

# Service Data und Notifications

## Hauptseminar Grid Computing

Steffen Pingel  
Universität Stuttgart  
Zentrale Dienste der Informatik  
Universitätsstr. 38  
D-70569 Stuttgart  
steffenp@gmx.de

**Überblick** *Die Open Grid Service Infrastructure (OGSI) definiert ein Protokoll, das die verteilte Nutzung von Computing Ressourcen ermöglicht. Service Data ist ein obligatorischer Teil dieser Schnittstelle. Es ermöglicht zustandsbehafteten Grid Services Teile des privaten Zustandes öffentlich sichtbar zu machen. Desweiteren können Metadaten über den Grid Service mittels der Service Data Schnittstelle abgefragt werden. Notifications erweitern das Konzept der Service Data mit Benachrichtigung über Zustandsänderungen. Sie sind eine Implementierung des Observer/Observables Entwurfsmusters. In dieser Ausarbeitung wird ein Überblick über die Verwendung von Service Data und Notifications gegeben und anhand eines Beispiels demonstriert.*

**Schlüsselwörter** Grid Computing, Service Data, Notifications

### Inhaltsverzeichnis

1	<b>Einleitung</b> . . . . .	1
2	<b>Service Data</b> . . . . .	2
3	<b>Notifications</b> . . . . .	6
4	<b>Zusammenfassung</b> . . . . .	10
	<b>Literatur</b> . . . . .	10

### 1 Einleitung

Grid Computing könnte ähnlich des World Wide Webs in Zukunft zu einer Schlüsseltechnologie des Internets werden, da die zunehmend automatisierte Vernetzung verschiedener Dienste einheitliche Schnittstellen erfordert. Dies wird heute bereits mit Web Services vielfach realisiert. Für das Online Shopping wird beispielsweise eine Verknüpfung von der Lagerverwaltung, über das Warenwirtschaftssystem, bis zu möglicherweise extern angesiedelten Abrechnungsdienstleistern benötigt.

In diesen Bereichen bietet sich das Grid Computing mit seinen umfangreichen Möglichkeiten und vorteilhaften Eigenschaften wie Fehlertoleranz, Sicherheit und Lastverteilung als hervorragende Plattform an. Grid Services bieten im Vergleich zu Web Services zusätzliche Schnittstellen für Lifecycle Management, Persistenz und Sicherheitskontrolle.

Die OGSI[5] (Open Grid Services Infrastructure) spezifiziert, basierend auf Web Services, technische Anforderungen an Grid Services. Service Data und Notifications sind Teil der Grid Service Spezifikation, die in der OGSI beschrieben ist. Zum Verständniss der Ausarbeitung ist es notwendig mit den Schlüsseltechnologien des Grid Computing wie GWSDL und Operation Providers vertraut zu sein. Diese Ausarbeitung bezieht sich auf

die Version 1.0 der Open Grid Services Infrastructure. Die gezeigten Quellcode Beispiele beziehen sich auf das Globus Toolkit[3] Version 3.0, das eine frei verfügbare Implementierung der OGSI ist.

Die Konzepte der Service Data und Notifications werden anhand eines Online-Auktionsdienstes beschrieben. Der Auktionsdienst ist so einfach wie möglich gehalten, und dient nur der Demonstration der Implementierung von Service Data und Notifications. Es soll mehreren Clients möglich sein sich mit dem Dienst zu verbinden, um Informationen über das zu versteigernde Produkt abzurufen und Gebote abzugeben. Der Quellcode ist in der Ausarbeitung, soweit zum Verständnis notwendig, angegeben bzw. referenziert, wobei ausgelassene Teile mit drei Punkten “. . .” markiert sind. Der vollständige Quellcode ist im Anhang angegeben.

## 2 Service Data

Web Services folgen dem Anfrage-Antwort Prinzip. Eine Anfrage wird von einem Web-Service, der auf einem Server läuft, verarbeitet und das Ergebniss als Antwort an den Client zurückgeschickt. Grundsätzlich unterliegen Web Services der Einschränkung, daß sie keinen persistenten Zustand haben. Jede Anfrage an einen Web Service erfolgt im Initialzustand<sup>1</sup>.

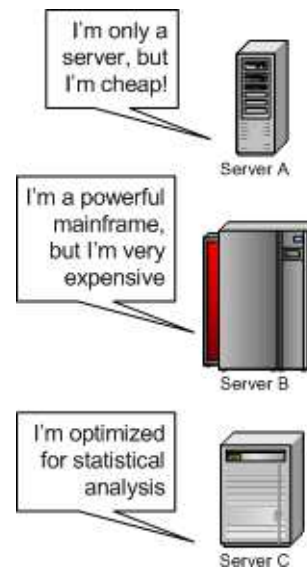
Anders als Web Services können Grid Services über lange Zeiträume laufen und einen internen Zustand verwalten. Dieser Zustand kann sich während der Lebenszeit einer Grid Service Instanz ändern. Um eine standardisierte Möglichkeit zur Abfrage, Änderung und Änderungsbenachrichtigung zu bieten wurde die `serviceData`-Schnittstelle eingeführt.

Die Schnittstelle bietet eine Möglichkeit typisierte Daten über einen Grid Service abzufragen. Die möglichen Anwendungen sind vielfältig. Zu erwähnen ist hier beispielsweise

<sup>1</sup>Es sei angemerkt, daß aktuelle Implementierungen von Web Service Containern diese Einschränkung mittlerweile aufheben.

die Service Discovery.

Abbildung 1: Anwendung von Service Data



Grid Services können Eigenschaften wie Kosten, Geschwindigkeit oder Auslastung mittels Service Data zur Verfügung stellen. Das ermöglicht Agenten einen Vergleich zwischen verschiedenen Diensten und die Auswahl des Dienstes, der die gewünschten Anforderungen am besten erfüllt.

In Verbindung mit dem Universal Description, Discovery and Integration (UDDI) Protokoll können sich Grid Services an zentralen Registern anmelden, so daß eine vollautomatisierte Auswahl von Diensten denkbar ist.

### 2.1 Einbettung von Service Data

Jeder Grid Service legt in seiner öffentlichen Beschreibung fest welche Informationen über den Dienst nach außen hin sichtbar sein sollen und mit welchen Operationen darauf zugegriffen werden kann. Diese Beschreibung erfolgt mittels Web Service Definition Language (WSDL), einem XML-Dialekt, und ähnelt der Deklaration einer objekt-orientierten Schnittstelle mittels einer Interface Definition Language (IDL). Sie ermöglicht Clients die Kommunikation mit dem Grid Service aufzunehmen.

Im abstrakten Teil des XML-Dokument werden sowohl Datentypen als auch Nachrichten beschrieben. Im `portType` Abschnitt werden die definierten Nachrichten benutzt, um die konkreten Operationen eines Dienstes zu definieren. Die `portType`-Deklaration ist im Gegensatz zu der Deklaration der Typen und Nachrichten nicht abstrakt und nicht wiederverwendbar, da ein spezifischer Dienst beschrieben wird.

Die Version 1.1 der WSDL beinhaltet allerdings zwei kritische Einschränkungen: Es ist nicht möglich die WSDL-`portType` Schnittstelle zu erweitern. Da die WSDL auf Web-Services zugeschnitten ist, die keinen Zustand verwalten, war es nicht notwendig eine solche Möglichkeit vorzusehen.

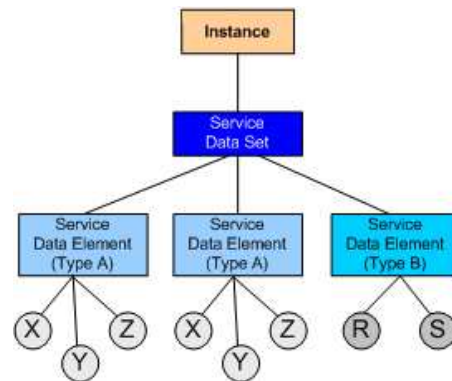
Um den Anforderungen an Service Data gerecht zu werden, wurde die GWSDL als temporäre Übergangslösung eingeführt<sup>2</sup>. Die GWSDL erweitert den WSDL-`portType` in der OGSi in einem neu eingeführten Namensraum. Dieser Namensraum wird in den folgenden Beispielen mit dem Prefix `gwsdl` benutzt. Der GWSDL-`portType` stellt ein `extends`-Attribut zur Verfügung, das es ermöglicht von bestehenden `portTypes` zu erben.

Die Mindestanforderung an Grid Services ist es, die OGSi `GridService`-Schnittstelle zu implementieren. Diese definiert ein Service Data Set, das von jeder Grid Service Instanz zur Verfügung gestellt wird. Das Service Data Set kann leer sein oder mehrere Service Data Elemente enthalten. Jedes Service Data Element muss in der `portType`-Spezifikation des Grid Services angegeben sein. Die Schnittstelle eines Service Data Elements wird in einer Service Data Declaration spezifiziert.

Die Service Data Elemente weisen Ähnlichkeiten mit JavaBeans auf. Jedes Element kann Operationen für Anfragen (`get`) und Änderungen (`set`) enthalten. Wobei die OG-

<sup>2</sup>Die GWSDL soll durch die WSDL Version 1.2 ersetzt werden, die alle benötigten Änderungen beinhaltet, sobald dieser vom W3C als Standard verabschiedet wird.

Abbildung 2: Organisation von Service Data



SI im Gegensatz zur JavaBean-Spezifikation keine Namenskonvention für die Zugriffsoptionen vorgibt. Die Schnittstelle eines JavaBeans wird über eine `BeanInfo` Klasse bekannt gemacht. Das Äquivalent eines Grid Services ist das `serviceData`-Element in der `portType`-Definition.

## 2.2 Service Data Declaration

In einer Service Data Declaration (SDD) werden die Eigenschaften eines Service Data Elements anhand von sieben Attributen festgelegt (siehe auch OGSi, §6.2.1). Die Definition des `sd:ServiceData`-Tags orientiert sich an der Definition des `xsd:element`-Tags aus XML-Schema. Fünf der Attribute haben den gleichen Namen und die gleiche Bedeutung. Die anderen beiden Attribute `mutability` and `modifiability` gibt es nur in der Service Data Deklaration. Obligatorisch ist das `name`- und `type`-Attribut.

- `maxOccurs`: Gibt die maximale Anzahl von Vorkommnissen des Elements in der Service Instanz an. Eine positive Zahl mit dem Standardwert 1.
- `minOccurs`: Gibt die minimale Anzahl von Vorkommnissen des Elements in der Service Instanz an. Eine positive Zahl mit dem Standardwert 1.
- `name`: Der Name des Elements. Der Name muss innerhalb des Namensraumes

mes eindeutig sein. Der Standardwert ist false.

- `nillable`: Gibt an, ob das Element einen Nullwert `xsi:nil='true'` annehmen kann.
- `type`: Ein QName, der den XML-Schema Typ des Elements angibt.
- `modifiable`: Gibt an, ob der Wert mittels der `setServiceData`-Operation geändert werden kann. Ist der Wert false, kann das Element nur gelesen werden. Allerdings kann sich der Wert ändern, wenn er von dem Grid Service direkt gesetzt wird. Der Standardwert ist false.
- `mutability`: Gibt die Art der Änderungen an. Mögliche Werte sind:
  - `static`: Der Wert wird in der WSDL Deklaration zugewiesen und ändert sich während der Lebenszeit eines Grid Services nicht. Das entspricht der Deklaration einer Konstante in einer Programmiersprache.
  - `constant`: Der Wert wird während der Instanzierung des Grid Services zugewiesen und ändert sich während der Lebenszeit nicht.
  - `extendable`: Sobald ein Wert dem Service Data Element zugewiesen wurde, darf er nicht mehr entfernt werden. Er kann nur um zusätzliche Werte erweitert werden.
  - `mutable`: Werte können sowohl hinzugefügt als auch entfernt werden.

Der Standardwert ist `extendable`.

Als Beispiel soll ein Teil der Service Data Deklaration der `GridService`-Schnittstelle dienen, die von jedem Grid Service zur Verfügung gestellt wird (siehe auch Abschnitt 2.5):

```
<sd:serviceData name="interface"
  type="xsd:QName"
  minOccurs="1"
  maxOccurs="unbounded"
  mutability="constant"/>
<sd:serviceData name="factoryHandle"
  type="ogsi:HandleType"
  minOccurs="1"
  maxOccurs="1"
  mutability="constant"
  nillable="true"/>
...
```

### 2.3 Erweiterung des PortTypes mit Service Data

Wie bereits erwähnt wird die öffentliche Schnittstelle eines Grid Services mit dem `portType`-Tag festgelegt. Der PortType bestimmt den Namen, die Art des Dienstes und die Operationen, die in dieser Ausarbeitung nicht betrachtet werden. Teil der öffentlichen Schnittstelle sind auch die Service Data Elements (SDEs), die durch Service Data Declarations definiert werden.

Hier angegeben ist der `portType` eines Auktionsdienstes, der pro Grid Service Instanz eine Auktion verwaltet. Es soll für Clients möglich sein über die Service Data Informationen wie das Enddatum der Auktion oder die Versandkosten abzurufen:

```
<gwsdl:portType name="AuctionPortType"
  extends="ogsi:GridService">
  <operation>...</operation>
  <sd:serviceData name="AuctionData"
    type="data:AuctionDataType"
    minOccurs="1"
    maxOccurs="1"
    mutability="mutable"
    modifiable="false"
    nillable="false"/>
</gwsdl:portType>
</definitions>
</wsdl:definitions>
```

Der angegebene PortType (siehe auch Abbildung 6) definiert einen Grid Service mit einem SDE mit dem Namen `AuctionData` vom Datentyp `AuctionDataType`. Der Datentyp ist in Abbildung 7 als in XML-Schema definiert. Er enthält typisierte Felder wie `endDate` oder `shipping`. Das Service Data Element kommt genau einmal vor und kann von außen nicht verändert werden. Außerdem ist garantiert, daß es keinen Null-Wert hat, d.h. es läuft immer genau eine Auktion.

## Statische Service Data Elements

Ergänzend sei angemerkt, daß zwei Arten von SDEs unterschieden werden: dynamische und statische. Einem statischen SDE wird in der portType-Deklaration bereits ein Wert zugewiesen. Das mutability Attribut muss auf static gesetzt sein und innerhalb des portType Elements in einem <sd:staticServiceDataValues>-Tag die Werte angegeben sein.

## 2.4 Grid Service Implementierung mit Service Data

Neben der Deklaration von Service Data, soll hier gezeigt werden wie die Service Data Elemente im Globus Toolkit 3.0 benutzt, d. h. mit Daten belegt, werden. Wie bereits erwähnt, ist erforderlich die Service Data Declaration, die als GWSDL festgelegt wurde, mit einem entsprechenden Kompilier in ein JavaBean zu transformieren.

Bezogen auf das Beispiel aus Abschnitt 7 wird ein JavaBean mit dem Klassennamen AuctionDataType erzeugt. Die Generierung kann beispielsweise mit dem GWSDL Kompiler aus dem Apache Axis Project[1] erfolgen. Diese Klasse kann instanziiert werden und über die generierten set-Methoden mit Daten belegt werden. Das Bean wird in eine vom Globus Toolkit bereit gestellte Container Klasse vom Typ ServiceData eingefügt und dem Dienst über die add-Methode des Service Data Sets bekannt gemacht.

```
...
// Create Service Data Element
serviceData = this.getServiceDataSet()
    .create("AuctionData");

// Set the value of the SDE
auctionData = new AuctionDataType();
serviceData.setValue(auctionData);

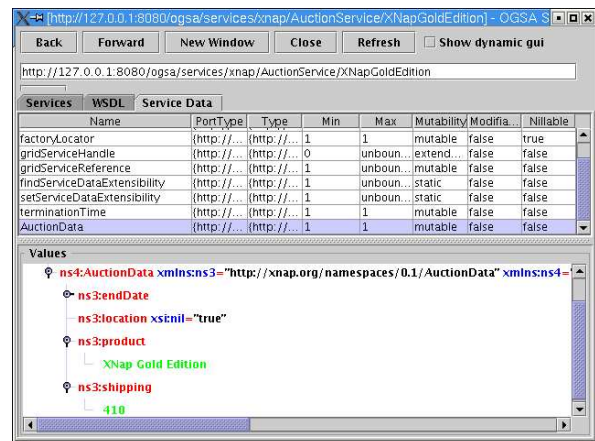
// Set initial values of AuctionServiceData
auctionData.setProduct("XNap Gold Edition");
auctionData.setShipping(410);
auctionData.setEndDate(System.currentTimeMillis()
    + 3 * 24 * 60 * 60 * 1000);

// Add SDE to Service Data Set
this.getServiceDataSet().add(serviceData);
...
```

## 2.5 Standard Service Data

Neben der Möglichkeit eigene Service Data Elemente im portType zu deklarieren, bietet jeder Grid Service Standard Elemente an. Diese dienen der Abfrage von Informationen über den Dienst. Das Globus Toolkit stellt dafür das Werkzeug globus-service-browser zur Verfügung, mit dem sich die Service Data Elemente einer Grid Service Instanz in einer Baum Ansicht darstellen lassen.

Abbildung 3: Globus Service-Browser



Einige der Standard Service Data Elemente des GridService-portTypes sind hier aufgelistet. Eine vollständige Liste mit Typen und Kardinalitäten sind in der OGSI, §6.2.2 beschrieben.

- interface: Eine Liste mit den Namen der implementierten portTypes.
- serviceName: Eine Liste der Namen aller Service Data Elemente. Die Liste beinhaltet die statischen Elemente und kann auch die dynamischen enthalten.
- factoryLocator: Eine Referenz auf die Factory von der der Service erzeugt wurde. Wurde keine Factory verwendet, hat das Element den Wert xsi:nil.
- gridServiceHandle: Eine Liste von Handles auf die Grid Service Instanz. Es

kann weitere Handles geben, die nicht in der Liste aufgeführt werden.

- `gridServiceReference`: Eine Liste von Referenzen auf die Grid Service Instanz. Es kann weitere Referenzen geben, die nicht in der Liste aufgeführt werden.
- `terminationTime`: Die Gültigkeit des Grid Services endet zu diesem Zeitpunkt.

## 2.6 Abfrage von Service Data

Bisher wurde nur die Seite des Grid Services betrachtet. Im folgenden soll auch die Client Seite, d.h. die Abfrage der Service Data, veranschaulicht werden. Zunächst ist es erforderlich ein Handle auf den gewünschten Grid Service zu erhalten. Die Adressierung erfolgt über eine URL, die eine Instanz des Dienstes eindeutig identifiziert. Über das Handle kann eine Verbindung mit dem Dienst hergestellt werden.

Über die `findServiceData` Methode kann ein Service Data Element abgefragt werden. Das Globus Toolkit stellt die Helferklasse `QueryHelper` zur Verfügung, die Toolkit konforme Anfragen erzeugen kann.

```
// Get Service Data Element
ExtensibilityType extensibility
= gridService.findServiceData
  (QueryHelper.getNamesQuery("AuctionData"));
ServiceDataValuesType serviceData
= AnyHelper.getAsServiceDataValues(extensibility);
AuctionDataType auctionData
= (AuctionDataType)AnyHelper.getAsSingleObject
  (serviceData, AuctionDataType.class);
```

In dem angegebenen Beispiel (siehe auch Abbildung 10) wird ein Service Data Element mit dem Namen `AuctionData` abgefragt. Der Rückgabewert ist vom Typ `ExtensibilityType` und kann mit der Helferklasse `AnyHelper` in den eigentlichen Datentyp `AuctionDataType` gecastet werden. Über dieses Objekt können nun die Werte mittels der `get`-Methoden abgefragt werden.

## Kommunikation

Die Kommunikation zwischen Client und Grid Service erfolgt über das Simple Object Access Protocol (SOAP). Die Betrachtung der ausgetauschten Nachrichten ermöglicht ein gutes Verständnis des Datenflusses.

Der Client eröffnet die HTTP-Sitzung mit einer Post-Nachricht. Hier findet sich die Anfrage an die `findServiceData`-Methode des Grid Service wieder, sowie der Ausdruck, der anhand des Namens das Service Data Element anfragt.

```
POST /ogsa/services/xnap/AuctionService/xnap HTTP/1.0
...
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
  xmlns:soapenv
    = "http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <findServiceData
      xmlns
        = "http://www.gridforum.org/namespaces/2003/03/OGSI">
      <queryExpression>
        <queryByServiceDataNames
          xsi:type="ns1:QNamesType"
          xmlns:ns1
            = "http://www.gridforum.org/namespaces/2003/03/OGSI">
          <name>AuctionData</name>
        </queryByServiceDataNames>
      </queryExpression>
    </findServiceData>
  </soapenv:Body>
</soapenv:Envelope>
```

Die Antwort des Grid Services enthält die Daten. Diese werden vom Toolkit auf der Client Seite in ein Java Klasse vom Typ `AuctionDataType` deserialisiert.

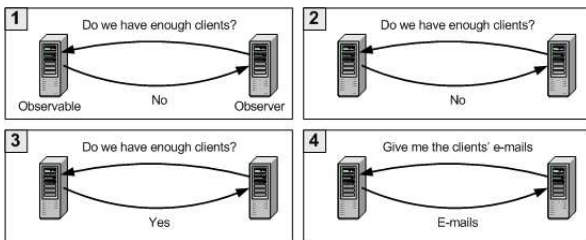
```
HTTP/1.0 200 OK
...
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <findServiceDataResponse
      xmlns="http://www.gridforum.org/...">
      <result
        xsi:type="ns1:ExtensibilityType"
        xmlns:ns1="http://www.gridforum.org/...">
        <ns2:serviceDataValues
          xmlns:ns2="http://www.gridforum.org/...">
          <ns4:AuctionData
            xsi:type="ns3:AuctionDataType"
            xmlns:ns3="http://xnap.org/namespaces/0.1/AuctionData"
            xmlns:ns4="http://xnap.org/namespaces/0.1/Auction">
            <ns3:endDate>1105543546415</ns3:endDate>
            <ns3:location xsi:nil="true"/>
            <ns3:product>XNep Gold Edition</ns3:product>
            <ns3:shipping>410</ns3:shipping>
          </ns4:AuctionData>
        </ns2:serviceDataValues>
      </result>
    </findServiceDataResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

## 3 Notifications

Notifications bezeichnen ein populäres Entwurfsmuster, das in vielen Architekturen ver-

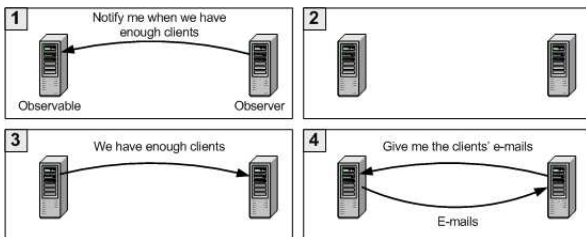
wendet wird. Bekannte Beispiele sind z.B. das Observer/Observable-Muster[2] oder die Model-View-Controller Architektur. Allgemein ausgedrückt benachrichtigt bei einer Notification ein Sender einen oder mehrere Empfänger über eine Zustandsänderung. Im Globus Toolkit Jargon wird von einem Notification Source (Observable) gesprochen, das Benachrichtigungen an einen Notification Sink (Observer) übermittelt.

Abbildung 4: Polling



Bei dem bisher betrachteten Polling-Ansatz war es Aufgabe des Clients den Zustand des Grid Services abzufragen. Um Zustandsänderungen zu bemerken war es für den Client notwendig, regelmäßig den Zustand abzufragen und mit dem Ergebnis der vorhergehenden Abfrage zu vergleichen (siehe Abbildung 4).

Abbildung 5: Notifying



Mit Notifications ist es für einen Client nicht mehr erforderlich periodisch den Zustand eines Grid Services abzufragen. Der Client kann sich bei dem Grid Service als Empfänger registrieren und wird bei einer Änderung des Zustandes benachrichtigt.

## Pull vs. Push Notifications

Es werden zwei Benachrichtigungsvarianten unterschieden: Pull und Push. Bei der Pull-Benachrichtigung wird dem Empfänger lediglich mitgeteilt, daß sich der Zustand geändert hat. Es werden keine Informationen über die Art der Änderung selbst übertragen. Es ist die Aufgabe des Empfängers entsprechende Abfragen zu machen. Daher die Bezeichnung "Pull": Der Empfänger holt sich aktiv Informationen über die Änderung.

Pull-Benachrichtigung ist in zwei Szenarien vorteilhaft:

- Die Empfänger benötigen verschieden Daten.
- Es werden keine zusätzlichen Daten benötigt, eine Benachrichtigung über eine Änderung ist ausreichend.

Bei der Push-Benachrichtigung werden Informationen über die Änderung bei der Benachrichtigung mitgeschickt. Das vermeidet einen zusätzlichen Kommunikationsaufwand, da die Empfänger diese Daten nicht in einem separaten Vorgang abfragen müssen.

## Service Data und Notifications

Im Globus Toolkit beziehen sich Benachrichtigungen auf Service Data Elemente. Clients können sich bei Service Data Elementen registrieren und über Änderungen benachrichtigen lassen. Im Globus Toolkit wird die Push-Benachrichtigung verwendet. Es wird immer das geänderte Service Data Element mitübertragen. Pull-Benachrichtigungen lassen sich allerdings mit Hilfe eines Service Data Elements, das keine Daten enthält, realisieren.

### 3.1 Erweiterung des PortTypes mit Notifications

Um einen Grid Service als Notification Source (Sender) zu kennzeichnen wird der portType des Grid Services um die `ogsi:NotificationSource`-Schnittstelle erweitert. Diese Schnittstelle

definiert die Operationen `subscribe` und `subscribeByServiceDataNames`, die es Notification Sinks (Empfängern) ermöglichen sich auf gewünschte Service Data Elemente zu registrieren.

Zur Demonstration soll wieder der Online-Auktionsdienst dienen. Um Bietern die Möglichkeit zu geben auf Gebote anderer Bieter zu reagieren, sollen Clients informiert werden, sobald ein Bieter ein Höchstgebot abgegeben hat. Dazu wird der `portType`, wie in Abbildung 11 angegeben, erweitert.

```
<gwsdl:portType name="AuctionPortType"
  extends="ogsi:GridService
    ogsi:NotificationSource">
  <operation>...</operation>
  <sd:serviceData>...</sd:serviceData>
</gwsdl:portType>
</definitions>
```

### 3.2 Grid Service Implementierung mit Notifications

Die Implementierung des Grid Service ändert sich durch die Änderung des `portTypes` nur geringfügig, da das Globus Toolkit Klassen zur Verfügung, die eine Implementierung von Benachrichtigungen enthalten. In der Implementierung in Abbildung 12 wird allerdings nicht von der abstrakten Klasse `GridServiceImpl` geerbt, sondern lediglich ein Operation-Provider implementiert. Für die eigentliche Grid Service Instanz wird eine Klasse aus dem Globus Toolkit verwendet, die im Deployment-Descriptor (siehe Abbildung 13) festgelegt ist. Hier ist auch ein weiterer Operation-Provider der Klasse `NotificationSourceProvider` angegeben, der die Implementierung der NotificationSource-Schnittstelle übernimmt. Ein Operation-Provider stellt nur eine Menge von Operationen zur Verfügung, die von einem Grid Service übernommen werden können. Das ermöglicht die Modularisierung und Wiederverwendbarkeit von Operationen.

Entscheidend ist die Änderung in der `bid`-Methode. Diese Methode wird von Clients aufgerufen, um ein Gebot abzugeben:

```
public synchronized boolean
```

```
    bid(String nick, int value)
    throws RemoteException {
    if (value > auctionData.getCurrentPrice()) {
    auctionData.setCurrentBuyer(nick);
    auctionData.setCurrentPrice(value);
    serviceData.notifyChange();
    return true;
    }
    return false;
}
```

Im Fall eines Höchstgebotes wird das Service Data Element in der Variable `auctionData` aktualisiert und `serviceData.notifyChange()` aufgerufen. Daraufhin wird das geänderte Service Data Element an alle Clients, die registriert sind, übertragen. Die Übertragung wird vom Globus Toolkit vorgenommen und wird im nächsten Abschnitt näher betrachtet.

### 3.3 Kommunikation

Die Kommunikation zwischen Client und Grid Service ist im Vergleich zum ersten Beispiel in dem nur eine Abfrage von Service Data betrachtet wurde erheblich komplexer geworden. Der Client ist streng genommen nicht mehr nur Client, sondern übernimmt auch die Rolle eines Servers.

Um Benachrichtigungen empfangen zu können muss der Client eine Adresse zur Verfügung stellen mit der sich der Sender, sprich der Grid Service, verbinden kann.

Als Beispiel soll eine Auktion mit zwei Bietern dienen. Beide Bieter sind mit dem Auktionsdienst verbunden und haben sich bei dem Grid Service als Empfänger registriert. Für eine Registrierung schickt jeder Client eine solche SOAP-Nachricht an den Grid Service:

```
POST .../xnap/AuctionService2/XNapGoldEdition
SOAPAction: "http://www.grid...#subscribe"
...
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <subscribe
      xmlns="http://www.gridforum.org/namespaces/2003/03/OGSI">
      <subscriptionExpression>
        <subscribeByServiceDataNames
          xsi:type="ns1:SubscribeByNameType"
          xmlns:ns1="http://www.gridforum.org/namespaces/2003/03/OGSI">
          <name>AuctionData</name>
        </subscribeByServiceDataNames>
      </subscriptionExpression>
    </subscribe>
  </body>
</handle>http://127.0.0.1:35724/ogsa/services/sinks
/1105313939301-127.0.0.1-35724-33459432-13948523</handle>
</sink>
```

```
<expirationTime>2005-01-10T00:39:00.974Z</expirationTime>
</subscribe>
</soapenv:Body>
</soapenv:Envelope>
```

In der Nachricht wird die `subscribeByServiceDataNames-Operation` verwendet, um Benachrichtungen über das `Service Data Element` mit dem Namen `AuctionData` zu erhalten. Als handle wird eine URL übertragen, die einen Socket, der vom Client geöffnet wurde identifiziert.

Nachdem alle Clients registriert sind, gibt "BieterA" ein Gebot von "200" ab, das von dem Auktionsdienst als Höchstgebot akzeptiert wird. Die Clientanwendung von "BieterA" sieht folgendermaßen aus:

```
Your nick: BieterA

Current buyer:
Current price: 100
-----
Your bid: 200

> Your bid is currently the highest!
Your bid:
```

Daraufhin werden alle verbundenen Clients über das neue Höchstgebot mittels einer Benachrichtung informiert. Hier ist die Clientanwendung von "BieterB" zu sehen. "BieterB" hat noch kein Gebot abgegeben, bekommt aber die Benachrichtung, daß "BieterA" zum Höchstbietenden geworden ist:

```
Your nick: BieterB

Current buyer:
Current price: 100
-----
Your bid:

Current buyer: BieterA
Current price: 200
-----
Your bid:
```

Die Benachrichtigung an den Client erfolgt über die in der `subscribe`-Nachricht angegebene Adresse. Im konkreten Beispiel an den Port 35724, der auf die IP-Adresse 127.0.0.1 gebunden ist. Die Benachrichtigung enthält das geänderte `Service Data Element`:

```
Host: 127.0.0.1:35724
SOAPAction: "http://www.grid...#deliverNotification"
...
<?xml version="1.0" encoding="UTF-8"?>
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <deliverNotification
xmlns="http://www.gridforum.org/...">
      <message>
        <ns1:serviceDataValues
xmlns:ns1="http://www.gridforum.org/...">
          <ns3:AuctionData
xsi:type="ns2:AuctionDataType"
xmlns:ns2="http://xnap.org/namespaces/0.2/AuctionData"
xmlns:ns3="http://xnap.org/namespaces/0.2/Auction">
            <ns2:currentBuyer>BieterA</ns2:currentBuyer>
            <ns2:currentPrice>200</ns2:currentPrice>
            <ns2:endDate>1105543538173</ns2:endDate>
            <ns2:location>Stuttgart</ns2:location>
            <ns2:product>XNap Gold Edition</ns2:product>
            <ns2:shipping>410</ns2:shipping>
          </ns3:AuctionData>
        </ns1:serviceDataValues>
      </message>
    </deliverNotification>
  </soapenv:Body>
</soapenv:Envelope>
```

### 3.4 Empfangen von Benachrichtigungen

Abschliessend soll noch die Implementierung des erweiterten Clients gezeigt werden. Der Client muss einen Port zur Verfügung stellen mit dem die Sender sich verbinden können. Das Globus Toolkit stellt dazu die `NotificationSinkManager`-Klasse zur Verfügung.

In dem hier angegebenen Beispiel wird, nachdem die Verbindung zum Auktionsdienst hergestellt wurde, der Port geöffnet und die `subscribe`-Nachricht mit dem Aufruf `notifManager.addListener()` verschickt. Als Empfänger wird ein Objekt vom Typ `NotificationSinkCallback`, in diesem Fall `this`, übergeben.

```
public class AuctionClient extends ServicePropertiesImpl
implements NotificationSinkCallback {

    public AuctionClient(String nick) throws Exception {
        // Connect to AuctionService
        ...

        // Start listening to the AuctionService
        NotificationSinkManager notifManager
        = NotificationSinkManager.getManager();
        notifManager.startListening
        (NotificationSinkManager.MAIN_THREAD);
        String sinkHandle
        = notifManager.addListener
        ("AuctionData", null, GSH, this);

        // Start the bidding
        ...

        // Clean up: Stop Listening
        notifManager.removeListener(sinkHandle);
        notifManager.stopListening();
    }
}
```

Die `NotificationSinkCallback`-Schnittstelle erfordert die Implementierung

der `deliverNotification`-Methode. Die Methode wird aufgerufen, sobald eine Benachrichtigung empfangen wird. Die Nachricht wird in einem `ExtensibilityType` übergeben, der in den entsprechenden Typ gecastet werden kann. Für den Auktionsclient ist die Implementierung hier angegeben:

```
public void deliverNotification(ExtensibilityType any)
    throws RemoteException {
    try {
        // Service Data has changed. Show new data.
        ServiceDataValuesType serviceData
            = AnyHelper.getAsServiceDataValues(any);
        AuctionDataType auctionData
            = (AuctionDataType)AnyHelper.getAsSingleObject
                (serviceData, AuctionDataType.class);

        print(auctionData);
    } catch(Exception exc) { ... }
}
```

Hier wird wie in dem bereits bekannten Beispiel zur Service Data das empfangene Service Data Element `any` mit Helfermethoden in einen `AuctionDataType` gecastet und anschliessend mit der `print`-Methode ausgegeben (siehe Abbildung 14).

## 4 Zusammenfassung

In dieser Ausarbeitung wurde gezeigt, daß Service Data eine notwendige Erweiterung für zustandsbehaftete Grid Services im Gegensatz zu zustandslosen Web Services bieten. Es wurde gezeigt wie Service Data deklariert wird, von Grid Services verfügbar gemacht wird und wie ein Client auf Service Data Elemente zugreifen kann.

Desweiteren wurde gezeigt wie mit Notifications Benachrichtigungen über Änderungen von Service Data Elementen an Clients weiter gegeben werden können.

### 4.1 Danksagungen

Dank gilt Magdalena Schneider für das geduldige Korrekturlesen dieser Ausarbeitung.

Desweiteren gilt Borja Sotomayor Dank für das Globus Toolkit Programmer's Tutorial[6] und die darin enthaltenen Abbildungen: This product includes material developed by the Globus Project (<http://www.globus.org/>).

## 4.2 Urheberrechtshinweis

Diese Ausarbeitung kann gemäß der Bedingungen der Globus Toolkit Public License[4] verwendet werden.

## Abbildungsverzeichnis

1	Anwendung von Service Data	2
2	Organisation von Service Data	3
3	Globus Service-Browser . . . .	5
4	Polling . . . . .	7
5	Notifying . . . . .	7
6	Auction.gwsdl . . . . .	11
7	AuctionDataType.xsd . . . . .	11
8	AuctionService.java . . . . .	12
9	Auction.wsdd . . . . .	12
10	AuctionClient.java . . . . .	13
11	Auction2.gwsdl . . . . .	14
12	AuctionProvider.java . . . . .	15
13	Auction2.wsdd . . . . .	15
14	AuctionClient2.java . . . . .	16

## Literatur

- [1] <http://ws.apache.org/axis/>.
- [2] GAMMA, ERICH ET AL.: *Design Patterns*. Addison-Wesely, 1994.
- [3] <http://www.globus.org/>.
- [4] <http://www.globus.org/toolkit/license.html>.
- [5] *Open Grid Services Infrastructure (OGSI)*. Global Grid Forum, 2003.
- [6] SOTOMAYOR, BORJA: *Globus Toolkit 3 Programmer's Tutorial*. 2003.

## Abbildung 6: Auction.gwsdl

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="AuctionService"
  targetNamespace="http://xnap.org/namespaces/0.1/Auction"
  xmlns:tns="http://xnap.org/namespaces/0.1/Auction"
  xmlns:data="http://xnap.org/namespaces/0.1/AuctionData"
  xmlns:ogsi="http://www.gridforum.org/namespaces/2003/03/OGSI"
  xmlns:gwsdl="http://www.gridforum.org/namespaces/2003/03/gridWSDLExtensions"
  xmlns:sd="http://www.gridforum.org/namespaces/2003/03/serviceData"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns="http://schemas.xmlsoap.org/wsdl/">

  <import location="../../../ogsi/ogsi.gwsdl" namespace="http://www.gridforum.org/namespaces/2003/03/OGSI"/>
  <import location="AuctionDataType.xsd" namespace="http://xnap.org/namespaces/0.1/AuctionData"/>

  <types>
  <xsd:schema targetNamespace="http://xnap.org/namespaces/0.1/Auction"
    attributeFormDefault="qualified"
    elementFormDefault="qualified"
    xmlns="http://www.w3.org/2001/XMLSchema">

    <xsd:element name="bid" type="tns:bid">
      <xsd:complexType name="bid">
        <xsd:sequence>
          <xsd:element name="nick" type="xsd:string"/>
          <xsd:element name="amount" type="xsd:int"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>

    <xsd:element name="bidResponse" type="tns:bidResponse">
      <xsd:complexType name="bidResponse">
        <xsd:sequence>
          <xsd:element name="return" type="xsd:boolean"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
  </xsd:schema>
</types>

  <message name="BidInputMessage">
    <part name="parameters" element="tns:bid"/>
  </message>
  <message name="BidOutputMessage">
    <part name="parameters" element="tns:bidResponse"/>
  </message>

  <gwsdl:portType name="AuctionPortType" extends="ogsi:GridService">
    <operation name="bid">
      <input message="tns:BidInputMessage"/>
      <output message="tns:BidOutputMessage"/>
      <fault name="Fault" message="ogsi:FaultMessage"/>
    </operation>
    <sd:serviceData name="AuctionData"
      type="data:AuctionDataType"
      minOccurs="1"
      maxOccurs="1"
      mutability="mutable"
      modifiable="false"
      nillable="false">
    </sd:serviceData>
  </gwsdl:portType>
</definitions>
```

## Abbildung 7: AuctionDataType.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="AuctionData"
  targetNamespace="http://xnap.org/namespaces/0.2/AuctionData"
  xmlns:tns="http://xnap.org/namespaces/0.2/AuctionData"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
<wsdl:types>
<schema targetNamespace="http://xnap.org/namespaces/0.2/AuctionData"
  attributeFormDefault="qualified"
  elementFormDefault="qualified"
  xmlns="http://www.w3.org/2001/XMLSchema">

  <complexType name="AuctionDataType">
    <sequence>
      <element name="currentBuyer" type="string"/>
      <element name="currentPrice" type="int"/>
      <element name="endDate" type="long"/>
      <element name="location" type="string"/>
      <element name="product" type="string"/>
      <element name="shipping" type="int"/>
    </sequence>
  </complexType>
</schema>
</wsdl:types>
</wsdl:definitions>
```

## Abbildung 8: AuctionService.java

```
package org.xnap.plugin.grid.auction.impl;

import org.globus.ogsa.*;
import org.globus.ogsa.impl.ogsi.GridServiceImpl;
import org.xnap.plugin.grid.auction.wsdl.AuctionPortType;
import org.xnap.plugin.grid.auction.AuctionDataType;
import java.rmi.RemoteException;

public class AuctionService extends GridServiceImpl implements AuctionPortType {
    public static final int THREE_DAYS = 3 * 24 * 60 * 60 * 1000;

    private ServiceData serviceData;
    private AuctionDataType auctionData;
    private String currentBuyer;
    private int currentPrice;

    public AuctionService() { super("Simple Auction Grid Service"); }

    public void postCreate(GridContext context) throws GridServiceException {
        // Create Service Data Element
        serviceData = this.getServiceDataSet().create("AuctionData");

        // Set the value of the SDE to a AuctionDataType instance
        auctionData = new AuctionDataType();
        serviceData.setValue(auctionData);

        // Set initial values of AuctionServiceData
        auctionData.setProduct("XNAP Gold Edition");
        auctionData.setShipping(410);
        auctionData.setEndDate(System.currentTimeMillis() + THREE_DAYS);

        currentPrice = 100;

        // Add SDE to Service Data Set
        this.getServiceDataSet().add(serviceData);
    }

    public synchronized boolean bid(String nick, int value) throws RemoteException {
        if (value > currentPrice) {
            currentPrice = value;
            currentBuyer = nick;
            return true;
        }
        return false;
    }
}
```

## Abbildung 9: Auction.wsdd

```
<?xml version="1.0"?>
<deployment name="defaultServerConfig" xmlns="http://xml.apache.org/axis/wsdd/"
  xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">
  <service name="xnap/AuctionService" provider="Handler" style="wrapped">
    <!-- Factory Parameters -->
    <parameter name="name" value="Auction Service"/>
    <parameter name="instance-name" value="Auction Instance"/>
    <parameter name="instance-schemaPath" value="schema/org.xnap.plugin.grid.auction/Auction/Auction_service.wsdl"/>
    <parameter name="instance-className" value="org.xnap.plugin.grid.auction.impl.AuctionService"/>

    <!-- FactoryService Parameters -->
    <parameter name="allowedMethods" value="*/>
    <parameter name="className" value="org.gridforum.ogsi.Factory"/>
    <parameter name="persistent" value="true"/>

    <parameter name="baseClassName" value="org.globus.ogsa.impl.ogsi.PersistentGridServiceImpl"/>
    <parameter name="schemaPath" value="schema/ogsi/ogsi_factory_service.wsdl"/>
    <parameter name="handlerClass" value="org.globus.ogsa.handlers.RPCURIPProvider"/>
    <parameter name="factoryCallback" value="org.globus.ogsa.impl.ogsi.DynamicFactoryCallbackImpl"/>
    <parameter name="operationProviders" value="org.globus.ogsa.impl.ogsi.FactoryProvider org.globus.ogsa.impl.ogsi.NotificationSourceProvider"/>
  </service>
</deployment>
```

## Abbildung 10: AuctionClient.java

```
package org.xnap.plugin.grid.auction.client;

import org.gridforum.ogsi.*;
import org.globus.ogsa.utils.*;
import org.xnap.plugin.grid.auction.wsdl.service.AuctionServiceGridLocator;
import org.xnap.plugin.grid.auction.wsdl.AuctionPortType;
import org.xnap.plugin.grid.auction.AuctionDataType;
import java.net.URL;
import java.io.BufferedReader;
import java.io.InputStreamReader;

public class AuctionClient {
    public static void main(String[] args) {
        try {
            // Get command-line arguments
            URL GSH = new java.net.URL("http://127.0.0.1:8080/ogsa/services/xnap/AuctionService/xnap");

            // Get a reference to the Auction PortType
            AuctionServiceGridLocator serviceLocator = new AuctionServiceGridLocator();
            AuctionPortType auction = serviceLocator.getAuctionServicePort(GSH);

            // Get a reference to the GridService PortType
            OGSIServiceGridLocator locator = new OGSIServiceGridLocator();
            GridService gridService = locator.getGridServicePort(GSH);

            // Get Service Data Element
            ExtensibilityType extensibility = gridService.findServiceData(QueryHelper.getNamesQuery("AuctionData"));
            ServiceDataValuesType serviceData = AnyHelper.getAsServiceDataValues(extensibility);
            AuctionDataType auctionData = (AuctionDataType)AnyHelper.getAsSingleObject(serviceData, AuctionDataType.class);

            // Start the bidding
            BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
            while (true) {
                System.out.println("Shipping: "
                    + auctionData.getShipping());
                System.out.println("Location: "
                    + auctionData.getLocation());
                System.out.println("-----");
                System.out.print("Your bid: ");

                String s = in.readLine();
                if (s == null) break;

                if (auction.bid("foobar", Integer.parseInt(s))) {
                    System.out.println("> Your bid is currently the highest!");
                } else {
                    System.out.println("> Sorry, try again.");
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

## Abbildung 11: Auction2.gwsdl

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="AuctionService"
  targetNamespace="http://xnap.org/namespaces/0.2/Auction"
  xmlns:tns="http://xnap.org/namespaces/0.2/Auction"
  xmlns:data="http://xnap.org/namespaces/0.2/AuctionData"
  xmlns:ogsi="http://www.gridforum.org/namespaces/2003/03/OGSI"
  xmlns:gwsdl="http://www.gridforum.org/namespaces/2003/03/gridWSDLExtensions"
  xmlns:sd="http://www.gridforum.org/namespaces/2003/03/serviceData"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns="http://schemas.xmlsoap.org/wsdl/">

  <import location="../../../ogsi/ogsi.gwsdl"
    namespace="http://www.gridforum.org/namespaces/2003/03/OGSI"/>
  <import location="AuctionDataType.xsd"
    namespace="http://xnap.org/namespaces/0.2/AuctionData"/>

  <types>
  <xsd:schema targetNamespace="http://xnap.org/namespaces/0.2/Auction"
    attributeFormDefault="qualified"
    elementFormDefault="qualified"
    xmlns="http://www.w3.org/2001/XMLSchema">

    <xsd:element name="bid" type="tns:bid">
      <xsd:complexType name="bid">
        <xsd:sequence>
          <xsd:element name="nick" type="xsd:string"/>
          <xsd:element name="amount" type="xsd:int"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>

    <xsd:element name="bidResponse" type="tns:bidResponse">
      <xsd:complexType name="bidResponse">
        <xsd:sequence>
          <xsd:element name="return" type="xsd:boolean"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
  </xsd:schema>
</types>

  <message name="BidInputMessage">
    <part name="parameters" element="tns:bid"/>
  </message>
  <message name="BidOutputMessage">
    <part name="parameters" element="tns:bidResponse"/>
  </message>

  <gwsdl:portType name="AuctionPortType" extends="ogsi:GridService ogsi:NotificationSource">
    <operation name="bid">
      <input message="tns:BidInputMessage"/>
      <output message="tns:BidOutputMessage"/>
      <fault name="Fault" message="ogsi:FaultMessage"/>
    </operation>
    <sd:serviceData name="AuctionData"
      type="data:AuctionDataType"
      minOccurs="1"
      maxOccurs="1"
      mutability="mutable"
      modifiable="false"
      nillable="false">
    </sd:serviceData>
  </gwsdl:portType>
</definitions>
```

## Abbildung 12: AuctionProvider.java

```
package org.xnap.plugin.grid.auction2.impl;

import org.globus.ogsa.*;
import org.xnap.plugin.grid.auction2.wsdl.AuctionPortType;
import org.xnap.plugin.grid.auction2.AuctionDataType;
import java.rmi.RemoteException;
import javax.xml.namespace.QName;

public class AuctionProvider implements OperationProvider,
GridServiceCallback {
    public static final int THREE_DAYS = 3 * 24 * 60 * 60 * 1000;

    private ServiceData serviceData;
    private AuctionDataType auctionData;

    private static final QName[] operations = new QName[] { new QName("", "") };
    private GridServiceBase base;

    public AuctionProvider() {}

    public void initialize(GridServiceBase base) throws GridServiceException { this.base = base; }

    public QName[] getOperations() { return operations; }

    public void postCreate(GridContext context) throws GridServiceException {
        // Create Service Data Element
        serviceData = base.getServiceDataSet().create("AuctionData");

        // Set the value of the SDE to a AuctionDataType instance
        auctionData = new AuctionDataType();
        serviceData.setValue(auctionData);

        // Set initial values of AuctionServiceData
        auctionData.setProduct("XNAP Gold Edition");
        auctionData.setCurrentPrice(100);
        auctionData.setLocation("Stuttgart");
        auctionData.setShipping(410);
        auctionData.setEndDate(System.currentTimeMillis() + THREE_DAYS);

        // Add SDE to Service Data Set
        base.getServiceDataSet().add(serviceData);
    }

    public synchronized boolean bid(String nick, int value)
    throws RemoteException {
        if (value > auctionData.getCurrentPrice()) {
            auctionData.setCurrentBuyer(nick);
            auctionData.setCurrentPrice(value);
            serviceData.notifyChange();
            return true;
        }
        return false;
    }

    // Empty callback methods
    public void preCreate(GridServiceBase base) throws GridServiceException {}
    public void preDestroy(GridContext context) throws GridServiceException {}
    public void activate(GridContext context) throws GridServiceException {}
    public void deactivate(GridContext context) throws GridServiceException {}
}
```

## Abbildung 13: Auction2.wsdd

```
<?xml version="1.0"?>
<deployment name="defaultServerConfig" xmlns="http://xml.apache.org/axis/wsdd/"
xmlns:java="http://xml.apache.org/axis/wsdd/providers/java">
<service name="xnap/AuctionService2" provider="Handler" style="wrapped">
<!-- Factory Parameters -->
<parameter name="name" value="Enhanced Auction Service"/>
<parameter name="instance-name" value="Enhanced Auction Instance"/>
<parameter name="instance-schemaPath" value="schema/org.xnap.plugin.grid.auction2/Auction/Auction_service.wsdl"/>
<parameter name="instance-className" value="org.xnap.plugin.grid.auction2.wsdl.AuctionPortType"/>
<parameter name="instance-baseClassName" value="org.globus.ogsa.impl.ogsi.GridServiceImpl"/>
<parameter name="instance-operationProviders"
value="org.xnap.plugin.grid.auction2.impl.AuctionProvider org.globus.ogsa.impl.ogsi.NotificationSourceProvider"/>

<!-- FactoryService Parameters -->
<parameter name="allowedMethods" value=""/>
<parameter name="className" value="org.gridforum.ogsi.Factory"/>
<parameter name="persistent" value="true"/>

<parameter name="baseClassName" value="org.globus.ogsa.impl.ogsi.PersistentGridServiceImpl"/>
<parameter name="schemaPath" value="schema/ogsi/ogsi_notification_factory_service.wsdl"/>
<parameter name="handlerClass" value="org.globus.ogsa.handlers.RPCURIPProvider"/>
<parameter name="factoryCallback" value="org.globus.ogsa.impl.ogsi.DynamicFactoryCallbackImpl"/>
<parameter name="operationProviders" value="org.globus.ogsa.impl.ogsi.FactoryProvider org.globus.ogsa.impl.ogsi.NotificationSourceProvider"/>
</service>
</deployment>
```

## Abbildung 14: AuctionClient2.java

```
package org.xnap.plugin.grid.auction2.client;

import org.globus.ogsa.client.managers.NotificationSinkManager;
import org.globus.ogsa.NotificationSinkCallback;
import org.globus.ogsa.impl.core.service.ServicePropertiesImpl;
import org.gridforum.ogsi.*;
import org.globus.ogsa.utils.*;
import org.xnap.plugin.grid.auction2.wsdl.service.AuctionServiceGridLocator;
import org.xnap.plugin.grid.auction2.wsdl.AuctionPortType;
import org.xnap.plugin.grid.auction2.AuctionDataType;
import java.rmi.RemoteException;
import java.net.URL;
import java.io.*;

public class AuctionClient2 extends ServicePropertiesImpl implements NotificationSinkCallback {
    public AuctionClient(String nick) throws Exception {
        // Set schema root, important: Trailing !!
        System.setProperty("org.globus.ogsa.schema.root", "http://localhost:8080/");

        // Get command-line arguments
        HandleType GSH = new HandleType("http://127.0.0.1:8080/ogsa/services/xnap/AuctionService2/XNapGoldEdition");

        // Get a reference to the Auction PortType
        AuctionServiceGridLocator serviceLocator = new AuctionServiceGridLocator();
        AuctionPortType auction = serviceLocator.getAuctionServicePort(GSH);

        // Get a reference to the GridService PortType
        OGSIServiceGridLocator locator = new OGSIServiceGridLocator();
        GridService gridService = locator.getGridServicePort(GSH);

        // Get Service Data Element
        ExtensibilityType extensibility = gridService.findServiceData(QueryHelper.getNamesQuery("AuctionData"));
        ServiceDataValuesType serviceData = AnyHelper.getAsServiceDataValues(extensibility);
        AuctionDataType auctionData = (AuctionDataType)AnyHelper.getAsSingleObject(serviceData, AuctionDataType.class);

        System.out.println("Your nick: " + nick);
        System.out.println("Shipping: " + auctionData.getShipping());
        System.out.println("Location: " + auctionData.getLocation());
        print(auctionData);

        // Start listening to the AuctionService
        NotificationSinkManager notifManager = NotificationSinkManager.getManager();
        notifManager.startListening(NotificationSinkManager.MAIN_THREAD);
        String sinkHandle = notifManager.addListener("AuctionData", null, GSH, this);

        // Start the bidding
        BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
        while (true) {
            System.out.println("Your bid: ");
            String s = in.readLine();
            if (s == null) {
                break;
            }
            else if (auction.bid(nick, Integer.parseInt(s)) {
                System.out.println("> Your bid is currently the highest!");
            }
            else {
                System.out.println("> Sorry, try again!");
            }
        }

        // Clean up: Stop Listening
        notifManager.removeListener(sinkHandle);
        notifManager.stopListening();
    }

    public void print(AuctionDataType auctionData) {
        System.out.println("-----");
        System.out.println("Current buyer: " + auctionData.getCurrentBuyer());
        System.out.println("Current price: " + auctionData.getCurrentPrice());
        System.out.println("-----");
    }

    public void deliverNotification(ExtensibilityType any) throws RemoteException {
        try {
            // Service Data has changed. Show new data.
            ServiceDataValuesType serviceData = AnyHelper.getAsServiceDataValues(any);
            AuctionDataType auctionData = (AuctionDataType)AnyHelper.getAsSingleObject(serviceData, AuctionDataType.class);

            print(auctionData);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    public static void main(String[] args) {
        try {
            new AuctionClient(args.length > 0 ? args[0] : "foobar");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```