smilehouse
RETHINKING COMMERCE

SMILEHOUSE WORKSPACE 17 OPEN INTERFACE API

Type: ***Technical document***

Created by: Smilehouse Research and development team

1. INTRODUCTION

Workspace provides a webservices interface for importing, updating and exporting data in XML format. Workspace Open Interface (or OI in short) has dedicated methods for importing/exporting orders as well as importing customer, product and pricing contract records. Additionally free Hibernate (HQL) object queries are supported for retrieving and combining data from products and orders, using complex selection criteria.

OpenInterface webservices are based on JAX-RPC framework and they use RPC/encoded WSDL (Web Services Description Language) binding style. W3C standard XML schemata of order, customer and product records are included in Appendix 1. A full WSDL 1.1 definition is provided in Appendix 2.

To complement the specifications, a couple of SOAP XML message examples are provided for getting started with Open Interface programming. The examples are programming language independent and make it easy to test OI API functionality by simply using a command line tool such as cURL for posting messages over HTTP.

JAX-RPC RPC/encoded style compatible webservice development tools may be used to automatically generate Open Interface client stub code for different programming languages. See Appendix 3 for webservices framework specific example client code.

2. PREREQUISITES OF OPEN INTERFACE USAGE

1. A user in Workspace with Open Interface privileges. For added security, it is recommended to create a new user with only Open Interface privileges and no access to Workspace administration interface modules.

2. The URL address of Open Interface. The address can be retrieved from http://host:port/workspace.admin/openinterfaceaddress web service. The OI address will be of format: http://host:port/workspace.admin_<licensetype>_<versionnumber>/openinterface , where <licensetype> is either 'standard' or 'operator'.

3. For importing customer and order data: customer and order questions have to be set up to match the data to be imported. Make sure that all questions referenced in the imported XML do exist and have the correct data and question type (customer/order). If they don't exist, create them at the Workspace administration interface to avoid errors while importing data.

4. For exporting orders: check that all the handling status and payment status names referenced in OrderCriteria do actually exist in Workspace. If not, create the statuses.

5. Optional: If you need to query OpenInterface WSDL definitions (see Appendix 2) run-time from Workspace, please add Xalan-J and Xerces-J libraries to Tomcat 5's shared/lib/ or Tomcat 6's lib/ directory. Missing jar files typically result in the following error when querying “openinterfaceaddress?WSDL” and “openinterface?WSDL”: “`javax.xml.transform.TransformerFactoryConfigurationError: Provider org.apache.xalan.processor.TransformerFactoryImpl not found`”. Should you encounter this problem, please don't hesitate to ask Workspace Support (**workspace.support@smilehouse.com**) to provide the required libraries and assist with installation. Note that Xalan-J and Xerces-J are not necessary for regular webshop data import and export operations, unless your preferred webservice framework needs the WSDL to be available online.

3. CONNECTING TO THE OPEN INTERFACE

3.1. RETRIEVING THE OPEN INTERFACE ADDRESS FOR AN ORGANIZATION

An easy way to get started with Open Interface is to get a HTTP POST capable command line tool such as cURL* and use it to test communication with the OI. First you have to retrieve the webshop specific Open Interface URL address from another web service called openinterfaceaddress. Workspace Open Interface is located inside the Workspace admin webapp, which has WS license type and version/build number written in its URL. The URL address of Open Interface will thus change every time Workspace is upgraded to a new version. If you are just testing Open Interface you may wish to skip this look-up phase and use directly the URL address of your current Workspace version.

The cURL command line for dynamically getting the OI address is:

```
curl -s -d "@TestMessage.xml" -H "Content-Type: text/xml"
http://host:port/workspace.admin/openinterfaceaddress
```

where TestMessage.xml is a file in the current directory, containing the following type of SOAP message:

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:enc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:ns0="http://www.smilehouse.com/wsdl"
    env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <env:Body>

        <!-- Specify the organization (database) name in String_1
parameter to ns0:getOpenInterfaceAddress -->
        <ns0:getOpenInterfaceAddress>
            <String_1 xsi:type="xsd:string">myshop</String_1>
```

```
        </ns0:getOpenInterfaceAddress>

    </env:Body>
</env:Envelope>
```

The above SOAP query receives the Open Interface address from Workspace for organization (=database name) called "myshop". Note that the web services require that the HTTP header specifies message content type as text/xml, otherwise an HTTP error 415 (Unsupported Media Type) will occur. If you get an empty response with cURL, try adding -i option to the command line to view all headers of the HTTP response.

Workspace's openinterfaceaddress web service responds to the query with (comments added):

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:enc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:ns0="http://www.smilehouse.com/types"
    env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <env:Body>

        <!-- Open Interface address is returned in result element inside
ans1:getOpenInterfaceAddressResponse.
             The address is always relative to the URL where Workspace is
installed, e.g.
               "http://localhost:8080" +
"/workspace.admin_operator_20050921111647/openinterface" ->

"http://localhost:8080/workspace.admin_operator_20050921111647/openinterf
ace"

             The result element will be returned empty if the specified
organization was not found.

        -->
        <ans1:getOpenInterfaceAddressResponse
xmlns:ans1="http://www.smilehouse.com/wsdl">
            <result
xsi:type="xsd:string">/workspace.admin_operator_20050921111647/openinterf
ace</result>
        </ans1:getOpenInterfaceAddressResponse>

    </env:Body>
</env:Envelope>
```

The response means that Open Interface messages to organization "myshop" should be sent to http://host:port/workspace.admin_operator_20050921111647/openinterface , where http://host:port/ is the base URL of your Workspace installation.

** Examples tested with cURL version 7.13.1 <http://curl.haxx.se>. Another popular command line tool for sending HTTP requests is wget <http://www.gnu.org/software/wget>.*

3.2. PREPARING TO SEND MESSAGES TO OPEN INTERFACE

The cURL syntax for sending queries to Open Interface is the same as it is for openinterfaceaddress web service, as described in the previous chapter:

```
curl -s -d "@TestMessage.xml" -H "Content-Type: text/xml"
http://host:port/workspace.admin_<licensetype>_<versionnumber>/openinterf
ace
```

where TestMessage.xml is the name of the message XML file to send and http://host:port/workspace.admin_<licensetype>_<versionnumber>/openinterface is the actual Open Interface URL. The OI response will be directed to console (stdout) by default.

4. EXPORTING DATA FROM WORKSPACE

This chapter will explain by example how to query data records from Workspace. As the data is in XML format, if the Workspace database contains characters that do not belong to well-formed XML, the interface filters them from the export output.

4.1. EXPORTING ORDERS

Open Interface retrieves orders based on rules specified in OrderCriteria element. The following example XML shows how to export orders that match all rules supported by the current version of WS Open Interface. You may save the example XML into a file (e.g. TestMessage.xml), edit the LoginInfo and OrderCriteria parameters and then send the XML message to Open Interface with cURL (see chapter 3.2 for a command line example).

Initially it might be a good idea to begin with empty ns0:OrderCriteria element and also comment out ns1:exportOrders/String_4 (the new status for retrieved orders). You will receive all orders from the specified Workspace database, which is an easy way to verify that both the Open Interface and database connections work.

Example query to retrieve orders, see comments for explanation of parameters:

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:enc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:ns0="http://www.smilehouse.com/types"
    xmlns:ns1="http://www.smilehouse.com/wsdl"
    xmlns:ns2="http://java.sun.com/jax-rpc-ri/internal"
    env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <env:Body>
```

```
        <ns1:exportOrders>
            <LoginInfo_1 href="#ID1"/>
            <OrderCriteria_2 href="#ID2"/>
            <Long_3 xsi:type="enc:long" xsi:nil="1"/>

            <!-- Set a new status (lowercase text only) - either handling
or payment status - to orders retrieved by this query.
                 If the status name is found on Workspace's handling
status list, the handling status of the order will be set.
                 If the status name is found on Workspace's payment
status list, the payment status of the order will be set.
                 Warning: if the status name is found on both handling
and payment status lists, the behavior of this parameter
                          is unknown! Please make sure that the handling
and payment statuses don't overlap.
                 Warning 2: the status must be pre-defined in Workspace
(as either handling status or payment status), otherwise
                            an error will occur.
            -->
            <String_4 xsi:type="xsd:string">retrieved</String_4>

        </ns1:exportOrders>
        <ns0:LoginInfo id="ID1" xsi:type="ns0:LoginInfo">

            <!-- Name of the Workspace database to connect -->
            <database xsi:type="xsd:string">myshop</database>

            <!-- Password of a Workspace user with Open Interface
privileges -->
            <password xsi:type="xsd:string">password</password>

            <!-- Name of the Workspace user with Open Interface
privileges -->
            <userName xsi:type="xsd:string">user</userName>

        </ns0:LoginInfo>
        <ns0:OrderCriteria id="ID2" xsi:type="ns0:OrderCriteria">

            <!-- Select orders with customer ID code any of the following
list -->
            <customerIdIn xsi:type="ns0:ArrayOfstring"
enc:arrayType="xsd:string[3]">
                <item xsi:type="xsd:string">C100</item>
                <item xsi:type="xsd:string">1234</item>
                <item xsi:type="xsd:string">S353</item>
            </customerIdIn>

            <!-- Select orders created/modified after a timestamp in
dateTime format (http://www.w3.org/TR/xmlschema-2/#dateTime) -->
            <dateAfter xsi:type="xsd:dateTime">2004-12-
27T23:59:41.000+02:00</dateAfter>

            <!-- Select orders created/modified before a timestamp in
dateTime format (http://www.w3.org/TR/xmlschema-2/#dateTime) -->
            <dateBefore xsi:type="xsd:dateTime">2005-07-
02T05:34:02.000+03:00</dateBefore>

            <!-- Select orders with handling status name in the following
list -->
```

```
            <handlingStatusNameIn xsi:type="ns0:ArrayOfstring"
enc:arrayType="xsd:string[2]">
                <item xsi:type="xsd:string">Received</item>
                <item xsi:type="xsd:string">Sent</item>
            </handlingStatusNameIn>

            <!-- handlingStatusNameNotIn: Currently not functional, do
not use -->
            <!-- <handlingStatusNameNotIn xsi:type="ns0:ArrayOfstring"
enc:arrayType="xsd:string[0]" xsi:nil="1"/> -->

            <!-- Select orders with ID number larger than the specified
integer -->
            <idGreaterThan xsi:type="enc:long">1</idGreaterThan>

            <!-- Select orders with ID number in the following list -->
            <idIn xsi:type="ns0:ArrayOfLong" enc:arrayType="enc:long[4]">
                <item xsi:type="enc:long">1</item>
                <item xsi:type="enc:long">2</item>
                <item xsi:type="enc:long">4</item>
                <item xsi:type="enc:long">5</item>
            </idIn>

            <!-- Select orders with ID number smaller than the specified
integer -->
            <idLessThan xsi:type="enc:long">10</idLessThan>

            <!-- Select orders with payment status name in the following
list -->
            <paymentStatusNameIn xsi:type="ns0:ArrayOfstring"
enc:arrayType="xsd:string[2]">
                <item xsi:type="xsd:string">no payment status</item>
                <item xsi:type="xsd:string">Cancelled</item>
            </paymentStatusNameIn>

            <!-- paymentStatusNameNotIn: Currently not functional, do not
use -->
            <!-- <paymentStatusNameNotIn xsi:type="ns0:ArrayOfstring"
enc:arrayType="xsd:string[0]" xsi:nil="1"/> -->

            <!-- Select orders with total sum greater than the specified
number (float) -->
            <sumGreaterThan xsi:type="enc:double">12.34</sumGreaterThan>

            <!-- Select orders with total sum less than the specified
number (float) -->
            <sumLessThan xsi:type="enc:double">12345.67</sumLessThan>

        </ns0:OrderCriteria>
    </env:Body>
</env:Envelope>
```

Note: handlingStatusName and paymentStatusName lists may NOT contain non-existing status names, otherwise an HTTP error 500 will result.

Note 2: idIn (list): the order of results is not going to be the same as the order of id numbers in the query.

The export format of the order data is the same as the import format. See Order XML Schema

4.2. EXPORTING CUSTOMER, PRODUCT, ORDER DATA USING HQLITERATOR

Open Interface supports exporting XML data related to customers, products and orders with HQL select queries (Hibernate Query Language v3, see http://hibernate.org for further info).

Open Interface's HQLIterator allows performing complex database queries and returning the result data in XML format conforming to the customer, product and order XML schemata - or fragments thereof where applicable. Another advantage of HQLIterator compared to OI's other export methods is the scalibility potential, as the amount of result records returned by OI at one iteration step can be specified along with the HQL query.

Table 1. Example HQL queries

HQL query	Explanation
`from Order`	Retrieves all orders.
`from Product`	Retrieves all products.
`from Customer`	Retrieves all customers.
`from Order o where timestamp(o.creationTime) >= timestampadd(day,-7,now())`	Retrieves orders which have been created during the last 7 days (or today). Note: Uses timestamp functions specific to MySQL 5.0+.
`from Product p where timestamp(p.lastModified) >= timestampadd(day,-3,now())`	Retrieves products which have been modified during the last 3 days (or today). Note: Uses timestamp functions specific to MySQL 5.0+.
`select o from Order o inner join o.answers as oa where oa.version.question.typeId=2 and oa.value='12345'`	Retrieves orders with postal code 12345 in the address. Please refer to Customer XML schema for questionType (question.typeId) numbers.
`select p from Basket b, Product p where b.itemCode = p.itemCode and not b.parent is null`	Retrieves all products that have been ordered from the shop. We choose only those Basket objects which are linked to an existing Order (Basket.parent is not null).
`from Product p where p.inventory.amount < p.inventory.alarmLimit`	Retrieves products with free stock amount below the product's stock alarm limit.
`select p from Product p inner join p.parentsHibernate as ph inner join ph.productGroup as pg where p.visible is true and pg.name = 'Software'`	Retrieves all products with visibility status set to true from a product group called Software.

HQL query	Explanation
`select p from Product p inner join p.parentsHibernate as ph inner join ph.productGroup as pg where p.visible is false and pg.groupCode in ('PG1234', 'XYZ1000')`	Retrieves all hidden products from product groups with groupcode either PG1234 or XYZ1000.
`select c from Customer c inner join c.groups as cg where cg.name = 'b-to-c new customers' and c.primaryGroup.name = 'b-to-c new customers'`	Retrieves customers that belong to customer group "b-to-c new customers" which is also set as the primary customer group of the customer.
`select c from Customer c inner join c.answers as ca where ca.question.typeId = 12 and ca.value = 'John'`	Retrieves customers whose first name is "John". Please refer to Customer XML schema for questionType (question.typeId) numbers.
`from Customer c where timestamp(c.modifiedByCustomer) >= timestampadd(hour,-24,now()) or timestamp(c.modifiedByAdmin) >= timestampadd(hour,-24,now())`	Retrieves customers whose data has been modified during the last 24 hours either by the customer herself or by the webshop administrator. Note: Uses timestamp functions specific to MySQL 5.0+.

The following SOAP message capture shows a conversation with Open Interface when querying product XML data. See added comments in the messages for parameter and return value descriptions.

Step 1: An HQL query is sent to `openHQLIterator` method:

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:enc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns0="http://www.smilehouse.com/types"
  xmlns:ns1="http://www.smilehouse.com/wsdl"
xmlns:ns2="http://java.sun.com/jax-rpc-ri/internal"
  env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <env:Body>
    <ns1:openHQLIterator>
      <LoginInfo_1 href="#ID1"/>

      <!-- The HQL query: Retrieve all Product objects that have been
modified during the last 7 days (or today).
           Errors in the query parsing and processing will cause Open
Interface to respond with OpenInterfaceException error. -->
      <String_2 xsi:type="xsd:string">from Product p where
timestamp(p.lastModified) &gt;= timestampadd(day,-7,now())</String_2>

      <!-- Iterator connection timeout in milliseconds. This iterator
will be automatically closed by Open Interface in 30 seconds. If a closed
or non-existent iterator ID is used in the subsequent method calls (Steps
```

```
2 to 4), Open Interface will respond with IteratorClosedException error.
-->
      <long_3 xsi:type="xsd:long">30000</long_3>

    </ns1:openHQLIterator>

    <!-- Login parameters: database/organization name, username and
password -->
    <ns0:LoginInfo id="ID1" xsi:type="ns0:LoginInfo">

      <!-- Name of the Workspace database to connect -->
      <database xsi:type="xsd:string">myshop</database>

      <!-- Password of a Workspace user with Open Interface privileges
-->
      <password xsi:type="xsd:string">password</password>

      <!-- Name of the Workspace user with Open Interface privileges -->
      <userName xsi:type="xsd:string">user</userName>

    </ns0:LoginInfo>

  </env:Body>
</env:Envelope>
```

The HQL was processed successfully and Open Interface responds with an iterator ID:

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:enc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns0="http://www.smilehouse.com/types"
  xmlns:ns1="http://java.sun.com/jax-rpc-ri/internal"
  env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <env:Body>
    <ans1:openHQLIteratorResponse
xmlns:ans1="http://www.smilehouse.com/wsdl">

      <!-- Iterator ID -->
      <result xsi:type="xsd:string">55</result>

    </ans1:openHQLIteratorResponse>
  </env:Body>
</env:Envelope>
```

Step 2: To get the first (or next) query result, iterate method is called with iterator ID as parameter:

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:enc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns0="http://www.smilehouse.com/types"
  xmlns:ns1="http://www.smilehouse.com/wsdl"
xmlns:ns2="http://java.sun.com/jax-rpc-ri/internal"
  env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
```

```
  <env:Body>
    <ns1:iterate>
      <LoginInfo_1 href="#ID1"/>

      <!-- Iterator ID from openHQLIteratorResponse -->
      <String_2 xsi:type="xsd:string">55</String_2>

      <!-- Number of results to return at once -->
      <int_3 xsi:type="xsd:int">1</int_3>

    </ns1:iterate>

    <!-- Login parameters: database/organization name, username and
password -->
    <ns0:LoginInfo id="ID1" xsi:type="ns0:LoginInfo">

      <!-- Name of the Workspace database to connect -->
      <database xsi:type="xsd:string">myshop</database>

      <!-- Password of a Workspace user with Open Interface privileges
-->
      <password xsi:type="xsd:string">password</password>

      <!-- Name of the Workspace user with Open Interface privileges -->
      <userName xsi:type="xsd:string">user</userName>

    </ns0:LoginInfo>

  </env:Body>
</env:Envelope>
```

Open Interface's response to the `iterate` call:

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:enc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns0="http://www.smilehouse.com/types"
  xmlns:ns1="http://java.sun.com/jax-rpc-ri/internal"
  env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <env:Body>
    <ans1:iterateResponse xmlns:ans1="http://www.smilehouse.com/wsdl">
      <result href="#ID1"/>
    </ans1:iterateResponse>
    <ns0:ExportResult id="ID1" xsi:type="ns0:ExportResult">
      <lastUpdateLogId xsi:type="enc:long" xsi:nil="1"/>

      <!-- Actual number of results in this response -->
      <resultSize xsi:type="xsd:int">1</resultSize>

      <!-- The result XML data as a String value.
           Each <product> is contained in a <result> element. -->
      <xml xsi:type="xsd:string">
        &lt;?xml version="1.0" encoding="UTF-8"?&gt;
        &lt;result&gt;
        &lt;product id="1" itemCode="1234" visible="true" created="2005-
11-01 18:14:07.000+0200" lastModified="2005-12-22 11:52:54.000+0200"
modifier="user"&gt;
```

```
        <!-- The rest of the product data omitted.
             (see Product XML Schema for a the XML schema of the data)
           -->

        &lt;/product&gt;
        &lt;/result&gt;
      </xml>
    </ns0:ExportResult>
  </env:Body>
</env:Envelope>
```

Step 3: Repeat Step 2 until a response with empty `<result/>` is returned:

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:enc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns0="http://www.smilehouse.com/types"
  xmlns:ns1="http://java.sun.com/jax-rpc-ri/internal"
  env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <env:Body>
    <ans1:iterateResponse xmlns:ans1="http://www.smilehouse.com/wsdl">

      <!-- Empty <result> means that there are no more results left to
retrieve. -->
      <result xsi:type="ns0:ExportResult" xsi:nil="1"/>

    </ans1:iterateResponse>
  </env:Body>
</env:Envelope>
```

This means there are no more results to the HQL query and iterator should be closed.

Step 4: After the query results have been received, the iterator should be closed by calling the `closeIterator` method:

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:enc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns0="http://www.smilehouse.com/types"
  xmlns:ns1="http://www.smilehouse.com/wsdl"
xmlns:ns2="http://java.sun.com/jax-rpc-ri/internal"
  env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <env:Body>
    <ns1:closeIterator>
      <LoginInfo_1 href="#ID1"/>

      <!-- Iterator ID from openHQLIteratorResponse -->
      <String_2 xsi:type="xsd:string">55</String_2>

    </ns1:closeIterator>
```

```
    <!-- Login parameters: database/organization name, username and
password -->
    <ns0:LoginInfo id="ID1" xsi:type="ns0:LoginInfo">

      <!-- Name of the Workspace database to connect -->
      <database xsi:type="xsd:string">myshop</database>

      <!-- Password of a Workspace user with Open Interface privileges
-->
      <password xsi:type="xsd:string">password</password>

      <!-- Name of the Workspace user with Open Interface privileges -->
      <userName xsi:type="xsd:string">user</userName>

    </ns0:LoginInfo>
  </env:Body>
</env:Envelope>
```

Open Interface reports that the iterator was closed successfully:

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:enc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns0="http://www.smilehouse.com/types"
  xmlns:ns1="http://java.sun.com/jax-rpc-ri/internal"
  env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
  <env:Body>
    <ans1:closeIteratorResponse
xmlns:ans1="http://www.smilehouse.com/wsdl"/>
  </env:Body>
</env:Envelope>
```

Data export with HQLIterator is now complete.

4.3. EXPORTING CUSTOMER INFORMATION USING CUSTOMER CRITERIA

First you need to create simplequery.xml file containing your query.

After thatyou need to write the curl syntax, ensuring that you are in the same directory as the xml file, then run this through the terminal:

```
curl -s -d "@simplequery.xml" -H "Content-Type: text/xml"
http://localhost:8080/workspace.<licensetype>_<versionnumber>/openinterfa
ce
```

Below is a simplequery.xml file description for exporting customers from workspace using customer criteria.

Replace `workspace_database`, `user_password` and `username` with your own workspace specific information. When this query is executed you get a result which contains user that match you criteria. For more criteria see Appendix 2: WSDL definitions

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:enc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns0="http://www.smilehouse.com/types"
xmlns:ns1="http://www.smilehouse.com/wsdl"
xmlns:ns2="http://java.sun.com/jax-rpc-ri/internal"
env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <env:Body>
        <ns1:exportCustomers>
            <LoginInfo_1
href="#ID1"/><CustomerCriteria_2 href="#ID2"/>
        </ns1:exportCustomers>

<!-- Login information, database name, and Workspace user information -->

        <ns0:LoginInfo id="ID1"
xsi:type="ns0:LoginInfo">
            <database xsi:type="xsd:string">workspace_database</database>

<!-- Password of a Workspace user with Open Interface privileges -->
<password xsi:type="xsd:string">user_password</password>

<!-- Username of a Workspace user with Open Interface privileges -->
<userName xsi:type="xsd:string">username</userName>
        </ns0:LoginInfo>

<!-- Login information section ends -->
<!-- Customer criteria section begins -->

        <ns0:CustomerCriteria id="ID2"
xsi:type="ns0:CustomerCriteria">
        <adminModifiedBefore xsi:type="xsd:dateTime">2007-04-
11T00:00:00.00+03:00</adminModifiedBefore>
            <customerGroup xsi:type="xsd:string" xsi:nil="1"/><customerId
xsi:type="xsd:string"
xsi:nil="1"/>
        <customerModifiedAfter xsi:type="xsd:dateTime">2007-04-
01T00:00:00.00+03:00</customerModifiedAfter>
            <firstVisitDateAfter xsi:type="xsd:dateTime" xsi:nil="1"/>
            <firstVisitDateBefore
xsi:type="xsd:dateTime" xsi:nil="1"></firstVisitDateBefore><idGreaterThan
xsi:type="xsd:string"
xsi:nil="1"/>
            <idIn xsi:type="ns0:ArrayOfstring"
enc:arrayType="xsd:string[0]" xsi:nil="1"/>
            <idLessThan
xsi:type="xsd:string" xsi:nil="1"/>
            <lastVisitDateAfter xsi:type="xsd:dateTime" xsi:nil="1"/>
<lastVisitDateBefore
xsi:type="xsd:dateTime" xsi:nil="1"/>

<!-- Operation type between customer and admin range (OR / AND) -->
        <modifyOperation xsi:type="xsd:string">AND</modifyOperation>
            <primaryCustomerGroup xsi:type="xsd:string"
xsi:nil="1"/>
        </ns0:CustomerCriteria>
```

```
<!-- Customer criteria section ends -->

    </env:Body>
</env:Envelope>
```

5. SENDING WEBSHOP EMAILS THROUGH OPEN INTERFACE

This chapter will explain by example how to use Workspace webshop's template based email sending feature via Open Interface.

Message templates define the email header (addresses, subject) and body contents. The templates can be created and edited in the Customers module / Message settings part of Workspace administration interface.

The message templates may contain variable references to data fields of orders, customers and products stored in the webshop database. For the value references to work, Open Interface “mailSending” SOAP XML query must contain orderId, customerId and/or product itemcode parameters.

A commented SOAP XML message format for sending an email from Workspace webshop follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:enc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns0="http://www.smilehouse.com/types"
xmlns:ns1="http://www.smilehouse.com/wsdl"
xmlns:ns2="http://java.sun.com/jax-rpc-ri/internal"
env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<env:Body>

        <ns1:mailSending>
            <LoginInfo_1 href="#ID1"/>

            <!-- Name of the message template to send email with. The
template must exist in the webshop. -->
            <String_2 xsi:type="xsd:string">Test message</String_2>

            <!-- Id number of an order to associate the message with
(optional, may be left empty) -->
            <String_3 xsi:type="xsd:string">100</String_3>

            <!-- Id number of a customer to associate the message with
(optional, may be left empty) -->
            <String_4 xsi:type="xsd:string"></String_4>

            <!-- Itemcode of a product to associate the message with
(optional, may be left empty) -->
            <String_5 xsi:type="xsd:string"></String_5>
        </ns1:mailSending>

        <ns0:LoginInfo id="ID1" xsi:type="ns0:LoginInfo">
```

```
            <!-- Name of the Workspace database to connect -->
            <database xsi:type="xsd:string">myshop</database>

            <!-- Password of a Workspace user with Open Interface
privileges -->
            <password xsi:type="xsd:string">password</password>

            <!-- Name of the Workspace user with Open Interface
privileges -->
            <userName xsi:type="xsd:string">user</userName>

        </ns0:LoginInfo>

   </env:Body>
</env:Envelope>
```

6. IMPORTING DATA TO WORKSPACE

This chapter will explain by example how to import data records to Workspace. WS Open Interface currently supports importing and updating customer, product, order and contract pricing information.

6.1. IMPORTING AND UPDATING CUSTOMERS

The following Open Interface example query imports two customers. Only the specified customer data fields will be written to the database, unspecified fields being left empty (insert mode) or unchanged (update mode). In update modes customers are first tried to match with the database id. If the id is not given the login name will be used instead. If neither of these are given in the XML or no matching records are found the customer is considered to be new.

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:enc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:ns0="http://www.smilehouse.com/types"
xmlns:ns1="http://www.smilehouse.com/wsdl"
    xmlns:ns2="http://java.sun.com/jax-rpc-ri/internal"
    env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <env:Body>
        <ns1:importCustomers>
            <LoginInfo_1 href="#ID1"/>

            <!-- String_2 contains the escaped customer data XML to be
imported,
                 one or more <customer> elements inside a <customers>
container -->

            <String_2 xsi:type="xsd:string">
 &lt;?xml version="1.0" encoding="UTF-8"?&gt;
&lt;customers version="0.93"&gt;
    &lt;customer id="101" login="btocdemo"&gt;
        &lt;statistics firstVisit="2007-04-19 00:00:00.000+0300"
latestVisit="2007-04-19 00:00:00.000+0300" timesVisited="6"
sumOfAllOrders="0.00" averageSum="0.00" averageProducts="0.0"/&gt;
        &lt;answers&gt;
```

```
            &lt;answer questionType="0"&gt;
                &lt;questionText&gt;Last name&lt;/questionText&gt;
                &lt;answerText&gt;Doe&lt;/answerText&gt;
            &lt;/answer&gt;
            &lt;answer questionType="12"&gt;
                &lt;questionText&gt;First name&lt;/questionText&gt;
                &lt;answerText&gt;John&lt;/answerText&gt;
            &lt;/answer&gt;
            &lt;answer questionType="5"&gt;
                &lt;questionText&gt;Phone&lt;/questionText&gt;
                &lt;answerText&gt;+358 123 123&lt;/answerText&gt;
            &lt;/answer&gt;
            &lt;answer questionType="3"&gt;
                &lt;questionText&gt;Postal office&lt;/questionText&gt;
                &lt;answerText&gt;Helsinki&lt;/answerText&gt;
            &lt;/answer&gt;
            &lt;answer questionType="8"&gt;
                &lt;questionText&gt;Email&lt;/questionText&gt;
                &lt;answerText&gt;email@example.com&lt;/answerText&gt;
            &lt;/answer&gt;
            &lt;answer questionType="4"&gt;
                &lt;questionText&gt;Country&lt;/questionText&gt;
                &lt;answerText&gt;FI&lt;/answerText&gt;
            &lt;/answer&gt;
            &lt;answer questionType="2"&gt;
                &lt;questionText&gt;Postal number&lt;/questionText&gt;
                &lt;answerText&gt;00100&lt;/answerText&gt;
            &lt;/answer&gt;
            &lt;answer questionType="1"&gt;
                &lt;questionText&gt;Street address&lt;/questionText&gt;
                &lt;answerText&gt;Streetaddress 102 C&lt;/answerText&gt;
            &lt;/answer&gt;
            &lt;answer questionType="16"&gt;
                &lt;questionText&gt;Company&lt;/questionText&gt;
                &lt;answerText&gt;&lt;/answerText&gt;
            &lt;/answer&gt;
        &lt;/answers&gt;
    &lt;/customer&gt;
&lt;/customers&gt;

            </String_2>

            <!-- Import mode:
                 0 = Insert or update - Insert if the customer does not
exist in the database or update the specified fields
                                        if it does exist
                 1 = Only insert      - Do not update any existing
customers in the database, only insert new customers
                 2 = Only update      - Update if the customer exists in
the database, otherwise do nothing
                 3 = Insert as new    - Does not try to match existing
customers, inserts all customers as new records
                 4 = Replace all      - Clears the entire customer
database before starting to import (insert) customers! -->
            <int_3 xsi:type="xsd:int">0</int_3>

            <!-- createCustomerGroups switch:
                 If true, Workspace will automatically create new
customer group(s) if non-existent group(s) are referenced
                 in <customerGroups> elements.
```

```
                If false, any references to non-existent customer groups
will result in OpenInterface errors.
            -->
            <boolean_4 xsi:type="xsd:boolean">true</boolean_4>

        </ns1:importCustomers>
        <ns0:LoginInfo id="ID1" xsi:type="ns0:LoginInfo">

            <!-- Name of the Workspace database to connect -->
            <database xsi:type="xsd:string">myshop</database>

            <!-- Password of a Workspace user with Open Interface
privileges -->
            <password xsi:type="xsd:string">password</password>

            <!-- Name of the Workspace user with Open Interface
privileges -->
            <userName xsi:type="xsd:string">user</userName>

        </ns0:LoginInfo>
    </env:Body>
</env:Envelope>
```

Open Interface response (with added comments) to the previous customer import message:

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:enc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:ns0="http://www.smilehouse.com/types"
xmlns:ns1="http://java.sun.com/jax-rpc-ri/internal"
    env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <env:Body>
        <ans1:importCustomersResponse
xmlns:ans1="http://www.smilehouse.com/wsdl">
            <result href="#ID1"/>
        </ans1:importCustomersResponse>
        <ns0:ImportResult id="ID1" xsi:type="ns0:ImportResult">

            <!-- List of Ids of customers that were successfully inserted
to the database -->
            <insertedIds xsi:type="ns1:linkedList"
enc:arrayType="xsd:anyType[2]">
                <item xsi:type="xsd:string">101</item>
            </insertedIds>

            <!-- List of Ids of customers that were successfully removed
from the database
                 (Lists deleted customer Ids when using import mode:
Replace all) -->
            <removedIds xsi:type="ns1:linkedList"
enc:arrayType="xsd:anyType[0]"/>

            <!-- List of Ids of customers that were successfully updated
in the database -->
            <updatedIds xsi:type="ns1:linkedList"
enc:arrayType="xsd:anyType[0]"/>
```

```
            <!-- List of warning messages resulting from the customer
import -->
            <warnings xsi:type="ns1:linkedList"
enc:arrayType="xsd:anyType[0]"/>

        </ns0:ImportResult>
    </env:Body>
</env:Envelope>
```

The response message states that the customer info was successfully imported to Workspace and was assigned an internal customer ID number 101. If the same customer import message was sent again, the reply would list the customers in `<updateIds>` instead of `<insertedIds>`, since the import mode was set to "Insert or update" and the customers with logins 'testeri' and 'johnsmith' would already exist in the database.

6.2. IMPORTING AND UPDATING PRODUCTS

The following Open Interface query will import two products or update the products' information if they already exist in Workspace. Only the specified product data fields will be written to the database with fields unspecified in product XML being either left empty (Insert mode) or set to keep their previous value (Update mode).

In Update mode the products are matched with their itemCode, comparing the itemCode specified in the XML to product records in the database. Insert/Update mode updates data for products that exist in the database and creates new product records for each new itemCode in the XML.

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:enc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:ns0="http://www.smilehouse.com/types"
    xmlns:ns1="http://www.smilehouse.com/wsdl"
    xmlns:ns2="http://java.sun.com/jax-rpc-ri/internal"
    env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <env:Body>
        <ns1:importProducts2>
            <LoginInfo_1 href="#ID1"/>

            <!-- String_2 contains the escaped product data XML to be imported,
                 one or more <product> elements inside a <products> container.
                 Workspace 1.5 and newer require at least version="0.92" for
                 <products> as shown in this example. Changes from Workspace 1.4
                 product XML version 0.91 are:

                 - <productGroup> -elements must now appear in <productGroups> container,
                   since one product may belong to multiple product groups.


                  -->
            <String_2 xsi:type="xsd:string">&lt;?xml version="1.0" encoding="UTF-8"?&gt;
                &lt;products version="0.98"&gt;
                &lt;product itemCode="12345" visible="true"&gt;
                &lt;productGroup path="/Books/Technical"/&gt;
                &lt;name&gt;MASTERING WORKSPACE OPEN INTERFACE VOL. 2&lt;/name&gt;
                &lt;pricing priceWithVat="25.50" vat="22"/&gt;
                &lt;openField1&gt;some&lt;/openField1&gt;
                &lt;openField2&gt;text&lt;/openField2&gt;
                &lt;/product&gt;
```

```
                &lt;product itemCode="23456" visible="false"&gt;
                &lt;productGroups&gt;
                  &lt;productGroup path="/Clothing/T-Shirts"/&gt;
                &lt;/productGroups&gt;
                &lt;name&gt;WORKSPACE USER T-SHIRT&lt;/name&gt;
                &lt;pricing priceWithVat="19.95" vat="22"/&gt;
                &lt;picture1 url="23456.jpg"/&gt;
                &lt;picture2 url="23456_large.jpg"/&gt;
                &lt;openField1&gt;more&lt;/openField1&gt;
                &lt;openField2&gt;details&lt;/openField2&gt;
                &lt;/product&gt;
                &lt;/products&gt;</String_2>

            <!-- Import mode:
                 0 = Insert or update - Insert if the product does not exist in the
database or update the specified fields
                                        if it does exist
                 1 = Only insert      - Do not update any existing products in the
database, only insert new products
                 2 = Only update      - Update if the product exists in the database,
otherwise do nothing
                (3 = Reserved         - This mode reserved for future use)
                 4 = Replace all      - Clears the entire product database before starting
to import (insert) products! -->
            <int_3 xsi:type="xsd:int">0</int_3>

            <!-- additiveOptionUpdate switch:
                 If true, after update the choice list will be a union of product's
existing choices in the database
                 and the new choices from XML (existing options and choices will not be
deleted).
                 If false, the option list from XML will replace the current options of the
product in database.
            -->
            <boolean_4 xsi:type="xsd:boolean">false</boolean_4>

            <!-- additiveGroupUpdate switch:
                 If true, after update the product group list will be a union of product's
existing group associations
                 in the database and the new product groups specified in the XML (existing
product group associations
                 will not be deleted).
                 If false, the product group list from XML will replace the product's
current group associations in database.
            -->
            <boolean_5 xsi:type="xsd:boolean">false</boolean_5>

            <!-- List of protected product groups (full paths): products's membership of
these groups
                 is always protected from modifications. Products can not be added to or
removed from
                 protected groups. The listed protected product groups must exist in the
database,
                 otherwise the import operation will fail with an error message.
                 This example is an empty list, meaning that products in all product groups
can be modified
                 by the imported XML -->
            <arrayOfString_6 xsi:type="ns0:ArrayOfstring" enc:arrayType="xsd:string[0]"
xsi:nil="1"/>

            <!-- This example would protect ProtectedItems1 and ProtectedItems2 product
groups,
                 e.g. not remove any product's membership of these groups even if
additiveGroupUpdate
                 is set to false:

                 <arrayOfString_6 xsi:type="ns0:ArrayOfstring"
enc:arrayType="xsd:string[2]">
                    <item xsi:type="xsd:string">/ProtectedItems1</item>
                    <item xsi:type="xsd:string">/ProtectedItems2</item>
                 </arrayOfString_6>
            -->

            <!-- createProductGroups:
```

```
                If true, new product group(s) will be automatically created if an unknown
product group path(s) are
                specified in <productGroup> element(s) -->
            <boolean_7 xsi:type="xsd:boolean">true</boolean_7>

        </ns1:importProducts2>
        <ns0:LoginInfo id="ID1" xsi:type="ns0:LoginInfo">

            <!-- Name of the Workspace database to connect -->
            <database xsi:type="xsd:string">myshop</database>

            <!-- Password of a Workspace user with Open Interface privileges -->
            <password xsi:type="xsd:string">password</password>

            <!-- Name of the Workspace user with Open Interface privileges -->
            <userName xsi:type="xsd:string">user</userName>

        </ns0:LoginInfo>
    </env:Body>
</env:Envelope>
```

Open Interface response (with added comments) to the above product import message states that the products with itemCodes 12345 and 23456 were successfully imported to Workspace:

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:enc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:ns0="http://www.smilehouse.com/types"
    xmlns:ns1="http://java.sun.com/jax-rpc-ri/internal"
    env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <env:Body>
        <ans1:importProducts2Response
xmlns:ans1="http://www.smilehouse.com/wsdl">
            <result href="#ID1"/>
        </ans1:importProducts2Response>
        <ns0:ImportResult id="ID1" xsi:type="ns0:ImportResult">

            <!-- List of itemCodes of products that were successfully
inserted to the database -->
            <insertedIds xsi:type="ns1:linkedList"
enc:arrayType="xsd:anyType[2]">
                <item xsi:type="enc:long">12345</item>
                <item xsi:type="enc:long">23456</item>
            </insertedIds>

            <!-- List of Ids of products that were successfully removed
from the database
                 (Lists deleted product Ids when using import mode:
Replace all) -->
            <removedIds xsi:type="ns1:linkedList"
enc:arrayType="xsd:anyType[0]"/>

            <!-- List of Ids of products that were successfully updated
in the database -->
            <updatedIds xsi:type="ns1:linkedList"
enc:arrayType="xsd:anyType[0]"/>

            <!-- List of warning messages resulting from the product
import -->
```

```
            <warnings xsi:type="ns1:linkedList"
enc:arrayType="xsd:anyType[0]"/>

        </ns0:ImportResult>
    </env:Body>
</env:Envelope>
```

If the same product import message was sent again, the reply would list the products in `<updateIds>` instead of `<insertedIds>`. This is because the import mode was set to "Insert or update" and the products with itemCodes 12345 and 23456 would already exist in the database.

Product import API and XML format has changed from Workspace version 1.4 to 1.5 (Product XML Schema v0.9 -> v0.92):

- importProducts2 method replaces importProducts, and importProducts became deprecated
- new import options: additiveOptionUpdate and additiveGroupUpdate
- import mode "insert as new" no longer available
- amountFactor attribute added to <amountFactor> element
- internal productID's no longer available; matching products for update is now based on itemCode only
- itemCodes are now unique, so multiple product instances with the same itemCode can not exist
- one product can belong to multiple product groups: <productGroup> element(s) need to be relocated inside a <productGroups> container element

6.3. IMPORTING AND UPDATING PRICING CONTRACTS

The following Open Interface query will import three pricing contracts or update if they already exist in Workspace. The pricing contracts have one or two keys. One of the keys is always the product itemCode. The other can be one of the following: customer Id, customer group Id or customer's external pricelist Id.

In Update mode the existing contracts are updated. The existing contract is the first match item code and the first match of the following other keys. If none of these specifying values is given, the contract to update is determined by the item code.

1. Customer id
2. External price list id
3. Customer group id

The rule is updated into the database even if there are no changes.

In Insert mode only new contracts are stored into database.

In Replace all mode the existing contracts are **removed** prior to inserting and updating. Thus it is not recommended unless you always replace the whole pricing info.

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:enc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:ns0="http://www.smilehouse.com/types"
    xmlns:ns1="http://www.smilehouse.com/wsdl"
    xmlns:ns2="http://java.sun.com/jax-rpc-ri/internal"
    env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <env:Body>
        <ns1:importPricelist>
          <LoginInfo_1 href="#ID1"/>

            <!-- String_2 contains escaped pricing contract XML data to
be imported,
                 one or more <pricecontract> elements inside a
<pricecontracts> container.
             -->
            <String_2 xsi:type="xsd:string">&lt;?xml version="1.0"
encoding="UTF-8"?&gt;
              &lt;pricecontracts version="0.91"&gt;
                &lt;pricecontract itemCode="12345" customerId=""
externalPricelistId="" customerGroupId="resellers" pricerule="100"/&gt;
                &lt;pricecontract itemCode="23456" customerId="1"
externalPricelistId="" customerGroupId="" pricerule="-50%"/&gt;
                &lt;pricecontract itemCode="23456" customerId=""
externalPricelistId="10" customerGroupId="" pricerule="-10.22"/&gt;
             &lt;/pricecontracts&gt;
            </String_2>
               <!-- Import mode:
                 0 = Insert or update - Insert if the pricing contract
does not exist in the database or update if it does exist
```

```
                  1 = Only insert      - Do not update any existing
pricing contracts in the database, only insert new ones
                  2 = Only update      - Update if the pricing contract
exists in the database, otherwise do nothing
                  4 = Replace all      - Clears the entire pricing
contract list database before starting to import (insert)
                                         pricing contracts!
                -->
            <int_3 xsi:type="xsd:int">0</int_3>
        </ns1:importPricelist>
        <ns0:LoginInfo id="ID1" xsi:type="ns0:LoginInfo">
            <!-- Name of the Workspace database to connect -->
            <database xsi:type="xsd:string">myshop</database>
            <!-- Password of a Workspace user with Open Interface
privileges -->
            <password xsi:type="xsd:string">password</password>
            <!-- Name of the Workspace user with Open Interface
privileges -->
            <userName xsi:type="xsd:string">user</userName>
        </ns0:LoginInfo>
    </env:Body>
</env:Envelope>
```

Open Interface response (with added comments) to the above pricing contracts import message states that the three contracts were successfully updated to the database:

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope
    xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:enc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:ns0="http://www.smilehouse.com/types"
    xmlns:ns1="http://java.sun.com/jax-rpc-ri/internal"
    env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <env:Body>
        <ans1:importPricelistResponse
xmlns:ans1="http://www.smilehouse.com/wsdl">
            <result href="#ID1"/>
        </ans1:importPricelistResponse>
        <ns0:ImportResult id="ID1" xsi:type="ns0:ImportResult">

            <insertedIds xsi:type="ns1:linkedList"
enc:arrayType="xsd:anyType[0]"/>

            <removedIds xsi:type="ns1:linkedList"
enc:arrayType="xsd:anyType[0]"/>

            <!-- List of Ids of pricing contracts that were successfully
updated in the database -->
            <updatedIds xsi:type="ns1:linkedList"
enc:arrayType="xsd:anyType[3]">
                <item xsi:type="enc:long">326</item>
                <item xsi:type="enc:long">327</item>
                <item xsi:type="enc:long">328</item>
            </updatedIds>
            <warnings xsi:type="ns1:linkedList"
enc:arrayType="xsd:anyType[0]"/>
        </ns0:ImportResult>
```

```
    </env:Body>
</env:Envelope>
```

The response message states that the contracts with ids 326, 327 and 328 were successfully updated to Workspace. If the pricing contracts hadn't already existed in database, the same result would been in `<insertedIds>`.

6.4. IMPORTING AND UPDATING ORDERS

The following Open Interface query will import one sales order to Workspace. The example creates a new order with auto-generated Id number. It is also possible to specify an order Id to update an existing order. See comments in the XML below for description of different import modes and update options.

The export format of the order data is the same as the import format. See Order XML Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:enc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:ns0="http://www.smilehouse.com/types"
xmlns:ns1="http://www.smilehouse.com/wsdl"
    xmlns:ns2="http://java.sun.com/jax-rpc-ri/internal"
    env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <env:Body>
        <ns1:importOrders>
            <LoginInfo_1 href="#ID1"/>

            <!-- String_2 contains escaped order XML data to be imported,
                 one or more <order> elements inside an <orders>
          container. The example data creates a new order with auto-
generated id number. The id number may also be given with an id-attribute
to the order element, which makes it possible to update an existing
order.
             -->
            <String_2 xsi:type="xsd:string">
          &lt;?xml version="1.0" encoding="UTF-8"?&gt;
          &lt;orders version="1.00"&gt;
&lt;!-- orders element may contain any number of order elements ...--&gt;
                    &lt;order
                              referenceNumber="2345"
                              followUpCode="100002"
                              cancelled="false"
                              time="2011-03-3112:34:56.000+0300"
                              effectivePricesWithVat="true"&gt;

&lt;!-- customerId is provided if an order is associated with a
registered customer in Workspace webshop --&gt;
                    &lt;customerId&gt;3&lt;/customerId&gt;
                    &lt;total&gt;
                              &lt;withoutVat currency="EUR"&gt;
                                        7651.22
                              &lt;/withoutVat&gt;
                              &lt;withVat currency="EUR"&gt;
```

```
                                        9411.00
                              &lt;/withVat&gt;
                              &lt;vatAmount currency="EUR"&gt;
                                        1759.78
                              &lt;/vatAmount&gt;
                    &lt;/total&gt;
                    &lt;payment method="invoice"&gt;
                              &lt;vatRate&gt;23&lt;/vatRate&gt;
                              &lt;cost&gt;
                                        &lt;withoutVat
                                        currency="EUR"&gt;
                                                  8.94
                                        &lt;/withoutVat&gt;
                                        &lt;withVat
                                        currency="EUR"&gt;
                                                  11.00
                                        &lt;/withVat&gt;
                                        &lt;vatAmount
                                        currency="EUR"&gt;
                                                  2.06
                                        &lt;/vatAmount&gt;
                              &lt;/cost&gt;
                    &lt;/payment&gt;
                    &lt;delivery code="PD"
                                method="PersonalDelivery"&gt;
                              &lt;vatRate&gt;0&lt;/vatRate&gt;
                              &lt;cost&gt;
                                        &lt;withoutVat
                                        currency="EUR"&gt;
                                                  0.00
                                        &lt;/withoutVat&gt;
                                        &lt;withVat currency="EUR"&gt;
                                                  0.00
                                        &lt;/withVat&gt;
                                        &lt;vatAmount
                                        currency="EUR"&gt;
                                                  0.00
                                        &lt;/vatAmount&gt;
                              &lt;/cost&gt;
                    &lt;/delivery&gt;
                    &lt;handlingStatus id="18"
                              name="Transferred to ERP system" /&gt;
                    &lt;paymentStatus id="3" name="Order paid" /&gt;
                    &lt;currency
                              code="EUR" label=" €"
                              secondaryCode="USD" secondaryLabel="USD"
                              primaryDecimals="2" secondaryDecimals="2"
                              currencyRelation="0.77" clientFormat="2"
                              adminFormat="0" /&gt;
                    &lt;baskets&gt;
                              &lt;basket id="17"&gt;
                                        &lt;itemCode&gt;
                                                  1000-spe
                                        &lt;/itemCode&gt;
                                        &lt;productName&gt;
                                        Smilehouse Workspace 2
                                        &lt;/productName&gt;
                                        &lt;vatRate&gt;
                                                  23
                                        &lt;/vatRate&gt;
```

```
                                        &lt;unitPrice&gt;
                                                  &lt;withoutVat
                                                  currency="EUR"&gt;
                                                            772.36
                                                  &lt;/withoutVat&gt;
                                                  &lt;withVat
                                                  currency="EUR"&gt;
                                                            950.00
                                                  &lt;/withVat&gt;
                                                  &lt;vatAmount
                                                  currency="EUR"&gt;
                                                            177.64
                                                  &lt;/vatAmount&gt;
                                        &lt;/unitPrice&gt;
                                        &lt;quantity&gt;
                                                  2
                                        &lt;/quantity&gt;
                                        &lt;total&gt;
                                                  &lt;withoutVat
                                                  currency="EUR"&gt;
                                                            1544.72
                                                  &lt;/withoutVat&gt;
                                                  &lt;withVat
                                                  currency="EUR"&gt;
                                                            1900.00
                                                  &lt;/withVat&gt;
                                                  &lt;vatAmount
                                                  currency="EUR"&gt;
                                                            355.28
                                                  &lt;/vatAmount&gt;
                                        &lt;/total&gt;
                                        &lt;options&gt;
                                                  &lt;option&gt;
                                                  Special Edition
                                                  &lt;/option&gt;
                                        &lt;/options&gt;
                                        &lt;supplier/&gt;
                                        &lt;openField1/&gt;
                                        &lt;openField2/&gt;
                                        &lt;openField3/&gt;
                                        &lt;openField4/&gt;
                                        &lt;openField5/&gt;
                                        &lt;openField6/&gt;
                                        &lt;openField7/&gt;
                                        &lt;openField8/&gt;
                                        &lt;openField9/&gt;
                                        &lt;openField10/&gt;
                              &lt;/basket&gt;
                    &lt;/baskets&gt;
                    &lt;answers&gt;
                              &lt;answer questionType="0"&gt;
                                        &lt;questionText&gt;
                                                  Full name
                                        &lt;/questionText&gt;
                                        &lt;answerText&gt;
                                        Mr. Workspace Developer
                                        &lt;/answerText&gt;
                              &lt;/answer&gt;
                              &lt;answer questionType="1"&gt;
                                        &lt;questionText&gt;
```

```
                                                Street address
                                        &lt;/questionText&gt;
                                        &lt;answerText&gt;
                                                Itälahdenkatu 22 A
                                        &lt;/answerText&gt;
                                &lt;/answer&gt;
                                &lt;answer questionType="2"&gt;
                                        &lt;questionText&gt;
                                                Postal code
                                        &lt;/questionText&gt;
                                        &lt;answerText&gt;
                                                00210
                                        &lt;/answerText&gt;
                                &lt;/answer&gt;
                                &lt;answer questionType="3"&gt;
                                        &lt;questionText&gt;
                                                Postal office
                                        &lt;/questionText&gt;
                                        &lt;answerText&gt;
                                                Helsinki
                                        &lt;/answerText&gt;
                                &lt;/answer&gt;
                                &lt;answer questionType="4"&gt;
                                        &lt;questionText&gt;
                                                Country
                                        &lt;/questionText&gt;
                                        &lt;answerText&gt;
                                                FI
                                        &lt;/answerText&gt;
                                &lt;/answer&gt;
                                &lt;answer questionType="100"&gt;
                                        &lt;questionText&gt;
                                                Custom question 0
                                        &lt;/questionText&gt;
                                        &lt;answerText&gt;
                                        An answer to custom question 0
                                        &lt;/answerText&gt;
                                &lt;/answer&gt;
                                &lt;answer questionType="10"&gt;
                                        &lt;questionText&gt;
                                                Payment method
                                        &lt;/questionText&gt;
                                        &lt;answerText&gt;
                                                invoice
                                        &lt;/answerText&gt;
                                &lt;/answer&gt;
                                &lt;answer questionType="5"&gt;
                                        &lt;questionText&gt;
                                                Phone number
                                        &lt;/questionText&gt;
                                        &lt;answerText&gt;
                                                +35892512210
                                        &lt;/answerText&gt;
                                &lt;/answer&gt;
                                &lt;answer questionType="8"&gt;
                                        &lt;questionText&gt;
                                                Email address
                                        &lt;/questionText&gt;
                                        &lt;answerText&gt;
                                                null@smilehouse.com
```

```
                                        &lt;/answerText&gt;
                                &lt;/answer&gt;
                                &lt;answer questionType="6"&gt;
                                        &lt;questionText&gt;
                                                Phone number
                                        &lt;/questionText&gt;
                                        &lt;answerText&gt;
                                                +358401234567
                                        &lt;/answerText&gt;
                                &lt;/answer&gt;
                                &lt;answer questionType="11"&gt;
                                        &lt;questionText&gt;
                                                Delivery method
                                        &lt;/questionText&gt;
                                        &lt;answerText&gt;
                                                Personal Delivery
                                        &lt;/answerText&gt;
                                &lt;/answer&gt;
                        &lt;/answers&gt;
                        &lt;orderHistory&gt;
                                &lt;entry
                                time="2011-03-31 12:34:56.000+0300"
                                id="8000"&gt;
                                        &lt;status id="1"
                                        name="Received" /&gt;
                                        &lt;creator&gt;
                                                Workspace
                                        &lt;/creator&gt;
                                        &lt;comment /&gt;
                                &lt;/entry&gt;
                                &lt;entry
                                time="2011-03-31 12:34:56.000+0300"
                                id="8001"&gt;
                                        &lt;status id="2"
                                        name="No payment status" /&gt;
                                        &lt;creator&gt;
                                                Workspace
                                        &lt;/creator&gt;
                                        &lt;comment /&gt;
                                &lt;/entry&gt;
                        &lt;/orderHistory&gt;
                        &lt;/order&gt;
                        &lt;/orders&gt;
            </String_2>
<!-- Import mode:
                    0 = Insert or update - Insert if the order (id) does
                                           not exist in the database or
                                           update if it does exist
                    1 = Only insert      - Do not update any existing
                                           orders in the database, only
                                           insert new ones
                    2 = Only update      - Update if the order (id) exists
                                           in the database, otherwise do
                                           nothing
-->
             <int_3 xsi:type="xsd:int">0</int_3>
<!-- additiveAnswerUpdate switch:
If true, new customer answers from XML are added to the existing order's
list of answers. If an answer to a questionType already exists, the
```

```
answerText will be updated (replaced). If false, the customer answers
list from XML always replaces the existing order's list. -->
            <boolean_4 xsi:type="xsd:boolean">false</boolean_4>
<!-- additiveBasketUpdate switch:
If true, baskets (order rows) from XML are appended to the existing
order. If false, baskets list from XML always replace the existing
order's baskets. -->
            <boolean_5 xsi:type="xsd:boolean">false</boolean_5>
<!-- invokeReceivedEvents switch:
If true, each new order inserted to Workspace will launch automatic
actions associated with the 'Received' event.
-->
            <boolean_6 xsi:type="xsd:boolean">false</boolean_6>
        </ns1:importOrders>
        <ns0:LoginInfo id="ID1" xsi:type="ns0:LoginInfo">
            <database xsi:type="xsd:string">myshop</database>
            <password xsi:type="xsd:string">password</password>
            <userName xsi:type="xsd:string">user</userName>
        </ns0:LoginInfo>
    </env:Body>
</env:Envelope>
```

Open Interface response (with added comments) to the above order import message states that the order was successfully added to the database. As we hadn't specified an Id number, the order was assigned the next free order Id number (=1).

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xmlns:enc="http://schemas.xmlsoap.org/soap/encoding/"
     xmlns:ns0="http://www.smilehouse.com/types"
xmlns:ns1="http://java.sun.com/jax-rpc-ri/internal"
     env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
     <env:Body>
          <ans1:importOrdersResponse
xmlns:ans1="http://www.smilehouse.com/wsdl">
               <result href="#ID1"/>
          </ans1:importOrdersResponse>
          <ns0:ImportResult id="ID1" xsi:type="ns0:ImportResult">

               <!-- List of Ids of orders that were successfully inserted
in the database.
                    The imported order was assigned an Id number 1. -->
               <insertedIds xsi:type="ns1:linkedList"
enc:arrayType="xsd:anyType[1]">
                    <item xsi:type="enc:long">1</item>
               </insertedIds>

               <removedIds xsi:type="ns1:linkedList"
enc:arrayType="xsd:anyType[0]"/>

               <updatedIds xsi:type="ns1:linkedList"
enc:arrayType="xsd:anyType[0]"/>

               <warnings xsi:type="ns1:linkedList"
enc:arrayType="xsd:anyType[0]"/>

          </ns0:ImportResult>
```

```
    </env:Body>
</env:Envelope>
```

7. ERRORS AND EXCEPTIONS

Errors and exceptions occurring during an Open Interface operation will result in a standard SOAP Fault response. Typical causes for errors are invalid or non-well-formed XML data, invalid datatypes and references to non-existent values.

For example if exportOrders operation is given a parameter to change the status of an exported order to "new status", but such handling or payment status has not been set up in Workspace administration interface, Open Interface will respond with a SOAP Fault as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:enc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:ns0="http://www.smilehouse.com/types"
    xmlns:ns1="http://java.sun.com/jax-rpc-ri/internal"
    env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
    <env:Body>
        <env:Fault xsi:type="env:Fault">
            <faultcode>env:Server</faultcode>

<faultstring>smilehouse.openinterface.OpenInterfaceException</faultstring
>
            <detail>
                <ans1:OpenInterfaceException
xmlns:ans1="http://www.smilehouse.com/wsdl"
                    xsi:type="ns0:OpenInterfaceException">
                    <message xsi:type="xsd:string">No status found with
name 'new status'</message>
                </ans1:OpenInterfaceException>
            </detail>
        </env:Fault>
    </env:Body>
</env:Envelope>
```

8. APPENDIX 1: OPEN INTERFACE XML W3C SCHEMATA

8.1. CUSTOMER XML SCHEMA

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
  <xs:annotation>
    <xs:documentation xml:lang="en">
          Workspace OpenInterface Customers XML W3C Schema v0.93.
          Compatible with Workspace 15+. Last modified: 31.01.2012.
    </xs:documentation>
  </xs:annotation>
```

```
  <xs:element name="customers">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="customer"/>
      </xs:sequence>
      <xs:attribute name="version" use="required" type="xs:double"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="customer">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="statistics" minOccurs="0" maxOccurs="1"/>
        <xs:element ref="customerGroups" minOccurs="0" maxOccurs="1"/>
        <xs:element ref="answers" minOccurs="0" maxOccurs="1"/>
        <xs:element ref="comments" minOccurs="0" maxOccurs="1"/>
      </xs:sequence>
      <xs:attribute name="contactPerson" use="optional"/>
      <xs:attribute name="id" use="optional" type="xs:integer"/>
      <xs:attribute name="login" use="optional" type="xs:string"/>
      <xs:attribute name="password" use="optional" type="xs:string"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="statistics">
    <xs:complexType>

      <xs:attribute name="firstVisit" use="optional">
        <xs:annotation>
          <xs:documentation xml:lang="en">Timestamp format: 2004-12-25
          23:40:56.751+0200</xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="latestVisit" use="optional">
        <xs:annotation>
          <xs:documentation xml:lang="en">Timestamp format: 2004-12-25
          23:40:56.751+0200</xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="sumOfAllOrders" use="optional" type="Money"/>

      <xs:attribute name="averageSum" use="optional" type="Money">
        <xs:annotation>
          <xs:documentation xml:lang="en">
           Average sum of customer's orders
          </xs:documentation>
        </xs:annotation>
      </xs:attribute>

      <xs:attribute name="averageProducts" use="optional"
type="xs:double">
        <xs:annotation>
          <xs:documentation xml:lang="en">
           Average amount of products in the customer's orders
           </xs:documentation>
        </xs:annotation>
      </xs:attribute>

      <xs:attribute name="timesVisited" use="optional"
type="xs:integer"/>

      <xs:attribute name="customerModified" use="optional">
```

```
        <xs:annotation>
          <xs:documentation xml:lang="en">
           Timestamp of latest modification of customer's data by the
            customer. Timestamp format: 2004-12-25 23:40:56.751+0200
          </xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="adminModified" use="optional">
        <xs:annotation>
          <xs:documentation xml:lang="en">
           Timestamp of latest modification of customer's data by the
           webshop admin.
           Timestamp format: 2004-12-25 23:40:56.751+0200
           </xs:documentation>
        </xs:annotation>
      </xs:attribute>

    </xs:complexType>
  </xs:element>
  <xs:element name="customerGroups">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="group" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>

      <xs:attribute name="primary" use="optional" type="xs:string">
        <xs:annotation>
          <xs:documentation xml:lang="en">
           Customer group names must be lowercase.
          </xs:documentation>
        </xs:annotation>
      </xs:attribute>

    </xs:complexType>
  </xs:element>

  <xs:element name="group" type="xs:string">
    <xs:annotation>
      <xs:documentation xml:lang="en">
          Customer group names must be in lowercase.
      </xs:documentation>
    </xs:annotation>
  </xs:element>

  <xs:element name="answers">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="answer"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="answer">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="questionText" minOccurs="0" maxOccurs="1"/>
        <xs:element ref="answerText"/>
      </xs:sequence>

      <xs:attribute name="questionType" use="required" type="xs:integer">
        <xs:annotation>
```

```
          <xs:documentation xml:lang="en">
           Each questionType may occur only once in answers.
           Workspace's built-in questions with their respective
            questionType numbers and descriptions are:

             NAME = 0: Customer's full name or last name.
             STREET = 1: Address/Street.
             POSTAL_CODE = 2: Address/Postal code.
             CITY = 3: Address/Postal office.
             COUNTRY = 4: Address/Country code (a 2 letter ISO 3166-1-
alpha-2 code), see http://www.iso.org/iso/en/prods-
services/iso3166ma/02iso-3166-code-lists/list-en1.html .
             PHONE = 5:  Regular phone number.
             GSM = 6: Cellphone number.
             FAX = 7: Fax number.
             EMAIL = 8:  Email address.
             DIRECT_MARKETING = 9: Direct marketing allowed (possible
values: "yes","no").
             PAYMENT_METHOD = 10: Payment method type.
             DELIVERY_METHOD = 11: Delivery method type.
             FIRST_NAME = 12: Customer's first name.
             STATE  = 13: Address/State (US Customers).
             ZIP = 14: Address/ZIP code (US Customers).
             E_INVOICE_ADDRESS = 15: Email address to send e-invoices to.
             COMPANY = 16: Company name.
             PRICING_CONTRACT_GROUP = 17: The group of pricing contracts.
             COUPON_CODE = 18: Discount coupon code.
             CUSTOMER_THEME = 19: Customer specific shop theme number.
             COMPANY_REGISTRATION_NUMBER = 20: Company's registration
code.


             Workspace's freely configurable questions, questionType
numbers 100 and up:
             USER_QUESTION_1 = 100: Custom question 1.
             USER_QUESTION_2 = 101: Custom question 2.
             USER_QUESTION_3 = 102: Custom question 3.
             ...
             USER_QUESTION_1000 = 999: Custom question 1000  (Unlimited
amount of custom questions).
          </xs:documentation>
        </xs:annotation>
      </xs:attribute>
    </xs:complexType>
  </xs:element>

  <xs:element name="questionText" type="xs:string">
    <xs:annotation>
      <xs:documentation xml:lang="en">
           The question text need not be specified (can be empty) since
           the questionType attribute specifies the question ID number in
           Workspace.
      </xs:documentation>
    </xs:annotation>
  </xs:element>
  <xs:element name="answerText" type="xs:string"/>

  <xs:element name="comments">
    <xs:annotation>
      <xs:documentation xml:lang="en">
```

```
          Webshop keeper's comments on the customer.
      </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" ref="comment"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="comment">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="text" minOccurs="0" maxOccurs="1"/>
      </xs:sequence>
      <xs:attribute name="id" use="optional" type="xs:integer"/>
      <xs:attribute name="creator" use="optional" type="xs:string"/>

      <xs:attribute name="time" use="optional">
        <xs:annotation>
          <xs:documentation xml:lang="en">
          Timestamp format: 2004-12-25 23:40:56.751+0200
          </xs:documentation>
        </xs:annotation>
      </xs:attribute>

    </xs:complexType>
  </xs:element>
  <xs:element name="text" type="xs:string"/>

  <xs:complexType name="Money">
    <xs:simpleContent>
      <xs:extension base="xs:decimal">
        <xs:attribute name="currency" use="optional" type="Currency" />
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>

  <xs:simpleType name="Currency">
    <xs:restriction base="xs:string">
      <xs:length value="3" fixed="true" />
    </xs:restriction>
  </xs:simpleType>


</xs:schema>
```

8.2. ORDER XML SCHEMA

Order export XML format changed from Workspace version 13 to 15 (Order XML Schema v0.98 -> 1.00). The reason was to make interchanges involving monetary values more explicit and thus more accurate, between different integration stakeholders. These changes are not backwards compatible and integrations on the 13 platform need to be changed accordingly when upgrading to the 15+ platform.

- the 0.98 version order attribute `total` was replaced with an element of the same name

- the order attribute `totalVatAmount` was removed

- the order now has an attribute `effectivePricesWithVat` which tells if the prices contain VAT or not

- the payment element attributes `payment/@vat` and `payment/@vatAmount` have been removed, use the element `payment/vatRate` instead. The same applies to delivery/@vat, delivery/@vatAmount , which have been replaced by delivery/vatRate

- The attribute `cost` of both payment and delivery elements has been replaced with an element of its own

- The basket attributes `total`, and `unitPrice` have been moved to child elements

- The basket attributes `totalVatAmount` and `vat` have been removed, use the `vatRate` child element instead

- basket elements have new open field child elements openField6 through 10.

- the elements `order/total`, `order/payment/cost`, `order/delivery/cost`, `order/baskets/basket/unitPrice`, `order/baskets/basket/total` have the following child elements: `withoutVat`, `withVat`, and `vatAmount`

- monetary values in the Order export are of XML Schema type decimal, and they have the attribute 'currency' for the currency code.

Order export XML format changed from Workspace version 1.12 to 13 (Order XML Schema v0.97 -> 0.98);

- Added campaign support with the `campaign` element and `campaign/discountAmount`

- Added the `supplier` element to baskets to refer to the supplier of the product

Order export XML format has changed from Workspace version 1.10 to 1.12 (Order XML Schema v0.96 -> v0.97):

- New attributes order/@followUpCode, order/@cancelled were added.
- New attributes currency/@secondaryCode, currency/@secondaryLabel, currency/@primaryDecimals, currency/@secondaryDecimals, currency/@currencyRelation, currency/@clientFormat, currency/@adminFormat were added.

Order import XML format has changed from Workspace version 1.6 to 1.7 (Order XML Schema v0.94 -> v0.95):

- New attribute currency/@code has the currency code which was previously stored in currency/@label. In v0.95 currency/@label contains the configurable label text of a currency shown in the webshop, which is an optional attribute when importing orders to Workspace.

Order export has changed from Workspace version 1.4 to 1.5 (Order XML Schema v0.92 -> v0.93): *Internal productIDs of products no longer exported, since products are now identified by a unique itemCode*

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
          Workspace OpenInterface Orders XML W3C Schema v0.99.
          Compatible with Workspace "2"+. Last modified: 16.12.2013.
    </xsd:documentation>
  </xsd:annotation>

  <xsd:element name="orders">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="order" maxOccurs="unbounded" type="Order" />
      </xsd:sequence>
      <xsd:attribute name="version" use="required" type="xsd:double" />
    </xsd:complexType>
  </xsd:element>


  <xsd:complexType name="Order">
    <xsd:all>
      <xsd:element name="customerId" minOccurs="0" maxOccurs="1"
type="xsd:integer"/>
      <xsd:element name="total" minOccurs="0" maxOccurs="1"
type="Price"/>
      <xsd:element name="payment" minOccurs="0" maxOccurs="1"
type="Payment"/>
      <xsd:element name="delivery" minOccurs="0" maxOccurs="1"
type="Delivery"/>
      <xsd:element name="handlingStatus" minOccurs="0" maxOccurs="1"
type="HandlingStatus"/>
      <xsd:element name="paymentStatus" minOccurs="0" maxOccurs="1"
type="PaymentStatus"/>
```

```
      <xsd:element name="currency" minOccurs="0" maxOccurs="1"
type="CurrencyFormat"/>
      <xsd:element name="baskets" minOccurs="0" maxOccurs="1"
type="Baskets"/>
      <xsd:element name="answers" minOccurs="0" maxOccurs="1"
type="Answers"/>
      <xsd:element name="orderHistory" minOccurs="0" maxOccurs="1"
type="OrderHistory"/>
    </xsd:all>
    <xsd:attribute name="id" use="required" type="xsd:integer"/>
    <xsd:attribute name="referenceNumber" use="optional"
type="xsd:integer"/>
    <xsd:attribute name="followUpCode" use="optional"
type="xsd:integer"/>
    <xsd:attribute name="cancelled" use="optional" type="xsd:boolean"/>
    <xsd:attribute name="time" use="optional"/>
    <xsd:attribute name="effectivePricesWithVat" use="required"
type="xsd:boolean"/>
  </xsd:complexType>



  <xsd:complexType name="Payment">
    <xsd:all>
      <xsd:element name="vatRate" minOccurs="0" type="xsd:decimal" />
      <xsd:element name="cost" minOccurs="0" type="Price" />
    </xsd:all>
    <xsd:attribute name="code" use="optional" type="xsd:string" />
    <xsd:attribute name="method" use="optional" type="xsd:string">
      <xsd:annotation>
        <xsd:documentation xml:lang="en">
          Payment methods built-in to Workspace:

          "2checkout": 2CheckOut.com payment,
          "alandsbanken": Ålandsbanken payment,
          "cashondelivery": Cash on delivery,
          "chronopay": ChronoPay payment,
          "clickandbuy": ClickAndBuy payment,
          "creditbank": CreditBank payment,
          "dibs": DIBS payment,
          "dksolo": Nordea Denmark payment,
          "ecredit": Nordea eCredit (Nettiluotto) payment,
          "estcard": Estcard payment,
          "estsampo": Estonian Danske Bank payment,
          "estsolo": Estonian Solo payment,
          "eyp": EYP payment,
          "fisonera": Sonera Finland payment,
          "googlecheckout": Google Checkout payment,
          "handelsbanken": Handelsbanken payment,
          "handelsbankensweden": Handelsbanken Sweden payment,
          "hanzanet": Hanza.net (Hansa Pank) payment,
          "invoice": Payment by invoice,
          "itransact": ITransact credit card payment,
          "luottokunta": Luottokunta payment,
          "lvhanzanet": Hansabanka payment,
          "lvsolo": Nordea Latvia payment,
          "nordeaduedate": Nordea payment with due date support,
          "ogone": Ogone payment,
          "okoduedate": Osuuspankki payment with due date support,
```

```
          "osuuspankki": Osuuspankki payment,
          "paypal": PayPal payment,
          "sampo": Danske Bank oyj payment,
          "sesolo": Nordea Sweden payment,
          "solo": Nordea Solo payment,
          "sppop": SP/POP payment,
          "tapiola": Tapiola payment,
          "poppankki": POP Pankki payment
          "aktia": Aktia payment
          "saastopankki": Säästöpankki payment
        </xsd:documentation>
      </xsd:annotation>
    </xsd:attribute>
  </xsd:complexType>


  <xsd:complexType name="Delivery">
    <xsd:all>
      <xsd:element name="vatRate" minOccurs="0" type="xsd:decimal" />
      <xsd:element name="cost" minOccurs="0" type="Price" />
    </xsd:all>
    <xsd:attribute name="code" use="optional" type="xsd:string" />
    <xsd:attribute name="method" use="optional" type="xsd:string" />
  </xsd:complexType>


  <xsd:complexType name="HandlingStatus">
    <xsd:attribute name="id" use="optional" type="xsd:integer" />
    <xsd:attribute name="name" use="required" type="xsd:string" />
  </xsd:complexType>


  <xsd:complexType name="PaymentStatus">
    <xsd:attribute name="id" use="optional" type="xsd:integer" />
    <xsd:attribute name="name" use="required" type="xsd:string" />
  </xsd:complexType>


  <xsd:complexType name="CurrencyFormat">
    <xsd:attribute name="adminFormat" use="optional"
type="xsd:integer" />
    <xsd:attribute name="clientFormat" use="optional"
type="xsd:integer" />
    <xsd:attribute name="code" use="optional" type="xsd:string" />
    <xsd:attribute name="currencyRelation" use="optional"
type="xsd:decimal" />
    <xsd:attribute name="label" use="optional" type="xsd:string" />
    <xsd:attribute name="primaryDecimals" use="optional"
type="xsd:integer" />
    <xsd:attribute name="secondaryCode" use="optional"
type="xsd:string" />
    <xsd:attribute name="secondaryDecimals" use="optional"
type="xsd:integer" />
    <xsd:attribute name="secondaryLabel" use="optional" type="xsd:string"
/>
  </xsd:complexType>


  <xsd:complexType name="Baskets">
    <xsd:sequence>
```

```
      <xsd:element name="basket" minOccurs="1" maxOccurs="unbounded">
        <xsd:complexType>
          <xsd:all>
            <xsd:element name="itemCode" type="xsd:string" />
            <xsd:element name="productName" type="xsd:string" />
            <xsd:element name="vatRate" minOccurs="0"
type="xsd:decimal" />
            <xsd:element name="unitPrice" minOccurs="0" type="Price" />
            <xsd:element name="quantity" minOccurs="0" type="xsd:integer"
/>
            <xsd:element name="total" minOccurs="0" type="Price" />
            <xsd:element name="openField1" minOccurs="0" maxOccurs="1"
type="xsd:string" />
            <xsd:element name="openField2" minOccurs="0" maxOccurs="1"
type="xsd:string" />
            <xsd:element name="openField3" minOccurs="0" maxOccurs="1"
type="xsd:string" />
            <xsd:element name="openField4" minOccurs="0" maxOccurs="1"
type="xsd:string" />
            <xsd:element name="openField5" minOccurs="0" maxOccurs="1"
type="xsd:string" />
            <xsd:element name="openField6" minOccurs="0" maxOccurs="1"
type="xsd:string" />
            <xsd:element name="openField7" minOccurs="0" maxOccurs="1"
type="xsd:string" />
            <xsd:element name="openField8" minOccurs="0" maxOccurs="1"
type="xsd:string" />
            <xsd:element name="openField9" minOccurs="0" maxOccurs="1"
type="xsd:string" />
            <xsd:element name="openField10" minOccurs="0" maxOccurs="1"
type="xsd:string" />
            <xsd:element name="options" minOccurs="0" maxOccurs="1"
type="BasketOptions" />
            <xsd:element name="supplier" minOccurs="0" maxOccurs="1"
type="xsd:string" />
            <xsd:element name="campaign" minOccurs="0" maxOccurs="1"
type="BasketCampaign" />
          </xsd:all>

          <xsd:attribute name="id" use="optional" type="xsd:integer">
            <xsd:annotation>
              <xsd:documentation xml:lang="en">
                    Workspace internal basket ID, not
                    recommended for order import use
              </xsd:documentation>
            </xsd:annotation>
          </xsd:attribute>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="BasketOptions">
    <xsd:sequence>
      <xsd:element name="option" minOccurs="0" maxOccurs="unbounded"
type="xsd:string" />
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="BasketCampaign">
```

```
    <xsd:all>
      <xsd:element name="id" minOccurs="0" maxOccurs="1"
type="xsd:integer" />
      <xsd:element name="discountAmount" minOccurs="0" maxOccurs="1"
type="Price" />
    </xsd:all>
  </xsd:complexType>


  <xsd:complexType name="Answers">
    <xsd:sequence>
      <xsd:element name="answer" maxOccurs="unbounded">
        <xsd:complexType>
          <xsd:all>
            <xsd:element name="questionText" type="xsd:string" />
            <xsd:element name="answerText" type="xsd:string" />
          </xsd:all>

          <xsd:attribute name="questionType" use="required"
type="xsd:integer">
            <xsd:annotation>
              <xsd:documentation xml:lang="en">
           Each questionType may occur only
            once in answers.
           Workspace's built-in questions with their respective
            questionType numbers and descriptions are:

                NAME = 0: Customer's full name or last name.
                STREET = 1: Address/Street.
                POSTAL_CODE = 2: Address/Postal code.
                CITY = 3: Address/Postal office.
                COUNTRY = 4: Address/Country code (a 2 letter ISO 3166-1-
alpha-2 code), see http://www.iso.org/iso/en/prods-
services/iso3166ma/02iso-3166-code-lists/list-en1.html .
                PHONE = 5:  Regular phone number.
                GSM = 6: Cellphone number.
                FAX = 7: Fax number.
                EMAIL = 8:  Email address.
                DIRECT_MARKETING = 9: Direct marketing allowed (possible
values: "yes","no").
                PAYMENT_METHOD = 10: Payment method type.
                DELIVERY_METHOD = 11: Delivery method type.
                FIRST_NAME = 12: Customer's first name.
                STATE  = 13: Address/State (US Customers).
                ZIP = 14: Address/ZIP code (US Customers).
                E_INVOICE_ADDRESS = 15: Email address to send e-invoices
to.
                COMPANY = 16: Company name.
                PRICING_CONTRACT_GROUP = 17: The group of pricing
contracts.
                COUPON_CODE = 18: Discount coupon code.
                CUSTOMER_THEME = 19: Customer specific shop theme number.
                COMPANY_REGISTRATION_NUMBER = 20: Company's registration
code.


                Workspace's freely configurable questions, questionType
numbers 100 and up:
                USER_QUESTION_1 = 100: Custom question 1.
                USER_QUESTION_2 = 101: Custom question 2.
```

```
                USER_QUESTION_3 = 102: Custom question 3.
                ...
                USER_QUESTION_1000 = 999: Custom question 1000
(Unlimited amount of custom questions).
              </xsd:documentation>
            </xsd:annotation>
          </xsd:attribute>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>



  <xsd:complexType name="OrderHistory">
    <xsd:sequence>
      <xsd:element name="entry" maxOccurs="unbounded">
        <xsd:complexType>
          <xsd:all>
            <xsd:element name="status" minOccurs="0" maxOccurs="1">
              <xsd:complexType>
                <xsd:attribute name="id" use="optional"
type="xsd:integer">
                  <xsd:annotation>
                    <xsd:documentation xml:lang="en">
                     The status id may refer to
                     any type of a status in Workspace, i.e. both
handling             status and payment status
                     </xsd:documentation>
                  </xsd:annotation>
                </xsd:attribute>
                <xsd:attribute name="name" use="required" />
              </xsd:complexType>
            </xsd:element>
            <xsd:element name="creator" type="xsd:string" />
            <xsd:element name="comment" minOccurs="0" maxOccurs="1"
type="xsd:string" />
          </xsd:all>
          <xsd:attribute name="id" use="optional" type="xsd:integer" />
          <xsd:attribute name="time" use="required">
            <xsd:annotation>
              <xsd:documentation xml:lang="en">
                     Timestamp format: 2004-12-25 23:40:56.751+0200
              </xsd:documentation>
            </xsd:annotation>
          </xsd:attribute>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="Price">
    <xsd:all>
      <xsd:element name="withoutVat" type="Money" />
      <xsd:element name="withVat" type="Money" />
      <xsd:element name="vatAmount" type="Money" />
    </xsd:all>
  </xsd:complexType>

  <xsd:complexType name="Money">
```

```
    <xsd:simpleContent>
      <xsd:extension base="xsd:decimal">
        <xsd:attribute name="currency" use="optional" type="Currency" />
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>

  <xsd:simpleType name="Currency">
    <xsd:restriction base="xsd:string">
      <xsd:length value="3" fixed="true" />
    </xsd:restriction>
  </xsd:simpleType>

</xsd:schema>
```

8.3. PRODUCT XML SCHEMA

The Product XML Schema changed to version 0.98 from 0.97 in Workspace 15.

Product import API and XML format changed from Workspace version 1.12 to 1.13 (Product XML Schema v0.95 -> v0.97):

- Added element <profit> for profit based price calculation

Product import API and XML format changed from Workspace version 1.9 to 1.12 (Product XML Schema v0.94 -> v0.95):

- Added element <productType> for the product card labeling type

- Added elements <openfield21> to <openField25> for new product open fields from 21 to 25.

Product import API and XML format has changed from Workspace version 1.8.3 to 1.9 (Product XML Schema v0.92 -> v0.94):

- <weight> and <dimensions> elements added

- vatCategory attribute added to <pricing> element. The attribute should contain the name of an existing VAT category. The old vat attribute is also still supported, but it's functionality has changed so, that instead of setting the VAT rate directly it tries to choose VAT category with matching default rate. It is, however, recommended to use vatCategory attribute in import instead.

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:annotation>
    <xsd:documentation xml:lang="en">
          Workspace OpenInterface Products XML W3C Schema v0.98.
          Compatible with Workspace "2"+. Last modified: 01.04.2011.
    </xsd:documentation>
    <xsd:documentation xml:lang="en">
          Note: As W3C XML Schema does not have a &lt;xsd:choice&gt;
          equivalent for attributes, the mandatory selection between
          attributes has been pointed out in the elements' documentation.
    </xsd:documentation>
  </xsd:annotation>

  <xsd:element name="products">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="product" minOccurs="0" maxOccurs="unbounded"
type="Product"/>
      </xsd:sequence>
      <xsd:attribute name="version" use="required" type="xsd:double"/>
    </xsd:complexType>
  </xsd:element>

  <xsd:complexType name="Product">
    <xsd:sequence>
      <xsd:annotation>
        <xsd:documentation xml:lang="en">
          Elements in this sequence after productGroups don't
          necessarily have to appear in the listed order. Due to the W3C
          XML schema limitations with combining xsd:all and
          minOccurs="0", these elements have been specified as a
          sequence.
        </xsd:documentation>
      </xsd:annotation>
      <xsd:element name="name" minOccurs="0" type="xsd:string"/>
      <xsd:element name="productType" minOccurs="0" type="ProductType"/>
      <xsd:element name="productGroups" minOccurs="0" maxOccurs="1"
type="ProductGroups"/>

      <xsd:element name="pricing" minOccurs="0" type="Pricing"/>
      <xsd:element name="orderAmount" minOccurs="0" type="OrderAmount"/>
      <xsd:element name="packaging" minOccurs="0" type="Packaging"/>
      <xsd:element name="supplier" minOccurs="0" type="Supplier"/>
      <xsd:element name="manufacturer" minOccurs="0"
type="Manufacturer"/>
      <xsd:element name="inventory" minOccurs="0" type="Inventory"/>
      <xsd:element name="options" minOccurs="0" type="Options"/>
      <xsd:element name="weight" minOccurs="0" type="xsd:double"/>
      <xsd:element name="dimensions" minOccurs="0" type="Dimensions"/>
      <xsd:element name="title1" minOccurs="0" type="xsd:string"/>
      <xsd:element name="title2" minOccurs="0" type="xsd:string"/>
      <xsd:element name="title3" minOccurs="0" type="xsd:string"/>
      <xsd:element name="title4" minOccurs="0" type="xsd:string"/>
      <xsd:element name="slogan1" minOccurs="0" type="xsd:string"/>
      <xsd:element name="slogan2" minOccurs="0" type="xsd:string"/>
      <xsd:element name="slogan3" minOccurs="0" type="xsd:string"/>
      <xsd:element name="slogan4" minOccurs="0" type="xsd:string"/>
      <xsd:element name="textInfo1" minOccurs="0" type="xsd:string"/>
      <xsd:element name="textInfo2" minOccurs="0" type="xsd:string"/>
      <xsd:element name="textInfo3" minOccurs="0" type="xsd:string"/>
```

```
      <xsd:element name="textInfo4" minOccurs="0" type="xsd:string"/>
      <xsd:element name="picture1" minOccurs="0" type="Picture"/>
      <xsd:element name="picture2" minOccurs="0" type="Picture"/>
      <xsd:element name="picture3" minOccurs="0" type="Picture"/>
      <xsd:element name="picture4" minOccurs="0" type="Picture"/>
      <xsd:element name="picture5" minOccurs="0" type="Picture"/>
      <xsd:element name="file1" minOccurs="0" type="File"/>
      <xsd:element name="file2" minOccurs="0" type="File"/>
      <xsd:element name="file3" minOccurs="0" type="File"/>
      <xsd:element name="file4" minOccurs="0" type="File"/>
      <xsd:element name="file5" minOccurs="0" type="File"/>
      <xsd:element name="openField1" minOccurs="0" type="xsd:string"/>
      <xsd:element name="openField2" minOccurs="0" type="xsd:string"/>
      <xsd:element name="openField3" minOccurs="0" type="xsd:string"/>
      <xsd:element name="openField4" minOccurs="0" type="xsd:string"/>
      <xsd:element name="openField5" minOccurs="0" type="xsd:string"/>
      <xsd:element name="openField6" minOccurs="0" type="xsd:string"/>
      <xsd:element name="openField7" minOccurs="0" type="xsd:string"/>
      <xsd:element name="openField8" minOccurs="0" type="xsd:string"/>
      <xsd:element name="openField9" minOccurs="0" type="xsd:string"/>
      <xsd:element name="openField10" minOccurs="0" type="xsd:string"/>
      <xsd:element name="openField11" minOccurs="0" type="xsd:string"/>
      <xsd:element name="openField12" minOccurs="0" type="xsd:string"/>
      <xsd:element name="openField13" minOccurs="0" type="xsd:string"/>
      <xsd:element name="openField14" minOccurs="0" type="xsd:string"/>
      <xsd:element name="openField15" minOccurs="0" type="xsd:string"/>
      <xsd:element name="openField16" minOccurs="0" type="xsd:string"/>
      <xsd:element name="openField17" minOccurs="0" type="xsd:string"/>
      <xsd:element name="openField18" minOccurs="0" type="xsd:string"/>
      <xsd:element name="openField19" minOccurs="0" type="xsd:string"/>
      <xsd:element name="openField20" minOccurs="0" type="xsd:string"/>
      <xsd:element name="openField21" minOccurs="0" type="xsd:string"/>
      <xsd:element name="openField22" minOccurs="0" type="xsd:string"/>
      <xsd:element name="openField23" minOccurs="0" type="xsd:string"/>
      <xsd:element name="openField24" minOccurs="0" type="xsd:string"/>
      <xsd:element name="openField25" minOccurs="0" type="xsd:string"/>
    </xsd:sequence>

    <xsd:attribute name="id" use="optional" type="xsd:integer">
      <xsd:annotation>
        <xsd:documentation xml:lang="en">
          id is the internal database id and is included in exported XML.
          It is ignored in import.
        </xsd:documentation>
      </xsd:annotation>
    </xsd:attribute>

    <xsd:attribute name="itemCode" use="optional" type="xsd:string">
      <xsd:annotation>
        <xsd:documentation xml:lang="en">
          itemCode is used as a key when updating product data
        </xsd:documentation>
      </xsd:annotation>
    </xsd:attribute>

    <xsd:attribute name="created" use="optional">
      <xsd:annotation>
          <xsd:documentation xml:lang="en">
                    Timestamp format: 2004-12-25 23:40:56.751+0200
          </xsd:documentation>
```

```
      </xsd:annotation>
    </xsd:attribute>

    <xsd:attribute name="lastModified" use="optional">
      <xsd:annotation>
          <xsd:documentation xml:lang="en">
          Timestamp format: 2004-12-25 23:40:56.751+0200
          </xsd:documentation>
      </xsd:annotation>
    </xsd:attribute>

    <xsd:attribute name="modifier" use="optional" type="xsd:string"/>
    <xsd:attribute name="visible" use="optional" type="xsd:boolean"/>
  </xsd:complexType>


  <xsd:complexType name="ProductType">
    <xsd:complexContent>
      <xsd:restriction base="xsd:anyType">
        <xsd:attribute name="code" use="required" type="xsd:string"/>
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>


  <xsd:complexType name="ProductGroups">
    <xsd:sequence>
      <xsd:element name="productGroup" minOccurs="0"
maxOccurs="unbounded">
        <xsd:complexType>
          <xsd:annotation>
            <xsd:documentation xml:lang="en">
                    When importing a new product to Workspace,
                    the productGroup element MUST contain either
                    groupCode or path attribute
           </xsd:documentation>
          </xsd:annotation>
          <xsd:attribute name="id" use="optional" type="xsd:integer"/>
          <xsd:attribute name="groupCode" use="optional"
type="xsd:string"/>
          <xsd:attribute name="path" use="optional" type="xsd:string">
            <xsd:annotation>
              <xsd:documentation xml:lang="en">
                    path must begin with a slash character ('/') and be
                    at least two levels deep, i.e.
                    "/TopLevelGroup/ProductGroupX". If only one level
                    (e.g."/TopLevelGroup") is given, the product will
                    not be visible in Workspace (both administration
                    interface and webshop views).
                    Workspace does not display products
                    directly under the top level productgroup.
              </xsd:documentation>
            </xsd:annotation>
          </xsd:attribute>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
```

```
  <xsd:complexType name="Pricing">
    <xsd:all>
      <xsd:element name="priceGroup1" minOccurs="0" type="PriceGroup"/>
      <xsd:element name="priceGroup2" minOccurs="0" type="PriceGroup"/>
      <xsd:element name="priceGroup3" minOccurs="0" type="PriceGroup"/>
      <xsd:element name="priceGroup4" minOccurs="0" type="PriceGroup"/>
      <xsd:element name="priceGroup5" minOccurs="0" type="PriceGroup"/>
      <xsd:element name="priceGroup6" minOccurs="0" type="PriceGroup"/>
      <xsd:element name="priceGroup7" minOccurs="0" type="PriceGroup"/>
      <xsd:element name="priceGroup8" minOccurs="0" type="PriceGroup"/>
      <xsd:element name="priceGroup9" minOccurs="0" type="PriceGroup"/>
      <xsd:element name="priceGroup10" minOccurs="0" type="PriceGroup"/>
      <xsd:element name="quantityPricing" minOccurs="0"
type="QuantityPricing"/>
    </xsd:all>

    <xsd:attribute name="vat" use="optional" type="xsd:decimal">
      <xsd:annotation>
        <xsd:documentation xml:lang="en">
          vat attribute is deprecated in product import from 0.94
          onwards. Use vatCategory instead. OpenInterface will try to
          select the right VAT category based on the vat attribute for
          backwards compatibility, but it is not as
          reliable as giving the category name in vatCategory attribute.
          NOTE: If vat contains a value which cannot be associated with
          any existing VAT category defined in Workspace,
          import will fail!
        </xsd:documentation>
      </xsd:annotation>
    </xsd:attribute>
    <xsd:attribute name="vatCategory" use="optional" type="xsd:string">
      <xsd:annotation>
        <xsd:documentation xml:lang="en">
          NOTE: If vatCategory contains a value which cannot be
          associated with any existing VAT category defined in Workspace,
          import will fail!
        </xsd:documentation>
      </xsd:annotation>
    </xsd:attribute>
    <xsd:attribute name="priceWithVat" use="optional" type="xsd:decimal">
      <xsd:annotation>
        <xsd:documentation xml:lang="en">
          In product data import it is recommended to set either
          priceWithVat or priceWithoutVat, not both at the same time.
          Workspace will automatically calculate the other value using
          VAT percentage from the specified vatCategory.
        </xsd:documentation>
      </xsd:annotation>
    </xsd:attribute>
    <xsd:attribute name="priceWithoutVat" use="optional"
type="xsd:decimal">
      <xsd:annotation>
        <xsd:documentation xml:lang="en">
          In product data import it is recommended to set either
          priceWithVat or priceWithoutVat, not both at the same time.
          Workspace will automatically calculate the other value using
          VAT percentage from the specified vatCategory.
        </xsd:documentation>
      </xsd:annotation>
    </xsd:attribute>
```

```
    <xsd:attribute name="unitPrice" use="optional" type="xsd:decimal"/>
    <xsd:attribute name="purchasePrice" use="optional"
type="xsd:decimal"/>
    <xsd:attribute name="profitPricing" use="optional" type="xsd:boolean"
/>
    <xsd:attribute name="profit" use="optional" type="xsd:decimal">
      <xsd:annotation>
        <xsd:documentation xml:lang="en">
          Used to calculate price from purchase price if profitPricing is
          true.
        </xsd:documentation>
      </xsd:annotation>
    </xsd:attribute>
  </xsd:complexType>


  <xsd:complexType name="PriceGroup">
      <xsd:attribute name="priceRule" use="optional" type="PriceRule"/>
      <xsd:attribute name="priceWithVat" use="optional"
type="xsd:decimal">
        <xsd:annotation>
          <xsd:documentation xml:lang="en">
           In product data import it is recommended to set either
            priceWithVat or priceWithoutVat, not both at the same time.
           Workspace will automatically calculate the other value using
           VAT percentage from the specified vatCategory.
          </xsd:documentation>
        </xsd:annotation>
      </xsd:attribute>
      <xsd:attribute name="priceWithoutVat" use="optional"
type="xsd:decimal">
        <xsd:annotation>
          <xsd:documentation xml:lang="en">
           In product data import it is recommended to set either
            priceWithVat or priceWithoutVat, not both at the same time.
           Workspace will automatically calculate the other value using
           VAT percentage from the specified vatCategory.
          </xsd:documentation>
        </xsd:annotation>
      </xsd:attribute>
  </xsd:complexType>


  <xsd:complexType name="QuantityPricing">
    <xsd:sequence>
      <xsd:element name="rule" minOccurs="0" maxOccurs="unbounded">
        <xsd:complexType>
          <xsd:attribute name="threshold" use="required"
type="xsd:integer"/>
          <xsd:attribute name="priceRule" use="required"
type="PriceRule"/>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="OrderAmount">
    <xsd:attribute name="default" use="optional" type="xsd:integer"/>
    <xsd:attribute name="maximum" use="optional" type="xsd:integer"/>
    <xsd:attribute name="minimum" use="optional" type="xsd:integer"/>
```

```
    <xsd:attribute name="amountFactor" use="optional"
type="xsd:integer"/>
  </xsd:complexType>

  <xsd:complexType name="Packaging">
    <xsd:attribute name="packetSize" use="optional" type="xsd:integer"/>
    <xsd:attribute name="packetType" use="optional" type="xsd:string"/>
    <xsd:attribute name="unitType" use="optional" type="xsd:string"/>
  </xsd:complexType>

  <xsd:complexType name="Supplier">
    <xsd:attribute name="id" use="optional" type="xsd:string"/>
    <xsd:attribute name="itemCode" use="optional" type="xsd:string"/>
    <xsd:attribute name="name" use="optional" type="xsd:string"/>
  </xsd:complexType>

  <xsd:complexType name="Manufacturer">
    <xsd:attribute name="id" use="optional" type="xsd:string"/>
    <xsd:attribute name="itemCode" use="optional" type="xsd:string"/>
    <xsd:attribute name="name" use="optional" type="xsd:string"/>
  </xsd:complexType>

  <xsd:complexType name="Inventory">
    <xsd:all>
      <xsd:element name="section" minOccurs="0" type="xsd:string"/>
      <xsd:element name="shelf" minOccurs="0" type="xsd:string"/>
      <xsd:element name="alarm" minOccurs="0">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="text" minOccurs="0" type="xsd:string"/>
          </xsd:sequence>
          <xsd:attribute name="limit" use="optional" type="xsd:integer"/>
        </xsd:complexType>
      </xsd:element>
    </xsd:all>
    <xsd:attribute name="amount" use="optional" type="xsd:integer"/>
    <xsd:attribute name="inventoryName" type="xsd:string"/>
  </xsd:complexType>


  <xsd:complexType name="Dimensions">
    <xsd:all>
      <xsd:element name="width" minOccurs="0" type="xsd:double"/>
      <xsd:element name="height" minOccurs="0" type="xsd:double"/>
      <xsd:element name="depth" minOccurs="0" type="xsd:double"/>
    </xsd:all>
  </xsd:complexType>


  <xsd:complexType name="Options">
    <xsd:sequence>
      <xsd:element name="option" maxOccurs="unbounded" minOccurs="0">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="choices" minOccurs="0">
              <xsd:complexType>
                <xsd:sequence>
                  <xsd:element name="choice" maxOccurs="unbounded"
minOccurs="0">
                    <xsd:complexType>
```

```
                        <xsd:attribute name="amount" use="optional"
type="xsd:integer"/>
                        <xsd:attribute name="id" use="optional"
type="xsd:integer"/>
                        <xsd:attribute name="itemCode" use="optional"
type="xsd:string"/>
                        <xsd:attribute name="priceRule" use="optional"
type="PriceRule"/>
                        <xsd:attribute name="value" use="required"
type="xsd:string"/>
                      </xsd:complexType>
                    </xsd:element>
                  </xsd:sequence>
                </xsd:complexType>
              </xsd:element>
            </xsd:sequence>
            <xsd:attribute name="id" use="optional" type="xsd:integer"/>
            <xsd:attribute name="name" use="optional" type="xsd:string"/>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>



    <xsd:complexType name="Picture">
      <xsd:attribute name="alt" use="optional" type="xsd:string"/>

      <xsd:attribute name="upload" use="optional" type="xsd:string">
        <xsd:annotation>
            <xsd:documentation xml:lang="en">
            Local filename of a picture file uploaded to Workspace.
            Alternative to the url attribute.
            </xsd:documentation>
        </xsd:annotation>
      </xsd:attribute>

      <xsd:attribute name="url" use="optional" type="xsd:anyURI">
        <xsd:annotation>
          <xsd:documentation xml:lang="en">
            URL or absolute path+filename of a picture file.
           Alternative to the upload attribute.
          </xsd:documentation>
        </xsd:annotation>
      </xsd:attribute>
    </xsd:complexType>


    <xsd:complexType name="File">
      <xsd:attribute name="linkText" use="optional" type="xsd:string"/>

      <xsd:attribute name="upload" use="optional" type="xsd:string">
        <xsd:annotation>
          <xsd:documentation xml:lang="en">
            Local filename of a file uploaded to Workspace. Alternative to
            the url attribute.
          </xsd:documentation>
        </xsd:annotation>
      </xsd:attribute>
```

```
    <xsd:attribute name="url" use="optional" type="xsd:anyURI">
      <xsd:annotation>
        <xsd:documentation xml:lang="en">
          URL or absolute path+filename of a file. Alternative to
          the upload attribute.
        </xsd:documentation>
      </xsd:annotation>
    </xsd:attribute>
  </xsd:complexType>


  <xsd:simpleType name="PriceRule">
    <xsd:annotation>
      <xsd:documentation xml:lang="en">
          priceRule consists of optional '+' or '-' prefix followed
           by decimal number (xsd:double) and optionally a percent ('%')
          sign.
      </xsd:documentation>
    </xsd:annotation>
    <xsd:restriction base="xsd:string"/>
  </xsd:simpleType>


</xsd:schema>
```

8.4. PRICING CONTRACTS XML SCHEMA

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
  <xs:annotation>
    <xs:documentation xml:lang="en">
          Workspace OpenInterface Price list XML W3C Schema v0.9.
          Compatible with Workspace 1.7+. Last modified: 15.02.2008.
          </xs:documentation>
  </xs:annotation>
  <xs:element name="pricecontracts">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="0" maxOccurs="unbounded"
ref="pricecontract"/>
      </xs:sequence>
      <xs:attribute name="version" use="required"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="pricecontract">
    <xs:complexType>

      <xs:annotation>
        <xs:documentation xml:lang="en">
          In determining the price for a customer, the first match is
          used: customer id, price list id, customer group id. So only
          one is taken into consideration.
        </xs:documentation>
      </xs:annotation>
      <xs:attribute name="customerGroupId" use="optional"/>
      <xs:attribute name="customerId" use="optional"/>
      <xs:attribute name="externalPricelistId" use="optional"/>
```

```
      <xs:attribute name="itemCode" use="required"/>
      <xs:attribute name="pricerule" use="required"/>

    </xs:complexType>
  </xs:element>
</xs:schema>
```

9. APPENDIX 2: WSDL DEFINITIONS

9.1. OPENINTERFACEADDRESS WSDL

```
<?xml version="1.0" encoding="UTF-8"?><definitions
xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:tns="http://www.smilehouse.com/wsdl"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
name="OpenInterfaceAddress"
targetNamespace="http://www.smilehouse.com/wsdl">
  <types/>
  <message name="OpenInterfaceAddressIF_getOpenInterfaceAddress">
    <part name="String_1" type="xsd:string"/></message>
  <message name="OpenInterfaceAddressIF_getOpenInterfaceAddressResponse">
    <part name="result" type="xsd:string"/></message>
  <portType name="OpenInterfaceAddressIF">
    <operation name="getOpenInterfaceAddress" parameterOrder="String_1">
      <input
message="tns:OpenInterfaceAddressIF_getOpenInterfaceAddress"/>
      <output
message="tns:OpenInterfaceAddressIF_getOpenInterfaceAddressResponse"/></o
peration></portType>
  <binding name="OpenInterfaceAddressIFBinding"
type="tns:OpenInterfaceAddressIF">
    <operation name="getOpenInterfaceAddress">
      <input>
        <soap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="http://www.smilehouse.com/wsdl"/></input>
      <output>
        <soap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="http://www.smilehouse.com/wsdl"/></output>
      <soap:operation soapAction=""/></operation>
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
style="rpc"/></binding>
  <service name="OpenInterfaceAddress">
    <port name="OpenInterfaceAddressIFPort"
binding="tns:OpenInterfaceAddressIFBinding">
      <soap:address xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
location="http://host:port/workspace.admin/openinterfaceaddress"/>
</port>
</service>
</definitions>
```

9.2. OPENINTERFACE WSDL

The WSDL 1.1 definition below can also be retrieved directly from a Workspace installation by getting contents of URL (constructed from the response of OpenInterfaceAddress service):

http://host:port/workspace.admin_<licensetype>_<versionnumber>/openinterface?WSDL

```
<?xml version="1.0" encoding="UTF-8"?><definitions
xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:tns="http://www.smilehouse.com/wsdl"
xmlns:ns2="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:ns3="http://www.smilehouse.com/types" name="OpenInterface"
targetNamespace="http://www.smilehouse.com/wsdl">
  <types>
    <schema xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:tns="http://www.smilehouse.com/types"
xmlns:ns2="http://java.sun.com/jax-rpc-ri/internal" xmlns:soap11-
enc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
targetNamespace="http://www.smilehouse.com/types">
      <import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
      <import namespace="http://java.sun.com/jax-rpc-ri/internal"/>
      <complexType name="LoginInfo">
        <sequence>
          <element name="database" type="string"/>
          <element name="password" type="string"/>
          <element name="userName"
type="string"/></sequence></complexType>
      <complexType name="IteratorClosedException">
        <complexContent>
          <extension base="tns:OpenInterfaceException">
            <sequence/></extension></complexContent></complexType>
      <complexType name="OpenInterfaceException">
        <sequence>
          <element name="message" type="string"
nillable="true"/></sequence></complexType>
      <complexType name="AccessDeniedException">
        <complexContent>
          <extension base="tns:OpenInterfaceException">
            <sequence/></extension></complexContent></complexType>
      <complexType name="CustomerCriteria">
        <sequence>
          <element name="adminModifiedAfter" type="dateTime"/>
          <element name="adminModifiedBefore" type="dateTime"/>
          <element name="customerGroup" type="string"/>
          <element name="customerId" type="string"/>
          <element name="customerModifiedAfter" type="dateTime"/>
          <element name="customerModifiedBefore" type="dateTime"/>
          <element name="firstVisitDateAfter" type="dateTime"/>
          <element name="firstVisitDateBefore" type="dateTime"/>
          <element name="idGreaterThan" type="string"/>
          <element name="idIn" type="tns:ArrayOfstring"/>
```

```
          <element name="idLessThan" type="string"/>
          <element name="lastVisitDateAfter" type="dateTime"/>
          <element name="lastVisitDateBefore" type="dateTime"/>
          <element name="modifyOperation" type="string"/>
          <element name="primaryCustomerGroup"
type="string"/></sequence></complexType>
      <complexType name="ArrayOfstring">
        <complexContent>
          <restriction base="soap11-enc:Array">
            <attribute ref="soap11-enc:arrayType"
wsdl:arrayType="string[]"/></restriction></complexContent></complexType>
      <complexType name="ExportResult">
        <sequence>
          <element name="lastUpdateLogId" type="soap11-enc:long"/>
          <element name="resultSize" type="int"/>
          <element name="xml" type="string"/></sequence></complexType>
      <complexType name="OrderCriteria">
        <sequence>
          <element name="customerIdIn" type="tns:ArrayOfstring"/>
          <element name="dateAfter" type="dateTime"/>
          <element name="dateBefore" type="dateTime"/>
          <element name="handlingStatusNameIn" type="tns:ArrayOfstring"/>
          <element name="handlingStatusNameNotIn"
type="tns:ArrayOfstring"/>
          <element name="idGreaterThan" type="soap11-enc:long"/>
          <element name="idIn" type="tns:ArrayOfLong"/>
          <element name="idLessThan" type="soap11-enc:long"/>
          <element name="paymentStatusNameIn" type="tns:ArrayOfstring"/>
          <element name="paymentStatusNameNotIn"
type="tns:ArrayOfstring"/>
          <element name="sortRules"
type="tns:ArrayOfOrderCriteriaSortRule"/>
          <element name="sumGreaterThan" type="soap11-enc:double"/>
          <element name="sumLessThan" type="soap11-
enc:double"/></sequence></complexType>
      <complexType name="ArrayOfLong">
        <complexContent>
          <restriction base="soap11-enc:Array">
            <attribute ref="soap11-enc:arrayType" wsdl:arrayType="soap11-
enc:long[]"/></restriction></complexContent></complexType>
      <complexType name="ArrayOfOrderCriteriaSortRule">
        <complexContent>
          <restriction base="soap11-enc:Array">
            <attribute ref="soap11-enc:arrayType"
wsdl:arrayType="tns:OrderCriteriaSortRule[]"/></restriction></complexCont
ent></complexType>
      <complexType name="OrderCriteriaSortRule">
        <sequence>
          <element name="asc" type="boolean"/>
          <element name="property"
type="string"/></sequence></complexType>
      <complexType name="ImportResult">
        <sequence>
          <element name="insertedIds" type="ns2:list"/>
          <element name="removedIds" type="ns2:list"/>
          <element name="updatedIds" type="ns2:list"/>
          <element name="warnings"
type="ns2:list"/></sequence></complexType></schema>
    <schema xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:tns="http://java.sun.com/jax-rpc-ri/internal" xmlns:soap11-
```

```
enc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
targetNamespace="http://java.sun.com/jax-rpc-ri/internal">
      <import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
      <import namespace="http://www.smilehouse.com/types"/>
      <complexType name="list">
        <complexContent>
          <extension base="tns:collection">
            <sequence/></extension></complexContent></complexType>
      <complexType name="collection">
        <complexContent>
          <restriction base="soap11-enc:Array">
            <attribute ref="soap11-enc:arrayType"
wsdl:arrayType="anyType[]"/></restriction></complexContent></complexType>
</schema></types>
  <message name="OpenInterfaceIF_closeIterator">
    <part name="LoginInfo_1" type="ns3:LoginInfo"/>
    <part name="String_2" type="xsd:string"/></message>
  <message name="OpenInterfaceIF_closeIteratorResponse"/>
  <message name="AccessDeniedException">
    <part name="AccessDeniedException"
type="ns3:AccessDeniedException"/></message>
  <message name="OpenInterfaceException">
    <part name="OpenInterfaceException"
type="ns3:OpenInterfaceException"/></message>
  <message name="OpenInterfaceIF_exportCustomers">
    <part name="LoginInfo_1" type="ns3:LoginInfo"/>
    <part name="CustomerCriteria_2"
type="ns3:CustomerCriteria"/></message>
  <message name="OpenInterfaceIF_exportCustomersResponse">
    <part name="result" type="ns3:ExportResult"/></message>
  <message name="OpenInterfaceIF_exportOrderXML">
    <part name="String_1" type="xsd:string"/>
    <part name="String_2" type="xsd:string"/>
    <part name="String_3" type="xsd:string"/>
    <part name="OrderCriteria_4" type="ns3:OrderCriteria"/>
    <part name="Long_5" type="ns2:long"/></message>
  <message name="OpenInterfaceIF_exportOrderXMLResponse">
    <part name="result" type="ns3:ExportResult"/></message>
  <message name="OpenInterfaceIF_exportOrders">
    <part name="LoginInfo_1" type="ns3:LoginInfo"/>
    <part name="OrderCriteria_2" type="ns3:OrderCriteria"/>
    <part name="Long_3" type="ns2:long"/>
    <part name="String_4" type="xsd:string"/></message>
  <message name="OpenInterfaceIF_exportOrdersResponse">
    <part name="result" type="ns3:ExportResult"/></message>
  <message name="OpenInterfaceIF_exportOrders2">
    <part name="LoginInfo_1" type="ns3:LoginInfo"/>
    <part name="OrderCriteria_2" type="ns3:OrderCriteria"/>
    <part name="String_3" type="xsd:string"/></message>
  <message name="OpenInterfaceIF_exportOrders2Response">
    <part name="result" type="ns3:ExportResult"/></message>
  <message name="OpenInterfaceIF_getVersion"/>
  <message name="OpenInterfaceIF_getVersionResponse">
    <part name="result" type="xsd:double"/></message>
  <message name="OpenInterfaceIF_importCustomers">
    <part name="LoginInfo_1" type="ns3:LoginInfo"/>
    <part name="String_2" type="xsd:string"/>
    <part name="int_3" type="xsd:int"/>
```

```
    <part name="boolean_4" type="xsd:boolean"/></message>
  <message name="OpenInterfaceIF_importCustomersResponse">
    <part name="result" type="ns3:ImportResult"/></message>
  <message name="OpenInterfaceIF_importOrders">
    <part name="LoginInfo_1" type="ns3:LoginInfo"/>
    <part name="String_2" type="xsd:string"/>
    <part name="int_3" type="xsd:int"/>
    <part name="boolean_4" type="xsd:boolean"/>
    <part name="boolean_5" type="xsd:boolean"/>
    <part name="boolean_6" type="xsd:boolean"/></message>
  <message name="OpenInterfaceIF_importOrdersResponse">
    <part name="result" type="ns3:ImportResult"/></message>
  <message name="OpenInterfaceIF_importPricelist">
    <part name="LoginInfo_1" type="ns3:LoginInfo"/>
    <part name="String_2" type="xsd:string"/>
    <part name="int_3" type="xsd:int"/></message>
  <message name="OpenInterfaceIF_importPricelistResponse">
    <part name="result" type="ns3:ImportResult"/></message>
  <message name="OpenInterfaceIF_importProducts">
    <part name="LoginInfo_1" type="ns3:LoginInfo"/>
    <part name="String_2" type="xsd:string"/>
    <part name="int_3" type="xsd:int"/>
    <part name="arrayOfString_4" type="ns3:ArrayOfstring"/>
    <part name="boolean_5" type="xsd:boolean"/>
    <part name="boolean_6" type="xsd:boolean"/></message>
  <message name="OpenInterfaceIF_importProductsResponse">
    <part name="result" type="ns3:ImportResult"/></message>
  <message name="OpenInterfaceIF_importProducts2">
    <part name="LoginInfo_1" type="ns3:LoginInfo"/>
    <part name="String_2" type="xsd:string"/>
    <part name="int_3" type="xsd:int"/>
    <part name="boolean_4" type="xsd:boolean"/>
    <part name="boolean_5" type="xsd:boolean"/>
    <part name="arrayOfString_6" type="ns3:ArrayOfstring"/>
    <part name="boolean_7" type="xsd:boolean"/></message>
  <message name="OpenInterfaceIF_importProducts2Response">
    <part name="result" type="ns3:ImportResult"/></message>
  <message name="OpenInterfaceIF_invokeEvent">
    <part name="LoginInfo_1" type="ns3:LoginInfo"/>
    <part name="String_2" type="xsd:string"/>
    <part name="Object_3" type="xsd:anyType"/></message>
  <message name="OpenInterfaceIF_invokeEventResponse"/>
  <message name="OpenInterfaceIF_iterate">
    <part name="LoginInfo_1" type="ns3:LoginInfo"/>
    <part name="String_2" type="xsd:string"/>
    <part name="int_3" type="xsd:int"/></message>
  <message name="OpenInterfaceIF_iterateResponse">
    <part name="result" type="ns3:ExportResult"/></message>
  <message name="IteratorClosedException">
    <part name="IteratorClosedException"
type="ns3:IteratorClosedException"/></message>
  <message name="OpenInterfaceIF_mailSending">
    <part name="LoginInfo_1" type="ns3:LoginInfo"/>
    <part name="String_2" type="xsd:string"/>
    <part name="String_3" type="xsd:string"/>
    <part name="String_4" type="xsd:string"/>
    <part name="String_5" type="xsd:string"/></message>
  <message name="OpenInterfaceIF_mailSendingResponse">
    <part name="result" type="xsd:string"/></message>
  <message name="OpenInterfaceIF_openHQLIterator">
```

```
    <part name="LoginInfo_1" type="ns3:LoginInfo"/>
    <part name="String_2" type="xsd:string"/>
    <part name="long_3" type="xsd:long"/></message>
  <message name="OpenInterfaceIF_openHQLIteratorResponse">
    <part name="result" type="xsd:string"/></message>
  <portType name="OpenInterfaceIF">
    <operation name="closeIterator" parameterOrder="LoginInfo_1
String_2">
      <input message="tns:OpenInterfaceIF_closeIterator"/>
      <output message="tns:OpenInterfaceIF_closeIteratorResponse"/>
      <fault name="AccessDeniedException"
message="tns:AccessDeniedException"/>
      <fault name="OpenInterfaceException"
message="tns:OpenInterfaceException"/></operation>
    <operation name="exportCustomers" parameterOrder="LoginInfo_1
CustomerCriteria_2">
      <input message="tns:OpenInterfaceIF_exportCustomers"/>
      <output message="tns:OpenInterfaceIF_exportCustomersResponse"/>
      <fault name="AccessDeniedException"
message="tns:AccessDeniedException"/>
      <fault name="OpenInterfaceException"
message="tns:OpenInterfaceException"/></operation>
    <operation name="exportOrderXML" parameterOrder="String_1 String_2
String_3 OrderCriteria_4 Long_5">
      <input message="tns:OpenInterfaceIF_exportOrderXML"/>
      <output message="tns:OpenInterfaceIF_exportOrderXMLResponse"/>
      <fault name="AccessDeniedException"
message="tns:AccessDeniedException"/>
      <fault name="OpenInterfaceException"
message="tns:OpenInterfaceException"/></operation>
    <operation name="exportOrders" parameterOrder="LoginInfo_1
OrderCriteria_2 Long_3 String_4">
      <input message="tns:OpenInterfaceIF_exportOrders"/>
      <output message="tns:OpenInterfaceIF_exportOrdersResponse"/>
      <fault name="AccessDeniedException"
message="tns:AccessDeniedException"/>
      <fault name="OpenInterfaceException"
message="tns:OpenInterfaceException"/></operation>
    <operation name="exportOrders2" parameterOrder="LoginInfo_1
OrderCriteria_2 String_3">
      <input message="tns:OpenInterfaceIF_exportOrders2"/>
      <output message="tns:OpenInterfaceIF_exportOrders2Response"/>
      <fault name="AccessDeniedException"
message="tns:AccessDeniedException"/>
      <fault name="OpenInterfaceException"
message="tns:OpenInterfaceException"/></operation>
    <operation name="getVersion" parameterOrder="">
      <input message="tns:OpenInterfaceIF_getVersion"/>
      <output
message="tns:OpenInterfaceIF_getVersionResponse"/></operation>
    <operation name="importCustomers" parameterOrder="LoginInfo_1
String_2 int_3 boolean_4">
      <input message="tns:OpenInterfaceIF_importCustomers"/>
      <output message="tns:OpenInterfaceIF_importCustomersResponse"/>
      <fault name="AccessDeniedException"
message="tns:AccessDeniedException"/>
      <fault name="OpenInterfaceException"
message="tns:OpenInterfaceException"/></operation>
    <operation name="importOrders" parameterOrder="LoginInfo_1 String_2
int_3 boolean_4 boolean_5 boolean_6">
```

```
      <input message="tns:OpenInterfaceIF_importOrders"/>
      <output message="tns:OpenInterfaceIF_importOrdersResponse"/>
      <fault name="AccessDeniedException"
message="tns:AccessDeniedException"/>
      <fault name="OpenInterfaceException"
message="tns:OpenInterfaceException"/></operation>
    <operation name="importPricelist" parameterOrder="LoginInfo_1
String_2 int_3">
      <input message="tns:OpenInterfaceIF_importPricelist"/>
      <output message="tns:OpenInterfaceIF_importPricelistResponse"/>
      <fault name="AccessDeniedException"
message="tns:AccessDeniedException"/>
      <fault name="OpenInterfaceException"
message="tns:OpenInterfaceException"/></operation>
    <operation name="importProducts" parameterOrder="LoginInfo_1 String_2
int_3 arrayOfString_4 boolean_5 boolean_6">
      <input message="tns:OpenInterfaceIF_importProducts"/>
      <output message="tns:OpenInterfaceIF_importProductsResponse"/>
      <fault name="AccessDeniedException"
message="tns:AccessDeniedException"/>
      <fault name="OpenInterfaceException"
message="tns:OpenInterfaceException"/></operation>
    <operation name="importProducts2" parameterOrder="LoginInfo_1
String_2 int_3 boolean_4 boolean_5 arrayOfString_6 boolean_7">
      <input message="tns:OpenInterfaceIF_importProducts2"/>
      <output message="tns:OpenInterfaceIF_importProducts2Response"/>
      <fault name="AccessDeniedException"
message="tns:AccessDeniedException"/>
      <fault name="OpenInterfaceException"
message="tns:OpenInterfaceException"/></operation>
    <operation name="invokeEvent" parameterOrder="LoginInfo_1 String_2
Object_3">
      <input message="tns:OpenInterfaceIF_invokeEvent"/>
      <output message="tns:OpenInterfaceIF_invokeEventResponse"/>
      <fault name="AccessDeniedException"
message="tns:AccessDeniedException"/>
      <fault name="OpenInterfaceException"
message="tns:OpenInterfaceException"/></operation>
    <operation name="iterate" parameterOrder="LoginInfo_1 String_2
int_3">
      <input message="tns:OpenInterfaceIF_iterate"/>
      <output message="tns:OpenInterfaceIF_iterateResponse"/>
      <fault name="AccessDeniedException"
message="tns:AccessDeniedException"/>
      <fault name="IteratorClosedException"
message="tns:IteratorClosedException"/>
      <fault name="OpenInterfaceException"
message="tns:OpenInterfaceException"/></operation>
    <operation name="mailSending" parameterOrder="LoginInfo_1 String_2
String_3 String_4 String_5">
      <input message="tns:OpenInterfaceIF_mailSending"/>
      <output message="tns:OpenInterfaceIF_mailSendingResponse"/>
      <fault name="OpenInterfaceException"
message="tns:OpenInterfaceException"/></operation>
    <operation name="openHQLIterator" parameterOrder="LoginInfo_1
String_2 long_3">
      <input message="tns:OpenInterfaceIF_openHQLIterator"/>
      <output message="tns:OpenInterfaceIF_openHQLIteratorResponse"/>
      <fault name="AccessDeniedException"
message="tns:AccessDeniedException"/>
```

```
      <fault name="OpenInterfaceException"
message="tns:OpenInterfaceException"/></operation></portType>
  <binding name="OpenInterfaceIFBinding" type="tns:OpenInterfaceIF">
    <operation name="closeIterator">
      <input>
        <soap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="http://www.smilehouse.com/wsdl"/></input>
      <output>
        <soap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="http://www.smilehouse.com/wsdl"/></output>
      <fault name="AccessDeniedException">
        <soap:fault name="AccessDeniedException"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="http://www.smilehouse.com/wsdl"/></fault>
      <fault name="OpenInterfaceException">
        <soap:fault name="OpenInterfaceException"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="http://www.smilehouse.com/wsdl"/></fault>
      <soap:operation soapAction=""/></operation>
    <operation name="exportCustomers">
      <input>
        <soap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="http://www.smilehouse.com/wsdl"/></input>
      <output>
        <soap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="http://www.smilehouse.com/wsdl"/></output>
      <fault name="AccessDeniedException">
        <soap:fault name="AccessDeniedException"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="http://www.smilehouse.com/wsdl"/></fault>
      <fault name="OpenInterfaceException">
        <soap:fault name="OpenInterfaceException"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="http://www.smilehouse.com/wsdl"/></fault>
      <soap:operation soapAction=""/></operation>
    <operation name="exportOrderXML">
      <input>
        <soap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="http://www.smilehouse.com/wsdl"/></input>
      <output>
        <soap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="http://www.smilehouse.com/wsdl"/></output>
      <fault name="AccessDeniedException">
        <soap:fault name="AccessDeniedException"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="http://www.smilehouse.com/wsdl"/></fault>
      <fault name="OpenInterfaceException">
        <soap:fault name="OpenInterfaceException"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="http://www.smilehouse.com/wsdl"/></fault>
      <soap:operation soapAction=""/></operation>
    <operation name="exportOrders">
      <input>
```

```
        <soap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="http://www.smilehouse.com/wsdl"/></input>
      <output>
        <soap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="http://www.smilehouse.com/wsdl"/></output>
      <fault name="AccessDeniedException">
        <soap:fault name="AccessDeniedException"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="http://www.smilehouse.com/wsdl"/></fault>
      <fault name="OpenInterfaceException">
        <soap:fault name="OpenInterfaceException"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="http://www.smilehouse.com/wsdl"/></fault>
      <soap:operation soapAction=""/></operation>
    <operation name="exportOrders2">
      <input>
        <soap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="http://www.smilehouse.com/wsdl"/></input>
      <output>
        <soap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="http://www.smilehouse.com/wsdl"/></output>
      <fault name="AccessDeniedException">
        <soap:fault name="AccessDeniedException"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="http://www.smilehouse.com/wsdl"/></fault>
      <fault name="OpenInterfaceException">
        <soap:fault name="OpenInterfaceException"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="http://www.smilehouse.com/wsdl"/></fault>
      <soap:operation soapAction=""/></operation>
    <operation name="getVersion">
      <input>
        <soap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="http://www.smilehouse.com/wsdl"/></input>
      <output>
        <soap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="http://www.smilehouse.com/wsdl"/></output>
      <soap:operation soapAction=""/></operation>
    <operation name="importCustomers">
      <input>
        <soap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="http://www.smilehouse.com/wsdl"/></input>
      <output>
        <soap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="http://www.smilehouse.com/wsdl"/></output>
      <fault name="AccessDeniedException">
        <soap:fault name="AccessDeniedException"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="http://www.smilehouse.com/wsdl"/></fault>
      <fault name="OpenInterfaceException">
```

```
        <soap:fault name="OpenInterfaceException"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="http://www.smilehouse.com/wsdl"/></fault>
      <soap:operation soapAction=""/></operation>
    <operation name="importOrders">
      <input>
        <soap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="http://www.smilehouse.com/wsdl"/></input>
      <output>
        <soap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="http://www.smilehouse.com/wsdl"/></output>
      <fault name="AccessDeniedException">
        <soap:fault name="AccessDeniedException"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="http://www.smilehouse.com/wsdl"/></fault>
      <fault name="OpenInterfaceException">
        <soap:fault name="OpenInterfaceException"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="http://www.smilehouse.com/wsdl"/></fault>
      <soap:operation soapAction=""/></operation>
    <operation name="importPricelist">
      <input>
        <soap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="http://www.smilehouse.com/wsdl"/></input>
      <output>
        <soap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="http://www.smilehouse.com/wsdl"/></output>
      <fault name="AccessDeniedException">
        <soap:fault name="AccessDeniedException"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="http://www.smilehouse.com/wsdl"/></fault>
      <fault name="OpenInterfaceException">
        <soap:fault name="OpenInterfaceException"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="http://www.smilehouse.com/wsdl"/></fault>
      <soap:operation soapAction=""/></operation>
    <operation name="importProducts">
      <input>
        <soap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="http://www.smilehouse.com/wsdl"/></input>
      <output>
        <soap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="http://www.smilehouse.com/wsdl"/></output>
      <fault name="AccessDeniedException">
        <soap:fault name="AccessDeniedException"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="http://www.smilehouse.com/wsdl"/></fault>
      <fault name="OpenInterfaceException">
        <soap:fault name="OpenInterfaceException"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="http://www.smilehouse.com/wsdl"/></fault>
      <soap:operation soapAction=""/></operation>
    <operation name="importProducts2">
      <input>
```

```
        <soap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="http://www.smilehouse.com/wsdl"/></input>
      <output>
        <soap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="http://www.smilehouse.com/wsdl"/></output>
      <fault name="AccessDeniedException">
        <soap:fault name="AccessDeniedException"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="http://www.smilehouse.com/wsdl"/></fault>
      <fault name="OpenInterfaceException">
        <soap:fault name="OpenInterfaceException"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="http://www.smilehouse.com/wsdl"/></fault>
      <soap:operation soapAction=""/></operation>
    <operation name="invokeEvent">
      <input>
        <soap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="http://www.smilehouse.com/wsdl"/></input>
      <output>
        <soap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="http://www.smilehouse.com/wsdl"/></output>
      <fault name="AccessDeniedException">
        <soap:fault name="AccessDeniedException"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="http://www.smilehouse.com/wsdl"/></fault>
      <fault name="OpenInterfaceException">
        <soap:fault name="OpenInterfaceException"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="http://www.smilehouse.com/wsdl"/></fault>
      <soap:operation soapAction=""/></operation>
    <operation name="iterate">
      <input>
        <soap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="http://www.smilehouse.com/wsdl"/></input>
      <output>
        <soap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="http://www.smilehouse.com/wsdl"/></output>
      <fault name="AccessDeniedException">
        <soap:fault name="AccessDeniedException"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="http://www.smilehouse.com/wsdl"/></fault>
      <fault name="IteratorClosedException">
        <soap:fault name="IteratorClosedException"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="http://www.smilehouse.com/wsdl"/></fault>
      <fault name="OpenInterfaceException">
        <soap:fault name="OpenInterfaceException"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="http://www.smilehouse.com/wsdl"/></fault>
      <soap:operation soapAction=""/></operation>
    <operation name="mailSending">
      <input>
```

```
        <soap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="http://www.smilehouse.com/wsdl"/></input>
      <output>
        <soap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="http://www.smilehouse.com/wsdl"/></output>
      <fault name="OpenInterfaceException">
        <soap:fault name="OpenInterfaceException"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="http://www.smilehouse.com/wsdl"/></fault>
      <soap:operation soapAction=""/></operation>
    <operation name="openHQLIterator">
      <input>
        <soap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="http://www.smilehouse.com/wsdl"/></input>
      <output>
        <soap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="http://www.smilehouse.com/wsdl"/></output>
      <fault name="AccessDeniedException">
        <soap:fault name="AccessDeniedException"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="http://www.smilehouse.com/wsdl"/></fault>
      <fault name="OpenInterfaceException">
        <soap:fault name="OpenInterfaceException"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" use="encoded"
namespace="http://www.smilehouse.com/wsdl"/></fault>
      <soap:operation soapAction=""/></operation>
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
style="rpc"/></binding>
  <service name="OpenInterface">
    <port name="OpenInterfaceIFPort"
binding="tns:OpenInterfaceIFBinding">
      <soap:address xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
location="http://host:port/workspace.admin_licensetype_version/openinterf
ace"/>
</port>
</service>
</definitions>
```

10. APPENDIX 3: EXAMPLE OPEN INTERFACE CLIENT CODE

10.1. APACHE AXIS FOR JAVA

This chapter's example assumes basic knowledge of Java programming language and Apache Axis webservice framework. The following Java code locates Open Interface API address of a webshop and exports as XML all sales orders that have handling status set to ”Received”. The code requires Open Interface client stub classes (com.smilehouse.* packages) which will be generated from OpenInterfaceAddress and OpenInterface WSDL documents (see Appendix 2) by Axis' WSDL2Java tool.

To try out the code, simply set workspaceURL, organizationName, oiUserName and oiUserPassword connection parameter values in OpenInterfaceTestClient class' main method. Then compile and run it with Open Interface client stubs in the classpath. Note that this simplified example does not implement proper error handling.

This example has been tested to work with **Apache Axis 1.1**.

```
import java.rmi.RemoteException;

import javax.xml.rpc.ServiceException;
import javax.xml.rpc.Stub;

import com.smilehouse.www.types.AccessDeniedException;
import com.smilehouse.www.types.ExportResult;
import com.smilehouse.www.types.LoginInfo;
import com.smilehouse.www.types.OpenInterfaceException;
import com.smilehouse.www.types.OrderCriteria;
import com.smilehouse.www.wsdl.OpenInterfaceAddressIF;
import com.smilehouse.www.wsdl.OpenInterfaceAddressLocator;
import com.smilehouse.www.wsdl.OpenInterfaceIF;
import com.smilehouse.www.wsdl.OpenInterfaceLocator;

public class OpenInterfaceTestClient
{
   public static void main(String [] args) {

          try {
              String workspaceURL = "http://localhost:8080";
              String organizationName = "myshop";

              // Remember to grant OpenInterface access to this WS admin
user
              String oiUserName = "user";
              String oiUserPassword = "password";

              // Query OpenInterface address
              String addressServiceEndpoint = workspaceURL +
                   "/workspace.admin/openinterfaceaddress";

              Stub addressStub = (Stub) (new
                     OpenInterfaceAddressLocator().
                     getOpenInterfaceAddressIFPort());
              addressStub._setProperty(
                  javax.xml.rpc.Stub.ENDPOINT_ADDRESS_PROPERTY,
                  addressServiceEndpoint);
              OpenInterfaceAddressIF oiAddress =
                  (OpenInterfaceAddressIF) addressStub;
```

```
                System.out.println(
                  "Calling getOpenInterfaceAddress method at "
                   +  addressServiceEndpoint);
                String oiEndpointPath = oiAddress.
                        getOpenInterfaceAddress(organizationName);
                System.out.println("Retrieved OpenInterface path: " +
                    oiEndpointPath);

                // Query sales orders from retrieved OpenInterface address
                String oiEndpoint = workspaceURL + oiEndpointPath;

                LoginInfo loginInfo = new LoginInfo();
                loginInfo.setDatabase(organizationName);
                loginInfo.setUserName(oiUserName);
                loginInfo.setPassword(oiUserPassword);

                OrderCriteria orderCriteria = new OrderCriteria();
                orderCriteria.setHandlingStatusNameIn(new String[]
                                                    {"Received"});

// Name of an existing handling status to assign to exported orders.
                // Null value means no change to handling status.
                String newHandlingStatusName = null; // or e.g. "Exported"

                addressStub = (Stub) (new OpenInterfaceLocator().
                                    getOpenInterfaceIFPort());
                addressStub._setProperty(
                    javax.xml.rpc.Stub.ENDPOINT_ADDRESS_PROPERTY,
                 oiEndpoint);
                OpenInterfaceIF oi = (OpenInterfaceIF) addressStub;
                System.out.println("Calling exportOrders2 method at "
                                + oiEndpoint);
                ExportResult result = oi.exportOrders2(loginInfo,
                                                    orderCriteria,
                                        newHandlingStatusName);

                System.out.println(
                       "Retrieved exportOrders2 XML response:\n" +
                       result.getXml() + "\n");

            } catch(AccessDeniedException e) {
                // The user was missing OpenInterface access rights
                e.printStackTrace();
            } catch(OpenInterfaceException e) {
                e.printStackTrace();
            } catch(RemoteException e) {
                e.printStackTrace();
            } catch(ServiceException e) {
                e.printStackTrace();
            }

       }
}
```

Newer versions of Axis, at least up to **1.4**, are known to have a minor compatibility issue with nillable (null) parameters. A rough workaround to errors such as ”Non nillable element 'customerIdIn' is null.” resulting from this, is to modify Open Interface WSDL before passing it to Axis WSDL2Java tool. First delete all '*nillable=”true”*' attributes and then proceed with replacing '*<element*' string occurrences with '*<element*

nillable=”true”" (excluding the single quotes). Alternatively, use the original WSDL and instead modify the generated Java source files, replacing '*elemField.setNillable(false);*' lines with '*elemField.setNillable(true);*'.