# eKashu Developer Integration Guide

An NMI hosted payment page

Document Version: AE
Date: May 2024

# Table of Contents

## Document History

| DOCUMENT VERSION | DATE RELEASED DD/MM/YYYY | PAGES AFFECTED | REMARKS | NAME |
|---|---|---|---|---|
| DRAFT A | 18/09/2006 | Various | Created | Nigel Jewell |
| DRAFT B | 27/09/2006 | Various | Revised draft | Pete Alcock |
| DRAFT C | 14/02/2007 | Various | Revised draft | Nigel Jewell |
| DRAFT D | 03/04/2007 | Various | Revised draft | Nigel Jewell |
| DRAFT E | 13/06/2007 | 14 | Card Hash/Ref | Nigel Jewell |
| DRAFT F | 06/08/2007 | Various | Input Options | Nigel Jewell |
| DRAFT G | 10/08/2007 | 7, 15 | Hash Code | Nigel Jewell |
| DRAFT H | 07/11/2007 | 6, 8 | Request Type | Nigel Jewell |
| DRAFT I | 07/02/2007 | 13 | Verification | Nigel Jewell |
| DRAFT J | 26/11/2008 | 17 | Hash Code | Nigel Jewell |
| DRAFT K | 20/03/2009 | Various | PayPal | Nigel Jewell |
| DRAFT L | 30/04/2009 | 16 | Return Text | Nigel Jewell |
| DRAFT M | 22/09/2009 | Various | Notifications | Nigel Jewell |

| | | | | |
|---|---|---|---|---|
| **DRAFT N** | 10/02/2010 | 20 | ACS Warning | Nigel Jewell |
| **DRAFT O** | 02/08/2010 | 20 | C# Hash Code | Nigel Jewell |
| **DRAFT P** | 10/01/2011 | 9 | Locale | Nigel Jewell |
| **DRAFT Q** | 17/08/2011 | 6.8 | Hash Key | Nigel Jewell |
| **DRAFT R** | 02/04/2012 | 9 | Recurring | Nigel Jewell |
| **DRAFT S** | 15/11/2012 | 10 | Viewport | Nigel Jewell |
| **DRAFT T** | 11/11/2014 | 21 | Test Cards | Peter Alcock |
| **DRAFT U** | 19/01/2105 | 18 | Scheme Text | Nigel Jewell |
| **DRAFT V** | 01/02/2015 | 16 | Seller Address | Nigel Jewell |
| **DRAFT V.2** | 25/11/2019 | Various | Branding | Ben Thomas |
| **DRAFT W** | 06/06/2020 | Various | Hash code v2 / 3DSv2 | Rowan Willis |
| **DRAFT W.2** | 30/11/2020 | 11,27,30,37,38 | Hash code v2 changes, important dates added | Jarrett Chamberlin |
| **DRAFT X** | 13/08/2021 | 25,26 | Added CoF properties, deprecated PayPal support and correction to .NET hash code validation example | Robbie Johnstone,Henry Mohanraj |
| **DRAFT X.2** | 16/09/2021 | 25 | Fixed typo 3 occurrences of a typo in the word “initiated” & updated general wording of CoF page. | Robbie Johnstone |
| **DRAFT X.3** | 19/11/2021 | All pages (rebrand)<br><br>5 | Document Rebrand.<br><br>Removed the mention of optional integration with the | Henry Mohanraj, Jarrett Chamberlin |

| | | | | |
|---|---|---|---|---|
| | | 8<br><br>36-43 | 3rd Man fraud screening service, as it's no more supported<br><br>Added EMV 3DS test cards.<br><br>Added 3DS 2.1.0 notice regarding First Name / Last Name requirement for EMV 3DS (3DS2) to be possible. | |
| **DRAFT X.4** | 15/02/2022 | 8, 9, 11, 13, 14, 18 | Documented new ekashu_card_name_required variable.<br><br>Updated EMV 3DS notice to refer to new field.<br><br>Documented ekashu_3d_secure_verify true_token / true_all values.<br><br>Clean up of variable names so that they can be searched.<br><br>Fixes to the below variables:<br><br>**ekashu_card_phone_number**<br><br>**ekashu_failure_return_text_url**<br><br>**ekashu_card_country**<br><br>**ekashu_delivery_email_address_mandatory** | Jarrett Chamberlin, Luke Denham |
| **DRAFT Y** | 16/05/2022 | 11 | Documented new variable: **ekashu_moto_payment** | Jarrett Chamberlin |
| **DRAFT Z** | 05/10/2022 | 10, 11, 14, 15, 21, 30 | Documented new variable: **ekashu_card_name** | Jarrett Chamberlin |

| | | | | |
|---|---|---|---|---|
| | | | _mandatory<br><br>Documented new output variable: ekashu_payment_account_reference<br><br>Documented ekashu_3d_secure_verify mask values for Amex / JCB / Diners / Discover. | |
| **DRAFT AA** | 05/01/2023 | 9, 41 | Removed 3DS1 Test PANs from documentation.<br><br>Updated advice around the CSC for American Express cards.<br><br>Documented ekashu_amount_format variable. | Jarrett Chamberlin |
| **DRAFT AB** | 06/02/2023 | 28 | Documented ekashu_3d_secure_v2_challenge_indicator eKashu variable.<br><br>Fixed documented default value for ekashu_amount_format. | Jarrett Chamberlin |
| **DRAFT AC** | 07/18/2023 | 9, 14, 32 | Fixed documentation for ekashu_amount_format to be properly listed as optional<br><br>Documented ekashu_3d_secure_version eKashu output variable | Joshua Barlas |
| **DRAFT AD** | 07/18/2023 | 14 | Deprecation of ekashu_request_type value: recurring | Joshua Barlas |
| **DRAFT AE** | 05/14/2024 | 17, 19, 20, 29 | Documented 50 character limit for ekashu_card_address_1, ekashu_card_address_2, | Joshua Barlas |

| | | | | |
|---|---|---|---|---|
| | | | ekashu_delivery_address_1, and ekashu_delivery_address_2<br><br>Document new variable: ekashu_3d_secure_v2_reject_missing_data that treats transactions with missing 3DS data as rejections | |

# Section 1 – Introduction

## EKASHU PAYMENT PAGE FEATURES

eKashu Payment Page is a simple, configurable, checkout page from NMI that can be called from any website seeking to offer visitors the facility to pay for goods or services by credit card, and debit card. It is designed for merchants who use shopping carts, have little experience in server side scripting, or who use shared web servers that do not offer secure database services.

- Easy to set up: eKashu Payment Page is readily integrated with most of the major shopping carts.
- Simple Integration: with just a few lines of code, the merchant's website can be ready to accept payment.
- Complete security: end customers enter their card details on the eKashu payment pages and we take care of the transaction security for them –we are certified under the PCI DSS (Payment Card Industry Data Security Standard). As the merchant never sees the card details, they are not responsible for security.
- Customisable payment pages: You can maintain your merchant's brand image by customising the payment pages to the look and feel of their website.

With eKashu Payment Page, all transaction information is held at eKashu including the shopping basket total. The customer is redirected to eKashu to enter their card details, so no sensitive information needs to be taken or stored on the merchant's site, thereby removing the need for them to maintain highly secure encrypted databases, or obtain digital certificates.

The final "Pay Now" button on the merchant's website is the link to the eKashu System. The customer selects their purchases, enters delivery details, billing address etc. on the merchant's site or alternatively if required these addresses can be entered on the eKashu payment page. They then press the final 'proceed' button and the customer's browser calls the eKashu payment page, where they enter their required contact details and card details. At the bottom of the eKashu payment page is a "Pay Now" button which submits the information to the eKashu Payment Page gateway.

## EKASHU PAYMENT PAGE

The eKashu Payment Page main page optionally carries the merchant's logo and a description of the goods the customer is paying for, so they can remain confident who they are buying from. You can even customise the payment pages to carry the look and feel of the merchant's site at no additional cost.

Once the customer has selected their payment method and confirmed they wish to complete the payment, eKashu requests authorisation from the bank. Once the bank has authorised the payment (and assuming the address and card value checks have passed any rules you may have set up), we redirect the customer back to the successful payment page on the merchant's site, or alternatively a default eKashu success page. If the authorisation fails, we redirect the customer to your order failure page,or alternative a default eKashu failure page. Both pages are sent information which you can access using standard web technology, to find out what happened to the transaction and extract any useful information

# Section 2 – Ekashu Checkout Integration Guide

## INTEGRATION OVERVIEW

eKashu is a flexible and secure hosted payment page that conforms to Internet standards and common integration methods. It allows for a large amount of customisation so that it can be specifically tailored to the seller's needs.

**eKashu has the following major features:**

- Optional validation of 3-D Secure enabled cardholders for qualified merchant accounts
- Customisation of payment pages using Cascading Style Sheets (CSS)
- Standards compliant web pages (HTTPS, HTML 4.01 Strict, CSS 2 and JavaScript)
- Supports the internationalisation of the payment process (Zip Code, Post Code, Code Postal ...)
- Email confirmation for the buyer and seller of a successful order
- URL redirection to success and failure pages at the conclusion of an authorisation (with POST data)

Integrating eKashu Checkout with an existing website is extremely easy and can be achieved with a few simple steps. The only requirement is to create a HTML form in the referring web page that contains a small number of mandatory fields. Once you have completed integration and performed your own testing we will provide you with the connection credentials to our live platform.

Test credentials consisting of a Terminal ID and Transaction Key can be obtained by registering with the Test WebMIS Platform at https://testwebmis.creditcall.com. The Test WebMIS Platform will allow for an integrator to view test transactions that have been submitted to the test eKashu platform. If a hash key is required (as described later) this should be requested from eKashu Support (support@nmi.com).

## FORM CREATION

- Create a form in the location that the "Pay" button should appear
- Set the form's action to https://test.ekashu.com
- Set the form's method to POST
- Create a hidden field in the form called ekashu_seller_id containing the eKashu ID of the seller. The ekashu_seller_id is the entire Terminal ID supplied and registration
- Create a hidden field in the form called ekashu_seller_key containing the eKashu key of the seller. The ekashu_seller_key is the first eight characters of the Transaction Key supplied at registration

- Create a hidden field in the form called ekashu_amount containing the amount required from the buyer
- Create a hidden field in the form called ekashu_currency containing the currency associated with the amount
- Create a submit button in the form

More information regarding the content of these fields can be found below.

*Note: for live transactions the URL should be set to https://live.ekashu.com. A different Terminal ID and Transaction Key will be supplied when live registration is performed.*

## EXAMPLE

The following sample shows a HTML form that will initiate the eKashu Checkout process for the Terminal ID 12345678 with the Transaction Key paiipH9yjgs4zvRl with a value of £123.00.

```
<form action="https://test.ekashu.com" method="post">

        <input type="hidden" name="ekashu_seller_id" value="12345678"/>
        <input type="hidden" name="ekashu_seller_key" value="paiipH9y"/>
        <input type="hidden" name="ekashu_amount" value="123.00"/>
        <input type="hidden" name="ekashu_currency" value="GBP"/>
        <input type="submit" value="Pay"/>
 </form>
```

## INPUT PROPERTIES

The following fields can be hidden within the eKashu Checkout form in order to pre-fill the checkout details or customise the buyer's experience. Apart from the four fields listed above, they are all optional.

In addition to the predefined eKashu fields listed, any number of seller fields can also be specified in the form as POST data. These fields will be passed through the eKashu checkout process and returned to the success and failure pages as POST data. In order to ensure that future enhancements do not disrupt your checkout process, they cannot begin with the prefix "ekashu_".

**PLEASE NOTE:** It is not valid to have POST fields named "submit" or "reset". Forms with these names conflict with the JavaScript methods form.submit() and form.reset()

## MANDATORY PROPERTIES

Each of these properties must be specified for the transaction to take place as they are used to identify the seller and determine the value of the transaction.

| Field Name | Purpose | Example |
|---|---|---|

| | | |
|---|---|---|
| **EKASHU_AMOUNT** | The amount that the seller requires from the buyer. This should be expressed in the major format required for the currency and should be greater than zero. For example, one British pound (one hundred pence) would be expressed as 1.00, not 100. Please note that on the test platform a minimum and maximum amount are configured and an amount of 5.00 automatically declines. The minimum and maximum amounts can be specified for the live platform at registration. | 123.00 |
| **EKASHU_CURRENCY** | The ISO country code or mnemonic for the currency of the amount specified. This will be displayed to the buyer in an appropriate format and therefore it should be supplied even if it is the same as the default currency associated with your ID. | EUR or 978 |
| **EKASHU_SELLER_ID** | The seller's eKashu ID. This is required to identify the account that should receive the payment. It is the same as 12345678 the Terminal ID assigned by NMI at registration. | 12345678 |
| **EKASHU_SELLER_KEY** | The seller's eKashu key. This is required to identify the account that should receive the payment. It is the first eight digits of the Transaction Key assigned by NMIl at registration. | paiipH9y |

## EMV 3DS (2.1.0) NOTICE

In order to be able to perform EMV 3DS 2.1.0+ authentications eKashu **requires** First Name and Last Name to be provided during the initial POST to eKashu or the card number entry form set up to allow collection of Cardholder information and for this data to be collected.

Without this data, EMV 3DS 2.1.0+ isn't possible, and 3DS 1.0.2 will be used instead, provided eKashu is configured to do this.

Therefore, the following options can be utilised to achieve this in eKashu:

a. To avoid the need to collect the rest of the Cardholder properties such as email address and the cardholder address, you can use the below fields to provide First Name / Last Name to eKashu as part of your initial POST:
   i. **EKASHU_CARD_FIRST_NAME**
   ii. **EKASHU_CARD_LAST_NAME**
b. Alternatively, you may enable the capture of "Cardholder Properties" during the first step of eKashu by providing a **true** value for the below eKashu property:
   i. **EKASHU_CARD_ADDRESS_EDITABLE**

## UPDATE TO NOTICE

A new eKashu variable has been added to allow for the easier collection of cardholder names.

The new variable is **EKASHU_CARD_NAME_REQUIRED**, and accepts a boolean value.

This variable is most useful for integrations that are providing the below variables / value combination:

- ekashu_card_address_required = false
- ekashu_card_address_verify = false
- ekashu_card_zip_code_verify = false

Normally, this combination of eKashu variables would cause eKashu not to collect any cardholder data and not show the Cardholder form.

When the cardholder form is editable, passing a true value for **EKASHU_CARD_NAME_REQUIRED** to eKashu will allow the cardholder to enter their name only.

When the cardholder form is non-editable, it will allow your integration to provide the cardholder name as a part of the initial post and ensure that the cardholder name is included when attempting 3DS2.

## UPDATE TO NOTICE #2

A new eKashu variable has been added to allow integrations to enforce collection of cardholder names as **mandatory**.

The new variable is **EKASHU_CARD_NAME_MANDATORY**, and accepts a boolean value.

This variable is most useful for integrations that are providing the below variables / value combination:

- ekashu_card_address_required = false
- ekashu_card_address_verify = false
- ekashu_card_zip_code_verify = false
- Ekashu_card_name_required = true

Normally, this combination of eKashu variables would cause eKashu to not validate the cardholder's name before moving on to the 3DS stage (and subsequent authorisation stage).

When the cardholder form is editable, passing a **true** value for **EKASHU_CARD_NAME_MANDATORY** to eKashu will make the entry of the cardholder's name mandatory and eKashu won't allow the cardholder to proceed.

## OPTIONAL PROPERTIES

Each of these properties is optional and is used to configure behavioural options with the payment page.

| Field Name | Purpose | Example |
|---|---|---|
| **EKASHU_AUTO_CONFIRM** | Whether the transaction should be automatically confirmed and committed for settlement when approved. If this is not enabled, each transaction must be manually checked using WebMIS before the authorisation is committed for settlement. This should only be enabled in situations where there are no physical goods to deliver as payment should not be taken before the items are shipped. | false (default) |
| **EKASHU_DUPLICATE_CHECK** | Whether the eKashu payment page should identify duplicate payment requests and automatically respond to these without payment taking place. This can occur when the cardholder uses their "back" button and tries to re-enter their card details. A duplicate is identified by looking for previous transactions that have occurred with the same amount, currency and reference in a specified period. Valid values are "false", "error" or "resend". If it is set to "error" and a duplicate is identified the cardholder is informed of the error, if it is set to "resend" the payment page replicates the previous result by redirecting the cardholder to the success or failure page. | false (default) |
| **EKASHU_DUPLICATE_MINUTES** | The number of minutes over which the duplicate check takes place. This must be greater than zero. | 60 (default) |
| **EKASHU_HASH_CODE** | A hash code with which eKashu can validate the source of the message. **Important details can be found at the end of this document** along with an example of how to generate the hash code in C# and PHP. | 3nftLyY7Vl kh4fCZGO 88U5h9fAY JSbZX+Py c00Wpulo= |
| **EKASHU_HASH_CODE_FORMAT** | The format of the hash code present in ekashu_hash_code. This should be "base64". | base64 (default) |

| | | |
|---|---|---|
| **EKASHU_CODE_TYPE** | The type of the hash code present in ekashu_hash_code. This should be "SHA1" or "SHA256HMAC". **Important details can be found at the end of this document.** | SHA1 or SHA256HMAC |
| **EKASHU_CODE_VERSION** | The version of the hash code present in ekashu_hash_code. This should be "1.0.0" or "2.0.0". **Important details can be found at the end of this document.** | 1.00 or 2.00 |
| **EKASHU_LOCALE** | The locale that the browser should be forced to use. If this is not specified the locale is determined from the primary language configured in the browser. Currently supported locales include: da_DK, de_DE, en_CA, en_GB, en_US, es_ES, fr_FR, nb_NO, nl_NL, pt_BR and sv_SE. If the language is not supported the locale falls back to en_GB. | |
| **EKASHU_MOTO_PAYMENT** | A boolean value to indicate whether eKashu is being used to perform a MOTO transaction.<br><br>If a value of **true** is provided for this variable then the following things will happen:<br>● eKashu will mark the transaction as being a MOTO transaction resulting in a **Keyed - CNP** transaction<br>● eKashu will bypass the 3D Secure flows | false (default) |
| **EKASHUE_REFERENCE** | A unique seller reference for order tracking. This will be returned as an output property without any modifications and recorded against the transaction by the eKashu system. If the request type is "recurring" the stored reference will have the date and time appended for uniqueness. | 098765432 1A |
| **EKASHU_REQUEST_TYPE** | The type of request to perform. Can either be "auth" which is a full authorisation that can be settled or "preauth" which is an authorisation that is used to confirm that the card details are valid. A preauth cannot be settled and the amount is fixed to a small value by the eKashu platform. | auth (default) |

| | | |
|---|---|---|
| **EKASHU_SELLER_EMAIL_ADDRESS** | The email address of the seller. | seller@example. com |
| **EKASHU_VIEWPORT** | What to set the viewport meta tag to. This is useful if the payment page is to be used on a mobile device, such as a smartphone. | device-width, initial-scale =1.0, maximum-scale=1.0, user-scalable=no |
| **EKASHU_AMOUNT_FORMAT** | The amount format that you wish the amount to be displayed in.<br>The value of this variable takes the form of a 3 character value.<br>The first character: **A** or **S**<br>A will display the alpha code for the currency.<br>S will display the symbol.<br>Default: **S**<br>The second character: **<** or **>** (or not present)<br>Which side of the amount that the symbol or alpha code should be displayed.<br>**<** means display the symbol / alpha code on the left of the amount"<br>**>** means display the symbol / alpha code on the right of the amount<br>Default: **<**<br>The third character: **_** (or not present)<br>The presence of the underscore will control whether or not a space is printed between the symbol / alpha code and the amount.<br>Default: Not present<br>**Note:** You must specify the 2nd character in order to use the 3rd character. | Default:<br>**S<**<br>Example Amount Format:<br>**£1.50**<br>Other examples:<br>**A<_**<br>Resulting Amount Format:<br>GBP 1.50<br>**A**<br>Example Amount Format:<br>**GBP1.50**<br>**A>**<br>Example Amount Format:<br>**1.50GBP** |

## CARDHOLDER PROPERTIES

Each of these properties is used to provide information about the cardholder, if it is known. If supplied, and AVS is required these properties are used as input to the AVS process.

| Field Name | Purpose | Example |
|---|---|---|

| | | |
|---|---|---|
| **EKASHU_CARD_ADDRESS_REQUIRED** | Whether the cardholder's address is required by the seller. If AVS checks are required, this optional value is overridden and the address will always be obtained; however this option should be set if you require that we store the address. | false (default) |
| **EKASHU_CARD_NAME_REQUIRED** | Whether or not the cardholder 's name is required by the seller.<br><br>This option should be set if you require just the cardholder name, and don't require the rest of the address.<br><br>This variable has no impact if:<br><br>EKASHU_CARD_ADDRESS_REQUIRED is set to true<br><br>Or<br><br>EKASHU_CARD_ADDRESS_VERIFY is set to a non "false" value<br><br>Or<br><br>EKASHU_CARD_ZIP_CODE_VERIFY is set to a non "false" value.<br><br>Note: This is particularly useful for 3DS2, where this data is considered mandatory. | false (default) |
| **EKASHU_CARD_NAME_MANDATORY** | Whether or not the cardholder 's name is considered mandatory by eKashu.<br><br>This variable is subject to the same restrictions as the EKASHU_CARD_NAME_REQUIRED variable. | false (default) |
| **EKASHU_CARD_ADDRESS_EDITABLE** | Whether the cardholder's address can be edited by the user. When specified the address will not be validated for completeness. | true (default) |
| **EKASHU_CARD_EMAIL_ADDRESS** | The cardholder's email address. This is used for optional fraud screening and to provide the seller with contact information for the buyer. | card@example. com |
| **EKASHU_CARD_EMAIL_ADDRESS_MANDATORY** | Whether collection of the cardholder's email address is a requirement. | true (default) |
| **EKASHU_CARD_TITLE** | The cardholder's title. This can be specified as "Mr", "Mrs", "Ms" or "Miss". This is used for optional fraud screening and to provide the seller with contact information for the buyer. | Ms |

| | | |
|---|---|---|
| **EKASHU_CARD_TITLE_MANDATORY** | Whether the specification of the cardholder's title is mandatory. | true (default) |
| **EKASHU_CARD_FIRST_NAME** | The cardholder's first name. This is used for optional fraud screening and is used to provide the seller with contact information for the buyer.<br>Note: This is considered mandatory for 3DS2. Without it, 3DS2 will not be performed. | Anne |
| **EKASHU_CARD_LAST_NAME** | The cardholder's last name. This is used for optional fraud screening and to provide the seller with contact information for the buyer.<br><br>Note: This is considered mandatory for 3DS2. Without it, 3DS2 will not be performed. | Other |
| **EKASHU_CARD_ADDRESS_1** | The first line of the cardholder's address. This is used for optional AVS/fraud screening and to provide the seller with contact information for the buyer.<br><br>Note: This field has a length limit of 50 characters. | Any Street |
| **EKASHU_CARD_ADDRESS_2** | The second line of the cardholder's address. This is used for optional AVS/fraud screening and to provide the seller with contact information for the buyer.<br><br>Note: This field has a length limit of 50 characters. | |
| **EKASHU_CARD_CITY** | The town/city of the cardholder's address. This is used for optional AVS/fraud screening and to provide the seller with contact information for the buyer. | Any Town |
| **EKASHU_CARD_STATE** | The county/state of the cardholder's address. This is used for optional AVS/fraud screening and to provide the seller with contact information for the buyer. | |
| **EKASHU_CARD_ZIP_CODE** | The post/zip code of the cardholder's address. This is used for optional AVS/fraud screening and to provide the seller with contact information for the buyer. | AN1 2OTH |
| **EKASHU_CARD_COUNTRY** | The country of the cardholder's address. This is used for optional AVS/fraud screening and to provide the seller with contact information for the buyer. | United Kingdom |

|  |  |  |
|---|---|---|
| **EKASHU_CARD_PHONE_NUMBER_MANDATORY** | Whether collection of the cardholder's telephone number is a requirement. | false (default) |
| **EKASHU_CARD_PHONE_NUMBER_TYPE** | The type of telephone number for the cardholder's address. Can either be "Home", "Work", "Mobile" or "Other". This is used for optional fraud screening and to provide the seller with contact information for the buyer. | home |
| **EKASHU_CARD_PHONE_NUMBER** | The cardholder's telephone number. This is used for optional fraud screening and to provide the seller with contact information for the buyer. | +12 (1234)<br><br>12345678 ext 123 |

## DELIVERY ADDRESS PROPERTIES

Each of these properties is used to provide information about the cardholder, if it is known. If supplied, and AVS is required these properties are used as input to the AVS process.

| Field Name | Purpose | Example |
|---|---|---|
| **EKASHU_DELIVERY_ADDRESS_REQUIRED** | Whether the buyer's delivery address is required by the seller. This facility should be used with care as a fraudulent buyer may use this to send goods to their own address rather than that of the cardholder. | false (default) |
| **EKASHU_DELIVERY_ADDRESS_IS_CARD_ADDRESS** | Whether by default, the delivery address provided is the same as the card address. | true (default) |
| **EKASHU_DELIVERY_ADDRESS_EDITABLE** | Whether the buyer's delivery address can be edited by the user. When specified the address will not be validated for completeness. | true (default) |
| **EKASHU_DELIVERY_EMAIL_ADDRESS_MANDATORY** | The buyer's delivery email address. This is used for optional fraud screening and to provide the seller with contact information for the delivery. | delivery@example.com |
| **EKASHU_DELIVERY_TITLE** | The buyer's title. This can be specified as "Mr", "Mrs", "Ms" or "Miss". This is used for optional fraud screening and to provide the seller with contact information for the delivery. | Ms |
| **EKASHU_DELIVERY_TITLE_MANDATORY** | Whether the specification of the buyer's title is mandatory | true (default) |
| **EKASHU_DELIVERY_FIRST_NAME** | The buyer's first name. This is used for optional fraud screening and to provide the seller with contact information for the delivery. | Anne |
| **EKASHU_DELIVERY_LAST_NAME** | The buyer's last name. This is used for optional fraud screening and to provide the seller with contact information for the delivery. | Other |
| **EKASHU_DELIVERY_ADDRESS_1** | The first line of the buyer's delivery address. This is used for optional fraud screening and to provide the seller with contact information for the delivery.<br><br>**Note:** This field has a length limit of 50 characters. | Any street |
| **EKASHU_DELIVERY_ADDRESS** | The second line of the buyer's delivery address. This is used for | |

| | | |
|---|---|---|
| **_2** | optional fraud screening and to provide the seller with contact information for the delivery.<br><br>Note: This field has a length limit of 50 characters. | |
| **EKASHU_DELIVERY_CITY** | The town/city if the buyer's delivery address. This is used for optional fraud screening and to provide the seller with contact information for the delivery. | Any Town |
| **EKASHU_DELIVERY_STATE** | The state of the buyer's delivery address. This is used for optional fraud screening and to provide the seller with contact information for the delivery. | |
| **EKASHU_DELIVERY_ZIP_CODE** | The zip code of the buyer's delivery address. This is used for optional fraud screening and to provide the seller with contact information for the delivery. | AN1 2OTH |
| **EKASHU_DELIVERY_COUNTRY** | The country of the buyer's delivery address. This is used for optional fraud screening and to provide the seller with contact information for the delivery. | United Kingdom (the default is determined from the buyer's IP address) |
| **EKASHU_DELIVERY_PHONE_NUMBER** | The  buyer's delivery telephone number. This is used for optional fraud screening and to provide the seller with contact information for the delivery. | +12 (1234)<br><br>12345678 ext 123 |
| **EKASHU_DELIVERY_PHONE_NUMBER_MANDATORY** | Whether collection of the delivery telephone number is a requirement. | false (default) |
| **EKASHU_DELIVERY_PHONE_NUMBER_TYPE** | The type of telephone number for the buyer's delivery address. Can either be "Home", "Work", "Mobile" or "Other". This is used for optional fraud screening and to provide the seller with contact information for the delivery. | home |

## INVOICE ADDRESS PROPERTIES

Each of these properties is used to provide information about the invoice address, if it is known.

| Field Name | Purpose | Example |
|---|---|---|
| EKASHU_INVOICE_ADDRESS_REQUIRED | Whether the buyer's invoice address is required by the seller. | false (default) |
| EKASHU_INVOICE_ADDRESS_IS_CARD_ADDRESS | Whether by default, the invoice address provided is the same as the card address. | true (default) |
| EKASHU_INVOICE_ADDRESS_EDITABLE | Whether the buyer's invoice address can be edited by the user. When specified the address will not be validated for completeness. | true (default) |
| EKASHU_INVOICE_EMAIL_ADDRESS | The buyer's invoice email address. This is used for optional fraud screening and to provide the seller with contact information for the invoice. | delivery@example.com |
| EKASHU_INVOICE_EMAIL_ADDRESS_MANDATORY | Whether collection of the invoice email address is a requirement | true (default) |
| EKASHU_INVOICE_TITLE | The buyer's invoice title. This can be specified as "Mr", "Mrs", "Ms" or "Miss". This is used for optional fraud screening and to provide the seller with contact information for the invoice. | Ms |
| EKASHU_INVOICE_TITLE_MANDATORY | Whether the specification of the buyer's invoice title is mandatory | true (default) |
| EEKASHU_INVOICE_FIRST_NAME | The buyer's invoice first name. This is used for optional fraud screening and to provide the seller with contact information for the invoice. | Anne |
| EKASHU_INVOICE_LAST_NAME | The buyer's invoice last name. This is used for optional fraud screening and to provide the seller with contact information for the invoice. | Other |
| EKASHU_INVOICE_ADDRESS_1 | The first line of the buyer's invoice address. This is used for optional fraud screening and to | Any Street |

| | | |
|---|---|---|
| | provide the seller with contact information for the invoice. | |
| **EKASHU_INVOICE_ADDRESS_2** | The second line of the buyer's invoice address. This is used for optional fraud screening and to provide the seller with contact information for the invoice. | |
| **EKASHU_INVOICE_CITY** | The town/city if the buyer's invoice address. This is used for optional fraud screening and to provide the seller with contact information for the invoice. | Any town |
| **EKASHU_INVOICE_STATE** | The state of the buyer's invoice address. This is used for optional fraud screening and to provide the seller with contact information for the invoice. | |
| **EKASHU_INVOICE_ZIP_CODE** | The zip code of the buyer's invoice address. This is used for optional fraud screening and to provide the seller with contact information for the invoice | AN1 2OTH |
| **EKASHU_INVOICE_COUNTRY** | The country of the buyer's invoice address. This is used for optional fraud screening and to provide the seller with contact information for the invoice. | United Kingdom (the default is determined from the buyer's IP address) |
| **EKASHU_INVOICE_PHONE_NUMBER** | The telephone number of the buyer's invoice address. This is used for optional fraud screening and to provide the seller with contact information for the invoice. | +12 (1234)<br><br>12345678 ext 123 |
| **EKASHU_INVOICE_PHONE_NUMBER_MANDATORY** | Whether collection of the invoice telephone number is a requirement | false (default) |
| **EKASHU_INVOICE_PHONE_NUMBER_TYPE** | The type of telephone number for the buyer's invoice address. Can either be "Home", "Work", "Mobile" or "Other". This is used for optional fraud screening and to provide the seller with contact information for the invoice. | Home |

## VERIFICATION PROPERTIES

Each of these properties is used to provide information about the invoice address, if it is known.

| Field Name | Purpose | Example |
|---|---|---|
| **EKASHU_3D_SECURE_VERIFY** | This option should be enabled if the seller requires that the cardholder is verified using 3-D Secure (Verified by Visa or MasterCard SecureCode).<br><br>If the verification does not match, the transaction will not be processed.<br><br>For finer control this can be set to be a mask of:<br>Visa = 1<br>MasterCard = 2<br>Maestro = 4<br>JCB = 8<br>American Express = 16<br>Diners = 32<br>Discover = 64<br><br>For example, to just verify Visa and Maestro cards, set this to be "5".<br><br>Due to card scheme rules Maestro cards will always be authenticated with SecureCode even if the option is disabled.<br><br>This variable can also take the following values:<br>• true_all<br>• true_token<br><br>The boolean-style values for this variable behave as follows:<br><br>**true**<br><br>eKashu will attempt 3DS1 against non-tokenized cards.<br><br>eKashu will attempt 3DS2 against all cards, regardless of whether the transaction is tokenized or not.<br><br>**true_all**<br><br>eKashu will attempt to do 3D Secure regardless of whether the transaction is tokenized or not.<br><br>**true_token**<br><br>eKashu will **only** attempt 3D Secure when the transaction is a tokenized transaction.<br><br>**false** | true (default) |

| | | |
|---|---|---|
| | 3D Secure will not be attempted (with the exception of Maestro) | |
| **EKASHU_CARD_ADDRESS_VERIFY** | This option should be enabled if the seller requires that the cardholder’s address is verified using the Address Verification System (AVS). It can have the value “true”, “false” or “check”. If the address does not match and verification is required (true) the transaction is voided and reported as being declined. If the value is “check” the result of the authorisation will be as returned by the bank and the result of the address check will be visible within WebMIS. | true (default) |
| **EKASHU_CARD_ZIP_CODE_VERIFY** | This option should be enabled if the seller requires that the cardholder’s post/zip code is verified using the Address Verification System (AVS). It can have the value “true”, “false” or “check”. If the post/zip code does not match and verification is required (true) the transaction is voided and reported as being declined. If the value is “check” the result of the authorisation will be as returned by the bank and the result of the post/zip code check will be visible within WebMIS. | true (default) |
| **EKASHU_VERIFICATION_VALUE_VERIFY** | The option should be enabled if the seller requires that the Card Verification Value (CVV) is verified. It can have the value “true”, “false” or “check”. If the verification value does not match and verification is required (true) the transaction is voided and reported as being declined. If the value is “check” the result of the authorisation will be as returned by the bank and the result of the verification value check will be visible within WebMIS. In most cases the bank will automatically decline a transaction where the CVV does not match. | true (default) |

## PRODUCT PROPERTIES

eKashu will accept as input a list of products which the buyer is purchasing. The list of products can be retrieved in WebMIS and then be used as an input to the fraud profiling service. Each product field is optional. As any number of products can be specified, the products must be specified in the form:

ekashu_products[index][field] for example, in HTML:

```
<input type=”hidden” name=”ekashu_products[0][amount]” value=”19.99”/>
<input type=”hidden” name=”ekashu_products[0][name]” value=”Hat”/>
<input type=”hidden” name=”ekashu_products[1][amount]” value=”89.99”/>
<input type=”hidden” name=”ekashu_products[1][name]” value=”Coat”/>
... and so on
```

The product fields are shown below:

| Field Name | Purpose | Example |
|---|---|---|
| **AMOUNT** | The cost of the product. | 1.00 |
| **CATEGORY** | The category of the product. | Category |
| **CURRENCY** | The ISO country code or mnemonic for the currency of the product amount specified. | EUR |
| **CODE** | The product description. | Description |
| **NAME** | The name of the product. | Product |
| **QUANTITY** | The quantity of the product. This must be numeric. | 2 |
| **RISK** | The risk of the product. The valid values are “VeryLow”, “Low”, “Medium”, “High” and “VeryHigh”. | High |
| **TYPE** | The type of the product. | Type |

## BROWSER PROPERTIES

Each of these properties is used to customise the experience for the seller by using the browser. The overall look and feel can be customised as well as more subtle items such as the browser title.

| Field Name | Purpose | Example |
|---|---|---|
| **EKASHU_TITLE** | The web page title text to be used for the eKashu checkout process. | eKashu Checkout (default) |
| **EKASHU_DESCRIPTION** | The description of the goods being purchased. This text will be displayed in the checkout page. | A Personal Computer |
| **EKASHU_SELLER_ADDRESS** | The address of the seller. This text will be displayed on the checkout page. | 1 High Street, City, Country |
| **EKASHU_SELLER_NAME** | The name of the seller. This text will be displayed on the checkout page. | Anne Other Shop |
| **EKASHU_STYLE_SHEET** | This specifies the URL of a CSS file to use for the checkout's style sheet. This file should contain visual settings for all of the classes and ID's present on the checkout pages. The default eKashu CSS file can be used as a guide. The CSS file should be hosted on a HTTPS server in order to avoid browser warnings. | https://example.com/stylesheet.cs s |
| **EKASHU_SHORTCUT_ICON** | The URL of an icon to use as the webpage shortcut icon (also known as a favicon or favourites icon). This will be displayed by compatible browsers. The icon should be hosted on a HTTPS server in order to avoid browser warnings. | https://example.com/favicon.ico |
| **EKASHU_FAILURE_URL** | The URL of a web page that the buyer should be directed to if the checkout process fails 3 times. This URL will be sent a set of POST data. This is described below. | http://example.com/failure.html |
| **EKASHU_FAILURE_RETURN_T EXT** | The URL of a web page that the buyer should be directed to if the checkout process fails 3 times. This URL will be sent a set of POST data. | http://example.com/failure.html |

| | | |
|---|---|---|
| | This is described below. | |
| **EKASHU_FAILURE_RETURN_TEXT** | The text to display as the link that allows the buyer to return to another URL if the checkout fails. Also see: ekashu_return_url. | Return (default) |
| **EKASHU_RETURN_TEXT** | The text to display as the link that allows the buyer to cancel the checkout process and return to another URL. Also see: ekashu_return_url. | Cancel and return (default) |
| **EKASHU_RETURN_URL** | The URL of a web page that a buyer can return to if they decide to cancel the checkout process. The URL will not be sent any POST data. Also see: ekashu_return_text. | http://example.com/shop.html |
| **EKASHU_SUCCESS_URL** | The URL of a web page that the buyer should be directed to if the checkout process succeeds. This URL will be sent a set of POST data. This is described below. | http://example.com/success.html |
| **EKASHU_INCLUDE_POST** | Whether to include POST data in the directions to the success and failure URL. This is useful when it is a requirement to return the status by a call-back rather than via a redirection. | true (default) |
| **EKASHU_CALLBACK_FAILURE_URL** | The URL of a web page the eKashu server should perform a background call to if the checkout process fails 3 times. This is useful when it is a requirement to return the status via a call- back rather than by a redirection. | http://example.com/c_failure.html |
| **EKASHU_CALLBACK_SUCCESS_URL** | The URL of a web page the eKashu server should perform a background call to if the checkout process succeeds. If the URL cannot be reached, the eKashu server will attempt to reach the URL at increasing intervals over 3 days. | http://example.com/c_success.htm l |
| **EKASHU_CALLBACK_INCLUDE_POST** | Whether to include POST data in the call-backs to the success and failure callback URLs. This is useful when it is a requirement to return the status by a call-back rather than by a redirection. | true (default) |

## CREDENTIAL ON FILE PROPERTIES

These properties are optional and can be used to mark a transaction as Credential on File (CoF).

If eKashu receives a value within the “ekashu_cof_initiated_by” property, then the “ekashu_cof_reason” property is considered mandatory and a valid value must be supplied.

Additionally, if the transaction is using stored credentials (tokenised) and the “ekashu_cof_initiated_by” and “ekashu_cof_reason” properties have been supplied, then the “ekashu_cof_id” property is considered mandatory and a valid value must be supplied.

| Field Name | Purpose | Example |
|---|---|---|
| **EKASHU_COF_INITIATED_BY** | Indicates who initiated the CoF transaction. The valid values are “CardHolder, Merchant”. | Merchant |
| **EKASHU_COF_REASON** | Indicates the reason for the CoF transaction. This value is mandatory if “ekashu_cof_initiated_by” has been passed.<br>The valid values are “Empty, Unscheduled, Installment, Recurring, Incremental, Resubmission, DelayedCharge, ReAuth, NoShow” | DelayedCharge |
| **EKASHU_COF_ID** | This value is mandatory performing a tokenised transaction that is being marked as CoF. This value must be the transaction GUID belonging to the original first store transaction. | a099ad60-abc6-41f8-b25b-5b5d0 10e3529 |

## EMV 3DS / 3DS2 INPUT PROPERTIES

These properties are optional and can be used to control various EMV 3DS (3DS2) behaviours within eKashu.

| Field Name | Purpose | Example |
|---|---|---|
| **EKASHU_3D_SECURE_V2_CHALLENGE_INDICATOR** | This eKashu variable allows you to control the EMV 3DS "Challenge Indicator" value used by eKashu during the 3DS2 process.<br><br>This variable can be used to control how the EMV 3DS authentication is flagged with regards to any challenge preferences and allow the Issuer's ACS to determine the best approach to the authentication request.<br><br>Valid Values<br><br>**01** - No Preference<br>**02** - No challenge requested<br>**03** - Challenge requested: 3DS Requestor Preference<br>**04** - Challenge requested: Mandate | 01 |
| **EKASHU_3D_SECURE_V2_REJECT_MISSING_DATA** | This eKashu variable allows you to control the behaviour when the 3DS server responds with incomplete data.<br><br>When set to true any response that would otherwise result in an authentication or attempted response will be rejected if any 3DS data is missing from the response. | false (default) |

## OUTPUT PROPERTIES

The following properties will be sent by the eKashu Checkout process to the success and failure URLs in order for a seller's website to display, store or process the collected information. These properties are returned in addition to those sent by the seller as input properties.

As well as the predefined eKashu fields listed, any number of seller fields could have been specified in the eKashu Checkout form as POST data. These fields will be passed through the eKashu checkout process and returned to the success and failure URLs as POST data. In order to ensure that future enhancements do not disrupt your checkout process, they will not begin with the prefix "ekashu_".

| Field Name | Purpose | Example |
|---|---|---|
| **EKASHU_AUTH_CODE** | The authorisation code received from the acquiring bank for this transaction. | 12C456 |
| **EKASHU_AUTH_RESULT** | The result of the payment authorisation. This can either be "failure" or "success". | failure |
| **EKASHU_CARD_HASH** | The hash of the card used. This can be used in conjunction with CardEaseXML for additional payments | fI9Y+uHNkXyzlkxaCUbc3sUYYOc= |
| **EKASHU_CARD_REFERENCE** | The reference of the card used. This can be used with CardEaseXML for additional payments. | 5757a17e-a1d7-db11 -bc1d-001422187e37 |
| **EKASHU_CARD_SCHEME** | The recognised card scheme that the card number belongs to. The text used for this field could change as receipting requirements are updated by the card schemes and card types change. | VISA |
| **EKASHU_DATE_TIME_LOCAL** | The date and time at which the transaction took place. This is the number of seconds since the Unix Epoch (January 1 1970 00:00:00 GMT). | 1245072170 |
| **EKASHU_DATE_TIME_LOCAL_FMT** | The date and time at which the transaction took place. This is in the time zone local to the seller's terminal. It is in the format: yyyyMMddHHmmss. | 20070101010101 |
| **EKASHU_DATE_TIME_UTC** | The date and time at which the transaction took place. This is the number of seconds since the Unix Epoch (January 1 1970 00:00:00 | 1245072170 |

| | | |
|---|---|---|
| | GMT). | |
| **EKASHU_DATE_TIME_UTC_FMT** | The date and time at which the transaction took place. This is in Universal Coordinated Time. It is in the format: yyyyMMddHHmmss. | 20070101010101 |
| **EKASHU_EXPIRES_END_MONTH** | The two digit month of the card expiry date as entered by the cardholder. | 01 |
| **EKASHU_ISSUE_NUMBER** | The issue number of the card as entered by the cardholder. | 02 |
| **EKASHU_MASKED_CARD_NUMBER** | A masked version of the card number that the cardholder used for the transaction. | XXXXXXX XXXXX12 34 |
| **EKASHU_TRANSACTION_ID** | A unique eKashu identifier that can be used to track this transaction. | 85761ABA-5415- DE1 1-9A1E-000F1F660B 7C |
| **EKASHU_VALID_FROM_MONTH** | The two digit month of the card valid from date as entered by the cardholder. | 10 |
| **EKASHU_VALID_FROM_YEAR** | The four digit year of the card valid from date as entered by the cardholder. | 2006 |
| **EKASHU_HASH_CODE_RESULT** | A hash code with which the calling website can validate the source of the message. Important details can be found at the end of this document along with an example of how to validate the hash code in C# and PHP. | VWqNER55xvoqu +u 7QwvQw GDNCyYXY 7yo7Fc5G2 mUM4A= |
| **EKASHU_HASH_CODE_RESULT_FORMAT** | The format of the hash code present in the ekashu_hash_code_ result. This should be “base64”. | base64 |
| **EKASHU_HASH_CODE_RESULT_TYPE** | The type of the hash code present in ekashu_hash_code_result. This should be “SHA1” or “SHA256HMAC”. Important details can be found at the end of this document. | SHA1 or SHA256HMAC |
| **EKASHU_HASH_CODE_RESULT_VERSION** | The version of the hash code present in ekashu_hash_code_ result. This should be “1.0.0” or “2.0.0”. Important details can be | 1.0.0 or 2.0.0 |

| | | |
|---|---|---|
| | found at the end of this document. | |
| **EKASHU_CARD_ADDRESS_RESULT** | The result of the cardholder's address verification. This can either be "matched", "not_checked", "partial_match" or "not_ matched". | not_checked |
| **EKASHU_CARD_ZIP_CODE_RESULT** | The result of the cardholder's zip code verification. This can either be "matched", "not_checked", "partial_match" or "not_matched". | matched |
| **EKASHU_VERIFICATION_VALUE_ RESULT** | The result of the card verification value verification. This can either be "matched", "not_checked" or "not_matched". | not_matched |
| **EKASHU_3D_SECURE_ENROLLED** | The result of the 3-D Secure enrolment check. The result can be "none", "yes", "no" or "unknown". | yes |
| **EKASHU_3D_SECURE_RESULT** | The result of the 3-D Secure authentication. This can either be "none", "success", "failure", "unknown" or "attempted". | success |
| **EKASHU_3D_SECURE_ECI** | The generated 3-D Secure E-Commerce Indicator. This will be only present in authorised 3-D Secure transactions. | 6 |
| **EKASHU_3D_SECURE_IAV** | The generated 3-D Secure CAVV (Verified by Visa) or AAV (Mastercard SecureCode). This will be only present in authorised 3-D Secure transactions and will be base64 encoded. | AAACAkZQV1EW NV aHOVB XAAAAAAA= |
| **EKASHU_3D_SECURE_XID** | The generated 3-D Secure transaction ID. This will be only present in authorised 3-D Secure transactions and will be base64 encoded. | QWt1dnNjZGF0S Xgz OHVKV3RHMno= |
| **EKASHU_3D_SECURE_V2_ENROLLED** | The result of the 3-D Secure version 2 enrolment check. The result can be "none", "yes", "no" or "unknown". | yes |
| **EKASHU_3D_SECURE_V2_REQUESTOR_TRANSACTION_ID** | The generated 3-D Secure version 2 Requestor Transaction ID. This will be only present in authorised 3-D Secure version 2 transactions and will be a 36 character string in UUID | 07AAFBF3-A368-4F7 4-94CA-50A969645E 18 |

| | | |
|---|---|---|
| | format. | |
| **EKASHU_3D_SECURE_V2_SERVER_TRANSACTION_ID** | The generated 3-D Secure version 2 Server Transaction ID. This will be only present in authorised 3-D Secure version 2 transactions and will be a 36 character string in UUID format. | 707377EF-DB01-403 5-B1F8-1D8E95395B 33 |
| **EKASHU_3D_SECURE_V2_ACS_TRANSACTION_ID** | The generated 3-D Secure version 2 ACS Transaction ID. This will be only present in authorised 3-D Secure version 2 transactions and will be a 36 character string in UUID format. | 24DB2C62-64CA-4F 13-8B4E-B4851476E 22A |
| **EKASHU_3D_SECURE_V2_DIRECTORY_SERVER_TRANSACTION_ID** | The generated 3-D Secure version 2 Directory Server Transaction ID. This will be only present in authorised 3-D Secure version 2 transactions and will be a 36 character string in UUID format. | 53F75BBE-B094-49A 0-AD9B-84D59E01A 51B |
| **EKASHU_3D_SECURE_V2_RESULT** | The result of the 3-D Secure version 2 authentication. This can either be “none”, “success”, “failure”, “unknown” or “attempted” | success |
| **EKASHU_3D_SECURE_V2_ECI** | The generated 3-D Secure E-Commerce Indicator. This will be only present in authorised 3-D Secure version 2 transactions. | 6 |
| **EKASHU_3D_SECURE_V2_IAV** | The generated 3-D Secure CAVV (Visa Secure) or AAV (Mastercard Identity Check). This will be only present in authorised 3-D version 2 Secure transactions and will be base64 encoded. | AAACAkZQV1E WNV aHOVBXA AAAAAA= |
| **EKASHU_3D_SECURE_VERSION** | The 3-D Secure version used to process this transaction | 2.1.0 |
| **EKASHU_PAYMENT_ACCOUNT_REFERENCE** | The Payment Account Reference associated with the cardholder’s PAN.<br>This value is a unique identifier associated with a specific cardholder PAN.<br>This is an alphanumeric string with a length of 29 characters. | ABC10011001100110011001123456 |

## TECHNICAL SUPPORT

eKashu provides business-hours technical support to web developers integrating the eKashu Payment Page with a customer's website. Email to support@ekashu.com and we will endeavour to assist within the shortest possible time.

## HASH CODE INPUT GENERATION

eKashu currently supports two versions of hash code that can be used for validation of the messages being sent to eKashu and the validation of the responses returned.

PLEASE NOTE: In early 2021, eKashu will be upgraded to allow for the automatic migration of accounts from version 1.0.0 to version 2.0.0 of the hash code. The automatic upgrade will be triggered when an account that is currently set up to use a version 1.0.0 hash code provides valid version 2.0.0 hash code eKashu variables.

Once an account has been successfully upgraded, the account will be forbidden from using a version 1.0.0 hash code for any future transactions, meaning that any subsequent requests to eKashu must supply a valid version 2.0.0 hash code.

If you wish to integrate with the version 2.0.0 hash code before eKashu has been upgraded with automatic migration support, please contact support@ekashu.com with the Seller IDs of the accounts you wish to have version 2.0.0 hash code set up against.

IMPORTANT DATES: Test eKashu (test.ekashu.com)

**7th October 2020**: Any new accounts on the test platform will be boarded to version 2.0.0 of the hash code. These accounts will not be able to perform transactions with a version 1.0.0 hash code at all.

**6th April 2021**: Test eKashu will be updated to prevent any accounts without a hash code or accounts with a version 1.0.0 hash code from being able to process transactions using test eKashu. All existing accounts must use 2.0.0 after this date.

IMPORTANT DATES: Live eKashu (live.ekashu.com)

**6th April 2021**: Any new accounts on the live platform will be boarded to version 2.0.0 of the hash code. These accounts will not be able to perform transactions with a version 1.0.0 hash code at all.

**6th December 2021**: Live eKashu will be updated to prevent any accounts without a hash code or accounts with a version 1.0.0 hash code from being able to process transactions using live eKashu. All existing accounts must use 2.0.0 after this date.

The version 1.0.0 hash code to be sent to eKashu is constructed from the base64 encoded SHA1 hash of: hash_key + ekashu_seller_id + ekashu_reference + ekashu_amount. By default, this functionality is not enabled, however it is highly recommended that a hash key is requested from eKashu Support. Once the hash key has been assigned ekashu_hash_code will have to be populated. Examples are provided below.

The version 2.0.0 hash code to be sent to eKashu is constructed from the base64 encoded SHA256HMAC hash of a number of input properties in alphabetical order with a delimiter of '&'. These properties are:

- ekashu_3d_secure_verify
- ekashu_amount
- ekashu_amount_format
- ekashu_auto_confirm
- ekashu_callback_failure_url
- ekashu_callback_include_post
- ekashu_callback_success_url
- ekashu_card_address_editable
- ekashu_card_address_required
- ekashu_card_address_verify
- ekashu_card_email_address_mandatory
- ekashu_card_phone_number_mandatory
- ekashu_card_title_mandatory
- ekashu_card_zip_code_verify
- ekashu_currency
- ekashu_delivery_address_editable
- ekashu_delivery_address_required
- ekashu_delivery_email_address_mandatory
- ekashu_delivery_phone_number_mandatory
- ekashu_delivery_title_mandatory
- ekashu_description
- ekashu_device
- ekashu_duplicate_check
- ekashu_duplicate_minutes
- ekashu_failure_return_text
- ekashu_failure_url
- ekashu_hash_code_format
- ekashu_hash_code_type
- ekashu_hash_code_version
- ekashu_include_post
- ekashu_invoice_address_editable
- ekashu_invoice_address_required
- ekashu_invoice_email_address_mandatory
- ekashu_invoice_phone_number_mandatory
- ekashu_invoice_title_mandatory
- ekashu_locale
- ekashu_payment_methods
- ekashu_reference
- ekashu_request_type
- ekashu_return_text
- ekashu_seller_address
- ekashu_seller_email_address
- ekashu_seller_id
- ekashu_seller_key
- ekashu_seller_name
- ekashu_shortcut_icon
- ekashu_style_sheet
- ekashu_success_url
- ekashu_title
- ekashu_verification_value_mask
- ekashu_verification_value_verify
- ekashu_viewport

Examples are provided below.

## HASH CODE INPUT GENERATION IN C#

Version 1.0.0. The following snippet of C# code demonstrates how to generate the ekashu_ hash_code based upon some sample data. The same process can be performed in other languages that support SHA1 and Base64 encoding.

```
using System;
using System.Security.Cryptography;
using System.Text;

public class Program
{
    public static void Main()
    {
            string hashKey = “trVxrnoz22bvwvnV”;
            string terminalId = “99999999”;
            string reference = “0000000765”;
            string amount = “1.23”;

            // 7PtU022473m+ntcZY2wt6pXzKWc=
            Console.WriteLine(
                    Convert.ToBase64String(
                            new SHA1CryptoServiceProvider().ComputeHash(
                                    Encoding.UTF8.GetBytes(
                                            string.Concat(hashKey, terminalId, reference, amount)))));

    }

}
```

Version 2.0.0. The following snippet of C# code demonstrates how to generate the ekashu_hash_code based upon some sample data. The same process can be performed in other languages that support SHA256HMAC and Base64 encoding.

```
using System;
using System.Collections.Generic;
using System.Security.Cryptography;
using System.Text;

public class Program
{
    public static void Main()
    {
            string hashKey = “trVxrnoz22bvwvnV”;

            SortedDictionary<string, string> hashcodeInput = new  SortedDictionary<string, string>
            {
                    { “ekashu_3d_secure_verify”, null },
                    { “ekashu_amount”, “1.23” },
                    { “ekashu_amount_format”, null },
                    { “ekashu_auto_confirm”, null },
                    { “ekashu_callback_failure_url”, null },
                    { “ekashu_callback_include_post”, null },
                    { “ekashu_callback_success_url”, null },
                    { “ekashu_card_address_editable”, null },
                    { “ekashu_card_address_required”, null },
                    { “ekashu_card_address_verify”, null },
                    { “ekashu_card_email_address_mandatory”, null },
                    { “ekashu_card_phone_number_mandatory”, null },
                    { “ekashu_card_title_mandatory”, null },
                    { “ekashu_card_zip_code_verify”, null },
                    { “ekashu_currency”, “GBP” },
                    { “ekashu_delivery_address_editable”, null },
                    { “ekashu_delivery_address_required”, null },
                    { “ekashu_delivery_email_address_mandatory”, null },
                    { “ekashu_delivery_phone_number_mandatory”, null },
                    { “ekashu_delivery_title_mandatory”, null }
                    { “ekashu_description”, null },
                    { “ekashu_device”, null },
                    { “ekashu_duplicate_check”, null },
                    { “ekashu_duplicate_minutes”, null },
                    { “ekashu_failure_return_text”, null },
                    { “ekashu_failure_url”, null },
                    { “ekashu_hash_code_format”, null },
                    { “ekashu_hash_code_type”, “SHA256HMAC” },
                    { “ekashu_hash_code_version”, “2.0.0” },
                    { “ekashu_include_post”, null },
                    { “ekashu_invoice_address_editable”, null },
                    { “ekashu_invoice_address_required”, null },
                    { “ekashu_invoice_email_address_mandatory”, null },
                    { “ekashu_invoice_phone_number_mandatory”, null },
                    { “ekashu_invoice_title_mandatory”, null },
                    { “ekashu_locale”, null },
                    { “ekashu_payment_methods”, null },
                    { “ekashu_reference”, “0000000765” },
                    { “ekashu_request_type”, null },
                    { “ekashu_return_text”, null },
                    { “ekashu_seller_address”, null },
                    { “ekashu_seller_email_address”, null },
                    { “ekashu_seller_id”, “99999999” },
                    { “ekashu_seller_key”, “klhy2V81” },
                    { “ekashu_seller_name”, null },
                    { “ekashu_shortcut_icon”, null },
                    { “ekashu_style_sheet”, null },
                    { “ekashu_success_url”, null },
                    { “ekashu_title”, null },
                    { “ekashu_verification_value_mask”, null },
                    { “ekashu_verification_value_verify”, null },
                    { “ekashu_viewport”, null }
            };


            byte[] hashcodeInputBytes = Encoding.UTF8.GetBytes(string.Join(“&”,
hashcodeInput.Values));


            using (HMACSHA256 hmac = new HMACSHA256(Encoding.UTF8.GetBytes(hashKey)))
            {
                    byte[] hash = hmac.ComputeHash(hashcodeInputBytes);

                     // ht3mNZVxrX+jUlN6C99ufuf3g/gfGY5JmXwCBi7nbq0=
                    Console.WriteLine(Convert.ToBase64String(hash));
            }
    }

}
```

## HASH CODE INPUT GENERATION IN PHP

Version 1.0.0. The following snippet of PHP code demonstrates how to generate the ekashu_hash_code based upon some sample data. The same process can be performed in other languages that support SHA1 and Base64 encoding.

```
<?php
        $hash_key = ‘trVxrnoz22bvwvnV’;
        $terminal_id = ‘99999999’;
        $reference = ‘0000000765’;
        $amount = ‘1.23’;

        // 7PtU022473m+ntcZY2wt6pXzKWc=
        echo base64_encode(pack(‘H*’, sha1($hash_key . $terminal_id .
    $reference . $amount))) . ”\n”;
?>
```

Version 2.0.0. The following snippet of PHP code demonstrates how to generate the ekashu_hash_code based upon some sample data. The same process can be performed in other languages that support SHA256HMAC and Base64 encoding.

```
<?php
        $hash_key = ‘trVxrnoz22bvwvnV’;

        $hashcode_input[‘ekashu_3d_secure_verify’] = null;
        $hashcode_input[‘ekashu_amount’] = ‘1.23’;
        $hashcode_input[‘ekashu_amount_format’] = null;
        $hashcode_input[‘ekashu_auto_confirm’] = null;
        $hashcode_input[‘ekashu_callback_failure_url’] = null;
        $hashcode_input[‘ekashu_callback_include_post’] = null;
        $hashcode_input[‘ekashu_callback_success_url’] = null;
        $hashcode_input[‘ekashu_card_address_editable’] = null;
        $hashcode_input[‘ekashu_card_address_required’] = null;
        $hashcode_input[‘ekashu_card_address_verify’] = null;
        $hashcode_input[‘ekashu_card_email_address_mandatory’] = null;
        $hashcode_input[‘ekashu_card_phone_number_mandatory’] = null;
        $hashcode_input[‘ekashu_card_title_mandatory’] = null;
        $hashcode_input[‘ekashu_card_zip_code_verify’] = null;
        $hashcode_input[‘ekashu_currency’] = ‘GBP’;
        $hashcode_input[‘ekashu_delivery_address_editable’] = null;
        $hashcode_input[‘ekashu_delivery_address_required’] = null;
        $hashcode_input[‘ekashu_delivery_email_address_mandatory’] = null;
        $hashcode_input[‘ekashu_delivery_phone_number_mandatory’] = null;
        $hashcode_input[‘ekashu_delivery_title_mandatory’] = null;
        $hashcode_input[‘ekashu_description’] = null;
        $hashcode_input[‘ekashu_device’] = null;
        $hashcode_input[‘ekashu_duplicate_check’] = null;
        $hashcode_input[‘ekashu_duplicate_minutes’] = null;
        $hashcode_input[‘ekashu_failure_return_text’] = null;
        $hashcode_input[‘ekashu_failure_url’] = null;
        $hashcode_input[‘ekashu_hash_code_format’] = null;
        $hashcode_input[‘ekashu_hash_code_type’] = ‘SHA256HMAC’;
        $hashcode_input[‘ekashu_hash_code_version’] = ‘2.0.0’;
        $hashcode_input[‘ekashu_include_post’] = null;
        $hashcode_input[‘ekashu_invoice_address_editable’] = null;
        $hashcode_input[‘ekashu_invoice_address_required’] = null;
        $hashcode_input[‘ekashu_invoice_email_address_mandatory’] = null ;
        $hashcode_input['ekashu_invoice_phone_number_mandatory'] = null ;
        $hashcode_input['ekashu_invoice_title_mandatory'] = null;
        $hashcode_input['ekashu_locale'] = null;
        $hashcode_input['ekashu_payment_methods'] = null;
        $hashcode_input['ekashu_reference'] = '0000000765';
        $hashcode_input['ekashu_request_type'] = null;
        $hashcode_input['ekashu_return_text'] = null;
        $hashcode_input['ekashu_seller_address'] = null;
        $hashcode_input['ekashu_seller_id'] = '99999999';
        $hashcode_input['ekashu_seller_email_address'] = null;
        $hashcode_input['ekashu_seller_key'] = 'kIhy2V81';
        $hashcode_input['ekashu_seller_name'] = null;
        $hashcode_input['ekashu_shortcut_icon'] = null;
        $hashcode_input['ekashu_style_sheet'] = null;
        $hashcode_input['ekashu_success_url'] = null;
        $hashcode_input['ekashu_title'] = null;
        $hashcode_input['ekashu_verification_value_mask'] = null;
        $hashcode_input['ekashu_verification_value_verify'] = null;
        $hashcode_input['ekashu_viewport'] = null; ksort($hashcode_input);

        $hash_code = null;

        foreach ($hashcode_input as $name => $value)
        {
                $hash_code .= $value;
                $hash_code .= '&';
        }

        $hash_code = substr($hash_code, 0, -1);

        // ht3mNZVxrX+jUlN6C99ufuf3g/gfGY5JmXwCBi7nbq0=
        echo base64_encode(hash_hmac('sha256', $hash_code, $hash_key, true)) . "\n";

?>
```

## HASH CODE RESULT VALIDATION

eKashu currently supports two versions of hash code that can be used for validation of the messages being sent to eKashu and the validation of the responses returned.

**PLEASE NOTE:** in early 2021, ekashu will be upgraded to allow for the automatic migration of accounts from version 1.0.0 To version 2.0.0 Of the hash code.

The automatic upgrade will be triggered when an account that is currently set up to use a version 1.0.0 Hash code provides valid version 2.0.0 Hash code ekashu variables.

Once an account has been successfully upgraded, the account will be forbidden from using a version 1.0.0 Hash code for any future transactions, meaning that any subsequent requests to ekashu must supply a valid version 2.0.0 Hash code.

If you wish to integrate with the version 2.0.0 Hash code before ekashu has been upgraded with automatic migration support, please contact support@ekashu.Com with the seller ids of the accounts you wish to have version 2.0.0 Hash code set up against.

**IMPORTANT DATES:** test ekashu (test.Ekashu.Com)

**7th october 2020:** any new accounts on the test platform will be boarded to version 2.0.0 Of the hash code. These accounts will not be able to perform transactions with a version 1.0.0 Hash code at all.

**6th april 2021**: test ekashu will be updated to prevent any accounts without a hash code or accounts with a version 1.0.0 Hash code from being able to process transactions using test ekashu. All existing accounts must use 2.0.0 After this date.

**IMPORTANT DATES:** live ekashu (live.Ekashu.Com)

**6th april 2021:** any new accounts on the live platform will be boarded to version 2.0.0 Of the hash code. These accounts will not be able to perform transactions with a version 1.0.0 Hash code at all. This note continues on the next page.

**6th December 2021**: live ekashu will be updated to prevent any accounts without a hash code or accounts with a version 1.0.0 Hash code from being able to process transactions using live ekashu. All existing accounts must use 2.0.0 After this date.

The version 1.0.0 hash code returned from eKashu is constructed from the base64 encoded SHA1 hash of: hash_key + ekashu_seller_id + ekashu_transaction_id + auth_result. Where "auth_result" is 0 for success or 1 for failure. By default, this functionality is not enabled, however it is highly recommended that a hash key is requested from eKashu Support. Once the hash key has been assigned ekashu_ hash_code_result should be checked to ensure that the parameters received result should be checked to ensure that the parameters received result in the value expected. Examples are provided below.

The version 2.0.0 hash code returned from eKashu is constructed from the base64 encoded SHA256HMAC hash of a number of parameters with delimiters of '&'. Due to the large number of parameters an endpoint is provided for an integrator to check that the hash code result matches what eKashu would generate from the parameters without having to calculate it themselves.

Test transactions:

*https://test.ekashu.com/validate_hash_code.php*

Live transactions:

*https://live.ekashu.com/validate_hash_code.php*

If the validation succeeds a HTTP Status Code of 200 will be returned. Examples are provided below.

## HASH CODE VALIDATION IN C#

Version 1.0.0. The following snippet of C# code demonstrates how to validate the ekashu_hash_code_result based upon returned data. The same process can be performed in other languages that support SHA1 and Base64 encoding. The Version 2.0.0 validation process can also be used to validate Version 1.0.0 ekashu_hash_ code_result if required.

```
using System;
using System.Collections.Specialized;
using System.Security.Cryptography;
using System.Text;
using System.Web;

public class HashCodeResultValidator
{
        public void CheckHashCode(HttpContext httpContext)
        {
                string hashKey = “trVxrnoz22bvwvnV”;
                NameValueCollection eKashuData = httpContext.Request.Form;
                string terminalId = eKashuData[“ekashu_seller_id”];
                string reference = eKashuData[“ekashu_transaction_id”];
                string authResult = eKashuData[“ekashu_auth_result”] == “success” ? “0” : “1”;
                string hashResult = Convert.ToBase64String(
                        new SHA1CryptoServiceProvider().ComputeHash(
                                Encoding.UTF8.GetBytes(
                                        string.Concat(hashKey, terminalId, reference, authResult))));

                if (hashResult != eKashuData[“ekashu_hash_code_result”])
                {
                        // Code to handle validation failure
                }
        }
}
```

Version 2.0.0. The following snippet of C# code demonstrates how to validate the ekashu_hash_code_result based upon returned data. The same process can be performed in other languages that support SHA256HMAC and Base64 encoding. The Version 2.0.0 validation process can also be used to validate Version 1.0.0 ekashu_hash_code_result if required.

```
using System.Collections.Specialized;
using System.IO;
using System.Net
using System.Text;
using System.Web;
public class HashCodeResultValidator
{
        public void CheckHashCode(HttpContext httpContext)
        {
                NameValueCollection eKashuData = httpContext.Request.Form;
                StringBuilder requestContent = new StringBuilder();

                foreach (string key in eKashuData)
                {
                        requestContent.Append(HttpUtility.UrlEncode(key) + “=” +
HttpUtility.UrlEncode(eKashuData[key]));
                        requestContent.Append(“&”);
                }

                requestContent.Length--;
```

```
                    byte[] requestBuffer =Encoding.UTF8.GetBytes(requestContent.ToString());


                    HttpWebRequest webRequest =
(HttpWebRequest)WebRequest.Create(@"https://test.ekashu.com/validate_hash_code.php");

                    webRequest.ContentLength = requestBuffer.Length;
                    webRequest.ContentType = "application/x-www-form-urlencoded";
                    webRequest.Method = "POST";

                    using (Stream requestStream = webRequest.GetRequestStream())
                    {
                              requestStream.Write(requestBuffer, 0, requestBuffer.Length);
                    }

                    HttpWebResponse result = (HttpWebResponse)webRequest.GetResponse();
                    result.Close();

                    if (result.StatusCode != HttpStatusCode.OK)
                    {
                              // Code to handle validation failure
                    }
          }
}
```

## HASH CODE VALIDATION IN PHP

Version 1.0.0.
The following snippet of PHP code demonstrates how to validate the ekashu_hash_code_result based upon returned data. The same process can be performed in other languages that support SHA1 and Base64 encoding. The Version 2.0.0 validation process can also be used to validate Version 1.0.0 ekashu_hash_ code_result if required.

```
<?php
          $hash_key='trVxrnoz22bvwvnV';

          if (base64_encode(pack('H*', sha1(
             $hash_key
             $_POST['ekashu_seller_id'].
             $_POST['ekashu_transaction_id'].
             ($_POST['ekashu_auth_result'] == 'success' ? 0 : 1)))) != $_POST['ekashu_hash_code_result'])
          {
                    // Code to handle validation failure
                    // ...
          }
?>
```

Version 2.0.0.
The following snippet of PHP code demonstrates how to validate the ekashu_hash_code_result based upon returned data. The same process can be performed in other languages that support SHA256HMAC and Base64 encoding. The Version 2.0.0 validation process can also be used to validate Version 1.0.0 ekashu_hash_code_result if required.

```
<?php

$post_data = null;

foreach ($_POST as $name => $value)
{
```

```
            if (preg_match('/^ekashu_/', $name) == 1)
            {
                        $post_data .= urlencode($name).'='.urlencode($value);
                        $post_data .= '&';
            }
}

$post_data = substr($post_data, 0, -1);

$curl = curl_init();

curl_setopt($curl,CURLOPT_URL, 'https://test.ekashu.com/validate_hash_code.php');
curl_setopt($curl,CURLOPT_POST, true);
curl_setopt($curl, CURLOPT_POSTFIELDS, $post_data);
curl_setopt($curl, CURLOPT_HEADER, true);
curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);

curl_exec($curl);

$http_code = curl_getinfo($curl, CURLINFO_HTTP_CODE); curl_close($curl);

if ($http_code != 200)
{
            // Code to handle validation failure
            // ...
}

?>
```

## NMI BRANDING

The NMI brand is respected as being synonymous with security and reliability. You may wish to include the NMI logo on your website using the following HTML:

```
<a href="https://www.nmi.com">
            <img alt="Secure Payment by NMI"
                  border="0"
                  src="https://www.nmi.com/assets/uploads/site-uploads/nmi-badge.png" />
</a>
```

## TEST CARDS

### EMV 3DS 2.x.x

Below are the details of the various test card numbers that can be used to test the different flows and results that can come from an EMV 3DS 2.x authentication.

### Expiry Date / Cardholder Name

The expiry date and cardholder name for all these cards is always the same. These must be used otherwise the enrolment request will fail.

**Please note the different CVV for American Express cards.**

| Expiry Date | Cardholder Name | CVV | CVV (Amex) |
|---|---|---|---|
| 08/2025 | Test Card | 123 | 1234 |

## Successful Frictionless Flow Authentication

| **Scheme** | **Card Number** |
|---|---|
| **AMEX** | 340000000000108 |
| **DISCOVER/DINERS** | 6440000000000104 |
| **DISCOVER/DINERS** | 36000000000008 |
| **MASTERCARD** | 5100000000000107 |
| **VISA** | 4100000000000100 |

3DS authentication will complete successfully without any challenge from the ACS.

ARes Result:

- Transaction Status = Y
- ECI = 05, or 02 (for Mastercard)
- Contains an Authentication Value

## Successful Challenge Flow Authentication

| Scheme | Card Number |
|---|---|
| AMEX | 340000000005008 |
| DISCOVER/DINERS | 6440000000005004 |
| DISCOVER/DINERS | 36000000005007 |
| MASTERCARD | 5100000000005007 |
| VISA | 4100000000005000 |

These card numbers will trigger a step up to a challenge and request the user enter a password.

The password **123456** should be used in order to successfully authenticate the user.

ARes Result:

- Transaction Status = C

RReq Result:

- Transaction Status = Y
- ECI = 05, or 02 (for Mastercard)
- Contains an Authentication Value

## Authentication Attempted

| Scheme | Card Number |
|---|---|
| AMEX | 340000000100007 |
| DISCOVER/DINERS | 6440000000100003 |
| DISCOVER/DINERS | 36000000100006 |
| MASTERCARD | 5100000000100006 |
| VISA | 4100000000100009 |

This transaction will attempt to perform authentication before returning an “Attempted” response.

ARes result:

- Transaction Status = A
- ECI = 06 or 01 (Mastercard)
- Contains an Authentication Value

## Authentication Failed

| Scheme | Card Number |
|---|---|
| AMEX | 340000000300003 |
| DISCOVER/DINERS | 6440000000300009 |
| DISCOVER/DINERS | 36000000300002 |
| MASTERCARD | 5100000000300002 |
| VISA | 4100000000300005 |

With these test PANs, a Challenge step up will be initiated and the user asked for a password.

The password **111111** should be used in order to successfully trigger the failure case.

ARes Result:

- Transaction Status = C

RReq Result:

- Transaction Status = N
- ECI = 00
- Does not contain an Authentication Value

## Authentication Unavailable

| Scheme | Card Number |
|---|---|
| **AMEX** | 340000000400001 |
| **DISCOVER/DINERS** | 6440000000400007 |
| **DISCOVER/DINERS** | 36000000400000 |
| **MASTERCARD** | 5100000000400000 |
| **VISA** | 4100000000400003 |

Transaction will end with the authentication being rejected by the ACS.

ARes result:

- Transaction Status = U
- ECI is dependant on card scheme specific rules
- Does not contain an Authentication Value

## Authentication Rejected

| Scheme | Card Number |
|---|---|
| AMEX | 340000000500008 |
| DISCOVER/DINERS | 6440000000500004 |
| DISCOVER/DINERS | 36000000500007 |
| MASTERCARD | 5100000000500007 |
| VISA | 4100000000500000 |

Transaction will end with the authentication being rejected by the ACS.

ARes result:

- Transaction Status = R
- ECI is dependant on card scheme specific rules
- Does not contain an Authentication Value

## About NMI

NMI is a leading global full commerce enablement platform, processing more than $203 billion in payments annually. We enable payments for over 3,900 partners and over 280,000 merchants around the world and across the entire commerce ecosystem: online, in-app, mobile, in-store, unattended and whatever's next.

We're constantly innovating in order to power the next era of payments, building in the latest technology so ISVs, ISOs, Banks and Fintech Innovators can focus on what they do best. NMI has offices in the US and UK and serves global customers.

To find out more how full commerce enablement can create value for you, visit www.nmi.com

**US:** (847) 352 4850 or (800) 617 4850 (toll free), use ext. 1 for support

**UK:** +44 117 930 4455 | hello@nmi.com