Fiscalization API documentation (technical integration)

v1.1 by Megamont d.o.o

Changelog

Version	Date	Author	Changes
1.0	2025-1-14	Hrvoje Oreč	Initial version
1.1	2025-22-11	Hrvoje Oreč	Add `cancel` endpoint

Table of contents

1. Overview

This API enables fiscalization of invoices according to Croatian tax legislation. It allows clients (typically point-of-sale terminals) to issue invoices, automatically submit them to the Tax Authority (Porezna uprava), and receive fiscal confirmation (JIR).

Key capabilities

- **Two fiscalization modes:**
    - **Direct mode:** Client applications (e.g., POS terminals) have full control over the fiscalization process. They issue invoices via API calls and receive immediate feedback and confirmations (e.g., JIR).
    - **Automated mode:** Fiscalization is triggered automatically on the backend via Megamont Fiscal as its service has direct integrations with various POS providers in a region. The system automatically takes care of fiscalization logic and stores results.
- Create invoices and submit to CIS
- Cancel or retry fiscalization if needed
- Receive fiscal confirmation via webhook
- Query invoice status or fetch daily fiscalized reports
- Secure communication via signature authentication

2. Authentication

2.1 Terminal id & key

Each client (terminal) receives:

- **Terminal id (`Tid`)**: Unique identifier assigned to each terminal (e.g., `TID-001`)
- **Terminal key**: Secret symmetric key known only to the client and server

These values must be stored securely and used for request signing.

2.2 Signature scheme

All requests must be **signed** using a SHA-512 HMAC-style scheme based on:

```
Signature = SHA512( Tid + RequestBody + TerminalKey )
```

The **`Signature`** must be sent via the `X-Signature` HTTP header, while the `Tid` is sent in the request body.

2.3 Required headers

Header	Example value	Description
`X-Signature`	`dfea203e...` (hex-encoded SHA-512 string)	Signature as per hash formula
`Content-Type`	`application/json`	Required for JSON body

2.4 Example signature generation (C#)

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

string terminalId = "TID-001";
string terminalKey = "SuperSecretTerminalKey:)";

// The JSON string, minified and excluding any signature field
string body = "{"invoiceNumber":"123","amount":100.0,"tid":"TID-001"}";

string stringToHash = terminalId + body + terminalKey;

using var sha512 = SHA512.Create();
byte[] hash = sha512.ComputeHash(Encoding.UTF8.GetBytes(stringToHash));
string signatureHex = BitConverter.ToString(hash).Replace("-", "").ToLower();

// Use signatureHex as the value for X-Signature
```

2.5 Server-side verification

On each request:

1. The server extracts the `Tid` from the JSON body.
2. It fetches the corresponding `TerminalKey` from secure storage.
3. It rebuilds the signature string and calculates the SHA-512 hash.
4. It compares the result with the value in `X-Signature`.

If they do not match, a `401 Unauthorized` is returned with:

```json
{
  "error": "Invalid signature."
}
```

3. Endpoints

3.1 Create invoice

Creates a new invoice and attempts immediate fiscalization with the Tax Authority (CIS).
If the CIS is unreachable, the invoice is stored and automatically retried later.

Method & URL

```
POST /api/invoices
```

Headers

Header	Required	Description
`Content-Type`	Yes	`application/json`
`X-Signature`	Yes	SHA-512 signature of the request body

Request body

```json
{
  "tid": "TID-001",
  "numerationIdentifier": "2025",
  "locationIdentifier": "POS1",
  "deviceIdentifier": "01",
  "dateIssued": "2025-07-31T13:15:00Z",
  "operatorOib": "12345678901",
  "paymentMethod": "cash",
  "isDelayedDelivery": false,
  "taxGroups": [
    {
        "taxRate": 5.0,
        "baseAmount": 100.00,
        "taxAmount": 5.00,
        "totalAmount": 105.00
    },
    {
        "taxRate": 25.0,
        "baseAmount": 15.20,
        "taxAmount": 3.80,
        "totalAmount": 19.00
    }
  ]
}
```

Field	Type	Required	Description
`tid`	string	Yes	Terminal identifier
`numerationIdentifier`	string	Yes	Invoice numbering series (e.g., year)
`locationIdentifier`	string	Yes	Location code of the invoice issuer
`deviceIdentifier`	string	Yes	Register/Device number at that location
`dateIssued`	ISO 8601	Yes	When the invoice was issued
`operatorOib`	string	Yes	OIB of the person issuing the invoice
`paymentMethod`	string	Yes	Must be one of: `cash` , `card` , `multiple` , `neither`

Field	Type	Required	Description
`isDelayedDelivery`	boolean	Yes	`true` if invoice was delivered to system *after* issuance
`taxGroups`	array	Yes	List of VAT calculation entries
→ `taxRate`	number	Yes	VAT rate applied
→ `baseAmount`	decimal	Yes	Amount before tax
→ `taxAmount`	decimal	Yes	Tax value
→ `totalAmount`	decimal	Yes	Total amount (base + tax) for that group

Response (fiscalized)

```
{
  "invoiceId": "b1e7f660-52ae-4bce-8415-9a8d69e1a00a",
  "status": "fiscalized",
  "jir": "1234567890ABCDEF1234567890ABCDEF",
  "zki": "1234567890ABCDEF1234567890ABCDEF",
  "fiscalizedAt": "2025-07-31T13:15:04Z"
}
```

Response (accepted but not fiscalized yet)

```
{
  "invoiceId": "b1e7f660-52ae-4bce-8415-9a8d69e1a00a",
  "status": "pendingFiscalization",
  "message": "Invoice stored but not yet fiscalized. Will retry automatically."
}
```

Possible errors

HTTP Code	Description
400	Validation error (e.g., invalid `paymentMethod`)
401	Invalid or missing signature
409	Duplicate invoice number
500	Internal server error

Notes

- Fiscalization is attempted immediately, but success is not guaranteed due to CIS availability.
- If the invoice is sent after being issued (`isDelayedDelivery: true`), the backend will tag it accordingly in the request to CIS.
- The backend is responsible for automatic retries and queuing.
- The `paymentMethod` is required for legal reporting and analytics.

3.2 Cancel invoice

Cancels a previously fiscalized invoice. Used when a customer returns goods or a transaction must be nullified for legal or operational reasons.

Method & URL

```
POST /api/v1/invoices/cancel
```

Headers

Header	Required	Description
`Content-Type`	Yes	`application/json`
`X-Signature`	Yes	SHA-512 signature of the request body

Request body

```
{
  "tid": "TID-001",
  "invoiceId": "57e5aa49-7a7e-4c52-8f93-fdb42f1d88b9",
  "cancellationReason": "Customer returned the product",
  "cancelDateTime": "2025-07-30T12:34:56Z",
  "operatorId": "OP-321",
  "isDelayedDelivery": false
}
```

Field	Type	Required	Description
`invoiceId`	string	Yes	The internal ID of the invoice to cancel (UUID from original response)
`cancellationReason`	string	Yes	Reason for cancellation
`cancelDateTime`	string	Yes	ISO8601 timestamp of when the invoice was cancelled
`operatorId`	string	No	Operator or cashier performing the cancellation
`isDelayedDelivery`	boolean	No	If true, indicates that the invoice was submitted after the original issue time

Response (fiscalized)

```
{
  "invoiceId": "c1e7f660-52ae-4bce-8415-9a8d69e1a00a",
  "status": "fiscalized",
  "jir": "1234567890ABCDEF1234567890ABCDEF",
  "zki": "1234567890ABCDEF1234567890ABCDEF",
  "fiscalizedAt": "2025-07-32T13:15:04Z"
}
```

Response (accepted but not fiscalized yet)

```
{
  "invoiceId": "c1e7f660-52ae-4bce-8415-9a8d69e1a00a",
  "status": "pendingFiscalization",
  "message": "Cancellation stored but not yet fiscalized. Will retry automatically."
}
```

Possible errors

HTTP Code	Description
400	Validation error (e.g., invalid `paymentMethod`)
401	Invalid or missing signature
409	Duplicate invoice number
500	Internal server error

3.3 Get invoice status

Retrieves the current fiscalization status for a given invoice using terminal ID, a per-request random key, and the invoice ID.

Method & URL

```
GET /api/invoices/{tid}/{randomKey}/{invoiceId}
```

Path parameters

Parameter	Type	Description
`tid`	string	Terminal identifier
`randomKey`	string	A per-request random string (GUID or similar)

Parameter	Type	Description
`invoiceId`	string	UUID of the invoice to look up

Headers

Header	Required	Description
`X-Signature`	Yes	SHA-512 signature: SHA512(tid + randomKey + invoiceId + TerminalKey)
`Accept`	Yes	`application/json`

Signature calculation

```
Signature = SHA512( Tid + RandomKey + InvoiceId + TerminalKey )
```

Response (fiscalized)

```json
{
  "invoiceId": "b1e7f660-52ae-4bce-8415-9a8d69e1a00a",
  "status": "fiscalized",
  "jir": "1234567890ABCDEF1234567890ABCDEF",
  "zki": "1234567890ABCDEF1234567890ABCDEF",
  "fiscalizedAt": "2025-07-31T13:15:04Z"
}
```

Response (pendingFiscalization)

```
{
  "invoiceId": "b1e7f660-52ae-4bce-8415-9a8d69e1a00a",
  "status": "pendingFiscalization",
  "message": "Invoice has not yet been fiscalized; retry attempts are in progress."
}
```

Possible errors

HTTP Code	Description
400	Invalid `tid` , `randomKey` , or `invoiceId` format
401	Missing or invalid signature
404	Invoice not found
500	Internal server error

3.4 Retry fiscalization

Manually re-enqueues a previously accepted invoice for fiscalization with the Tax Authority (CIS). Use this endpoint when you need to expedite an invoice that remains in a pendingFiscalization state, for example after a system outage or rate-limit pause. It immediately attempts another call to CIS, returning either a successful fiscalized status or a continued pendingFiscalization response. All retry attempts are logged for audit and support purposes.

Method & URL

```
POST /api/invoices/{tid}/{randomKey}/{invoiceId}/retry
```

Path parameters

Parameter	Type	Description
`tid`	string	Terminal identifier
`randomKey`	string	A per-request random string (GUID or similar)
`invoiceId`	string	UUID of the invoice to retry fiscalization for

Headers

Header	Required	Description
`Content-Type`	Yes	`application/json`
`X-Signature`	Yes	SHA-512 signature: SHA512(tid + randomKey + invoiceId + TerminalKey)

Signature calculation

```
Signature = SHA512( Tid + RandomKey + InvoiceId + TerminalKey )
```

Request body

Empty.

Response (fiscalized)

```
{
  "invoiceId": "b1e7f660-52ae-4bce-8415-9a8d69e1a00a",
  "status": "fiscalized",
  "jir": "1234567890ABCDEF1234567890ABCDEF",
  "zki": "1234567890ABCDEF1234567890ABCDEF",
  "fiscalizedAt": "2025-07-31T14:00:00Z"
}
```

Response (pendingFiscalization)

```
{
  "invoiceId": "b1e7f660-52ae-4bce-8415-9a8d69e1a00a",
  "status": "pendingFiscalization",
  "message": "Retry initiated; fiscalization still pending."
}
```

Possible errors

HTTP Code	Description
400	Invalid `tid` , `randomKey` , or `invoiceId` format
401	Missing or invalid signature
404	Invoice not found or already fiscalized
429	Retry rate limit exceeded
500	Internal server error

User story: manual retry of fiscalization

Title

As a merchant support agent, I want to manually trigger a retry of fiscalization for a specific invoice, so that I can resolve pending invoices when the automated process has failed or stalled.

Narrative

- Who? A back-office user or merchant support agent
- What? Initiates a retry of fiscalization for an invoice whose status is pendingFiscalization
- Why? Because the automatic background attempts may have hit a system outage or rate limit, and the agent needs to expedite resolution to comply with tax rules and close out daily reporting.

Pre-conditions

- The invoice was created and initially submitted, but CIS returned no response (network failure, rate-limit, etc.).
- The system marked the invoice status as pendingFiscalization.
- The invoice appears in the “Pending fiscalization” dashboard for merchant support.

Flow

1. Support agent locates the invoice in the dashboard (filtered by status = pendingFiscalization).
2. Agent clicks “Retry fiscalization” on the invoice row.
3. Front-end calls:

   ```
   POST /api/invoices/{tid}/{randomKey}/{invoiceId}/retry
   ```

4. API immediately enqueues a retry and attempts a new call to CIS.
5. API returns one of:
   - fiscalized (if the CIS call succeeds)
   - pendingFiscalization (if the CIS is still unavailable, e.g. offline test environment)

Acceptance criteria

- Agent sees immediate feedback (status = fiscalized or still pending).
- If successful, the invoice is updated with JIR & timestamp, and removed from the pending list.
- If still pending, the invoice remains queued and the agent can retry again later or investigate logs.
- All retry attempts are logged with timestamps and outcomes for audit.

3.5 Webhook callbacks

Delivers asynchronous notifications to the merchant’s endpoint when fiscalization events occur. Use these callbacks to update the local system state immediately upon successful or failed fiscalization without polling. Callback payloads are signed with the same HMAC-SHA512 scheme as requests, ensuring authenticity.

Method & URL

The merchant must expose an HTTP(S) endpoint to receive POST requests:

```
POST /api/webhooks/fiscalization --sample path
```

Headers

Header	Required	Description
`X-Signature`	Yes	SHA-512 signature: SHA512(tid + randomKey + payload + TerminalKey)
`Content-Type`	Yes	`application/json`

Signature calculation

```
Signature = SHA512( Tid + JSON.stringify(payload) + TerminalKey )
```

Payload

```
{
  "tid": "TID-001",
  "invoiceId": "b1e7f660-52ae-4bce-8415-9a8d69e1a00a",
  "eventType": "invoice_fiscalized",
  "status": "fiscalized",
  "jir": "1234567890ABCDEF1234567890ABCDEF",
  "zki": "1234567890ABCDEF1234567890ABCDEF",
  "fiscalizedAt": "2025-07-31T13:15:04Z",
  "message": null
}
```

Field	Type	Description
`tid`	string	Terminal identifier
`invoiceId`	string	UUID of the invoice

Field	Type	Description
`eventType`	string	One of: `invoice_fiscalized`, `invoice_pending`, `invoice_failed`, `invoice_cancelled`
`status`	string	Fiscalization status: `fiscalized`, `pendingFiscalization`, or `failed`
`jir`	string	JIR code if available
`zki`	string	ZKI signature
`fiscalizedAt`	string	ISO 8601 timestamp of fiscalization
`message`	string?	Optional human-readable message or error

Delivery guarantees

- Retries: The sender retries delivery up to 5 times with exponential backoff if a non-2xx response is received.
- Idempotency: Callbacks include the same `randomKey` per attempt, and the merchant must respond with `2xx` only once per `invoiceId + eventType`.

Example delivery

```
POST /api/webhooks/fiscalization HTTP/1.1
Host: app.example-merchant.com
X-Tid: TID-001
X-Signature: df0a1b... (hex-encoded SHA-512)
Content-Type: application/json

{ ...payload above... }
```

3.6 Get fiscalized data (daily report)

Retrieves all invoices successfully fiscalized for a given day and terminal. This endpoint is typically used by merchants who rely on **automated fiscalization**, and need to download results for reporting, reconciliation, or local archiving.

Method & URL

```
GET /api/reports/{tid}/{date}
```

Path parameters

Parameter	Type	Description
`tid`	string	Terminal identifier
`date`	string	ISO 8601 date (`yyyy-MM-dd`)

Headers

Header	Required	Description
`X-Signature`	Yes	SHA-512 signature: SHA512(tid + date + TerminalKey)
`Accept`	Yes	`application/json`

Signature calculation

```
Signature = SHA512( Tid + Date + TerminalKey )
```

Example:

```
SHA512("TID-0012025-07-31SuperSecretTerminalKey:)")
```

Response

```
{
  "tid": "TID-001",
  "date": "2025-07-31",
  "fiscalizedInvoices": [
    {
      "invoiceId": "b1e7f660-52ae-4bce-8415-9a8d69e1a00a",
      "numerationIdentifier": "2025",
      "locationIdentifier": "POS1",
      "deviceIdentifier": "01",
      "operatorOib": "12345678901",
      "paymentMethod": "card",
      "fiscalizedAt": "2025-07-31T13:15:04Z",
      "jir": "1234567890ABCDEF1234567890ABCDEF",
      "zki": "1234567890ABCDEF1234567890ABCDEF",
      "taxGroups": [
            {
            "taxRate": 5.0,
            "baseAmount": 100.00,
            "taxAmount": 5.00,
            "totalAmount": 105.00
            },
            {
            "taxRate": 25.0,
            "baseAmount": 15.20,
            "taxAmount": 3.80,
            "totalAmount": 19.00
            }
        ]
    }
  ]
}
```

Possible errors

HTTP Code	Description
400	Invalid date format or unknown terminal ID

HTTP Code	Description
401	Missing or invalid signature
404	No invoices found for given `tid` and `date`
500	Internal server error

Notes

- The report includes only successfully **fiscalized** invoices.
- Invoices with `pendingFiscalization` or `failed` status are excluded.
- Use this endpoint once per day per terminal for end-of-day reconciliation.
- The order of invoices in the array is chronological by `fiscalizedAt` .