

# Service Data and Notifications

Steffen Pingel

Fakultät für Elektrotechnik und Informatik  
Universität Stuttgart

Hauptseminar Grid Computing  
16.12.2003

Service Data

Notifications

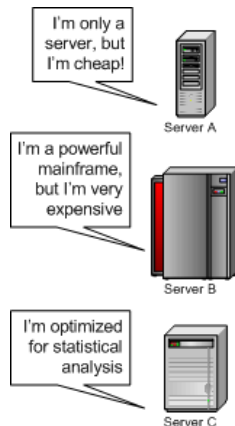
Fazit

# Service Data

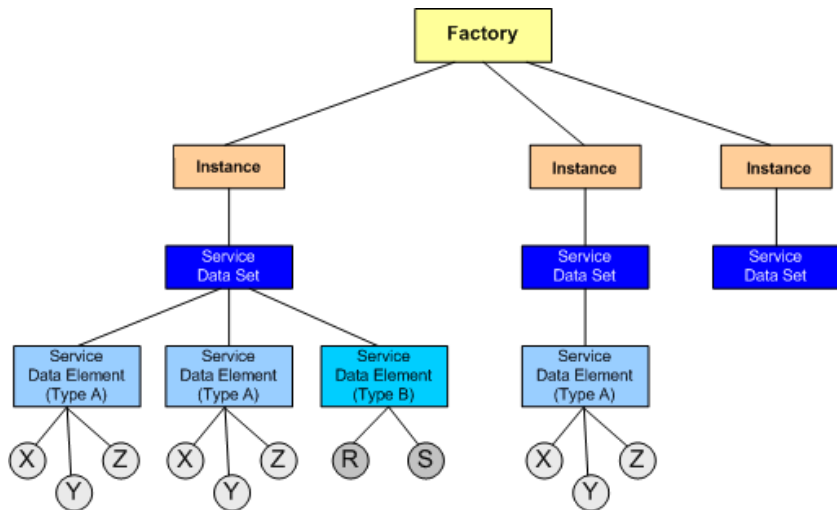
# Definition von Service Data

*Every Grid service can expose internal state as serviceData elements.*

- Notwendig für stateful Services
- “Bewirbt” einen Grid Service
- Auswahl z.B. über einen Index Service
- Implementiert als Java Beans
- Jede Instanz hat ein Service Data Set
- Service Data Sets enthalten Service Data Elements



# Organisation von Service Data



# Service Data Element

## Definition des SDE: AuctionData.xsd

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

  <complexType name="AuctionDataType">
    <sequence>
      <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>
```

# Service Data Element

## Das generierte Java Bean: AuctionDataType.java

```
public class AuctionDataType implements java.io.Serializable {
    private long endDate;
    private java.lang.String location;
    private java.lang.String product;
    private int shipping;

    public AuctionDataType() {
    }

    public long getEndDate() {
        return endDate;
    }

    public void setEndDate(long endDate) {
        this.endDate = endDate;
    }
}
```

...

# Service Interface

## GWSDL Schnittstellen Beschreibung: Auction.gwsdl (1)

```
<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>
```

# Service Interface

## 2. GWSDL Schnittstellen Beschreibung: Auction.gwsdl (2)

```
<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>
```

# Grid Service Implementierung

## AuctionService.java (1)

```
public class AuctionService extends GridServiceImpl
    implements AuctionPortType {

    private String currentBuyer;
    private int currentPrice = 100;

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

    public synchronized boolean bid(String nick, int value)
        throws RemoteException {

        if (value > currentPrice) {
            currentPrice = value;
            currentBuyer = nick;
            return true;
        }
        return false;
    }
}
```

# Grid Service Implementierung

## AuctionService.java (2)

```
public void postCreate(GridContext context)
    throws GridServiceException {

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

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

    // Set intial values of AuctionServiceData
    auctionData.setProduct("XNap Gold Edition");
    auctionData.setShipping(410);
    auctionData.setEndDate(System.currentTimeMillis() + THREE_DAYS);
    auctionData.setLocation("Stuttgart");

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

# Deployment

## Ant Ausgabe

```
Buildfile: build.xml
copyFiles:
mergeMapping:
mappingAvailable:
mergePackageMapping:
setenv:
generateWSDLfromJava:
generateWSDLfromGWSDL:
WSDLUptodate:
GWSDL2WSDL: [echo] Generating wsd1 for
Auction.gwsdl
BindingUptodate:
generateBinding: [echo] Generating
schema for Auction
stubs:
generateStubs: [echo] Generating stubs
compileStubs: [javac] Compiling 29
source files
compile: [javac] Compiling 1 source file

stubjar: [jar] Building jar
jar: [jar] Building jar
gar:
makeGar:
testJars:
copyJars: [copy] Copying 2 files
testSchema:
copySchema: [copy] Copying 6 files
testEtc:
copyEtc:
testDocs:
copyDocs:
testBin:
copyBin:
setGarID:
testPostDeployAvailable:
copyPostDeploy: [copy] Copying 1 file
[jar] Building jar [delete] Deleting
directory
all:
BUILD SUCCESSFUL Total time: 40 seconds
```

# Zusammenfassung der Schritte

Erstellung eines Grid Services mit Service Data

1. Definition des Service Data Elements
2. Definition der Grid Service Schnittstelle
3. Implementierung des Grid Services
4. Deployment

# Finden von Service Data

findServiceData (req., OGSi)

- queryByServiceDataNames (req., OGSi)
  - Übertragung des ganzen Service Data Elements
- queryByXPath (GT3)

## Beispiel

Anfrage:

```
Host[Name=@marvin.informatik.uni-stuttgart.de]/CPULoad
```

Ergebnis:

```
<CPULoad Last1Min="4.8" Last5Min="0.1" Last15Min="00">
```

# Client Implementierung

## AuctionClient.java (1)

```
URL GSH = new java.net.URL
    ("http://127.0.0.1:8080/ogsa/services/xnap/AuctionService/xnap");

// Get a reference to the Math 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);
```

# Client Implementierung

## AuctionClient.java (2)

```
// Print Service Data
System.out.println("Shipping: " + auctionData.getShipping());
System.out.println("Location: " + auctionData.getLocation());

// Start the bidding
BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
while (true) {
    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.");
    }
}
```

# Standard Service Data

Der obligatorische GridService PortType stellt Standard SDEs zur Verfügung:

- interfaces
- serviceName
- factoryLocator
- gridServiceHandle
- gridServiceReference
- findServiceDataExtensibility
- setServiceDataExtensibility
- terminationTime

# Demonstration

▶ Skip Demonstration

[Back](#)
[Forward](#)
[New Window](#)
[Close](#)
[Refresh](#)
 Show dynamic gui

http://127.0.0.1:8080/ogsa/services/xnap/AuctionService/XNapGoldEdition

**Services** | **WSDL** | **Service Data**

Name	PortType	Type	Min	Max	Mutability	Modifia...	Nillable
factoryLocator	{http://...}	{http://...}	1	1	mutable	false	true
gridServiceHandle	{http://...}	{http://...}	0	unbound...	extend...	false	false
gridServiceReference	{http://...}	{http://...}	1	unbound...	mutable	false	false
findServiceDataExtensibility	{http://...}	{http://...}	1	unbound...	static	false	false
setServiceDataExtensibility	{http://...}	{http://...}	1	unbound...	static	false	false
terminationTime	{http://...}	{http://...}	1	1	mutable	false	false
<b>AuctionData</b>	{http://...}	{http://...}	1	1	mutable	false	false

---

**Values**

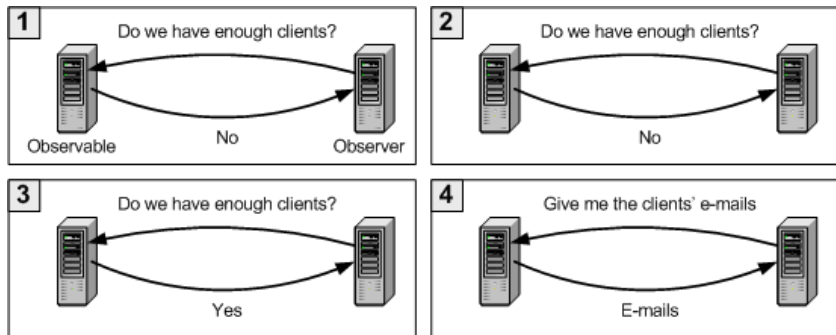
```

ns4:AuctionData xmlns:ns3="http://xnap.org/namespaces/0.1/AuctionData" xmlns:ns4="
  ns3:endDate
  ns3:location xsi:nil="true"
  ns3:product
    XNap Gold Edition
  ns3:shipping
    410
  
```

# Notifications

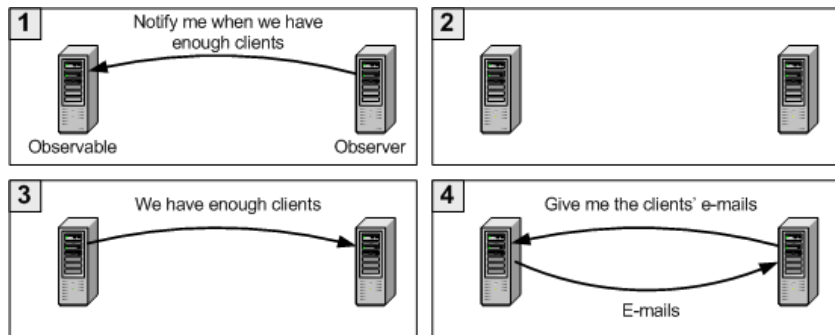
# Polling Approach

Der Client fragt in Intervallen so lange nach, bis der gewünschte Zustand erreicht ist.



# Notification Approach

Der Server schickt eine Nachricht an den Client, sobald sich der Zustand ändert.



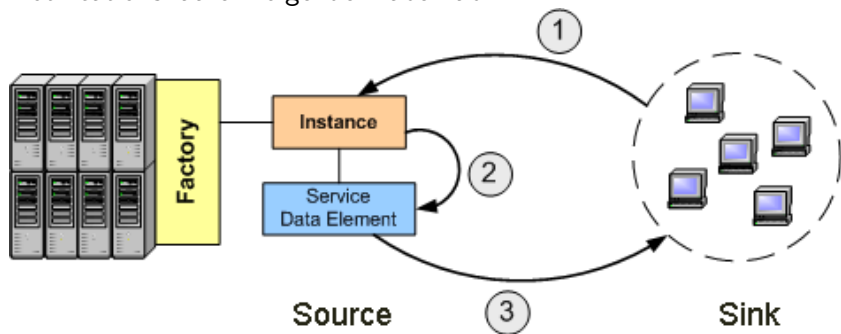
# Einführung

*The purpose of notification is to deliver interesting messages from a notification source to a notification sink.*

- Notifications sind ein populäres Entwurfsmuster
  - Observer/Observable
  - Model-View-Controller
- Client Anwendungen “beobachten” Service Data Elemente
- Im GT3 Jargon
  - Observer → Notification Sink
  - Observables → Notification Source

# Einführung

Notifications laufen folgendermaßen ab:



1. **addListener** Die Sink hängt sich an ein SDE
2. **notifyChange** Die Instanz beauftragt sein SDE die Sinks zu benachrichtigen
3. **deliverNotification** Die Source (das SDE) benachrichtigt alle Sinks

# Service Data Element

## Erweiterung des SDE: AuctionData.xsd

```
<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>
```

# Service Interface

## Erweiterung der Schnittstelle: Auction.gwsdl

```
<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>
```

# Grid Service Implementierung

## Erweiterung des Services: AuctionProvider.java (1)

```
public class AuctionProvider
    implements OperationProvider, GridServiceCallback {

    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;
    }
}
```

# Grid Service Implementierung

## Erweiterung des Services: AuctionProvider.java (2)

```
public void postCreate(GridContext context)
    throws GridServiceException {
    // Create Service Data Element
    [...]
    // Set initial values of AuctionServiceData
    [...]
    // Add SDE to Service Data Set
    [...]
}

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;
}
```

# Client Implementierung

## Der Client wird zum Server: AuctionClient.java (2)

```
public class AuctionClient 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/");

        // Start listening to the MathService
        NotificationSinkManager nm = NotificationSinkManager.getManager();
        nm.startListening(NotificationSinkManager.MAIN_THREAD);
        String sh = nm.addListener("AuctionData", null, GSH, this);

        // Start the bidding
        [...]

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

# Client Implementierung

## Der Client wird zum Server: AuctionClient.java (1)

```
public void deliverNotification(ExtensibilityType any)
    throws RemoteException {

    try {
        // Service Data has changed
        ServiceDataValuesType serviceData
            = AnyHelper.getAsServiceDataValues(any);
        AuctionDataType data
            = (AuctionDataType)AnyHelper.getAsSingleObject
                (serviceData, AuctionDataType.class);

        // Show current price and current buyer
        print(data);
    }
    catch(Exception e) {
        System.out.println("ERROR!");
        e.printStackTrace();
    }
}
```

# Demonstration

▶ Skip Demonstration

```
Shell - Konsole <4>
Session Edit View Bookmarks Settings Help

Current buyer: null
Current price: 100
-----
Your bid:
120
> Your bid is currently the highest!
Your bid:
-----
Current buyer: foo
Current price: 120
-----
Current buyer: bar
Current price: 140
-----
█

New Shell
```

```
Shell - Konsole <5>
Session Edit View Bookmarks Settings Help

Your nick: bar
Shipping: 410
Location: Stuttgart
-----
Current buyer: foo
Current price: 120
-----
Your bid:
140
> Your bid is currently the highest!
Your bid:
-----
Current buyer: bar
Current price: 140
-----
█

New Shell
```

# Fazit

# Zusammenfassung

## Service Data

- Macht Informationen über stateful Services zugänglich
- Nutzbar über die findServiceData Schnittstelle oder indirekt z.B. über Index Services
- Implementierung als Java Bean
- Ein Service Data Set pro Grid Service Instanz
- Mehrere Service Data Element pro Service Data Set

## Notifications

- Benachrichtigung über Änderungen von Service Data Elements
- Implementiert ein gängiges Observer Pattern

Vielen Dank für Ihre  
Aufmerksamkeit!