Smilehouse Workspace 1.12.2

Identification Gateway API

Document Info

Document type:	Technical document
Creator:	Smilehouse Workspace Development Team
Date approved:	19.08.2009

Address
Itälahdenkatu 22 A
00210 Helsinki
Finland, Europe

Phone & Fax
+358 – 9 – 25 122 10
+358 – 9 – 25 122 119

Internet
web: www.smilehouse.com
e-mail: info@smilehouse.com

Table of Content

Address
Itälahdenkatu 22 A
00210 Helsinki
Finland, Europe

Phone & Fax
+358 – 9 – 25 122 10
+358 – 9 – 25 122 119

Internet
web: www.smilehouse.com
e-mail: info@smilehouse.com

1. Introduction

Smilehouse Workspace Identification Gateway API (**Identification Gateway**) provides a remote interface for customers identification in the web. Companies wanting to enable reliable customer identification in the Internet can integrate their applications with Identification Gateway and enable services for multiple identification vendors.

Finnish identification vendors services are based on the TUPAS standard of the Finnish Banker's Association. More information on the standard is documented at http://www.pankkiyhdistys.fi

Terms used in this document

- **Identification Gateway** Smilehouse Workspace Identification Gateway API. A Web Services API enabling reliable customer's identification for multiple identification vendors.
- **Application** the program using Identification Gateway for customer's identification.
- **Customer** uses the application via a browser. Is redirected to Identification Gateway for identification.
- **Identification Vendor** Organization or company offering online identification services for their customer's. Typically banks and credit card firms.
- **PIN** Personal Identity Number
- **BIC** Business Identity Code

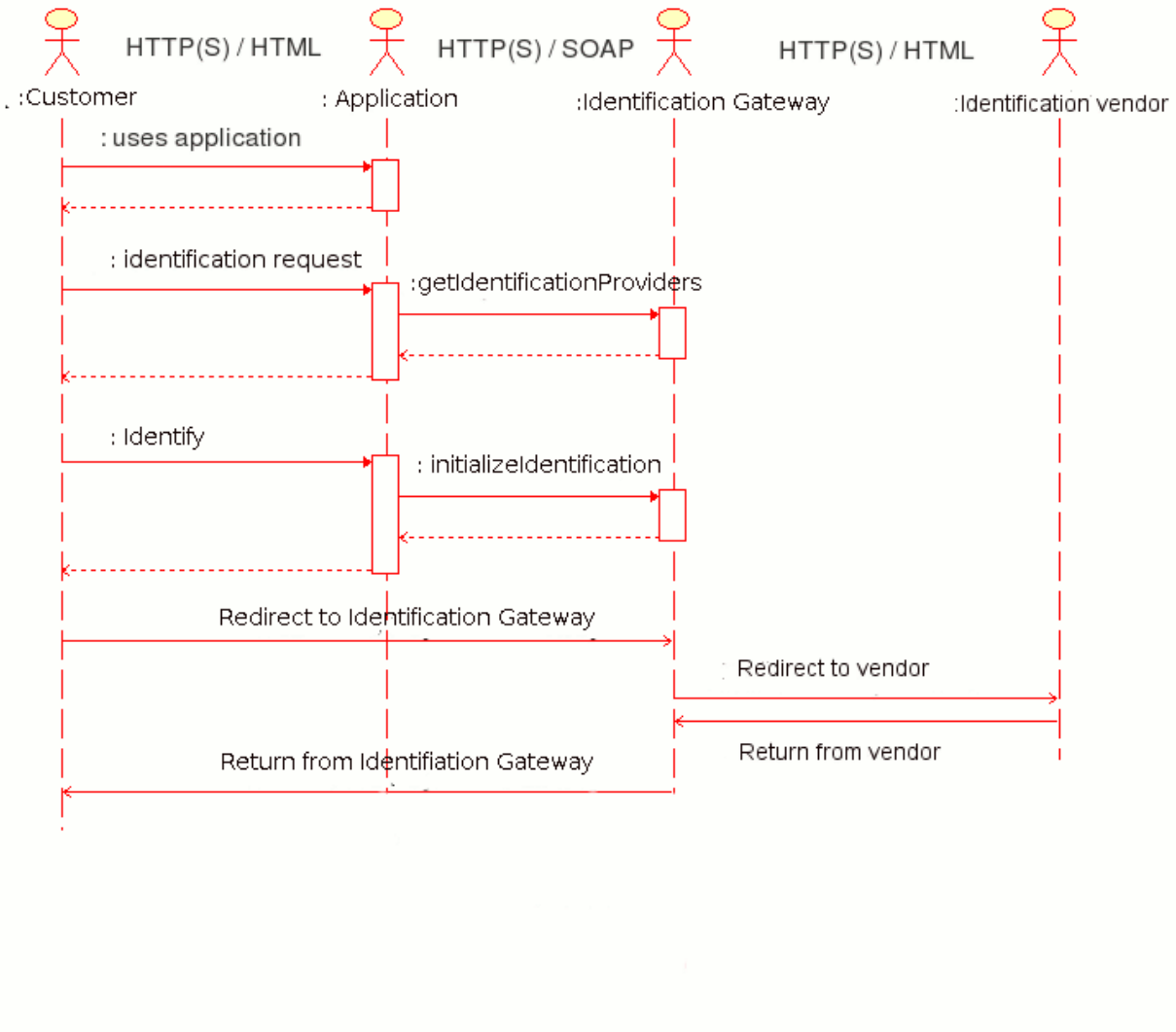
2. General description

The **application** using the **Identification Gateway** communicates by Web Services calls. When the **customer** using the application wants to identify, the application sends the customer to Identification Gateway, which redirects the customer to the chosen identification vendor's interface. The customer also returns from the identification vendor to the application through the Identification Gateway, which handles the information related to the identification event.

Prerequisites of Identification Gateway usage

- Installed Workspace Identification Gateway with a valid identification check key. Also identification type should be selected. Identification type selection should correspond on the function agreed in the identification vendor's service agreement. See Workspace Manual for configuring these parameters correctly from administration interface.
- The URL address of Identification Gateway. The format of the address is usually http://<host>/workspace.client_<organization>/IdentificationGateway. Organization is specified during Workspace installation. If Workspace client is installed to a virtualhost, the URL is http://<host>/IdentificationGateway .

3. Identification Gateway usage sequence



3.1. Get available identification providers

Direction: Application-> Identification Gateway

This API call is used by Application to build their 'choose identification vendor' selection. This API call is optional, if its already known, which vendors available.

Soap Call Parameters

Field	***Type***	***Description***
Organization	String	Unique id per application
Hash	String	MD5 hash of the above values

Soap Response Parameters

Field	***Type***	***Description***
Code	String	Check available response codes below (*)
IdentificationProviders	List	List of identification vendors.
Hash	String	MD5 hash of the above values

Identification providers

Field	***Type***	***Description***
ProviderId	String	id of the identification vendor
Name	String	Name of identification vendor. Can be modified from Workspace Identification Gateway administration page.
Description	String	Description of the identification vendor. Can be modified from Workspace Identification Gateway administration page.
IconURL	String	URL address of identification vendor's logo

Response Codes (*)

Code	***Description***
OK	API call successful.
unknown_organization	The given organization is unknown
missing_chek_key	MD5 hash check key is not set.
failed_mac	MD5 hash mac is corrupted

Request example:

```
<?xml version="1.0" encoding="UTF-8"?>
<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body>
    <GetIdentificationProviders
xmlns:sig="http://www.smilehouse.com/SmilehouseIdentificationGateway">
       <Organization>deve</Organization>
       <Hash>5ca785e86a515db99ef9ea5c35a6c5be</Hash>
```

```
    </GetIdentificationProviders>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Simple example how to use Java proxy classes for getIdentificationProviders operation

```
/**
   * Getting Identification providers.
   * This API call is used by Application to build their 'choose
   * identification
   * method'-selection.
   * The information returned contains provider id's set active,
   * possible costs,  and some other vendor related data.
   * @param stub
   * @return GetPaymentProvidersResponse
   * @throws RemoteException
   * @throws GenericFault
   */
public  GetIdentificationProvidersResponse
getIdentificationProvidersResponse(IdentificationGatewayServiceStub  stub)
throws RemoteException, GenericFault {
         GetIdentificationProviders providers = new
GetIdentificationProviders();
         MD5 hash = new MD5(organization + gatewayCheckKey);
         providers.setOrganization(organization);
         providers.setHash(hash.toString());
          return stub.GetIdentificationProvidersOperation(providers);
    }
```

Response example:

```
<SOAP-ENV:Envelope  xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/">
   <SOAP-ENV:Body>
      <sig:GetIdentificationProvidersResponse
xmlns:sig="http://www.smilehouse.com/SmilehouseIdentificationGateway">
          <sig:Code>OK</sig:Code>
          <sig:IdentificationProviders>
                 <sig:IdentificationProvider>
                       <sig:ProviderId>nordea</sig:ProviderId>
                       <sig:Name>Nordea E-Tunniste</sig:Name>
                       <sig:Description>Nordean TUPAS E-tunniste
järjestelmä</sig:Description>
                       <sig:IconURL>http://www.nordea.fi/solobutton.png</
sig:IconURL>
                  </sig:IdentificationProvider>
           </sig:IdentificationProviders>
            <sig:Hash>e6d6a3b61a75baf7e58003cbb49482c6</sig:Hash>
       </sig:GetIdentificationProvidersResponse>
    </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Simple example how to calculate MD5 hash from parameters included into GetPaymentProvidersResponse

```
StringBuffer hashBuffer = new StringBuffer();
// Initialize main GateWay Stub
IdentificationGatewayServiceStub stub = new
IdentificationGatewayServiceStub(null,  gatewayURL);
//  ------------------------------------------------------
// Getting vendor ids available and active at gateway
//  ------------------------------------------------------
GetIdentificationProvidersResponse providersResponse =
// check request java example for this method
gateway.getIdentificationProvidersResponse(stub);
hashBuffer = new StringBuffer();
hashBuffer.append(providersResponse.getCode());
for(int i=0;
i<providersResponse.getIdentificationProviders().getIdentificationProvider
().length; ++i) {
hashBuffer.append(providersResponse.getIdentificationProviders().getIdent
ificationProvider()[i].getProviderId());
hashBuffer.append(providersResponse.getIdentificationProviders().getIdent
ificationProvider()[i].getName());
hashBuffer.append(providersResponse.getIdentificationProviders().getIdent
ificationProvider()[i].getDescription());
hashBuffer.append(providersResponse.getIdentificationProviders().getIdent
ificationProvider()[i].getIconURL());
System.out.println("[GetIdentificationProvidersResponse] " + (i + 1) +
". provider id at list:[" +
providersResponse.getIdentificationProviders().getIdentificationProvider(
)[i].getProviderId() + "]");
}
hashBuffer.append(gatewayCheckKey);
System.out.println("[GetIdentificationProvidersResponse] Hash returned:
[" + providersResponse.getHash() + "]");
System.out.println("[GetIdentificationProvidersResponse] Hash excpected:
[" + MD5.getAsString(hashBuffer.toString()) + "]");
```

3.2. Initialize Identification

Direction: Application-> Identification Gateway

This API call is performed in order to control that the identification method exists, and also to get a common (IdentificationId) to use in further calls.

Soap Call Parameters

Field	Type	Description

Organization	String	Unique id per application
CallerId	String	Parameter identifying the system where the identification originates
Description	String	String to display to user in identification window
ProviderId	String	Id of identification vendor (Got from API call 1.)
ReturnURL	String	URL to redirect back to application after identification
Hash	String	MD5 hash of the above values

Soap Response Parameters

Field	**Type**	**Description**
Code	String	Check available response codes below (*)
IdentificationId	Integer	Id of this identification event to be used by subsequent API calls
Hash	String	MD5 hash of the above values

Response Codes (*)

Code	***Description***
OK	API call successful
unknown_organization	The given organization is unknown
unknown_provider	The given identificatin provider id is unknown
missing_chek_key	MD5 hash check key is not set.
failed_mac	MD5 hash mac is corrupted

Request example:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/">
   <SOAP-ENV:Body>
     <InitializeIdentification
xmlns:sig="http://www.smilehouse.com/SmilehouseIdentificationGateway">
         <Organization>deve</Organization>
         <CallerId>SOAPTester</CallerId>
         <Description>Description text would be here,
testing</Description>
         <ProviderId>nordea</ProviderId>
         <ReturnURL>http://back.to.the.shop</ReturnURL>
         <Hash>fe7432065ff07591a4802d41f6dc17e8</Hash>
     </InitializeIdentification>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Simple example how to execute InitializeIdentification operation with java proxy classes

```
/**
   * Make InitializePurchase request and get response.
   * This API call is performed in order to initialize Identification
transaction.
   * @param stub
```

```
   * @return InitializeIdentificationResponse response for initialize
   * @throws RemoteException
   * @throws GenericFault
   * @throws MalformedURIException
   */
public  InitializeIdentificationResponse
getInitializeIdentificationResponse(IdentificationGatewayServiceStub
stub) throws RemoteException, GenericFault, MalformedURIException  {
     StringBuffer hashBuffer = new StringBuffer();
      InitializeIdentification initialize = new InitializeIdentification();
      initialize.setOrganization(organization);
         initialize.setCallerId("SOAPTester");
         initialize.setDescription("Description text would be here");
         initialize.setProviderId("nordea");
         initialize.setReturnURL(new URI(returnURL));

        // build hash
         hashBuffer.append(organization);
         hashBuffer.append("SOAPTester");
         hashBuffer.append("Description text would be here");
         hashBuffer.append("nordea");
         hashBuffer.append(returnURL);
         hashBuffer.append(gatewayCheckKey);
          initialize.setHash(MD5.getAsString(hashBuffer.toString()));

          return stub.InitializeIdentificationOperation(initialize);
    }
```

Response exmple:

```
<SOAP-ENV:Envelope  xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body>
     <sig:InitializeIdentificationResponse
xmlns:sig="http://www.smilehouse.com/SmilehouseIdentificationGateway">
        <sig:Code>OK</sig:Code>
         <sig:IdentificationId>56</sig:IdentificationId>
         <sig:Hash>1d571b1d5810bbc6ba9d71aa0d23f306</sig:Hash>
     </sig:InitializeIdentificationResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Simple example how to calculate MD5 hash from parameters included into InitializeIdentificationResponse

```
//  ---------------------------------------
// Response for Initialize request
//  ---------------------------------------
InitializeIdentificationResponse initializeResponse   =
// check request java example for this method
gateway.getInitializeIdentificationResponse(stub);
hashBuffer = new StringBuffer();
            // Create MD5 hash string
             hashBuffer.append(initializeResponse.getCode());
             hashBuffer.append(initializeResponse.getIdentificationId());
             hashBuffer.append(gatewayCheckKey);
            // Refer to API documentation for available return codes
```

```
            System.out.println("[InitializeIdentificationResponse] Code
returned: [" + initializeResponse.getCode() + "]");
            System.out.println("[InitializeIdentificationResponse]
Identification transaction id returned: [" +
initializeResponse.getIdentificationId() + "]");
            System.out.println("[InitializeIdentificationResponse] Hash
returned: [" + initializeResponse.getHash() + "]");
            System.out.println("[InitializeIdentificationResponse] Hash
excpected:[" + MD5.getAsString(hashBuffer.toString()) + "]");
           // Check the MD5 hash
            assert(MD5.getAsString(hashBuffer.toString()).equals(initializ
eResponse.getHash()));
```

3.3. Redirect to Identification Gateway

Direction: Customer -> Identification Gateway

After identification initialization, Application should redirect the customer's browser to the Identification Gateway with the IdentificationId, Organization and Hash as a parameter,
e.g.

```
.../IGRedirect?Organization=ASDFGH&IdentificationId=123&Hash=xxx
```

MD5 hash should be calculated from these request parameters *IdentificationId+Organization+GatewayCheckKey.*

Gateway responds with a html page that contains a identification form with only a single submit button. The form will then be instantly submitted to vendor service by using client side scripting. If scripting is turned off, user must press the submit button to send the form. This page is required because most identification service vendors only accept the POST-method from forms.

3.4. Client Identification (Identification Gateway)

Client is directed to the identification vendor's service interface.

3.5. Return of client

Direction: Identification Gateway -> Customer

When returning from vendor's interface, The Identification Gateway redirects the customer's browser back to the Application (location specified with the *ReturnURL* parameter in previous calls). Identification event data is transmitted as a GET request parameters in ReturnURL, e.g. All parameters are listed at the table below.

Return Parameters

Field	Type	Description
Organization	String	
CustomerName	String	Plain text customer name in identification vendor's register. (***)
CustomerId	String	Customer Id returned from bank. Depending on settings, either encrypted or plain text.(***)

IdType (**)	Integer	00 – Not known 01 – plain text PIN 02 - plain text control sign of PIN 03 – plain text BIC 04 – plain text electronic password 05 – encrypted PIN 06 – encrypted BIC 07 – encrypted electronic password (***)
TimeStamp (*)	NNNyyyyMMddhhmmssxxxxxx	Unique identification confirm stamp generated by vendor (***)
Stamp (*)		Unique identification request stamp generated by Identification Gateway and returned from vendor (***)
IdNbr (*)	Integer	Unique vendor's identification event number. (***)
IdentificationStatus	Integer	A code indication the success or failure level. IDENTIFIED CANCELLED FAILED PENDING
IdentificationId	Integer	Reference for this identification event
ProviderId	String	Id of identification vendor
Hash	String	MD5 hash of the above values

MD5 hash of returned parameter values should be calculated in order these paramaters are listed above. Use Gateway's check key for hash calculation.

(*) - When using encrypted identification type then CustomerId is encrypted using these parameters and vendors mac key.
(TimeStamp&IdNbr&Stamp&customerID&vendors_mac_key&)
customerID is plain text Customer id in application's register

(**) - IdType return codes depends on identification type selected in Identification Gateway administration. Identification type selection should correspond on the function agreed in the identification vendor's service agreement. Below are available return codes. These codes tell, what kind of customer data returned from vendor.

See Workspace Manual for more information about identification types.

It is important to notice, that identification type selected from Workspace Identification Gateway will be used for all identification vendor services, so service agreement about identification type should be same for all vendors.

Identification type selections are:
Encrypted basic code (01) – possible return codes 05, 06, 07
Plain text basic code (02) – possible return codes 01, 03, 04
Plain text truncated code (03) – possible return code 02

(***) - These parameters returned only, if identification is successful. If customer cancels identification process at vendor's service, then these parameters are not returned. Hash should be calculated then from returned paramters only and using the same order as in case of succesful event.

4. More information

More technical information is available on request.

These include for example:

- Example application (SOAP tester) that integrates with the Payment Gateway
- Sample code how to use Java proxy classes.
- Example application that integrates with the Identification Gateway

For more information, please contact sales@smilehouse.com

5. Appendix 1: XML W3C Schematas

5.1.GetIdentificationProviders Query

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified"
targetNamespace="http://www.smilehouse.com/SmilehouseIdentificationGatew
ay"
xmlns:ns1="http://www.smilehouse.com/SmilehouseIdentificationGateway">
   <xs:element name="GetIdentificationProviders">
     <xs:complexType>
       <xs:sequence>
          <xs:element name="Organization" form="unqualified"
type="xs:string"/>
          <xs:element name="Hash" form="unqualified" type="xs:string"/>
       </xs:sequence>
     </xs:complexType>
   </xs:element>
</xs:schema>
```

5.2.GetIdentificationProviders Response

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified"
targetNamespace="http://www.smilehouse.com/SmilehouseIdentificationGate
way" xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:sig="http://www.smilehouse.com/SmilehouseIdentificationGateway">
  <!-- Get available e-identification providers -->
  <xs:element name="GetIdentificationProvidersResponse">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="sig:Code"/>
        <xs:element ref="sig:IdentificationProviders"/>
        <xs:element ref="sig:Hash"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="Code" type="xs:string"/>
  <xs:element name="IdentificationProviders">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="0" maxOccurs="unbounded"
ref="sig:IdentificationProvider"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="IdentificationProvider">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="sig:ProviderId"/>
        <xs:element ref="sig:Name"/>
        <xs:element ref="sig:Description"/>
        <xs:element ref="sig:IconURL"/>
      </xs:sequence>
    </xs:complexType>
 </xs:element>
 <xs:element name="ProviderId" type="xs:string"/>
  <xs:element name="Name" type="xs:string"/>
  <xs:element name="IconURL" type="xs:anyURI"/>
  <xs:element name="Hash" type="xs:string"/>
  <xs:element name="Description" type="xs:string"/>
</xs:schema>
```

5.3.InitializeIdentification Query

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified"
targetNamespace="http://www.smilehouse.com/SmilehouseIdentificationGatew
ay"
xmlns:ns1="http://www.smilehouse.com/SmilehouseIdentificationGateway">
  <xs:element name="InitializeIdentification">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Organization" form="unqualified"
type="xs:string"/>
        <xs:element name="CallerId" form="unqualified" type="xs:string"/>
```

```
        <xs:element name="Description" form="unqualified"
type="xs:string"/>
        <xs:element name="ProviderId" form="unqualified"
type="xs:string"/>
        <xs:element name="ReturnURL" form="unqualified" type="xs:anyURI"/
>
        <xs:element name="Hash" form="unqualified" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

5.4.InitializeIdentification Response

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified"
targetNamespace="http://www.smilehouse.com/SmilehouseIdentificationGatew
ay"
xmlns:sig="http://www.smilehouse.com/SmilehouseIdentificationGateway">
   <xs:element name="InitializeIdentificationResponse">
     <xs:complexType>
       <xs:sequence>
         <xs:element ref="sig:Code"/>
         <xs:element ref="sig:IdentificationId"/>
         <xs:element ref="sig:Hash"/>
       </xs:sequence>
     </xs:complexType>
   </xs:element>
   <xs:element name="Code" type="xs:string"/>
   <xs:element name="IdentificationId" type="xs:integer"/>
   <xs:element name="Hash" type="xs:string"/>
</xs:schema>
```

Address
Itälahdenkatu 22 A
00210 Helsinki
Finland, Europe

Phone & Fax
+358 – 9 – 25 122 10
+358 – 9 – 25 122 119

Internet
web: www.smilehouse.com
e-mail: info@smilehouse.com

6. Appendix 2: Smilehouse Identification Gateway WSDL

There are couple of restrictions in the IdentificationGateway SOAP API that may require some manual fine-tuning of the automatically generated client stub code.

IdentificationGateway SOAP API does not support multiRef elements and XML schema type attributes. A SOAP query containing a either of these will not pass server's XML schema validation step. If you are using Axis 1.x WSDL2Java for Java stub generation, you need to change a few settings in order to disable sending multiRef elements and XML schema type attributes. See IdentificationGatewayPortStub.java, at the end of createCall() method, insert just before the "return _call;" line:

```
 _call.setProperty(org.apache.axis.client.Call.SEND_TYPE_ATTR,
Boolean.FALSE);
 _call.setProperty(org.apache.axis.AxisEngine.PROP_DOMULTIREFS,
Boolean.FALSE);
```

Every SOAP call parameter element MUST have either an explicit namespace prefix ("sig:") or default namespace set to URI "http://www.smilehouse.com/SmilehouseIdentificationGateway" by the command element (e.g. "GetIdentificationProviders").
An example follows:

Correct:

```
<GetIdentificationProviders
xmlns="http://www.smilehouse.com/SmilehouseIdentificationGateway">
      <Organization>exampleshop</Organization>
      <Hash>MD5HashCodeHere</Hash>
</GetIdentificationProviders>
```

Correct 2:

```
<ns1:GetIdentificationProviders
xmlns:ns1="http://www.smilehouse.com/SmilehouseIdentificationGateway">
      <ns1:Organization>exampleshop</ns1:Organization>
      <ns1:Hash>MD5HashCodeHere</ns1:Hash>
</ns1:GetIdentificationProviders>
```

Incorrect (this will not pass server side schema validation):

```
<ns1:GetIdentificationProviders
xmlns:ns1="http://www.smilehouse.com/SmilehouseIdentificationGateway">
            <Organization>exampleshop</Organization>
            <Hash>MD5HashCodeHere</Hash>
</ns1:GetPaymentIdentificationProviders>
```

Current Identification Gateway Webservice definition (**WSDL 1.1, SOAP 1.1**) is compatible with **Java Axis2 v1.3, Axis1 v1.4** and also with Microsoft Visual Studio .NET 2003 proxy generator (wsdl.exe).

If you're having trouble generating proxy classes with one of the Axis2 WSDL2Java proxy generators, then contact us at workspace.support@smilehouse.com

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<wsdl:definitions name="IdentificationGateway"
  targetNamespace="http://www.smilehouse.com/SmilehouseIdentificationGatew
ay"
  xmlns:tns="http://www.smilehouse.com/SmilehouseIdentificationGateway"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:sig="http://www.smilehouse.com/SmilehouseIdentificationGateway">
<wsdl:documentation>Smilehouse Identification
Gateway</wsdl:documentation>

<!-- SOAP command schemata -->
<wsdl:types>
<xs:schema
   xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified"
   targetNamespace="http://www.smilehouse.com/SmilehouseIdentificationGate
way"
   xmlns:SOAPENV="http://schemas.xmlsoap.org/soap/envelope/"
   xmlns:ns1="http://www.smilehouse.com/SmilehouseIdentificationGateway"
   xmlns:sig="http://www.smilehouse.com/SmilehouseIdentificationGateway">
<xs:element name="GetIdentificationProviders">
               <xs:complexType>
                   <xs:sequence>
                       <xs:element name="Organization"
form="unqualified" type="xs:string"/>
```

Address
Itälahdenkatu 22 A
00210 Helsinki
Finland, Europe

Phone & Fax
+358 – 9 – 25 122 10
+358 – 9 – 25 122 119

Internet
web: www.smilehouse.com
e-mail: info@smilehouse.com

```
                        <xs:element name="Hash" form="unqualified"
type="xs:string"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>

            <xs:element name="GetIdentificationProvidersResponse">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element ref="sig:Code"/>
                        <xs:element ref="sig:IdentificationProviders"/>
                        <xs:element ref="sig:Hash"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="Code" type="xs:string"/>

            <xs:element name="IdentificationProviders">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element minOccurs="0" maxOccurs="unbounded"
ref="sig:IdentificationProvider"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>

            <xs:element name="IdentificationProvider">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element ref="sig:ProviderId"/>
                        <xs:element ref="sig:Name"/>
                        <xs:element ref="sig:Description"/>
                        <xs:element ref="sig:IconURL"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>

            <xs:element name="ProviderId" type="xs:string"/>
            <xs:element name="Name" type="xs:string"/>
            <xs:element name="IconURL" type="xs:anyURI"/>
            <xs:element name="Hash" type="xs:string"/>
            <xs:element name="Description" type="xs:string"/>

            <xs:element name="InitializeIdentification">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="Organization"
form="unqualified"  type="xs:string"/>
    <xs:element name="CallerId" form="unqualified" type="xs:string"/>
    <xs:element name="Description" form="unqualified" type="xs:string"/>
    <xs:element name="ProviderId" form="unqualified" type="xs:string"/>
    <xs:element name="ReturnURL" form="unqualified" type="xs:anyURI"/>
    <xs:element name="Hash" form="unqualified" type="xs:string"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>

            <xs:element name="InitializeIdentificationResponse">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element ref="sig:Code"/>
```

Address
Itälahdenkatu 22 A
00210 Helsinki
Finland, Europe

Phone & Fax
+358 – 9 – 25 122 10
+358 – 9 – 25 122 119

Internet
web: www.smilehouse.com
e-mail: info@smilehouse.com

```
                        <xs:element ref="sig:IdentificationId"/>
                       <xs:element ref="sig:Hash"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <!--
             <xs:element name="Code" type="xs:string"/-->
             <xs:element name="IdentificationId" type="xs:integer"/>
             <!--xs:element name="Hash" type="xs:string"/-->

             <xsd:element name="FaultDetail">
                 <xs:complexType>
                     <xs:sequence>
     <xs:element name="faultstring" form="unqualified" type="xsd:string"/>
     <xs:element name="faultcode" form="unqualified" type="xsd:string"/>
                     </xs:sequence>
                 </xs:complexType>
             </xsd:element>

         </xs:schema>
     </wsdl:types>

<!-- SOAP command message content mapping to schemata -->
<wsdl:message  name="GetIdentificationProvidersRequest">
   <wsdl:part name="body" element="sig:GetIdentificationProviders"/>
</wsdl:message>
<wsdl:message  name="GetIdentificationProvidersResponse">
    <wsdl:part name="body"
element="sig:GetIdentificationProvidersResponse"/>
     </wsdl:message>
      <wsdl:message name="InitializeIdentificationRequest">
          <wsdl:part name="body" element="sig:InitializeIdentification"/>
     </wsdl:message>
      <wsdl:message name="InitializeIdentificationResponse">
         <wsdl:part name="body"
element="sig:InitializeIdentificationResponse"/>
     </wsdl:message>
     <!-- Generic SOAP Fault message shared by all commands -->
     <wsdl:message name="GenericFault">
         <wsdl:part name="fault" element="sig:FaultDetail"/>
     </wsdl:message>

     <!-- Declaring the SOAP commands as Operations, mapping their
input/output/fault  messages
          to above message definitions -->
      <wsdl:portType name="IdentificationGatewayPort">
          <wsdl:operation name="GetIdentificationProvidersOperation">
              <wsdl:input message="tns:GetIdentificationProvidersRequest"/>
             <wsdl:output
message="tns:GetIdentificationProvidersResponse"/>
              <wsdl:fault name="GenericFault" message="tns:GenericFault"/>
         </wsdl:operation>
          <wsdl:operation name="InitializeIdentificationOperation">
              <wsdl:input message="tns:InitializeIdentificationRequest"/>
              <wsdl:output message="tns:InitializeIdentificationResponse"/>
              <wsdl:fault name="GenericFault" message="tns:GenericFault"/>
         </wsdl:operation>
     </wsdl:portType>
     <!-- Bind IdentificationGateway Operations to
IdentificationGatewayPort  -->
```

```
    <wsdl:binding name="IdentificationGatewayPort"
type="tns:IdentificationGatewayPort">
        <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
        <wsdl:operation name="GetIdentificationProvidersOperation">
            <soap:operation
                soapAction="http://www.example.com/workspace.client_examp
leshop/IdentificationGateway"
                style="document"/>
            <wsdl:input>
                <soap:body use="literal"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                    namespace="http://www.smilehouse.com/SmilehouseIdenti
ficationGateway"/>
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                    namespace="http://www.smilehouse.com/SmilehouseIdenti
ficationGateway"/>
            </wsdl:output>
            <wsdl:fault name="GenericFault">
                <soap:fault name="GenericFault" use="literal"
                    encodingStyle="http://schemas.xmlsoap.org/soap/encodi
ng/"
                    namespace="http://www.smilehouse.com/SmilehouseIdenti
ficationGateway"/>
            </wsdl:fault>
        </wsdl:operation>
        <wsdl:operation name="InitializeIdentificationOperation">
            <soap:operation
                soapAction="http://www.example.com/workspace.client_examp
leshop/IdentificationGateway"
                style="document"/>
            <wsdl:input>
                <soap:body use="literal"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                    namespace="http://www.smilehouse.com/SmilehouseIdenti
ficationGateway"/>
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                    namespace="http://www.smilehouse.com/SmilehouseIdenti
ficationGateway"/>
            </wsdl:output>
<wsdl:fault  name="GenericFault">
   <soap:fault name="GenericFault" use="literal"
                    encodingStyle="http://schemas.xmlsoap.org/soap/encodi
ng/"
                    namespace="http://www.smilehouse.com/SmilehouseIdenti
ficationGateway"/>
   </wsdl:fault>
  </wsdl:operation>
 </wsdl:binding>
 <!-- Finally we declare the IdentificationGateway service -->
  <wsdl:service name="IdentificationGatewayService">
    <wsdl:port name="IdentificationGatewayPort"
binding="tns:IdentificationGatewayPort">
       <documentation>Smilehouse Identification Gateway SOAP
API</documentation>
```

```
            <soap:address
                 location="http://www.example.com/workspace.client_example
shop/IdentificationGateway"/>
        </wsdl:port>
    </wsdl:service>
</wsdl:definitions>
```

7. Setting up 3rd party libraries

Identification Gateway needs **Xalan-Java** libraries. Xalan-java is an XSLT processor for transforming XML documents.
Files **xalan.jar** and **serializer.jar** are needed.
Version:2.7.0 License:Apache License,Version 2.0, January 2004, http://www.apache.org/licenses/
Download libraries from URL:http://xml.apache.org/xalan-j/

After downloading just copy the **xalan.jar** and **serializer.jar** packages into directory, where tomcat's shared libraries are being held.
Usually its $CATALINA_HOME/shared/lib