

Merchant API

Preliminary Version

Table of contents

1. Introduction	4
1.1. Purpose	4
1.2. Scope.....	4
2. Payment overview	4
2.1. Life cycle of a normal payment transaction	4
2.2. How the merchant-API works	8
3. Payment functional specification	10
3.1. Payment API functions -overview.....	10
3.2. Payment API functions –details.....	10
3.2.1. Create Disposition	10
3.2.2. Get Disposition State	12
3.2.3. GetSerialNumbers	12
3.2.4. Modify Disposition Value	13
3.2.5. Execute Debit	14
4. Merchant API functional specification	15
4.1. Description of input parameters.....	15
4.1.1. Encoding of (N)OK-URLs:	16
4.1.2. Usage of locale and language parameter	16
4.2. Description of results:	18
4.3. Cleanup / Recovery (only on productive systems).....	18
4.4. Java API	18
4.5. PHP libcurl API	19
4.6. .NET API.....	19
4.7. API Proxy	19
4.8. ‘Native’ API by HTTPS-URLs	19
5. Test-Scenario	22
5.1. Payment Test-Scenario.....	22
5.1.1. Create Disposition	23
5.1.2. Get Disposition Status	24
5.1.3. Assign Cards to Disposition	25
5.1.4. Get Serial Numbers	26
5.1.5. A) Execute Debit	27
5.1.6. B) Modify Disposition for Payment Transaction.....	28
6. Initialize Merchant Test Data	29
7. Cash-Ticket transaction XML format	30

7.1.	Overview of schema.....	30
7.2.	Test Scenario with XML output	32
7.2.1.	Create disposition	32
7.2.2.	Get Disposition State	34
7.2.3.	Modify Disposition Value	35
7.2.4.	Get Serial Numbers	36
7.2.5.	Execute Debit	37
7.3.	Requesting new schemas.....	38

All API clients and documentation are downloadable at

<http://www.paysafecard.com/uk/business/info-for-webshops/diy-installation/step-2-download/>

Contact:

For technical questions about the implementation please contact

techsupport@paysafecard.com

1. Introduction

This document provides an overview of the Merchant API functions. It describes the whole payment scenario to give an overview to the Cash-Ticket-merchants, starting at the creation of a payment transaction by a merchant to executing the payment debits.

1.1. Purpose

This document provides an overview of the Merchant API functional specifications.

1.2. Scope

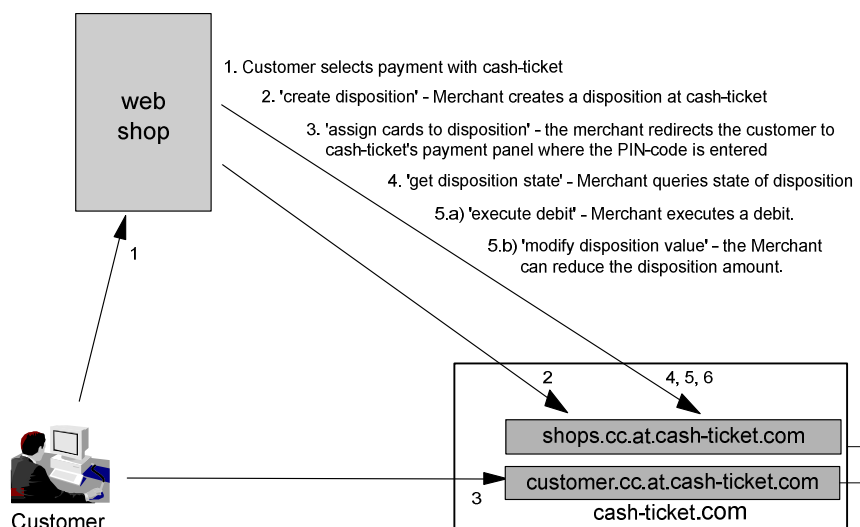
This document applies to all Merchant APIs regardless of the programming language. For each particular Merchant API a language specific document which follows the functional specification must exist.

2. Payment overview

In each payment there are three parties involved: a customer, a web-shop (whom we refer to as 'merchant') and the Cash-Ticket company.

Payments take place in 'payment transactions' or 'dispositions', which are uniquely identified by a 'merchant transaction ID' and hold a value called 'amount' that is typically the amount of money for which a customer buys something

2.1. Life cycle of a normal payment transaction



1. When a customer indicates that they want to buy, this is typically done by activating a button in the merchant's web shop.
2. The merchant now initiates the payment process by sending a 'create disposition' request to Cash-Ticket that creates a disposition at the server. It is the responsibility of the merchant to provide the unique merchant transaction ID and remember it for future reference. Now that Cash-Ticket knows about the payment, the merchant has to forward the customer to the Cash-Ticket payment site (this is typically done automatically by redirecting the customer's web browser to this site). Along with the forwarding HTTPS request the merchant sends the merchant transaction ID and some other parameters so that Cash-Ticket can associate the incoming customer with the previously created disposition.
3. The customer is now presented with a page where they can enter the PIN (a secret number of a Cash-Ticket) and optionally a password, in case the customer has set a password at the Cash-Ticket Website. Upon completion Cash-Ticket sends the customer back to the merchant's site (the exact URL ("OK-URL") is specified by the merchant during the creation of the payment transaction). The amount shall not be transferred in the OK-URL.
4. The merchant checks the correct payment by the request "get disposition state". After the successful check, the merchant typically thanks the customer for the payment or invites them to shop further. At this stage Cash-Ticket reserves the amount for the merchant, but the amount is not yet captured.
5. The merchant can prepare shipment of the goods. Immediately before shipment the merchant confirms that the payment transaction is completed by issuing an 'execute debit' request to Cash-Ticket. Only after the successful 'execute debit' request Cash-Ticket transfers the money to the merchant. In case the merchant does not call the 'execute' debit request within a certain time period (disposition time¹, which has to be set by Cash-Ticket) the reserved amount will be credited back to the card and can be reused by the customer. The merchant has lost the amount and will never be able to capture it. Unless otherwise noted Cash-Ticket automatically sets the

¹ If not specified on the contract, the disposition time will be set to 1 hour. If needed, it can be changed by Cash-Ticket any time.

disposition time to 1 hour. Usually the merchant will send the 'execute debit' request with a 'close flag' set to indicate that no more activity will take place for this disposition. Sometimes the merchant may discover that he cannot deliver the goods, for example when the desired items are no longer in stock. In this case the merchant can send a 'modify disposition' request to Cash-Ticket to reduce the amount of the payment transaction or even set it to zero. The amount, which was up to this time reserved for the transaction, is freed on the customer's card and they can use it for other payments. Another possible scenario is that part of the order can be shipped immediately, while the rest, which for example has to be fetched from a central stock, will take longer. In this case the merchant can decide to ship in two parts and send an 'execute debit' call for only a part of the amount, where the 'close flag' is not set so that Cash-Ticket knows that further debits will take place for the transaction.

The following sequence diagram below shows the sequence followed for payments.

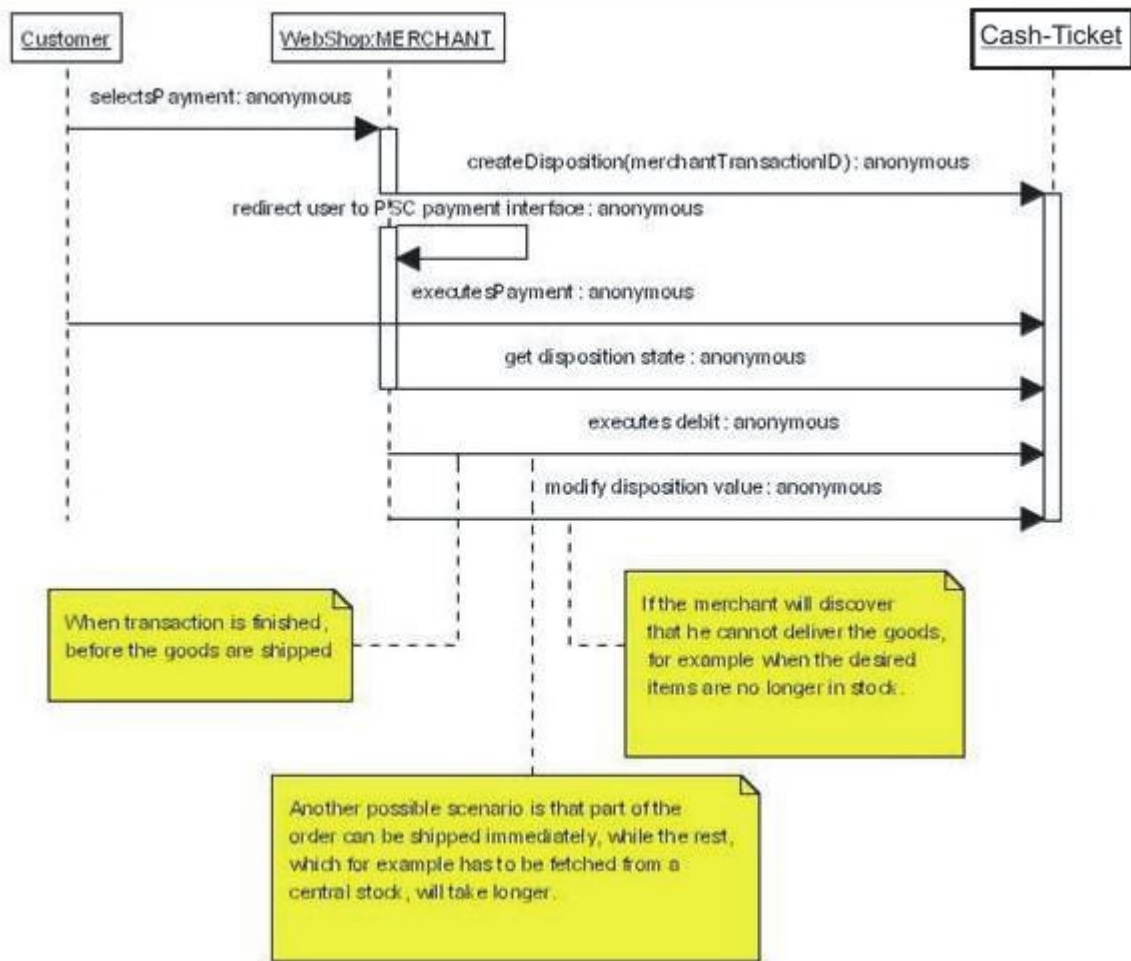


Figure 1 – Normal payment procedure – sequence diagram

2.2. How the merchant-API works

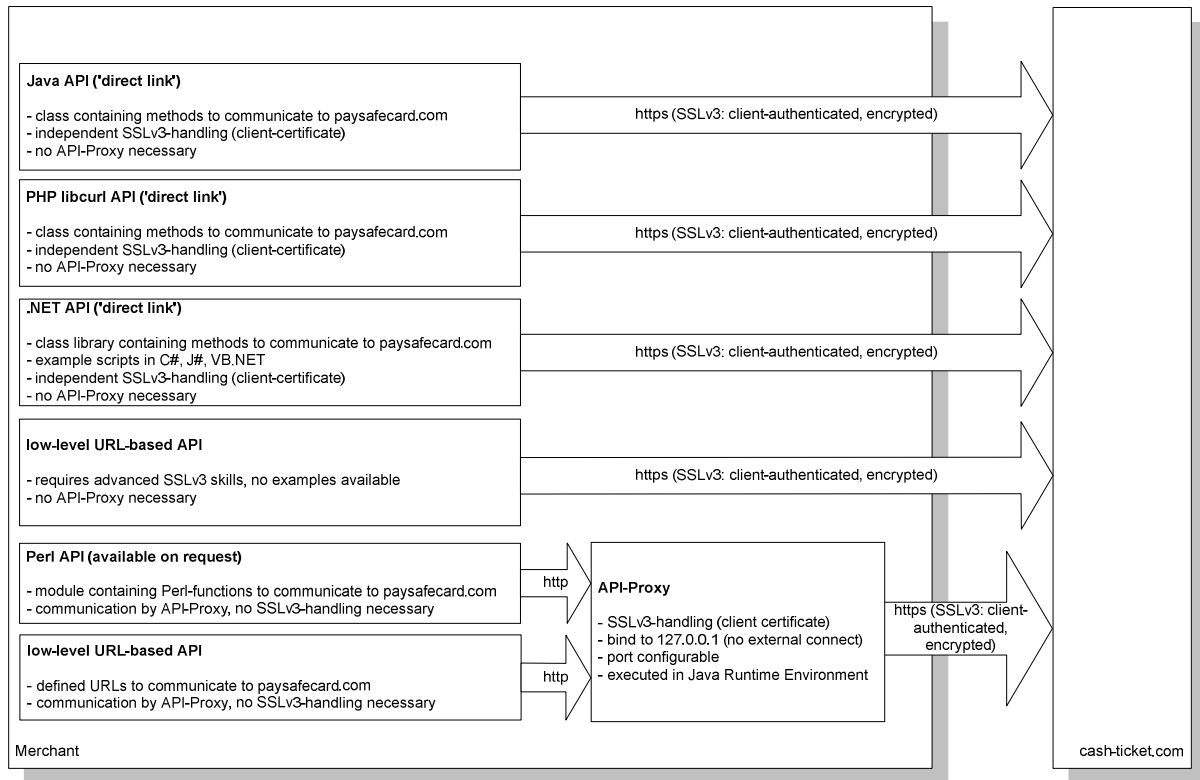


Figure 2 – different Merchant APIs

The Cash-Ticket server is a web server which reacts to certain HTTP requests.

Since confidential data are transferred between merchants or customers and Cash-Ticket, all connections are based on SSL, Netscape's Secure Socket Layer protocol. Most modern browsers support this protocol.

SSL not only encodes the communication in a secure manner but also allows to authenticate the parties involved with the use of certificates.

The communication between customer and Cash-Ticket is done using SSLv2 protocol, in which the server presents its certificate to the customer's machine to assure its identity but does not require the customer's machine to identify itself (the customer remains anonymous).

Communication between the merchant and Cash-Ticket is done using the SSLv3 protocol which also requires the merchant's machine to present a certificate so that Cash-Ticket can assure the identity of the merchant.

All functions of the merchant-API result in HTTP requests to the Cash-Ticket server via this SSLv3 connection. The merchant API has the purpose to encapsulate this

functionality and make it easily integrable into the merchant's application. It is available in the programming languages Java, PHP and .NET and contains sample code. Additionally, we provide an API Proxy if direct SSL connections are not available. The API-Proxy will run on every operating system for which a Java Virtual Machine exists.

During integration and test of the merchant application the merchant can use the 'Cash-Ticket merchant test system' (MTS).

3. Payment functional specification

3.1. Payment API functions -overview

The following functions are available for a merchant:

API function	Input parameters
create disposition	mid, mtid, amount, currency, businesstype, reportingcriteria, okurl, nokurl, config, outputFormat
get disposition state	mid, mtid, config, outputFormat
get serial numbers	mid, mtid, config, outputFormat
execute debit	mid, mtid, amount, currency, close, config, outputFormat
modify disposition value	mid, mtid, amount, currency, config, outputFormat

Table 1 – Payment functions

The URLs for the payment functions must be set in proxy.properties (if a proxy is used), or in merchant_direct.properties (if the merchant is not using the proxy).

A call to the Merchant API with wrong parameters passed MUST return an error code as specified in the Appendix of this document. More information on these functions and their parameters can be found below, details on the return codes can be found in chapters 5 and 7.

3.2. Payment API functions –details

3.2.1. Create Disposition

The merchant initiates the payment process by sending a 'create disposition' request to Cash-Ticket that creates a disposition on the server. The following data must be passed with the request:

- mid: the 'merchantID', is the unique ID of the merchant provided by Cash-Ticket (10 digits)
- mtid: the 'merchantTransactionID', has to be generated by the merchant and has to be unique for each merchant (typically it includes a customer ID and the timestamp e.g. user321-2006082413246789)

- amount: the value of the goods ordered by the customer; max. 11 digits before the decimal point, exactly two digits after the decimal point
- currency is the merchant currency (3 characters ISO-Code)
- businesstype: optional parameter; can be 'T' for TANGIBLE, 'I' for INTANGIBLE or left empty, which evaluates to OLDSTYLE. Each business type corresponds to a feature for which the merchant has to be enabled by Cash-Ticket. That means that the feature must be enabled for the merchant so that they can create a disposition with business type INTANGIBLE. If the parameter is empty, the business type is automatically converted to OLDSTYLE.
- 'reportingcriteria' is an optional attribute for merchant reporting purpose. The possible values for the merchant have to be created by Cash-Ticket
- 'OKURL' this is the URL to which the customers are forwarded by Cash-Ticket after they successfully assigned their cards and completed the payment. The merchant may include some information in the URL (e.g. the merchantTransactionID) to get back the context of the payment transaction, e.g. to show a confirmation panel to the customer.
- 'NOKURL' this is the URL to which customers are forwarded by Cash-Ticket when they hit "cancel" on the payment panel.

After successful 'create disposition' the merchant has to redirect the customer to the Cash-Ticket payment site. This URL has to include 'merchantID', 'merchantTransactionID', 'amount' and 'currency'.

- 'outputFormat' is an optional attribute. It specifies the type of the response message (plain text or xml). If this attribute has the value 'xml_v1' then the response message is in xml format as described in chapter 7. If it is missing or has any other value, the response message is in plain text format.

3.2.2. Get Disposition State

Dispositions also have a 'state' which changes during the life cycle. The possible states are listed in the table below:

State	Perform Debit	Description
C (created)	No	The disposition has been created successfully, but the customer did not yet assign cards
D (disposed)	Yes	Customer already assigned cards and completed payment on his side, the transaction still has to be debited by the merchant
X (expired)	No	The disposition is either totally consumed (no open amount) or cancelled by the customer or expired by Cash-Ticket (see Cleanup)

Table 2 – Disposition State returned by 'GetDispositionState'

With the 'get disposition state' request the merchant can query the Cash-Ticket application regarding the current state of a transaction. The function 'get disposition state' returns the amount of the disposition and can also be used to check how big the amount is that is still open for a certain payment transaction.

The mid ('merchantID') and the mtid ('merchantTransactionID') have to be passed as input parameters.

3.2.3. GetSerialNumbers

The 'GetSerialNumbers' function is an extension of 'GetDispositionState'. The return codes are separated by carriage returns/line feeds containing result code, error code, disposition-value (current), currency, state and a list of serial numbers and their associated disposition-values. This list is separated by semicolons. Example (successful call):

```
0
0

8.00
EUR
S
0000000001200000;7.50;0000000001200001;0.50
```

Unlike the 'function GetDispositionState' which maps the disposition states, the function 'GetSerialNumbers' returns the raw states. These are:

One letter code	Meaning	Description
R	Created	The disposition has successfully been created. If nothing happens within 30 minutes the disposition will be transferred to state "X" by the cleanup job
S	Disposed	Cards have successfully been assigned to the disposition, the merchant can debit, no debits have taken place so far
E	Debited	Partially Debited, the disposition is still open, further debits are possible
O	Consumed	Final debit with closeflag=1 has been called. No further debits are possible
L	Cancelled	The disposition has actively been cancelled by the customer
I	Invalid	This disposition state is currently not in use
X	Expired	The time window for this disposition has ended (either before cards were assigned or before debit calls were made

Table 3 – Disposition State returned by GetSerialNumbers

3.2.4. Modify Disposition Value

The merchant can reduce the originally disposed amount with the 'modify disposition value' request, e.g. if he cannot ship the goods, he can partially cancel part of the order. The disposition has to be in state disposed or (partially) debited. The input parameters are 'mid', 'mtid', the 'amount' to which the merchant wants to reduce the disposition's value (the difference will be made available on the card(s) used for this reservation) and merchant 'currency'.

3.2.5. Execute Debit

The 'execute debit' function performs the process of withdrawing money from the customer's Cash-Ticket. For each disposition <n> debits can be executed. Each debit reduces the open disposition amount. It will differ from merchant to merchant whether the whole disposition amount will be withdrawn by a single debit or several debits will be executed.

The following data must be passed:

- mid: the 'merchantID'
- mtid: the 'merchantTransactionID': unique identifier for each disposition
- 'amount': the debit amount; max. 11 digits before the decimal point, exactly two digits after the decimal point
- 'currency': the merchant currency
- 'closeFlag': indicates if further debits will be executed or not. If the close flag is '1' the disposition will be set to totally consumed and no further debits are possible.

4. Merchant API functional specification

4.1. Description of input parameters

Parameter Name	Parameter Description
mid	merchant ID, uniquely identifies merchant, length = 10
mtid	merchant transaction ID, uniquely generated by merchant, max. length = 60 characters
amount	amount of payment transaction, fixed point decimal with max. 11 digits before comma and exactly 2 digits after comma (e.g. '3200.00'); use a point as decimal separator
currency	ISO code for currency, length 3 - 5 character (e.g. 'EUR')
businesstype	1 letter, possible values: 'O' (Oldstyle), 'T' (Tangible), 'I' (Intangible)
reportingcriteria	Optional parameter, length = 8, specifying the merchant reporting criteria, must be enabled by Cash-Ticket
okurl	OK-URL to which the customer is forwarded by Cash-Ticket after successful payment. OK-URL has to be passed URL encoded.
nokurl	NOT-OK-URL to which customers are forwarded by Cash-Ticket when they hit "cancel" on the payment panel. NOT-OK-URL has to be passed URL encoded
close	close transaction flag, possible values '0' (don't close transaction) or '1' (close transaction, this is the last debit)
config	local path, defines the merchant configuration-file containing URLs, SSLv3-Certificate etc...
locale	The locale of the payment page enabling country-specific interfaces ('en_uk', 'de_de' and many more) – see chapter 4.1.2 for details about supported locale parameters and application behaviour when locale and language parameters are used
language	Supported for backward compatibility only - language of the payment page ('en', 'de') - see chapter 4.1.2 for details about supported locale parameters and application behaviour when locale and language parameters are used
outputFormat	Optional parameter, specifies the format of the output (plain text or xml), see chapter 7 for details

Table 4 – Parameters used by Merchant API functions

All parameters will be passed as 7 Bit-ASCII Strings

4.1.1. Encoding of (N)OK-URLs:

URLs have to be passed URL-encoded. The encoding is done in two steps:

- the parameter of the URL have to be converted to standard HTML code, e.g. the character (ä Ä ö Ö ü Ü ß) are replaced by (ä Ä ö Ö ü Ü ß)
- all non-alphanumeric characters in the parameter of the (N)OK-URL have to be replaced by '%HH' (HH is the hex code of the character).

EXAMPLE:

"Danke für die Bezahlung" looks like "Danke für die Bezahlung" in standard HTML and finally has to be passed on as "Danke%20f%26uuml%3br%20die%20Bezahlung".

4.1.2. Usage of locale and language parameter

Basically, the locale parameter uses a combination of language and country code and will replace the language parameter in the long run, because it enables different payment interfaces depending on language AND country.

Currently, the following locales are supported:

de_de, de_at, en_uk, en_us

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. We would therefore suggest to use only the locale parameter.

Supported language parameters:

de and en

If more than one country exists for a language, a default country is defined:

en → en_uk

de → de_de

The following rules apply for the usage of one or both parameters:

- If a **valid** locale is sent, it overrides any language parameter
- If no language **AND** no locale is sent, the default is "de_de"
- A **completely invalid** locale like "xy_xy" defaults to "en_uk"
- In case of a **partly invalid** (or not yet supported) locale like "en_xy", the language part will still be used and the result page will be "en_uk"

Examples:

locale	language	payment page displayed in
de_de	German	de_de (german Germany)
de_at	German	de_at (german Austria)
en_uk	English	en_uk (british english)
en_us	English	en_us (american english)

4.2. Description of results:

Result Name	Result Description
resultcode	0 : successful, 1 : logical problem, 2 : technical problem
errorcode	Contains an error number if resultcode is not equal to 0.
errormessage	Error message plain text
amount	Amount, only when calling 'get disposition state' or 'get serial number'
currency	Currency, only when calling 'get disposition state' or 'get serial number'
state	Disposition state, possible values 'C', 'D' or 'X' with 'get_disposition_state' or 'R', 'S', 'E', 'O', 'L', 'I' or 'X' with 'get_serial_number'

Table 5 – Results returned by Merchant API functions

4.3. Cleanup / Recovery (only on productive systems)

Once a disposition is in state 'disposed', the merchants have to perform their debits within a certain 'payment timeout window' which varies from merchant to merchant and is configurable by Cash-Ticket. If this time window is exceeded, the disposition will automatically expire and the amount will be freed on the customer's card.

Furthermore, there is another job which sets all dispositions that have been created but not paid for to state "expired".

Please note, that these jobs are only active on the productive servers.

4.4. Java API

To get the demo working, you will need a servlet container for JSPs (e.g. Apache Tomcat). For a detailed description of the Java API classes please refer to the documentation on this API client on our website

4.5. PHP libcurl API

To get the demo working, you will need an Apache web server with Open SSL enabled. For a detailed description of the PHP libcurl API please refer to the documentation on this API client on our website

4.6. .NET API

To get the demo working, you will need Windows 2000 Server, Windows XP with IIS 5 or above or Windows 2003 Server with IIS 6 and .NET (version 1.1 or above) installed. For a detailed description of the .NET API please refer to the documentation on this API client on our website

4.7. API Proxy

To get the API Proxy working, you will need a Java Runtime Environment. For further information please refer to the documentation on the API Proxy on our website.

4.8. 'Native' API by HTTPS-URLs

This is a further approach (not yet mentioned), which should only be used if none of the above possibilities can be applied, because it requires advanced SSLv3-skills. In this case you connect to the Cash-Ticket server directly, without using the API-Proxy, using SSLv3 (client-authenticated) on port 443

Create Disposition:

```
https://<SSLv3 test or online-system>/pscmerchant/CreateDispositionServlet
?mid=MID
&mtid=MTID
&amount=AMOUNT
&currency=CURRENCY
&businessstype=BUSINESSTYPE
&reportingcriteria=REPORTINGCRITERIA
&okurl=OKURL&nokurl=NOKURL
&outputFormat=OUTPUTFORMAT
```

Get Disposition State:

```
https://<SSLv3 test or online-system>/pscmerchant/GetDispositionStateServlet
?mid=MID
&mtid=MTID
&outputFormat=OUTPUTFORMAT
```

Get Serial Numbers:

```
https://<SSLv3 test or online-system>/pscmerchant/GetSerialNumbersServlet
?mid=MID
&mtid=MTID
&outputFormat=OUTPUTFORMAT
```

Execute Debit:

```
https://<SSLv3 test or online-system>/pscmerchant/DebitServlet
?mid=MID
&mtid=MTID
&amount=AMOUNT
&currency=CURRENCY
&close=CLOSE
&outputFormat=OUTPUTFORMAT
```

Modify Disposition Value:

```
https://<SSLv3 test or online-system>/pscmerchant/ModifyDispositionServlet
?mid=MID
&mtid=MTID
&amount=AMOUNT
&currency=CURRENCY
&outputFormat=OUTPUTFORMAT
```

Initialize Merchant Test Data (only on the Merchant Test System)

```
http:// <SSLv3 test or online-system>/pscmerchant/InitializeMerchantTestDataServlet
?mid=mid
&config=CONFIG
&outputFormat=OUTPUTFORMAT
```

The connection to the API-Proxy uses http, the API-Proxy uses https to connect to the Cash-Ticket server. The port number (9335 in the example) is configurable.

All parameters have to be passed URL-encoded. This means that the parameter of the (NOT-) OK-URL have to be encoded twice, e.g. the following ok-URL

```
http://www.shop.com/ok?mtid=MER1&text=Danke%20f%26uuml%3br%20den%20Einkauf
```

has to be sent as:

```
http%3a%2f%2fwww%2eshop%2ecom%2fok%3fmtid%3dMER1%26text%3dDanke%2520f%2526uuml%253br%2520den%2520Einkauf
```

Results

The output result depends on the existence of outputFormat parameter and its value.

If it has the value "xml_v1" then the result is in xml format, which is presented in chapter 7.

If it is missing or has any other value the result will be three, five or six lines text output (plain text - not http), containing resultcode, errorcode, errormessage, amount, currency and state. For the description of these values refer to the 'Overview' section of this chapter.

5. Test-Scenario

The following scenario is a test scenario including example data. In practice it will differ from merchant to merchant whether one or more debits or status-requests will be executed. Do not use the enclosed example data! For testing each merchant you will receive a consistent set of test data.

You will find your specific test-data in the file test_.data_<YOUR mid>.txt in the package you received.

5.1. Payment Test-Scenario

The scenario described in the following chapter contains the following steps:

- Create disposition: The merchant creates a disposition, which means the customer initiated the payment process.
- Get disposition status: The merchant verifies that the reservation has been successfully created and that it has not expired before redirecting to the payment panel.
- Assign card to disposition: One or more cards are assigned to the disposition. This is carried out by the customer. The respective amount is reserved on the card.
- Get Serial Numbers: The merchant checks whether the reservation on the card has been done successfully.
- A) Debit: A debit is performed by the merchant. The debit represents the actual withdrawal of credit from the card.
B) Modify disposition: A merchant can reduce the amount of the disposition, e.g. if shipment is not possible (dispositions can only be reduced, not increased)
- Get Serial Numbers: The merchant verifies the proper status of the disposition after debiting.

5.1.1. Create Disposition

Who:

Merchant

Description:

The merchant creates a disposition by sending the 'create disposition' request passing all necessary parameters.

URL and parameters:

```
https://<SSLv3 test or online-system>/psmerchant/CreateDispositionServlet
?mid=1000000001
&mtid=xyz
&amount=100.00
&currency=EUR
```

Input:

mid:	1000000001
mtid:	xyz
amount:	100.00
currency:	EUR
businessstype:	T
reportingcriteria:	<empty>
okurl:	e.g. http://www.my-shop.com/index.jsp?mtid=xyz
nokurl:	e.g. http://www.my-shop.com/index.jsp?mtid=xyz

Return Values:

resultcode:	0 (function completed successfully)
errorcode:	0 (no error)
errormessage:	NULL

5.1.2. Get Disposition Status

Who:

Merchant

Description:

After having called `create_disposition` the merchant queries the Cash-Ticket server for the status of the payment to verify it has the expected state "C" (CREATED).

URL and parameters:

```
https://<SSLv3 test- or live-system>/pscmerchant/GetDispositionStateServlet  
?mid=1000000001  
&mtid=xyz
```

Input:

mid:	1000000001
mtid:	xyz

Return Values:

resultcode:	0 (function completed successfully)
errorcode:	0 (no error)
errormessage:	NULL
amount	100.00
currency	EUR
state	C ("Created")

5.1.3. Assign Cards to Disposition

Who:

Customer (via Cash-Ticket's payment panel)

Prerequisite:

The command 'create disposition' was successfully executed and returned "0 0". Thus, the customer can be forwarded to the panel for assigning cards to the disposition.

Description:

One or more cards are assigned to the created disposition. This task is executed by the customer.

If customers click cancel without assigning one or more cards to the disposition, they will be forwarded to the NOKURL.

After successful card assignment customers will be shown the second page of the payment panel where they will be forwarded to the OKURL when clicking on the OK-button on that page.

URL and parameters:

```
https://<SSLv3 test- or live-system >/ctcustomer/GetCustomerPanelServlet  
?mid=1000000001  
&mtid=xyz  
&amount=100.00  
&currency=EUR  
&language=en (optional parameter)  
&locale=en_uk (optional parameter)
```

Input:

PIN-Code, e.g.: 4725 4983 6548 7393
Password: <default empty>
Terms of Use: <checkbox, default unchecked>

5.1.4. Get Serial Numbers

Who:

Merchant

Description:

After the customer has assigned cards to the disposition the merchant queries the Cash-Ticket server for the status of the payment to verify it has the expected state "S" (DISPOSED) before calling the debit function.

URL and parameters:

```
https://<SSLv3 test- or live-system>/pscmerchant/GetSerialNumbersServlet  
?mid=1000000001  
&mtid=xyz
```

Input:

mid:	1000000001
mtid:	xyz

Return Values:

resultcode:	0 (function completed successfully)
errorcode:	0 (no error)
errormessage:	NULL
amount	100.00
currency	EUR
state	S (cards assigned, debits allowed)
serial numbers:	0000000001200000;100.00 (semicolon separated list containing serial numbers and amounts reserved per card)

5.1.5. A) Execute Debit

Who:

Merchant

Prerequisite:

The assignment of the cards to disposition was executed successfully.

Description:

The debiting is the process of withdrawing money from the customer's Cash-Ticket. For each disposition <n> debits can be executed and each debit reduces the open disposition amount. The merchant uses the closeFlag to inform Cash-Ticket, whether further debits are to be expected. After the merchant-specific time window expires, each open disposition is closed - no further debits will be accepted, remaining amounts reserved on the customers card will be made available again. In this example the merchant debits EUR 60,- (and the disposition's remaining value yields EUR 40,-)

URL and parameters:

```
https://<SSLv3 test or online-system>/psmerchant/DebitServlet
    ?mid=1000000001
    &mtid=xyz
    &amount=60.00
    &currency=EUR
    &close=1          (or '0', if further debits shall be possible)
```

Input:

mid:	1000000001
mtid:	xyz
amount:	60.00
currency:	EUR
closeFlag:	0

Return Values:

resultcode:	0 (function completed successfully)
errorcode:	0 (no error)
errormessage:	<empty>

5.1.6. B) Modify Disposition for Payment Transaction

Who:

Merchant

Description:

At any time after 'create disposition' the merchant can reduce the originally disposed amount, e.g. if they cannot ship the goods. In this example the open amount is reduced to 30.00 EUR and 10.00 EUR are made available on the customer's Cash-Ticket again.

URL and parameters:

```
https://<SSLv3 test or online-system>/pscmerchant/ModifyDispositionServlet
    ?mid=1000000001
    &mtid=xyz
    &amount=30.00
    &currency=EUR
```

Input:

mid:	1000000001
mtid:	xyz
amount:	30.00
currency:	EUR

Return Values:

resultcode:	0 (function completed successfully)
errorcode:	0 (no error)
errormessage:	<empty>

6. Initialize Merchant Test Data

Please note that this function is only available on the merchant test-system (MTS).

Description:

To simplify testing, the Merchant API also includes an interface to reset the test data. All dispositions and debits of the specific merchant will be undone, passwords for PINs will be reset to their default.

URL and parameters:

*[https://<SSLv3 test system>/pscmerchant/InitializeMerchantTestDataServlet
?mid=1000000001](https://<SSLv3 test system>/pscmerchant/InitializeMerchantTestDataServlet?mid=1000000001)*

Input:

mid: 1000000001

ReturnValues:

resultcode: 0 (function completed successfully)
errorcode: 0 (no error)
errormessage: NULL

7. Cash-Ticket transaction XML format

7.1. Overview of schema

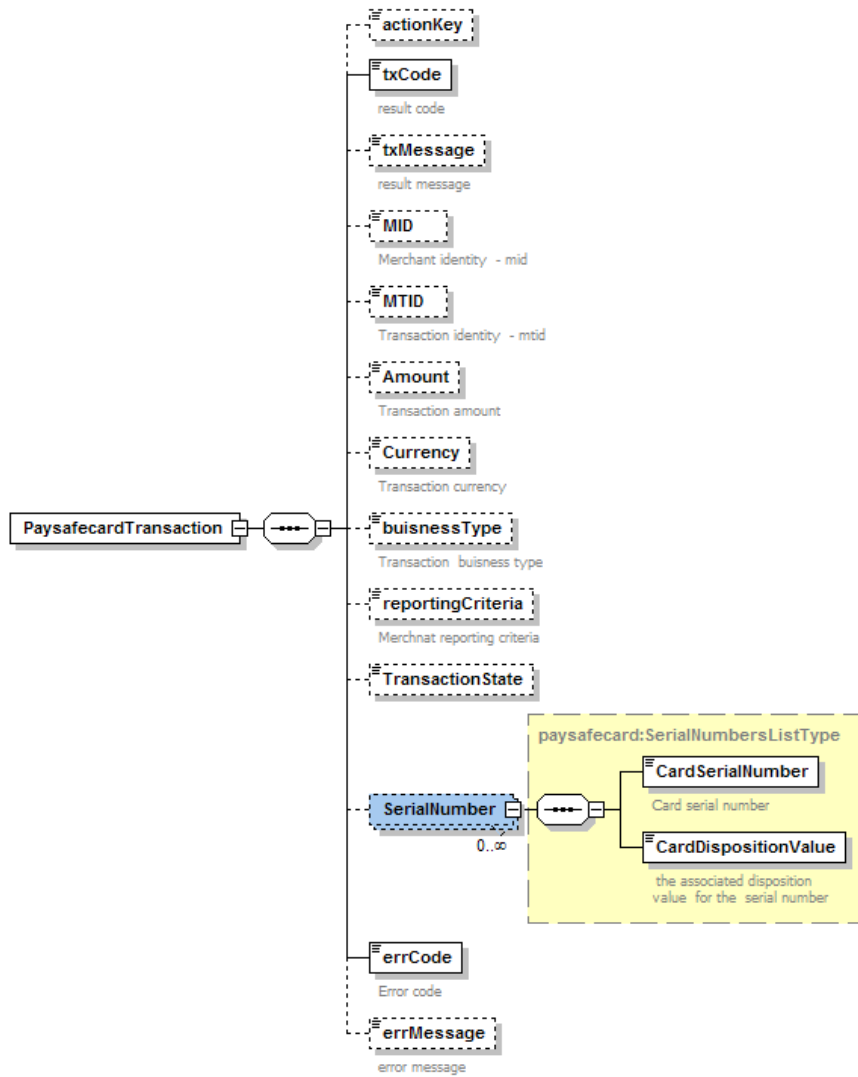


Figure 3 - Schema v1 (xml_v1) overview

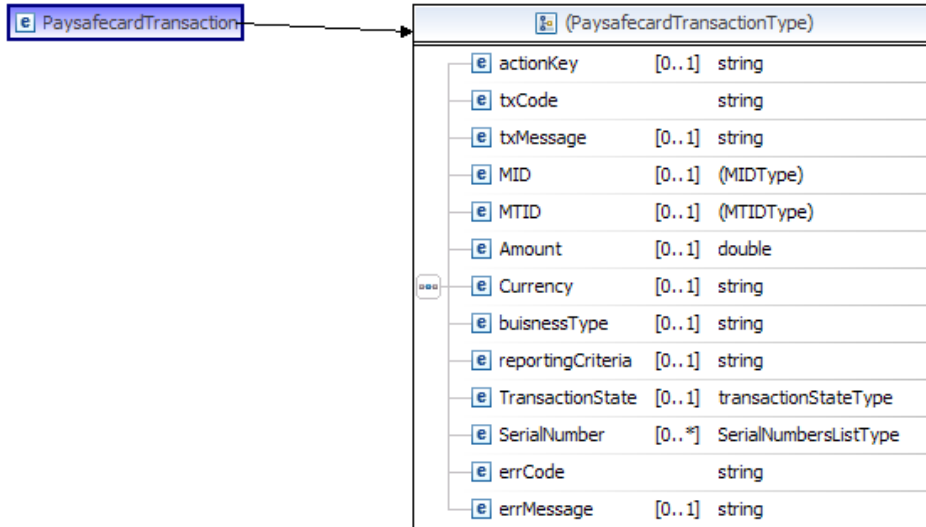


Figure 4 - Schema v1 overview (yet another view)

Please note that the resultCode is called txCode in this XML output schema and that txMessage and errMessage are always equal.

7.2. Test Scenario with XML output

7.2.1. Create disposition

Description:

The merchant initiates the payment process by sending a 'create disposition' request, passing all necessary parameter, to Cash-Ticket that creates a disposition on the server.

URL and parameters:

```
https://<SSLv3 test- or live-system>/pscmerchant/CreateDispositionServlet
    ?mid=1000000001
    &mtid=xyz
    &amount=100.00
    &currency=EUR
    &outputFormat=xml_v1
```

Input:

mid:	1000000001
mtid:	xyz
amount:	100.00
currency:	EUR
businesstype:	T
reportingcriteria:	<empty>
okurl:	e.g. http://www.my-shop.com/index.jsp?mtid=xyz
nokurl:	e.g. http://www.my-shop.com/index.jsp?mtid=xyz

Return Values:**a. Success response**

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<Paysafecard:PaysafecardTransaction xmlns:Cash-Ticket="http://www.Cash-
Ticket.com/MerchantApi" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.paysafecard.com/MerchantApi MerchantApi_v1.xsd">
  <actionKey>com.psc.pay.CreateDispositionAction_1159109437_127.0.0.2</actionKey>
  <txCode>0</txCode>
  <txMessage></txMessage>
  <mid>1000000001</mid>
  <MTID>uniqueTransID</MTID>
  <errCode>0</errCode>
  <errMessage></errMessage>
</Paysafecard:PaysafecardTransaction>
```

b. Sample failure response

NOTE: this response was obtained by trying to recreate a disposition with the same
mtid

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<Paysafecard:PaysafecardTransaction xmlns:Cash-Ticket="http://www.Cash-
Ticket.com/MerchantApi" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.paysafecard.com/MerchantApi MerchantApi_v1.xsd">
  <actionKey>com.psc.pay.CreateDispositionAction_1159109437_127.0.0.2</actionKey>
  <txCode>1</txCode>
  <txMessage>The transaction (1000000001, uniqueTransID) already exists.</txMessage>
  <mid>1000000001</mid>
  <MTID>uniqueTransID</MTID>
  <errCode>2001</errCode>
  <errMessage>The Transaction (1000000001, uniqueTransID) already
exists.</errMessage>
</Paysafecard:PaysafecardTransaction>
```

7.2.2. Get Disposition State

Description:

After having created a disposition the merchant may ask the Cash-Ticket server for the status of the payment to verify it has the expected state "C" (CREATED).

URL and parameters:

```
https://<SSLv3 test or online-system>/pscmerchant/GetDispositionStateServlet  
    ?mid=1000000001  
    &mtid=xyz  
    &outputFormat=xml_v1
```

Input:

mid:	1000000001
mtid:	xyz

Return Values:

Success response:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>  
<Paysafecard:PaysafecardTransaction xmlns:Cash-Ticket="http://www.Cash-  
Ticket.com/MerchantApi" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
xsi:schemaLocation="http://www.paysafecard.com/MerchantApi MerchantApi_v1.xsd">  
    <actionKey>com.psc.pay.GetDispositionStateAction_1159192198218_127.0.0.2  
</actionKey>  
    <txCode>0</txCode>  
    <txMessage></txMessage>  
    <mid>1000000001</mid>  
    <MTID>uniqueTransID</MTID>  
    <Amount>100.00</Amount>  
    <Currency>EUR</Currency>  
    <TransactionState>C</TransactionState>  
    <errCode>0</errCode>  
    <errMessage></errMessage>  
</Paysafecard:PaysafecardTransaction>
```

7.2.3. Modify Disposition Value

Description:

At any time after 'create disposition' the merchant can reduce the originally disposed amount, e.g. if they cannot ship the goods. In our sample the open amount is reduced to 30.00 EUR and 70.00 EUR are made available on the customer's Cash-Ticket again.

URL and parameters:

```
https://<SSLv3 test or online-system>/pscmerchant/ModifyDispositionServlet
    ?mid=1000000001
    &mtid=xyz
    &amount=30.00
    &currency=EUR
    &outputFormat=xml_v1
```

Input:

mid:	1000000001
mtid:	xyz
amount:	30.00
currency:	EUR

Return Values:**Success response**

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<Paysafecard:PaysafecardTransaction
    xmlns:Cash-Ticket="http://www.Cash-
Ticket.com/MerchantApi"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.paysafecard.com/MerchantApiMerchantApi_v1.xsd">
    <actionKey>com.psc.pay.ModifyDispositionAction_1159193017796_127.0.0.2
</actionKey>
    <txCode>0</txCode>
    <txMessage></txMessage>
    <mid>1000000001</mid>
    <MTID>uniqueTransID</MTID>
    <errCode>0</errCode>
    <errMessage></errMessage>
</Paysafecard:PaysafecardTransaction>
```

7.2.4. Get Serial Numbers

Description:

To evaluate a disposition's state on the debit process you must not use `get_disposition_state`, but rather `get_serialnumbers`

URL and parameters:

```
https://<SSLv3 test or online-system>/pscmerchant/GetSerialNumbersServlet?  
mid=1000000001  
&mtid=uniqueTransID  
&outputFormat=xml_v1
```

Return Values:

Success response

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>  
<Paysafecard:PaysafecardTransaction xmlns:Cash-Ticket="http://www.Cash-  
Ticket.com/MerchantApi" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
xsi:schemaLocation="http://www.paysafecard.com/MerchantApi MerchantApi_v1.xsd">  
  <actionKey>com.psc.pay.GetSerialNumbersAction_1159193173875_127.0.0.2  
</actionKey>  
  <txCode>0</txCode>  
  <txMessage></txMessage>  
  <mid>1000000001</mid>  
  <MTID>uniqueTransID</MTID>  
  <Amount>30.00</Amount>  
  <Currency>EUR</Currency>  
  <TransactionState>S</TransactionState>  
  <SerialNumber>  
    <CardSerialNumber>5000000000000751</CardSerialNumber>  
    <CardDispositionValue>1.00</CardDispositionValue>  
  </SerialNumber>  
  <errorCode>0</errorCode>  
  <errorMessage></errorMessage>  
</Paysafecard:PaysafecardTransaction>
```

7.2.5. Execute Debit

URL and parameters:

```
https://<SSLv3 test or online-system>/pscmerchant/DebitServlet?  
    mid=1000000001  
    &mtid=uniqueTransID  
    &amount=30.00  
    &currency=EUR  
    &outputFormat=xml_v1  
    &close=1
```

Return Values:

a. Success response

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>  
<Paysafecard:PaysafecardTransaction xmlns:Cash-Ticket="http://www.Cash-  
Ticket.com/MerchantApi" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
    xsi:schemaLocation="http://www.paysafecard.com/MerchantApiMerchantApi_v1.xsd">  
<actionKey>com.psc.pay.DebitAction_1159193398031_127.0.0.2</actionKey>  
    <txCode>0</txCode>  
    <txMessage></txMessage>  
    <mid>1000000001</mid>  
    <MTID>uniqueTransID</MTID>  
    <errCode>0</errCode>  
    <errMessage></errMessage>  
</Paysafecard:PaysafecardTransaction>
```

b. Sample failure response

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>  
<Paysafecard:PaysafecardTransaction xmlns:Cash-Ticket="http://www.Cash-  
Ticket.com/MerchantApi" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
    xsi:schemaLocation="http://www.paysafecard.com/MerchantApiMerchantApi_v1.xsd">  
<actionKey>com.psc.pay.DebitAction_1159193469031_127.0.0.2</actionKey>  
    <txCode>1</txCode>  
    <txMessage>Transaction (9000000046/uniqueTransID) is in invalid State ERR: 8013, 8011,  
8012</txMessage>  
    <mid>1000000001</mid>  
    <MTID>uniqueTransID</MTID>  
    <errCode>2017</errCode>  
    <errMessage>Transaction (1000000001/uniqueTransID) is in invalid State ERR: 8013,  
8011, 8012</errMessage>  
</Paysafecard:PaysafecardTransaction>
```

7.3. Requesting new schemas

The documentation and examples presented here refer to our standard xml output format (xml_v1). If you would like to learn about the possibility to obtain a customized schema and output, please don't hesitate to contact us.

Appendix A: Error / success messages

general messages - errors: 0001 - 0600

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).

25=Distributor ID is empty.

26=Distributor ID with value %1 is too long (max. %2 characters are allowed).

30=Distributor Name is empty.

31=Distributor Name with value %1 is too long (max. %2 characters are allowed).

35=Logistics Company ID is empty.

36=Logistics Company ID with value %1 is too long (max. %2 characters are allowed).

40=Logistics Company Name is empty.

41=Logistics Company Name with value %1 is too long (max. %2 characters are allowed).

45=Merchant Name is empty.

46=Merchant Name 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='Not-OK' URL is empty.

65='OK' URL is empty.

70=Password is empty.

71=Password too long, max. number of characters 30.

72=Syntax error in password, allowed characters 0-9, a-z, A-Z.

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.

100=Distributor State %1 is invalid.

101=User Type %1 is invalid.

102=Assign action %1 is invalid.

103=Business type %1 is invalid.

104=Feature type %1 is invalid.

105=Import State %1 is invalid.

106=Submitted Import State %1 of field %2 is not equal to the expected Import State %3.

107=Actor Type %1 is invalid.

110=Logistics Company State %1 is invalid.

115=Merchant State %1 is invalid.

116=Submitted Merchant State %1 of field %2 is not equal to the expected Merchant 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).

130=Actor State %1 is invalid.

131=Submitted Actor State %1 of field %2 is not equal to the expected Actor State %3.

135=Actor Name is empty.

136=Actor Name with value %1 is too long (max. %2 characters are allowed).

140=Currency Name is empty.

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

145=Batch ID Name is empty.

146=Batch ID Name with value %1 is too long (max. %2 characters are allowed).

147=Batch ID %1 is not numeric.

150=Wrong position of delimiter %1 between month and year in date %2 (current position = %3, expected position = %4 or %5).

151=Wrong position of delimiter %1 between day and month in date %2 (current position = %3, expected position = %4 or %5).

152=The year must have 4 digits (you specified %1 as year).

153=Day %1 is not numeric.

154=Month %1 is not numeric.

155=Year %1 is not numeric.

156=Year must be between %1 and %2 (you specified %3).

- 157=Month must be between %1 and %2 (you specified %3).
- 158=Day must be between %1 and %2 (you specified %3).
- 159=A blank character is expected between date and time (you specified %1 as date/time).
- 160=Wrong position of delimiter %1 between hours and minutes (current position = %2, expected position = %3 or %4).
- 161=Wrong position of delimiter %1 between minutes and seconds (current position = %2, expected position = %3 or %4).
- 162=Wrong position of delimiter %1 between seconds and nanoseconds (current position = %2, expected position = %3 or %4).
- 163=Hours %1 are not numeric.
- 164=Minutes %1 are not numeric.
- 165=Seconds %1 are not numeric.
- 166=Nanoseconds %1 are not numeric.
- 167=Hours must be between %1 and %2 (you specified %3).
- 168=Minutes must be between %1 and %2 (you specified %3).
- 169=Seconds must be between %1 and %2 (you specified %3).
- 170=Nanoseconds must be between %1 and %2 (you specified %3).
- 171=Wrong position of delimiter %1 between month and day of date %2 (current position = %3, expected position = %4 or %5).
- 172=Date %1 is not a valid date in format YYYY-MM-DD.
- 175=Stock Location is empty.
- 176=Stock Location with value %1 is too long (max. %2 characters are allowed).
- 180=Factory is empty.
- 181=Factory with value %1 is too long (max. %2 characters are allowed).
- 185=SAP Reference Number is empty.
- 186=SAP Reference Number with value %1 is too long (max. %2 characters are allowed).
- 190=Material Number is empty.
- 191=Material Number with value %1 is too long (max. %2 characters are allowed).
- 192=Sequence Number is empty.
- 193=Sequence Number with value %1 is too long (max. %2 characters are allowed).
- 194=Sequence Number %1 is not numeric.
- 195=Row Number is empty.
- 196=Row Number with value %1 is too long (max. %2 characters are allowed).
- 197=Row Number %1 is not numeric.
- 200='date from' not specified.
- 201=Date filter and Transaction ID filter can only be used exclusively.
- 202=Entered 'date from' is after 'date to'.
- 203=You must not specify both Currency Code and Exchange Rate.
- 204=You must not specify 'valid to' without 'valid from'.
- 205=Unexpected application error - please contact system administrator; error details: %1.
- 206=The time window (%1 days, %2 hours, %3 minutes) is not valid.
- 207=The submitted 'valid from' date-time %1 of the submitted Exchange Rate for Currency ID %2 is less than or equal to the 'valid from' date-time of an existing Exchange Rate for this Currency ID.
- 208=The days must be between 0 and 365 (you specified %1).
- 209=The hours must be between 0 and 24 (you specified %1).
- 210=The minutes must be between 0 and 59 (you specified %1).
- 211=Buying Rate must be larger than Selling Rate.
- 212=Bad input parameter.
- 213=State filter and Transaction ID Filter can only be used exclusively.

214=Entered 'date from' %1 must not be a future date (current time %2).
215=The Terms Of Use checkbox is not activated.
216=Reporting Criteria and TransactionID Filter can only be used exclusively.
217=Either both Reporting Criteria or none must be specified.
218=Business Types and TransactionID Filter can only be used exclusively.
219='date to' must not be more than %1 days after 'date from'.
220='date from' and 'date to' must not be empty if no invoice number is specified.
221=Hit list would contain more than %1 rows. Please be more specific!
222=Buying rate must be larger than 0.
223=Selling rate must be larger than 0.
224=Selling rate must be larger or equal than half buying rate.

300=Time interval is empty.
301=Time interval with value %1 is too long (max. %2 characters are allowed).
302=Time interval of field %1 is not numeric (you specified %2).
303=Time interval of field %1 must be greater than 0 (you specified %2).

305=User name is empty.
306=User name is wrong.
307=User name contains invalid characters.
308=User password is empty.
309=User password violates password rules.
310=User password contains invalid characters.
311=Invalid base64 encoded certificate

315=Object id with name %1 is empty.
316=Object id with name %1 and value %2 is not a number.

350=Technical error authenticating user
351=LDAP username must be empty for this actor type

LDAP error messages 400 to 449 reserved

400=Base distinguished name not found for certificate to be added.
401=Certificate distinguished name not found.
402=User not found.
403=Certificate distinguished name cannot be added. It already exists.
404=User cannot be added. It already exists.
405=The password and the confirmation password don't match.
406=Cannot find user %1 to delete from user management.

general messages - success messages 0601 - 0900

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

1001=%1 is not allowed to activate Cards.
1002=%2 is not allowed to have Cards assigned.
1003=Serial Numbers are not continuous.
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.

1010=The state of one or more Cards could not be set to State 'INVALID'.
1011=Currency %1 does not match Card Currency %2.
1012=Card State %1 is not valid for this request, expected Card State is %2.
1013='from-serial number' is greater than 'to-serial number'.
1014=No Password set for this Card.
1015=Access denied because of a repeated recent access violation.
1016=Wrong Password entered.
1017=New Password 1 and new Password 2 are different.
1018=The Card with Serial Number %1 could not be set to State 'INVALID'.
1019=Access violation.
1020=Challenge Question Answer 1 and Challenge Question Answer 2 are different.
1021=No Card with Batch ID %1 could be found.
1022=Invalid Batch Number for card with Serial Number %1.
1023=Duplicate Card found in print file.
1024=Card is missing in print file.
1025=Card is in an invalid State, expected State is 'GENERATED'.
1026=Number of copies printed is invalid, Card is set to State 'INVALID'.
1027=%1 is not allowed to deactivate Cards.
1028=Password could not be set since Challenge Question Answer 1 is different from Challenge Question Answer 2.
1029=You need to specify question, answer and answer verification to set the Challenge Question.
1030=Card with Serial Number %1 has an unexpected Material Number %2, expected is %3.
1031=Load Cards completed with error: %1.
1032=Card Batch %1 not found.
1033=At least one Card Type has to be selected.
1034=No Cards have been selected for this range.
1035=Card State %1 is not valid for this request, expected Card State %2 or %3.
1036=Card State %1 is not valid for this request, expected Card State %2, %3 or %4.
1037=Card in print file does not exist.
1038=Duplicate entry in card list found.
1039=Card with Serial Number %1 has not a location "%3" or "%4", but is at "%2".
1040=Card with Serial Number %1 has already been assigned to Cash-Ticket stock %2.
1041=Card with Serial Number %1 has already been assigned to Logistics Company %2.
1042=Card with Serial Number %1 cannot be assigned to a distributor before it has been assigned to a logistics company or PSC stock.
1043=Invalid material number %1 for card with serial number %2, expected: %3
1044=Card with Serial Number %1 already exists.
1045=Card with Random Number %1 already exists.
1046=Card with Random Number %1 has currently no disposition amount available.

card messages - success messages: 1601 - 1900

1601=Password has been set successfully.
1602=Password has been reset successfully.
1603=Password has been changed successfully.
1604=Challenge Question Answer has been set successfully.
1605=Challenge Question answered correctly, Password has been changed.
1606=The Cards have been invalidated successfully.
1607=All cards have been transferred to the Print Shop.
1608=Card(s) %1 to %2 activated successfully.
1609=Card(s) %1 to %2 deactivated successfully.
1610=Load Cards completed successfully: %1.
1611=Load Cards completed with warning: %1.
1612=Receive print file completed successfully: %1.

1613=Password counter has been reset successfully.

card messages - error messages: 1901 - 2000

1901=Card %1 could not be set to state 'INVALID'.

1902=Receive print file failed for card %1 with %2.

1903=Receive print file failed with %1.

1904=Cards with serial number from %1 to %2 have been transferred to printshop.

1905=Cards with serial number from %1 to %2 have been assigned.

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.

2005=The entered password is incorrect.

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.

payment messages - success messages: 2601 - 2900

2601=Payment completed successfully.

2602=Command completed successfully. No transactions found.

payment messages - error messages: 2901 - 3000

2901=Cleanup for open transactions failed: %1.

2902=Cleanup for expired transactions failed for card dispositions: %1.

2903=Cleanup for expired transactions failed for dispositions: %1.

2904=Cleanup for open transactions completed successfully: %1 dispositions expired.

2905=Cleanup for expired transactions completed successfully: expired %1 dispositions and %2 card dispositions.

2906=Debit for expired transaction (%1/%2) completed successfully.

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.
3004=Distributor %1 does not exist.
3005=Logistics Company %1 does not exist.
3006=Card Type %1 is not accepted by the merchant.
3007=Merchant %1 exceeded time window to debit the transaction.
3008=Cards cannot be assigned to logistics company %1.
3009=Cards cannot be assigned to distributor %1.
3010=Distributor %1 is not active.
3011=Logistics Company %1 is not active.
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.

master reference - success messages: 3601 - 3900

3601=Command completed successfully.
3602=Command completed successfully.
3603=Command completed successfully.

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 (€ 1000 or equivalent in selected 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 no 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.

technical success messages: 5601-6000

5601=%1 objects of type %2 have been marked for archiving successfully at %3.
5602=%1 rows in table %2 have been deleted successfully at %3.