CLASSIC PAYMENT API

SOPG (Service Oriented Prepaid Gateway - xml based protocol) Documentation

Version history

Version	Date	Description	Author
0.1	2013-10-03	Initial draft	Paul Kneidinger
0.2	2013-20-03	Added details and restructured	Matthias Vilsecker
0.3	2013-21-03	Added In App payment	Matthias Vilsecker
0.4	2013-24-04	Review and further input	Natasa Jeremic
1.0	2013-02-10	Final document	Natasa Jeremic
1.1	2014-27-01	Minor changes	Natasa Jeremic

For technical questions about the implementation please contact integration@paysafecard.com

Content

1. Introduction

This document gives a detailed overview about the usage and parameters of paysafecard's Service Oriented Prepaid Gateway (SOPG). The gateway is a SOAP XML Web Service which exposes API client functionalities that can be used with any SOAP capable client system.

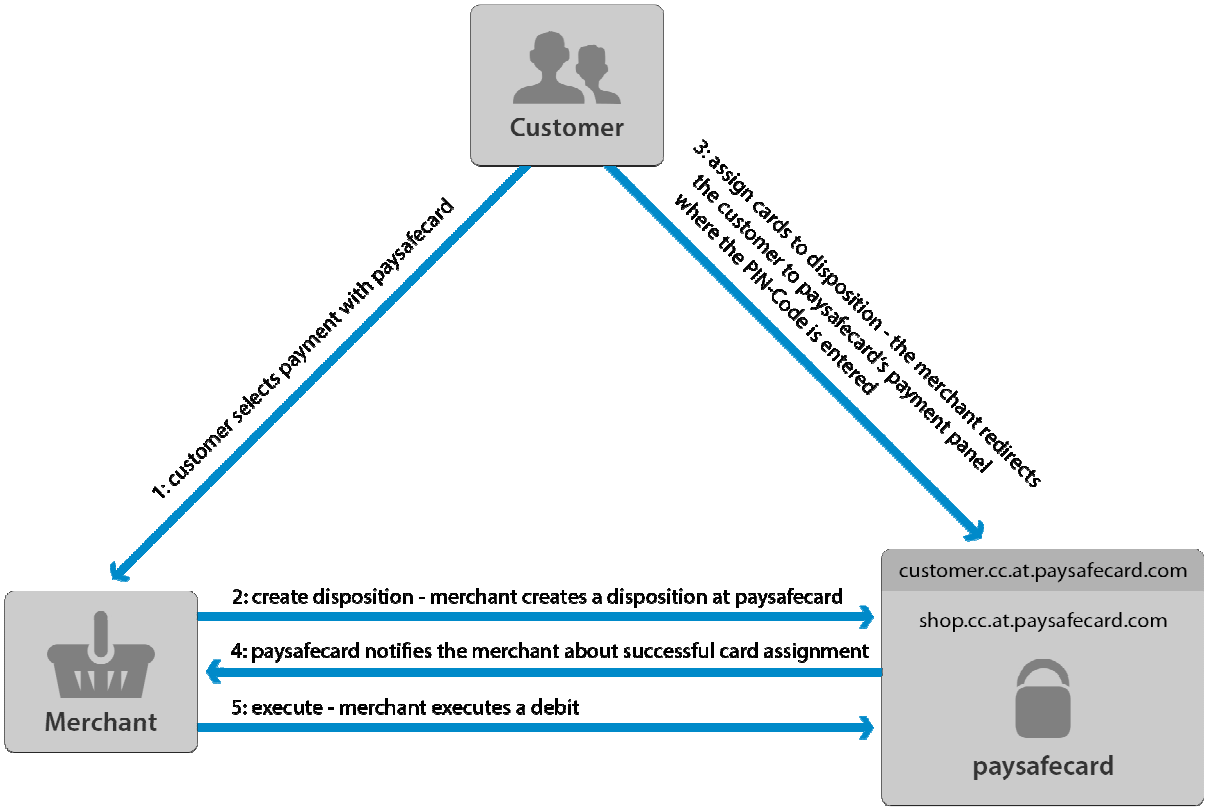
In the first chapter, a functional overview of the payment process is given. After that, paysafecard's SOPG is presented in detail. Then, the systems of paysafecard, as well as the technical details with a complete description of functions and parameters, are given.

An example implementation is given at the end of the document as well as an introduction to "inApp" payment within Android applications.

2. Functional Overview Payment

In each payment there are three parties involved: a customer, a webshop business partner and the paysafecard company ('PSC').
Payments take place in 'payment transactions' or 'dispositions', which are uniquely identified by a 'merchant transaction ID' (MTID) and hold a value called 'amount' that is typically the amount of money for which a customer buys something.

2.1 How payments work



2.2 my paysafecard

'my paysafecard' is:

- A payment account that allows paysafecard customers to register themselves and pay with a username and password; and
- A customer-friendly way that makes managing PINs easier and provides a clear overview.

For the integration, there is no difference in the payment process if the customer decides to pay with a 'my paysafecard' account.

2.2.1 KYC Levels

my paysafecard knows two different KYC Levels:

- **Simple**

  The customer has successfully done the initial registration and has confirmed a mobile phone number and an email address.

- **Full**

  If the customer wants to extend the limits, he can upgrade by following an identification process where the following documents are verified (this process may differ in different countries according to the different legal frameworks):

  - Identification document containing a photograph and names issued from a governmental entity (such as a passport, an approved driving licence, an approved national ID card); or
  - Proof of address such as a tax bill or a utility bill (gas, electric, water, cable, etc.), for example.

3. About SOPG

Service Orientated Prepaid Gateway (SOPG) provides paysafecard payment features as a web service. This is based on the SOAP protocol and can be used by any SOAP client, regardless of the programming language used.
The complete payment process is handled between the business partner-side system and paysafecard SOPG.

3.1 Prerequisites

A business partner can only connect to paysafecard's systems if the following prerequisites are fulfilled:

- Login credentials (username/password) provided by paysafecard
- Authorisation of the payment server IP address (if a '403 error' is received when trying to access the service, it is likely that the IP address is not yet allowed to access).

3.2 Command and Query Operations

The operations can be divided into query and command operations. A command operation will modify the state of an object, whereas a query operation will return details about the current state of an object.

Command operations in SOPG are protected against unwanted side-effects of repeated executions (e.g., calling 'executeDebit' twice on a transaction will return an error code on the second invocation).

3.3 Error Handling

All SOPG operations will return an 'errorCode' and 'resultCode'. A 'resultCode' can have a value of '0' (successful), '1' (logical problem) or '2' (technical problem). In general, the following rules can be applied:

- **'1'** indicates that there is a problem with the submitted data (e.g., wrong credentials, transaction has expired, etc.) retries with the same request data won't be successful.
- **'2'** (technical problem) means that the service is temporarily not available – the request can be retried.

3.3.1 Error Messages Description

Please consider that error messages to the customer will only be shown on the paysafecard payment panel. For other messages, only the error code will be provided.

%1, %2,… are only place holders for several values, e.g., MTID, MID.
All error codes and messages can be found in chapter 8.

3.4 Content-Type and Charset

Please make sure that the content type in the http header, when submitting requests, is set to:

Content-Type: text/xml;charset=UTF-8

4. Definition of paysafecard systems

4.1 Test Environment (M-Test)

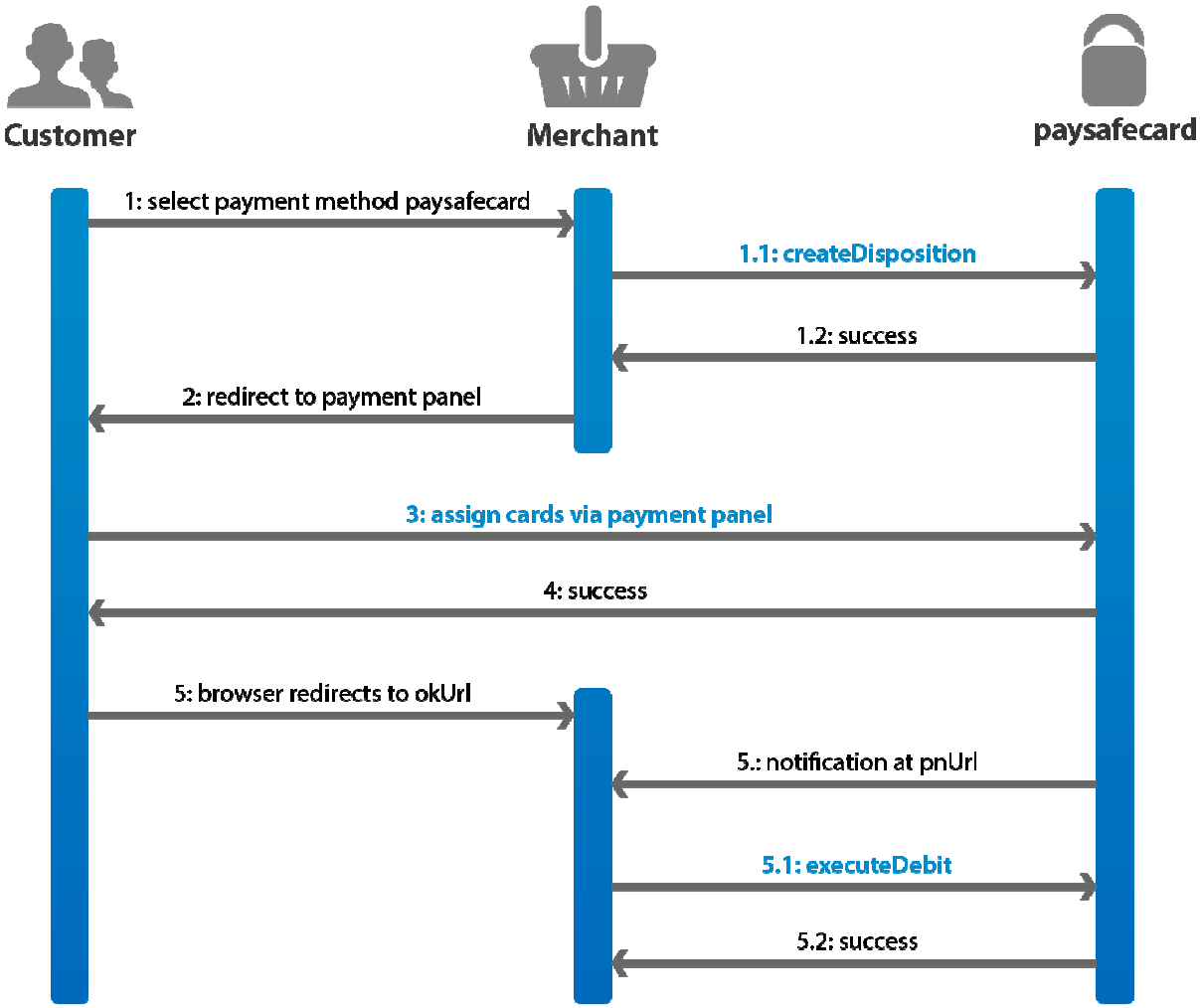
paysafecard provides the ‘paysafecard test system’ (SOATEST), a test environment for integration of new business partners. Every business partner is integrated in this system first where the Partner Integration and Support Team provides support to new business partners as well as performing the acceptance tests.

Once the integration is accepted, the business partner can be moved to the productive systems which, for the business partner, consist of the following steps:

- Update to productive credentials (all provided by paysafecard)
- Exchange of the Service Endpoint URL
- Exchange of the WSDL URL

All data are provided by the Partner Integration and Support Team.

5. Technical Overview



6. Operations details and WSDL contract

Although the SOPG WSDL service contract includes many more operations, all required operations for the basic payment process are described in this document.
All necessary payment parameters are required, and a transmission is obligatory although the value remains NULL.
If the webservice framework requires a WSDL at runtime, the SOPG WSDL from the WSDL URL needs to be downloaded and provided in the local environment.
NOTE: WSDL must not be fetched from paysafecard servers at runtime.

6.1 Functions

Operation Name	Type[1]	Description	Request Elements	Response Elements
createDisposition	C	The business partner initiates the payment process by sending a 'createDisposition' request to paysafecard to create a disposition at the server. Maximum allowed amount is 1000.00 EUR (or equivalent in different transaction currency)	username **[required]**	mtid, subId, mid, resultCode, errorCode
			password **[required]**	
			mtid **[required]**	
			subId **[required]**	
			amount **[required]**	
			currency **[required]**	
			okUrl **[required]**	
			nokUrl **[required]**	
			merchantclientId **[required]**	
			pnUrl **[required]**	
			clientIp **[required]**	
			dispositionrestrictions **[required]**	
			shopId **[required]**	
			shoplabel **[required]**	

[1] Q = Query, C = Command

getCustomerPanel	C	Allows the business partner to deliver the paysafecard payment landing page to the customer. The customer is forwarded to 'okURL' in case of a successful card assignment. In case the customer hits the cancel button, the customer is forwarded to 'nokURL'.	mid [required] mtid [required] amount [required] currency [required] language [optional] locale [optional]	-
executeDebit	C	Withdraws the money from the customer's paysafecard. This step concludes the payment if the 'close' flag is set to '1'. The business partner can keep the transaction open (while the full amount is reserved) until the end of the disposition time. If the business partner wants to close the disposition without executing anything, the function must be called with 'amount=0.00'	username **[required]** password **[required]** mtid **[required]** subId **[required]** amount **[required]** currency **[required]** close **[required]** partialDebitId **[optional]**	mtid, subId, resultCode, errorCode
getMid (optional)	Q	The business partner can query the assigned MID for the requested currency (unique ID of the business partner).	username **[required]** password **[required]** currency **[required]**	currency, mid, resultCode, errorCode

getSerialNumbers (optional)	Q	Gets the state of the disposition to verify the expected state before calling the next function in case peer implementation doesn't allow reacting on calls to 'okURL', 'nokURL' or 'pnURL'	username **[required]**	mtid, subId, resultCode, errorCode, amount, currency, dispositionState, serialNumbers
			password **[required]**	
			mtid **[required]**	
			subId **[required]**	
			currency **[required]**	
modifyDispositionValue (optional)	C	Option of reducing the originally disposed amount	username **[required]**	mtid, subId, resultCode, errorCode
			password **[required]**	
			mtid **[required]**	
			subId **[required]**	
			amount **[required]**	
			currency **[required]**	

6.2 Payment Notification

The payment notification is used to notify the business partner independently from the customer's behaviour after the card assignment. This service ensures that dispositions can be completed before the top-up of the customer's account and is therefore highly recommended in order to avoid incomplete transactions.

API parameter	Type[1]	Description	Output parameters	Merchant response elements
pnURL	C	payment notification is only delivered in case of a successful card assignment payment notification is submitted in addition to the 'okURL' redirection payment notification is resubmitted in case of technical or application errors on the business partner'sside	mtid	HTTP 200
			eventType (ASSIGN_CARDS is returned)	
			serialNumbers	
			currency	
			disposition amount	
			cardTypeId[2]	

[1] Q = Query, C = Command

[2] The parameter provides a combination of default ISO country code (country a paysafecard has been sold in) and the ID of the card type (defines the type of the paysafecard, e.g., 'paysafecard junior'). More information can be found in Appendix B: Payment notification supported country codes.

6.2.1 Resubmission of the payment notification

In case of technical errors (e.g., Socket timeout) or application errors (e.g., HTTP 500 response) the payment notification is resubmitted at a regular interval until one of the following criteria is fulfilled:

- The payment notification is successfully delivered (i.e., HTTP 200 response from payment server)
- The maximum number of retry attempts has been reached (currently configured 5 retries).

6.3 Parameters

6.3.1 Description of Parameters

username – custom account username

- provided by paysafecard for the authentication

password – custom account password

- provided by paysafecard for the authentication

mtid – **(unique)** transaction id, unique identifier for each disposition

- max. length: 60 characters
- recommended value: up to 20 characters
- provided by business partner
- only the following is allowed: A-Z, a-z, 0-9 as well as – (hyphen) and _ (underline)
- example: 3516-6s4dfsad41

subId – mandatory parameter, value must be left empty if nothing else is agreed

- so-called 'reporting criteria', offers the possibility to classify transactions
- max. length: 8 characters (case sensitive)
- agreement with paysafecard needed
- example: shop1

amount – disposition amount

- requested amount is not allowed to exceed 1000.00 EUR (or equivalent in a different transaction currency) in value
- max. 11 digits before – exactly 2 digits after the decimal point
- use a point as a decimal separator
- example: 100.00

currency – disposition currency

- max. length: 3 characters, all uppercase
- ISO Currency Code
- example: EUR

pnUrl – payment notification URL at which paysafecard notifies the business partner as soon as an assignment was successfully performed (more details in chapter 6.2)

- URL has to be absolute and URL encoded as they are sent as a parameter
- URL has to be provided by the business partner
- max. length: 765 characters

okUrl – is the URL to which the customers are forwarded by paysafecard after they have successfully assigned their paysafecards and completed the payment. The business partner may include some information in the URL.

- URL has to be absolute and URL encoded as they are sent as a parameter
- max. length: 765 characters

nokUrl – this is the URL to which customers are forwarded by paysafecard when they hit 'cancel' button on the paysafecard payment panel

- URL has to be absolute and URL encoded as they are sent as a parameter
- max. length: 765 characters

NOTE: It is crucial to send the **'okURL'**,**'nokURL'** and optional **'pnUrl'** in an URL encoded (also called percent-encoding) form. Otherwise, the result will be a wrong redirect of the customer to the confirmation page, in addition to a possible failure of the payment.

merchantclientId – a unique end customer identifier (the unique ID of the end customer as registered within the business partner database, for example). If using e-mail address, please encrypt the values.

For promotional activities, paysafecard checks the clientId and avoids multiple redemptions.

- **NOTE:** for security reasons do not use the customer's registered username
- max. length: 50 characters
- example: client123

clientIp – the IP address of the paysafecard customer

shopId – identification of the shop which is the originator of the request. This is most likely used by payment service providers who act as a proxy for other payment methods as well.

- max. length: 60 characters
- recommended value: up to 20 characters
- provided by business partner
- only the following is allowed A-Z, a-z, 0-9 as well as – (hyphen) and (underline)
- example: 2568-B415rh_785

shopLabel – label or URL of the shop which is the originator of the request, related to the 'shopId'. This is most likely used by payment service providers which act as a proxy for other payment methods as well.

- max. length: 60 characters
- example: www.foodstore.com

mid – merchant ID, unique ID of the merchant/currency pair

- 10 digits long
- provided by paysafecard
- example: 1000001234

dispositionState – current state of the disposition (see chapter 6.3.3 for more details).

serialNumbers – serial number(s) of assigned paysafecard(s) by the customer, after entering the PIN at the paysafecard payment panel (semicolon separated details).

- currency: ISO Currency Code
- disposition amount: reserved amount of the customer's paysafecard associated with this disposition
- cardTypeId: paysafecards are grouped into card types: e.g., junior_paysafecard; adult_paysafecard; inhouse_paysafecard
- examples:
  0000000001200000;EUR;7.50;00002;
  0000000001300000;EUR;5.50;00002;

close – the close flag of the disposition can be '0' or '1' to indicate if further actions will be executed or not.

- '0' [don't close transaction]
- '1' [close transaction, this is the last debit]

partialDebitId – this parameter allows the classification of partial debit.

resultCode – result code of the operation (see the result codes chapter for details).

errorCode – error code of the operation (see the error codes chapter for details).

dispositionRestrictions – disposition restrictions can be set by the business partner in order to restrict a payment transaction, according to their individual needs. See chapter 6.3.2 for details.

- multiple repeats possible
- each restriction is a pair of key and value:
- key – the name of the restriction
- value – the value of the restriction

6.3.2 Restrictions

Please provide with createDisposition API request a country code value (ISO 3166-1) to restrict payment to desired country.

key	value example	possible values	description
COUNTRY	DE	all countries where paysafecard is distributed (example: FR, ES,…)	Restricts the payment to be processed exclusively from country Germany. The value accepts ISO 3166-1 country codes.

The following restrictions are available for a paysafecard payment with a paysafecard account (my paysafecard):

key	value example	possible values	description
MIN_AGE	18	must be a positive number value	Restricts my paysafecard user account holder to be at least 18 years old.
MIN_KYC_LEVEL	FULL	SIMPLE or FULL	Restricts my paysafecard user account holder to be at least in the state, i.e. simple.

6.3.3 Disposition States

One letter code	Meaning	Description
R	Created	The disposition has been successfully created. If nothing happens within 30 minutes, the disposition will be transferred to state 'X' by the paysafecard cleanup job.
S	Disposed	The customer's paysafecard has been successfully assigned to the disposition. The business partner can 'executeDebit'; no debits have taken place so far.
O	Consumed	The final debit with close=1 has been called; no further debits are possible.
L	Cancelled	The disposition has been actively cancelled by the customer.
X	Expired	The time window for this disposition has ended (either before a paysafecard was assigned or before 'executeDebit' was called).
E	Debited	Partially debited: the disposition is still open; further debits are possible.

6.4 Disposition Time Window

Once a disposition is in state 'S' ('DISPOSED'), the business partners have to perform their debits within a certain time period (disposition time). The disposition time window is configurable by per MID. If this time window is exceeded, the disposition will automatically expire, and the amount will be available again on the customer's paysafecard.

Furthermore, all dispositions which have been created, but not successfully debited, will be set to 'EXPIRED'.

NOTE: These jobs are only active on the productive paysafecard servers.

6.5 Payment Panel Details

6.5.1 Payment Panel Desktop

The paysafecard payment panel can be presented in a popup window or, alternatively, in an iFrame.
To make sure the whole payment panel is visible to the user, please always allow vertical scrolling or dynamic sizing.

- The width is fixed to **600px**
- The height is fixed up to **1040px**

6.5.2 Payment Panel Mobile

paysafecard's payment panel is optimised for mobile devices. If a customer is using a device with a smaller resolution than 600px, an optimised payment panel will be shown. This is also the case if the embedded iframe has a smaller width than 600px.

NOTE: The iframe for embedding the desktop payment panel always needs to have at least a width of 600px. Otherwise, the mobile version of the payment panel will be displayed.

6.6 Locale and language settings

For backward compatibility, all existing language parameters still yield the same result as in former versions of the API, but every language will be automatically transformed into a locale.

Basically, the language and locale of the payment panel are determined by following rule:

1. Has the customer already visited the payment panel? Take the locale from the set cookie.
2. Take the locale from the IP address of the customer.[1]
3. **Take the value from the locale parameter.**
4. Take the value from the language parameter.
5. Take the locale from the browser header.
6. Take the fallback locale (de_de).

Therefore, it is not obligatory to set a locale parameter.

[1] paysafecard uses a GeoIP service check.

7. Example Payment

In this chapter a test scenario is presented using example data. In practice, it will differ from business partner to business partner whether one or more debits or status-requests will be executed.

Do not use the enclosed example data! Each business partner will receive a consistent set of test data for testing purposes.

7.1 createDisposition

The **business partner** initiates the payment process by sending a 'createDisposition' request.

Example Request:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:urn="urn:pscservice">
 <soapenv:Header/>
 <soapenv:Body>
 <urn:createDisposition>
 <urn:username>USER</urn:username>
 <urn:password>PASSWORD</urn:password>
 <urn:mtid>18b02d230-a6822f-4cbb-aee9-0bc07d90cfa4</urn:mtid>
 <!--Zero or more repetitions:-->
 <urn:subId></urn:subId>
 <urn:amount>10.00</urn:amount>
 <urn:currency>EUR</urn:currency>
 <urn:okUrl>http%3a%2f%2fwww%2epaysafecardokURL%2ecom</urn:okUrl>
 <urn:nokUrl>http%3a%2f%2fwww%2epaysafecardnokURL%2ecom</urn:nokUrl>
 <!--Optional:-->
 <urn:merchantclientid>cID_919191</urn:merchantclientid>
 <!--Optional:-->
 <urn:pnUrl> http%3a%2f%2fwww%2emerchantpnURL%2ecom </urn:pnUrl>
 <!--Zero or more repetitions:-->
 <urn:dispositionRestrictions>
 <urn:key>COUNTRY</urn:key>
 <urn:value>FR</urn:value>
 </urn:dispositionRestrictions>
 <urn:dispositionRestrictions>
 <urn:key>MIN_AGE</urn:key>
 <urn:value>18</urn:value>
 </urn:dispositionRestrictions>
 <!--Optional:-->
 <urn:shopId>3516-6s4dfsad41</urn:shopId>
 <!--Optional:-->
 <urn:shopLabel>www.foodstore.com</urn:shopLabel>
 </urn:createDisposition>
 </soapenv:Body>
</soapenv:Envelope>
```

Example Response:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
 <soapenv:Body>
 <ns1:createDispositionResponse xmlns:ns1="urn:pscservice">
 <ns1:createDispositionReturn>
 <ns1:mtid>18b02d230-a6822f-4cbb-aee9-0bc07d90cfa4</ns1:mtid>
 <ns1:mid>1000001234</ns1:mid>
 <ns1:resultCode>0</ns1:resultCode>
 <ns1:errorCode>0</ns1:errorCode>
 </ns1:createDispositionReturn>
 </ns1:createDispositionResponse>
 </soapenv:Body>
</soapenv:Envelope>
 </soapenv:Body>
</soapenv:Envelope>
```

7.2 getMid (optional)

With ‘getMid’, the **business partner** can query the assigned unique merchant identifier (MID) for the requested currency.

Example Request:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:urn="urn:pscservice">
 <soapenv:Header/>
 <soapenv:Body>
 <urn:getMid>
 <urn:username>USER</urn:username>
 <urn:password>PASSWORD</urn:password>
 <urn:currency>EUR</urn:currency>
 </urn:getMid>
 </soapenv:Body>
</soapenv:Envelope>
```

Example Response:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
 <soapenv:Body>
 <ns1:getMidResponse xmlns:ns1="urn:pscservice">
 <ns1:getMidReturn>
 <ns1:currency>EUR</ns1:currency>
 <ns1:mid>1000001234</ns1:mid>
 <ns1:resultCode>0</ns1:resultCode>
 <ns1:errorCode>0</ns1:errorCode>
 </ns1:getMidReturn>
 </ns1:getMidResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

7.3 getCustomerPanel

The command ‘createDisposition’ was successfully executed. Thus, the customer can be forwarded to the paysafecard payment panel for assigning cards to the disposition.

Example URL Test System

```
https:// customer.test.at.paysafecard.com/psccustomer/GetCustomerPanelServlet
```

Example URL Productive System

```
https:// customer.cc.at.paysafecard.com/psccustomer/GetCustomerPanelServlet
?mid=1000001234
&mtid=18b02d230-a6822f-4cbb-aee9-0bc07d90cfa4
&amount=10.00
&currency=EUR
```

Example Input Parameters

PIN:	0000 0000 1234 5678
Terms of Use:	<checkbox, default unchecked>

7.4 pnUrl request

paysafecard system sends an ‘HTTP POST’ request to the business partners’s system (‘pnUrl’) in order to give notice of the successful assignment of the customer’s paysafecards.

Example URL

```
http://www.merchantpnURL.com/notifyME
?mtid=3516-6s4dfsad41
&eventType=ASSIGN_CARDS
&serialNumbers=0000000001200000;EUR;100.00;DE00002
```

Example Response

```
HTTP 200
```

7.4.1 executeDebit

After customer successfully assignedthe cards to the disposition, the **business partner** executes the debit to withdraw the money from the customer's paysafecard.

Example Request:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:urn="urn:pscservice">
 <soapenv:Header/>
 <soapenv:Body>
 <urn:executeDebit>
 <urn:username>USER</urn:username>
 <urn:password>PASSWORD</urn:password>
 <urn:mtid>18b02d230-a6822f-4cbb-aee9-0bc07d90cfa4</urn:mtid>
 <!--Zero or more repetitions:-->
 <urn:subId></urn:subId>
 <urn:amount>10.00</urn:amount>
 <urn:currency>EUR</urn:currency>
 <urn:close>1</urn:close>
 </urn:executeDebit>
 </soapenv:Body>
</soapenv:Envelope>
```

Example Response:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
 <soapenv:Body>
 <ns1:executeDebitResponse xmlns:ns1="urn:pscservice">
 <ns1:executeDebitReturn>
 <ns1:mtid>18b02d230-a6822f-4cbb-aee9-0bc07d90cfa4</ns1:mtid>
 <ns1:subId/>
 <ns1:resultCode>0</ns1:resultCode>
 <ns1:errorCode>0</ns1:errorCode>
 </ns1:executeDebitReturn>
 </ns1:executeDebitResponse>
 </soapenv:Body>
</soapenv:Envelope>
```

8. Appendix A: Errorcodes

If an error code appears that is not listed here, please contact integration@paysafecard.com

Description of Result Codes:

Result Name	Description
resultcode	0 : successful 1 : logical problem 2 : technical problem
errorcode	Contains an error number if the resultcode is not equal to 0.

```
# general messages - errors: 0001 - 0141
1=No data selected.
2=%1 is not numeric.
3=Mandatory field %1 is empty.
4=Decimal field with name %1 and value %2 has no decimal point.
5=Decimal field with name %1 and value %2 has no digits before the decimal point.
6=Decimal field with name %1 and value %2 has too many digits before the decimal point (max. %3
        allowed).
7=Decimal field with name %1 and value %2 has too few digits after the decimal point (must have %3).
8=Decimal field with name %1 and value %2 has too many digits after the decimal point (max. %3
        allowed).
9=Decimal field with name %1 and value %2 is not a number with format N.M (where N are 1 to %3
        digits, M are exactly %4 digits and M and N are numeric).
10=Decimal field with name %1 and value %2 has no digits after the decimal point (must have at least 1
        and at most %3).
11=Decimal field with name %1 and value %2 must not be negative.
12=Decimal field with name %1 and value %2 is not a number with format N.M (where N are 1 to %3
        digits, M are 1 to %4 digits and M and N are numeric).
13=Decimal field with name %1 is empty.
14=Cannot process more than %1 objects per transaction.
15=Answer to Challenge Question is empty.
16=Answer to Challenge Question is wrong.
17=Answer to Challenge Question contains invalid characters.

20=Challenge Question is empty.
21=Challenge Question with value %1 is too long (max. %2 characters are allowed).

50=Merchant ID is empty.
51=Merchant ID with value %1 is too long (max. %2 characters are allowed).

55=Merchant-transaction ID is empty.
56=Merchant-transaction ID with value %1 is too long (max. %2 characters are allowed).

60=nokURL is empty.

65=okURL is empty.

75=Serial number is empty.
76=Serial number with value %1 is too long (max. %2 characters are allowed).
77=Serial number %1 is not numeric.

80=Card State %1 is invalid.
81=Submitted Card State %1 of field %2 is not equal to the expected Card State %3.
```

```
85=Card Type %1 is invalid.

90=Debit State %1 is invalid.
95=Disposition State %1 is invalid.
96=Submitted Disposition State %1 of field %2 is not equal to the expected Disposition State %3.

120=Close Debit-flag %1 is invalid (must be 0 or 1).

125=Currency is empty.
126=Currency with value %1 has invalid length (must have 3 characters).

140=Currency Name is empty.
141=Currency Name with value %1 is too long (max. %2 characters are allowed).

# general messages - success messages 0601 - 0603
601=The command completed successfully.
602=The command completed successfully; no data found.
603=The command completed successfully; more data match filter criteria (change filter criteria to
        reduce amount of data returned).

# card messages - error messages: 1001 - 1600
1004=Card with Serial Number %1 has an unexpected state %2, expected is %3.
1005=Card with Serial Number %1 has not a location '%3', but is at '%2'.
1006=Card with Serial Number %1 is not assigned to %2.
1007=Card with Serial Number %1 does not exist.
1008=Access denied.
1009=%1 is not allowed to have Cards assigned.
1012=Card State %1 is not valid for this request; expected Card State is %2.
1015=Access denied because of a repeated recent access violation.
1020=Challenge Question Answer 1 and Challenge Question Answer 2 are different.
1025=Card is in an invalid State, expected State is 'GENERATED'.
1026=Number of copies printed is invalid; Card is set to State 'INVALID'.
1029=You need to specify question, answer and answer verification to set the Challenge Question.
1035=The card status of at least one card used for this inquiry is not valid.
1046=There is currently no available credit on this card; in the case of reserved amounts.
1049=At least one of the PINs used is not valid.
1050=Card does not exist

# payment messages - error messages: 2001 - 2600
2001=Transaction (%1/%2) already exists. Please contact your webshop.
2002=Transaction (%1/%2) does not exist. Please contact your webshop.
2003=Transaction (%1/%2) is in invalid state %3; expected is %4.
2004=Insufficient funds for payment; open amount is %1.
2006=Transaction currency %1 is invalid for transaction (%2/%3). Please contact your webshop.
2007=The amount %1 is invalid for the used card.
2008=The amount %1 exceeds the card balance.
2009=The amount %1 is invalid for the transaction (%2/%3). Please contact your webshop.
2010=The amount %1 is insufficiently disposed for the transaction (%2/%3).
2011=The Currency %1 is invalid for this transaction; expected is %2.
2012=Payment transaction failed.
2013=Error finding transaction: %1.
2014=No disposition has been made for this payment transaction.
2015=Payment transaction failed.
2016=Error finding Merchant: %1.
2017=Transaction (%1/%2) is in invalid State %3; expected is %4 or %5.
2018=MicroDebits for transaction (%1/%2) are not sequential.
2019=MicroDebit for transaction (%1/%2) does not exist.
2020=Business type of transaction (%1/%2) is %3. Amount cannot be modified.
2021=The amount %1 is invalid for the transaction (%2/%3). The minimum transaction amount is %4.
2022=Transaction (%1/%2) has no cards assigned.
```

```
2023=Business type of transaction (%1/%2) is %3; expected: %4. Please contact your webshop.
2024=Debit cannot be performed; business type of transaction (%1/%2) is %3.
2025=Transaction amount is empty. Please contact your webshop.
2026=Transaction amount %1 is not numeric. Please contact your webshop.
2027=Transaction amount %1 is invalid. Please contact your webshop.
2028=Business type %1 is invalid for transaction.
2029=An error has occurred with this transaction – the amount must be greater than zero.
2039=Invalid restriction parameters.
2044=customerdetailsrequested {0} is invalid (must be 0 or 1).
2623=Shop ID is greater than 60 characters.
2624=Shop Label is greater than 60 characters.


# payment messages - success messages: 2601 - 2900
2601=Payment completed successfully.
2602=Command completed successfully. No transactions found.

# master reference - error message: 3001 - 3600
3001=Merchant %1 is not active. Please contact your webshop.
3002=Currency %1 is not valid for merchant %2. Please contact your webshop.
3003=Merchant %1 does not exist. Please contact your webshop.
3006=Card Type %1 is not accepted by the merchant.
3007=Merchant %1 exceeded time window to debit the transaction.
3012=Merchant %1 exceeded time window for Micropayment.
3013=Merchant %1 already exists.
3014=Reporting Criterion %1 for Merchant %2 doesn't exist.
3015=Reporting Criterion %1 for Merchant %2 is in state %3; expected %4 or %5.

# feature messages: 3901 - 4000
3901=Feature with primary key (%1 %2 %3) cannot be found.
3902=The user %1 is not allowed to access the feature %2.

# merchant API technical messages - error messages: 4001 - 4600
4001=SSL error.
4002=Invalid function request.
4003= above maximum disposition amount (1,000.00 EUR or equivalent in different transaction
        currency)
4004=Invalid proxy request.
4005=Connection error.
4006=Unexpected response from server.
4007=Undefined error - this should not happen.
4008=Error reported from backend.
4010=Error opening configuration file.
4011=Configuration file is not a regular readable file.
4012=Incorrect syntax in configuration file.
4013=Incorrect value in configuration file.
4014=Error HTTP response from API proxy: %1.

# technical error messages: 5001-5500
5001=General technical error.
5002=MAC check.

# SOPG specific error messages: 10000-20001
10003= HTTPS request error
10004= General technical error
10005= General technical error
10006= PIN validation failed
10007= Unexpected error
10008= Authentication failed
10010= cancelPayment too late
```

10011= Insufficient Balance
10012= Zero Balance
10013= Card not active
10014= Method not allowed for SOPG User
10015= Currency not valid for SOPG User

9. Appendix B: Payment notification supported country codes

As an additional information parameter, the country code (country a paysafecard is sold in) is part of the standard payment notification API request.

The parameter 'cardTypeId' within the payment notification provides a combination of default ISO country code and cardtype ID.

Exception:

Some cards are not assigned to a specific country. Therefore, no country code will be provided.

basic API request	mtid=<Mtid> &eventType=<eventType> &serialNumbers=<serialNr1>;<currency1>;<amount1>;<cardTyp1>; <serialNr2>;<currency2>;<amount2>;<cardType2>;
sample API request	mtid=123456 &eventType=ASSIGN_CARDS &serialNumbers=0000000001200000;EUR;50.00;**XX**00004;0000000001200001;EUR;50.00;**XX**00004

10. Appendix C: In App Payment

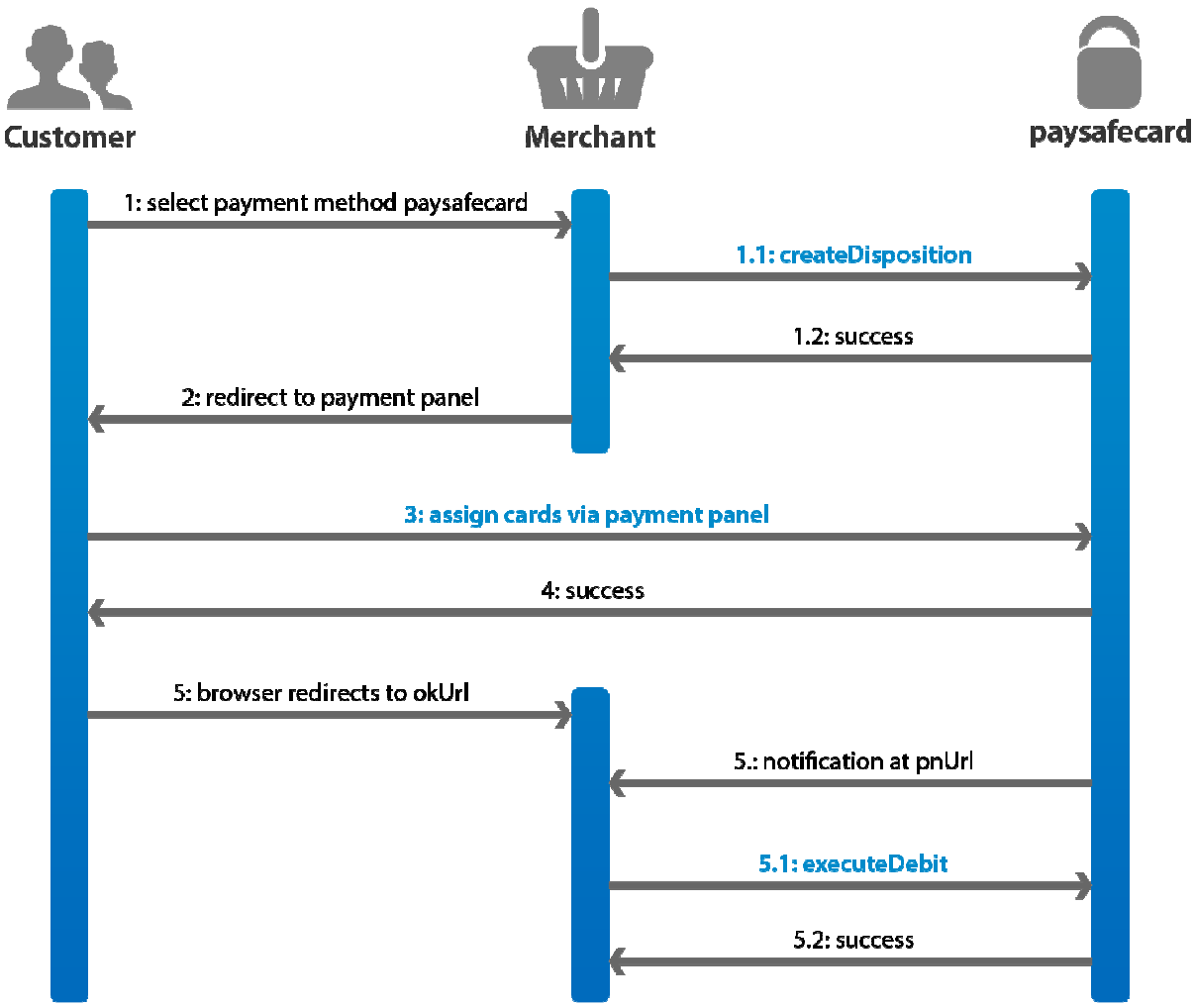
This chapter describes how to integrate a paysafecard payment in an Android app by using our sample codes from the download area.

10.1 Payment Panel Details for in App

To keep a clean and seamless payment flow experience to the customer, we strongly recommend embedding the payment panel into the mobile app.

NOTE: Please always close the paysafecard payment upon receipt of payment notification. The reason is, if customer redirection is executed outside the mobile app, be aware that redirection back to the mobile app is dependant on the customer.

10.2 Technical Overview



11. In App Payment example integration

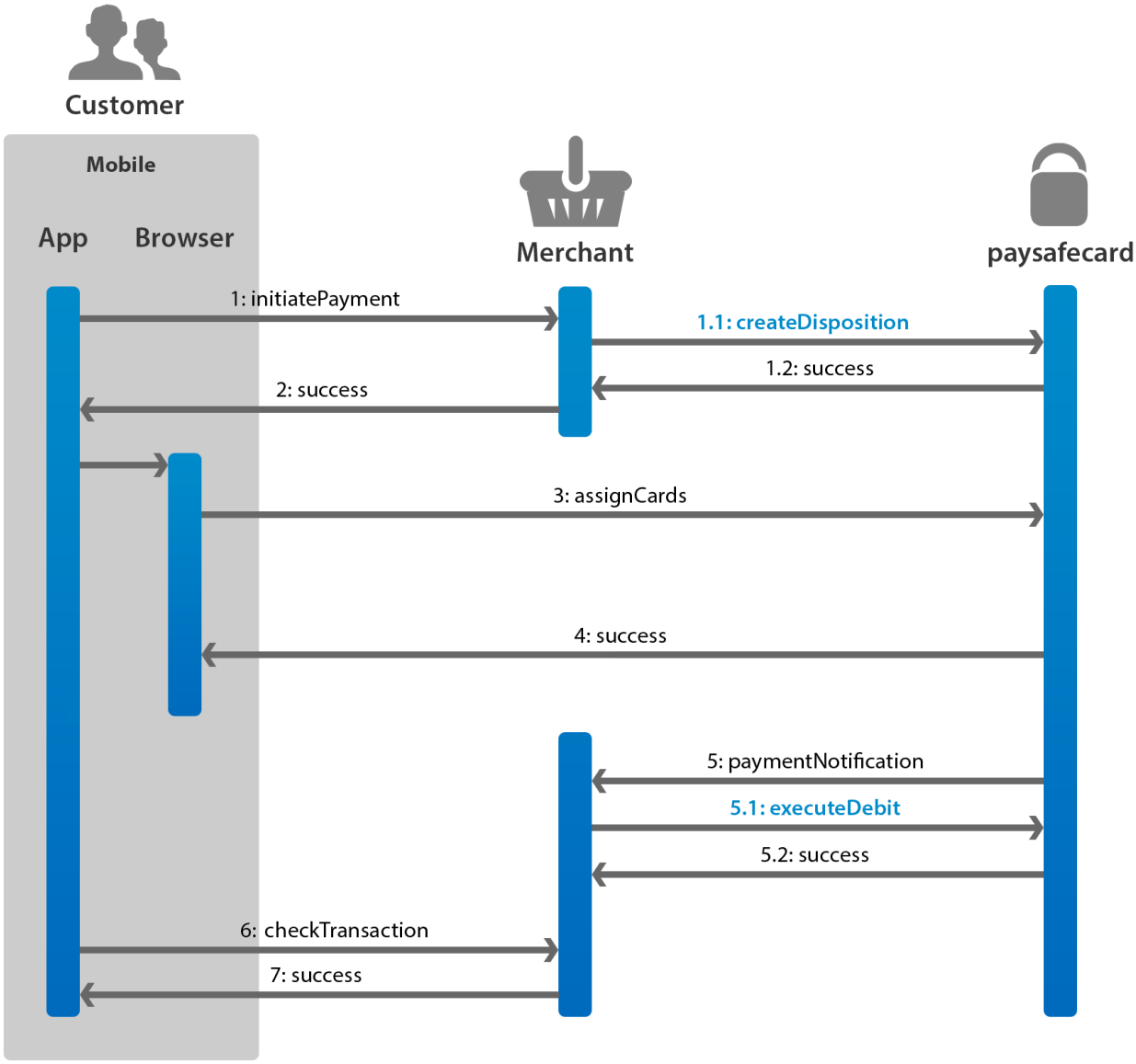
This chapter describes sample integration by using JSON requests.

11.1 Supported OS

The following versions of Android are tested and supported:

- Android 2.1-update1
- Android 2.2
- Android 2.3.1
- Android 2.3.3
- Android 3.0
- Android 3.1
- Android 3.2
- Android 4.0

11.2 Technical Overview for example integration



JSON:initiatePayment

- The business partnerapp initiates the payment at the business partner's back-end server.

JSON:checkTransaction

- The Mobile app asks the business partner back-end server for the status of succeeded payment disposition.
- The Mobile application must be informed if the payment notification was received.
- **NOTE:** The payment notification will only be sent in case of a successful assignment.

11.3 Implementation Example

These examples are based on ANDROID OS. Please check the code of the demo Android application and the webshop server for a full understanding. In the following steps we will describe the most interesting code parts in reference implementation.

11.3.1 Initiate payment communication between the business partner app and the business partner back-end server

To enable payment for paysafecard, the following steps need to be achieved in the Android application:

1. The user initiates payment in an Android application (e.g., the user wants to play a certain game).
2. The business partner application initates the payment (creates a disposition in the paysafecard system) over the back-end server.
3. After a successful disposition creation, paysafecard payment panel should be opened.
4. The user enters the paysfecard PIN and assigns the card to the disposition. After a successful assignment, the application is responsible for the communication to the business partner's back-end server, which requests the 'executeDebit' to withdraw money from the customer's paysafecard in the last step.

11.3.1.1 Code examples:

After starting **PaySafeCardActivityWithBrowser** (see in code), the **handleIntent** method is called.

```
public void handleIntent() {
        final Intent intent = getIntent();

        if (PaySafeCard.ACTION_PAY.equals(intent.getAction())) {
                handlePayAction(intent);
        } else if (Intent.ACTION_VIEW.equals(intent.getAction())) {
                handleBrowserCallbackAction(intent);
        }
}
```

This method handles intents with different actions **(ACTION_PAY vs. ACTION_VIEW),** and, according to this action, the next steps in the application are performed. On the first activity start, the parent activity sends **ACTION_PAY** action which inititates the following request:

Request:

```
private void handlePayAction(final Intent intent) {
        payment = (Payment) intent.getParcelableExtra(PaySafeCard.EXTRA_PAYMENT);
        pscBean = PscBean.from(payment);
 initPaymentTask = new InitPaymentAsyncTask(ConnectorFactory.instance().get());
        initPaymentTask.bindContext(this);
        initPaymentTask.registerListener(new InitPaymentResultListener());
        initPaymentTask.execute(pscBean);
 }
```

The **HandlePayAction** method creates a new disposition through the business partner back-endserver. This action occurs in a separate AsyncTask.

The **InitPaymentResultListener** method returns on a successful creation or in the case of an error.

In case everything went well, the **onResult** callback method within **InitPaymentResultListener** is called:

```
public void onResult(final String result) {
 final Intent browserIntent = PaySafeCardActivityWithBrowser.createBrowserIntent(Uri.parse(result));
 isBrowserStarted = true;
 startActivity(browserIntent);
}
```

This method starts and opens a new browser window with paysafecard mobile payment panel.

For a better understanding, check the full code of the following classes: **InitPaymentResultListener, InitPaymentAsyncTask** and **InitPaymentResultListener.**

11.3.1.2 Request examples

For communication between an Android application and SOPG WSDL-based web-service, we implemented simple REST-based, back-end server. So the demo app communicates only directly with a REST-based back-end server.

Communication is made over lightweight JSON objects.
NOTE: this is only a sample implementation.

The following request/response examples are sent from/to the business partner back-end server reference implementation:

Request:

```
Http-Method: POST
Content-Type: application/x-www-form-urlencoded
Headers: {connection=[Keep-Alive], Content-Length=[75], content-type=[application/x-www-form-urlencoded], host=[my.merchantserver.biz:8080], user-agent=[Android - PSC (v0.1 Prototype)]}
Payload: transactionID=f24f06f5-0c4a-497e-aa66-90c3b34a07bf &amount=1.00&currency=EUR
```

This request contains a payload sent to the business partner back-end server that contains information about the unique transaction id (MTID), amount and currency for a 'createDisposition' request.

Response:

```
Response-Code: 200
Content-Type: application/json
Headers: {Date=[Wed, 27 Jul 2011 16:44:23 GMT]}
Payload:
{"initiatePaymentResponse":{"paymentPanelURL":"https%3A%2F%2Fcustomer.test.at.paysafecard.com%2Fpsccustomer%2FGetCustomerPanelServlet%3Fmid%3D1000003265%26mtid%3D f24f06f5-0c4a-497e-aa66-90c3b34a07bf %26amount%3D1%2C00%26currency%3DEUR"}}
```

The back-end server returns a small JSON object with an encoded payment panel URI. In the next step, this URI is opened in an external browser window.

Please check our reference implementation for more details on how to implement the back-end server.

11.3.2 Communication between mobile app and closing of mobile client browser

This example describes the handling of **'okURL'** and **'nokURL'** on the mobile device. **'okURL'** and **'nokURL'** are specified in the back-end server and an ANDROID app. The back-end server provides **'okURL'** and **'nokURL'** to the application. After the successful assignment of the paysafecard PIN), the payment application (from the user's point of view on the web-browser) sends **Intent** with the specified 'okURL' and 'nokURL'.

NOTE: it is important to react to the created event. To achieve this, we have specified the following configuration for **PaySafeCardActivityWithBrowser** in **AndroidManifest.xml:**

```
<activity android:name=".payment.PaySafeCardActivityWithBrowser"
            android:theme="@android:style/Theme.Translucent.NoTitleBar"
            android:screenOrientation="portrait">
 <intent-filter>
 <data android:scheme="paysafecard" />
 <action android:name="android.intent.action.VIEW" />
 <category android:name="android.intent.category.DEFAULT" />
 <category android:name="android.intent.category.BROWSABLE" />
 </intent-filter>
</activity>
```

PaySafeCardActivityWithBrowser activity is started at any time when **'okURL'** and **'nokURL'** redirection with the scheme **'paysafecard'** is called in an ANDROID environment.

In this case, **handleBrowserCallbackAction** is called:

```
private void handleBrowserCallbackAction(final Intent intent) {
            String uri = intent.getDataString();
            if (uri.contains(PAYMENT_OK)) {
             finishActivityWithSuccess();
            } else {
             finishActivityWithFail();
            }
}
```

Using this method enables the check if 'createDisposition' and the card assignment are successfully performed.

It is necessary to check if the back-end server responds with 'okURL' or 'nokURL'.

In case of success, payment can be fulfilled and the amount can be debited from the user's paysafecard:

```
private void finishActivityWithSuccess() {
 fulfillPaymentTask = new FulfillPaymentTask(ConnectorFactory.instance().get());
 fulfillPaymentTask.bindContext(PaySafeCardActivityWithWebView.this);
 fulfillPaymentTask.registerListener(new FulfillPaymentResultListener());
 fulfillPaymentTask.execute(pscBean);
}
```

This can also be done in AsyncTask to prevent the locking of the GUI main thread. This task could also respond on some callback method to provide the information if the **'executeDebit'** action was successfully executed. After that, **PaySafeCardActivityWithBrowser** activity should display some information to the end-user.