Smilehouse Workspace 1.13

Payment Gateway API

Document Info

Document type:	Technical document
Creator:	Smilehouse Workspace Development Team
Date approved:	31.05.2010

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 Payment Gateway API (**Payment Gateway**) provides a remote interface for conducting payment transactions in the web. Company's wanting to enable payment in the web can integrate their applications with Payment Gateway and enable payment services for multiple payment vendors.

For supported payment vendors refer to Workspace Technical Whitepaper which is available for download from http://workspace.smilehouse.com

Terms used in this document

- **Payment Gateway** Smilehouse Workspace Payment Gateway API. A Web Services API enabling payment transactions for multiple payment vendors
- **Application** the program using Payment Gateway for payment transactions.
- **Customer** uses the application via a browser. Is redirected to Payment Gateway for payment.
- **Payment Vendor** Organization or company offering online payment services for their customer's. Typically banks and credit card firms.
- **GatewayCheckKey** Secret key saved at Payment Gateway administration interface. Key is used in MD5 hash calculation. MD5 hash is included in every SOAP message sent and received. Hash should be calculated from parameters in SOAP message. All parameter values should be concatenated in order they present in SOAP message and GatewayCheckKey should be then added at the end of the string.
  For example
  MD5(parametervalue1parametervalue2parametervalue3GatewayCheckKey)

2. General description

The **application** using the **Payment Gateway** communicates by Web Services calls. When the **customer** using the application, wants to pay a transaction, the application sends the customer to Payment Gateway which redirects the customer to the chosen payment vendor's interface. The customer also returns from the payment vendor to the application through the Payment Gateway which handles the information related to the transaction. When the customer returns to the application, the application can check the transaction's status from Workspace via a Web Services call.

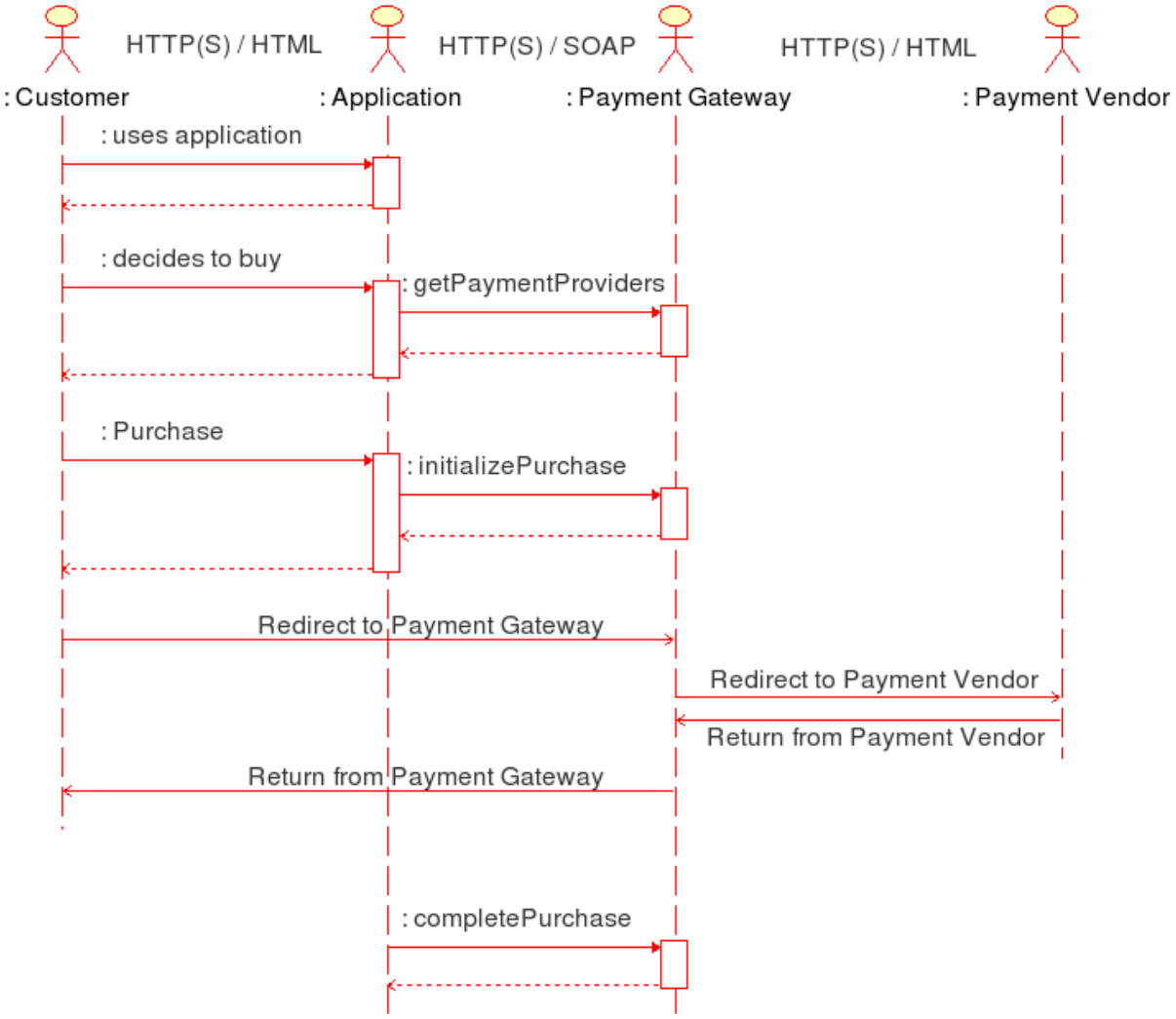
Prerequisites of Payment Gateway usage

- Installed Workspace Payment Gateway with a valid Payment check key set. See Workspace Manual for configuring the Gateway key.
- The URL address of Payment Gateway. The format of the address is http://<host>/workspace.client_<organization>/PaymentGateway. Organization is specified during Workspace installation. If Workspace client is installed to a virtualhost, the URL is http://<host>/PaymentGateway .

3. Payment Gateway Usage Sequence

This chapter describes the communication between the integrated application, Payment Gateway and the customer's browser. Diagram 1 is an UML sequence diagram describing the communication between parties. The next chapters describes the API calls mentioned in the diagram in order they should be used by Application.

Diagram 1. Payment Gateway Usage Sequence diagram



3.1.Get available payment providers

Direction: Application-> Payment Gateway

This API call is used by Application to build their 'choose payment'-selection. The information returned contains payment provider id's, possible costs related to the payment for the client application and some other payment vendor related data. If Application wishes to use these, they must be added as items to the Order by Application. API call is not mandatory, if you know the ProviderIds. ProviderId should be used at call 2 – Initialize purchase.

Soap Call Params

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

Soap Response Params

Field	***Type***	***Description***
Code	String	`Value is always "OK"`
Description	String	`Value is always "OK"`
PaymentProviders	List	List of Payment Providers (see the payment providers table below)
Hash	String	MD5 hash of the above values. MD5(Code+Description+PaymentProviders+GatewayCheckKey). Check PaymentProviders List structure below.

Payment providers

Field	***Type***	***Description***
ProviderId	String	id of the payment provider
Name	String	Name of payment provider
Description	String	Description of the payment provider
Price	Integer	Price of the payment in lowest monetary unit (in the supported currency)
VAT	Double	Included VAT %
Currency	String	Supported currency (EUR,USD)
AmountExp	Integer	Exponent to convert from the lowest monetary unit to the actual currency (-2 for cents -> euro)
IconURL	String	URL address of payment form button image. NOTE! Luottokunta payment doesn't have this.
PaymentCode	String	The code for the payment provider.

Request example:

```
<?xml version="1.0" encoding="UTF-8"?>
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:enc="http://schemas.xmlsoap.org/soap/encoding/"
env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xs="http://www.w3.org/1999/XMLSchema">
<env:Body>
  <GetPaymentProviders
xmlns:spg="http://www.smilehouse.com/SmilehousePaymentGateway">
  <Organization>shop2</Organization>
  <Hash>b0f133ca57d815e57886a7ce0f550ac9</Hash>
 </GetPaymentProviders>
</env:Body>
</env:Envelope>
```

Simple example how to use Java proxy classes for getPaymentProviders operation

```
/**
 * Getting payment providers
 * @param stub
 * @return GetPaymentProvidersResponse
 * @throws RemoteException
 * @throws GenericFault
 */
private GetPaymentProvidersResponse
getPaymentProvidersResponse(PaymentGatewayServiceStub stub) throws
Exception {
  GetPaymentProviders providers = new GetPaymentProviders();
  MD5 hash = new MD5(organization + gatewayCheckKey);
  providers.setOrganization(organization);
  providers.setHash(hash.toString());
  return stub.GetPaymentProvidersOperation(providers);
}
```

Response example:

```
<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Body>
<spg:GetPaymentProvidersResponse
xmlns:spg="http://www.smilehouse.com/SmilehousePaymentGateway">
<spg:Code>OK </spg:Code>
<spg:Description>OK </spg:Description>
<spg:PaymentProviders>
       <spg:PaymentProvider>
              <spg:ProviderId>invoice </spg:ProviderId>
              <spg:Name>Invoice </spg:Name>
              <spg:Description>Maksu laskulla </spg:Description>
              <spg:Price>1100 </spg:Price>
              <spg:VAT>22.0 </spg:VAT>
              <spg:Currency>EUR </spg:Currency>
              <spg:AmountExp>-2 </spg:AmountExp>
              <spg:IconURL> </spg:IconURL>
```

```
              <spg:PaymentCode>IN </spg:PaymentCode>
       </spg:PaymentProvider>
      <spg:PaymentProvider>
              <spg:ProviderId>solo</spg:ProviderId>
              <spg:Name>Nordea Solo</spg:Name>
              <spg:Description>Solon E-maksu </spg:Description>
              <spg:Price>0 </spg:Price>
              <spg:VAT>0.0 </spg:VAT>
              <spg:Currency>EUR </spg:Currency>
              <spg:AmountExp>-2 </spg:AmountExp>
<spg:IconURL>http://someshop.smilehouse.com/workspace.client_shop2/pics/p
ayment/nordea_logo_100x22.gif</spg:IconURL>
              <spg:PaymentCode>NO </spg:PaymentCode>
       </spg:PaymentProvider>
</spg:PaymentProviders>
<spg:Hash>ad1dd57d8fe7450b7681d86eed783ac8 </spg:Hash>
</spg:GetPaymentProvidersResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

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

```
// ------------------------------------------------------
// Getting vendor ids available and active at gateway
// ------------------------------------------------------
// Initialize main GateWay Stub
PaymentGatewayServiceStub stub = new PaymentGatewayServiceStub(null,
gatewayURL);
GetPaymentProvidersResponse providersResponse =
// check request java example for this method
gateway.getPaymentProvidersResponse(stub);

hashBuffer = new StringBuffer();
hashBuffer.append(providersResponse.getCode());
hashBuffer.append(providersResponse.getDescription());
for(int i=0; i<providersResponse.getPaymentProviders()
.getPaymentProvider().length; ++i) {
   hashBuffer.append(providersResponse.getPaymentProviders()
           .getPaymentProvider()[i].getProviderId());
   hashBuffer.append(providersResponse.getPaymentProviders()
           .getPaymentProvider()[i].getName());
   hashBuffer.append(providersResponse.getPaymentProviders()
           .getPaymentProvider()[i].getDescription());
   hashBuffer.append(providersResponse.getPaymentProviders()
           .getPaymentProvider()[i].getPrice());
   hashBuffer.append(providersResponse.getPaymentProviders()
           .getPaymentProvider()[i].getVAT());
   hashBuffer.append(providersResponse.getPaymentProviders()
           .getPaymentProvider()[i].getCurrency());
   hashBuffer.append(providersResponse.getPaymentProviders()
           .getPaymentProvider()[i].getAmountExp());
   hashBuffer.append(providersResponse.getPaymentProviders()
           .getPaymentProvider()[i].getIconURL());
   hashBuffer.append(providersResponse.getPaymentProviders()
           .getPaymentProvider()[i].getPaymentCode());
   System.out.println("[GetPaymentProvidersResponse] "
+ (i + 1) + ". provider id at list:[" +
providersResponse.getPaymentProviders().getPaymentProvider()[i]
           .getProviderId() + "]");
}
```

```
hashBuffer.append(gatewayCheckKey);
```

3.2.Initialize purchase

Direction: Application-> Payment Gateway

This API call is performed in order to initialize payment transaction and that the amount and currency is valid etc. Request returns unique transaction Id just initialized within Payment Gateway, which should be used as common reference in further calls. After initialization every Transaction gets "Pending" status within Payment Gateway. More precise description of possible TransactionStatuses available at chapter 3.6

Soap Call Params

Field	Type	Description
Organization	String	Unique id per store
OrderId	String	Parameter identifying order, just pass it around to enable later calls to check orderId. Should be generated by Application.
CallerId	String	Parameter identifying the system where the purchase originates. Could be the Name of Application for example.
ReferenceNumber	String	Standard reference number to be used in bank payments. If left empty the Payment Gateway will generate the reference number using the reference number sequence defined in Workspace.
Description	String	String to display to user in payment window
ProviderId	String	id of bank (Got from API call 1.) *Cannot be left blank.*
Currency	String	Currency code (Got from API call 1.)
Language	String	Language in which the bank should display the payment UI. (See Appendix 3)
AmountExp	Integer	Exponent used in the amounts (Always -2, means that the amount is expressed in cents)
*FirstName	String	
*LastName	String	
*StreetAddress	String	
*PostalCode	String	
*City	String	
*State	String	
*Country	String	
*EMail	String	
*Phone	String	
Items	List	List of the ordered items. See the Items table below.
ReturnURL	String	URL to redirect back to after payment. This is usually the Application URL, which

		handles TransactionStatuses
Hash	String	MD5 hash of the above values

*Required only for PayPal and ITransact payments. It is preferred to always pass this information as it can be used to give a better user experience (to inform the user what he's paying for in the payment provider's service).

Items

Field	***Type***	***Description***
ItemName	String	Name of the item
ItemCode	String	Item code
ItemPrice	Integer	Price of the item in lowest monetary unit (See AmountExp)
Quantity	Integer	The quantity of the item

Soap Response Params

Field	**Type**	**Description**
Code	String	Response codes are explained below (*)
Description	String	Description of the result code.
TransactionId	Integer	Unique transactionId to be used by subsequent api calls
ReferenceNumber	String	Either the same as given in the request or reference number generated by Payment Gateway if not given
Hash	String	MD5 hash of the above values

(*) Possible response codes:

1. `OK`
2. `unknown_provider` - The given payment provider is not known
3. `invalid_currency` - The given currency code was not valid.
4. `unsupported_currency` - The given currency is not supported by the given payment provider

Request example:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
<env:Body>
<spg:InitializePurchase
xmlns:spg="http://www.smilehouse.com/SmilehousePaymentGateway">
<Organization>shop2 </Organization>
<OrderId>12345 </OrderId>
<CallerId>SOAPTester </CallerId>
<Description>Description text would be here, testing </Description>
<ProviderId>solo </ProviderId>
<Currency>EUR </Currency>
```

```
<Language>en</Language>
<AmountExp>-2 </AmountExp>
<FirstName>Jack </FirstName>
<LastName>Bauer </LastName>
<StreetAddress>Street Address 123 </StreetAddress>
<PostalCode>XYZ-50000 </PostalCode>
<City>Los Angeles </City>
<Country>USA </Country>
<EMail>jack.bauer@ctu.com.invalid </EMail>
<Phone>555-3000-4000 </Phone>
<Items>
            <Item>
                <ItemName>Product #1 </ItemName>
                <ItemCode>PROD0001 </ItemCode>
                <ItemPrice>2510 </ItemPrice>
                <Quantity>1 </Quantity>
           </Item>
            <Item>
                <ItemName>Product #2 </ItemName>
                <ItemCode>PROD0002 </ItemCode>
                <ItemPrice>550 </ItemPrice>
                <Quantity>3 </Quantity>
            </Item>
            <Item>
                  <ItemName>Product #3 </ItemName>
                  <ItemCode>PROD0003 </ItemCode>
                  <ItemPrice>12399 </ItemPrice>
                  <Quantity>2 </Quantity>
            </Item>
 </Items>
<ReturnURL>http://localhost:8080/soaptester.html </ReturnURL>
<Hash>6704c04ffaa1b926b763a6223c82579e  </Hash>
</spg:InitializePurchase>
</env:Body>
</env:Envelope>
```

Simple example how to execute InitializePurchase operation with java proxy classes

```
/**
 * Make InitializePurchase request and get response
 * @param stub
 * @return InitializePurchaseResponse, response for initialize
 * @throws MalformedURIException
 * @throws RemoteException
 * @throws GenericFault
 */
private InitializePurchaseResponse
getInitializePurchaseResponse(PaymentGatewayServiceStub stub) throws
Exception {
        InitializePurchase purchase = new InitializePurchase();
        Items_type0 items = new Items_type0();
        purchase.setOrganization(organization);
        purchase.setOrderId("12345");
        purchase.setCallerId("SOAPTester");
        purchase.setDescription("Description text would be here");
        purchase.setProviderId("solo");
        purchase.setCurrency("EUR");
```

```
purchase.setLanguage("en")
purchase.setAmountExp(BigInteger.valueOf(-2));
purchase.setFirstName("Jack");
purchase.setLastName("Bauer");
purchase.setStreetAddress("Street Address 123");
purchase.setPostalCode("XYZ-50000");
purchase.setCity("Los Angeles");
purchase.setCountry("USA");
purchase.setEMail("jack.bauer@ctu.com.invalid");
purchase.setPhone("555-3000-4000");

Item_type0 item = new Item_type0();
item.setItemCode("PROD0001");
item.setItemName("Product #1");
item.setItemPrice(BigInteger.valueOf(1000));
item.setQuantity(BigInteger.valueOf(1));
items.addItem(item);

item = new Item_type0();
item.setItemCode("PROD0002");
item.setItemName("Product #2");
item.setItemPrice(BigInteger.valueOf(90));
item.setQuantity(BigInteger.valueOf(1));
items.addItem(item);

item = new Item_type0();
item.setItemCode("PROD0003");
item.setItemName("Product #3");
item.setItemPrice(BigInteger.valueOf(40));
item.setQuantity(BigInteger.valueOf(1));
items.addItem(item);

purchase.setItems(items);
purchase.setReturnURL(new URI("http://someurl.com/returnURL"));
//Create MD5 hash of parameters in order they appear in SOAP
//message
hashBuffer.append(organization);
hashBuffer.append("12345");
hashBuffer.append("SOAPTester");
hashBuffer.append("Description text would be here");

hashBuffer.append("solo");
hashBuffer.append("EUR");
hashBuffer.append("en");
hashBuffer.append("-2");
hashBuffer.append("Jack");
hashBuffer.append("Bauer");
hashBuffer.append("Street Address 123");
hashBuffer.append("XYZ-50000");
hashBuffer.append("Los Angeles");
hashBuffer.append("USA");
hashBuffer.append("jack.bauer@ctu.com.invalid");
hashBuffer.append("555-3000-4000");
hashBuffer.append("Product #1");
hashBuffer.append("PROD0001");
hashBuffer.append("1000");
hashBuffer.append("1");
hashBuffer.append("Product #2");
hashBuffer.append("PROD0002");
hashBuffer.append("90");
hashBuffer.append("1");
hashBuffer.append("Product #3");
```

```
        hashBuffer.append("PROD0003");
        hashBuffer.append("40");
        hashBuffer.append("1");
        hashBuffer.append("http://someurl.com/returnURL");
        hashBuffer.append(gatewayCheckKey);
        purchase.setHash(hashBuffer.toString());
        return stub.InitializePurchaseOperation(purchase);
    }
```

Response example:

```
<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/"> <SOAP-ENV:Body>
<spg:InitializePurchaseResponse
xmlns:spg="http://www.smilehouse.com/SmilehousePaymentGateway">
   <spg:Code>OK </spg:Code>
   <spg:Description>OK </spg:Description>
   <spg:TransactionId>67 </spg:TransactionId>
   <spg:ReferenceNumber>1672 </spg:ReferenceNumber>
   <spg:Hash>24d3e77d92793c51084524774ecef497 </spg:Hash>
</spg:InitializePurchaseResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

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

```
// ---------------------------------------
// Response for Initialize purchase request
// ---------------------------------------
InitializePurchaseResponse initializeResponse  =
gateway.getInitializePurchaseResponse(stub);
// Create MD5 hash string
hashBuffer.append(initializeResponse.getCode());
hashBuffer.append(initializeResponse.getDescription());
hashBuffer.append(initializeResponse.getTransactionId());
hashBuffer.append(initializeResponse.getReferenceNumber());
hashBuffer.append(gatewayCheckKey);
```

3.3.Redirect to Payment Gateway

Direction: Customer -> Payment Gateway

After the successful initialization call, and retrieval of unique TransactionId Application should redirect the customer's browser to the Payment Gateway with the TransactionId

as a parameter,
e.g.

```
.../PGRedirect?Organization=ASDFGH&TransactionId=LGKRELGKLRKGL&Hash=xxxxx
```

Hash should be calculated from request parameters
TransactionId+Organization+GatewayCheckKey

Payment gateway responds with a html page that contains a form with only a single submit button. The form will be instantly submitted 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 payment providers only accept the POST-method from forms.

3.4. Redirect to Payment Vendor (Payment Gateway)

Customer is directed to the bank of choice or submits credit card information.

3.5. Return from Payment Gateway

Direction: Payment Gateway -> Application

After successful payment at bank, customer returns to Payment Gateway and the Gateway redirects the customer back to the application (location specified with the returnURL parameter in previous calls). The related TransactionID is transmitted as a request parameter back to the application. Other parameters included are Organization and Hash, e.g. Application then should initialize "Complete Purchase" call (chapter 3.6) to find out the final status of customer's order registered at Payment Gateway.

```
RETURN_URL?Organization=ASDFGH&TransactionId=LGKRELGKLRKGL&Hash=xxxxx
```

Hash is calculated from request parameters
TransactionId+Organization+GatewayCheckKey

3.6. Complete purchase

Direction: Application -> Payment Gateway

Application uses this call to query the Payment Gateway to control if the payment was successful, and if customer returned correctly from vendor's service. This is the last call application makes to Gateway in payment process. Based on TransactionStatus application knows how payment process ended.

Soap Call Parameters

Field	Type	Description
Organization	String	Unique id per store, see call 2 for a description.
TransactionId	Integer	Reference for this transaction

Hash	String	MD5 hash of the above values

Soap Response Parameters

Field	**Type**	**Description**
Code	String	Response code (*)
Description	String	OK if OK, otherwise longer description (e.g. unknown transaction)

TransactionStatus	Integer	A code indication the success or failure level. (**).
		`0 = pending`
		`1 = paid`
		`2 = cancelled`
		`3 = failed`
		`Look at more descriptive explanations below.`

TransactionId	Integer	Reference for this transaction
OrderId	String	The OrderId sent from shop in initialize method
ProviderId	String	id of payment provider
Amount	Integer	Amount charged, in lowest monetary unit (cents)
AmountExp	Integer	Amount exponent
Hash	String	MD5 hash of the above values

(*) Possible response codes:

5. `OK`
6. `transaction_not_found` - No transaction was found with the given Transaction Id

(**) Explanations of TransactionStatuses:

`Pending` – Initial status of every transaction. If customer never returns from bank (e.g. closes browser), then status will remain Pending.

Credit card vendors use server-to-server call functionality to confirm the credit card transactions. This is used as fail-safe functionality, in case customer closes the browser after completing payment process at vendors site and never returns to Payment Gateway. Until this server-to-server call, TransactionStatus at Payment Gateway stays Pending.This is also a case, when customer returns from credit card vendor to Gateway correctly. Transaction will be confirmed only with server-to-server call and sometimes processing of credit card transaction at payment vendor may last a few minutes. Thus Application should always inform the customer of delay, and that her/his order is now being processed.

All regular banks have immediate response to customer's actions, and do not have this delay problem with transactions.

`Paid` – Transaction, which has been successfully paid at bank. Paid transactions status cant be changed.

`Cancelled` – Transaction, which has been cancelled by customer.

`Failed` – Transaction, which has been failed for some unknown reason. The reason for failure can be error in payment vendor's service or incorrect MD5 hash returned from payment vendor.

Request example:

```
<?xml version="1.0" encoding="UTF-8"?> <env:Envelope
xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
```

```
<env:Body>
<spg:CompletePurchase
xmlns:spg="http://www.smilehouse.com/SmilehousePaymentGateway">
  <Organization>shop2</Organization>
  <TransactionId>68</TransactionId>
  <Hash>9f12ecc56feb09edbb993c7e414c191d</Hash>
</spg:CompletePurchase>
</env:Body>
</env:Envelope>
```

Simple example how to execute CompletePurchase operation with java proxy classes

```
/**
* Make CompletePurchase request and get response
* @param stub
* @return CompletePurchaseResponse
* @throws RemoteException
* @throws GenericFault
*/
public CompletePurchaseResponse
getCompletePurchaseResponse(PaymentGatewayServiceStub stub) throws
Exception{
  CompletePurchase complete = new CompletePurchase();
  int transactionId = 123;
  // Calculate hash for message
  MD5 hash = new MD5(organization + transactionId + gatewayCheckKey);
  complete.setOrganization(organization);
  complete.setTransactionId(String.valueOf(transactionId));
  complete.setHash(hash.toString());
  return stub.CompletePurchaseOperation(complete);
}
```

Response example:

```
<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/">
<SOAP-ENV:Body>
<spg:CompletePurchaseResponse
xmlns:spg="http://www.smilehouse.com/SmilehousePaymentGateway">
  <spg:Code>OK </spg:Code>
  <spg:Description>OK</spg:Description>
  <spg:TransactionStatus>1</spg:TransactionStatus>
  <spg:TransactionId>68</spg:TransactionId>
  <spg:OrderId>12345</spg:OrderId>
  <spg:ProviderId>solo </spg:ProviderId>
  <spg:Amount>28958</spg:Amount>
  <spg:AmountExp>-2</spg:AmountExp>
   <spg:Hash>436808f85bec2176c21ee9b4ebc321af </spg:Hash>
</spg:CompletePurchaseResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

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

```
// ------------------------------------
```

```
// Response for CompletePurchase request
// -------------------------------------
// Initialize main GateWay Stub
PaymentGatewayServiceStub stub = new PaymentGatewayServiceStub(null,
gatewayURL);
CompletePurchaseResponse completeResponse =
gateway.getCompletePurchaseResponse(stub);
hashBuffer = new StringBuffer();
// Create MD5 hash string
hashBuffer.append(completeResponse.getCode());
hashBuffer.append(completeResponse.getDescription());
hashBuffer.append(completeResponse.getTransactionStatus());
hashBuffer.append(completeResponse.getTransactionId());
hashBuffer.append(completeResponse.getOrderId());
hashBuffer.append(completeResponse.getProviderId());
hashBuffer.append(completeResponse.getAmount());
hashBuffer.append(completeResponse.getAmountExp());
hashBuffer.append(gatewayCheckKey);
```

4. Check Order

Direction: Application -> Payment Gateway

Application can use this call for checking the transaction's status from the Payment vendor. **This call is not part of basic payment process.** This is helper functionality for merchants to check if customer actually has paid for the order. When initializing this call, Payment Gateway makes query to bank to check the status of requested transaction. Only some banks support order checking.

Soap Call Parameters

Field	Type	Description
Organization	String	Unique id per store, see call 2 for a description.
TransactionId	Integer	Reference for this transaction
Hash	String	MD5 hash of the above values

Soap Response Parameters

Field	Type	Description
Code	String	Response code (*)
Description	String	OK if OK, otherwise longer description (e.g unknown order)
TransactionStatus	Integer	A code indicating the success or failure level. *0 = Payment not found* *1 = Payment OK* 2 = Error in bank *3 = Failed return MAC* *5 = Check failed*

OrderId	String	The OrderId sent from shop in initialize method
ProviderId	String	id of payment provider
Amount	Integer	Amount charged, in lowest monetary unit (cents)
AmountExp	Integer	Amount exponent (-2 with cents)
Hash	String	MD5 hash of the above values

(*) Possible response codes:

7. `OK`
8. `transaction_not_found` - No transaction was found with the given TransactionId
9. `check_not_supported` - Payment check functionality is not supported with the payment provider used for this transaction.

Request example:

```
<?xml version="1.0" encoding="UTF-8"?> <env:Envelope
xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:enc="http://schemas.xmlsoap.org/soap/encoding/"
env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xs="http://www.w3.org/1999/XMLSchema">
<env:Body>
<spg:CheckOrder
xmlns:spg="http://www.smilehouse.com/SmilehousePaymentGateway">
<Organization>shop2 </Organization>
<TransactionId>68 </TransactionId>
<Hash>9f12ecc56feb09edbb993c7e414c191d </Hash>
</spg:CheckOrder>
</env:Body>
</env:Envelope>
```

Simple example how to execute CheckOrder operation with java proxy classes

```
/**
* Make CheckOrder request and get response.
* @param stub
* @return CheckOrderResponse
* @throws GenericFault
* @throws RemoteException
*/
public CheckOrderResponse getCheckOrderResponse(PaymentGatewayServiceStub
stub) throws Exeption {
  CheckOrder checkOrder = new CheckOrder();
  int transactionId = 123;
  // Calculate hash for message
  MD5 hash = new MD5(organization + transactionId + gatewayCheckKey);
  checkOrder.setOrganization(organization);
  checkOrder.setTransactionId("123");
  checkOrder.setHash(hash.toString());
  return stub.CheckOrderOperation(checkOrder);
}
```

Response example:

```
<SOAP-ENV:Envelope xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/"> <SOAP-ENV:Body>
<spg:CheckOrderResponse
xmlns:spg="http://www.smilehouse.com/SmilehousePaymentGateway">
<spg:Code>OK </spg:Code>
<spg:Description>OK </spg:Description>
```

```
<spg:TransactionStatus>1 </spg:TransactionStatus>
<spg:OrderId>12345 </spg:OrderId>
<spg:ProviderId>solo </spg:ProviderId>
<spg:Amount>28958 </spg:Amount>
<spg:AmountExp>-2 </spg:AmountExp>
<spg:Hash>0ef5be45d7af66046afe8434d6363c28 </spg:Hash>
</spg:CheckOrderResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

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

```
// -------------------------------
// Response for CheckOrder request
// -------------------------------
// Initialize main GateWay Stub
PaymentGatewayServiceStub stub = new PaymentGatewayServiceStub(null,
gatewayURL);
CheckOrderResponse checkResponse = gateway.getCheckOrderResponse(stub);
hashBuffer = new StringBuffer();
hashBuffer.append(checkResponse.getCode());
hashBuffer.append(checkResponse.getDescription());
hashBuffer.append(checkResponse.getTransactionStatus());
hashBuffer.append(checkResponse.getOrderId());
hashBuffer.append(checkResponse.getProviderId());
hashBuffer.append(checkResponse.getAmount());
hashBuffer.append(checkResponse.getAmountExp());
hashBuffer.append(gatewayCheckKey);
```

5. 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 Payment Gateway

For more information, please contact sales@smilehouse.com

6. Appendix 1: XML W3C Schematas

6.1.GetPaymentProviders 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/SmilehousePaymentGateway"
xmlns:ns1="http://www.smilehouse.com/SmilehousePaymentGateway">
  <xs:element name="GetPaymentProviders">
    <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>
```

6.2.GetPaymentProviders 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/SmilehousePaymentGateway"
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:spg="http://www.smilehouse.com/SmilehousePaymentGateway">
  <!-- Get available payment providers -->
  <xs:element name="GetPaymentProvidersResponse">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="spg:Code"/>
        <xs:element ref="spg:Description"/>
        <xs:element ref="spg:PaymentProviders"/>
        <xs:element ref="spg:Hash"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="Code" type="xs:string"/>
  <xs:element name="PaymentProviders">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="0" maxOccurs="unbounded"
ref="spg:PaymentProvider"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="PaymentProvider">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="spg:ProviderId"/>
        <xs:element ref="spg:Name"/>
        <xs:element ref="spg:Description"/>
        <xs:element ref="spg:Price"/>
        <xs:element ref="spg:VAT"/>
        <xs:element ref="spg:Currency"/>
        <xs:element ref="spg:AmountExp"/>
        <xs:element ref="spg:IconURL"/>
        <xs:element ref="spg:PaymentCode"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="ProviderId" type="xs:string"/>
  <xs:element name="Name" type="xs:string"/>
  <xs:element name="Price" type="xs:integer"/>
  <xs:element name="VAT" type="xs:double"/>
  <xs:element name="Currency" type="xs:string"/>
  <xs:element name="AmountExp" type="xs:integer"/>
  <xs:element name="IconURL" type="xs:anyURI"/>
  <xs:element name="PaymentCode" type="xs:string"/>
  <xs:element name="Hash" type="xs:string"/>
  <xs:element name="Description" type="xs:string"/>
</xs:schema>
```

6.3.Initialize Purchase 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/SmilehousePaymentGateway"
xmlns:ns1="http://www.smilehouse.com/SmilehousePaymentGateway">
  <xs:element name="InitializePurchase">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Organization" form="unqualified"
type="xs:string"/>
        <xs:element name="OrderId" form="unqualified" type="xs:string"/>
        <xs:element name="CallerId" form="unqualified" type="xs:string"/>
        <!-- ReferenceNumber is optional -->
        <xs:element name="ReferenceNumber" form="unqualified"
type="xs:string" minOccurs="0"/>
        <xs:element name="Description" form="unqualified"
type="xs:string"/>
        <xs:element name="ProviderId" form="unqualified"
type="xs:string"/>
        <xs:element name="Currency" form="unqualified" type="xs:string"/>
        <!-- Language is optional -->
        <xs:element name="Language" form="unqualified" type="xs:string"
minOccurs="0"/>
        <xs:element name="AmountExp" form="unqualified"
type="xs:integer"/>
        <xs:element name="FirstName" form="unqualified"
type="xs:string"/>
        <xs:element name="LastName" form="unqualified" type="xs:string"/>
        <xs:element name="StreetAddress" form="unqualified"
type="xs:string"/>
        <xs:element name="PostalCode" form="unqualified"
type="xs:string"/>
        <xs:element name="City" form="unqualified" type="xs:string"/>
        <xs:element name="Country" form="unqualified" type="xs:string"/>
        <xs:element name="EMail" form="unqualified" type="xs:string"/>
        <xs:element name="Phone" form="unqualified" type="xs:string"/>
        <xs:element name="Items" form="unqualified">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="Item" maxOccurs="unbounded"
form="unqualified">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="ItemName" form="unqualified"
type="xs:string"/>
        <xs:element name="ItemCode" form="unqualified" type="xs:string"/>
      <xs:element name="ItemPrice" form="unqualified" type="xs:integer"/>
     <xs:element name="Quantity" form="unqualified" type="xs:integer"/>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <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>
```

6.4.Initialize Purchase 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/SmilehousePaymentGateway"
xmlns:spg="http://www.smilehouse.com/SmilehousePaymentGateway">
  <xs:element name="InitializePurchaseResponse">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="spg:Code"/>
        <xs:element ref="spg:Description"/>
        <xs:element ref="spg:TransactionId"/>
        <xs:element ref="spg:ReferenceNumber"/>
        <xs:element ref="spg:Hash"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="Code" type="xs:string"/>
  <xs:element name="Description" type="xs:string"/>
  <xs:element name="TransactionId" type="xs:integer"/>
  <xs:element name="ReferenceNumber" type="xs:string"/>
  <xs:element name="Hash" type="xs:string"/>
</xs:schema>
```

6.5.Complete Purchase 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/SmilehousePaymentGateway"
xmlns:ns1="http://www.smilehouse.com/SmilehousePaymentGateway">
  <xs:element name="CompletePurchase">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Organization" form="unqualified"
type="xs:string"/>
```

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

6.6.Complete Purchase 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/SmilehousePaymentGateway"
xmlns:spg="http://www.smilehouse.com/SmilehousePaymentGateway">
  <xs:element name="CompletePurchaseResponse">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="spg:Code"/>
        <xs:element ref="spg:Description"/>
        <xs:element ref="spg:TransactionStatus"/>
        <xs:element ref="spg:TransactionId"/>
        <xs:element ref="spg:OrderId"/>
        <xs:element ref="spg:ProviderId"/>
        <xs:element ref="spg:Amount"/>
        <xs:element ref="spg:AmountExp"/>
        <xs:element ref="spg:Hash"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="Code" type="xs:string"/>
  <xs:element name="Description" type="xs:string"/>
  <xs:element name="TransactionStatus" type="xs:integer"/>
  <xs:element name="TransactionId" type="xs:integer"/>
  <xs:element name="OrderId" type="xs:string"/>
  <xs:element name="ProviderId" type="xs:string"/>
  <xs:element name="Amount" type="xs:integer"/>
  <xs:element name="AmountExp" type="xs:integer"/>
  <xs:element name="Hash" type="xs:string"/>
</xs:schema>
```

6.7.Check Order 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/SmilehousePaymentGateway"
xmlns:ns1="http://www.smilehouse.com/SmilehousePaymentGateway">
  <xs:element name="CheckOrder">
```

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

6.8.Check Order 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/SmilehousePaymentGateway"
xmlns:spg="http://www.smilehouse.com/SmilehousePaymentGateway">
  <xs:element name="CheckOrderResponse">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="spg:Code"/>
        <xs:element ref="spg:Description"/>
        <xs:element ref="spg:TransactionStatus"/>
        <xs:element ref="spg:OrderId"/>
        <xs:element ref="spg:ProviderId"/>
        <xs:element ref="spg:Amount"/>
        <xs:element ref="spg:AmountExp"/>
        <xs:element ref="spg:Hash"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="Code" type="xs:string"/>
  <xs:element name="Description" type="xs:string"/>
  <xs:element name="TransactionStatus" type="xs:integer"/>
  <xs:element name="OrderId" type="xs:string"/>
  <xs:element name="ProviderId" type="xs:string"/>
  <xs:element name="Amount" type="xs:integer"/>
  <xs:element name="AmountExp" type="xs:integer"/>
  <xs:element name="Hash" type="xs:string"/>
</xs:schema>
```

7. Appendix 2: Smilehouse Payment Gateway WSDL

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

PaymentGateway 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 PaymentGatewayPortStub.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 ("spg:") or default namespace set to URI "http://www.smilehouse.com/SmilehousePaymentGateway" by the command element (e.g. "GetPaymentProviders").
An example follows:

Correct:

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

Correct 2:

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

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

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

Current Payment 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="PaymentGateway"
    targetNamespace="http://www.smilehouse.com/SmilehousePaymentGateway"
    xmlns:tns="http://www.smilehouse.com/SmilehousePaymentGateway"
    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:spg="http://www.smilehouse.com/SmilehousePaymentGateway">
    <wsdl:documentation>Smilehouse Payment 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/SmilehousePaymentG
ateway"
            xmlns:SOAPENV="http://schemas.xmlsoap.org/soap/envelope/"
            xmlns:ns1="http://www.smilehouse.com/SmilehousePaymentGateway
"
            xmlns:spg="http://www.smilehouse.com/SmilehousePaymentGateway
">
            <xs:element name="GetPaymentProviders">
                <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:element name="GetPaymentProvidersResponse">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element ref="spg:Code"/>
                        <xs:element ref="spg:Description"/>
                        <xs:element ref="spg:PaymentProviders"/>
                        <xs:element ref="spg:Hash"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="Code" type="xs:string"/>
            <xs:element name="PaymentProviders">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element minOccurs="0" maxOccurs="unbounded"
ref="spg:PaymentProvider"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="PaymentProvider">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element ref="spg:ProviderId"/>
                        <xs:element ref="spg:Name"/>
                        <xs:element ref="spg:Description"/>
                        <xs:element ref="spg:Price"/>
                        <xs:element ref="spg:VAT"/>
                        <xs:element ref="spg:Currency"/>
                        <xs:element ref="spg:AmountExp"/>
                        <xs:element ref="spg:IconURL"/>
                        <xs:element ref="spg:PaymentCode"/>
                    </xs:sequence>
                </xs:complexType>
```

```
            </xs:element>
            <xs:element name="ProviderId" type="xs:string"/>
            <xs:element name="Name" type="xs:string"/>
            <xs:element name="Price" type="xs:integer"/>
            <xs:element name="VAT" type="xs:double"/>
            <xs:element name="Currency" type="xs:string"/>
            <xs:element name="AmountExp" type="xs:integer"/>
            <xs:element name="IconURL" type="xs:anyURI"/>
            <xs:element name="PaymentCode" type="xs:string"/>
            <xs:element name="Hash" type="xs:string"/>
            <xs:element name="Description" type="xs:string"/>

            <xs:element name="InitializePurchase">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="Organization"
form="unqualified" type="xs:string"/>
                        <xs:element name="OrderId" form="unqualified"
type="xs:string"/>
                        <xs:element name="CallerId" form="unqualified"
type="xs:string"/>
                        <!-- ReferenceNumber is optional -->
                        <xs:element name="ReferenceNumber"
form="unqualified" type="xs:string" minOccurs="0"/>
                        <xs:element name="Description" form="unqualified"
type="xs:string"/>
                        <xs:element name="ProviderId" form="unqualified"
type="xs:string"/>
                        <xs:element name="Currency" form="unqualified"
type="xs:string"/>
                <!-- Language is optional -->
                <xs:element name="Language" form="unqualified"
type="xs:string" minOccurs="0"/>
                        <xs:element name="AmountExp" form="unqualified"
type="xs:integer"/>
                        <xs:element name="FirstName" form="unqualified"
type="xs:string"/>
                        <xs:element name="LastName" form="unqualified"
type="xs:string"/>
                        <xs:element name="StreetAddress"
form="unqualified" type="xs:string"/>
                        <xs:element name="PostalCode" form="unqualified"
type="xs:string"/>
                        <xs:element name="City" form="unqualified"
type="xs:string"/>
                        <xs:element name="Country" form="unqualified"
type="xs:string"/>
                        <xs:element name="EMail" form="unqualified"
type="xs:string"/>
                        <xs:element name="Phone" form="unqualified"
type="xs:string"/>
                        <xs:element name="Items" form="unqualified">
                            <xs:complexType>
                                <xs:sequence>
                                    <xs:element name="Item"
maxOccurs="unbounded" form="unqualified">
                                        <xs:complexType>
                                            <xs:sequence>
                                                <xs:element
name="ItemName" form="unqualified" type="xs:string"/>
```

```
                                                <xs:element
name="ItemCode" form="unqualified" type="xs:string"/>
                                                <xs:element
name="ItemPrice" form="unqualified" type="xs:integer"/>
                                                <xs:element
name="Quantity" form="unqualified" type="xs:integer"/>
                                            </xs:sequence>
                                        </xs:complexType>
                                    </xs:element>
                                </xs:sequence>
                            </xs:complexType>
                        </xs:element>
                        <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="InitializePurchaseResponse">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element ref="spg:Code"/>
                        <xs:element ref="spg:Description"/>
                        <xs:element ref="spg:TransactionId"/>
                        <xs:element ref="spg:ReferenceNumber"/>
                        <xs:element ref="spg:Hash"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>

            <xs:element name="TransactionId" type="xs:integer"/>
            <xs:element name="ReferenceNumber" type="xs:string"/>

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

            <xs:element name="CompletePurchaseResponse">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element ref="spg:Code"/>
                        <xs:element ref="spg:Description"/>
                        <xs:element ref="spg:TransactionStatus"/>
                        <xs:element ref="spg:TransactionId"/>
                        <xs:element ref="spg:OrderId"/>
                        <xs:element ref="spg:ProviderId"/>
                        <xs:element ref="spg:Amount"/>
                        <xs:element ref="spg:AmountExp"/>
                        <xs:element ref="spg:Hash"/>
                    </xs:sequence>
```

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

            <xs:element name="TransactionStatus" type="xs:integer"/>
            <xs:element name="OrderId" type="xs:string"/>
            <xs:element name="Amount" type="xs:integer"/>

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

            <xs:element name="CheckOrderResponse">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element ref="spg:Code"/>
                        <xs:element ref="spg:Description"/>
                        <xs:element ref="spg:TransactionStatus"/>
                        <xs:element ref="spg:OrderId"/>
                        <xs:element ref="spg:ProviderId"/>
                        <xs:element ref="spg:Amount"/>
                        <xs:element ref="spg:AmountExp"/>
                        <xs:element ref="spg:Hash"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>

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

    <!-- SOAP command message content mapping to schemata -->
    <wsdl:message name="GetPaymentProvidersRequest">
        <wsdl:part name="body" element="spg:GetPaymentProviders"/>
    </wsdl:message>
    <wsdl:message name="GetPaymentProvidersResponse">
        <wsdl:part name="body"
element="spg:GetPaymentProvidersResponse"/>
    </wsdl:message>
    <wsdl:message name="InitializePurchaseRequest">
        <wsdl:part name="body" element="spg:InitializePurchase"/>
    </wsdl:message>
    <wsdl:message name="InitializePurchaseResponse">
        <wsdl:part name="body" element="spg:InitializePurchaseResponse"/>
```

```
    </wsdl:message>
    <wsdl:message name="CompletePurchaseRequest">
        <wsdl:part name="body" element="spg:CompletePurchase"/>
    </wsdl:message>
    <wsdl:message name="CompletePurchaseResponse">
        <wsdl:part name="body" element="spg:CompletePurchaseResponse"/>
    </wsdl:message>
    <wsdl:message name="CheckOrderRequest">
        <wsdl:part name="body" element="spg:CheckOrder"/>
    </wsdl:message>
    <wsdl:message name="CheckOrderResponse">
        <wsdl:part name="body" element="spg:CheckOrderResponse"/>
    </wsdl:message>

    <!-- Generic SOAP Fault message shared by all commands -->
    <wsdl:message name="GenericFault">
        <wsdl:part name="fault" element="spg:FaultDetail"/>
    </wsdl:message>

    <!-- Declaring the SOAP commands as Operations, mapping their
input/output/fault messages
         to above message definitions -->
    <wsdl:portType name="PaymentGatewayPort">
        <wsdl:operation name="GetPaymentProvidersOperation">
            <wsdl:input message="tns:GetPaymentProvidersRequest"/>
            <wsdl:output message="tns:GetPaymentProvidersResponse"/>
            <wsdl:fault name="GenericFault" message="tns:GenericFault"/>
        </wsdl:operation>
        <wsdl:operation name="InitializePurchaseOperation">
            <wsdl:input message="tns:InitializePurchaseRequest"/>
            <wsdl:output message="tns:InitializePurchaseResponse"/>
            <wsdl:fault name="GenericFault" message="tns:GenericFault"/>
        </wsdl:operation>
        <wsdl:operation name="CompletePurchaseOperation">
            <wsdl:input message="tns:CompletePurchaseRequest"/>
            <wsdl:output message="tns:CompletePurchaseResponse"/>
            <wsdl:fault name="GenericFault" message="tns:GenericFault"/>
        </wsdl:operation>
        <wsdl:operation name="CheckOrderOperation">
            <wsdl:input message="tns:CheckOrderRequest"/>
            <wsdl:output message="tns:CheckOrderResponse"/>
            <wsdl:fault name="GenericFault" message="tns:GenericFault"/>
        </wsdl:operation>
    </wsdl:portType>

    <!-- Bind PaymentGateway Operations to PaymentGatewayPort -->
    <wsdl:binding name="PaymentGatewayPort"
type="tns:PaymentGatewayPort">
        <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http"/>
        <wsdl:operation name="GetPaymentProvidersOperation">
            <soap:operation
                soapAction="http://www.example.com/workspace.client_examp
leshop/PaymentGateway"
                style="document"/>
            <wsdl:input>
                <soap:body use="literal"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                    namespace="http://www.smilehouse.com/SmilehousePaymen
tGateway"/>
            </wsdl:input>
```

```
            <wsdl:output>
                <soap:body use="literal"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                    namespace="http://www.smilehouse.com/SmilehousePaymen
tGateway"/>
            </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/SmilehousePaymen
tGateway"/>
            </wsdl:fault>
        </wsdl:operation>
        <wsdl:operation name="InitializePurchaseOperation">
            <soap:operation
                soapAction="http://www.example.com/workspace.client_examp
leshop/PaymentGateway"
                style="document"/>
            <wsdl:input>
                <soap:body use="literal"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                    namespace="http://www.smilehouse.com/SmilehousePaymen
tGateway"/>
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                    namespace="http://www.smilehouse.com/SmilehousePaymen
tGateway"/>
            </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/SmilehousePaymen
tGateway"/>
            </wsdl:fault>
        </wsdl:operation>
        <wsdl:operation name="CompletePurchaseOperation">
            <soap:operation
                soapAction="http://www.example.com/workspace.client_examp
leshop/PaymentGateway"
                style="document"/>
            <wsdl:input>
                <soap:body use="literal"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                    namespace="http://www.smilehouse.com/SmilehousePaymen
tGateway"/>
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                    namespace="http://www.smilehouse.com/SmilehousePaymen
tGateway"/>
            </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/SmilehousePaymen
tGateway"/>
            </wsdl:fault>
        </wsdl:operation>
        <wsdl:operation name="CheckOrderOperation">
            <soap:operation
                soapAction="http://www.example.com/workspace.client_examp
leshop/PaymentGateway"
                style="document"/>
            <wsdl:input>
                <soap:body use="literal"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                    namespace="http://www.smilehouse.com/SmilehousePaymen
tGateway"/>
            </wsdl:input>
                        <wsdl:output>
                <soap:body use="literal"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
                    namespace="http://www.smilehouse.com/SmilehousePaymen
tGateway"/>
            </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/SmilehousePaymen
tGateway"/>
            </wsdl:fault>
        </wsdl:operation>
    </wsdl:binding>

    <!-- Finally we declare the PaymentGateway service -->
    <wsdl:service name="PaymentGatewayService">
        <wsdl:port name="PaymentGatewayPort"
binding="tns:PaymentGatewayPort">
            <documentation>Smilehouse Payment Gateway SOAP
API</documentation>
            <soap:address
                location="http://www.example.com/workspace.client_example
shop/PaymentGateway"/>
        </wsdl:port>
    </wsdl:service>

</wsdl:definitions>
```

8. Appendix 3: Payment language settings

!NB Language is optional parameter at transaction initialization request, so if you don't want to, you dont need to use it in SOAP requests. That is because not all payments support this setting.

If the payment supports different languages, and Language parameter isn't used at transaction initialization request, then Gateway uses language set at Gateway's payment administration interface as default.

While sending Language parameter to Payment Gateway you should use the following available language codes.

Language name	Variable value
Danish	da
Swedish	sv
Norway	no
Dutch	nl
German	de
French	fr
Finnish	fi
Spanish	es
Italy	it
English	en
Estonia	ee
Russian	ru
Latvian	lv

Next table displays language codes available for every payment method. If the payment is not represented in the list, then it doesnt support the language option at all.

Payment Name	Languages supported by this payment
Finland Luottokunta	fi, en, sv
Finland Nordea Solo	fi, en, sv
Finland Sampo	fi, en, sv
Finland AlandsBanken	fi, sv
Finland Tapiola	fi, sv
Finland S-Pankki	fi, sv
Estonia Hansa Bank	ee, en
Estonia Uhisbank(Swedbank)	ee, en
Estonia Sampo	ee, en
Estonia Krediidipank	ee, ru, en
Estonia Nordea	ee, en
Estonia Estcard	ee, en
Latvia Hansabanka	lv, ru, en
Latvia Nordea	lv, en
Danish Nordea	da, en
International Ogone	en, fr, nl, de, es, no
International Dibs	da, sv, no, en, nl, de, fr, fi, es, it

International 2Checkout	en, es
Swedish Handelsbanken	en, sv

9. Setting up 3rd party libraries

Payment 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