cashU Merchant Service Integration Guide

Table of Content

Introduction

Welcome to cashU!
cashU is the electronic payment solution developed specifically for the Arab speaking world with the aim to help any business becoming a successful online merchant in the region. It is available for almost everyone selling products or services without restrictions, setup fees or bureaucracy.

The cashU Merchant Service offers you two ways to accept payments, either through the cashU Payment Gateway or through cashU Direct Pay.
The cashU Payment Gateway is the preferred solution and gives you additional benefits. The Payment Gateway is integrated on your website and allows your clients to pay the exact amount for the product/service they are purchasing.
cashU Direct Pay allowed your clients to use a cashU Refill Coupon (prepaid card) direct on your website, but must redeem the full value of the Refill Coupon at time of purchase.

This document provides information on how to integrate the cashU Payment Gateway and cashU Direct Pay It describes the features, services, parameters and technical environment that are required by you and provided by cashU to achieve a successful implementation.
The document describes the integration using sample code.

cashU supports the set up of cashU payments in both Test and Production Environment.

A typical implementation of cashU Payment Gateway takes around 4 working hours depending on the desired integration of features and the complexity of the merchant's platform.

For a comprehensive description of cashU Merchant Services please read the cashU Merchant Account User Manual.

Intended audience and required skills

This document is intended for software developers with knowledge and authority to integrate third party payment solutions on a Merchant's e-commerce platform. Since cashU Payment Gateway does not require any software installation the required skills is determined by the technologies used on the Merchant’s website and the necessity to integrate the Payment Gateway with back end systems.

The integration details in this document apply to all Merchants regardless of programming language. The document contains some code examples for the most common programming languages.

Payment Gateway Integration

Payment Transaction overview

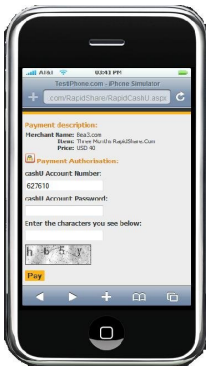
A payment transaction on cashU's Payment Gateway works as follows:

1. A Client visits your website and selects to purchase a Product/Service.
2. The Client follows your purchase flow and proceeds to the payment section (checkout page) of your website.
3. The Client selects to pay with cashU.
4. The client will be redirected to a cashU Payment Page where the description of the Product/Service and the total price to is presented.
5. The Client authorities the payment.
6. cashU validates the payment. If the transaction is approved, it is executed in real time and stored in the cashU system.
7. In case of any error, cashU will display a relevant error message for the Client
8. After execution of a transaction the Client is redirected to a predefined section of the Merchant's website (Return URL).

cashU Payment Page

On your checkout page you will display a cashU button, once a Client clicks the button the cashU payment procedure is initiated and the required parameters are passed from the Merchant to the cashU Payment Page.
The cashU Payment Page resides on a secure cashU server and is the page on which a Client will authorize his/her cashU payment.

cashU also provide a mobile interface for the payment page. The mobile payment page does not need different integration; cashU will determine if the transaction is being processed via mobile and view the appropriate interface.

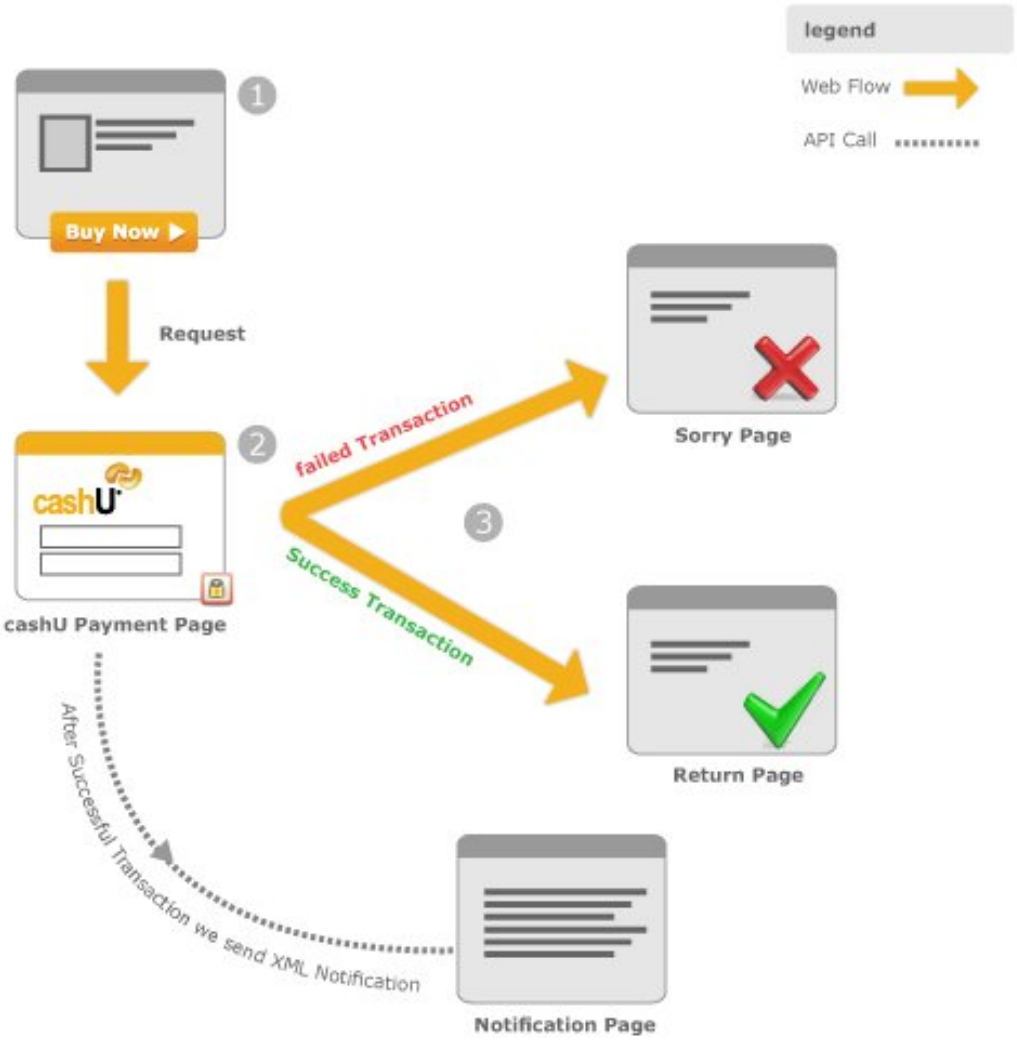


Payment Gateway Integration Types

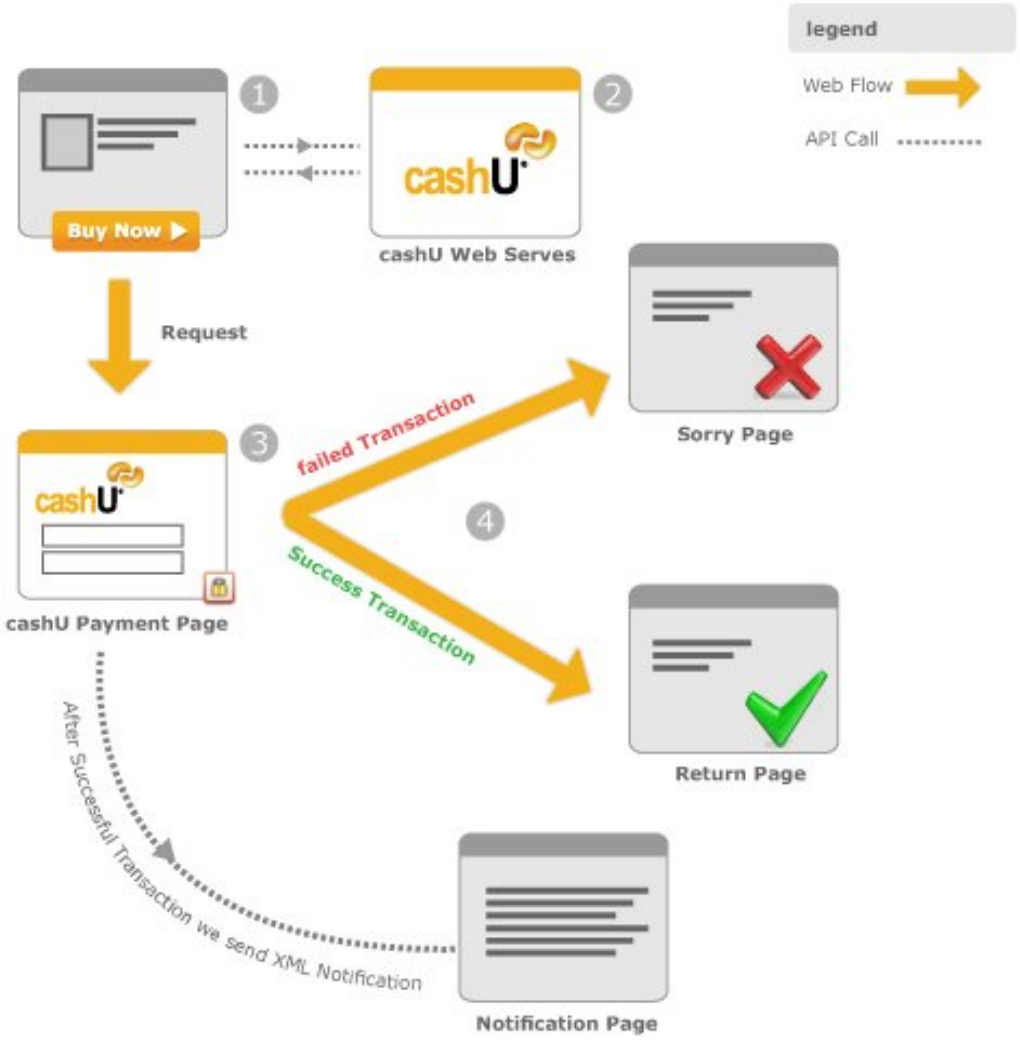
cashU offers two types of types of integrations for the payment gateway:

- Standard Integration
- Premier Integration

The following drawings illustrate the difference between the two types of integrations



Standard Integration Flow



Premier Integration Flow

Both integrations will redirect the customer to cashU payment page; the only difference between the two types of integrations is the way the Merchant send the parameters to cashU.

- In Standard Integration, the Merchant send all the required parameters through secure HTTP POST to cashU server (for more details please read implementation section of this document).
- In Premier Integration, the Merchant send all the required parameters to a cashU web service that returns a confirmation code, which should be sent to cashU server through secure HTTP POST (for more details please read implementation section of this document).

Getting started

Merchants do not need to do much to get cashU working. First step is to register for a cashU Merchant Account which is done online on https://www.cashu.com/Merchants/index.
After account approval you only need to add the cashU button with a link to the cashU Payment Page and a scripted page that cashU will redirect the Client to after a successful transaction.

The Merchant's website does not need a secured server since the Client enters the authentication details on cashU payment page which is hosted on cashU secured servers.

Required Setup before Implementation:

Login to your cashU Merchant Account and under the "Payment Security" tab you have to select "Single Checkout" or "Multiple Checkout" depending on how many domains or checkout pages you want to integrate cashU on (you can find out more about this in the cashU Merchant account guide). In either case, the Merchant should enter the following data (it is recommended that all setup is initially done in Test Mode only):

- "Encryption Keyword": The Encryption Keyword will be used to create MD5 hash of a string for each payment. Merchant should NEVER expose the Encryption Keyword to the Client. It is recommended to change the Encryption Keyword frequently.

- The Return URL is the webpage hosted by the Merchant to where the Client will be redirected after a successful payment transaction. NEVER expose the Return URL to the Client

- If you select “Multiple Checkout”, you also have to enter a value for “Domain Name”. You can select any Domain Name but each Domain Name must have a unique Return URL. The Domain Name will not be displayed to the Client, but it must be passed as a parameter to the Payment Page. You can have as many Domain Names as you like.

Implementation

Description of the Parameters:

Standard Integration

Interaction between the Merchant and cashU server is based on secure HTTP POST and requires the following parameters:

Parameter Name	Description
merchant_id	• The Merchant ID as entered during registration. • Must be between 1 and 20 characters long with characters in [A-Za-z_\.\-@0-9]
amount	• Number identifying the value of the payment transaction. • Must be presented in maximum 2 decimal points.
currency	• Currency in which the service/item is being sold on the Merchant's website. The accepted values are: • USD (American Dollar) • CSH (cashU Points) • AED (UAE Dirham) • EUR (Euro) • JOD (Jordanian Dinar) • EGP (Egyptian Pound) • SAR (Saudi Riyal) • DZD (Algerian Dinar) • LBP (Lebanese Pound) • MAD (Moroccan Dirham) • QAR (Qatar Riyal) • TRY (Turkish Lira)
language	• String identifying the interface language of the payment page. • Supported values are: "en" for the English interface and "ar" for the Arabic interface and "ir" for Farsi interface.
display_text	• English or Arabic description of the transaction, this will be used to display for the customer on the payment page. • Maximum allowed length is 250 characters. • Allowed characters in [A-Za-z_\.\-@\'\"0-9] and Arabic characters.
txt1	• English description of the transaction that will not be displayed for the customer. • Maximum allowed length is 150 characters. • Allowed characters are in [A-Za-z_\.\-0-9]
token	Must be a hex-encoded MD5 HASH (32 digit hexadecimal number) of a concatenation of the following parameter values separated by":" (values should be in lower case), appended with the Encryption Keyword. The Encryption Keyword is selected by Merchant (please read step 1 for more details.)

	The example below shows how to create the hash. The used values are fictitious. First, values should be converted to lower case: Parameter: merchant_id value: test Parameter: amount value: 15.25 Parameter: currency value: aed Then their values are contacted: test:15.25:aed The Encryption Key is appended to the end of the string: test:15.25:aed:Hello World The MD5 function is called on the resulting string: MD5("test:15.25:aed:Hello World") The result of the function is the hash to be included in the request. This must be presented in 32 digit hexadecimal number. Please read the appendix I for details on how to create MD5 hash in some selected programming languages.
test_mode	• Number identifies if the transaction is done on live or test modes. • Accepted values are: "1" for the test mode and "0" for the live mode.

Below is a description of optional parameters that cashU provides for Merchants to include any data related to the payment transaction and needed to be passed to the Return URL

Parameter Name	Description
txt2	• String of optional data. • Maximum allowed length is 250 characters in [A-Za-z_\.\-@0-9]
txt3	• String of optional data. • Maximum allowed length is 250 characters in [A-Za-z_\.\-@0-9]
txt4	• String of optional data. • Maximum allowed length is 250 characters in [A-Za-z_\.\-@0-9]
txt5	• String of optional data. • Maximum allowed length is 250 characters in [A-Za-z_\.\-@0-9]
session_id	• Unique reference to the transaction generated by Merchant. • Maximum allowed length is 100 characters in [A-Za-z_\.\-@0-9]
service_name	• Name of the service or domain entered by Merchant that has multiple checkout pages (details on this available on cashU Merchant account guide) • This parameter can not be empty if the Merchant uses multiple checkout pages. • Maximum allowed length is 50 characters in [A-Za-z_\.\-@0-9]

Premier Integration

Interaction between Merchant and cashU server is different in Premier Integration and it consists of two steps. In the first step the interaction is based on SOAP calls initiated by the Merchant to cashU server, but it requires same list of parameters the Standard Integration requires. In the second step the interaction is based on secure HTTP POST and it requires one parameter which returned from cashU server as a result to the SOAP call in the first step. The following describes the parameter needed in the second call:

Parameter Name	Description
Transaction_Code	• This is a unique reference to the transaction, which returns by cashU if parameters check was successful. • Length is 40 characters in [a-z0-9]
test_mode	• Number identifies if the transaction is done on live or test modes. • Accepted values are: “1” for the test mode and “0” for the live mode.

Encryption Type

Merchant can change from Default Encryption described above to Enhanced Encryption. To do this you must login to your cashU Merchant Account and under the Payment Security tab you will be able to change from Default to Enhanced Encryption. The selected encryption type is always applied to both Test and Live Mode.

In Enhanced Encryption you add the Session id it to the list of parameters which increases the complexity of the MD5 hash.
The Enhanced Encryption is MD5 HASH of the lower case of following parameters:
Merchant_id:amount:currency:session_id appended with the Encryption Key.
The Merchant MUST have value fort the *session id* in order to use the Enhanced Encryption.

Posting of Parameters

Standard Integration

The above parameters should be posted to the following cashU URL:

https://www.cashu.com/cgi-bin/pcashu.cgi

Merchant should use the secure POST method to send the parameters; cashU does not accept any other method.

Posting the parameters can be done in different ways depending on the programming language that is used by the Merchant.

Below is an example (but not recommended) of how to post the parameters, it can be used with any programming language and is based on HTML FORM tag (values are fictitious):

```
<form action="https://www.cashu.com/cgi-bin/pcashu.cgi" method="post">
<input type="hidden" name="merchant_id" value="XYZ">
<input type="hidden" name="token" value="66a31cd699d8d9cb454df1f6cec30c2c">
```

```
<input type="hidden" name="display_text" value="Baseball Hat">
<input type="hidden" name="currency" value="CSH">
<input type="hidden" name="amount" value="125">
<input type="hidden" name="language" value="en">
<input type="hidden" name="session_id" value="asdasd-234-asdasd">
<input type="hidden" name="txt1" value="item27">
<input type="hidden" name="test_mode" value="1">
<input type="submit" value="Pay with cashU!">
</form>
```

Merchant should set the parameters as hidden fields of the HTML FORM tag and set the action attribute for it to be the cashU POST URL.

The FORM should be submitted when the Client decides to pay with cashU on the Merchant's checkout page. The FORM could be static or created dynamically by a script.

Premier Integration

In the first step of the premier integration, Merchant should send the list of parameters described in pages 9 and 10 of this guide to the following URL using SOAP:
https://secure.cashu.com/payment.wsdl

Available methods:

- DoPaymentRequest: This method will return the transaction code that should be resend to cashU server in the second step.

Input Description

The previous method expects the same parameters described in pages 9 and 10 in the following order:
merchant_id. token, display_text, currency, amount, language, session_id, txt1, txt2, txt3, txt4, txt5, test_mode, service_name

For more details about the expected values, please read pages 9 and 10 of this guide.

Output Description

cashU sends the response as a string. Below is an example of the returned string:

Transaction_Code=8e3e540d3a0bb2d49ac9cc4e732441bbec22dea2

In case of any error, cashU will return the corresponding error message. Please read appendix III for more details on error message returned on first call of premier integration.

In the second call, the Merchant should submit the value returned by cashU in the first call along with the value of the environment that he is performing the transaction in to the following URL:
https://www.cashu.com/cgi-bin/payment/pcashu.cgi

The Merchant should use the secure POST method to send the parameter and no other way will be accepted.

Below is an example of how to post the parameter, it can be used with any programming language and is based on HTML FORM tag (values are fictitious):

```
<form action="https://www.cashu.com/cgi-bin/payment/pcashu.cgi" method="post">
<input type=”hidden” name=”Transaction_Code”
value=”8e3e540d3a0bb2d49ac9cc4e732441bbec22dea2” >
<input type="hidden" name=”test_mode” value="0”>
<input type="submit" value="Pay with cashU!”>
</form>
```

Payment

The above steps complete posting of a message to cashU server. cashU performs the required checks on the received data and if there are no errors in the data cashU displays the Payment Page to the Client where he/she can enter the cashU authentication details.
cashU verifies the account authentication details, sufficient balance, restricted country and maximum amount, and upon successful authentication, cashU redirect the user to the Return URL predefined by the Merchant in the Payment Security page (Please refer to cashU Merchant Account Guide for details).

Test Mode:

If the Merchant is performing a test transaction, the authentication details for the test account can be found in your Merchant Account (login) under the “Test Mode” section in the Payment Security page. Test account cannot be used to perform a transaction on the live mode.

Return Page

The page to which the Client is redirected after a successful transaction is the Return URL.

NEVER EXPOSE THE RETURN URL IN YOUR CODE!
All the images, scripts and style sheets that Merchant calls in Return URL should be presented in full path and never use a reference path. cashU does not expose the Merchant’s Return URL to the Client, it keeps the URL as:
https://www.cashu.com/cgi-bin/payment.cgi

Transaction Notification:

cashU can notify the Merchant about successful transactions in three different ways:

Return URL Notification. This is the default type of notification. cashU use the HTTP POST to submit the following data to Merchant's Return URL.

Parameter Name	Description
amount	Same amount sent by the Merchant.
currency	Same currency sent by the Merchant.
language	Interface language sent by the Merchant.
txt1	Item description sent by the Merchant.
token	The MD5 String sent by the Merchant.
test_mode	The value that sent by the Merchant and indicate the mode that the transaction was done on.
trn_id	Unique reference number to the transaction created by cashU.
session_id	Merchant's unique reference to the transaction (if it was sent by the Merchant)
verificationString	It is a hex-encoded SHA1 HASH (40 HEX characters) of a concatenation of the following parameter values separated by":" merchant id, cashU transaction id and the Encryption Key. Please note that merchant id should be in lower case. The example below shows how to create the hash. The used values are fictitious. First, values should be converted to lower case: Parameter: merchant_id value: test Parameter: trn_id value: 4566 Parameter: Encryption Key value: hello world Then their values are contacted: test: 4566:hello world The SHA1 HASH function is called on the resulting string: SHA1 HASH ("test: 4566:hello world") The result of the function is the hash to be included in the response. Please read the appendixes for details on how to create SHA1 HASH hash in some selected programming languages.

In addition to the above table, cashU also return the values of the parameters from txt2 to txt5 as they were sent by the Merchant.

Email Notification. (Please refer to cashU Merchant Account Guide for details).

XML Notification, is the recommended way of transactions' notifications since cashU will continue sending the Notification until a confirmation is received from the Merchant. cashU calls a URL on the Merchant side on the backend sending data in XML format.

In order to receive the XML notification, the Merchant has to login to his/her cashU Merchant Account and submit the backend URL in the “Payment Security” page. This URL should be hosted on a secure server (HTTPS) for the live mode. (Please refer to Appendix II for script samples on how to process and parse the XML string).

The following shows the XML structure that returns by cashU (values are fictitious):

```
<cashUTransaction>
   <merchant_id>XYZ</merchant_id>
   <token>66a31cd699d8d9cb454df1f6cec30c2c</token >
   <display_text>Baseball Hat</display_text>
   <currency>AED</currency >
   <amount>100</amount>
   <language>en</language>
   <session_id>asdasd-234-asdasd</session_id>
   <txt1>item27</txt1>
   <txt2>123</txt2>
   <txt3></txt3>
   <txt4></txt4>
   <txt5></txt5>
   <servicesName></servicesName >
   <responseCode>OK</responseCode >
   <trnDate>2008-01-01 09:20:01</trnDate>
   <cashU_trnID>401231</cashU_trnID>
   <cashUToken>234c36b77ffac7905165bb72d582342</cashUToken>
</cashUTransaction>
```

The following are short descriptions of the fields in the XML notification:

Parameter Name	Description
responseCode	Available value if “OK”
trnDate	Transaction date and time in GMT
cashU_trnID	Unique reference number to the transaction created by cashU.
cashUToken	The MD5 string of the following values: Merchant ID, cashU transaction and Encryption key. The Merchant ID and the Encryption Key should be in lower case. The example below shows how to create the hash. The used values are fictitious. First, values should be converted to lower case: Parameter: merchant_id value: test Parameter: cashU_trnID value: 401231 Parameter: Encryption Key value: hello world Then their values are contacted: test:401231:hello world The MD5 function is called on the resulting string: MD5(“test:401231:hello world”) The result of the function is the hash to be included in the response.

Please note that cashU will send the above XML Notification as a value of parameter called “sRequest”.

The following is an example of an XML Notification sent by cashU to the Merchant:

sRequest=”<cashUTransaction><merchant_id>XYZ</merchantID><token>66a31cd699d8d9cb454df1f6cec30c2c</token><display_text>Baseball Hat</display_text><currency>AED</currency><amount>100</amount><language>en</language><session_id>asdasd-234-asdasd</session_id><txt1>item27</txt1><txt2>123</txt2><txt3></txt3><txt4></txt4><txt5></txt5><servicesName></servicesName><responseCode>OK</responseCode><trnDate>2008-01-01
09:20:01</trnDate><cashU_trnID>401231</cashU_trnID><cashUToken>234c36b77ffac7905165bb72d582342</cashUToken></cashUTransaction>”;

Merchant Response to XML Notification:

Only if the Merchant is using XML Notification does cashU expects a response from the Merchant, the response should be sent to the following URL:

https://www.cashu.com/cgi-bin/notification/MerchantFeedBack.cgi

The response is important to assure cashU that the Merchant has received the Notification. cashU expects the following parameters to be sent in XML format:

Parameter Name	Description
merchant_id	The Merchant ID registered with cashU
cashU_trnID	The unique reference number to the transaction created by cashU.
cashUToken	The MD5 string sent by cashU for the notification
responseCode	The expected value is “OK”
responseDate	The timestamp of the response. It should date and time in GMT. Accepted format is: “yyyy-mm-dd hh:mm:ss”

The response should be sent via secure HTTP POST.
cashU expects to receive the response in the following format:

```
sRequest=”
<cashUTransaction><merchant_id>XYZ</merchant_id><cashU_trnID>401231</cashU_trnID><c
ashUToken>234c36b77ffac7905165bb72d582342</cashUToken><responseCode>OK</responseCo
de><responseDate>2008-01-01 09:21:01</responseDate></cashUTransaction>”;
```

To test the response of the XML Notification; please send it to the following URL:

https://www.cashu.com/cgi-bin/test_notification/MerchantFeedBack.cgi

In the case cashU does not receive a response from the Merchant; cashU will resend the XML Notification (maximum of six times with a time interval of two hours) for each transaction.

Failed Transaction Notification:

cashU does not send notification to the Merchant in the case of a failed transaction, but will show the error message to the Client in the Payment Page window. However, cashU offers an optional service to the Merchants to redirect the Client back to the Merchant's website with an error code represents the error message.
The Merchant can then display the error message to the Client on his/her website.
In order to use this service, the Merchant has to login to his/her cashU Merchant Account and enter the "Sorry URL" under My Account/Payment Security. Please read the following section for more details about the Sorry URL.

Sorry URL

If the payment transaction failed for any reason (insufficient funds, restricted country, KYC compliance restriction or incorrect authentication details) cashU will redirect the Client to the Sorry URL with list of parameters. cashU will submit the parameters to the requested URL using HTTP POST.

The following are list of parameters returned to the Sorry URL:

Parameter Name	Description
errorCode	Details on the error codes are available on appendix III
txt1	Item description sent by the Merchant.
session_id	Merchant's unique reference to the transaction (if it was sent by the Merchant)

.
The Sorry URL is optional. If the Merchant is not using it, cashU will display a default error message on the Payment Page

Security of Transactions

Security is essential to online payments. cashU ensures the security of every transaction by the following:

- The transferred data from the Merchant website to cashU Payment Page. This is done by using the “token” parameter which is created using an “Encryption Key” known only by the Merchant and saved on cashU database. The Merchant should NEVER, in any format expose this “Encryption Key” to the Client. cashU recommends changing this key regularly.
- The payment process. The customer has to enter his/her cashU payment account ID and password on the securely hosted Payment Page.
- Notification of a successful transaction. cashU redirects the Client back to the Merchant “Return URL” with a unique transaction reference and an encrypted value. cashU can also sends “XML Notification” in the backend to a secure path.
- The transaction id. cashU offers a web service (please read section 4 for more details) to check the status of any transaction id.
- The “Return URL” The content of the “Return URL” is displayed but the path itself is not displayed to the Client, cashU keep it as https://www.cashu.com/payment.cgi
- Avoid submitting the same payment more than once. cashU requires the Client to enter random turing numbers and characters for each payment, so the Client always has to return to the Merchant website to complete a new purchase request in order to perform a new payment on cashU.

Web Services

cashU offers web services to Merchants through web methods. These services are available through SOAP communication methods. These services are available to ensure that Merchants are getting all services in the most efficient and simple way.

Please note that access to these services is IP restricted. Merchants who are interested in this service should provide us with their IP addresses on: integration@cashu.com, please provide us with your Merchant ID and the range of IP addresses that you expect to use to call the web service.

WSDL Path

The web service supports a SOAP interface and the WSDL definition file is found on the following path:

https://secure.cashu.com/MerchantServices.wsdl

Methods

- ❖ checkMerchantTransactionStatus: This method will return the status of the transaction on cashU server.

Input Description

- transactionID: string
  This is the transaction ID on cashU. cashU returns the transaction ID after each successful transaction.
- merchantID: string
  This is Merchant ID as entered during registration.

Output Description

cashU sends the response as a string containing XML. Below is an example of the XML:

```
<cashU>
 <merchantID>XYZ</merchantID>
 <trnID>523</trnID>
 <responseCode>1</responseCode>
 <responseDate>2008-01-01 09:21:01</responseDate>
</cashU>
```

Error Code	Description
1	Invalid transaction ID
2	Incorrect Merchant ID
4	XML response is 'GENERAL_ERROR'
5	No XML response available
6	Successful transaction

- ❖ getMerchantStatment: This method will retrieve the account statement for the Merchant in a specific period of time. This method will return the same results that the Merchant gets from the excel sheet inside the account login area.

Input Description

- merchantID: string
  This is Merchant ID as entered during registration.
- fromDate: string
  This is the starting date to get the transactions. Acceptable format is: yyyy-mm-dd
- toDate: string
  This is the ending date to get the transactions. Acceptable format is: yyyy-mm-dd
- token: string
  This is the MD5 output of a string. String is “merchant_id:yyyy-mm-dd hh”

Output Description

cashU sends the response as a string containing XML. Below is an example of the XML:

```
<cashU>
  <transaction>
    <trnID>384644</trnID>
    <date>14/09/2007 23:07</date>
    <sessionID>2722453870</sessionID>
    <item>test</item>
    <credit>13.24</credit>
    <debit>0.00</debit>
    <paymentCurrency>EUR</paymentCurrency>
    <paymentAmount>10</paymentAmount>
    <balance>768</balance>
  </transaction>
  <transaction>
    <trnID>384645</trnID>
    <date>14/09/2007 23:07</date>
    <sessionID>2722453870</sessionID>
    <item>test</item>
    <credit>0.00</credit>
    <debit>0.35</debit>
     <paymentCurrency> </paymentCurrency>
     <paymentAmount></paymentAmount>
     <balance>767.65</balance>
  </transaction>
</cashU>
```

Response Parameters	Description
trnID	Transaction ID on cashU system
date	Transaction date format is: “yyyy-mm-dd hh:mm:ss”
sessionID	The unique reference to the transaction generated by the Merchant.
item	Description of the item displayed to the customer.
credit	Transaction’s value credited to Merchant’s balance in USD.
debit	Transaction’s value debited from Merchant’s balance in USD.
paymentCurrency	Transaction’s original currency
paymentAmount	Transaction’s original value.
balance	Merchant’s balance.

In case of any error, cashU send the following XML:

```
<cashU>
        <merchantID>DXYZ</merchantID>
        <startDate>2008-10-15</startDate>
        <endDate>2008-10-20</endDate>
        <token>token</token>
        <errorCode>Merchant_Not_Found</errorCode>
        <responseCode>3</responseCode>
        <responseDate>2008-11-01 08:11:56</responseDate>
</cashU>
```

Error Code	Description
1	One of the required parameter is missing or NULL
2	Date format is not correct
3	Invalid Date
4	Incorrect Merchant ID
5	Inactive Merchant
6	Your IP address is not authorized to use this service
7	Incorrect MD5 string
8	No data available in the selected period

- ❖ getMerchantTransferDates: This method will retrieve the transfers from Merchant account to the related Payment account that happened in a specific period of time, and transactions related to each transfer.

Input Description

- merchantID: string
  This is Merchant ID as entered during registration.
- fromDate: string

This is the starting date to get the transactions. Acceptable format is: yyyy-mm-dd

- toDate: string
  This is the ending date to get the transactions. Acceptable format is: yyyy-mm-dd
- token: string
  This is the MD5 output of a string. String is “merchant_id:yyyy-mm-dd hh”

Output Description

cashU sends the response as a string containing XML. Below is an example of the XML:

```
<cashU>
   <transfer transferId='768' transferDate='2009-01-25' transferValue='1200.00'>
   <transactionsDates>
      <trnDate>2009-01-22</trnDate>
      <trnDate>2009-01-18</trnDate>
   </transactionsDates>
   </transfer>
</cashU>
```

Response Parameters	Description
transferId	Transaction ID on cashU system that is related to this transfer.
transferDate	Transfer date format is: “yyyy-mm-dd”
transferValue	The value of the transfer in USD.
trnDate	Transactions dates that are related to the transfer, format is: “yyyy-mm-dd”

In case of any error, cashU send the following XML:

```
<cashU>
        <merchantID>DXYZ</merchantID>
        <startDate>2010-01-01</startDate>
        <endDate>2010-01-31</endDate>
        <token>token</token>
        <errorCode>Incorrect_Token</errorCode>
        <responseCode>2</responseCode>
        <responseDate>2010-02-01 08:11:56</responseDate>
</cashU>
```

Error Code	Description
1	One of the required parameter is missing or NULL
2	Incorrect Merchant ID
3	Inactive Merchant ID
4	Incorrect token
5	Your IP address is not authorized to use this service
6	Date format is not correct
7	Invalid Date
8	No data available in the selected period

Appendix I (How to create the MD5 digest)

In cryptography, MD5 (Message-Digest algorithm 5) is a widely used cryptographic hash function with a 128-bit hash value. As an Internet standard (RFC 1321), MD5 has been employed in a wide variety of security applications

The following are examples on how to create the MD5 digest. The following examples are on the default encryption type. Values in the examples are fictitious

PHP

```
<?php
// $token will have the MD5 digest value of the string.
$token = md5("maktoob:35.50:usd:keyword1");
?>
```

Perl

```
use Digest::Perl::MD5 'md5_hex';
$token = md5_hex('maktoob:35.50:usd:keyword1');
```

JAVA

```
import java.util.*;
import java.io.*;
import java.security.*;
public class Test {
public static String hex(byte[] array) {
StringBuffer sb = new StringBuffer();
for (int i = 0; i < array.length; ++i) {
Version 4.10 15 25/06/2008
sb.append(Integer.toHexString((array[i] & 0xFF) |
0x100).toUpperCase().substring(1,3));
}
return sb.toString();
}
public static String md5 (String message) {
try {
MessageDigest md = MessageDigest.getInstance("MD5");
return hex (md.digest(message.getBytes("CP1252")));
} catch (NoSuchAlgorithmException e) {
```

```
} catch (UnsupportedEncodingException e) {
}
return null;
}
public static void main(String[] args) {
System.out.println (md5 ("maktoob:35.50:usd:keyword1"));
}
}
```

ASP

```
Dim token As String
token = " & merchant & " & ":" & amount & ":" & currency & ":" & " &
secret_word & "
Dim btyScr() As Byte = ASCIIEncoding.ASCII.GetBytes(token)
Dim objMd5 As New MD5CryptoServiceProvider()
Dim btyRes() As Byte = objMd5.ComputeHash(btyScr)
Dim intTotal As Integer = (btyRes.Length * 2 + (btyRes.Length / 8))
Dim strRes As StringBuilder = New StringBuilder(intTotal)
Dim intI As Integer
For intI = 0 To btyRes.Length - 1
strRes.Append(BitConverter.ToString(btyRes, intI, 1))
Next intI
token_result = strRes.ToString().TrimEnd(New Char() {" "c}).ToLower
```

Appendix II (XML Notification script)

The creation of the script for processing the "XML Notification" and the "redirect" are not explained in this guide because they are depend on the Merchant's website. In this paragraph however you will find some examples for parsing the "XML Notification" which might be helpful when you are creating the scripts for your website.

The below describe how to parse the following example:

```
$sRequest="<?xml version='1.0' encoding='utf-8' ?>
<cashUTransaction>
    <merchant_id>merchantID</merchant_id>
    <token>token</token>
    <display_text>display_text</display_text>
    <currency>currency</currency>
    <amount>amount</amount>
    <language>language</language>
    <session_id>session_id</session_id>
    <txt1>txt1</txt1>
    <txt2>txt2</txt2>
    <txt3>txt3</txt3>
    <txt4>txt4</txt4>
    <txt5>txt5</txt5>
    <servicesName>servicesName</servicesName>
    <responseCode>responseCode</responseCode>
    <trnDate>trnDate</trnDate>
    <cashU_trnID>cashU_trnID</cashU_trnID>
    <cashUToken>cashUToken</cashUToken>
</cashUTransaction>
```

Perl

```
use XML::DOM;

use CGI qw(:standard);

$XML=param('sRequest');


$merchant_id=$token=$display_text=$currency=$amount=$language=$session_id
='';

$txt1=$txt2=$txt3=$txt4=$txt5=$servicesName=$responseCode=$trnDate=$cashU
_trnID=$cashUToken='';


$parser = new XML::DOM::Parser;

$doc = $parser->parse($XML);

$root = $doc->getDocumentElement();


foreach $node ($root->getElementsByTagName("*"))

{

        if (defined ($node->getFirstChild())){
```

```
if($node->getNodeName() eq 'merchant_id')
{
$merchant_id=$node->getFirstChild()->getData();
}
elsif($node->getNodeName() eq 'token')
{
$token=$node->getFirstChild()->getData();
}
elsif($node->getNodeName() eq 'display_text')
{
$display_text=$node->getFirstChild()->getData();
}
elsif($node->getNodeName() eq 'currency')
{
$currency=$node->getFirstChild()->getData();
}
elsif($node->getNodeName() eq 'amount')
{
$amount=$node->getFirstChild()->getData();
}
elsif($node->getNodeName() eq 'language')
{
$language=$node->getFirstChild()->getData();
}
elsif($node->getNodeName() eq 'session_id')
{
$session_id=$node->getFirstChild()->getData();
}
elsif($node->getNodeName() eq 'txt1')
{
$txt1=$node->getFirstChild()->getData();
}
elsif($node->getNodeName() eq 'txt2')
{
```

```
            $txt2=$node->getFirstChild()->getData();
            }
            elsif($node->getNodeName() eq 'txt3')
            {
            $txt3=$node->getFirstChild()->getData();
            }
            elsif($node->getNodeName() eq 'txt4')
            {
            $txt4=$node->getFirstChild()->getData();
            }
            elsif($node->getNodeName() eq 'txt5')
            {
            $txt5=$node->getFirstChild()->getData();
            }
            elsif($node->getNodeName() eq 'servicesName')
            {
            $servicesName=$node->getFirstChild()->getData();
            }
            elsif($node->getNodeName() eq 'responseCode')
            {
            $responseCode=$node->getFirstChild()->getData();
            }
            elsif($node->getNodeName() eq 'trnDate')
            {
            $trnDate=$node->getFirstChild()->getData();
            }
            elsif($node->getNodeName() eq 'cashU_trnID')
            {
            $cashU_trnID=$node->getFirstChild()->getData();
            }
            elsif($node->getNodeName() eq 'cashUToken')
            {
            $cashUToken=$node->getFirstChild()->getData();
            }
        }
```

```
}


$doc->dispose;
```

PHP

```
$XML=$_POST['sRequest'];
$merchant_id=$token=$display_text=$currency=$amount=$language=$session_id
='';

$txt1=$txt2=$txt3=$txt4=$txt5=$servicesName=$responseCode=$trnDate=$cashU
_trnID=$cashUToken='';

 $xmlr = new XMLReader();
 $xmlr->XML($XML);
 while ($xmlr->read())
 {

        if($xmlr->nodeType != XMLReader::ELEMENT)
        {
        continue;
        }

        if($xmlr->name == 'merchant_id')
        {
        $merchant_id = $xmlr->readString();
        }
        elseif($xmlr->name == 'token')
        {
         $token = $xmlr->readString();
        }
        elseif($xmlr->name == 'display_text')
        {
        $display_text = $xmlr->readString();
        }
        elseif($xmlr->name == 'currency')
        {
```

```
$currency = $xmlr->readString();
}
elseif($xmlr->name == 'amount')
{
$amount = $xmlr->readString();
}
elseif($xmlr->name == 'language')
{
$language = $xmlr->readString();
}
elseif($xmlr->name == 'session_id')
{
$session_id = $xmlr->readString();
}
elseif($xmlr->name == 'txt1')
{
$txt1 = $xmlr->readString();
}
elseif($xmlr->name == 'txt2')
{
$txt2 = $xmlr->readString();
}
elseif($xmlr->name == 'txt3')
{
$txt3 = $xmlr->readString();
}
elseif($xmlr->name == 'txt4')
{
$txt4 = $xmlr->readString();
}
elseif($xmlr->name == 'txt5')
{
$txt5 = $xmlr->readString();
}
elseif($xmlr->name == 'servicesName')
```

```
        {
        $servicesName = $xmlr->readString();
        }
        elseif($xmlr->name == 'responseCode')
        {
        $responseCode = $xmlr->readString();
        }
        elseif($xmlr->name == 'trnDate')
        {
        $trnDate = $xmlr->readString();
        }
        elseif($xmlr->name == 'cashU_trnID')
        {
        $cashU_trnID = $xmlr->readString();
        }
        elseif($xmlr->name == 'cashUToken')
        {
        $cashUToken = $xmlr->readString();
        }
 }
$xmlr->close();
```

JAVA

```
import org.w3c.dom.*;
import javax.xml.parsers.*;
import org.xml.sax.*;
import java.io.*;
import java.util.*;

String xmlString=request.getParameter("sRequest");

DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
DocumentBuilder builder = null;
Document doc = null;
```

```
try
{
builder = factory.newDocumentBuilder();
}
catch (ParserConfigurationException pce)
{
System.out.println("ParserConfigurationException");
return;
}

try
{
doc = builder.parse(new InputSource(new StringReader(xmlString)));
}
catch(IOException ioe)
{
System.out.println("IOException");
return;
}
catch(SAXException sax)
{
System.out.println("SAXException");
sax.printStackTrace();
return;
}
catch(IllegalArgumentException iae)
{
System.out.println("IllegalArgumentException");
return;
}
Element e = doc.getDocumentElement();

e.getChildNodes();

NodeList list=e.getChildNodes();
```

```
String
merchant_id="",token="",display_text="",currency="",amount="",language="",sessi
on_id="";
String
txt1="",txt2="",txt3="",txt4="",txt5="",servicesName="",responseCode="",trnDate
="",cashU_trnID="",cashUToken="";


for(int i = 0; i < list.getLength(); i++)
{
Node n = list.item(i);
String NodeValue=n.getChildNodes().item(0).getNodeValue();
String NodeName=n.getNodeName();


        if(NodeName == "merchant_id")
        {
                merchant_id=NodeValue;
        }
        else if(NodeName == "token")
        {
                token=NodeValue;
        }
        else if(NodeName == "display_text")
        {
                display_text=NodeValue;
        }
        else if(NodeName == "currency")
        {
                currency=NodeValue;
        }
        else if(NodeName == "amount")
        {
                amount=NodeValue;
        }
        else if(NodeName == "language")
        {
```

```
        language=NodeValue;
    }
    else if(NodeName == "session_id")
    {
        session_id=NodeValue;
    }
    else if(NodeName == "txt1")
    {
        txt1=NodeValue;
    }
    else if(NodeName == "txt2")
    {
        txt2=NodeValue;
    }
    else if(NodeName == "txt3")
    {
        txt3=NodeValue;
    }
    else if(NodeName == "txt4")
    {
        txt4=NodeValue;
    }
    else if(NodeName == "txt5")
    {
        txt5=NodeValue;
    }
    else if(NodeName == "servicesName")
    {
        servicesName=NodeValue;
    }
    else if(NodeName == "responseCode")
    {
        responseCode=NodeValue;
    }
    else if(NodeName == "trnDate")
```

```
        {
                trnDate=NodeValue;
        }
        else if(NodeName == "cashU_trnID")
        {
                cashU_trnID=NodeValue;
        }
        else if(NodeName == "cashUToken")
        {
                cashUToken=NodeValue;
        }

}
```

ASP

```
Dim xmlString As String = Request.Form("sRequest")


Dim merchant_id As String = ""
Dim token As String = ""
Dim display_text As String = ""
Dim currency As String = ""
Dim amount As String = ""
Dim language As String = ""
Dim session_id As String = ""
Dim txt1 As String = ""
Dim txt2 As String = ""
Dim txt3 As String = ""
Dim txt4 As String = ""
Dim txt5 As String = ""
Dim servicesName As String = ""
Dim responseCode As String = ""
Dim trnDate As String = ""
Dim cashU_trnID As String = ""
Dim cashUToken As String = ""
```

```
Dim doc As System.Xml.XmlDocument = New System.Xml.XmlDocument()
doc.LoadXml(xmlString)
Dim arrDoc As System.Xml.XmlNodeList = doc.DocumentElement.ChildNodes
Dim length1 As Integer = arrDoc.Count
Dim i As Integer
Dim node As String = ""

For i = 0 To length1 - 1
  node  = doc.DocumentElement.ChildNodes(i).Name
   If node = "merchant_id" Then
      merchant_id = doc.DocumentElement.ChildNodes(i).InnerText
   ElseIf node = "token" Then
      token = doc.DocumentElement.ChildNodes(i).InnerText
   ElseIf node = "display_text" Then
      display_text = doc.DocumentElement.ChildNodes(i).InnerText
   ElseIf node = "currency" Then
      currency = doc.DocumentElement.ChildNodes(i).InnerText
   ElseIf node = "amount" Then
      amount = doc.DocumentElement.ChildNodes(i).InnerText
   ElseIf node = "language" Then
      language = doc.DocumentElement.ChildNodes(i).InnerText
   ElseIf node = "session_id" Then
      session_id = doc.DocumentElement.ChildNodes(i).InnerText
   ElseIf node = "txt1" Then
      txt1 = doc.DocumentElement.ChildNodes(i).InnerText
   ElseIf node = "txt2" Then
      txt2 = doc.DocumentElement.ChildNodes(i).InnerText
   ElseIf node = "txt3" Then
      txt3 = doc.DocumentElement.ChildNodes(i).InnerText
   ElseIf node = "txt4" Then
      txt4 = doc.DocumentElement.ChildNodes(i).InnerText
   ElseIf node = "txt5" Then
      txt5 = doc.DocumentElement.ChildNodes(i).InnerText
   ElseIf node = "servicesName" Then
```

```
      servicesName = doc.DocumentElement.ChildNodes(i).InnerText
   ElseIf node = "responseCode" Then
      responseCode = doc.DocumentElement.ChildNodes(i).InnerText
   ElseIf node = "trnDate" Then
      trnDate = doc.DocumentElement.ChildNodes(i).InnerText
   ElseIf node = "cashU_trnID" Then
      cashU_trnID = doc.DocumentElement.ChildNodes(i).InnerText
   ElseIf node = "cashUToken" Then
      cashUToken = doc.DocumentElement.ChildNodes(i).InnerText

   End If
Next
```

Appendix III (Sorry URL error code)

The following describes the error codes returned to “Sorry URL” and the error messages related to them

Error Code	Error Message
2	Inactive Merchant ID.
4	Inactive Payment Account.
6	Insufficient funds.
7	Incorrect Payment account details.
8	Invalid account.
15	Password for the Payment Account has expired.
17	The transaction has not been completed.
20	The Merchant has limited his sales to some countries; the purchase attempt is coming from a country that is not listed in the profile.
21	The transaction value is more than the limit. This limitation is applied to Payment Accounts that not complied by KYC rules.
22	The Merchant has limited his sales to only KYC complied Payment accounts; the purchase attempt is coming from a Payment account that is KYC complied.
23	The transaction has been cancelled by the customer. If the user clicks on the “Cancel” button.

The following are error messages that the Merchant might get during the integration process

Error Message	Description
A required parameter is missing	One of the required parameters is not included in the submitted data to cashU server.
Inactive Merchant	The Merchant account is inactive.
Incorrect merchant token value	The submitted value for the token parameter is incorrect.
Invalid Character Found	An invalid character found in one of the submitted parameters.
Data is too long for the Item Description	Length of one of the submitted parameter is more than the expected.

The following are error messages returned by cashU as output from the first call of the premier integration

Error Message	Description
INSECURE_REQUEST	Request was not sent through HTTPS
SYSTEM_NOT_AVAILABLE	cashU servers are not responding
INVALID_PARAMETER	One or more of the required parameters is null or has invalid character.
INACTIVE_MERCHANT	The Merchant account is inactive.
TOKEN_CHECK_FAILURE	The submitted value for the token parameter is incorrect.
GENERAL_SYSTEM_ERROR	Error happened while processing the transaction