensions
- Standard Interfaces

Review: Transient Service Instances

- Web Services address persistent services
  - Interface to persistent state of entire enterprise
- Grid services created/destroyed dynamically
  - Interfaces to the states of distributed activities
  - E.g. workflow, video conferencing, distributed data analysis
- Implications for how services are managed, named, discovered and used

Review: OGSI Grid Services

- Defines WSDL conventions/GSDL extensions
  - For describing and structuring services
  - Working with W3C WSDL working group to drive GSDL extensions into WSDL
- Defines fundamental interfaces (using WSDL) and behaviors that define a Grid Service
  - A unifying framework for interoperability & establishment of total system properties

WSDL Conventions

- portType (WSDL v1.1)
  - Defines an interface: a named set of related operations
- WSDL v1.1 <service> element is ambiguous about the relationship of its ports
  - Are all ports in a service related?
  - Mixes service definition with service binding

portType inheritance

- Allows interface composition to define services
- Separates interface from implementation
- WSDL v1.2 draft has this

Service Description

- Abstraction: independent of any particular instance
  - Describes how client interacts with a service,
  - In WSDL, this is the portType (and the messages and types that it implies)
- Primary purposes:
  - Discovery: find services of interest
  - Tooling: generate client proxies & server code
- Any number of service instances may bind to a particular service description

Discovery

- Examples: Find me a service that…
  - supports a particular set of operations.
  - can create a service that supports operations.
  - will respond as I expect to an op request.
  - I can use.
  - is currently suspended waiting for input.
  - has 10MB bandwidth to my machine.
  - has 5ms latency to any copy of my database.
  - various combinations of these…

Capturing Semantics

- Service description only specifies types
- Semantic meaning is critical for discovery
  - Not only does the service accept an operation request with a particular signature
  - But it should also respond as expected
- Approach: name everything
  - Name implies semantics
  - Semantics are either formally defined (e.g. semantic web), or informally (e.g. specs)

portType Definition

- Below is an example of a portType definition using GSDL extensions

```
<wsdl:portType name="ncname"> *
    <wsdl:documentation .... /> ?
    <wsdl:extends portType="qname"> * (in flux)
    <wsdl:operation name="ncname"> *
        …
    <gsdl:serviceData name="ncname" … /> *
    <gsdl:staticServiceDataValues>?
        <some element> *
    </gsdl:staticServiceDataValues>
</wsdl:portType>
```

GSDL Extensions

- serviceData (GSDL extension)
  - Description of data associated with interface
- staticServiceDataValues (GSDL extension)
  - WSDL defined values for serviceData
- serviceDataValues (GSDL extension)
  - Instance specific values for serviceData

Service Data

- A Grid service instance maintains a set of service data elements (SDE)
  - Declared via an extended XSD element declaration, placed in a WSDL portType
  - Values carried in a portType or in an instance
  - Includes basic introspection information, interface-specific data, and application state

Why Service Data?

- Discovery often requires instance-specific, perhaps dynamic information

- Service data offers a general solution
  - Every service must support some common service data
  - A service may support any additional service data desired
  - Not just meta-data, but also instance state

serviceData

- Extension of xsd:element declaration:

```
<gsdl:serviceData
    name="ncname" type="qname"
    minOccurs="…"? maxOccurs="…"?
    …
    mutability="static"|"constant"|
                          "extendable"|"mutable"?
    modifiable="boolean"? >
</gsdl:serviceData>
```

mutability options

- Static
  - The value is assigned in the WSDL and remains that value (tag <staticServiceDataValues>)
- Constant
  - The value is set when the instance is initialised, and never changes afterwards
- Extendable (default)
  - Once added, elements will remain the same. New values may be added
- Mutable
  - Elements may be removed and new ones may be added

Other options

- maxOccurs (default 1)
  - maximum number of elements
- minOccurs (default 1)
  - minimum number of elements
  - if 0, the element is optional
- Modifiable
  - if true, the requestor may update the value using SetServiceData()
  - subject to constraints of minOccurs, maxOccurs, and mutability

serviceDataValues

- Container for a bag of elements that conform to serviceData declarations

```
<gsdl:serviceDataValues>
    <-- serviceData conformant element -->*
</gsdl:serviceDataValues>
```

  - As child of WSDL portType: "static" value
  - Element in service data, for use with FindServiceData and Subscribe operations

Models for information query

- Pull: GridService::FindServiceData operation
  - Queries this information via extensible query language
- Push: NotificationSource::Subscribe
  - Subscribe to notification of changes to information

Service Data Lifetime Annotations (1)

- goodFrom
  - Declares the time from which the value of the element is said to be valid.
  - This is typically the time at which the contained element was created or aggregated
- goodUntil
  - Declares the time until which the value of the element is said to be valid.
  - This value MUST be greater than the goodFrom time

Service Data Lifetime Annotations (2)

- availableUntil
  - Declares the time until which this element is expected to be available
  - Prior to this time, a client SHOULD be able to query for an updated value of this element
  - This value MUST be greater than the goodFrom time

portType Example

Below is a complete Grid Service portType Example

```
<wsdl:portType name="example">
    <wsdl:operation name="go"> …
    <wsdl:extends portType="ns:GridService">
    <gsdl:serviceData name="sd1" type="xsd:String"
                      mutability="static"/>
    <gsdl:serviceData name="sd2"
                      type="tns:SomeComplexType"/>
    <gsdl:staticServiceDataValues>
            <ns:sd1>initValue</ns:sd1>
    </gsdl:staticServiceDataValues>
</wsdl:portType>
```

Service Data Value Example

- Here is an example of a serviceDataValue element

```
<gsdl:serviceDataValues>
  <n1:e1 goodFrom="2002-04-27T10:20:00.000-06:00"
         goodUntil="2002-04-27T11:20:00.000-06:00"
         availableUntil="2002-04-28T10:20:00.000-06:00">
    <n1:e2>
         abc  ← inherits attributes from n1:e1
    </n1:e2>
    <n1:e3 gsdl:goodUntil="2002-04-27T10:30:00.000-06:00">
         def ← overrides goodUntil attribute
    </n1:e3>
    <n1:e4 gsdl:availableUntil="2002-04-27T20:20:00.000-06:00">
         ghi  ← overrides availableUntil attribute
    </n1:e4>
  </n1:e1>
</gsdl:serviceDataValues>
```

Fundamental Interfaces & Behaviors

- OGSI defines basic patterns of interaction
  - These can be combined with each other with custom patterns in a myriad of ways
- Grid Service Specification focuses on:
  - Atomic, composable patterns in the form of portTypes & service data element types
  - A model for how these are composed
- Complete service descriptions are left to other groups that are defining real services

OGSI Interfaces and behaviours

- Naming and binding
  - Every service has a unique name from which users can discover supported bindings
- Lifecycle
  - Service instances created by factories
  - Destroyed explicitly or by soft state
- Information model
  - Service-specific "Service data" associated with instances
  - Operations for accessing this info
- Notification
  - Interfaces for registering existence and delivering notifications

Defined to Date

- GridService (Required)
  - FindServiceData
  - SetServiceData
  - Destroy
  - RequestTerminationAfter
  - RequestTerminationBefore
- Factory
  - CreateService
- SubscriptionManagement
  - serviceData only
- Registration
  - RegisterService
- HandleResolver
  - FindByHandle
- NotificationSource
  - Subscribe
- NotificationSink
  - DeliverNotification

Lifetime Management

- GS instances created by factory or manually; destroyed explicitly or via soft state
  - Negotiation of initial lifetime with a factory (=service supporting Factory interface)
- GridService interface supports
  - Destroy operation for explicit destruction
  - RequestTerminationBefore/After operations for keepalive
- Soft state lifetime management avoids
  - Explicit client teardown of complex state
  - Resource "leaks" in hosting environments

Summary

- Introduction
- WSDL Conventions
- GSDL Extensions
- Standard Interfaces