KEKS – SELL request specification **DRAFT**

For WEBSHOP, draft 0.51, January 2021

Contents

Introduction

This document applies to KEKS retail transactions in a situation when external merchant application (WEBSHOP) prefers to generate sell request information and forward it to KEKS.

There is a minimal set of information that WEBSHOP must include in sell request in order for it to be paid by KEKS. This set of information is described in chapter **SELL** specifications. It is the information that KEKS app will read or that the KEKS server will expect.

QR code is displayed

A dynamic QR code is used (one that includes info on actual invoice and amount). When a transaction is completed by KEKS, the result is returned from KEKS server to WEBSHOP server. The return information is specified in **ADVICE** chapter. Advice will contain enough information so that WEBSHOP may uniquely identify a single transaction – this is the information originally supplied by WEBSHOP in SELL request. WEBSHOP shall have a Web service capable of receiving Advices.

Alternatively to QR code, the same information can be exchanged via deep-link (when browsing on mobile devices). Info on this can be found in SELL specification.

Optionally, Advice may contain information about transaction that is not approved by issuer of customer's account (insufficient funds, limit exceed and other reasons). This type of Advice is sent to WEBSHOP for information purposes only.

WEBSHOP may reject Advice if billing information is not correct (duplicate bill processing, non-existing point of sale, non-existing bill id, etc.). In such a case KEKS will generate reversal to customer account.

All exchange between KEKS and WEBSHOP servers is done through HTTP POST with JSON content. The values of String fields must be URL-encoded in order to support non-ASCII characters. KEKS determines the URL of WEBSHOP server based on identification. Exchange of information between WEBSHOP and KEKS server is secured through SSL (https). Additional security such as client certificate or VPN may be implemented in system, but is outside of the scope of this document. Currently supported in KEKS application is client certificate embedding in all outgoing requests and client certificate verification for all incoming requests.

Base URL for all API calls is **https://dttlinuxdev.erstebank.hr/eretailer** (test)

SELL specification

API endpoint : n/a (delivered to KEKS backend via mobile client)

SELL should be a text that is formatted as JSON string.

JSON content should have the following fields

Key	Data type	Description
qr_type	Integer	Required for KEKS client app. Numeric code from table delivered by KEKS
cid	String	Unique id for WEBSHOP within KEKS system. Will be given in advance by KEKS
tid	String	Unique id for point of service within WEBSHOP system. Will be returned to vendor server when transaction completes.
bill_id	String	Unique ID of the transaction in WEBSHOP system. May be combined by WEBSHOP with tid to get unique value. Will be returned to WEBSHOP server when transaction completes.
amount	Number	Amount to be charged to customer.
success_url	String	This redirect-url should be part of the deep-link, not needed in QR code. *(KEKS will open the URL after successful transaction in native browser set in the smart phone!)*
fail_url	String	This redirect-url should be part of the deep-link, not needed in QR code.

JSON may contain additional info that vendor may use for other purposes, but will be ignored by KEKS.

Example of content

```
{
“qr_type” : 1,
“cid” : “95522456”,
“tid” : “123456666”,
“bill_id” : “HGHGHG121222”,
“amount”: 123.45 ,
 “success_url”:”https://uzishop.hr/index.php?controller=history”,
“fail_url”:”https://uzishop.hr/index.php?controller=order&step=3”,
}
```

Example of URL content:

https://KEKSpay.hr/pay?cid=95522456&tid=123456666 &bill_id=HGHGHG121222&amount=123.45

ADVICE

API endpoint : to be delivered by WEBSHOP

Advice is sent to WEBSHOP with following fields.

Key	Data type	Description
bill_id	String	Unique id for application generating the QR. Returned from KEKS server
keks_id	String	Unique id of Keks Pay transaction. Returned from KEKS server
tid	String	Returned 'tid' from QR value
store	String	Description of the store (will be displayed to customer on wallet). May be any description as defined by WEBSHOP.
amount	Numeric	Original amount from SELL request
status	Integer	Numeric code of the result. Only value 0 means that transaction was authorized.
message	String	Text message

Example of Advice content

```
{
 "bill_id" : "HGHGHG121222",
"keks_id" : "8547254",
"tid" : "123456666",
"store" :" WEBSHOP",
"amount": 123.45 ,
 "status" : 0,
 "message" : "Paid" ,
}
```

Advice response from WEBSHOP shall have the following fields.

Key	Data type	Description
status	Integer	Numeric code of the result. Only value 0 means that transaction was successfully processed by WEBSHOP.
message	String	Text message

Example of Advice response

```
{
  "status" : 0,
   "message" : "Accepted" ,
}
```

HASH

‘**hash’** field must be added to any request JSON data coming from WEBSHOP

Hash is calculated by :

1.) Forming UTF-8 String concatenation (epochtime + tid + amount + bill_id)
2.) Calculating MD5 value from the resulting string
3.) Encryping the MD5 value **(in bytes)** with the secret key

tid must always be present in data that is hashed. Other values are optional.

Default values are used in queries that do not include certain fields that are included in hashing.

Default values for hashing are

epochtime ’0’

amount ‘0’

bill_id ‘ ’ (one blank)

<u>Example of hash calculation in JS</u>

```
function calculateHash(tid, epochtime, amount, billId, deskey) {
   let hashInput = new String(epochtime).concat(tid, amount, billId);
   let md5Value = enc.Hex.stringify(MD5(hashInput)).toUpperCase();
   let desEncrypt = TripleDES.encrypt(enc.Hex.parse(md5Value), enc.Utf8.parse(deskey), {
      iv: enc.Hex.parse("0000000000000000"),
      mode: mode.CBC,
      padding: pad.Pkcs7
   });
   let hash = enc.Hex.stringify(desEncrypt.ciphertext).toUpperCase();
   return hash;
}
```

Refunds

API endpoint : /keksrefund

WEBSHOP POS will be able to trigger partial and full refunds.

Full refund – a transaction for which 100% of original amount is returned to KEKS user

Partial refund – a transaction for which only a part of original amount is returned to KEKS user

KEKS will expose service to trigger such refunds.

Prerequisite for this service is a successful transaction.

A refund is sent from WEBSHOP with following fields.

Key	**Data type**	**Description**
bill_id	String	Unique id for application generating the QR
keks_id (for qr_type=1) or transaction_id (for qr_type=3)	String	Unique id of Keks Pay transaction. Returned from KEKS server
tid	String	Unique ID of POS
amount	Numeric	Amount to be refunded. Optional, if omitted, the total original amount is refunded

*currency assumed from original transaction

Register response from KEKS to WEBSHOP shall have the following fields.

Key	**Data type**	**Description**
bill_id	String	Unique id for application generating the QR
keks_refund_id	String	Unique id of KEKS Pay refund. A single KEKS Pay transaction can have multiple KEKS Pay refunds. Returned from KEKS server
amount	Numeric	Amount refunded
currency	String	HRK default
status	Integer	Numeric code of the result. Only value 0 means that refund was successfully processed by KEKS
message	String	Text message

<u>Example of refund request content</u>

```
{
“bill_id” : “HGHGHG121222”,
“keks_id” : “15488457”,
“tid” : “123456666”,
 “amount”: 23.95 ,
}
```

Example of refund response content

{

“bill_id” : “HGHGHG121222”,

“keks_refund_id” : “26588458”,

“amount”: 23.95 ,

“status” : 0,

“message” : “refunded”,

}

Request/Response samples

Test DESkey **FC011AEDA9632ED96446F8CF**

TID **P00372**

CID **C00371**

Refund with bad hash does not pass

```
curl -i -k -H "Content-Type: application/json" -H "Accept: application/json; charset=utf-8" -X POST -d
'{"bill_id":"C003214PxV9NnsckaSc","tid":"P00372","cid":"C00371","amount":"1.00","epochtime":159
3095191,"hash":"BE897077FD635C1B4272000CC93C2E1AE3B2A0340BA0766F","currency":"HRK"}'
https://hrs000lin096/eretailer/keksrefund
```

```
HTTP/1.1 200

Server: nginx/1.12.2

Date: Tue, 05 Jan 2021 13:46:21 GMT

Content-Type: application/json;charset=utf-8

Content-Length: 82

Connection: keep-alive

{"epochtime":1609854381568,"status_code":-13,"message":"Processed","tid":"P00372"}
```

Refund with good hash

```
curl -i -k -H "Content-Type: application/json" -H "Accept: application/json; charset=utf-8" -X POST -d
'{"bill_id":"C00371","tid":"P00372","cid":"C00371","amount":"1.00","epochtime":1593095191,"hash
":"6C4E6CCD85BBCCC0276634BF026BD8D32DAE4A8C76596182","currency":"HRK"}'
https://hrs000lin096/eretailer/keksrefund
```

```
HTTP/1.1 200

Server: nginx/1.12.2

Date: Tue, 05 Jan 2021 13:55:40 GMT

Content-Type: application/json;charset=utf-8

Content-Length: 119

Connection: keep-alive


{"bill_id":"C00371","amount":1,"keks_refund_id":"1609854940202276907","currency":"HRK","mess
age_type":9944,"status":0}
```

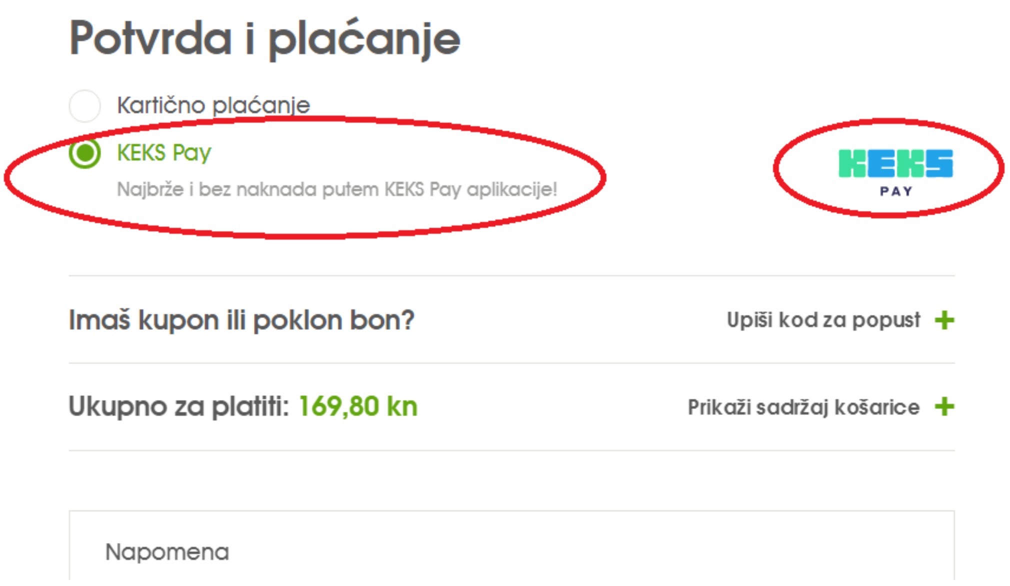
QR codes

The minimal set of information that WEBSHOP must include in SELL request is to be presented in QR form at the beginning. This QR code should adhere to the following specifications:

- Standard model 2 QR code (version up to 40), with up to 4296 alphanumeric characters (a maximum of 177 modules supported)
- Hex (string) encoding
- Minimal symbol size : 93 modules (with error correction), version 15-19 – for suggested dataset, subject to change
- Minimal physical print size : approx. 1mm per module or :
  - For print-head with 600 dpi density (laser) - 0,42 mm per module
  - For print-head with 300 dpi density (thermal) - 0,75 mm per module (6-dot module)
  - backlit LED 3840x2160 pix, 331 ppi, approx. 0,81 mm per module
  - LED 2560x1440 pix, 221 ppi, approx. 0,90 mm per module
  - LCD 1920x1080 pix, 166 ppi, approx 1,06 mm per module
  - DOT-matrix, approx. 60 ppi, approx. 1,4 mm per module
  - Error correction : M (or L, depending on resolution)
  - Safe margin : 4 or more modules
  - FrameQR : supported, up to 25% center frame

Visual Requierments

The payment method must be written as following: **KEKS Pay**. The Payment method description must be “**Najbrže i bez naknada putem KEKS Pay aplikacije!”** (see example below). If possible, the KEKS Pay logo should be applied accordingly. KEKS Pay logo will be provided by the KEKS Pay team.



Potvrda i plaćanje

( ) Kartično plaćanje

(●) KEKS Pay

Najbrže i bez naknada putem KEKS Pay aplikacije!

Imaš kupon ili poklon bon? Upiši kod za popust +

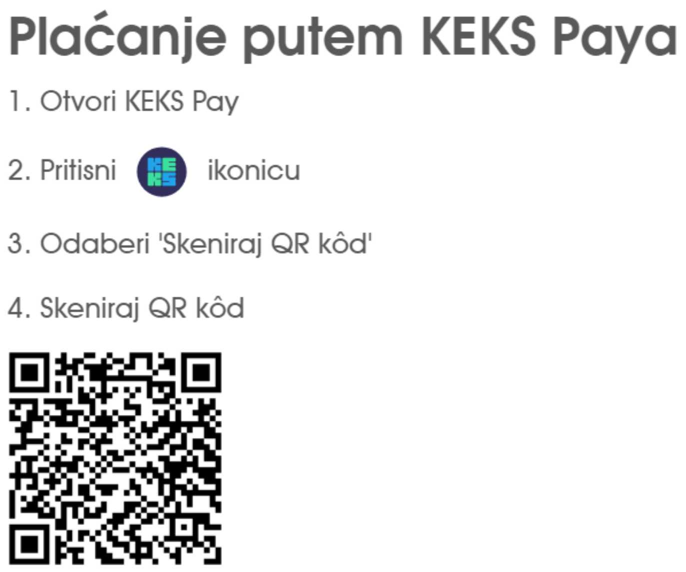
Ukupno za platiti: 169,80 kn Prikaži sadržaj košarice +

Napomena

The instruction how to scan the QR code with KEKS **Pay from the computer** screen must be shown together with the QR code (see example below).

The instruction must be as following:

1. **Otvori KEKS Pay**
2. **Pritisni ikonicu** *(the icon picture will be provided by KEKS Pay team)*
3. **Pritisni Skeni raj QR kôd**
4. **Skeniraj QR kôd**



Plaćanje putem KEKS Paya

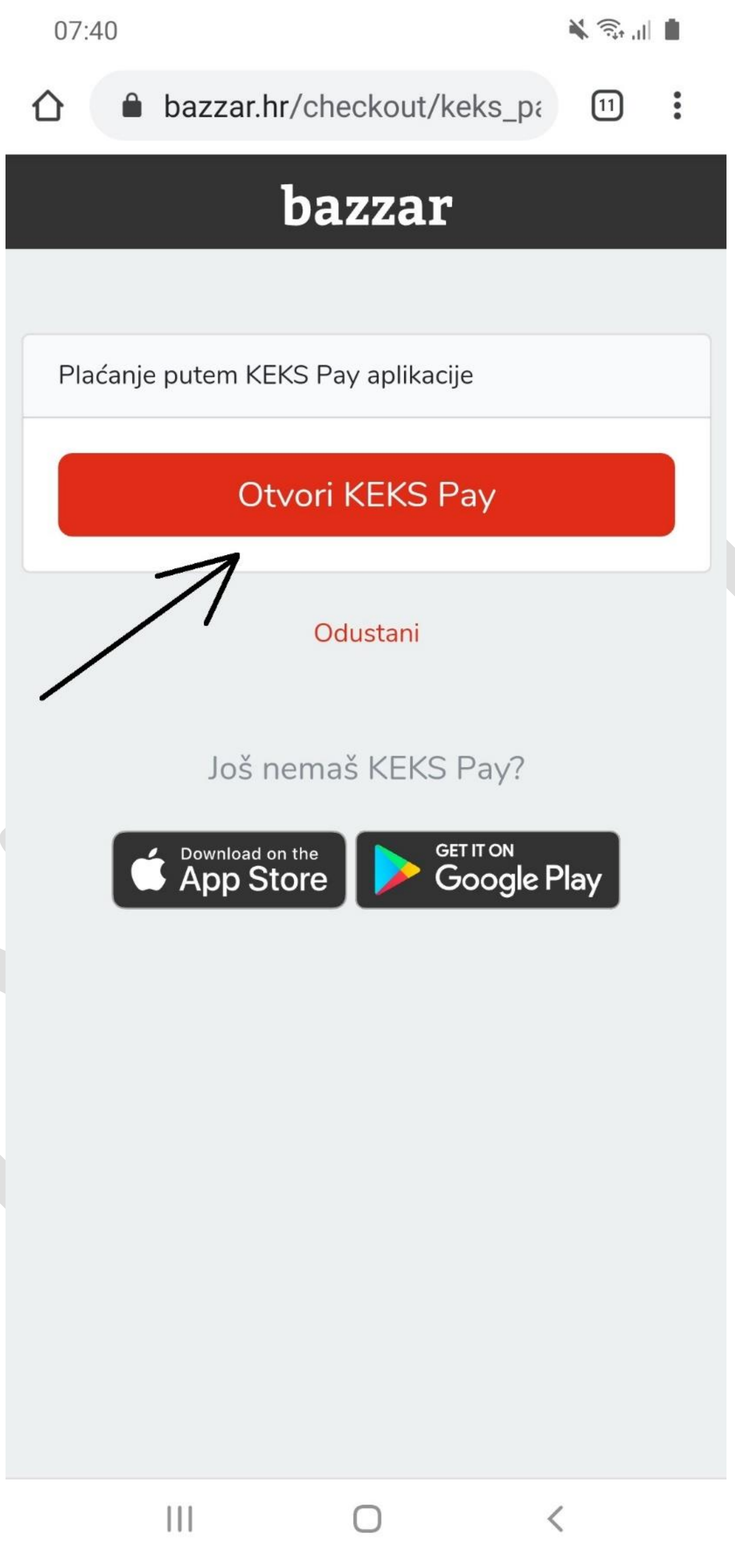
1. Otvori KEKS Pay

2. Pritisni ikonicu

3. Odaberi 'Skeniraj QR kôd'

4. Skeniraj QR kôd

The generated deep link on smart phone must be shown as a button with the following text: “**Otvori KEKS Pay”** (Meaning “Open KEKS Pay”, see example below)


07:40
bazzar.hr/checkout/keks_pa
bazzar
Plaćanje putem KEKS Pay aplikacije
Otvori KEKS Pay
Odustani
Još nemaš KEKS Pay?
Download on the App Store
GET IT ON Google Play