Victory Pay eKasa

API documentation

Document version	1.2.1
Document date	12.11.2024
App version	2.16.x

Document contents

General API information

Protocol & communication

The service is listening on port 12412 on the target device (terminal).
HTTP protocol is used, requests/responses are encoded as JSON.

Authentication

It is mandatory to include an Authentication header with each request.
Username for HTTP Basic auth: p3kasa
Password for HTTP Basic auth: p3kPW

```
Authorization: Basic cDNrYXNhOnAza1BX
```

Date-time format

All date-time fields, unless explicitly stated otherwise, are formatted as:

```
{day 01-31}.{month 01-12}.{4-digit year} {hours 00-23}:{minutes 00-59}:{seconds 00-59}
```

Example: 31.12.2023 23:59:59
Java format: `dd.MM.yyyy HH:mm:ss`
PHP format: `d.m.Y H:i:s`

Merchant configuration

In order to use the eKasa functionality, a merchant needs to be configured.
This is done by uploading two files (authentication data and identity data) provided by the tax office for the merchant.
This needs to be done when a new WORM memory module (CHDÚ) is installed or when the previously used merchant authentication data has expired.

Note that when using a mock application, actual merchant authentication/identification data sent in the request are ignored.

Uploading merchant data

URL: /api/merchant/store
Method: POST

Request body

Field	**Type**	**Description**	**Rules**
`identificationData`	`string`	Identification data XML file as a string	not blank
`authenticationData`	`string`	Authentication data XML file as a string	not blank
`keyStorePassword`	`string`	Authentication data password	not blank
`privateKeyPassword`	`string`	Authentication data password	not blank

Example request body

```
{
        "authenticationData": "<eu:AuthData xmlns:eu=\"...\">...</AuthData>",
        "identificationData": "<eu:IdentityData xmlns:eu=\"...\">...</IdentityData>",
        "keyStorePassword": "Heslo123",
        "privateKeyPassword": "Heslo123"
}
```

Response body

Field	**Type**	**Description**	**Rules**
`merchant`	Merchant	Merchant data	not null

`Merchant`

Field	**Type**	**Description**	**Rules**
`id`	`int32`	Sequence number of merchant generated by Victory Pay eKasa	not null
`corporateFullName`	`string`	Name of the merchant	not blank
`ico`	`string`	Registration number of the merchant (IČO)	
`dic`	`string`	TAX ID of the merchant	not blank
`icDph`	`string`	VAT ID of the merchant	
`physicalAddress`	`PhysicalAddress`	Address where the business is registered	not null
`organizationUnit`	`OrganizationUnit`	Information about the organization unit (branch)	not null
`securityData`	-	Always **NULL** in current versions of Victory Pay eKasa	**NOT USED**

OrganizationUnit

Field	Type	Description	Rules
name	string	Organization unit (branch) name	max. 255 chars
cashRegisterCode	string	Code given to the cash register by the tax office	not blank
cashRegisterType	CashRegisterTypeEnum	Either "STANDARD" or "PORTABLE"	not null
location	Location	Location of the cash register.	

Location

Field	Type	Description	Rules
physicalAddress	PhysicalAddress	Physical address	
gps	Gps	GPS coordinates	
other	string	Other location, e.g. vehicle plate number	

PhysicalAddress

Field	Type	Description	Rules
country	string	Country name (long), e.g. "Slovenská Republika"	max. 255 chars
municipality	string	Municipality, e.g. "Bratislava-Ružinov"	not blank max. 100 chars
street	string	Street name, e.g. "Bajkalská"	not blank max. 100 chars
buildingNumber	string	Street number, e.g. "29/B"	max. 20 chars
propertyRegistrationNumber	string	Property number in municipal register, e.g. "717"	[0-9]{1,10}
postalCode	string	ZIP code, e.g. "82105"	[0-9]{5}

Gps

Field	Type	Description	Rules
x	decimal	GPS X coordinate, up to 16 fractional digit precision	not null
y	decimal	GPS Y coordinate, up to 16 fractional digit precision	not null

Retrieving merchant data

URL: /api/merchant/get
Method: GET

Response body

Field	Type	Description	Rules
merchant	Merchant	Merchant data	not null

Printing

Victory Pay eKasa gives you some control over the printing.
For every request that results in something being printed, you can include a **`printer`** field in the root of the request body to set print parameters.

Printer

Object used for setting printing parameters.

Field	Type	Description	Rules
`socket`	`string`	IP address (tcp://1.2.3.4:12345) or handle (/dev/example) of the printer	
`charSize`	`int32`	Character size	

Printer selection

The text is normally printed using the built-in printer, which Victory Pay eKasa controls exclusively.
There is a dedicated application available which is capable of using external ESCP printers over IP.
To use the external printer, you need to specify its IP address in every request where printer output is expected.

Print text size

We support selection of character size on some printers.
Using smaller characters allows to fit more text on a single line.
You can request a list of supported print character sizes using an API documented below.

Retrieving supported print character sizes

URL: /api/printer/supported_char_sizes
Method: GET

Response body

Field	Type	Description	Rules
`supportedCharSizes`	`int32[]`	List of supported character sizes for printing	not null

`printLineLengthsByCharSizes`	`Map`
	`Key: int32`
	`Value: int32`
		Map of line lengths (number of characters on line) by character sizes.
		Key: character size
		Value: line length (number of characters on line)
			not null

Printing arbitrary text

Slovak law requires that everything that is printed on a printer that's connected to a certified eKasa client (cash register) must be stored in the WORM memory module (CHDÚ), meaning it must go through Victory Pay eKasa.

In order to print arbitrary text on the printer and have it stored in the WORM memory, you can use the following API:

URL: /api/print
Method: POST

Request body

Field	Type	Description	Rules
`printData`	string	Text to print. Lines are separated using the newline (\n) character.	not blank
`printer`	Printer	Print parameters	

Example request body

```
{
        "printData": "Hello world!\nThis text is on a new line.",
        "printer": {
                "charSize": 18
        }
}
```

Response body

Field	Type	Description	Rules
`rawData`	string	Text that was printed	not blank

Checking if printer is ready

Send text "PrinterCheck" to the endpoint for printing arbitrary text.
The request should look like this:

URL: /api/print
Method: POST
Body:

```
{
        "printData": "PrinterCheck"
}
```

If the printer is ready, you will receive a response with HTTP code 200 that will look like this:

```
{
        "rawData": "Printer is ready!"
}
```

If the printer is NOT ready, you will receive an error response with HTTP code 400 that will look something like this:

```
{
        "error": "Print error: error description"
}
```

Storing & retrieving eKasa documents

Documents (receipts) can be stored and later retrieved using an identifier provided by the client (the **`clientDocId`**).

When a document request is repeated with the same **`clientDocId`** as before, a response with the current document state will be returned instead of storing the document again as a new one.

When a document is stored while the device is offline (or unable to reach servers of the tax office), some fields like document UID (the **`uuid`** field) that are normally present will be null. Once the document has been successfully sent to the tax office, these fields can be retrieved using the document retrieval (status) API.

Notes about returnable packaging

- When selling returnable packaging:
  - the itemType must be "SALE"
  - the name must start with "VO:"
  - the vatRate must be "VAT_0"
- When refunding returnable packaging:
  - the itemType must be "PACKING_REFUND"
  - the vatRate must be "VAT_0"

Notes about rounding

Every cash payment is rounded to the closest 0.05€, or exactly 5 cents if the rounding would yield 0 payment amount.
Rounding applies to **cash payment amount**, not to item's price or the document value - those are not affected by rounding.

Cash payments

In case of purely cash payment (no other cashless methods), the entire document value is rounded to the closest 0.05€.

Example 1: document amount 10,43 -> cash payment 10,45
Example 2: document amount 10,42 -> cash payment 10,40
Example 3: document amount 00,02 -> cash payment 00,05

Cashless payments

In case of other (cashless) payments, the rounding doesn't apply!

Example 4: document amount 10,43 -> card payment = 10,43

Combined payments

In case of combined payment (cash and cashless methods), the amount remaining after subtracting the cashless payment amount from the document value is rounded to the closest 0.05€.

Example 5: document amount 10,43 -> card payment = 10,00; cash payment = 0,45
Example 6: document amount 10,43 -> card payment = 0,43; cash payment = 10,00

Notes about payments & document value

The declared document value must be equal to the value of items on the receipt (sum of all rows), plus or minus the rounding amount applied to the cash payment. So basically the declared document value must be equal to the sum of all payments.

Example: value of items = 100,12, payments: card = 100, cash = 0,10 (0,12 rounded to 0,10), final document value = 100,10.

You can have multiple payments of the same type (e.g. payment using 2 different cards).
You should only declare one cash payment. If there are multiple cash payments, their sum will be used internally.

Notes about VAT rate changes since January 2025

What is changing?

The following VAT rates are being **removed**: 10%, 20%
The following VAT rates are being **added**: 5%, 19%, 23%

Signs / letters identifying VAT rates printed on receipt will also change.
Letters A, B and C are now allocated to the new VAT rates, whereas old VAT rates have been assigned D and E letters.

VAT rate	Old sign / letter	New sign / letter
23 %	-	**A**
19 %	-	**B**
5 %	-	**C**
20%	**A**	**D**
10%	**B**	**E**
0%	**C**	**N**

When is the change?

January 1st, 2025, Central European Standard Time.

UNIX timestamp: 1735686000

Victory Pay eKasa application will automatically switch to new VAT rates at midnight
between Tuesday, December 31st, 2024 and Wednesday, January 1st, 2025.

How does this affect the API?

The enumeration of VAT rates (`VatRateTypeEnum`) now includes the new VAT rates (`VAT_5`, `VAT_19`, `VAT_23`).
Note that the old VAT rates are still present in the enum, and will remain present essentially forever.

Usage before / after 01.01.2025

Prior to 01.01.2025, you can **NOT** use new VAT rates at all, except in a special app version intended for integrators.

Starting on 01.01.2025, new documents can NOT contain items of type(`DocumentItemTypeEnum`) SALE with old VAT rate, unless they're being paid for by an item of type DEPOSIT (advance payment deduction) or VOUCHER (exchange of single purpose voucher) with old VAT rate, and only up to the amount of the given DEPOSIT or VOUCHER.

You can still REFUND and UPDATE items with old VAT rates.
If you previously sold it with an old VAT rate (before the VAT rate changed), you can still refund/update it with that old VAT rate.

You can combine old and new VAT rates, e.g. refund an item sold with old VAT, and sell an item with new VAT.

You can learn more at https://integrations.victorypay.eu/integrations/new-vat-rates-for-2025/

Storing documents

URL: /api/document/store
Method: POST

Request body

Field	**Type**	**Description**	**Rules**
`clientDocId`	`UUIDv4`	ID of the document generated by the client. UUID is used since the fiscal service may potentially be called by multiple clients and we need to guarantee universal uniqueness here. This field is optional, but omitting it is strongly discouraged, as it prevents document duplication prevention mechanisms from working. You also won't be able to retrieve such documents.	
`type`	`DocumentTypeEnum`	Type of document	not null
`externalId`	`string`	External ID of the document	max. 100 chars
`invoiceId`	`string`	ID of invoice, required for type=UF	max. 50 chars
`invoiceText`	`string`	Text to display on invoice payment receipt	
`amount`	`decimal`	Declared document value. This is basically a sum of all payments. See notes about payments & document value for more info.	not null, max. 8 integer digits max. 2 fractional digits
`paragonDate`	`string`	Date-time. When present, the document is stored as paragon.	
`documentEntries`	`ReceiptItemInputModel[]`	Items on receipt	
`payments`	`Payment[]`	Payments with their respective method, amount and label. Required when document value != 0.	
`header`	`string`	Custom header to print at the top of the receipt	
`footer`	`string`	Custom footer to print at the bottom of the receipt	
`electronicReceipt`	`boolean`	Whether the receipt will be sent electronically. Setting this to true will result in the document not being printed if the cash register is online and able to reach the tax office's servers. It's the responsibility of the application calling Victory Pay eKasa service to send an electronic receipt to the customer.	
`exception`	`boolean`		
`invertNegativePrice`	`boolean`	Whether to invert the value of unit price and amount on the receipt. Only works for items with negative value. For example, it will turn **-5,00 €** x **4 pcs** to **5,00 €** x **-4 pcs**.	
`ZTP`	`boolean`	Whether the merchant is officially a "Zdravotne ťažko postihnutá osoba" (disabled person). By law, a severely disabled person is not required to send documents to the tax office. Setting this to true prevents the document from being sent to the tax office. The document is still stored in the WORM memory module.	
`printer`	`Printer`	Print parameters	

DocumentTypeEnum

Enum value	Description
PD	Receipt
UF	Invoice payment
ND	Non-valid receipt (test receipt)
VY	Withdrawal
VK	Deposit

ReceiptItemInputModel

Field	Type	Description	Rules
itemType	DocumentItemTypeEnum	Item type	not null
externalId	string	External ID of the item	max. 100 chars
name	string	Name of the item	not blank max. 255 chars
price	decimal	Unit price of the item (incl. VAT)	not null not zero max. 8 integer digits max. 4 fractional digits
quantity	decimal	Quantity of the item	not null positive value max. 8 integer digits max. 4 fractional digits
measureUnitCode	MeasureUnitCodeEnum	Measure unit code, used to indicate measure unit on receipt.	
vatRate	VatRateTypeEnum	VAT rate assigned to the item	not null
referenceDocumentId	string	If itemType=UPDATE or REFUND, a reference to the original receipt on which the item being updated or refunded was present is needed. This can either be UID, OKP or document sequence number.	
specialRegulation	SpecialRegulationTypeEnum	Reason for 0% VAT rate	
seller	Seller		
voucherNumber	string	Number of voucher for itemType=VOUCHER	max. 50 chars
plu	int32	PLU of the item in a price list. Used for journals.	not zero
priceListId	int32	ID of a price list. Used for journals.	not zero
ean	string	EAN of item. Used for journals.	8-13 digits

DocumentItemTypeEnum

Enum value	Description
SALE	Product or service sold (ekasa=K)
PACKING_REFUND	Returned returnable packaging (ekasa=VO)
REFUND	Refund of returned product (ekasa=V)
UPDATE	Fix of a previously wrongly declared receipt item (ekasa=O)
DISCOUNT	Discount (ekasa=Z)
DEPOSIT	Redeemed deposit (ekasa=OZ)
VOUCHER	Redeemed single-purpose voucher (ekasa=VP)

VatRateTypeEnum

Enum value	Description
VAT_0	0% VAT rate (marked as "N" on receipts)
~~VAT_10~~	10% VAT rate (marked as "E" on receipts). Valid until **31.12.2024 23:59:59**.
~~VAT_20~~	20% VAT rate (marked as "D" on receipts). Valid until **31.12.2024 23:59:59**.
VAT_5	5% VAT rate (marked as "C" on receipts). Valid from **01.01.2025 00:00:00**.
VAT_19	19% VAT rate (marked as "B" on receipts). Valid from **01.01.2025 00:00:00**.
VAT_23	23% VAT rate (marked as "A" on receipts). Valid from **01.01.2025 00:00:00**.

SpecialRegulationTypeEnum

Enum value	Description
PDP	Transfer of TAX liability (prenesenie daňovej povinnosti)
OOD	Exempt from TAX (oslobodené od dane)
CK	Travel agencies (cestovné kancelárie)
PT	Used goods (použitý tovar)
UD	Works of art (umelecké diela)
ZPS	Collectibles and antiques (zberateľské predmety a starožitnosti)

Seller

Field	Type	Description	Rules
id	string	Seller ID	not null
sellerIdType	SellerIdTypeEnum	Type of seller ID	not null

SellerIdTypeEnum

Enum value	Description
DIC	TAX ID (Daňové Identifikačné Číslo)
IC_DPH	VAT ID (IČ DPH)

Payment

Field	Type	Description	Rules
method	Payment.Method	Payment method	not null
amount	decimal	Payment amount	not null, max. 8 integer digits max. 2 fractional digits
label	string	Label displayed next to the amount on the receipt	

Payment.Method

Enum value	Description
CASH	Cash
CARD	Payment card
VOUCHER	Voucher, e.g. meal voucher. Not single-purpose voucher - that one is supposed to be declared as an item of type VOUCHER (that's the law)!
OTHER	Other type of payment. Note that cash rounding doesn't apply to OTHER payment type.

MeasureUnitCodeEnum

Enum value	Name	Abbreviation	Slovak name	Slovak abbreviation (actually printed)
PCS	piece	pcs	kus	ks
PR	pair	pr	pár	pr
NMP	pack	pck	balenie	bal
ZP	page	pg	strana	str
MGM	milligram	mg	miligram	mg
CGM	centigram	cg	centigram	cg
DG	decigram	dg	decigram	dg
GRM	gram	g	gram	g
KGM	kilogram	kg	kilogram	kg
TNE	tonne	t	tona	t
MLT	millilitre	ml	mililiter	ml
CLT	centilitre	cl	centiliter	cl
DLT	decilitre	dl	deciliter	dl
LTR	litre	l	liter	l
HLT	hectolitre	hl	hektoliter	hl
MMT	millimetre	mm	milimeter	mm
CMT	centimetre	cm	centimeter	cm
DMT	decimetre	dm	decimeter	dm
MTR	metre	m	meter	m
MTK	square metre	m^2	meter štvorcový	m^2

MTQ	cubic metre	m^3	meter kubický	m^3
KMT	kilometre	km	kilometer	km
KMK	square kilometre	km^2	kilometer štvorcový	km^2
ARE	are	a	ár	a
HAR	hectare	ha	hektár	ha
LM	linear metre	lm	bežný meter	bm
SEC	second	sec	sekunda	sek
MIN	minute	min	minúta	min
HUR	hour	hr	hodina	hod
DAY	day	d	deň	d
WEE	week	w	týždeň	tyz
MON	month	mon	mesiac	mes
ANN	year	y	rok	r
WHR	watt-hour	Wh	watt-hodina	Wh
KWH	kilowatt-hour	kWh	kilowatt-hodina	kWh
MWH	megawatt-hour	MWh	megawatt-hodina	MWh
GWH	gigawatt-hour	GWh	gigawatt-hodina	GWh
TWH	terawatt-hour	TWh	terawatt-hodina	TWh
INH	inch	in	palec	in
INK	square inch	in^2	palec štvorcový	in^2
INQ	cubic inch	in^3	palec kubický	in^3
FOT	foot	ft	stopa	ft
FTK	square foot	ft^2	stopa štvorcová	ft^2
FTQ	cubic foot	ft^3	stopa kubická	ft^3
YRD	yard	yd	yard	yd
YDK	square yard	yd^2	yard štvorcový	yd^2
YDQ	cubic yard	yd^3	yard kubický	yd^3
SMI	mile	mi	míľa	mi
MIK	square mile	mi^2	míľa štvorcová	mi^2
ACR	acre	ac	aker	ac
ONZ	ounce	oz	unca	oz
LBR	pound	lb	libra	lb
LTN	long ton	tn	dlhá tona	tn
STN	short ton	tn	krátka tona	tn
OZI	fluid ounce (UK)	fl oz	dutá unca (UK)	fl oz
OZA	fluid ounce (US)	fl oz	dutá unca (US)	fl oz

PTI	pint (UK)	pt	pinta (UK)	pt
PT	pint (US)	pt	pinta (US)	pt
QTI	quart (UK)	qt	kvart (UK)	qt
QT	quart (US)	qt	kvart (US)	qt
GLI	gallon (UK)	gal	galón (UK)	gal
GLL	gallon (US)	gal	galón (US)	gal

Example request body

```
{
        "clientDocId": "128f59d5-9149-4654-9888-f39c0a1e48d8",
        "header": "This is a header text",
        "footer": "This is a footer text",
        "type": "PD",
        "amount": 227.25,
        "documentEntries": [
                {
                        "price": 75,
                        "quantity": 2,
                        "name": "Article 1",
                        "vatRate": "VAT_20",
                        "itemType": "SALE"
                },
                {
                        "price": 87.23,
                        "quantity": 1,
                        "name": "Article 2",
                        "vatRate": "VAT_10",
                        "itemType": "SALE"
                },
                {
                        "price": 10,
                        "quantity": 1,
                        "name": "Article 3",
                        "vatRate": "VAT_0",
                        "itemType": "SALE"
                },
                {
                        "price": -20,
                        "quantity": 1,
                        "name": "Returned article",
                        "vatRate": "VAT_20",
                        "itemType": "REFUND",
                        "referenceDocumentId": "O-D75E61D31B1D4D889E61D31B1DE-TEST"
                }
        ],
        "payments": [
                {
                        "method": "CARD",
                        "amount": 200.00,
                        "label": "VISA ####-####-####-1234"
                },
                {
                        "method": "CASH",
                        "amount": 27.25,
                        "label": "Cash"
                }
        ]
}
```

Response body

Field	Type	Description	Rules
`document`	`Document`	Document data	not null

Document

Field	Type	Description	Rules
`clientDocId`	`UUIDv4`	ID of the document generated by the client	
`internalDocumentId`	`int32`	Internal ID generated by Victory Pay eKasa	not null
`sequenceId`	`int32`	Sequence number of the document. Starts at 1 every calendar month.	not null
`paragonId`	`int32`	Sequence number of the paragon (if the document is a paragon)	
`isParagon`	`boolean`	Whether the document is a paragon	not null
`type`	`DocumentTypeEnum`	Type of document	not null
`externalId`	`string`	External ID of the document	max. 100 chars
`invoiceId`	`string`	ID of invoice, present for type=UF	max. 50 chars
`invoiceText`	`string`	Text printed on invoice payment receipt	
`amount`	`decimal`	Declared document value. This is basically a sum of all payments. See notes about payments & document value for more info.	not null max. 8 integer digits max. 2 fractional digits
`documentEntries`	`DocumentItem[]`	Items on receipt	
`payments`	`Payment[]`	Payments with their respective method, amount and label. Present when document value != 0.	
`header`	`string`	Custom header printed at the top of the receipt	
`footer`	`string`	Custom footer printed at the bottom of the receipt	
`electronic`	`boolean`	Whether the receipt was sent electronically (not printed)	not null
`exception`	`boolean`		
`invertNegativePrice`	`boolean`	Whether the unit price and amount were inverted on the receipt	
`merchant`	`Merchant`	Merchant data	not null
`cashRegisterCode`	`string`	Code given to the cash register by the tax office	not null
`issueDate`	`string`	Date-time when the document was issued.	not null
`createDate`	`string`	Date-time when the document was registered in ECR.	not null
`processDate`	`string`	Date-time when the document was registered in eKasa.	
`uuid`	`UUIDv4`	Receipt UUID generated by eKasa.	
`pkp`	`string`	PKP (merchant's digital signature)	not null, 344 chars
`okp`	`string`	OKP (formatted digest of the merchant's digital signature)	not null, 44 chars
`vatRateSums`	`VateRateSum[]`	VAT summary	not null
`cashRoundingAmount`	`decimal`	Amount used to round the cash payment to the closest 0.05€	not null
`sendingCount`	`int32`	How many attempts to send the document to the tax office were made. Normally 1. Incremented for every attempt to fix the document.	not null
`qrCode`	`string`	Contents of the QR code printed on the receipt	not null

DocumentItem

Field	Type	Description	Rules
type	DocumentItemTypeEnum	Item type	not null
externalId	string	External ID of the item	max. 100 chars
name	string	Name of the item	not blank, max. 255 chars
price	decimal	Unit price of the item (incl. VAT)	not null not zero max. 8 integer digits max. 4 fractional digits
quantity	decimal	Quantity of the item	not null positive value max. 8 integer digits max. 4 fractional digits
totalPrice	decimal	Calculated total price incl. VAT (price * quantity). Essentially the value of a row on the document. Always at least 0.01, even if price*quantity is < 0.01.	not null min. value = 0.01 max. 2 fractional digits
measureUnitCode	MeasureUnitCodeEnum	Measure unit code, used to indicate measure unit on receipt.	
vatRate	VatRateTypeEnum	VAT rate assigned to the item	not null
referenceDocumentId	string	A reference to an original receipt on which the item being updated or refunded was present. Present if itemType=UPDATE or REFUND. This can either be UID, OKP or document sequence number.	
specialRegulation	SpecialRegulationTypeEnum	Reason for 0% VAT rate	
seller	Seller		
voucherNumber	string	Number of voucher for itemType=VOUCHER	max. 50 chars
plu	int32	PLU of the item in a price list. Used for journals.	not zero
priceListId	int32	ID of a price list. Used for journals.	not zero
ean	string	EAN of item. Used for journals.	8-13 digits
invertNegativePrice	boolean	Whether the unit price and amount were inverted on the receipt.	not null

VatRateSum

Field	Type	Description	Rules
title	VatRateTypeEnum	VAT rate	not null
base	decimal	Sum without VAT (TAX base)	not null
vat	decimal	VAT amount	not null
sum	decimal	Sum with VAT	not null

Retrieving documents

You can retrieve the current state of any document using this API.
In order to use this functionality, you must include the `clientDocId` field when storing documents.

URL: /api/documents/*{clientDocId}*
Method: GET

Response body

See Document.

Printing copy of documents

You can print a copy of any document using this API.
In order to use this functionality, you must include the `clientDocId` field when storing documents.

URL: /api/print/document-copy
Method: POST

Request body

Field	Type	Description	Rules
`clientDocId`	`UUIDv4`	The clientDocId used when creating the document.	not null
`printer`	`Printer`	Print parameters (optional).	

Example request body

```
{
        "clientDocId": "128f59d5-9149-4654-9888-f39c0a1e453"
}
```

Response body

-

Retrieving last document

This legacy API allows you to retrieve the last document, even if you don't know its `clientDocId`.
You can also print a copy of the last document if you include the `printer` object in the request body.

URL: /api/document/get
Method: POST

Request body

Field	Type	Description	Rules
`printer`	`Printer`	Print parameters. Setting this to non-null value will print the offline documents report.	

Example request body 1 - get last document, don't print anything

```
{}
```

Example request body 2 - get last document and print copy of the receipt with default print settings

```
{
	"printer": {}
}
```

Response body

Field	Type	Description	Rules
`document`	`Document`	Last document	not null

Fixing document

If a previous document was invalid and **rejected by the tax office**, the cash register is required by law to block itself and prevent creation of new records of any kind until the rejected document is fixed.
Using this operation, you can fix certain fields of the document and its items, and submit it to the tax office again.
Note that the count and order of items to fix must match the original count and order of items.
Fields that are left null (or not present in the request) won't be updated and their original value will be used.

URL: /api/document/update
Method: POST

Request body

Field	**Type**	**Description**	**Rules**
`fixDocument`	`FixDocument`	Data used to fix the document	not null

FixDocument

Field	**Type**	**Description**	**Rules**
`invoiceId`	`string`	New invoice ID	max. 50 chars
`createDate`	`string`	New Date-time when the document was registered in ECR.	
`issueDate`	`string`	New Date-time when the document was issued.	
`documentEntries`	`FixDocumentItem[]`	Fixes to items.	
`useLastMerchant`	`boolean`	Whether the latest merchant data should be used instead of the original merchant data from back when the document was originally created. Defaults to FALSE.	

FixDocumentItem

Field	**Type**	**Description**	**Rules**
`type`	`DocumentItemTypeEnum`	New item type	
`name`	`string`	New name of the item	max. 255 chars
`quantity`	`decimal`	New quantity of the item	not null positive value max. 8 integer digits max. 4 fractional digits
`vatRate`	`VatRateTypeEnum`	New VAT rate	
`seller`	`Seller`	New seller	
`voucherNumber`	`string`	New number of voucher for itemType=VOUCHER	max. 50 chars
`referenceDocumentId`	`string`	New reference to the original receipt on which the item being updated or refunded (itemType=UPDATE or REFUND) was present. This can either be UID, OKP or document sequence number.	
`specialRegulation`	`SpecialRegulationTypeEnum`	New reason for 0% VAT rate	

Response body

Field	**Type**	**Description**	**Rules**
`document`	`Document`	Document with merged fixes	not null

Storing & retrieving eKasa cash deposits / withdrawals

Cash operations can be stored and later retrieved using an identifier provided by the client (the **`clientDocId`**).
Same logic applies as with documents.

Storing cash deposits / withdrawals

URL: /api/cash
Method: POST

Request body

Field	Type	Description	Rules
`clientDocId`	`UUIDv4`	ID of the cash operation generated by the client. UUID is used since the fiscal service may potentially be called by multiple clients and we need to guarantee universal uniqueness here. This field is optional, but omitting it is strongly discouraged, as it prevents cash operation duplication prevention mechanisms from working. You also won't be able to retrieve such documents.	
`amount`	`decimal`	Amount of the cash operation. Positive amount = deposit. Negative amount = withdrawal.	not null not zero max. 8 integer digits max. 2 fractional digits
`externalId`	`string`	External ID	max. 100 chars
`exception`	`boolean`		
`printer`	`Printer`	Print parameters	

Example request body

```
{
        "clientDocId": "18460181-ac04-4635-8161-42a157cb449b",
        "amount": 100
}
```

Response body

Field	Type	Description	Rules
`cash`	`Cash`	Cash operation data	not null

Cash

Highlighted in **green** are fields added by Victory Pay eKasa on top of the request data.
Highlighted in **yellow** are fields which are only present for cash operations successfully sent to the tax office.

Field	Type	Description	Rules
`clientDocId`	`UUIDv4`	ID of the document generated by the client	
`internalDocumentId`	`int32`	Internal ID generated by Victory Pay eKasa	not null
`sequenceId`	`int32`	Sequence number of the document. Starts at 1 every calendar month.	not null
`externalId`	`string`	External ID of the document	max. 100 chars
`amount`	`decimal`	Document amount	not null max. 8 integer digits max. 2 fractional digits
`electronic`	`boolean`	Whether the receipt was sent electronically (not printed)	not null
`exception`	`boolean`		
`merchant`	[`Merchant`]()	Merchant data	not null
`cashRegisterCode`	`string`	Code given to the cash register by the tax office	not null
`issueDate`	`string`	Date-time when the document was issued.	not null
`createDate`	`string`	Date-time when the document was registered in the ECR.	not null
`processDate`	`string`	Date-time when the document was registered in eKasa.	
`uuid`	`UUIDv4`	Receipt UUID generated by eKasa.	
`pkp`	`string`	PKP (merchant's digital signature)	not null, 344 chars
`okp`	`string`	OKP (formatted digest of the merchant's digital signature)	not null, 44 chars
`sendingCount`	`int32`	How many attempts to send the document to the tax office were made. Normally 1. Incremented for every attempt to fix the document.	not null
`qrCode`	`string`	Contents of the QR code printed on the receipt	not null

Retrieving cash deposits / withdrawals

You can retrieve the current state of a cash operation using this API.

URL: /api/cash-operations/*{clientDocId}*
Method: GET

Response body

See Cash.

eKasa cash register location

Updating cash register location

Use this API to report the location of a portable cash register to the tax office.
You can choose to report the location as either an address, gps coordinates or something else (other).

URL: /api/location/update
Method: POST

Request body

Field	Type	Description	Rules
physicalAddress	PhysicalAddress	Physical address	
gps	Gps	GPS coordinates	
otherLocation	string	Other location, e.g. vehicle plate number	max. 255 chars

Response body

Field	Type	Description	Rules
location	Location	Location data	

Location

Highlighted in **green** are fields added by Victory Pay eKasa on top of the request data.
Highlighted in **yellow** are fields which are only present for location updates successfully sent to the tax office.

Field	Type	Description	Rules
clientDocId	UUIDv4	ID of the document generated by the client	
internalId	int32	Internal ID generated by Victory Pay eKasa	not null
physicalAddress	PhysicalAddress	Physical address	
gps	Gps	GPS coordinates	
other	string	Other location, e.g. vehicle plate number	max. 255 chars
merchant	Merchant	Merchant data	not null
cashRegisterCode	string	Code given to the cash register by the tax office	not null
createDate	string	Date-time when the document was registered in the ECR.	not null
processDate	string	Date-time when the document was registered in eKasa.	
sendingCount	int32	How many attempts to send the location update to the tax office were made. Normally 1. Incremented for every attempt to fix the location update that was previously rejected by the tax office.	not null

Fixing cash register location update

If a previous location update was invalid and **rejected by the tax office**, the cash register is required by law to block itself and prevent creation of new records of any kind until the rejected location update is fixed.

Using this operation, you can fix certain fields of the location update and submit it to the tax office again.
Fields that are left null (or not present in the request) won't be updated and their original value will be used.

URL: /api/location/update
Method: POST

Request body

Field	**Type**	**Description**	**Rules**
`fixLocation`	`FixLocation`	Data used to fix the location update	

FixLocation

Field	**Type**	**Description**	**Rules**
`createDate`	`string`	New Date-time when the location update was registered in the ECR.	
`physicalAddress`	`PhysicalAddress`	New physical address	
`gps`	`Gps`	New GPS coordinates	
`other`	`string`	New other location, e.g. new vehicle plate number	max. 255 chars
`useLastMerchant`	`boolean`	Whether the latest merchant data should be used instead of the original merchant data from back when the location update was originally registered in the ECR. Defaults to FALSE.	

Response body

Field	**Type**	**Description**	**Rules**
`location`	`Location`	Location with merged fixes	

Offline documents

Offline documents are created when the device is not connected to any network or is unable to reach the tax office's servers and receive a response within a set timeout (approximately ~2s).

Offline documents are stored in a queue and must be sent to the tax office within 2 days.

Each time a new online document is created, 3 oldest offline documents are automatically sent to the tax office.
Three documents is the minimum set by the tax office, which Victory Pay eKasa follows.
Increasing this number would result in major slowdowns when many offline documents are queued, therefore we chose not to do that, and provide an API to manually send a larger batch of offline documents at a time that's convenient for the merchant.

Retrieving and printing offline documents

This API allows you to retrieve all the offline documents that are yet to be sent to the tax office.
If you also want to print the offline documents report, set the **`printer`** field to any non-null value (empty object is enough).

URL: /api/document/get/offline
Method: POST

Request body

Field	Type	Description	Rules
`printer`	`Printer`	Print parameters. Setting this to non-null value will print the offline documents report.	

Example request body 1 - get offline documents, don't print anything

```
{}
```

Example request body 2 - get offline documents and print report with default print settings

```
{
        "printer": {}
}
```

Response body

Field	Type	Description	Rules
`documents`	`Document[]`	Offline documents that are yet to be sent to the tax office	not null
`locations`	`Location[]`	Offline location updates that are yet to be sent to the tax office	not null

Sending offline documents to the tax office

Use this API to send offline documents to the tax office in batch.

You can also configure how many queued offline documents you want to send per batch using the **`limit`** parameter.
If you want to send **ALL** offline documents, omit the **`limit`** parameter or set it to NULL.

URL: /api/document/send/offline
Method: POST

Request body

Field	**Type**	**Description**	**Rules**
`limit`	`int32`	Maximum number of documents & location updates to send in this batch	positive

Response body

Field	**Type**	**Description**	**Rules**
`documents`	`Document[]`	Documents successfully sent to the tax office in this batch	not null
`locations`	`Location[]`	Location updates successfully sent to the tax office in this batch	not null

Software information retrieval

URL: /api/swinfo
Method: GET

Response body

Field	Type	Description	Rules
`ppekkVersion`	`string`	PPEKK version (Victory Pay eKasa version as declared in certification)	not null

Merchant auth-data certificate expiration retrieval

URL: /api/merchant/expiration
Method: GET

Response body

Field	Type	Description	Rules
`expiration`	`int64`	UNIX timestamp after which the auth-data certificate is no longer valid	not null

CHDU usage retrieval

URL: /api/chdu/usage
Method: GET

Response body

Field	Type	Description	Rules
`usage`	`float`	CHDU usage in percent. Actual value is rounded using a ceiling function to two (2) fractional digits.	not null