ASAP & Wf-XML 2.0 Demo Scenario June 2004

Overview

There are two levels of involvement in the demonstration and hence two demonstration scenarios:

1) Demonstration of compliance to ASAP
2) Demonstration of compliance to Wf-XML

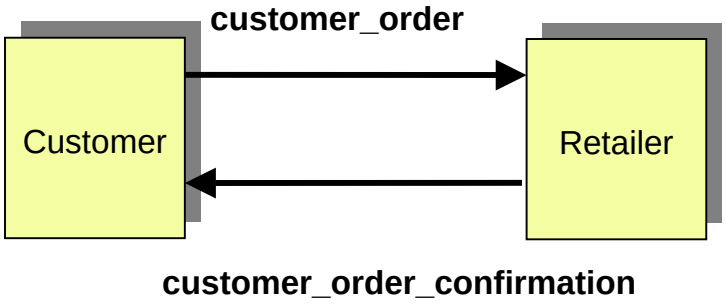
Since ASAP and Wf-XML are Internet based protocols there is no need for the participating organizations to actually bring their systems to the demonstrations. In both cases the demonstrations will be initiated from a test harness at BrainStorm event and will communicate with the participating BPM systems over the Internet.

To the outside world both scenarios will appear almost the same. However, for test of compliance to ASAP only one BPM engine will be involved whilst for the Wf-XML scenario a number of BPM engines can participate. In common with previous interoperability demonstrations a supply chain scenario is to be used.

ASAP Scenario

The ASAP scenario simulates a customer purchasing one or more of the same product from a retailer.

A customer provides personal details (name and address etc.) plus details of the product to be ordered. The test harness will transmit the details to the target BPM system which will respond with an order confirmation.



Exact details of the data to be exchanged is contained within the accompanying .XSD file but a summary is provided below:

The customer will supply the following information (**customer_order**):

Customer Details
Name, Address, Phone Number
Order Details
Product Code, Product Description, Quantity Required
Order Date

```
<xs:complexType name="customer_order">
    <xs:sequence>
        <xs:element name="customer_first_name"  type="xs:string"/>
        <xs:element name="customer_surname"     type="xs:string"/>
        <xs:element name="address_first_line"   type="xs:string"/>
        <xs:element name="address_second_line"  type="xs:string"/>
        <xs:element name="address_city"         type="xs:string"/>
        <xs:element name="address_zipcode"      type="xs:string"/>
        <xs:element name="phone_number"         type="xs:string"/>
        <xs:element name="product_code"         type="xs:int"/>
        <xs:element name="product_description"  type="xs:string"/>
        <xs:element name="product_quantity"     type="xs:int"/>
        <xs:element name="order_date"           type="xs:date"/>
    </xs:sequence>
</xs:complexType>
```

It should be possible for any Name, Address and phone number to be input as they will not be validated.

For the purpose of the demonstration any numeric value can be used for a product code, any textual description can be used for a product code and any numeric value used for the quantity of product ordered.

Normally a system would automatically generate the order date. However, to allow the demonstrator to influence the outcome of the order this is to be input manually during the demonstration.

The target BPM product that has implemented the Retailer process will perform the necessary actions to supply the ordered products to the customer and will respond with an order confirmation.

The order confirmation will consist of the following information (**customer_order_confirmation**):

Customer Details
    Name, Address, Phone Number
Order Date
Retailer Name
Order Number
Estimated Delivery Date
Order Details
    Product Code, Product Description, Quantity Required, Price per Unit, Total Order Price
Product was in stock or had to be manufactured to order
Manufacturer details if product was manufactured to order
    Name of Manufacturer, Manufacturer Code, Estimated Manufacture Date

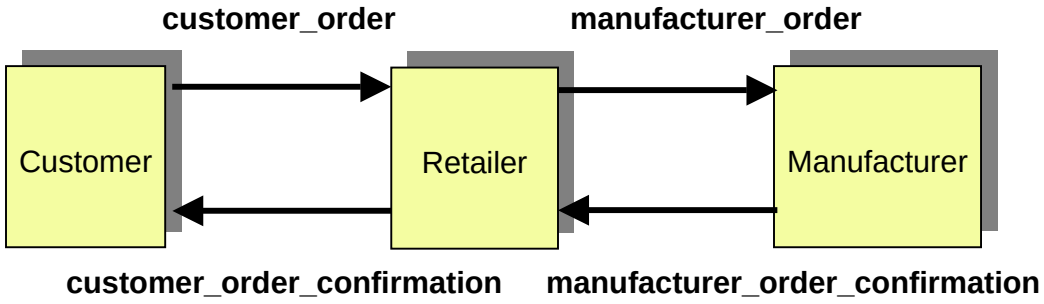
```
<xs:complexType name="customer_order_confirmation">
    <xs:sequence>
        <xs:element name="customer_first_name"      type="xs:string"/>
        <xs:element name="customer_surname"         type="xs:string"/>
        <xs:element name="address_first_line"       type="xs:string"/>
        <xs:element name="address_second_line"      type="xs:string"/>
        <xs:element name="address_city"             type="xs:string"/>
        <xs:element name="address_zipcode"          type="xs:string"/>
        <xs:element name="phone_number"             type="xs:string"/>
        <xs:element name="order_date"               type="xs:date"/>
        <xs:element name="retailer_name"            type="xs:string"/>
        <xs:element name="order_number"             type="xs:int"/>
        <xs:element name="est_delivery_date"        type="xs:date"/>
        <xs:element name="product_code"             type="xs:int"/>
        <xs:element name="product_description"      type="xs:string"/>
        <xs:element name="product_quantity"         type="xs:int"/>
        <xs:element name="price_per_unit"           type="xs:float"/>
        <xs:element name="order_price"              type="xs:float"/>
        <xs:element name="in_stock"                 type="xs:boolean"/>
        <xs:element name="manufactured_by"          type="xs:string"/>
        <xs:element name="manufacturer_code"        type="xs:int"/>
        <xs:element name="manufacturer_receipt_date" type="xs:date"/>
    </xs:sequence>
</xs:complexType>
```

Although any numeric code can be used for the Product Code an odd number will cause the Retailer to behave as if the product ordered was not in stock and had to be manufactured to order.

Each BPM system can decide its own strategy for generating order numbers, price per unit, manufacturer code as well as manufacturing and delivery dates. Similarly, each BPM system can assign its own Retailer and Manufacturer names but should ideally include the name of the BPM product or vendor name.

Wf-XML 2.0 Scenario

As described above the scenario for the Wf-XML 2.0 will be very similar to that used by the ASAP scenario. The customer will make an order and will receive an order confirmation. However, rather than the retailer simulating the manufacture of the products the BPM system that is acting as the Retailer will initiate an instance of a manufacturing process on another BPM system.



The same logic as above (odd Product Code) will determine whether the product is to be manufactured to order or is in stock at the retailer. If the product does need to be "manufactured" to order the retailer will supply the following information (**manufacturer_order**) with the process instantiation:

Order Date
Product Quantity
Product Code
Order Number
Retailer Name

```
<xs:complexType name="manufacturer_order">
    <xs:sequence>
        <xs:element name="product_code"            type="xs:int"/>
        <xs:element name="product_quantity"        type="xs:int"/>
        <xs:element name="order_date"              type="xs:date"/>
        <xs:element name="order_number"            type="xs:string"/>
        <xs:element name="retailer_name"           type="xs:string"/>
    </xs:sequence>
</xs:complexType>
```

The BPM system that has implemented the manufacturing process will "manufacture" the product and respond on completion to the retailer with an order confirmation (**manufacturer_order_confirmation**) which contains the following information:

Availability Date
Manufacturer Name
Manufacturer Code

```
<xs:complexType name="manufacturer_order_confirmation">
    <xs:sequence>
        <xs:element name="availability_date"     type="xs:date"/>
        <xs:element name="manufacturer_name"     type="xs:string"/>
        <xs:element name="manufacturer_code"     type="xs:int"/>
    </xs:sequence>
</xs:complexType>
```

Upon receipt of the order confirmation from the manufacturer, the retailer will respond to the customer with an order confirmation (customer_order_confirmation).

As with the simpler ASAP scenario each BPM system implementing the retailer and manufacturer processes can decide its own strategy for generating order numbers, price per unit, manufacturer code as well as manufacturing and delivery dates. Similarly, each BPM system can assign its own Retailer and Manufacturer names but again should ideally include the name of the BPM product or vendor name.

Ideally each participating BPM engine should implement both the retailer and manufacturer processes. This will not only assist each BPM vendor test its implementation but also provide more demonstration permutations. Each vendor implementing the retailer should design their retailer process such that more than one manufacturer can be used to “manufacture” the products.

Sample Messages

In order to clarify the exact message structure, below are a set of sample messages which might be exchanged between two conforming applications.

The first message is from the client to the customer_order_factory asking for details on the context data structure. The request is a fixed message and would look like this

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soap:Header>
  <as:Request xmlns:as="http://www.oasis-open.org/asap/0.9/asap.xsd">
   <as:SenderKey>http://harness/customer</as:SenderKey>
   <as:ReceiverKey>http://SAMEERP:9004/iflowjsp/jsp/ProcDef.jsp?planName=
    Retailer
   </as:ReceiverKey>
   <as:ResponseRequired>Yes</as:ResponseRequired>
  </as:Request>
 </soap:Header>
 <soap:Body>
  <as:GetPropertiesRq xmlns:as="http://www.oasis-open.org/asap/0.9/asap.xsd"/>
 </soap:Body>
</soap:Envelope>
```

The response from the customer_order_factory would include the schema of the context and response data. It would look like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:as="http://www.oasis-open.org/asap/0.9/asap.xsd"
 xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
 <env:Header>
  <as:Response>
   <as:SenderKey>http://SAMEERP:9004/iflowjsp/jsp/ProcDef.jsp?planName=
    Retailer</as:SenderKey>
   <as:ReceiverKey>http://harness/customer</as:ReceiverKey>
  </as:Response>
 </env:Header>
 <env:Body>
  <as:GetPropertiesRs>
   <as:Key>http://SAMEERP:9004/iflowjsp/jsp/ProcDef.jsp?planName=Retailer
   </as:Key>
   <as:Name>Retailer</as:Name>
   <as:Subject>Retailer process</as:Subject>
   <as:Description>Business process to accept a customer order and fulfill it,
    placing a manufacturer order if necessary.
   </as:Description>
   <as:ContextDataSchema>
    <xsd:schema
     targetNamespace="http:SAMEERP:9004/iflowjsp/jsp/cds.jsp?planName=Retailer"
     xmlns:pd="http:SAMEERP:9004/iflowjsp/jsp/cds.jsp?planName=Retailer"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema"
     elementFormDefault="qualified">
     <xsd:complexType name="ContextDataType">
      <xsd:sequence>
       <xsd:element name="customer_first_name" type="xsd:string" minOccurs="0"/>
       <xsd:element name="customer_surname" type="xsd:string" minOccurs="0"/>
       <xsd:element name="address_first_line" type="xsd:string" minOccurs="0"/>
       <xsd:element name="address_second_line" type="xsd:string" minOccurs="0"/>
       <xsd:element name="address_city" type="xsd:string" minOccurs="0"/>
       <xsd:element name="address_zipcode" type="xsd:string" minOccurs="0"/>
       <xsd:element name="phone_number" type="xsd:string" minOccurs="0"/>
       <xsd:element name="product_code" type="xsd:int" minOccurs="0"/>
       <xsd:element name="product_description" type="xsd:string" minOccurs="0"/>
       <xsd:element name="product_quantity" type="xsd:int" minOccurs="0"/>
       <xsd:element name="order_date" type="xsd:date" minOccurs="0"/>
        minOccurs="0"/>
      </xsd:sequence>
     </xsd:complexType>
    </xsd:schema>
   </as:ContextDataSchema>
```

```
   <as:ResultDataSchema>
    <xsd:schema
     targetNamespace="http:SAMEERP:9004/iflowjsp/jsp/rds.jsp?planName=Retailer"
     xmlns:pd="http:SAMEERP:9004/iflowjsp/jsp/rds.jsp?planName=Retailer"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema"
     elementFormDefault="qualified">
     <xsd:complexType name="ResultDataType">
      <xsd:sequence>
       <xsd:element name="customer_first_name" type="xsd:string" minOccurs="0"/>
       <xsd:element name="customer_surname" type="xsd:string" minOccurs="0"/>
       <xsd:element name="address_first_line" type="xsd:string" minOccurs="0"/>
       <xsd:element name="address_second_line" type="xsd:string" minOccurs="0"/>
       <xsd:element name="address_city" type="xsd:string" minOccurs="0"/>
       <xsd:element name="address_zipcode" type="xsd:string" minOccurs="0"/>
       <xsd:element name="phone_number" type="xsd:string" minOccurs="0"/>
       <xsd:element name="order_date" type="xsd:date" minOccurs="0"/>
       <xsd:element name="retailer_name" type="xsd:string" minOccurs="0"/>
       <xsd:element name="order_number" type="xsd:int" minOccurs="0"/>
       <xsd:element name="est_delivery_date" type="xsd:date" minOccurs="0"/>
       <xsd:element name="product_code" type="xsd:int" minOccurs="0"/>
       <xsd:element name="product_description" type="xsd:string" minOccurs="0"/>
       <xsd:element name="product_quantity" type="xsd:int" minOccurs="0"/>
       <xsd:element name="price_per_unit" type="xsd:float" minOccurs="0"/>
       <xsd:element name="order_price" type="xsd:float" minOccurs="0"/>
       <xsd:element name="in_stock" type="xsd:boolean" minOccurs="0"/>
       <xsd:element name="manufactured_by" type="xsd:string" minOccurs="0"/>
       <xsd:element name="manufacturer_code" type="xsd:int" minOccurs="0"/>
       <xsd:element name="manufacturer_receipt_date" type="xsd:date"
        minOccurs="0"/>
      </xsd:sequence>
     </xsd:complexType>
    </xsd:schema>
   </as:ResultDataSchema>
   <as:Expiration>P120D</as:Expiration>
  </as:GetPropertiesRs>
 </env:Body>
</env:Envelope>
```

This gives the client the information that it needs in order to validate what it is sending to the factory in order to start the service instance. The message to start the process instance might look like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soap:Header>
  <as:Request xmlns:as="http://www.oasis-open.org/asap/0.9/asap.xsd">
   <as:SenderKey>http://harness/customer</as:SenderKey>
   <as:ReceiverKey>http://SAMEERP:9004/iflowjsp/jsp/ProcDef.jsp?planName=
    Retailer
   </as:ReceiverKey>
   <as:ResponseRequired>Yes</as:ResponseRequired>
  </as:Request>
 </soap:Header>
 <soap:Body>
  <as:CreateInstanceRq xmlns:as="http://www.oasis-open.org/asap/0.9/asap.xsd">
   <as:StartImmediately>true</as:StartImmediately>
   <as:ObserverKey>http://harness/customer.observer</as:ObserverKey>
   <as:Name>TestRetailer</as:Name>
   <as:Subject>Test retailer process</as:Subject>
   <as:Description>Test retailer process with proper context data</as:Description>
   <as:ContextData
    xmlns:pd="http:SAMEERP:9004/iflowjsp/jsp/cds.jsp?planName=Retailer"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <pd:customer_first_name>Bill</pd:customer_first_name>
    <pd:customer_surname>Smith</pd:customer_surname>
    <pd:address_first_line>285 Garnet Grove</pd:address_first_line>
    <pd:address_second_line>Suite 48</pd:address_second_line>
    <pd:address_city>San Jose</pd:address_city>
    <pd:address_zipcode>95134</pd:address_zipcode>
    <pd:phone_number>(408)-555-2476</pd:phone_number>
    <pd:product_code>245</pd:product_code>
    <pd:product_description>Lawn mower</pd:product_description>
    <pd:product_quantity>1</pd:product_quantity>
    <pd:order_date>2004-06-01</pd:order_date>
   </as:ContextData>
  </as:CreateInstanceRq>
 </soap:Body>
</soap:Envelope>
```

The confirmation message response back to the creation would be:

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:as="http://www.oasis-open.org/asap/0.9/asap.xsd"
 xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
 <env:Header>
  <as:Response>
   <as:SenderKey>http://SAMEERP:9004/iflowjsp/jsp/ProcDef.jsp?planName=Retailer
   </as:SenderKey>
   <as:ReceiverKey>http://harness/customer</as:ReceiverKey>
  </as:Response>
 </env:Header>
 <env:Body>
  <as:CreateInstanceRs>
   <as:InstanceKey>http://SAMEERP:9004/iflowjsp/jsp/ProcInst.jsp?pid=6071
   </as:InstanceKey>
  </as:CreateInstanceRs>
 </env:Body>
</env:Envelope>
```

After a certain amount of time, the completion message will come back to the customer:

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soap:Header>
  <as:Request xmlns:as="http://www.oasis-open.org/asap/0.9/asap.xsd">
   <as:SenderKey>http://SAMEERP:9004/iflowjsp/jsp/ProcInst.jsp?pid=6071
   </as:SenderKey>
   <as:ReceiverKey>http://harness/customer.observer</as:ReceiverKey>
   <as:ResponseRequired>Yes</as:ResponseRequired>
  </as:Request>
 </soap:Header>
 <soap:Body>
  <as:CompletedRq xmlns:as="http://www.oasis-open.org/asap/0.9/asap.xsd">
   <as:InstanceKey>http://SAMEERP:9004/iflowjsp/jsp/ProcInst.jsp?pid=6071
   </as:InstanceKey>
   <as:ResultData
    xmlns:pd="http:SAMEERP:9004/iflowjsp/jsp/rds.jsp?planName=Retailer"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <pd:customer_first_name>Bill</pd:customer_first_name>
    <pd:customer_surname>Smith</pd:customer_surname>
    <pd:address_first_line>285 Garnet Grove</pd:address_first_line>
    <pd:address_second_line>Suite 48</pd:address_second_line>
    <pd:address_city>San Jose</pd:address_city>
    <pd:address_zipcode>95134</pd:address_zipcode>
    <pd:phone_number>(408)-555-2476</pd:phone_number>
    <pd:order_date>2004-06-01</pd:order_date>
    <pd:retailer_name>ACME Systems</pd:retailer_name>
    <pd:order_number>87404584</pd:order_number>
    <pd:est_delivery_date>2004-06-15</pd:est_delivery_date>
    <pd:product_code>245</pd:product_code>
    <pd:product_description>Lawn mower</pd:product_description>
    <pd:product_quantity>1</pd:product_quantity>
    <pd:price_per_unit>59.99</pd:price_per_unit>
    <pd:order_price>59.99</pd:order_price>
    <pd:in_stock>false</pd:in_stock>
    <pd:manufactured_by>ALPHA Manufacturing</pd:manufactured_by>
    <pd:manufacturer_code>2734</pd:manufacturer_code>
    <pd:manufacturer_receipt_date>2004-06-10</pd:manufacturer_receipt_date>
   </as:ResultData>
  </as:CompletedRq>
 </soap:Body>
</soap:Envelope>
```

And the response to this completion notification would be:

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:as="http://www.oasis-open.org/asap/0.9/asap.xsd"
 xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
 <env:Header>
  <as:Response>
   <as:SenderKey>http://harness/customer.observer</as:SenderKey>
   <as:ReceiverKey>http://SAMEERP:9004/iflowjsp/jsp/ProcInst.jsp?pid=6071
   </as:ReceiverKey>
  </as:Response>
 </env:Header>
 <env:Body>
  <as:CompletedRs/>
 </env:Body>
</env:Envelope>
```

The customer may choose to retrieve the ResultData from the Retailer service instance by sending it a GetProperties message:

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soap:Header>
  <as:Request xmlns:as="http://www.oasis-open.org/asap/0.9/asap.xsd">
   <as:SenderKey>http://harness/customer.observer</as:SenderKey>
   <as:ReceiverKey>http://SAMEERP:9004/iflowjsp/jsp/ProcInst.jsp?pid=6071
   </as:ReceiverKey>
   <as:ResponseRequired>Yes</as:ResponseRequired>
  </as:Request>
 </soap:Header>
 <soap:Body>
  <as:GetPropertiesRq xmlns:as="http://www.oasis-open.org/asap/0.9/asap.xsd"/>
 </soap:Body>
</soap:Envelope>
```

The response from the Retailer service instance:

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:as="http://www.oasis-open.org/asap/0.9/asap.xsd"
 xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
 <env:Header>
  <as:Response>
   <as:SenderKey>http://SAMEERP:9004/iflowjsp/jsp/ProcInst.jsp?pid=6071
   </as:SenderKey>
   <as:ReceiverKey>http://harness/customer.observer</as:ReceiverKey>
  </as:Response>
 </env:Header>
 <env:Body>
  <as:GetPropertiesRs>
   <as:Key>http://SAMEERP:9004/iflowjsp/jsp/ProcInst.jsp?pid=6071</as:Key>
   <as:State>closed.completed</as:State>
   <as:Name>TestRetailer</as:Name>
   <as:Subject>Test retailer process</as:Subject>
   <as:Description>Test retailer process with proper context data</as:Description>
   <as:FactoryKey>http://SAMEERP:9004/iflowjsp/jsp/ProcDef.jsp?planName=
    Retailer
   </as:FactoryKey>
   <as:Observers>
    <as:ObserverKey>http://harness/customer.observer</as:ObserverKey>
   </as:Observers>
   <as:ContextData
    xmlns:pd="http:SAMEERP:9004/iflowjsp/jsp/cds.jsp?planName=Retailer"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <pd:customer_first_name>Bill</pd:customer_first_name>
    <pd:customer_surname>Smith</pd:customer_surname>
    <pd:address_first_line>285 Garnet Grove</pd:address_first_line>
    <pd:address_second_line>Suite 48</pd:address_second_line>
    <pd:address_city>San Jose</pd:address_city>
    <pd:address_zipcode>95134</pd:address_zipcode>
    <pd:phone_number>(408)-555-2476</pd:phone_number>
    <pd:product_code>245</pd:product_code>
    <pd:product_description>Lawn mower</pd:product_description>
    <pd:product_quantity>1</pd:product_quantity>
    <pd:order_date>2004-06-01</pd:order_date>
   </as:ContextData>
```

```
   <as:ResultData
    xmlns:pd="http:SAMEERP:9004/iflowjsp/jsp/rds.jsp?planName=Retailer"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <pd:customer_first_name>Bill</pd:customer_first_name>
    <pd:customer_surname>Smith</pd:customer_surname>
    <pd:address_first_line>285 Garnet Grove</pd:address_first_line>
    <pd:address_second_line>Suite 48</pd:address_second_line>
    <pd:address_city>San Jose</pd:address_city>
    <pd:address_zipcode>95134</pd:address_zipcode>
    <pd:phone_number>(408)-555-2476</pd:phone_number>
    <pd:order_date>2004-06-01</pd:order_date>
    <pd:retailer_name>ACME Systems</pd:retailer_name>
    <pd:order_number>87404584</pd:order_number>
    <pd:est_delivery_date>2004-06-15</pd:est_delivery_date>
    <pd:product_code>245</pd:product_code>
    <pd:product_description>Lawn mower</pd:product_description>
    <pd:product_quantity>1</pd:product_quantity>
    <pd:price_per_unit>59.99</pd:price_per_unit>
    <pd:order_price>59.99</pd:order_price>
    <pd:in_stock>false</pd:in_stock>
    <pd:manufactured_by>ALPHA Manufacturing</pd:manufactured_by>
    <pd:manufacturer_code>2734</pd:manufacturer_code>
    <pd:manufacturer_receipt_date>2004-06-10</pd:manufacturer_receipt_date>
   </as:ResultData>
   <as:History>
    <as:Event>
     <as:Time>2004-06-01T13:20:00.000-08:00</as:Time>
     <as:EventType>InstanceCreated</as:EventType>
     <as:SourceKey>http://SAMEERP:9004/iflowjsp/jsp/ProcDef.jsp?planName=
      Retailer
     </as:SourceKey>
     <as:Details>Instance created</as:Details>
     <as:OldState>open.notrunning</as:OldState>
     <as:NewState>open.running</as:NewState>
    </as:Event>
    <as:Event>
     <as:Time>2004-06-10T16:47:22.576-08:00</as:Time>
     <as:EventType>StateChanged</as:EventType>
     <as:SourceKey>http://SAMEERP:9004/iflowjsp/jsp/ProcInst.jsp?pid=6071
     </as:SourceKey>
     <as:Details>state changed</as:Details>
     <as:OldState>open.running</as:OldState>
     <as:NewState>closed.completed</as:NewState>
    </as:Event>
   </as:History>
  </as:GetPropertiesRs>
 </env:Body>
</env:Envelope>
```

That describes a complete basic interaction necessary for the ASAP demo. Then, the more complete Wf-XML demo includes the above message, but also includes a set of messages to/from the manufacturer.

The message to the manufacturer to create the service instance:

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soap:Header>
  <as:Request xmlns:as="http://www.oasis-open.org/asap/0.9/asap.xsd">
   <as:SenderKey>http://SAMEERP:9004/iflowjsp/jsp/ProcInst.jsp?pid=6071
   </as:SenderKey>
   <as:ReceiverKey>http://SAMEERP:9004/iflowjsp/jsp/ProcDef.jsp?planName=
    Manufacturer
   </as:ReceiverKey>
   <as:ResponseRequired>Yes</as:ResponseRequired>
  </as:Request>
 </soap:Header>
 <soap:Body>
  <as:CreateInstanceRq xmlns:as="http://www.oasis-open.org/asap/0.9/asap.xsd">
   <as:StartImmediately>true</as:StartImmediately>
   <as:ObserverKey>http://SAMEERP:9004/iflowjsp/jsp/ActivityObserver.jsp?pid=6071
    &amp;aid=6077
   </as:ObserverKey>
   <as:Name>TestManufacturer</as:Name>
   <as:Subject>Test manufacturer process</as:Subject>
   <as:Description>Test manufacturer process with proper context
    data</as:Description>
   <as:ContextData
    xmlns:pd="http:SAMEERP:9004/iflowjsp/jsp/cds.jsp?planName=Retailer"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <pd:product_code>245</pd:product_code>
    <pd:product_quantity>1</pd:product_quantity>
    <pd:order_date>2004-06-01</pd:order_date>
    <pd:order_number>87404584</pd:order_number>
    <pd:retailer_name>ACME Systems</pd:retailer_name>
   </as:ContextData>
  </as:CreateInstanceRq>
 </soap:Body>
</soap:Envelope>
```

The response:

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:as="http://www.oasis-open.org/asap/0.9/asap.xsd"
 xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
 <env:Header>
  <as:Response>
   <as:SenderKey>http://SAMEERP:9004/iflowjsp/jsp/ProcDef.jsp?planName=
    Manufacturer
   </as:SenderKey>
   <as:ReceiverKey>http://SAMEERP:9004/iflowjsp/jsp/ProcInst.jsp?pid=6071
   </as:ReceiverKey>
  </as:Response>
 </env:Header>
 <env:Body>
  <as:CreateInstanceRs>
   <as:InstanceKey>http://SAMEERP:9004/iflowjsp/jsp/ProcInst.jsp?pid=6085
   </as:InstanceKey>
  </as:CreateInstanceRs>
 </env:Body>
</env:Envelope>
```

The completed message:

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <soap:Header>
  <as:Request xmlns:as="http://www.oasis-open.org/asap/0.9/asap.xsd">
   <as:SenderKey>http://SAMEERP:9004/iflowjsp/jsp/ProcInst.jsp?pid=6085
   </as:SenderKey>
   <as:ReceiverKey>http://SAMEERP:9004/iflowjsp/jsp/ActivityObserver.jsp?
    pid=6071&amp;aid=6077
   </as:ReceiverKey>
   <as:ResponseRequired>Yes</as:ResponseRequired>
  </as:Request>
 </soap:Header>
 <soap:Body>
  <as:CompletedRq xmlns:as="http://www.oasis-open.org/asap/0.9/asap.xsd">
   <as:InstanceKey>http://SAMEERP:9004/iflowjsp/jsp/ProcInst.jsp?pid=6085
   </as:InstanceKey>
   <as:ResultData
    xmlns:pd="http:SAMEERP:9004/iflowjsp/jsp/rds.jsp?planName=Retailer"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <pd:availability_date>2004-06-10</pd:availability_date>
    <pd:manufacturer_name>ALPHA Manufacturing</pd:manufactured_by>
    <pd:manufacturer_code>2734</pd:manufacturer_code>
   </as:ResultData>
  </as:CompletedRq>
 </soap:Body>
</soap:Envelope>
```

The response to that :

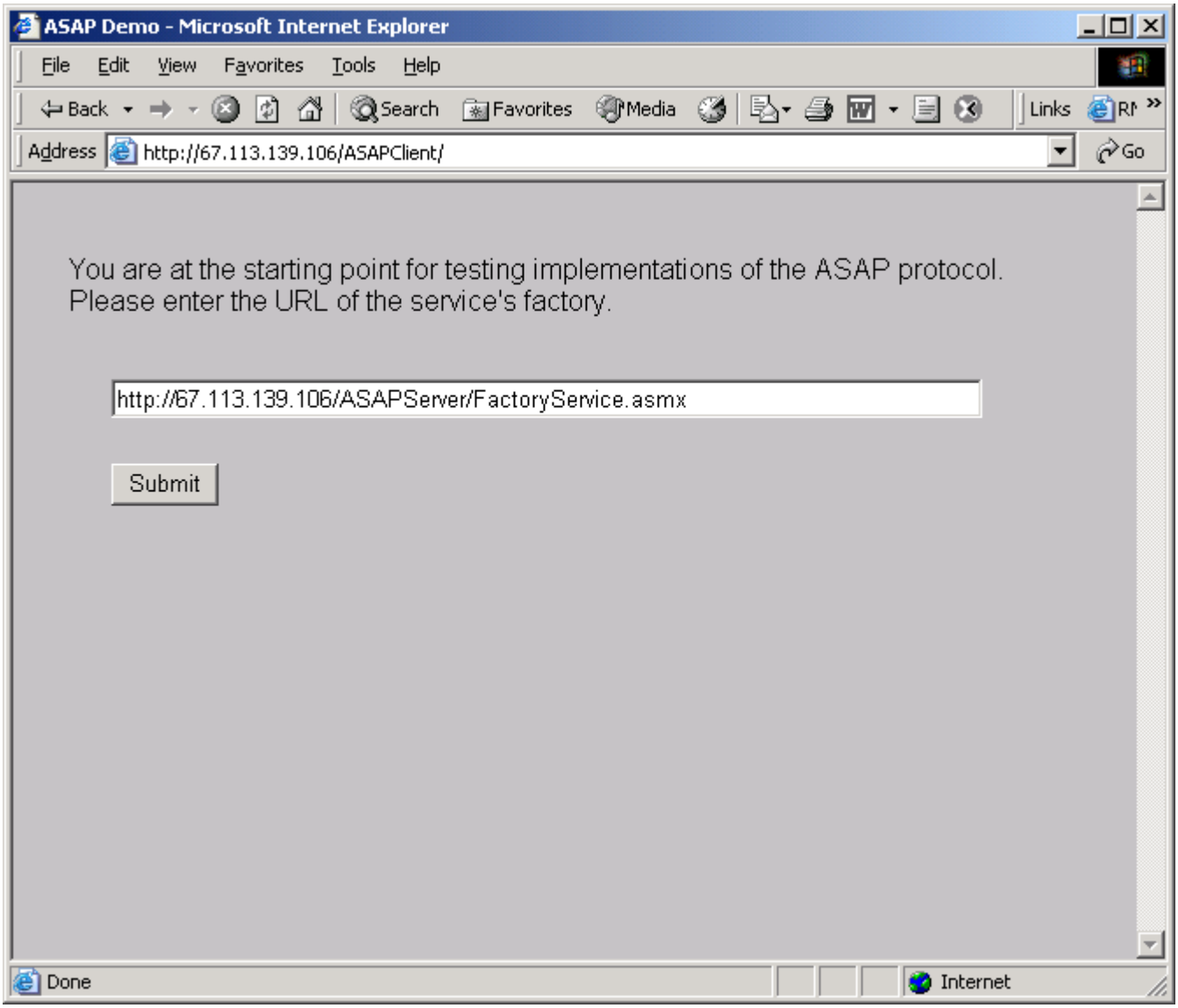
```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:as="http://www.oasis-open.org/asap/0.9/asap.xsd"
 xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
 <env:Header>
  <as:Response>
   <as:SenderKey>http://SAMEERP:9004/iflowjsp/jsp/ActivityObserver.jsp?pid=6071
    &amp;aid=6077
   </as:SenderKey>
   <as:ReceiverKey>http://SAMEERP:9004/iflowjsp/jsp/ProcInst.jsp?pid=6085
   </as:ReceiverKey>
  </as:Response>
 </env:Header>
 <env:Body>
  <as:CompletedRs/>
 </env:Body>
</env:Envelope>
```

How to use the test client

The test client is available at the following URL:

http://67.113.139.106/ASAPClient

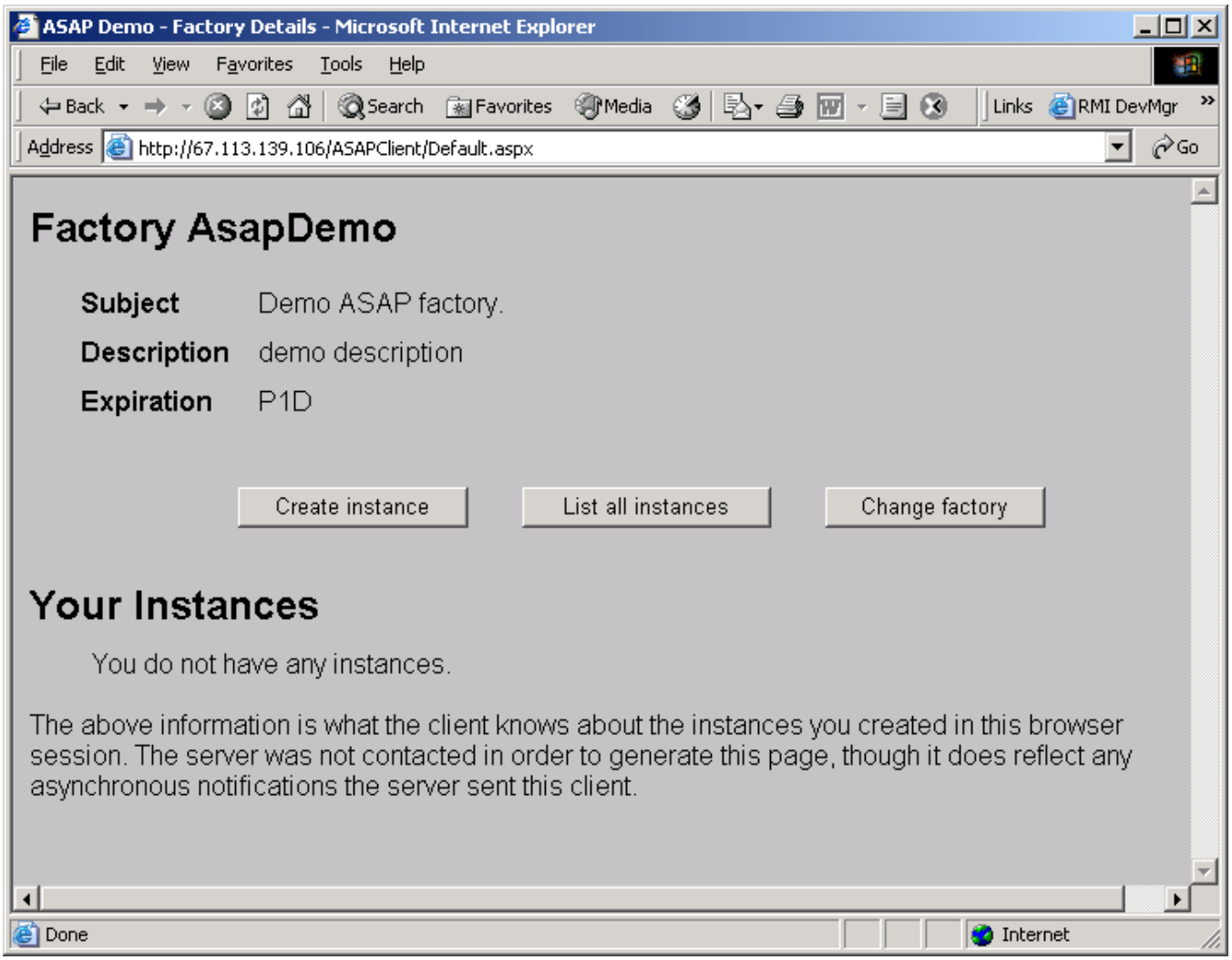
It looks like this:



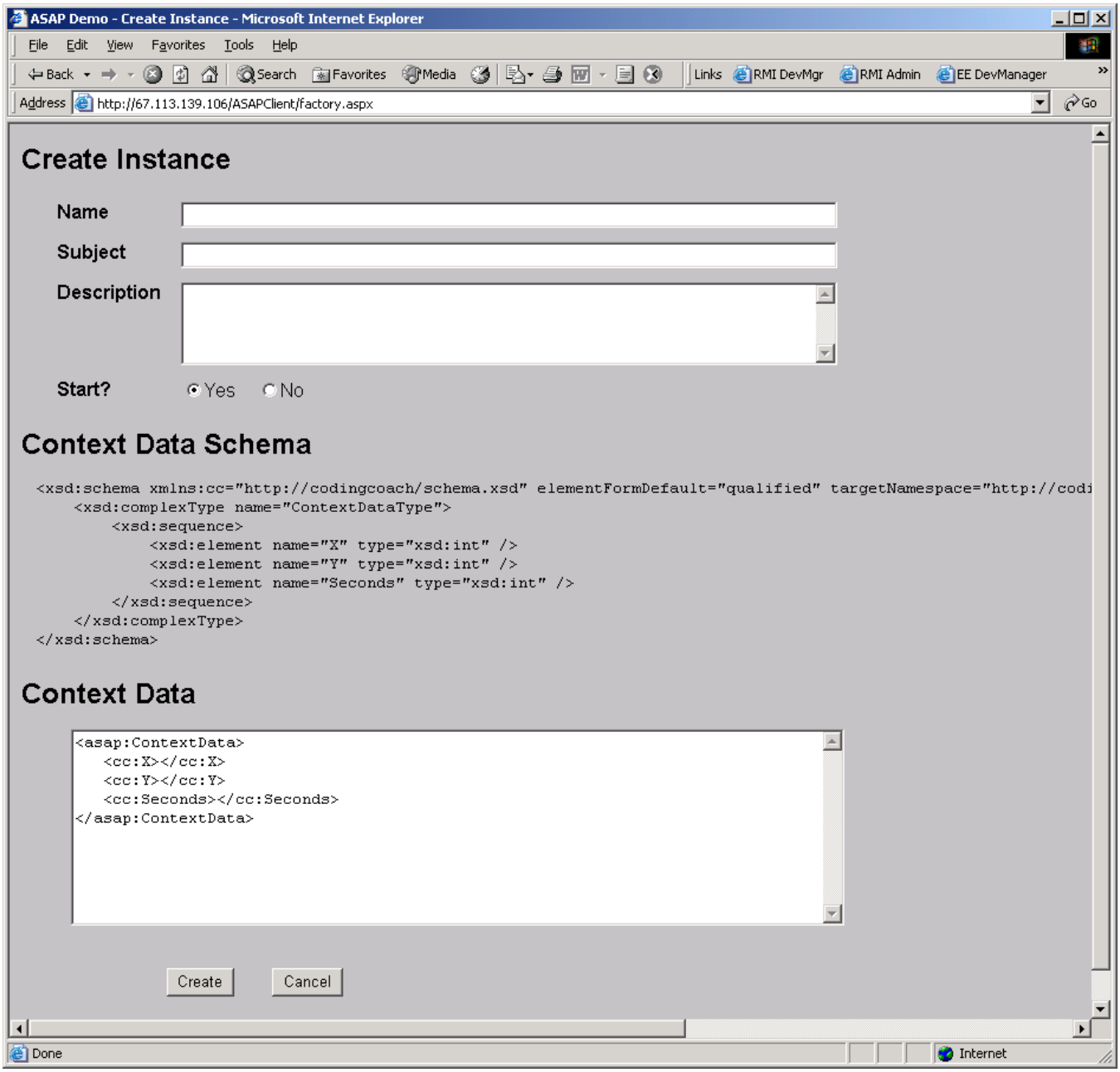
It is setup to work with a demo ASAP server. To use it to access your ASAP Server, replace the Factory URL in the text box with the URL of your factory.

The rest of this section assumes that the test client is working against the demo server.

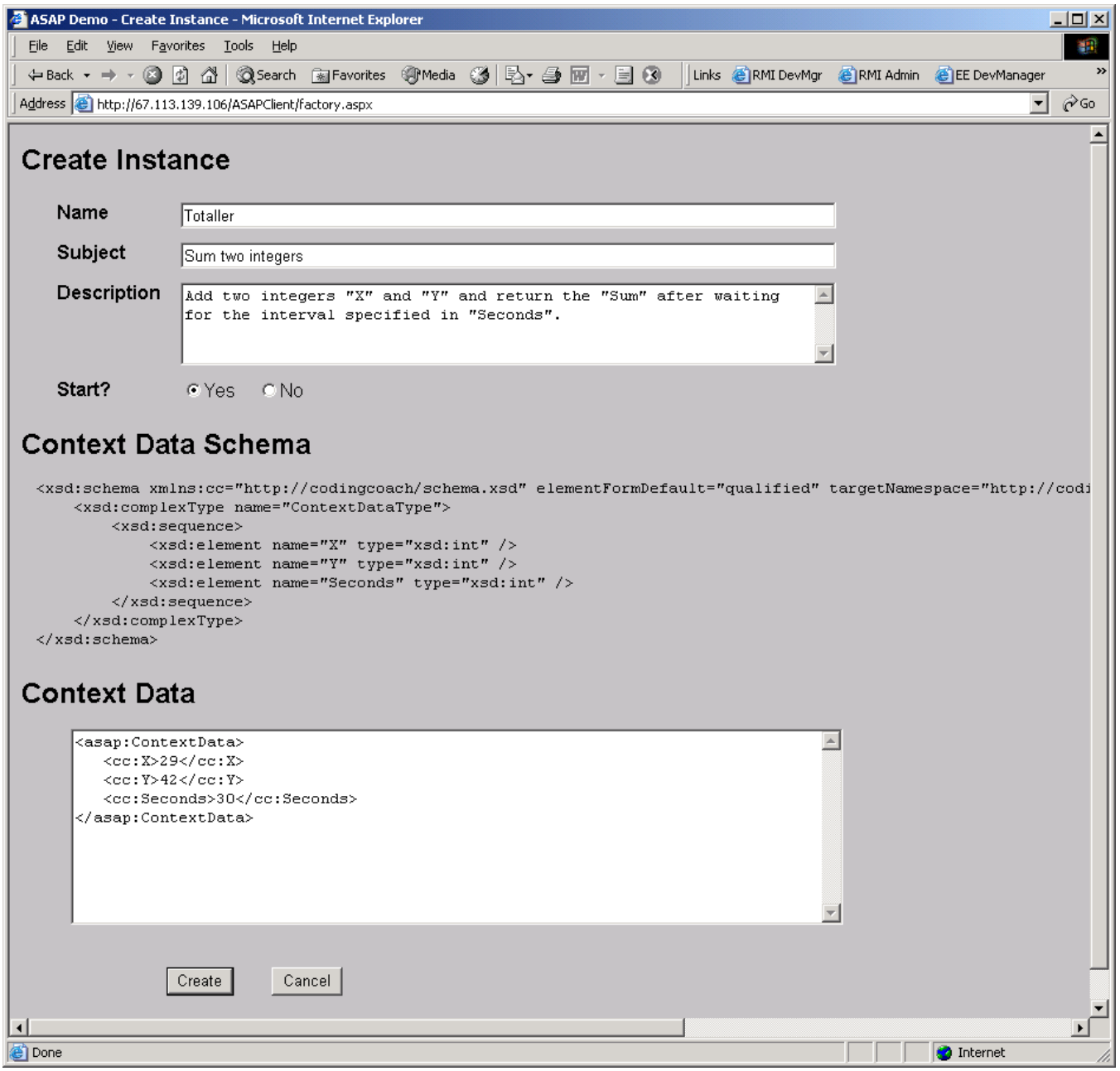
Click submit. This will send an ASAP GetProperties request to the factory. It will bring up the Factory details page which looks like this:



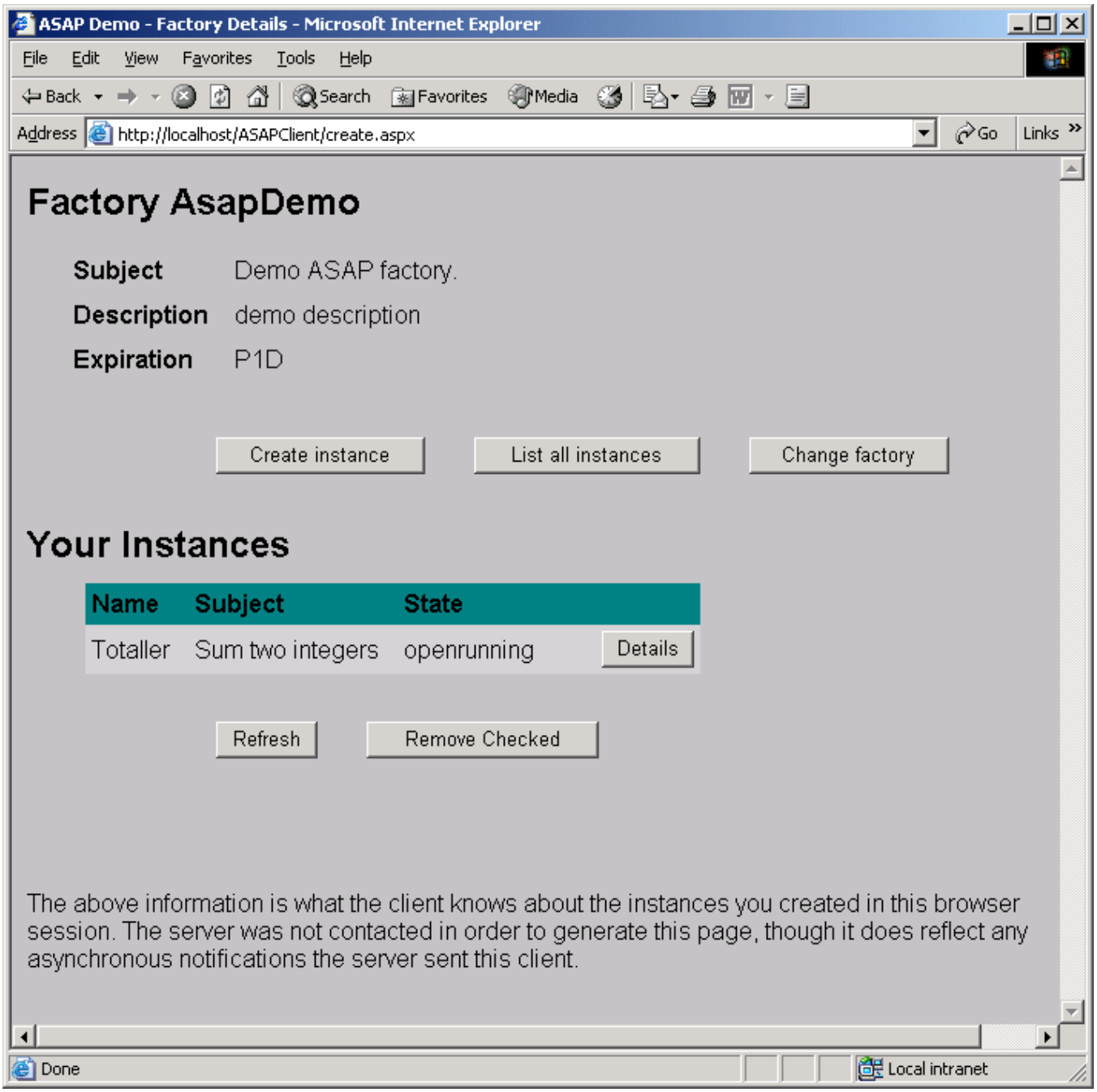
Click the Create Instance button. This should bring up the Create Instance page:



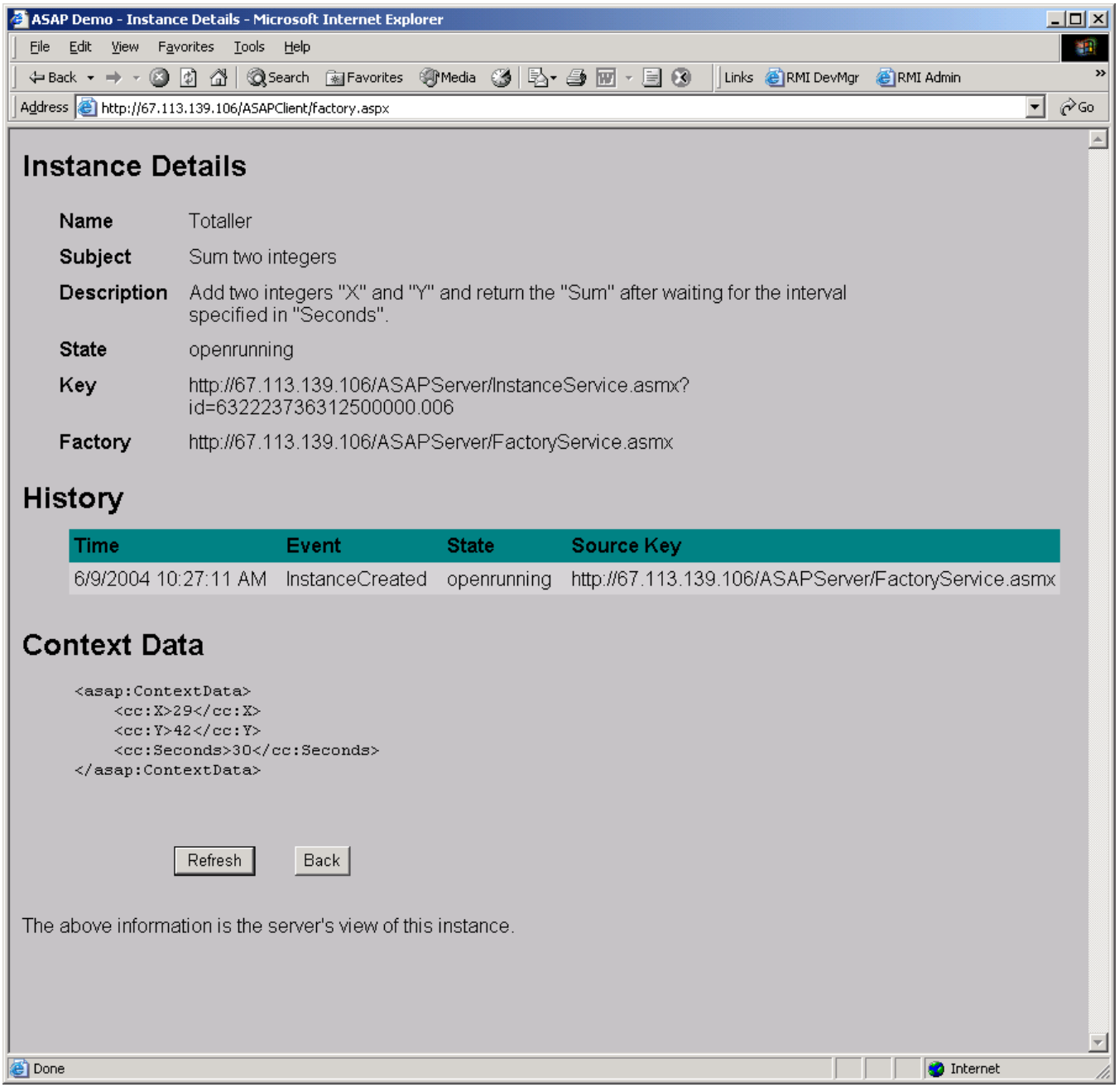
Enter the Name, Subject and Description enter the context data according to the context data schema displayed:



Click Create. This will send the CreateInstance request to the factory, and bring up the Factory Details page with the new Instance listed.

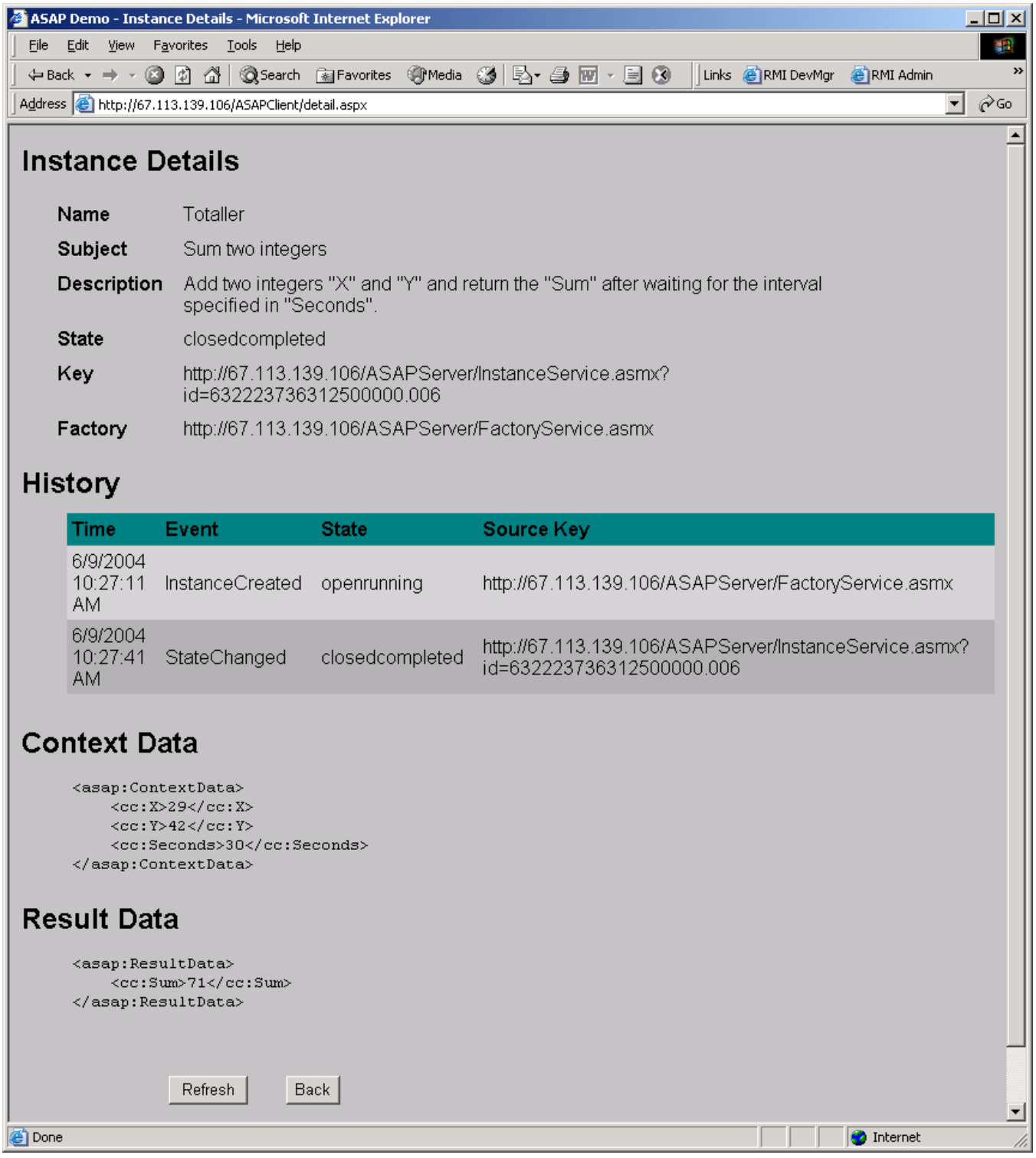


Click on the Details button to bring up the Instance details:



Notice that the instance state is open.running and the ResultData has not been populated yet.

Click on the Refresh button after approximately 30 seconds.



Notice that the instance state is now closed.completed, and ResultData includes the Sum of the two integers. This demonstrates the round trip.