Card Storage

Version 1.0

Document Version 1.0.1

Table of Contents

Version control

N	Author	Description	Date posted
1	Suzan Dermendzhieva	Card Storage Version 1.0	12.2016

Security and availability

Connection between Merchant and iCARD is handled through internet using HTTPS protocol (SSL over HTTP). Requests and responses are digitally signed both. iCARD host is located at tier IV datacenter in Luxembourg.

Exchange folder for partners (if needed) is located at a SFTP server which enables encrypted file sharing between parties. The partner receives the account and password for the SFTP directory via fax, email or SMS.

Introduction

This document describes the commands and interface for card storage functionality. The Merchant should integrate the Card Storage at the site accepting card payments. The API will gain access to the entry point of Card storage service managed by Intercard Finance AD (iCARD).

Card Storage generates a unique token value to be used in the calling system instead of a clear PAN, potentially reducing PCI scope considerably. The tokenization service provides storing of payment card details in a PCI compliant manner.

Card Storage API provides:

- Secured page and Secured communication channel;
- Storing of card details.

The purpose of this document is to specify the Card Storage functionality and demonstrate how it is used in the most common way.

All techniques used within the interface are standard throughout the industry and should be very easy to implement on any platform.

Continue on next page

Test Card Storage

A “by appointment” test service is available which allows the validation of the API calls. Testers should negotiate an exclusive access to the testing service and ensure monitoring by iCARD engineer.

Continue on next page

XML Card Storage standard interface

Card Storage interface runs as a server instance and accepts commands formatted with standard XML protocol version 1.0. This plain text protocol is easy to implement and monitor. All the commands are handled as RPC, so they are transparent for the remote developer, who handles the response code upon operation completion. Responses give the whole needed information for a successful command or exact error codes mapped to cases of failure.

All properties are defined with small letters as xml protocol is case sensitive.

All commands and responses have standard wrapping which is defined like this:

```
<?xml version="1.0" encoding="Windows-1251"?>
<ipayin_request>
  <command>NNN</command>
  <custom_ident>STORE20219</custom_ident>
</ipayin_request>
```

Standard wrapping for responses:

```
<?xml version="1.0" encoding="Windows-1251"?>
<ipayin_response>
  <command>NNN</command>
  <status>NN</status>
  <status_msg>Status Message text</status_msg>
  </ipayin_response>
```

Presence Notations

Presence notations indicate if and how data is present for each method.

Presence	Description
M	Mandatory. The data element is required in the message.
C	Conditional. The data element is required in the message if the conditions described in the accompanying text apply.
O	Optional. The data element is not required, but may be included in the message at the message initiator's option.
ME	Mandatory Echo. The data element is required in a response message and must contain the same value ("echoed") from the original request or advice message.
CE	Conditional Echo. The data element is required in a response message if it was present in the original request or advice message, and it must contain the same value ("echoed") from the original message.

Data Type Formats

Data Type in document	Description	Example
int	integer	1
String	string	This is a string
Date	ISO 8601 date string YYYY-MM-DD	2012-03-31
DateTime	ISO 8601 datetime string YYYY-MM-DD HH:mm:SS	2012-03-31 23:59:59
A(n)	Alpha string. [n] characters required	Alpha string
AN(n)	Alphanumeric string. [n] characters required	Alphanumeric string
ANS(n)	Alphabetic, numeric, space, and special characters. [n] characters required	Alphanumeric, space and special characters
N(n)	Numeric string. [n] characters required. Number is left-padded with zeroes.	000123
double	Numeric string with decimal point. Only point is used (no commas or other characters for decimal point)	34.56
BASE64	Sting used to pass binary data. The binary data should be converted to base64 standard.	YW55IGNhcm5hbCBwbGVhc3VyZQ==
XML	Simple in place XML array.	<body> <param>1</param> <value>2</value> </body>

Command standard properties

Property	Typical value	Type	Presence	Description
command	101	N(3)	M	Code for the command which is executed.
custom_ident	STORE20219	AN(15)	M	Value that uniquely identifies the merchant.

Response standard properties

Property	Typical value	Type	Presence	Description
command	999	N(3)	ME	Code for the command which has been executed (echo).
status	0	N(3)	M	Code upon command completion. 0 is success otherwise error. Please refer to Appendix I.
status_msg	Command completed successfully	String	M	Description for <status> code
status_details	-	String	O	Additional detail upon success or failure.

Card Storage command codes (numerical order)

Command Number	Description
101	Store card.
102	Update a stored card details.
103	Retrieve a stored card details.

101 Command. Store card

Purpose

This command allows cardholder to securely store card data.

Method properties

Property	Typical value	Type	Presence	Description
pan	5326000000004885	N(19)	M	Card account number.
exp_date	1703	N(4)	M	Card expiry date.
card_type	1	N(1)	M	Card type. Please refer to Appendix II.
cardholder_name	John Smith	String	M	Cardholder name.
custom_name	Mastercard Business	String	O	Short name for card which will help the client recognize it easily.
mid	000000000099999	N(15)	C	Card acceptor code assigned to the site/outlet or to the merchant. Used in the request if card_verification = 2.
card_verification	2	N(1)	M	Specify whether the inputted card data to be verified or not before storing. For possible values please refer to Appendix III.
amount	3.50	Double	C	Amount of the transaction. Used in the request if card_verification = 2.
currency	978	N(3)	C	ISO numeric currency code. Used in the request if card_verification = 2.
cvc2	999	N(3)	C	Card verification code. Used in the request if card_verification = 2.

eci	0 – MC – Merchant not participating in 3D program or card enrollment service is unavailable 1 – MC – Attempted card 2 – MC – full 3D authentication 5 – VISA - full 3D authentication 6 – VISA - Attempted card or not participating but the merchant is certified for 3D 7 – VISA - Merchant not participating in 3D program or card enrollment service is unavailable	N(1)	C	Electronic commerce indicator. Shows the enrollment of the cardholder in MasterCard 3D Secure or Verified by Visa programs. Used in the request if card_verification = 2.
avv*	BwABBEUzaIElYgBgkDNoAAAAAAA=	ANS(28)	C	UCAF value for MasterCard and AVV for VISA. Base64 string.
xid*	jJJLtQa+Iws8AREAEbjsA1MAAAA=	ANS(28)	C	XID stain for a 3D transaction (VISA). Originally 20 bytes/characters (e.g. 20110808000000000450), base64 encoded
stan	111111	N(6)	C	Sequential number for the transmission. Unique in combination with dttm. Minimum value: 000001. Maximum value: 999999. Once the maximum value is reached then the counter is restarted and presented as 000001. Used in the request if Card_verification = 2.
dttm	2016-12-01 12:34:55	DateTime	C	Date and time. Format is: YYYY-MM-DD HH:mm:SS. Used in the request if card_verification = 2.

**The parameters <avv> and <xid> must be supplied only if <eci> is 1, 2, 5 or 6.*

Request example

```
<?xml version="1.0" encoding="Windows-1251"?>
<ipayin_request>
        <command>101</command>
        <custom_ident>STORE20219</custom_ident>
        <pan>5326000000004885</pan>
        <exp_date>1712</exp_date>
        <card_type>1</card_type>
        <cardholder_name>John Smith</cardholder_name>
        <custom_name>MasterCard Business</custom_name><mid>000000000099999</mid>
        <card_verification>2</card_verification>
        <amount>2.00</amount>
```

```
        <currency>978</currency>
        <cvc2>000</cvc2>
        <eci>2</eci>
        <avv>BwABBEUzaIElYgBgkDNoAAAAAAA=</avv>
        <xid>JJJLTQA+IWS8AREAEBJSA1MAAAA=</xid>
        <stan>111111</stan>
        <dttm>2016-12-01 12:34:55</dttm>
</ipayin_request>
```

Response properties

Property	Typical value	Type	Required	Description
token	1041333312721BC752C1AB7743D0821AA1C9CA09	string	M	Uniquely generated card token.
pan	************4885	N(19)	M	Last four digits of the account number (PAN).
exp_date	1703	N(4)	M	Expiry date.
card_type	1	N(1)	M	Card type.
cardholder_name	John Smith	String	M	Cardholder's name.
custom_name	Mastercard Business	String	O	Custom name.
resp_code	00	AN(2)	C	Response provided by issuer or acquirer system. Will be returned if card_verification = 2.
trn	20161201123455123456	N(20)	C	Transaction Reference Number (TRN). Will be returned if card_verification = 2.
approval	999999	AN(6)	C	Approval or authorization code returned by card issuer. Will be returned if card_verification = 2.
stan	111111	N(6)	CE	Echo from the request.
dttm	2016-12-01 12:34:55	DateTime	CE	Echo from the request.

Response example

```
<?xml version="1.0" encoding="Windows-1251"?>
<ipayin_response>
  <command>101</command>
      <token>1041333312721BC752C1AB7743D0821AA1C9CA09</token>
      <pan>************4885</pan>
      <exp_date>1703</exp_date>
      <card_type >1</card_type >
      <cardholder_name>John Smith</cardholder_name>
      <custom_name>MasterCard Business</custom_name>
      <resp_code>00</resp_code>
      <trn>20161201123455123456</trn>
      <approval>999999</approval>
      <stan>111111</stan>
      <dttm>2016-12-01 12:34:55</dttm>
  <status>0</status>
  <status_msg>Command completed successfully</status_msg>
```

```
</ipayin_response>
```

102 Command. Update a stored card details

Purpose

This command allows cardholder to securely update data of an already stored card.

Method properties

Property	Typical value	Type	Required	Description
token	1041333312721BC752C1AB7743D0821AA1C9CA09	String	M	Uniquely generated token of the stored card.
exp_date	1912	N(4)	M	Card expiry date.
card_type	1	N(1)	M	Card type. Please refer to Appendix II.
cardholder_name	John Smith	String	M	Cardholder name.
custom_name	Mastercard Business	String	O	Short name for card which will help the client recognize it easily.
mid	000000000099999	N(15)	C	Card acceptor code assigned to the site/outlet or to the merchant. Used in the request if card_verification = 2.
card_verification	1	N(1)	M	Specify whether the inputted card data to be verified or not before storing. For possible values please refer to Appendix III.
amount	3.50	Double	C	Amount of the transaction. Used in the request if card_verification = 2.
currency	978	N(3)	C	ISO numeric currency code. Used in the request if card_verification = 2.
cvc2	999	N(3)	C	Card verification code. Used in the request if card_verification = 2.

eci	0 – MC – Merchant not participating in 3D program or card enrollment service is unavailable 1 – MC – Attempted card 2 – MC – full 3D authentication 5 – VISA - full 3D authentication 6 – VISA - Attempted card or not participating but the merchant is certified for 3D 7 – VISA - Merchant not participating in 3D program or card enrollment service is unavailable	N(1)	C	Electronic commerce indicator. Shows the enrollment of the cardholder in MasterCard 3D Secure on Verified by Visa programs. Used in the request if card_verification = 2.
avv*	BwABBEUzaIElYgBgkDNoAAAAAAA=	ANS(28)	C	UCAF value for MasterCard and AVV for VISA. Base64 string.
xid*	jJJLtQa+Iws8AREAEbjsA1MAAAA=	ANS(28)	C	XID stain for a 3D transaction (VISA). Originally 20 bytes/characters (e.g. 20110808000000000450), base64 encoded
stan	222222	N(6)	C	Sequential number for the transmission. Unique in combination with dttm. Minimum value: 000001. Maximum value: 999999. Once the maximum value is reached that the counter is restarted and presented as 000001. Used in the request if card_verification = 2.
dttm	2016-12-01 12:40:55	DateTime	C	Date/time of the request. Used in the request if card_verification = 2.
card_number_check		N(19)	O	

**The parameters <avv> and <xid> must be supplied only if <eci> is 1, 2, 5 or 6.*

Request example

```
<?xml version="1.0" encoding="Windows-1251"?>
<ipayin_request>
        <command>102</command>
        <custom_ident>STORE20219</custom_ident>
        <token>1041333312721BC752C1AB7743D0821AA1C9CA09</token>
        <exp_date>1912</exp_date>
        <card_type>1</card_type>
        <cardholder_name>John Smith</cardholder_name>
        <custom_name>MasterCard Business</custom_name><mid></mid>
        <card_verification>1</card_verification>
```

```
        <amount></amount>
        <currency></currency>
        <cvc2></cvc2>
        <eci></eci>
        <avv></avv>
        <xid></xid>
        <stan></stan>
        <dttm></dttm>
</ipayin_request>
```

Response properties

Property	Typical value	Type	Presence	Description
token*	10415556668872721BC752C1AB7743D0821AA1C9CA11	String	M	Uniquely generated card token.
pan	************4885	N(19)	M	Last four digits of the account number (PAN).
exp_date	1912	N(4)	M	Expiry Date.
card_type	1	N(1)	M	Card type.
cardholder_name	John Smith	String	M	Cardholder's name.
custom_name	MasterCard Business	String	O	Custom name.
resp_code	00	AN(2)	C	Response provided by issuer or acquirer system. Will be returned if card_verification = 2. Please refer to Appendix IV.
trn	20161201123455123456	N(20)	C	Transaction Reference Number (TRN). Will be returned if card_verification = 2.
approval	999999	AN(6)	C	Approval or authorization code returned by card issuer. Will be returned if card_verification = 2.
stan	222222	N(6)	CE	Echo from the request.
dttm	2016-12-01 12:40:55	DateTime	CE	Echo from the request.

In case the cardholder edit the expiry date a new card token will be generated.

Response example

```
<?xml version="1.0"?>
<ipayin_response>
    <command>102</command>
    <token>10415556668872721BC752C1AB7743D0821AA1C9CA11</token>
    <pan>************4885</pan>
    <exp_date>1912</exp_date>
    <card_type >1</card_type >
    <cardholder_name>John Smith</cardholder_name>
    <custom_name>MasterCard Business</custom_name>
    <resp_code></resp_code>
      <trn></trn>
      <approval></approval>
```

```
      <stan></stan>
      <dttm></dttm>
   <status>0</status>
   <status_msg>Command completed successfully</status_msg>
</ipayin_response>
```

103 Command. Retrieve a stored card data

Purpose

This method allows to retrieve the details of a successfully stored card by token.

Method properties

Property	Typical value	Type	Presence	Description
token	10415556668872721BC752C1AB7743D0821AA1C9CA11	String	M	Uniquely generated card token.
card_parameters	1	N(1)	O	Card details. Specify whether additional details related to card (issuer country, region, product and etc.) to be included in the response. Possible values are 0 (No) and 1 (Yes).

Request example

```
<?xml version="1.0" encoding="Windows-1251"?>
<ipayin_request>
          <command>103</command>
          <custom_ident>STORE20219</custom_ident>
          <token>10415556668872721BC752C1AB7743D0821AA1C9CA11</token>
</ipayin_request>
```

Response properties

Property	Typical value	Type	Presence	Description
pan	**********4885	N(19)	M	Last four digits of the account number (PAN).
exp_date	1912	N(4)	M	Expiry date.
card_type	1	N(1)	M	Card type.
cardholder_name	John Smith	String	M	Cardholder's name.
custom_name	Mastercard Business	String	O	Short name of the card.

Response example

```
<?xml version="1.0" encoding="Windows-1251"?>
<ipayin_response>
        <command>103</command>
        <results>
                <pan>************4885</pan>
                <exp_date>1912</exp_date>
                <card_type >1</card_type >
                <cardholder_name>John Smith</cardholder_name>
                <custom_name>MasterCard Business</custom_name>
        </results>
        <status>0</status>
        <status_msg>Command completed successfully</status_msg>
</ipayin_response>
```

Appendix I – Error messages

Below are the error messages received from Card Storage interface:

Code	Description	Note
0	Command completed successfully	
1	General error	
2	Database error	
3	Invalid input parameters	Missing or wrong format parameters in request.
4	Incoming data parse error	Invalid XML format.
5	Unsupported command	Value specified in <command> property is unknown.
6	Communication error	Unable to transmit or receive data from the card schemes.
7	Card not found	There is no card corresponding to the token or to the customer identifier.
8	Card verification failed	

Appendix II – Card type

Code	Card type
1	MasterCard
2	Maestro
3	VISA
4	Visa Electron
5	VPAY
6	JCB

Appendix III – Card verification

Code	Description
1	**Without verification.** The card will be stored without any validation.
2	**Verify card with debit operation.** The card data provided will be verified by running a real debit transaction (includes also a zero-amount transaction). Upon successful transaction the card data will be stored successfully.

Appendix IV – Response codes

<resp_code> property where applicable:

- 00, 85 – transaction is approved
- <> 00 and 85 – transaction is declined

<resp_code> is returned from the Card Schemes and is corresponding to ISO-8583 field 39.

<resp_code> is an alpha-numeric value. Don’t convert to integer.