

Connections and API

Guide for integrators: WooCommerce and Shopify, PrestaShop, OpenCart, Magento and BigCommerce, n8n, Zapier and Make, and your own system through the REST API. Issuing, cancellation, refund, advance, copy, document status and outgoing notifications.

SOFTWARE

Tezga eKasa, approved ESIR
ИБ 1597, version 1.1

API

Version 1.6.0
[pos.narbiz.com/api/openapi.json?](https://pos.narbiz.com/api/openapi.json?lang=en)
[lang=en](https://pos.narbiz.com/api/openapi.json?lang=en)

DOCUMENT

Version 1.0
Updated 30.10.2026.

PUBLISHER

Go Simple, Ivan Dabetić, sole trader
Trgovačka 1, 11307 Boleč, Serbia

TAX ID / COMPANY NO.

115468360 / 68389631

SUPPORT

podrska@tezgafiskal.com
tezgafiskal.com/en/fiscalization-api

Written so you can follow it with no prior knowledge of fiscalisation; every term is explained in the Glossary (chapter 1). The most up-to-date examples, with your key, are in the app: Settings → Integrations.

Contents

- | | |
|--|---|
| 1. Glossary | 14. The response you get |
| 2. How Tezga creates an invoice | 15. Errors and HTTP codes |
| 3. Two ways to connect: which one is yours | 16. Limits |
| 4. API key | 17. n8n step by step |
| 5. Two-way connection: WooCommerce | 18. Troubleshooting |
| 6. Two-way connection: Shopify | 19. Document status (GET documents) |
| 7. Connection settings (switches) | 20. Cancellation and refund (POST refund) |
| 8. Webhook: the basics | 21. Tax labels and environment (GET tax-labels) |
| 9. Webhook by platform | 22. Outgoing notifications (webhook) |
| 10. Custom contract (n8n / Zapier / Make) | 23. OpenAPI description |
| 11. Flow (tok): what exactly gets created | 24. Advance invoice (POST advance) |
| 12. Pro forma → payment → invoice | 25. Invoice copy (POST copy) |
| 13. Tax rates | 26. Products and customers from an ERP (POST products, customers) |

1. Glossary

If something in this document is unclear, it is probably explained here. Worth reading before anything else.

TERM	WHAT IT MEANS, IN PLAIN LANGUAGE
Fiscal receipt	An invoice reported to the Serbian Tax Administration, with a QR code. It is issued when you sell <i>at retail</i> (to individuals, or to anyone in a retail outlet).
Promet	The name of the fiscal receipt type for an ordinary sale. When this guide says “a promet is created”, it means “a fiscal receipt is created”.
eFaktura	An electronic invoice for companies (B2B) and the state. It is not fiscalised; it goes to the state system SEF. The buyer's tax ID is mandatory for it.
SEF	The e-invoicing system, the state site through which companies exchange invoices.
PFR	Fiscal invoice processor, the part of the Tax Administration that approves a fiscal receipt and returns its number. The fiscal receipt number is assigned by the PFR and cannot be set from outside.
ESIR	Software allowed to issue fiscal receipts. Tezga is an accredited ESIR.
PIB	A company's tax identification number, 9 digits. If there is one, the customer is a company.
Predračun	Pro forma invoice: a request for payment. It is not an invoice and creates no tax liability. It is sent to the customer to pay, and the real invoice is created when the money arrives.
Webhook	When something happens in your system (such as a new order), it sends a message to Tezga by itself. You “push” the data.
Endpoint	The address that message is sent to.
Header	Extra data sent with a request, such as your key. It is not visible in the address.
Idempotency	If you send the same message twice, two invoices are not created. It is safe to retry.
HMAC signature	A mathematical seal on a message that proves it really came from you and that nobody changed it on the way.

Serbian names in the API

Field names, enum values and error messages are in Serbian and stay exactly as written, in every language of this guide. Send them as shown. The ones you will meet most often:

NAME	MEANING
kupac , naziv	customer, name
stavke , sifra	line items, product code (SKU)
kolicina , jedinicaMere	quantity, unit of measure (kom = piece)
cenaSaPdv , poreskaStopa	unit price including VAT, tax rate
popustProcentat , popustRsd	discount in percent, discount in dinars
nacinPlacanja	payment method (Gotovina = cash, Kartica = card, Prenos = bank transfer)
tok , tipDokumenta	flow, document type (racun = invoice, predracun = pro forma)
pozivNaBroj , model	payment reference number, reference model
posaljiMejl , posaljiNaSef	send by email, send to SEF
storno , refundacija	cancellation, refund
kreirano , postojalo , greska	created, already existed, error
izdat , storniran , nacrt	issued, cancelled, draft

2. How Tezga creates an invoice

The whole system fits in one sentence: **Tezga learns about an order, decides which document is needed, creates it and reports it to the state.**

- Tezga learns about the order.** Either it pulls the order from your shop itself (two-way connection), or you send it (webhook). That is the only difference between the two ways of connecting.
- Tezga recognises the products.** If a line item has a `sifra` that exists in your catalogue, the name and tax rate are always taken from the catalogue. The price depends on the source: an order from a web shop (WooCommerce, Shopify, PrestaShop, OpenCart, Magento, BigCommerce) uses the price the customer **paid** in the shop; a custom call uses the price and discount from the catalogue, unless you send `"cenaIzZahteva": true` (chapter 10). Unknown code → what was sent is used.
- Tezga decides which document is needed.** No PIB → fiscal receipt. PIB present → you choose with the `tok` field (chapter 11). Order paid in advance with `tok:"avans"` → advance invoice (chapter 24).
- The document is created and reported.** A fiscal receipt goes to the PFR and gets a QR code; an eFaktura goes to SEF.

Why this matters: the tax rates on an invoice come from the **register** whenever a product is recognised by its code. For a custom call without `cenaIzZahteva` the price comes from the register too: if such an invoice seems to have the “wrong” price, the reason is almost always that the code exists in the Tezga catalogue with a different price. Until 16.09.2026 the same applied to web shops, so an invoice could carry the catalogue amount instead of the amount paid; this has been fixed.

3. Two ways to connect: which one is yours

There are exactly two. Do not use both for the same shop.

A. Two-way connection (Tezga pulls by itself)

You leave your shop's access details in Tezga. Every 30 minutes Tezga checks for new orders, creates invoices and **writes the invoice number back** to the shop.

Available for: WooCommerce and Shopify.

B. Webhook (your system sends)

Your platform sends the order to a Tezga address as soon as it is created.

For: everything else: PrestaShop, OpenCart, Magento, BigCommerce, your own site, n8n, Zapier, Make.

Which to choose

YOUR SITUATION	TAKE
You have WooCommerce	Two-way connection (ch. 5). It is safer, also catches missed orders, and respects mappings and delivery.
You have Shopify	Two-way connection (ch. 6). A webhook does not work for Shopify, see the warning below.
You have PrestaShop / OpenCart / Magento / BigCommerce	Webhook (ch. 9).
You have your own site or an automation (n8n, Zapier, Make)	Webhook with the custom contract (ch. 10). The most precise, with the most control.
You charge by bank transfer, so the invoice comes only after payment	Pro forma (ch. 12).

Shopify and webhooks do not go together. A Shopify webhook cannot send a custom header, and Tezga requires `X-Tezga-API-Key`. For Shopify use the two-way connection only.

Do not enable both ways for the same shop: you get double work for no benefit. (You will not get duplicate invoices, because Tezga recognises the same order by `externalId` and provider, but there is no reason to process it twice.)

4. API key

The key is your register's password. Whoever has it can issue invoices in your name.

Which plan

The API, outgoing webhooks and the two-way shop connection (WooCommerce, Shopify) are part of the **Posao** and **Biznis** plans. A company on the Tezga, Start or Sezonski plan cannot create an integration key, a webhook address or a shop connection, and every API request from such a company gets **402** with an explanation (chapter 15).

Companies that used the connection before 16.09.2026 keep it on their plan, even after changing plans: keys, shop connections and webhook addresses keep working, and they can create new ones. A key meant for a “register terminal” (Android register) is not an integration and works on every plan.

A company with the **Enterprise** level enabled has the API and web shop connection on any plan, because Enterprise includes everything from the Biznis plan.

How to create one

1. Open **Settings → Integrations**.
2. In the **API keys** section, enter a **Key name** (for example “WooCommerce shop”) and create it. Use a separate key for each integration, so you can pause or delete one without breaking the others.
3. **The key is shown exactly once.** Copy it right away, paste it where needed, then confirm that you have saved it.

Why only once: Tezga does not store your key in readable form, only its cryptographic fingerprint (hash), the way serious systems handle passwords. So even we cannot show it to you later. If you lose it, create a new one, move the integration to it, then delete the old one.

How to send it

Always as a header, never in the address:

```
X-Tezga-API-Key: <your key>
```

What it looks like

The key starts with `tzg_` followed by 48 hexadecimal characters. **Test and production keys look the same:** the environment is a property of the account (the company's security element), not of the key. To

make sure your test never issues a real invoice, read the `okruzenje` field from chapter 21 at startup, and keep test and production keys in separate configurations.

Binding a key to IP addresses (optional)

Each key can have a list of IP addresses it is accepted from. Fill it in under **Settings → Integrations**, when creating the key or later, with the shield icon next to the key. Accepted are an exact IPv4, a range in CIDR notation (for example `10.20.0.0/16`) and an exact IPv6, one per line or separated by commas. An empty field means no restriction.

Once the list is set, a request from another address gets **403** and never reaches invoice issuing, so a stolen key is worthless outside your network. Enter the address of the SERVER that calls the API, not of your computer.

The old `?apiKey=...` in the address NO LONGER works. It was removed on purpose after a security review: a key in the address leaks into server logs, the *Referer* header and browser history. If you still have an old integration with the key in the URL, move it to the header.

Storage rules

- Do not send the key by email, Viber or message. Paste it directly into the field where it is needed.
- Do not put it in code that goes to GitHub.
- Suspect it leaked → generate a new one, immediately. It takes 5 seconds.

90-day rotation

Keys rotate every 90 days. When a key is 76 days old, the company gets an email asking it to create a new one. After 90 days the key switches off by itself, but only if a newer key exists and the old one has received no calls for 14 days; a key that still receives calls stays on until you move the integration, so your sales do not stop. Settings shows the creation date and last use of each key.

5. Two-way connection: WooCommerce

Result: Tezga pulls orders itself, creates invoices and writes the invoice number back to the Woo order.

Step 1: create keys in WooCommerce

1. In the WordPress admin panel open **WooCommerce → Settings → Advanced → REST API**.
2. Click **Add key**.
3. **Description:** write “Tezga” (so you know what it is later).
4. **User:** choose an administrator account.
5. **Permissions:** choose **Read/Write**.

It must be Read/Write, not just Read: “Write” is what lets Tezga write the invoice number back to the order and record refunds.

6. Click **Generate API key**.

7. You get a **Consumer key** (starts with `ck_`) and a **Consumer secret** (starts with `cs_`). They are shown **only once**, so keep that window open until you have copied them.

Step 2: enter them in Tezga

1. **Settings** → **Integrations** → **E-commerce connection**, WooCommerce tab.

2. Fill in:

FIELD	WHAT GOES IN
Name	Anything, just for you (such as “My Woo shop”).
WordPress URL	The shop address, such as <code>https://my-shop.rs</code> . It must be https.
Consumer key	The one starting with <code>ck_</code> .
Consumer secret	The one starting with <code>cs_</code> . It is stored encrypted.
Status that triggers fiscalisation	See Step 3.

Save, then click **Check connection**. If it reports an error, see chapter 18.

Step 3: when the invoice is created

This chooses the moment a Woo order becomes an invoice:

CHOICE IN TEZGA	WOO STATUS THAT TRIGGERS IT
Processing or completed (default)	<code>processing</code> or <code>completed</code> , so as soon as it is paid.
Completed	Only <code>completed</code> , once you have actually closed the order.

If you sell goods shipped by courier and want the invoice only after delivery, take “Completed”. If you sell services or digital products, keep the default.

Step 4: PIB for companies (if you sell to legal entities)

For Tezga to know the customer is a company, the PIB must arrive from Woo. Tezga looks for it in the order's meta fields, under any of these names:

```
_billing_eu_vat_number  _billing_vat  vat_number  pib  _pib
```

If you use a PIB/VAT plugin, it almost certainly writes to one of these. If you have your own field, name it `pib`.

Without a PIB, Tezga issues a fiscal receipt (retail), which is correct for individual customers.

6. Two-way connection: Shopify

For Shopify this is the **only** way. A webhook does not work (ch. 3).

Step 1: create a custom app in Shopify

1. Shopify Admin → **Settings** → **Apps and sales channels**.
2. **Develop apps** → **Create an app**. Name it “Tezga”.
3. **Configure Admin API scopes** → enable reading orders and customers (`read_orders` , `read_customers`), and for writing back and refunds also `write_orders` .
4. **Install app**.
5. Take the **Admin API access token**, which starts with `shpat_` . It is shown **only once**.

Step 2: enter it in Tezga

FIELD	WHAT GOES IN
Shop domain	Your <code>.myshopify.com</code> domain, such as <code>my-shop.myshopify.com</code> . Not your custom domain.
Admin API token	The one starting with <code>shpat_</code> . It is stored encrypted.

Step 3: when the invoice is created

CHOICE	SHOPIFY CONDITION
Paid (default)	<code>financial_status = paid</code>
Fulfilled	<code>fulfillment_status = fulfilled</code>
Paid and fulfilled	Both conditions together.

Shopify “protected customer data”. Newer Shopify shops on the basic plan do not give apps the customer's email and data until you approve access to protected data in the app's settings. The consequence: the invoice is created, but it **cannot be emailed to the customer**, because the email never arrived. If you need that, approve the access in Shopify.

7. Connection settings (switches)

You see these on the connection card, below the access details. Worth going through once.

SWITCH	WHAT IT DOES	DEFAULT
Automatic order import	Every 30 minutes Tezga checks for new orders. This is what also catches orders a webhook would miss.	On
Automatic invoice creation	As soon as an order reaches the chosen status, the invoice is issued by itself. Off → the order waits for you in the Orders module and you click “Fiscalise”.	On
Automatic sending of invoices to the customer	After issuing, sends the invoice to the customer's email.	Off
Write a note back to the shop	The invoice number and status are visible in the order in the shop. This is the “two-way” part.	On
Show the delivery cost on the invoice	Adds delivery as a line item (also when free, as 0).	On
Catalogue auto-sync	Once a day, transfers products and customers from the shop to Tezga. It adds and updates, and deletes nothing .	Off

Email for error notifications

Enter an address you actually read. If an import fails (shop unavailable, key expired, PFR rejected), you get a message. Without this, an error stays silent until you notice it yourself.

Payment mapping

If your shop names payment methods differently from the Tax Administration, translate them here, line by line:

```
cod = Prenos
stripe = Kartica
paypal = Drugo bezgotovinsko
```

Why `cod = Prenos` and not cash: with cash on delivery the courier receives the money and remits it to your account. For the Tax Administration that is a **bank transfer**, not a cash sale. Tezga already does this automatically; this is only if you want to change it.

Tax mapping

The same, line by line, if the tax classes in your shop are named differently than in Tezga.

8. Webhook: the basics

This applies to all webhook platforms from chapters 9 and 10.

Address

```
POST https://pos.narbiz.com/api/integrations/order?provider=<PLATFORM>
```

Allowed values for `provider` :

`woocommerce` • `shopify` • `prestashop` • `opencart` • `magento` • `bigcommerce` • `custom`

If you omit `provider`, `custom` is used. An unknown value → error 400.

Required headers

```
X-Tezga-API-Key: <your key>  
Content-Type: application/json
```

Body signature: HMAC OPTIONAL

If you set a **webhook secret** in Settings, from that moment Tezga **rejects** every request without a valid signature. Add the header:

```
X-Tezga-Signature: sha256=<HMAC_SHA256(body, secret) in hex>
```

You sign **exactly the body you send**, byte for byte. The comparison is resistant to timing attacks.

Do I need this? The key alone is enough for most. Use HMAC if you send from a system you do not fully trust, or if you want proof that nobody changed the message on the way. If you do not set a secret, everything works as before and no signature is required.

What Tezga does when a request arrives

1. Checks `provider`. Unknown → **400**.
2. Checks the body size. Over 512 KB → **413**.
3. Checks the key. Missing or wrong → **401**.
4. Checks the key's rights: read-only, register terminal key, or an address outside the IP list → **403**.
5. Checks the subscription and plan (the API is in the Posao and Biznis plans) → **402**.
6. Checks the rate. Over the account limit (by default 120 requests per minute) → **429**. This is why a request with a bad signature also counts toward the limit.
7. Checks the signature, if a secret is set. Invalid → **401**.
8. Reads the JSON. Invalid → **400**.
9. Creates the document and reports it → **200** (or **202**, see ch. 15).

9. Webhook by platform

For each platform: where to set it up and what is sent. The headers are the same for all (ch. 8).

WooCommerce A BETTER WAY EXISTS

It works, but **we recommend the two-way connection** (ch. 5). If you really want a webhook:

1. **WooCommerce** → **Settings** → **Advanced** → **Webhooks** → **Add webhook**.
2. **Topic:** `Order created`.
3. **Delivery URL:** `.../api/integrations/order?provider=woocommerce`
4. Add a custom header `X-Tezga-API-Key` = your key.

Shopify DOES NOT WORK

A Shopify webhook does not send custom headers → the key cannot arrive → every request fails with 401.

Use the two-way connection (ch. 6).

PrestaShop

A webhook module (such as *Webhooks*) or the `actionValidateOrder` hook → POST the order to:

```
.../api/integrations/order?provider=prestashop
```

Tezga expects the usual PrestaShop shape: `id / reference`, `customer`, `order_rows[]`. Prices are read as **including VAT** (`unit_price_tax_incl`).

OpenCart

An event or custom module on *checkout order* → POST to:

```
.../api/integrations/order?provider=opencart
```

Careful with prices. OpenCart usually keeps prices **excluding VAT**, so Tezga adds 20% itself. If you sell at the 10% rate or are outside the VAT system, this will give a wrong amount, so **use the custom contract** (ch. 10), where you set the tax rate explicitly.

Magento / Adobe Commerce

The `sales_order_place_after` observer or a webhook → POST the order JSON to:

```
.../api/integrations/order?provider=magento
```

It reads `increment_id`, `items[]` and `customer_taxvat` (as the PIB). The line price is the amount actually paid for the row (`row_total` with tax, less the discount) divided by the quantity; `price_incl_tax` only when those fields are missing. The rate is read from `tax_percent`.

BigCommerce

The `store/order/created` webhook, then fetch the whole order with `products[]` and forward it to:

```
.../api/integrations/order?provider=bigcommerce
```

A note for all ready-made providers: they do “the best they can” with the standard data shape of that platform. If you have anything non-standard (your own tax rates, discounts, a PIB in an unusual field), **the custom contract is always the more accurate choice**, because you state explicitly what is what.

10. Custom contract (n8n / Zapier / Make / your own site)

The most precise way. You say exactly what is being sold, to whom and at what rate. Nothing is guessed.

```
POST https://pos.narbiz.com/api/integrations/order?provider=custom
X-Tezga-API-Key: <key>
Content-Type: application/json

{
  "externalId": "ORDER-1001",
  "kupac": { "naziv": "Pera Perić", "pib": null, "email": "p@x.rs" },
  "nacinPlacanja": "Kartica",
  "posaljiMejl": true,
  "stavke": [
    { "naziv": "Knjiga X", "sifra": "USL-001", "kolicina": 1,
      "cenaSaPdv": 1200, "poreskaStopa": "20" }
  ]
}
```

All fields

FIELD	REQUIRED	EXPLANATION
<code>externalId</code>	YES	Your unique order number, up to 200 characters. The idempotency key (ch. 14). It must not be empty.
<code>stavke</code>	YES	At least 1, at most 500.
<code>stavke[].naziv</code>	YES	Product name as it goes on the invoice (if the code is not recognised).
<code>stavke[].kolicina</code>	YES	Must be greater than zero.
<code>stavke[].cenaSaPdv</code>	YES	Price of one piece, including VAT . Not the line total.
<code>stavke[].sifra</code>	NO	If it exists in the Tezga catalogue → our name and rate are used, and the price and discount from the catalogue (except with <code>cenaIzZahteva: true</code>).
<code>stavke[].popustProcenat</code>	NO	Line discount in percent (0 to 100), as in the invoice editor. Not together with <code>stavke[].popustRsd</code> .
<code>stavke[].popustRsd</code>	NO	Discount on the whole row in dinars, not per piece: the line amount is quantity × <code>cenaSaPdv</code> minus this amount. Not together with <code>stavke[].popustProcenat</code> .
<code>stavke[].poreskaStopa</code>	NO	See ch. 13. Default <code>20</code> .
<code>stavke[].jedinicaMere</code>	NO	Unit of measure on the invoice (“kom”, “usluga”, “h”, “kg”), up to 20 characters. Default “kom”.
<code>kupac.naziv</code>	NO	Default “Webshop kupac”.
<code>kupac.pib</code>	NO	This field decides everything. PIB present → the customer is a company → <code>tok</code> applies. Absent → fiscal receipt.
<code>kupac.email</code>	NO	Required if you want <code>posaljiMejl</code> .
<code>kupac.adresa</code> , <code>kupac.mesto</code> , <code>kupac.postanskiBroj</code> , <code>kupac.maticniBroj</code>	NO	Full company details for the customer record (address, city, postal code, company number). When <code>kupac.pib</code> arrives with them, Tezga writes them to the record it keeps by PIB, so the document for the company carries the address and not just a bare name. An existing record is not overwritten.

nacinPlacanja	NO	<p>Free text. Default “Kartica”. It is printed on the invoice exactly as sent, and one of the seven official payment types goes to the Tax Administration. Recognised: gotovina (cash), kartica (card), ček (cheque), instant or IPS (instant payment), vaučer (voucher), prenos/virman/uplatnica/račun (bank transfer). “Drugo bezgotovinsko” and any unrecognised text go as other non-cash, which is correct for PayPal. Watch out for the word “račun”: “PayPal račun” would go as a bank transfer.</p> <p>Cash on delivery: send <code>cod</code> , <code>pouzeće</code> or <code>otkupnina</code> and Tezga records it as Prenos, the same as cash on delivery from WooCommerce. The reason: with cash on delivery the courier receives the money and remits it to your account, so for the Tax Administration it is a bank transfer, not a cash sale. Until 06.10.2026 this word fell through to “other non-cash” in the API, so the same sale got a different payment method depending on whether it came through a shop or the API.</p>
posaljiMejl	NO	Default <code>false</code> . Sends the document to <code>kupac.email</code> .
tok	NO	See ch. 11. Default <code>sef</code> . <code>avans</code> for an order paid in advance: ch. 24.
datumUplate	NO	Only with <code>tok:"avans"</code> : the day the payment landed (<code>YYYY-MM-DD</code>). Default today.
posaljiNaSef	NO	With <code>tok:"sef"</code> , send the eFaktura to SEF right away. Default <code>false</code> (waits for manual sending).
tipDokumenta	NO	<code>racun</code> (default) or <code>predracun</code> . See ch. 12.
pozivNaBroj	NO*	* Required with <code>predracun</code> . See ch. 12.
model	NO	Reference number model, such as <code>97</code> .
fiskalizujPredracun	NO	See ch. 12. Default <code>false</code> .
popustProcenat	NO	Discount on the whole invoice in percent. Not together with <code>popustRsd</code> . See “Discounts” below.
popustRsd	NO	Discount on the whole invoice in dinars. Not together with <code>popustProcenat</code> .
cenaIzZahteva	NO	<code>true</code> : for a line whose code exists in the catalogue, the price and discount from the request apply; the link to the product, name, unit of measure and tax rate stay from the catalogue, and the catalogue discount is not added. Default <code>false</code> : the catalogue applies, as before.

Discounts

A discount is sent as a number, never by reducing the price: that way the invoice shows both the price and the discount, the same as an invoice from the app. Each level (line, whole invoice) takes **one** field: percent or

dinars.

- **A line discount** changes only that line. `popustRsd` is the amount for the whole row.
- **A discount on the whole invoice** is not written as a separate row. Tezga distributes it across the lines in proportion to their amounts (after line discounts), so the base and VAT at each rate are correct even when an invoice has goods at 20% and at 10%. The rounding remainder (paras) goes to the lines with the largest discarded part, so the discounts add up to exactly the amount sent.
- **A code from the catalogue:** the discount applies only with `"cenaIzZahteva": true`. Without it Tezga takes the price and discount from the catalogue and rejects a request with a discount (**422**), so that your discount is not silently ignored.

```
{
  "externalId": "WEB-2044",
  "cenaIzZahteva": true,
  "popustRsd": 500,
  "stavke": [
    { "naziv": "Jakna", "sifra": "JAK-01", "kolicina": 1, "cenaSaPdv": 12000,
      "popustProcenat": 10 },
    { "naziv": "Knjiga", "kolicina": 2, "cenaSaPdv": 1100, "poreskaStopa": "10",
      "popustRsd": 200 }
  ]
}
```

Jacket: $12,000 - 10\% = 10,800$. Books: $2 \times 1,100 - 200 = 2,000$. Total 12,800, so the 500 dinar invoice discount is split $10,800 : 2,000$ (421.88 and 78.12). Invoice: 12,300.00.

422 is returned for: both discount fields at the same level, a percentage over 100, a discount larger than the line or invoice amount, and a discount with a catalogue code without `cenaIzZahteva: true`. Nothing is issued in that case.

The most common mistake: `cenaSaPdv` is the price **per piece**, including VAT. If you send the line total, the invoice will be wrong by a factor of the quantity.

11. Flow (tok): what exactly gets created

The `tok` field answers the question “which document do I want”. It applies **only when the customer has a PIB**.

TOK	WHAT GETS CREATED	WHEN
<code>sef</code> default	eFaktura. Not fiscalised. With <code>posaljiNaSef:true</code> it goes to SEF right away; otherwise it waits for you to send it manually.	A sale to a company outside a retail outlet (services, wholesale, notary fees).
<code>fiskalni</code>	Fiscal receipt with the customer's PIB on it.	A sale to a company in a retail outlet ; then it is a retail sale and no eFaktura is issued.
<code>fiskalni_i_sef</code>	Both. First a fiscal receipt, then the same document goes to SEF with a reference to the fiscal receipt number.	When the customer explicitly asks for an eFaktura along with the fiscal receipt.
<code>avans</code>	Advance invoice (АВАНС ПРОДАЈА) for the order total, with or without a PIB. The final invoice is created when you close the advance (ch. 24).	An order paid in advance, with delivery later.

Rules Tezga applies by itself

- **No PIB → always a fiscal receipt**, whatever you sent in `tok` (except with `tok:"avans"`, which issues an advance invoice). An eFaktura without a PIB does not exist.
- `tok:"fiskalni_i_sef"` without a PIB → an **error**, not a silently wrong document.
- **Public sector customer** (has a JBKJS) → the fiscal flow is forbidden, an eFaktura is mandatory. Tezga rejects with a clear message.
- **The company is not subject to fiscalisation** → the fiscal flow is rejected, use `sef`.

Why a fiscal receipt and an eFaktura are usually not both: the Law on Electronic Invoicing (Art. 3) exempts retail sales from the eFaktura obligation. Simply put: if a fiscal receipt was issued for a retail sale, an eFaktura is **not** issued. That is why `fiskalni_i_sef` is not the default; use it only when you know you need it.

12. Pro forma → payment → invoice

For everything paid by **bank transfer**: send a pro forma, and Tezga closes it by itself when the money arrives. With no other call from your side.

What it looks like

```
{
  "externalId": "NALOG-118",
  "tipDokumenta": "predracun",
  "tok": "sef",
  "model": "97",
  "pozivNaBroj": "97-118-2026",
  "kupac": { "naziv": "Firma D00", "pib": "115468360", "email": "f@doo.rs" },
  "nacinPlacanja": "Prenos",
  "posaljiMejl": true,
  "stavke": [ { "naziv": "Notarska taksa", "kolicina": 1, "cenaSaPdv": 30240 } ]
}
```

What happens next, by itself

- 1 The pro forma is issued. With `posaljiMejl: true` it goes to the customer right away, with an **IPS QR code**, so they pay by scanning from mobile banking, with the reference number already correct.
- 2 The bank statement arrives at the register's email (in the morning, from the bank).
- 3 Tezga matches the payment with the pro forma by reference number.
- 4 The final document is created by itself, according to the `tok` the pro forma remembered: eFaktura to SEF, fiscal receipt, or both. It is linked to the pro forma and to the payment.

The key idea: here `tok` does not mean “what happens now” but “**what is created when the money arrives**”. That is why a notary sends `tok:"sef"` and a fiscalising shop sends `tok:"fiskalni"`: same endpoint, same shape, different outcome.

Why `pozivNaBroj` is required

It is the only reliable way to tie a payment to the right pro forma. Without it Tezga would have to guess by amount, and would match only if there is **exactly one** unmatched payment with that amount. With recurring amounts, such as fees, memberships and subscriptions, this either stays silent (so your invoice is not created) or matches someone else's payment (so a **wrong** invoice is created). That is why the API would rather reject a pro forma without a reference number than take the risk.

Fiscalising a pro forma ALMOST NEVER NEEDED

By default a pro forma is **not** fiscalised, which is both the simplest option and always lawful:

- A pro forma is **not** a fiscal receipt and **does not have to** go through a fiscal device: Rulebook on types of fiscal invoices, Art. 2(4)(3) and (10).
- In the `sef` flow a fiscal Proforma **does not even exist as a concept** (it is defined as a document before a *retail sale*), so Tezga rejects it.
- If you still want a Proforma with a QR code, add `"fiskalizujPredracun": true`, possible only with a fiscal flow. The final invoice then **must** carry the reference number of that Proforma (Art. 11(1)(4)); Tezga does this by itself.

13. Tax rates

WHAT YOU SEND	WHAT TEZGA UNDERSTANDS
"20" , "pdv20" , "20%" or "0.2"	VAT 20% (general rate)
"10" , "pdv10" , "10%" , "10,00" or "0.1"	VAT 10% (special rate)
"0" , "oslobodjeno" (also "oslobodeno"), "exempt" , "zero" , "zerorate"	No VAT / exempt (label A)
empty or omitted	VAT 20%
anything else	On <code>/order</code> (custom): error 400 with this list, no invoice is created. When closing an advance: 422 . In the catalogue (<code>products</code>): an error for that item, the response is 200. For WooCommerce, Magento, BigCommerce and PrestaShop the rate is read from the tax amount, for Shopify from the rate the platform sends, and OpenCart is always 20%.

Why an unknown value never becomes exempt: on purpose. If it were the other way round, a mistake in your integration would silently issue invoices without tax, a tax offence discovered only during an audit. So an unknown value returns an error right away, and an omitted one means 20%. **Tezga never assumes an exemption.**

If you are outside the VAT system (flat-rate taxpayer)

You do not have to send or set anything. When a company is marked as not liable for VAT, Tezga automatically treats **all** line items as exempt (label A), whatever arrives in `poreskaStopa` .

If you get a tax rate error: the actual fiscal device (PFR) dictates which rates exist on your account. If a product has a rate that is not in your tax profile at the Tax Administration, Tezga reports this clearly instead of issuing a wrong invoice. It is solved by aligning the rates with the Tax Administration.

14. The response you get

```
{
  "ok": true,
  "status": "kreirano",
  "documentId": "b3f1...",
  "broj": "RAC2026000123",
  "tip": "promet",
  "statusDokumenta": "izdat",
  "ukupnoSa": "510.00",
  "nacinPlacanja": "Gotovina",
  "fiskalni": {
    "pfrBroj": "AB12CD34-AB12CD34-123",
    "pfrVreme": "2026-08-21T10:00:00.000Z",
    "brojacRacuna": "123/130ΠΠ",
    "esirBroj": "1597/1.1",
    "verifikacioniUrl": "https://suf.purs.gov.rs/v/?vl=...",
    "qrPayload": "https://suf.purs.gov.rs/v/?vl=...",
    "referentniBroj": null,
    "referentnoVreme": null
  },
  "javniUrl": "https://pos.narbiz.com/r/7f3a...",
  "javniPdfUrl": "https://pos.narbiz.com/r/7f3a.../pdf",
  "verifikacioniUrl": "https://suf.purs.gov.rs/v/?vl=...",
  "pdfUrl": "/racuni/b3f1.../pdf"
}
```

FIELD	WHAT IT IS
<code>ok</code>	<code>true</code> = all went through. <code>false</code> = the document was issued, but fiscalisation failed.
<code>status</code>	<code>kreirano</code> = new document. <code>postojalo</code> = this <code>externalId</code> has already been processed, the existing one is returned. <code>greska</code> = see <code>fiskalGreska</code> .
<code>fiskalGreska</code>	Present only when the PFR did not accept. The document is issued and waits for a retry : do not send it again; Tezga retries by itself every 10 minutes (up to 12 times in 48 hours) and reports by webhook when it succeeds.
<code>sef</code>	Outcome of sending to SEF, an object <code>{ status, poruka }</code> . Absent if sending was not requested. <code>sef.status: "nije_povezan"</code> = the company has no SEF key of its own, and the document waits for manual sending.
<code>documentId</code>	Internal document ID in Tezga.
<code>broj</code>	Document number in Tezga (such as <code>RAC2026000123</code> , <code>EF2026000045</code>).
<code>tip</code> , <code>statusDokumenta</code>	Document type (<code>promet</code> , <code>racun</code> , <code>predracun</code> , <code>refundacija</code> ...) and its status (<code>izdat</code> , <code>storniran</code>). The document status is separate from the response's <code>status</code> , which says what happened to the request.
<code>ukupnoSa</code> , <code>nacinPlacanja</code>	Amount including VAT as issued, and the payment method. A refund is negative.
<code>fiskalni</code>	Mandatory elements of a fiscal receipt , for systems that print themselves: <code>pfrBroj</code> (invoice number assigned by the PFR), <code>pfrVreme</code> (PFR time, ISO 8601 in UTC; printed in the Belgrade time zone with seconds), <code>brojacRacuna</code> (such as <code>123/130ПП</code>), <code>esirBroj</code> (ESIR approval number), <code>verifikacioniUrl</code> and <code>qrPayload</code> (QR code content). For a refund, <code>referentniBroj</code> and <code>referentnoVreme</code> carry the PFR number and time of the original invoice. <code>null</code> = the document is not fiscalised (eFaktura, or the PFR has not gone through yet).
<code>javniUrl</code> , <code>javniPdfUrl</code>	Public invoice link and its PDF, working without an account . The same link a cashier creates when sharing an invoice with a customer: your system can open it, print it or forward it to a guest. The PDF follows the company's print format (A4 or thermal roll, Settings → Print). The link is stable: the same document always gives the same URL.
<code>verifikacioniUrl</code>	Same as <code>fiskalni.verifikacioniUrl</code> ; it stays at the top level for integrations that already read it.
<code>pdfUrl</code>	Legacy field: a PDF inside the app, requires login. New integrators use <code>javniPdfUrl</code> .

PDF without login: `javniPdfUrl`. Pull it from n8n, open it in a CRM or forward it to a guest. If the customer should get the PDF by email, send `"posaljiMejl": true` and `kupac.email`, and Tezga sends it. Whoever knows the link can see the invoice: keep it like the invoice itself and do not publish it.

Idempotency: retry freely

The same `externalId` with the same `provider` never creates a second document. If your connection drops and you do not know whether it went through, **send the same request again**. You get

`status: "postojalo"` and the number of the document already created. This also protects against a duplicate webhook.

So `externalId` must be truly unique. If you send the same `externalId` for two different sales, the second one **will not be created**; you get the first one back. Use your order number, not something that repeats.

15. Errors and HTTP codes

CODE	MEANS	WHAT TO DO
200	All went through.	Nothing.
202	Document issued, fiscalisation failed (<code>ok: false</code>).	Do not send again. The document exists. Tezga retries fiscalisation by itself every 10 minutes, at most 12 times in 48 hours, and reports <code>racun.fiskalizovan</code> by webhook when it succeeds (chapter 22). The reason is in <code>fiskalGreska</code> .
404	The document does not exist in your company .	Check <code>documentId</code> / <code>externalId</code> . Someone else's document gets the same response as a non-existent one.
409	The rules do not allow the action (cancellation/refund).	Read <code>error</code> : the invoice is not "izdat", it is partly refunded so it cannot be cancelled, or there is nothing left to refund.
422	The request failed field or rule validation.	<code>fields</code> lists what is missing (such as <code>kupacIdBroj</code> for a refund without a customer). On <code>order</code> : two discounts at the same level, a discount larger than the amount, or a discount with a catalogue code without <code>cenaIzZahteva</code> (ch. 10). Fix the body and send again.
400	The body is not valid JSON, unknown <code>provider</code> , or the data is invalid (no items, no <code>externalId</code> , pro forma without a reference number...).	Read <code>error</code> ; it says exactly what is missing. Fix the body and send again.
401	No key, wrong key, or invalid signature.	Check that <code>X-Tezga-API-Key</code> really arrives as a header . If you use HMAC, check that you sign exactly the body you send.
402	The company's subscription is not active, or its plan does not include the API and web shop connection (chapter 4: Posao or Biznis, Enterprise, or a company that used the connection before 16.09.2026).	Pay at <code>/uplata</code> in the register. For the plan: upgrade to Posao or Biznis in Settings → Plan. Do not retry until the cause is removed; the exact reason is in <code>error</code> .
413	Body over 512 KB (order, products, customers), 64 KB (refund, advance) or 32 KB (copy).	You are sending something that is not an order (images? the whole catalogue?). Send only the items.
403	The key has an IP list and the request came from another address (or the source IP is not readable to us). The same when a read-only key writes, or when a register terminal key is used for the API.	Add the server address in Settings → Integrations, shield icon next to the key, or empty the list to remove the restriction.

429	Over the account limit (by default 120 requests per minute).	Wait and try again. If you hit this regularly, your integration is probably stuck in a loop.
500	An error on our side while checking the signature.	Let us know.

Messages you are most likely to see

Error messages come in Serbian. This is what they mean:

MESSAGE	WHAT IT ACTUALLY MEANS
<i>Nedostaje X-Tezga-API-Key</i>	The header did not arrive at all. This is expected with a Shopify webhook, see ch. 3.
<i>Zahtev dolazi sa IP adrese koja nije na listi ovog ključa</i>	The key is valid but bound to IP addresses. Check which address your server actually uses to reach the internet; it is not always the one you expect.
<i>Neispravan API ključ</i>	The key arrived but matches no company. It was probably regenerated, so the old one no longer applies.
<i>Predračun zahteva "pozivNaBroj"</i>	A pro forma needs a reference number. See ch. 12.
<i>Tok "fiskalni_i_sef" zahteva kupac.pib</i>	An eFaktura is not possible without a PIB.
<i>Firma nije obveznik fiskalizacije</i>	You asked for a fiscal flow, but the company is exempt by regulation. Use <code>tok:"sef"</code> .
<i>Kupac je javni sektor (ima JBKJS)</i>	A state customer must get an eFaktura; a fiscal receipt does not cover it.
<i>Poreska stopa X% nije u poreskom profilu V-PFR-a</i>	See ch. 13, the last box.
<i>Narudžbina ima previše stavki (maks 500)</i>	Something is wrong with your mapping.
<i>Porudžbina se već obrađuje (paralelni zahtev)</i>	You sent the same <code>externalId</code> twice at the same moment. Wait a second and ask again; you will get <code>postojalo</code> .

Where to see what happened: every order from an integration is written to the **Log** in the register, with the error message if there was one. The **Orders** module shows all received orders and their status.

16. Limits

WHAT	HOW MUCH	WHY
Requests per minute (issuing, cancellation, refund, writing products and customers)	120 per company, adjustable	A normal shop never gets near this. It protects against loops and abuse. This is the DEFAULT value: the limit is tied to the account, so for announced peak traffic (for example the start of ticket sales) it is raised to an agreed value, with no change to your code.
Reading (document status, tax labels)	240 per company, adjustable	Every response after the key and plan checks carries <code>X-RateLimit-Limit</code> , <code>X-RateLimit-Remaining</code> and <code>X-RateLimit-Reset</code> (seconds until the new window), and a 429 also carries <code>Retry-After</code> .
Body size	512 KB	An order is small. A larger body means something is being sent that should not be.
Line items per order	500	A real order does not have hundreds of different products.
Products or customers per call (products, customers)	500	Send a larger list in several calls; repeating the same call is safe.
Pull frequency (two-way connection)	30 min	A compromise between freshness and the load on your shop.
Catalogue auto-sync	Once a day	A catalogue does not change every minute.
API key rotation	90 days	Email on day 76; the old key switches off only when a newer one exists and it has received no calls for 14 days (chapter 4).
Outgoing webhook addresses	5 per company	Each with its own secret and its own choice of events (chapter 22).

What Tezga does NOT do

- **It does not move money.** A refund is recorded both fiscally and in the shop, but you make the actual repayment to the customer, through the bank or payment system.
- **It does not change a fiscalised invoice.** There is no editing or deleting: every correction is a new document (refund) referring to the original, chapter 20.
- **It does not set the fiscal receipt number.** The PFR assigns it. Your order number is kept as a reference number; it is used for searching and for linking payments.
- **It does not track stock.** Tezga is an invoicing and fiscalisation system, not a warehouse solution.
- **It does not delete anything** when syncing the catalogue or when writing products and customers through the API (chapter 26); it only adds and updates.

17. n8n step by step

The same procedure applies to Zapier and Make; only the nodes have different names.

1. In your workflow, after the trigger (such as “document signed”, “order received”), add an **HTTP Request** node.
2. **Method:** `POST`
3. **URL:** `https://pos.narbiz.com/api/integrations/order?provider=custom`
4. **Headers** → add two:

```
X-Tezga-API-Key: <your key>
Content-Type: application/json
```
5. **Body** → choose “JSON” and paste the shape from chapter 10. Pull the values (service name, amount, PIB) from the previous step with expressions such as `{{ $json.field }}`.
6. Tie **externalId** to your unique number, such as `{{ $json.order_number }}`. Never random: that loses the protection against duplicates.
7. For a legal entity fill in `kupac.pib`; for an individual leave it `null`.
8. Want the register to send the document to the customer by itself? Add `"posaljiMejl": true` and fill in `kupac.email`.
9. Charging by bank transfer? Do not make two nodes: send a **pro forma** (ch. 12) and let the register close it when the statement arrives.

If you use the HMAC signature

Add a **Crypto** node before the HTTP Request: algorithm `SHA256`, type `HMAC`, secret = your webhook secret, input = exactly the body you send. Put the result in the header as `sha256=<hex>`.

Handling the response

After the HTTP Request add an **IF** node on `{{ $json.ok }}`:

- `true` → write `broj` and `verifikacioniUrl` back to your system.
- `false` → the document **was** issued but not fiscalised. **Do not repeat the call**. Send yourself a notification with `fiskalGreska` and resolve it in the register.

18. Troubleshooting

SYMPTOM	CAUSE AND SOLUTION
Everything fails with 401, and the key is correct	The key most likely does not arrive as a header . With a Shopify webhook that is impossible (ch. 3). Also check that you have not left the old <code>?apiKey=</code> in the URL; it no longer works.
The invoice has a different price than the shop	In a custom call, the product code exists in the Tezga catalogue with a different price. Send <code>"cenaIzZahteva": true</code> or fix the price in the catalogue. Web shop connections always fiscalise the price the customer paid.
Orders do not arrive (two-way connection)	Click "Check connection". Most often: the key expired or was deleted in the shop, the key is not Read/ Write , the shop is not reachable from outside, or the order has not yet reached the chosen status (ch. 5, step 3).
A "Račun" was created and I wanted a fiscal receipt	The customer has a PIB, and <code>tok</code> defaults to <code>sef</code> . Send <code>"tok": "fiskalni"</code> .
A fiscal receipt was created and I wanted an eFaktura	The PIB did not arrive. Without a PIB it is always fiscal. Check the meta field (ch. 5, step 4).
The eFaktura did not go to SEF	Either you did not send <code>posaljiNaSef:true</code> , or the company has no SEF key of its own (the response says <code>nije_povezan</code>). The key is entered under Settings → Government services → SEF • e-Invoices , and the Check connection button on the same page tells you right away whether it works.
The invoice did not reach the customer by email	Check that you sent <code>posaljiMejl:true</code> and that <code>kupac.email</code> is not empty. With Shopify a common cause is "protected customer data" (ch. 6). The outcome of sending is visible in the Log .
The payment arrived but no invoice was created	The reference number does not match, or there are several unmatched payments with the same amount, so Tezga deliberately skipped instead of guessing. Open Bank → Incoming payments and reconcile manually.
I get "postojalo" and expect a new invoice	You are repeating the same <code>externalId</code> (ch. 14).
Duplicate invoices	This should not be possible. If it happens, you are almost certainly sending a different <code>externalId</code> for the same order (such as a random one). Tie it to your actual number.

The first place to look when something does not work: the **Log** in the register. Every order from an integration is written there with the exact error message. Right next to it is the **Orders** module, where you see the status of every received order.

19. Document status (GET documents)

The same JSON as in chapter 14, for any document of your company. It lets you check after a **202** (PFR failed) whether the invoice has been fiscalised in the meantime, and pick up the fiscal data and PDF link whenever you need them.

```
GET https://pos.narbiz.com/api/integrations/documents/<documentId>
X-Tezga-API-Key: tzg_...
```

Response: `ok: true` + all fields from chapter 14 (`statusDokumenta` , `fiskalni` , `javniPdfUrl` ...). A non-existent or foreign document: **404**. Limit: 240 requests per minute per company.

After a 202 do not send the order again. The document exists. Tezga retries fiscalisation by itself every 10 minutes (up to 12 times in 48 hours) and reports `racun.fiskalizovan` , and your system also sees it here: `fiskalni` stops being `null` .

Or by your order number:

GET `.../api/integrations/documents?externalId=NALOG-1042&provider=custom` (without `externalId` or with an unknown provider: **400**).

20. Cancellation and refund (POST refund)

A fiscalised invoice is **not changed or deleted**. When a waiter makes a typing mistake or a guest returns a drink, a **new document, a refund**, is created, referring to the original. This call does exactly what the “Cancel invoice” and “Refund” buttons do in the register.

```
POST https://pos.narbiz.com/api/integrations/refund
X-Tezga-API-Key: tzg_...
Content-Type: application/json
```

Two kinds

VRSTA	WHEN	WHAT GETS CREATED
storno	An invoice issued by mistake (wrong table, entered twice).	A refund of the full amount , with no money returned. Nothing is asked of the customer: identification comes from the original (B2B), and for a retail invoice the seller's PIB is entered, per the Technical Instructions (the company must have its PIB entered). The original becomes storniran only when the refund goes through at the Tax Administration. An invoice that is partly refunded cannot be cancelled (409).
refundacija	Money returned to the guest, for the whole invoice or a part.	A refund of the whole invoice, or only of the listed stavke . For a cash repayment the Tax Administration requires customer identification: if the original has no customer, send kupacIdBroj and kupacIdTip (such as a foreigner's passport), otherwise 422 . Repayment of a non-cash payment (card, bank transfer, PayPal) does not require it.

Body

```
{
  "documentId": "b3f1...",
  "vrsta": "refundacija",
  "kupacIdTip": "30",
  "kupacIdBroj": "X1234567",
  "stavke": [ { "redosled": "0", "kolicina": 1 } ],
  "razlog": "guest returned a drink"
}
```

FIELD	MEANING
documentId	The ID from the POST order response. Or instead of it:
externalId (+ provider)	Your order number from POST order; provider is the same as then (default custom). Tezga finds the invoice by itself.
vrsta	storno or refundacija .
stavke	Only with refundacija . redosled is the line's ordinal on the original (the first is "0"), kolicina how much is returned. More than sold is lowered to the amount sold; what is already refunded is not counted again. Without stavke = the whole invoice.
kupacIdBroj	The customer's document number, only for refundacija when the original has no customer and the money is returned in cash. It is not needed for a non-cash repayment; if you send it anyway, it must have a valid form (otherwise 422) and it is recorded. The customer from the original always takes precedence. It is not read for a cancellation.
kupacIdTip	Document code from the Tax Administration's list. Default "20" (ID card), so old calls work as before. For 10 , 11 and 20 the number is digits only, for the others letters and digits (lowercase is converted to uppercase), with no spaces or hyphens. Code, colon and number together at most 20 characters. An unknown code is 422 .
razlog	Optional reason, up to 500 characters. Written on the refund and in the log.

Customer document codes

KUPACIDTIP	DOCUMENT
20	ID card (Serbian resident), 6 to 12 digits
23	Passport of a Serbian citizen
30	Passport of a foreigner
22	Personal number (EBS) of a foreigner with registered residence
31	Diplomatic ID or ID card
32 to 35	ID card of North Macedonia, Montenegro, Albania, Bosnia and Herzegovina
36	ID card of the EU, Switzerland, Norway or Iceland
10 , 11 , 12	PIB (9 digits), JMBG (13 digits), PIB:JBKJS
13 , 14 , 15 , 16 , 21 , 40	Pensioner card, PIB and JMBG of a farm, farm registration number (BPG), refugee ID, foreign tax number (TIN)

When the customer's document is required. The Tax Administration requires customer identification when money is returned **in cash**, on the spot. When it is returned to a card, to an account or by other non-cash means (PayPal), the customer's document is not required, so Tezga issues the refund without `kupacIdBroj`. The repayment method is the payment method of the original invoice; with a mixed payment that includes cash, the document is required. If the invoice was simply issued by mistake and no money is returned, use `storno`, where the seller's PIB is entered.

Response

```
{
  "ok": true,
  "status": "kreirano",
  "vrsta": "storno",
  "original": { "documentId": "b3f1...", "broj": "RAC2026000123", "statusDokumenta": "storniran" },
  "refundacija": { ...same shape as in chapter 14, fiskalni.referentniBroj = PFR number of the original... }
}
```

- **200**, `status: "kreirano"`: done. `"postojalo"` (refund only): this invoice was already refunded; that refund is returned and no second one is created. A repeated cancellation returns 409.
- **202**, `ok: false`, `fiskalGreska`: the refund was created but the PFR did not accept it. The original stays `izdat`. For a **cancellation or full refund**, send the same request again when the PFR is back: Tezga sends the same refund and does not create a new one. For a **refund by line items** do NOT repeat the request (the requested quantities are already recorded as refunded): Tezga retries fiscalisation by itself every 10 minutes, and you read the state through `GET documents/<refundacija.documentId>`.

- **409:** rules (not issued, partly refunded, nothing left, or the company has no PIB entered when cancelling a retail invoice). **422:** a field or `kupacIdBroj` is missing. **404:** the original is not in your company.
- An invoice for which the PFR failed at issuing (not fiscalised) does not exist at the Tax Administration: a cancellation only marks it, `refundacija: null`.

We do not move money. A refund is the fiscal record of a repayment. Returning the money to the guest (cash, card) is up to the merchant.

21. Tax labels and environment (GET tax-labels)

Which rates and labels your account actually uses, and in which environment it fiscalises. Call it once when the integration starts, so you do not have to guess.

```
GET https://pos.narbiz.com/api/integrations/tax-labels
X-Tezga-API-Key: tzg_...
```

```
{
  "ok": true,
  "obveznikPdv": false,
  "obveznikFiskalizacije": true,
  "okruzenje": "produkcija",
  "oznake": [
    { "stopa": "pdv20", "oznaka": "G", "procenat": 20, "koristiSe": false },
    { "stopa": "pdv10", "oznaka": "E", "procenat": 10, "koristiSe": false },
    { "stopa": "oslobodjeno", "oznaka": "A", "procenat": 0, "koristiSe": true }
  ],
  "napomena": "Firma NIJE u sistemu PDV-a..."
}
```

Not in the VAT system? Tezga records every line item as exempt (0%), whatever rate you send. You do not have to calculate anything.

The `okruzenje` field

VALUE	WHAT IT MEANS
<code>produkcija</code>	Real fiscal receipts at the Tax Administration, with your company's element.
<code>sandbox</code>	The Tax Administration's official test environment with a test element. The invoice goes through a real PFR but is not real; the verification link leads to the subdomain <code>sandbox.suf.purs.gov.rs</code> .
<code>simulacija</code>	An account without a security element. The invoice is simulated locally, the PFR number starts with <code>MOCK</code> , and nothing goes to the Tax Administration.

A guard in your code: a test configuration may run only when `okruzenje` is not `produkcija`. The key does not tell you this, because test and production keys look the same (ch. 4).

22. Outgoing notifications (webhook)

Instead of polling the API, you can give us an address and we notify you. Addresses are added in the register: **Settings** → **Integrations** → **Outgoing notifications**.

EVENT	WHEN
<code>racun.fiskalizovan</code>	A document accepted by the Tax Administration: invoice, advance, copy or a refund fiscalised later.
<code>racun.greska</code>	An invoice from <code>POST order</code> was issued, but the PFR did not accept it (the same case as HTTP 202). Advance, copy and refund do not send this event.
<code>refundacija.izvršena</code>	Refund issued (full or by line items).
<code>storno.izvršen</code>	Invoice cancelled.

`podaci` always also carries your `externalId` and `provider` when the document came from an order (POST order). For `refundacija.izvršena` and `storno.izvršen` that is the order number of the **original** invoice, and the refund's `documentId` is in `podaci.refundacija.documentId`. Rely on `externalId` for matching; do not read it as our `documentId`.

What arrives at your address

```
POST your-address
X-Tezga-Event: racun.fiskalizovan
X-Tezga-Delivery: 7c1f...           (id of this message, for your idempotency)
X-Tezga-Signature: sha256=...      (HMAC of the whole body, with your secret)

{
  "dogadjaj": "racun.fiskalizovan",
  "vreme": "2026-08-21T10:00:00.000Z",
  "tenantId": "...",
  "podaci": { ...for an invoice the shape from chapter 14, for a cancellation or refund
the response from chapter 20... }
}
```

Checking the signature

Calculate `HMAC-SHA256` over the **whole request body**, with the secret shown to you when you added the address, and compare it with the header. If it does not match, discard the message.

- **Reply 2xx** as soon as you receive it. Anything else (or silence) counts as a failure.
- **We retry** after 1, 5, 15, 60 and 360 minutes. After the last attempt the message stays marked as undelivered and is visible in Settings.

- **The same event can arrive twice** (for example if your response got lost on the way back). Use `X-Tezga-Delivery` for idempotency.
- The address must be public **https**; local and private addresses are rejected.

Later fiscalisation is reported by webhook. When the PFR fails, Tezga retries fiscalisation by itself every 10 minutes (up to 12 times, for documents younger than 48 hours) and, when it succeeds, sends `racun.fiskalizovan`. The same applies when a cashier clicks “Fiscalise again” in the register. The state can always be read through `GET /documents/:id` as well (chapter 19).

23. OpenAPI description

For tools: `https://pos.narbiz.com/api/openapi.json?lang=en` (no key, OpenAPI 3.1). Import it into Postman or Swagger UI, or generate a client from it. A readable list of endpoints is at `tezgafiskal.com/en/fiscalization-api/reference`.

The description covers all routes in this guide, the shape of bodies and responses, error codes and limit headers. The guide you are reading is for people; the description is for machines. It is written by hand, and a test compares it with the schemas in the code, so it cannot silently go out of date; the response is cached for up to 5 minutes. The version in `info.version` grows when a field or route is added.

24. Advance invoice (POST advance)

For when you receive an advance payment and deliver later. The law requires an exact order and Tezga carries it out by itself; your system sends two calls.

Step 1: advance invoice (АВАНС ПРОДАЈА)

```
POST https://pos.narbiz.com/api/integrations/advance
X-Tezga-API-Key: tzg_...

{
  "externalId": "NALOG-118",
  "kupac": { "naziv": "Firma D00", "pib": "115468360" },
  "placeniAvans": 30000,
  "ugovoreniIznos": 120000,
  "datumUplate": "2026-08-20",
  "poreskaStopa": "pdv20"
}
```

The amount ACTUALLY received is fiscalised (`placeniAvans`), not the contract value, and the invoice carries **the date the payment landed** (`datumUplate` is mandatory). For a bank transfer the advance invoice is issued no later than the next working day.

Response: `avansId`, the advance `broj` and `avansniRacun` in the same shape as in chapter 14. The same `externalId` returns the existing advance (`status: "postojalo"`), never a second one.

- `poreskaStopa` here accepts only `"pdv20"`, `"pdv10"` or `"oslobodjeno"` (default `"pdv20"`); `"20"` or `"10%"` return **422**. The final invoice's line items in step 2 accept all forms from ch. 13.
- The body is strict: an unknown field returns **422**, and `kupac.email` must be a valid address.
- When the PFR rejects the advance invoice at issuing, the reason is in `error`.

Step 2: closing the advance

```
{
  "zatvori": true,
  "avansId": "...",          // or the advance's "externalId"
  "punIznos": 120000
}
```

Tezga then issues two documents, in this order:

- **АВАНС РЕФУНДАЦИЈА**: technically closes the advance, for exactly the amount fiscalised on the advance invoice. Internal, not sent to the customer.
- **ПРОМЕТ ПРОДАЈА**: the final invoice for the full amount, referring to the refund. This way the advance and its refund net to zero, and the sale remains the only one recorded (no double VAT).

If the refund does not pass the PFR, the final invoice is **NOT** issued (you get 409). Otherwise the final one would go out with no link to the advance, and VAT would be calculated twice. Try again when the connection with the Tax Administration is back.

The full amount must not be lower than the fiscalised advance. A repeated closing returns the existing final invoice (`status: "postojalo"`).

Advance from an order (POST order, `tok: "avans"`)

A web order paid in advance, with goods or a service coming later: send it to `/order` like any other, with `"tok": "avans"`. Tezga issues the advance invoice the same way as above, and you do not need step 1.

POST `https://pos.narbiz.com/api/integrations/order?provider=custom`

```
{
  "externalId": "WEB-5521",
  "tok": "avans",
  "datumUplate": "2026-09-16",
  "nacinPlacanja": "Kartica",
  "popustRsd": 1000,
  "stavke": [
    { "naziv": "Bicikl", "kolicina": 2, "cenaSaPdv": 25000 },
    { "naziv": "Kaciga", "kolicina": 1, "cenaSaPdv": 3000 }
  ]
}
```

- **The advance amount and contract value** are the order total after all discounts, calculated the same way as an invoice from an order (in the example 52,000.00).
- **One tax rate.** An advance invoice carries one rate. An order with line items at 20% and at 10% returns **422**: issue such an advance per rate through `/advance` . A 0 dinar line item (free delivery) does not matter.
- **The response** has the same shape as for an invoice (document at the root, ch. 14), plus `avansId` and `avansBroj` . The same `externalId` repeated returns the same advance, never a second one.
- It does not go with `tipDokumenta:"predracun"` or `posaljiNaSef` (**422**). If the same `externalId` was already issued as an ordinary invoice, you get **409**.

On delivery, close the advance as in step 2, by the same `externalId` :

```
POST https://pos.narbiz.com/api/integrations/advance

{
  "zatvori": true,
  "externalId": "WEB-5521",
  "provider": "custom",
  "stavke": [
    { "naziv": "Bicikl", "kolicina": 2, "cenaSaPdv": 24500 },
    { "naziv": "Kaciga", "kolicina": 1, "cenaSaPdv": 3000 }
  ]
}
```

`provider` is the one the order was sent with; without it Tezga looks for an advance from `/advance` , then an advance from an order with `custom` . Without `stavke` the final invoice has one row with the names from the order.

25. Invoice copy (POST copy)

For when a customer asks for another copy of an already issued invoice. A copy is a separate fiscal document (КОПИЈА) that refers to the original.

```
POST https://pos.narbiz.com/api/integrations/copy
X-Tezga-API-Key: tzg_...

{ "documentId": "b3f1..." }    // or { "externalId": "NALOG-1042" }
```

The response carries `original` , `kopija` (same shape as in chapter 14) and `naPfr` .

- **naPfr: true** : the original was fiscalised, so the copy also went to the Tax Administration (Copy; for a copy of a refund, Copy/Refund).
- **naPfr: false** : the original is not fiscal (such as an eFaktura), so the copy exists in Tezga but is not sent to the Tax Administration.
- A draft cannot be copied (409); a foreign or non-existent document is 404.
- `provider` goes with `externalId` , default `custom` .

Every call creates a new copy at the Tax Administration. A customer may ask for several copies, so a repeated request is not an error. When you retry because of a dropped connection, send the `Idempotency-Key` header (up to 100 characters, longer is 400): the same key returns the same copy with `status: "postojalo"`, and while the first copy is in progress the response is 409.

26. Products and customers from an ERP (POST products, customers)

When your ERP is the source of the master data: you send products and customers, and Tezga writes them by the same rules as a sync from a web shop. Nothing is deleted.

Products

```
POST https://pos.narbiz.com/api/integrations/products
X-Tezga-API-Key: tzg_...

{
  "stavke": [
    { "sifra": "KAF-01", "naziv": "Espresso", "cenaSaPdv": 180, "poreskaStopa": "20",
    "jedinicaMere": "kom" },
    { "sifra": "HL-05", "naziv": "Hleb beli 500 g", "cenaSaPdv": 75, "poreskaStopa":
    "10", "gtin": "8600123456789" }
  ]
}
```

FIELD	MEANING
sifra REQUIRED	The code in your ERP, up to 100 characters. The write key: the same code (regardless of case) changes the same product. A product with no code in Tezga is recognised by name and gets this code.
naziv REQUIRED	Name as it goes on the invoice, up to 200 characters. It changes the name of an existing product.
cenaSaPdv REQUIRED	Unit price including VAT, 0 or more. It also accepts text ("1990,50"). A price change enters the product's price history, as when a person changes it.
poreskaStopa	Same as in an order (chapter 13): "20" , "10" , "0" or "exempt" ; empty text is 20%, an unknown value is an error for that item. Omitted: a new product gets 20%, an existing one keeps its own.
jedinicaMere	Up to 20 characters (kom , kg , usluga). A new product without it: kom .
gtin	Barcode, 8 to 14 digits.
aktivan	false removes the product from the register without deleting it.

Stock levels are not sent. The fields stanje , zaliha , zalihe , kolicina , pratiZalihu and minStanje return an item error. Stock changes only through receipt, write-off, stocktake and sale, so that every change leaves a trace in the stock ledger. The same applies to any field not in the table (such as a discount): an error, not a silent skip.

What is **not touched** on change: favourite products, the product's discount, category, description and stock. Customer price list: an entry with a fixed price stays the same, and a percentage discount is calculated from the new price.

Customers

```
POST https://pos.narbiz.com/api/integrations/customers
X-Tezga-API-Key: tzg_...
```

```
{
  "kupci": [
    { "naziv": "Primer doo", "pib": "100200309", "maticniBroj": "12345678",
      "adresa": "Bulevar 1", "mesto": "Novi Sad", "postanskiBroj": "21000",
      "email": "racuni@primer.rs", "telefon": "021 123 456" }
  ]
}
```

- **Only naziv is required.** Other fields: `pib` (9 digits, the check digit is verified), `maticniBroj` (8 digits, PIB is required with it), `jbkjs` (5 digits), `email`, `telefon`, `adresa`, `mesto`, `postanskiBroj` (5 digits).
- **How a customer is recognised:** by PIB, then by email, then by name. A customer with a PIB is looked up by email and name only among customers without a PIB, because a different PIB is a different company. POST order uses the same rule when it creates a customer.
- A customer that is found gets the name and the fields you send; a field you do not send or send empty stays as it is.
- A customer with a JBKJS is public sector: POST order rejects the fiscal flow for them, and only an eFaktura to SEF goes through (chapter 11).

Response

```
{
  "ok": false,
  "ukupno": { "kreiran": 1, "izmenjen": 0, "bez_izmene": 0, "greska": 1 },
  "stavke": [
    { "indeks": 0, "sifra": "KAF-01", "status": "kreiran", "id": "5d2c...", "nadjenPo": null, "poruka": null },
    { "indeks": 1, "sifra": "HL-05", "status": "greska", "id": null, "nadjenPo": null, "poruka": "poreskaStopa: Nepoznata poreska stopa. ..." }
  ]
}
```

For customers the array is `kupci`, and instead of `sifra` there are `pib` and `naziv`; `nadjenPo` is `pib`, `email` or `naziv`.

- **200:** the request was processed. **One bad item does not break the others**, so always check `ok` and the `status` of each item: `kreiran` (created), `izmenjen` (changed), `bez_izmene` (same data, nothing written) or `greska` (see `poruka`, fix it and send only that item).
- **422:** no array, it is empty or has more than 500 items; nothing was written. **400:** the body is not JSON. **413:** body over 512 KB.
- **403:** a read-only key or a register terminal key may not write. **401:** key or HMAC signature (same as on order, chapter 8). **429:** these routes use the same window of 120 requests per minute as invoice issuing.

- The same call repeated after a dropped connection is safe: nothing is duplicated, and items already written return `bez_izmene` .

This guide reflects the state of the system on the day it was generated. The most up-to-date examples, with **your** key already filled in, are always in the register: **Settings → Integrations**.