



المستودعات السعودية
 وإدارة الفواتير

SA4S BUSINESS ERP

SA4S Developer API Guide

Connect your online store, website, accounting or delivery app to SA4S Business ERP

Edition	Updated	API version
1.1.0	2026-10-06	v1 · 1.0.0

<https://erp.sa4s.com/api/v1>



Contents

Overview	5
What you can build	5
Key concepts	5
Base URL and versioning	5
Quick start	7
1. Create a key	7
2. Make the first call	7
3. A reusable client	8
Authentication	13
Sending the key	13
Key format and environments	13
Rotation (24-hour grace)	13
Revocation and expiry	13
IP allow-list	14
Security checklist	14
Scopes	15
Scope catalogue	15
What keys can never do	16
Conventions	17
Pagination	17
Sorting	18
Search and filters	18
Dates and time zone	19
Money and quantities	19
VAT	19
Request IDs	20
Idempotency and safe retries	21
Rate limits	23
Errors	25



Error body	25
Error codes	26
Sandbox	29
Recipes — products	30
List and search products	30
Create a product with units and barcodes	31
Update a product	34
Look up a product by barcode	34
Recipes — customers	36
Create a customer	36
Find or update a customer	38
Customer statement	39
Recipes — sales	41
Simplified (B2C) invoice, VAT-inclusive, paid by card	41
Tax (B2B) invoice, VAT-exclusive, partially paid	43
Draft first, issue when the total matches	45
Retry an invoice safely	46
Download the invoice PDF	46
ZATCA status and QR code	47
Refund or return: credit note	48
Cancel an invoice	49
Recipes — payments	50
Record a customer receipt	50
List unpaid invoices	51
Pay a supplier	53
Recipes — purchasing	55
Create a purchase invoice	55
Create a purchase order	56
Recipes — inventory	58
Stock balances by warehouse	58
Stock movements since a date	60
Recipes — reports	62



Report catalogue	62
Run a report as JSON	63
Webhooks	64
Subscribe	64
Delivery format	64
Event catalogue	65
Verify the signature	67
Replay protection	69
Retries and redelivery	69
Testing locally	70
End-to-end: online store orders → SA4S invoices	71
End-to-end: keep stock in sync	75
Changelog and versioning	78
Support	79



Overview

The SA4S Developer API lets your own software read and write the business data of **your** SA4S company: products, customers, suppliers, sales and purchase invoices, payments, stock and reports. It is a JSON-over-HTTPS REST API, secured with API keys that you create and control in the ERP, plus signed **webhooks** that notify your systems the moment something happens.

The API is part of the plans that include the **API** feature (Business and Enterprise). Keys, webhooks and the interactive reference live in **Settings** → **API & Integrations**.

What you can build

- **Online store integration** — turn paid orders into ZATCA-ready invoices and keep store quantities in sync with the warehouse.
- **Point-of-sale or delivery apps** — look products up by barcode, create invoices with payments, download the invoice PDF.
- **Accounting and BI** — pull invoices, receipts, statements and ready-made reports (sales, stock, profit) as JSON or files.
- **Procurement tools** — create purchase orders and supplier invoices, record supplier payments.

Key concepts

Concept	What it means for an integration
Company (tenant)	Every key belongs to exactly one company. All data you read or write is that company's; other companies' records simply do not exist for you (<code>404 NOT_FOUND</code>). You never send a company id.
Branches & warehouses	Documents belong to a branch and move stock in a warehouse. When you omit <code>branchId</code> / <code>warehouseId</code> , the company's main branch and its default warehouse are used. Warehouse ids appear in stock rows (<code>warehouse.id</code>) and in every document.
Products & units	Stock is kept in the product's base unit (e.g. piece). Extra units (box = 24 pieces) have a <code>factor</code> ; a line in a box sells 24 base units (<code>baseQuantity</code>).
Documents	Invoices go <code>DRAFT</code> → <code>ISSUED</code> → <code>(CANCELLED)</code> . Issuing assigns the gap-free number, moves stock, posts the ledger and queues the e-invoice for ZATCA. Issued documents are never edited — you cancel them or issue a credit note.
Environments	<code>live</code> keys act for real. <code>sandbox</code> keys read the same company but can only create drafts (see Sandbox).

Base URL and versioning

All endpoints are relative to the base URL:

Base URL

`https://erp.sa4s.com/api/v1`



The version is part of the path (`/api/v1`). Within v1, changes are **additive only**: new endpoints, new optional request fields, new response fields, new error codes and new webhook events. Write tolerant clients — ignore unknown fields and events. A breaking change would ship as `/api/v2` with advance notice. The machine-readable OpenAPI 3.1 document is published at `GET https://erp.sa4s.com/api/v1/developer/openapi.json` (no key needed).



Quick start

From zero to your first successful call in about two minutes.

1. Create a key

1. In the ERP open **Settings** → **API & Integrations** → **API keys** → **New API key**.
2. Name it after the system that will use it (e.g. *Online store*), choose **Sandbox** while you build, and tick the scopes it needs — for this quick start `products:view` is enough.
3. Copy the secret (`sk_test_...`). It is shown **once**; SA4S stores only a hash of it.

Shell

```
# keep the key out of your code: read it from the environment / a secret manager
export SA4S_API_KEY="sk_test_....."
export SA4S_BASE_URL="https://erp.sa4s.com/api/v1"
```

2. Make the first call

curl

```
curl "https://erp.sa4s.com/api/v1/products?pageSize=2" \
-H "Authorization: Bearer $SA4S_API_KEY"
```

JavaScript

```
// Node.js 18+ (built-in fetch)
const res = await fetch(`${process.env.SA4S_BASE_URL}/products?pageSize=2`, {
  headers: { Authorization: `Bearer ${process.env.SA4S_API_KEY}` },
});
console.log(res.status, res.headers.get('X-RateLimit-Remaining'));
console.log(await res.json());
```

Python

```
import os, requests

res = requests.get(
    f"{os.environ['SA4S_BASE_URL']}/products",
    params={"pageSize": 2},
    headers={"Authorization": f"Bearer {os.environ['SA4S_API_KEY']}"},
    timeout=30,
)
print(res.status_code, res.headers.get("X-RateLimit-Remaining"))
print(res.json())
```

PHP

```
<?php
$ch = curl_init(getenv('SA4S_BASE_URL') . '/products?pageSize=2');
curl_setopt_array($ch, [
    CURLOPT_RETURNTRANSFER => true,
```



```
CURLOPT_HTTPHEADER => ['Authorization: Bearer ' . getenv('SA4S_API_KEY')],
]);
$page = json_decode(curl_exec($ch), true);
echo curl_getinfo($ch, CURLINFO_RESPONSE_CODE), ' ', $page['total'], " products\n";
```

C#

```
using System.Net.Http.Headers;

using var http = new HttpClient();
http.DefaultRequestHeaders.Authorization =
    new AuthenticationHeaderValue("Bearer", Environment.GetEnvironmentVariable("SA4S_API_KEY"));
var res = await http.GetAsync($"{Environment.GetEnvironmentVariable("SA4S_BASE_URL")}/products?pageSize=2");
Console.WriteLine($"{(int)res.StatusCode} {await res.Content.ReadAsStringAsync()}");
```

200 OK (abridged)

```
{
  "items": [
    {
      "id": "51f771bf-cfb5-4814-8e79-55b990be69c9",
      "sku": "SKU-2026-000001",
      "nameAr": "مياه معدنية 330 مل",
      "nameEn": "Mineral water 330 ml",
      "type": "PHYSICAL",
      "sellingPrice": "1.15",
      "vatCategory": "S",
      "isActive": true,
      "stock": { "onHand": "282", "reserved": "0", "available": "282" }
    }
  ],
  "total": 3,
  "page": 1,
  "pageSize": 2
}
```

A **200** with **items** means the key, the scopes and the plan are all in order. A **401** or **403** tells you exactly what is wrong — see [Errors](#).

3. A reusable client

The recipes in this guide use the small helper below (one per language) so each example stays short. It sends the key, parses JSON, raises a typed error carrying **code**, **details** and **requestId**, and retries safely: always on **429**, and on network errors / **5xx** only for **GET** requests and for creates that carry a **clientId** (see [Idempotency](#)).

JavaScript

```
// sa4s.mjs - a tiny client for the SA4S Developer API (Node.js 18+, no dependencies)
const BASE = process.env.SA4S_BASE_URL ?? 'https://erp.sa4s.com/api/v1';
const KEY = process.env.SA4S_API_KEY; // sk_live... or sk_test...

export class Sa4sError extends Error {
  constructor(status, body) {
    super(`${body.code ?? status}: ${body.message ?? 'request failed'}`);
    Object.assign(this, { status, code: body.code, details: body.details, requestId: body.requestId });
  }
}

const sleep = (ms) => new Promise((resolve) => setTimeout(resolve, ms));
const backoff = (attempt) => Math.min(30_000, 500 * 2 ** attempt) * (0.5 + Math.random() / 2); // exponential + jitter
```



```

/** sa4s('GET', '/products?pageSize=50') · sa4s('POST', '/customers', { name: '...' }) */
export async function sa4s(method, path, body, { retries = 4 } = {}) {
  // GETs and creates that carry a clientId are safe to repeat; anything else is retried only on 429 (rejected before
  processing)
  const idempotent = method === 'GET' || Boolean(body?.clientId);
  for (let attempt = 0; ; attempt++) {
    let res;
    try {
      res = await fetch(BASE + path, {
        method,
        headers: { Authorization: `Bearer ${KEY}` }, ...(body ? { 'Content-Type': 'application/json' } : {}),
        body: body ? JSON.stringify(body) : undefined,
        signal: AbortSignal.timeout(30_000),
      });
    } catch (err) { // network error or timeout
      if (!idempotent || attempt >= retries) throw err;
      await sleep(backoff(attempt));
      continue;
    }
    if (res.ok) {
      if (res.status === 204) return null;
      return res.headers.get('content-type')?.includes('json') ? res.json() : Buffer.from(await res.arrayBuffer());
    }
    if (attempt < retries && (res.status === 429 || (res.status >= 500 && idempotent))) {
      const after = Number(res.headers.get('Retry-After'));
      await sleep(after > 0 ? after * 1000 : backoff(attempt));
      continue;
    }
    throw new Sa4sError(res.status, await res.json().catch(() => ({})));
  }
}

```

Python

```

# sa4s.py – a tiny client for the SA4S Developer API (pip install requests)
import os, random, time
import requests

BASE = os.environ.get("SA4S_BASE_URL", "https://erp.sa4s.com/api/v1")
session = requests.Session()
session.headers["Authorization"] = f"Bearer {os.environ['SA4S_API_KEY']}"

class Sa4sError(Exception):
    def __init__(self, status, body):
        super().__init__(f"{body.get('code', status)}: {body.get('message', 'request failed')}")
        self.status, self.code = status, body.get("code")
        self.details, self.request_id = body.get("details"), body.get("requestId")

def backoff(attempt):
    return min(30, 0.5 * 2 ** attempt) * random.uniform(0.5, 1) # exponential + jitter

def sa4s(method, path, json=None, params=None, retries=4):
    """sa4s("GET", "/products", params={"pageSize": 50}) · sa4s("POST", "/customers", {"name": "..."})"""
    idempotent = method == "GET" or bool(json and json.get("clientId"))
    for attempt in range(retries + 1):
        try:
            res = session.request(method, BASE + path, json=json, params=params, timeout=30)
        except requests.RequestException:
            if not idempotent or attempt == retries:
                raise
            time.sleep(backoff(attempt))
            continue

```



```

if res.ok:
    if res.status_code == 204:
        return None
    return res.json() if "json" in res.headers.get("Content-Type", "") else res.content
if attempt < retries and (res.status_code == 429 or (res.status_code >= 500 and idempotent)):
    time.sleep(float(res.headers.get("Retry-After") or backoff(attempt)))
    continue
try:
    body = res.json()
except ValueError:
    body = {}
raise Sa4sError(res.status_code, body)

```

PHP

```

<?php
// sa4s.php - a tiny client for the SA4S Developer API (PHP 8.1+, ext-curl, no dependencies)

final class Sa4sError extends RuntimeException
{
    public function __construct(public readonly int $status, public readonly array $body)
    {
        parent::__construct(($body['code'] ?? $status) . ': ' . ($body['message'] ?? 'request failed'));
    }
}

function sa4s_backoff_ms(int $attempt): int
{
    return (int) (min(30000, 500 * 2 ** $attempt) * (0.5 + mt_rand() / mt_getrandmax() / 2));
}

/** sa4s('GET', '/products?pageSize=50') · sa4s('POST', '/customers', ['name' => '...']) */
function sa4s(string $method, string $path, ?array $body = null, int $retries = 4): mixed
{
    $base = getenv('SA4S_BASE_URL') ?: 'https://erp.sa4s.com/api/v1';
    $idempotent = $method === 'GET' || !empty($body['clientId']);
    for ($attempt = 0; ; $attempt++) {
        $headers = [];
        $ch = curl_init($base . $path);
        $opts = [
            CURLOPT_CUSTOMREQUEST => $method,
            CURLOPT_RETURNTRANSFER => true,
            CURLOPT_TIMEOUT => 30,
            CURLOPT_HTTPHEADER => ['Authorization: Bearer ' . getenv('SA4S_API_KEY'), 'Content-Type: application/json'],
            CURLOPT_HEADERFUNCTION => function ($ch, string $line) use (&$headers): int {
                if (str_contains($line, ':')) {
                    [$k, $v] = explode(':', $line, 2);
                    $headers[strtolower(trim($k))] = trim($v);
                }
                return strlen($line);
            },
        ];
        if ($body !== null) {
            $opts[CURLOPT_POSTFIELDS] = json_encode($body, JSON_UNESCAPED_UNICODE | JSON_THROW_ON_ERROR);
        }
        curl_setopt_array($ch, $opts);
        $raw = curl_exec($ch);
        $status = (int) curl_getinfo($ch, CURLINFO_RESPONSE_CODE);
        if ($raw === false) { // network error or timeout
            if (!$idempotent || $attempt >= $retries) {
                throw new RuntimeException('SA4S request failed: ' . curl_error($ch));
            }
            usleep(sa4s_backoff_ms($attempt) * 1000);
            continue;
        }
    }
}

```

```
}
if ($status >= 200 && $status < 300) {
    if ($status === 204) return null;
    return str_contains($headers['content-type'] ?? '', 'json') ? json_decode($raw, true) : $raw;
}
if ($attempt < $retries && ($status === 429 || ($status >= 500 && $idempotent))) {
    $wait = isset($headers['retry-after']) ? (int) $headers['retry-after'] * 1000 : sa4s_backoff_ms($attempt);
    usleep($wait * 1000);
    continue;
}
throw new Sa4sError($status, json_decode($raw, true) ?? []);
}
}
```

C#

```
// Sa4s.cs - a tiny client for the SA4S Developer API (.NET 8+, no dependencies)
using System.Net;
using System.Net.Http.Headers;
using System.Net.Http.Json;
using System.Text.Json;

public sealed class Sa4sException(int status, JsonElement body) : Exception(
    $"{(body.TryGetProperty("code", out var c) ? c.GetString() : status.ToString())}: " +
    $"{(body.TryGetProperty("message", out var m) ? m.GetString() : "request failed")}"
)
{
    public int Status { get; } = status;
    public JsonElement Body { get; } = body; // code, message, details, requestId
}

public static class Sa4s
{
    public static readonly HttpClient Http = new()
    {
        BaseAddress = new Uri((Environment.GetEnvironmentVariable("SA4S_BASE_URL") ??
"https://erp.sa4s.com/api/v1").TrimEnd('/') + "/"),
        Timeout = TimeSpan.FromSeconds(30),
    };

    static Sa4s() => Http.DefaultRequestHeaders.Authorization =
        new AuthenticationHeaderValue("Bearer", Environment.GetEnvironmentVariable("SA4S_API_KEY"));

    static TimeSpan Backoff(int attempt) =>
        TimeSpan.FromMilliseconds(Math.Min(30_000, 500 * Math.Pow(2, attempt)) * (0.5 + Random.Shared.NextDouble() /
2));

    /// Sa4s.Send(HttpMethod.Get, "products?pageSize=50") · Sa4s.Send(HttpMethod.Post, "customers", new { name = "..." })
    public static async Task<JsonElement> Send(HttpMethod method, string path, object? body = null, int retries = 4)
    {
        bool idempotent = method == HttpMethod.Get
            || (body is not null && JsonSerializer.SerializeToElement(body).TryGetProperty("clientId", out _));
        for (int attempt = 0; ; attempt++)
        {
            using var req = new HttpRequestMessage(method, path.TrimStart('/'));
            if (body is not null) req.Content = JsonContent.Create(body);
            HttpResponseMessage res;
            try { res = await Http.SendAsync(req); }
            catch (Exception e) when ((e is HttpRequestException or TaskCanceledException) && idempotent && attempt <
retries)
            {
                await Task.Delay(Backoff(attempt));
                continue;
            }
            using (res)
            {

```



```
        if (res.IsSuccessStatusCode)
            return res.StatusCode == HttpStatusCode.NoContent ? default : await
res.Content.ReadFromJsonAsync<JsonElement>();
        int status = (int)res.StatusCode;
        if (attempt < retries && (status == 429 || (status >= 500 && idempotent)))
        {
            await Task.Delay(res.Headers.RetryAfter?.Delta ?? Backoff(attempt));
            continue;
        }
        throw new Sa4sException(status, await res.Content.ReadFromJsonAsync<JsonElement>());
    }
}
}
```



Authentication

Sending the key

Send the key on **every** request, in either header. Both are equivalent; use one, not both.

HTTP

```
GET /api/v1/products HTTP/1.1
Host: erp.sa4s.com
Authorization: Bearer sk_live_.....

# or, equivalently
X-API-Key: sk_live_.....
```

Calls without a key get `401 UNAUTHORIZED`. Keys must only ever be used **server-side**: never ship them in a browser, a mobile app or a public repository — anyone holding the secret can act with its scopes.

Key format and environments

Prefix	Environment	Behaviour
sk_live_ + 40 characters	live	Full access within the granted scopes.
sk_test_ + 40 characters	sandbox	Same company data, read-only except creating drafts (see Sandbox).

The first 12 characters (e.g. `sk_live_XTPx`) are the key **prefix** shown in the ERP so you can tell keys apart; the rest is secret. Each key also has an optional expiry date, an optional IP allow-list and a per-minute rate limit (10–5000, default 300).

Rotation (24-hour grace)

Rotate secret on a key in the ERP to get a new secret with the same name, scopes and limits. The old secret keeps working for **24 hours** (status *Rotating*) and then expires, so you can deploy the new secret without downtime:

1. Rotate → copy the new secret.
2. Deploy it to your integration (secret manager / environment).
3. Verify traffic on the new key in the usage chart; the old one stops by itself after 24 h.

A key can be rotated once; rotate the new key next time. Rotate periodically and immediately when a person with access leaves.

Revocation and expiry

Revoke stops a key immediately (`401 API_KEY_REVOKED`) and cannot be undone — create a new key instead. A key past its expiry date answers `401 API_KEY_EXPIRED`. Downgrading to a plan without the API feature makes every key answer `403 FEATURE_NOT_AVAILABLE` until the plan includes it again.



IP allow-list

For server-to-server integrations, restrict a key to the addresses your servers use: one IPv4/IPv6 address or CIDR block per line (`203.0.113.10` , `198.51.100.0/24` , `2001:db8::/32`). Calls from any other address get `401 API_KEY_IP_DENIED` . An empty list allows any address.

Security checklist

- One key per integration, with the **fewest scopes** it needs; separate keys for sandbox and live.
- Store secrets in a secret manager or environment variables — never in source code, logs or URLs.
- Add an expiry for temporary access and an IP allow-list for fixed servers.
- Watch *Last used* and the usage chart per key; revoke keys you no longer use.
- Verify every webhook signature and reject old timestamps (see [Webhooks](#)).



Scopes

A key can do only what its scopes allow. Each endpoint requires one scope (a few need two); a call without it returns `403 PERMISSION_DENIED` with the missing scope in `details.missing`. Scopes follow `resource:action`.

Scope catalogue

Actions: **view** read · **create** create · **edit** update · **delete** delete · **approve** approve/post · **cancel** cancel · **refund** refunds & credit notes · **print** PDFs · **export** files (CSV/XLSX/PDF reports). Sandbox keys keep only view, create, print and export.

Group	Scope resource	Actions	What it unlocks
Products & catalog	<code>products</code>	view, create, edit, delete, export, print	Products, units of a product, barcodes, lookup
Products & catalog	<code>categories</code>	view, create, edit, delete, export	Category list and tree
Products & catalog	<code>units</code>	view, create, edit, delete, export	Units of measure
Customers & suppliers	<code>customers</code>	view, create, edit, delete, export, print	Customers and their statements
Customers & suppliers	<code>suppliers</code>	view, create, edit, delete, export, print	Suppliers and their statements
Sales	<code>sales.invoice</code>	view, create, edit, delete, export, approve, cancel, refund, print	Sales invoices, issuing, cancelling, credit notes, PDF
Sales	<code>sales.discount</code>	manage	Line and document discounts on invoices (<code>discount</code> , <code>discountPercent</code> , <code>documentDiscount</code>)
Sales	<code>sales.return</code>	view, create, approve	Sales returns
Sales	<code>sales.delivery</code>	view, print	Delivery notes (deliver-later / backorder invoices) and the undelivered invoices report, delivery note PDF
Purchasing	<code>purchases.request</code>	view, create, edit, delete, export, approve	Purchase requests
Purchasing	<code>purchases.order</code>	view, create, edit, delete, export, approve, cancel, print	Purchase orders
Purchasing	<code>purchases.receipt</code>	view, create, cancel	Goods receipts
Purchasing	<code>purchases.invoice</code>	view, create, edit, delete, export, approve, cancel, print	Purchase (supplier) invoices; <code>approve</code> posts them



Group	Scope resource	Actions	What it unlocks
Purchasing	<code>purchases.return</code>	view, create, approve	Purchase returns
Inventory	<code>inventory</code>	view, create, export	Balances, movements, low stock, opening stock
Inventory	<code>inventory.cost</code>	view	Purchase prices, average cost and stock value fields
Inventory	<code>inventory.adjust</code>	create, approve	Stock adjustments
Inventory	<code>inventory.transfer</code>	view, create, approve, cancel	Transfers between warehouses
Inventory	<code>inventory.count</code>	view, create, approve	Stock counts
Finance	<code>payments</code>	view, create, edit, delete, export, refund, cancel	Customer receipts and supplier payments
Finance	<code>expenses</code>	view, create, edit, delete, export, approve, cancel	Expenses
Finance	<code>revenues</code>	view, create, edit, delete, export, cancel	Other revenues
Finance	<code>cash</code>	view	Cash and bank accounts
Accounting	<code>accounting</code>	view, export	Ledger-based reports (profit & loss, balance sheet, trial balance, ageing)
Reports	<code>reports</code>	view, export	The report engine; each report also needs the scope of its data

Tip

A typical online store needs `products:view`, `customers:view`, `customers:create`, `sales.invoice:view`, `sales.invoice:create`, `sales.invoice:print`, `sales.invoice:refund` and `inventory:view`. Add `inventory.cost:view` only if you really need costs.

What keys can never do

Administrative areas are outside the Developer API whatever the scopes: users and roles, company settings, subscription and billing, ZATCA onboarding, API keys and webhooks management, audit logs, notifications, global search and the POS terminal endpoints. They answer `403 API_KEY_NOT_ALLOWED`. The VAT return report (`vat:view`) is not grantable to keys either.



Conventions

Pagination

Every list returns the same envelope and accepts `page` (from 1) and `pageSize` (1–200, default 25):

List envelope

```
{ "items": [ ... ], "total": 128, "page": 2, "pageSize": 50 }
```

Keep requesting the next page while $\text{page} \times \text{pageSize} < \text{total}$. A `pageSize` above 200 is rejected with `400 VALIDATION_ERROR` (report endpoints allow up to 500).

JavaScript

```
// walk every page (pageSize ≤ 200); stop when a page comes back short
async function* allProducts(filters = {}) {
  for (let page = 1; ; page++) {
    const q = new URLSearchParams({ ...filters, page: String(page), pageSize: '200', sort: 'createdAt', order: 'asc' });
  });
  const res = await sa4s('GET', `/products?${q}`);
  yield* res.items;
  if (page * res.pageSize >= res.total) return;
}
for await (const p of allProducts({ active: 'true' })) console.log(p.sku);
```

Python

```
def all_products(**filters):
    page = 1
    while True:
        res = sa4s("GET", "/products", params={**filters, "page": page, "pageSize": 200, "sort": "createdAt", "order": "asc"})
        yield from res["items"]
        if page * res["pageSize"] >= res["total"]:
            return
        page += 1

for p in all_products(active="true"):
    print(p["sku"])
```

PHP

```
<?php
function all_products(array $filters = []): Generator
{
    for ($page = 1; ; $page++) {
        $res = sa4s('GET', '/products?' . http_build_query($filters + ['page' => $page, 'pageSize' => 200, 'sort' => 'createdAt', 'order' => 'asc']));
        yield from $res['items'];
        if ($page * $res['pageSize'] >= $res['total']) return;
    }
}
foreach (all_products(['active' => 'true']) as $p) echo $p['sku'], "\n";
```



C#

```
static async IEnumerable<JsonElement> AllProducts()
{
    for (int page = 1; ; page++)
    {
        var res = await Sa4s.Send(HttpMethod.Get, $"products?page={page}&pageSize=200&sort=createdAt&order=asc");
        foreach (var p in res.GetProperty("items").EnumerateArray()) yield return p;
        if (page * res.GetProperty("pageSize").GetInt32() >= res.GetProperty("total").GetInt32()) yield break;
    }
}

await foreach (var p in AllProducts()) Console.WriteLine(p.GetProperty("sku"));
```

Sorting

`sort` takes a column name and `order` is `asc` or `desc` (default `desc`). Each list accepts a fixed set of columns; an unknown `sort` silently falls back to the default (usually `createdAt`).

Endpoint	Sortable columns
GET /products	sku, nameAr, nameEn, sellingPrice, minStock, createdAt, updatedAt (+ purchasePrice, cost with inventory.cost:view)
GET /customers	name, code, balance, creditLimit, createdAt
GET /suppliers	nameAr, nameEn, code, balance, creditLimit, createdAt
GET /sales/invoices	createdAt, issueDate, number, total, status, paymentStatus
GET /purchases/invoices	number, date, dueDate, status, total, createdAt
GET /payments	date, number, amount, createdAt
GET /inventory/balances	sku, name, onHand, available, updatedAt, warehouse (+ value, avgCost with cost scope)
GET /inventory/movements	always by time — use <code>order=asc</code> to read oldest first

Search and filters

`search` is a case-insensitive "contains" match on the natural keys of each list:

- products — Arabic/English name, SKU and any barcode;
- customers — name, English name, phone, VAT number, code, e-mail, CR number; suppliers likewise;
- sales invoices — number, notes and the customer's name, phone or VAT number;
- stock balances — SKU and names.

Each list also has exact filters (for example `GET /products?categoryId=...&active=true&type=SERVICE`, `GET /sales/invoices?customerId=...&status=ISSUED&paymentStatus=UNPAID`). Booleans are `true` / `false`. **Unknown query parameters are rejected** with `400 VALIDATION_ERROR` (e.g. `property from should not exist`), so typos never pass silently. The full list per endpoint is in the API reference tab.

Dates and time zone

- Timestamps in responses are ISO 8601 in UTC: `"2026-10-06T17:01:02.289Z"`.
- Date filters given as `YYYY-MM-DD` (`from` , `to` , `dateFrom` , `dateTo` , `asOf`) are calendar days in **Asia/Riyadh** (UTC+3): `to=2026-10-31` includes the whole of 31 October, Riyadh time.
- `GET /sales/invoices` also accepts a full timestamp in `from` / `to` (`from=2026-10-06T14:00:00Z`) — handy for incremental sync.
- In request bodies, date-only fields such as `dueDate` take `YYYY-MM-DD` ; `issueDate` may be a full timestamp.

Money and quantities

Amounts and quantities are **decimal strings** in both directions (`"115.00"` , `"2.5"`). Never parse them into binary floating point for arithmetic — use a decimal type (`decimal` in C#/Python, `bcmath` in PHP, integer halalas in JavaScript). Responses carry the canonical value without trailing zeros (`"270.25"` , `"100"` , `"47.5"`); format for display yourself. All amounts are in the company currency (SAR). Quantities in a line are in the line's unit; `baseQuantity` gives the same quantity in base units.

VAT

Totals are always computed by SA4S — you send lines, never totals (an unknown `total` field is rejected). Each product carries a VAT category:

Category	Meaning	Rate
<code>S</code>	Standard rated	the company rate (15 %) unless the product sets <code>vatRate</code>
<code>Z</code>	Zero rated (e.g. qualifying exports)	0 %
<code>E</code>	Exempt (e.g. some financial services)	0 %, reported separately
<code>O</code>	Out of scope	0 %, not reported

`pricesIncludeVat` says how the unit prices you send are meant: `true` = VAT-inclusive (typical for B2C and online stores), `false` = net prices (typical for B2B). When omitted, the company's default setting applies. Amounts are rounded to 2 decimals **per line** and document totals are the sums of the rounded lines, so printed lines always add up to the printed total:

- VAT-inclusive** (`true`): the line total is `quantity × unitPrice - discounts` (incl. the line's share of a document discount), rounded to 2 decimals — exactly what the customer pays. The VAT is extracted from it, `vatAmount = round(lineTotal × rate / (100 + rate))`, and `taxableAmount = lineTotal - vatAmount`. Example: `2 × 40.00 → lineTotal 80.00 = taxableAmount 69.57 + vatAmount 10.43` ; `3 × 50.00 → 150.00 = 130.43 + 19.57`.
- VAT-exclusive** (`false`): `taxableAmount = round(quantity × unitPrice - discounts)`, `vatAmount = round(taxableAmount × rate / 100)`, `lineTotal = taxableAmount + vatAmount`.

The e-invoice sent to ZATCA carries exactly these line and document amounts (the line VAT and the category VAT stay within the rounding tolerance the Fatoora validation rules allow). `total` in the response is authoritative — charge or compare against it.

**Important**

Line discounts (`discount` , `discountPercent`) and `documentDiscount` need the `sales.discount:manage` scope on the key (sandbox keys may hold it too). Without it they are refused with `403 DISCOUNT_NOT_ALLOWED` ; you can also send the already-discounted `unitPrice` .

Request IDs

Every response carries `X-Request-Id` , and every error body repeats it as `requestId` . Log it; quote it when you contact support. You may send your own `X-Request-Id` header to correlate calls with your logs.

HTTP

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
X-RateLimit-Limit: 300
X-RateLimit-Remaining: 297
X-Request-Id: 7dfe4b4e-31f0-4721-af65-2553d47bc596
```



Idempotency and safe retries

Networks fail: a request may reach SA4S while the response is lost. Retrying a plain `POST` could then create a second invoice. The creates that move money accept a `clientUuid` — a UUID you choose:

Endpoint	Field
<code>POST /sales/invoices</code>	<code>clientUuid</code>
<code>POST /sales/invoices/{id}/credit-note</code>	<code>clientUuid</code>
<code>POST /payments/receipts</code>	<code>clientUuid</code>
<code>POST /payments/disbursements</code>	<code>clientUuid</code>
<code>payments[]</code> inside an invoice	<code>clientUuid</code> per payment

Repeating a request with the same `clientUuid` returns the **original** record (`201`, same `id` and `number`); for invoices the response also says `"idempotentReplay": true`. Nothing is created twice, even when two retries race. Rules of thumb:

1. Generate the UUID **before** the first attempt and store it with your own record (order, transfer...). Better still, derive it from your own id with a UUID v5 so a lost local state still maps to the same key (see the [order sync example](#)).
2. Retry with exactly the same body; a different body with a used `clientUuid` still returns the first record.
3. Retry on network errors, timeouts, `429` and `5xx`. Do not retry other `4xx` — fix the request.
4. Other writes (products, customers, issue/cancel) are not keyed: before retrying them, read back to check whether the first attempt succeeded (e.g. issuing an already issued invoice returns `409 INVOICE_NOT_ISSUABLE`, which you can treat as done).

JavaScript

```
// 1. create the key once, persist it next to the order (DB column, order meta...)
order.sa4sClientUuid ??= randomUUID();
await saveOrder(order);

// 2. any number of attempts with the SAME body + clientUuid – at most one invoice is ever created
const invoice = await sa4s('POST', '/sales/invoices', buildInvoice(order)); // the client helper retries
429/5xx/timeouts
if (invoice.idempotentReplay) console.log('already created earlier:', invoice.number);
```

Python

```
order.setdefault("sa4s_client_uuid", str(uuid.uuid4()))
save_order(order)
invoice = sa4s("POST", "/sales/invoices", build_invoice(order))
if invoice.get("idempotentReplay"):
    print("already created earlier:", invoice["number"])
```

PHP

```
<?php
function uuid4(): string
{
    $b = random_bytes(16);
```



```
$b[6] = chr(ord($b[6]) & 0x0f | 0x40); // version 4
$b[8] = chr(ord($b[8]) & 0x3f | 0x80); // RFC 4122 variant
return vsprintf('%s%s-%s-%s-%s-%s', str_split(bin2hex($b), 4));
}

$order['sa4s_client_uuid'] ??= uuid4();
save_order($order);
$invoice = sa4s('POST', '/sales/invoices', build_invoice($order));
if (!empty($invoice['idempotentReplay'])) echo "already created earlier: {$invoice['number']}\n";
```

C#

```
order.Sa4sClientUuid ??= Guid.NewGuid();
await SaveOrder(order);
var invoice = await Sa4s.Send(HttpMethod.Post, "sales/invoices", BuildInvoice(order));
if (invoice.TryGetProperty("idempotentReplay", out var replay) && replay.GetBoolean())
    Console.WriteLine($"already created earlier: {invoice.GetProperty("number")}");
```

201 Created — replay of the same clientUuid (abridged)

```
{
  "id": "442ea37d-b9ac-4535-b726-b3ad69832afe",
  "number": "INV-2026-000007",
  "status": "ISSUED",
  "total": "46.9",
  "clientUuid": "09c3d295-3127-400b-ab9c-819ab51b011c",
  "idempotentReplay": true
}
```

Rate limits

Each key has its own limit (default **300 requests per minute**, configurable per key from 10 to 5000), measured over a sliding one-minute window. Every response tells you where you stand:

Header	Meaning
X-RateLimit-Limit	requests allowed per minute for this key
X-RateLimit-Remaining	requests left in the current window
Retry-After	on 429 only: seconds to wait before the next attempt

Above the limit the API answers 429 RATE_LIMITED (the request was not processed, so retrying is always safe). A per-IP protection limit also applies to all traffic. The recipe: honour Retry-After; otherwise back off exponentially with jitter (0.5 s, 1 s, 2 s, 4 s... capped at 30 s); spread bulk jobs evenly; and prefer webhooks to polling.

HTTP

```
HTTP/1.1 429 Too Many Requests
X-RateLimit-Limit: 10
X-RateLimit-Remaining: 0
Retry-After: 60
Content-Type: application/json; charset=utf-8

{"statusCode":429,"code":"RATE_LIMITED","message":"Rate limit of 10 requests per minute exceeded","details":
{"limit":10,"retryAfterSeconds":60},"requestId":"71b8d797-785b-416a-b56c-9bf23f98011c"}
```

JavaScript

```
// spread a bulk job under the key's limit instead of hitting 429 (e.g. 300/min → one call every 200 ms)
const LIMIT_PER_MINUTE = 300;
const gap = 60_000 / LIMIT_PER_MINUTE;
for (const sku of skus) {
  const started = Date.now();
  await syncOne(sku);
  await new Promise((r) => setTimeout(r, Math.max(0, gap - (Date.now() - started))));
}
```

Python

```
import time

LIMIT_PER_MINUTE = 300
gap = 60 / LIMIT_PER_MINUTE
for sku in skus:
    started = time.monotonic()
    sync_one(sku)
    time.sleep(max(0, gap - (time.monotonic() - started)))
```

PHP

```
<?php
$gapUs = (int) (60_000_000 / 300);
foreach ($skus as $sku) {
    $t = hrtime(true);
```



```
sync_one($sku);  
usleep(max(0, $gapUs - (int) ((hrtime(true) - $t) / 1000)));  
}
```

C#

```
var gap = TimeSpan.FromMinutes(1) / 300;  
foreach (var sku in skus)  
{  
    var started = DateTime.UtcNow;  
    await SyncOne(sku);  
    var wait = gap - (DateTime.UtcNow - started);  
    if (wait > TimeSpan.Zero) await Task.Delay(wait);  
}
```

Errors

Error body

Errors use standard HTTP status codes and always the same JSON body. Branch on `code` (stable, UPPER_SNAKE) — never on `message`, which is English diagnostic text that may change. `details` carries code-specific data such as the missing scope or the validation messages.

403 Forbidden

```
{
  "statusCode": 403,
  "code": "PERMISSION_DENIED",
  "message": "You do not have permission to perform this action",
  "details": { "missing": ["products:create"] },
  "requestId": "f20f408d-cc0a-4d0f-9f92-e1d989bfb96e"
}
```

400 Bad Request

```
{
  "statusCode": 400,
  "code": "VALIDATION_ERROR",
  "message": "Validation failed",
  "details": ["property total should not exist"],
  "requestId": "aca1aa3b-cbc9-4d07-912a-121298863808"
}
```

JavaScript

```
import { sa4s, Sa4sError } from './sa4s.mjs';

export async function pushInvoice(order, invoice) {
  try {
    return await sa4s('POST', '/sales/invoices', invoice);
  } catch (e) {
    if (!(e instanceof Sa4sError)) throw e; // network problem after all retries
    switch (e.code) {
      case 'INSUFFICIENT_STOCK': return holdOrder(order, e.details); // business rule: fix data, do not retry
      blindly
      case 'VALIDATION_ERROR': return logBug(e.details); // your payload is wrong
      case 'PERMISSION_DENIED': return alertAdmin(`grant ${e.details.missing.join(', ')}`);
      case 'API_KEY_REVOKED': case 'API_KEY_EXPIRED': return stopSync('key no longer valid');
      default: throw new Error(`${e.message} (requestId ${e.requestId})`);
    }
  }
}
```

Python

```
from sa4s import sa4s, Sa4sError

try:
    sa4s("POST", "/sales/invoices", invoice)
except Sa4sError as e:
    if e.code == "INSUFFICIENT_STOCK":
        hold_order(order, e.details)
```



```
elif e.code == "PERMISSION_DENIED":
    alert_admin("grant " + " ".join(e.details["missing"]))
else:
    raise RuntimeError(f"{e} (requestId {e.request_id})") from e
```

PHP

```
<?php
try {
    sa4s('POST', '/sales/invoices', $invoice);
} catch (Sa4sError $e) {
    match ($e->body['code'] ?? null) {
        'INSUFFICIENT_STOCK' => hold_order($order, $e->body['details']),
        'PERMISSION_DENIED' => alert_admin('grant ' . implode(' ', $e->body['details']['missing'])),
        default => throw new RuntimeException($e->getMessage() . ' (requestId ' . ($e->body['requestId'] ?? '?') . ')',
0, $e),
    };
}
```

C#

```
try
{
    await Sa4s.Send(HttpMethod.Post, "sales/invoices", invoice);
}
catch (Sa4sException e) when (e.Body.GetProperty("code").GetString() == "INSUFFICIENT_STOCK")
{
    await HoldOrder(order, e.Body.GetProperty("details"));
}
```

Error codes

Codes an API-key client can receive. Business codes come with `400`, `404` or `409`.

Code	Status	Meaning	What to do
UNAUTHORIZED	401	No key (and no session) was sent	Send <code>Authorization: Bearer sk_...</code> or <code>X-API-Key</code>
API_KEY_INVALID	401	Malformed or unknown key, or prefix/environment mismatch	Check the secret was copied completely
API_KEY_REVOKED	401	The key was revoked	Create a new key
API_KEY_EXPIRED	401	Past its expiry date (also the old secret 24 h after a rotation)	Use the new / a new key
API_KEY_IP_DENIED	401	Caller address not in the key's allow-list	Add the server's address or CIDR
FEATURE_NOT_AVAILABLE	403	The plan does not include the API (or the module, e.g. reports)	Upgrade the plan
COMPANY_SUSPENDED	403	The company account is suspended	Contact the account owner
API_KEY_NOT_ALLOWED	403	The endpoint is not part of the Developer API	Use only documented endpoints



Code	Status	Meaning	What to do
<code>PERMISSION_DENIED</code>	403	A scope is missing (<code>details.missing</code>)	Edit the key and add the scope
<code>SANDBOX_READ_ONLY</code>	403	A sandbox key tried to change data	Create drafts only, or use a live key
<code>DISCOUNT_NOT_ALLOWED</code>	403	Discount fields sent by a key without <code>sales.discount:manage</code>	Grant the scope, or send the discounted <code>unitPrice</code>
<code>SUBSCRIPTION_EXPIRED</code>	402	Subscription ended — writes are blocked, reads still work	Renew the subscription
<code>PLAN_LIMIT_REACHED</code>	402	A plan quota is used up (e.g. invoices per month; <code>details.key</code> , <code>details.limit</code>)	Upgrade or wait for the next period
<code>RATE_LIMITED</code>	429	Too many requests	Wait <code>Retry-After</code> seconds
<code>VALIDATION_ERROR</code>	400	Body or query invalid; unknown fields are rejected (<code>details</code> lists the messages)	Fix the request
<code>BAD_REQUEST</code>	400	Malformed path parameter (e.g. an id that is not a UUID)	Send a valid id
<code>NOT_FOUND</code>	404	No such record in your company	Check the id
<code>DUPLICATE</code>	409	A unique value already exists (<code>details.fields</code>)	Use another value or update the existing record
<code>CONFLICT</code> / <code>IN_USE</code>	409	The record changed meanwhile / is referenced by other data	Re-read and retry, or leave it
<code>SKU_TAKEN</code> , <code>BARCODE_TAKEN</code> , <code>CODE_TAKEN</code> , <code>VAT_NUMBER_TAKEN</code>	409	The value belongs to another record (<code>details</code> names it)	Update that record instead
<code>BARCODE_NOT_FOUND</code>	404	<code>GET /products/lookup</code> found nothing	Treat as an unknown product
<code>COST_PERMISSION_REQUIRED</code>	403	<code>purchasePrice</code> sent without <code>inventory.cost:view</code>	Add the scope or drop the field
<code>PRODUCT_NOT_FOUND</code> , <code>PRODUCT_INACTIVE</code> , <code>UNIT_NOT_FOUND</code>	400/409	A line points to a missing/inactive product or a unit it does not have	Fix the line
<code>CUSTOMER_REQUIRED</code> , <code>CUSTOMER_NOT_FOUND</code> , <code>CUSTOMER_INACTIVE</code>	400/409	A tax invoice needs a valid active customer	Send <code>customerId</code>
<code>INSUFFICIENT_STOCK</code>	409	Not enough stock (<code>details.available</code> , <code>details.requested</code>)	Restock, or change the quantity
<code>CREDIT_LIMIT_EXCEEDED</code>	409	The sale exceeds the customer's credit limit	Collect payment or raise the limit



Code	Status	Meaning	What to do
PAYMENT_REQUIRES_ISSUED	400	Payments sent with a draft	Send <code>status: "ISSUED"</code> or pay when issuing
OVERPAYMENT	409	Payments exceed the invoice total / open balance	Pay at most <code>total</code>
INVOICE_NOT_ISSUABLE	409	The invoice is already issued or cancelled	Treat as done (idempotent issue)
INVOICE_HAS_PAYMENTS , INVOICE_HAS_NOTES	409	Cancel blocked by payments / credit notes	Issue a credit note instead
ZATCA_REPORTED_USE_CREDIT_NOTE	409	The invoice already went to ZATCA	Issue a credit note
ALLOCATION_EXCEEDS_PAYMENT	400	Allocations add up to more than the payment	Fix the allocation amounts
INVOICE_NOT_PAYABLE , INVOICE_PARTY_MISMATCH	409	Allocation to an unissued invoice / another party's invoice	Allocate to that party's open invoices
INSUFFICIENT_FUNDS	409	The paying cash/bank account balance is too low	Choose another <code>accountId</code>
INVOICE_NOT_DRAFT , DUPLICATE_SUPPLIER_INVOICE	409	Purchase invoice already posted / supplier number already used	Read the existing document
REPORT_NOT_FOUND	404	Unknown report name (<code>details.available</code> lists them)	Use a name from <code>GET /reports</code>
INTERNAL_ERROR	500	Unexpected failure on our side	Retry with backoff; report the <code>requestId</code>

409 Conflict

```
{
  "statusCode": 409,
  "code": "INSUFFICIENT_STOCK",
  "message": "Insufficient stock for Sukkari dates 1 kg",
  "details": { "productId": "f24f96e1-7e8a-4ba6-8fc3-c1f235a16f41", "warehouseId": "6f46787b-84c8-453c-a722-7ed26ae45313", "available": "45", "requested": "100000" },
  "requestId": "d3601ec7-6c3c-4c8d-ade8-50d7bdcfaf06"
}
```

Sandbox

Sandbox keys (`sk_test_...`) let you build and test an integration against your **real company data** without side effects:

- every `GET` works with the key's view/print/export scopes (PDFs and report files included);
- `POST` is allowed only to create **drafts** of: sales invoices, purchase requests, orders, receipts, invoices and returns, and stock adjustments, transfers and counts — with `status` omitted or `"DRAFT"` and without `post: true`;
- everything else — creating customers or products, issuing, posting, payments, cancelling, updates and deletes — answers `403 SANDBOX_READ_ONLY`;
- scopes other than view, create, print and export are removed from sandbox keys automatically.

Drafts created with a sandbox key are ordinary drafts: they appear in the ERP and can be deleted there. When the integration works, create a live key with the same scopes and switch the secret.

403 Forbidden — a sandbox key tried to issue an invoice

```
{
  "statusCode": 403,
  "code": "SANDBOX_READ_ONLY",
  "message": "Sandbox keys can only read data and create DRAFT documents",
  "details": {
    "allowed": ["/sales/invoices", "/purchases/requests", "/purchases/orders", "/purchases/receipts",
"/purchases/invoices", "/purchases/returns", "/inventory/adjustments", "/inventory/transfers", "/inventory/counts"]
  },
  "requestId": "22bb9db1-0f29-48d7-bad4-fb18a5330f7a"
}
```



Recipes — products

Each recipe shows the request in curl and in the client of your language (see [A reusable client](#)), followed by a real response from the API (long responses are abridged; fields not shown are still returned).

List and search products

GET /products — scope `products:view`. Filters: `search`, `categoryId` (includes sub-categories), `brandId`, `active`, `type` (`PHYSICAL`, `SERVICE`, `COMPOSITE`, `BUNDLE`), `lowStock`, `warehouseId` (limit the `stock` figures to one warehouse).

curl

```
curl "https://erp.sa4s.com/api/v1/products?page=1&pageSize=20&search=water&sort=nameEn&order=asc&active=true" \
-H "Authorization: Bearer $SA4S_API_KEY"
```

JavaScript

```
import { sa4s } from './sa4s.mjs';

const q = new URLSearchParams({ page: '1', pageSize: '20', search: 'water', sort: 'nameEn', order: 'asc', active: 'true' });
const { items, total } = await sa4s('GET', `/products?${q}`);
console.log(`${items.length} of ${total}`, items.map(p => `${p.sku} ${p.nameEn} ${p.sellingPrice}`));
```

Python

```
from sa4s import sa4s

page = sa4s("GET", "/products", params={"page": 1, "pageSize": 20, "search": "water", "sort": "nameEn", "order": "asc", "active": "true"})
for p in page["items"]:
    print(p["sku"], p["nameEn"], p["sellingPrice"], p["stock"]["available"])
```

PHP

```
<?php
require 'sa4s.php';

$q = http_build_query(['page' => 1, 'pageSize' => 20, 'search' => 'water', 'sort' => 'nameEn', 'order' => 'asc', 'active' => 'true']);
$page = sa4s('GET', "/products?$q");
foreach ($page['items'] as $p) {
    echo "{$p['sku']} {$p['nameEn']} {$p['sellingPrice']}\n";
}
```

C#

```
var page = await Sa4s.Send(HttpMethod.Get, "products?page=1&pageSize=20&search=water&sort=nameEn&order=asc&active=true");
foreach (var p in page.GetProperty("items").EnumerateArray())
    Console.WriteLine($"{p.GetProperty("sku")} {p.GetProperty("nameEn")} {p.GetProperty("sellingPrice")}");
```

**200 OK (abridged)**

```
{
  "items": [
    {
      "id": "51f771bf-cfb5-4814-8e79-55b990be69c9",
      "sku": "SKU-2026-000001",
      "nameAr": "مياه معدنية 330 مل",
      "nameEn": "Mineral water 330 ml",
      "type": "PHYSICAL",
      "categoryId": "c33a89cb-5d9d-4112-b1cc-4dfebc304c7a",
      "baseUnitId": "0757ef16-15fa-4b58-996d-ebb3d8bb5fcb",
      "sellingPrice": "1.15",
      "purchasePrice": "0.6",
      "cost": "0.6",
      "vatCategory": "S",
      "vatRate": null,
      "minStock": "48",
      "isActive": true,
      "trackInventory": true,
      "category": { "id": "c33a89cb-5d9d-4112-b1cc-4dfebc304c7a", "nameAr": "مشروبات", "nameEn": "Beverages" },
      "baseUnit": { "id": "0757ef16-15fa-4b58-996d-ebb3d8bb5fcb", "code": "PCS", "nameAr": "قطعة", "nameEn": "Piece" },
      "units": [
        { "unitId": "0757ef16-15fa-4b58-996d-ebb3d8bb5fcb", "factor": "1", "isDefaultSale": true, "sellingPrice": null,
        "unit": { "code": "PCS" } },
        { "unitId": "b82dbc97-8f7b-43c0-8b73-8b4aed1ebdb4", "factor": "24", "isDefaultPurchase": true, "sellingPrice":
        "25", "unit": { "code": "BOX" } }
      ],
      "barcodes": [
        { "code": "6281000616970", "symbology": "EAN13", "unitId": null, "isPrimary": true },
        { "code": "6281010616977", "symbology": "EAN13", "unitId": "b82dbc97-8f7b-43c0-8b73-8b4aed1ebdb4", "isPrimary":
        false }
      ],
      "stock": { "onHand": "282", "reserved": "0", "available": "282" },
      "createdAt": "2026-10-06T17:01:01.711Z",
      "updatedAt": "2026-10-06T17:01:01.711Z"
    }
  ],
  "total": 1,
  "page": 1,
  "pageSize": 20
}
```

Create a product with units and barcodes

POST /products — scope products:create (purchasePrice also needs inventory.cost:view). Only nameAr is required: nameEn defaults to it, the base unit defaults to PCS, and the SKU is numbered automatically (SKU-2026-000001) when omitted. units adds alternative units with their factor; barcodes may be attached to a unit (the carton code sells the whole carton); the symbology (EAN-13, EAN-8, UPC, Code 128, QR) is detected from the value.

curl

```
# the ids of PCS, BOX ... (every company starts with a standard set of units)
curl "https://erp.sa4s.com/api/v1/units?pageSize=50" -H "Authorization: Bearer $SA4S_API_KEY"
```

curl

```
curl -X POST "https://erp.sa4s.com/api/v1/products" \
-H "Authorization: Bearer $SA4S_API_KEY" \
-H "Content-Type: application/json" \
-d '{
```



```

"nameAr": "مياه معدنية 330 مل",
"nameEn": "Mineral water 330 ml",
"categoryId": "c33a89cb-5d9d-4112-b1cc-4dfebc304c7a",
"baseUnitId": "0757ef16-15fa-4b58-996d-ebb3d8bb5fcb",
"sellingPrice": "1.15",
"purchasePrice": "0.60",
"vatCategory": "S",
"minStock": "48",
"units": [
  { "unitId": "b82dbc97-8f7b-43c0-8b73-8b4aed1ebdb4", "factor": "24", "sellingPrice": "25.00", "isDefaultPurchase":
true }
],
"barcodes": [
  { "code": "6281000616970", "isPrimary": true },
  { "code": "6281010616977", "unitId": "b82dbc97-8f7b-43c0-8b73-8b4aed1ebdb4" }
]
}'

```

JavaScript

```

const units = await sa4s('GET', '/units?pageSize=50');
const pcs = units.items.find((u) => u.code === 'PCS');
const box = units.items.find((u) => u.code === 'BOX');

const product = await sa4s('POST', '/products', {
  nameAr: 'مياه معدنية 330 مل',
  nameEn: 'Mineral water 330 ml',
  baseUnitId: pcs.id,
  sellingPrice: '1.15',           // per base unit (piece)
  purchasePrice: '0.60',         // needs the inventory.cost:view scope
  vatCategory: 'S',
  minStock: '48',
  units: [{ unitId: box.id, factor: '24', sellingPrice: '25.00', isDefaultPurchase: true }], // 1 box = 24 pieces
  barcodes: [{ code: '6281000616970', isPrimary: true }, { code: '6281010616977', unitId: box.id }],
});
console.log(product.id, product.sku); // SKU is numbered automatically when omitted

```

Python

```

units = {u["code"]: u["id"] for u in sa4s("GET", "/units", params={"pageSize": 50})["items"]}

product = sa4s("POST", "/products", {
    "nameAr": "مياه معدنية 330 مل",
    "nameEn": "Mineral water 330 ml",
    "baseUnitId": units["PCS"],
    "sellingPrice": "1.15",
    "purchasePrice": "0.60",
    "vatCategory": "S",
    "minStock": "48",
    "units": [{"unitId": units["BOX"], "factor": "24", "sellingPrice": "25.00", "isDefaultPurchase": True}],
    "barcodes": [{"code": "6281000616970", "isPrimary": True}, {"code": "6281010616977", "unitId": units["BOX"]}],
})
print(product["id"], product["sku"])

```

PHP

```

<?php
$units = array_column(sa4s('GET', '/units?pageSize=50')['items'], 'id', 'code');

$product = sa4s('POST', '/products', [
    'nameAr' => 'مياه معدنية 330 مل',
    'nameEn' => 'Mineral water 330 ml',
    'baseUnitId' => $units['PCS'],
    'sellingPrice' => '1.15',

```



```
'purchasePrice' => '0.60',
'vatCategory' => 'S',
'minStock' => '48',
'units' => [['unitId' => $units['BOX'], 'factor' => '24', 'sellingPrice' => '25.00', 'isDefaultPurchase' => true]],
'barcodes' => [['code' => '6281000616970', 'isPrimary' => true], ['code' => '6281010616977', 'unitId' =>
$units['BOX']]],
]);
echo $product['id'], ' ', $product['sku'], "\n";
```

C#

```
var product = await Sa4s.Send(HttpMethod.Post, "products", new
{
    nameAr = "مياه معدنية 330 مل",
    nameEn = "Mineral water 330 ml",
    baseUnitId = pcsUnitId,
    sellingPrice = "1.15",
    purchasePrice = "0.60",
    vatCategory = "S",
    minStock = "48",
    units = new[] { new { unitId = boxUnitId, factor = "24", sellingPrice = "25.00", isDefaultPurchase = true } },
    barcodes = new object[] { new { code = "6281000616970", isPrimary = true }, new { code = "6281010616977", unitId =
boxUnitId } },
});
Console.WriteLine(product.GetProperty("sku"));
```

201 Created (abridged)

```
{
  "id": "51f771bf-cfb5-4814-8e79-55b990be69c9",
  "sku": "SKU-2026-000001",
  "nameAr": "مياه معدنية 330 مل",
  "nameEn": "Mineral water 330 ml",
  "type": "PHYSICAL",
  "baseUnitId": "0757ef16-15fa-4b58-996d-ebb3d8bb5fcb",
  "purchasePrice": "0.6",
  "sellingPrice": "1.15",
  "cost": "0",
  "vatCategory": "S",
  "minStock": "48",
  "isActive": true,
  "trackInventory": true,
  "units": [
    { "unitId": "0757ef16-15fa-4b58-996d-ebb3d8bb5fcb", "factor": "1", "isDefaultSale": true, "unit": { "code": "PCS",
"nameEn": "Piece" } },
    { "unitId": "b82dbc97-8f7b-43c0-8b73-8b4aed1ebdb4", "factor": "24", "isDefaultPurchase": true, "sellingPrice":
"25", "unit": { "code": "BOX", "nameEn": "Box" } }
  ],
  "barcodes": [
    { "id": "5e5043c6-49e1-4eac-b89f-8f147cb20f6f", "code": "6281000616970", "symbology": "EAN13", "unitId": null,
"isPrimary": true },
    { "id": "7f6457f4-1571-4fe4-aca7-3b8927691c48", "code": "6281010616977", "symbology": "EAN13", "unitId": "b82dbc97-
8f7b-43c0-8b73-8b4aed1ebdb4", "isPrimary": false }
  ],
  "stock": { "onHand": "0", "reserved": "0", "available": "0" },
  "stockByWarehouse": []
}
```

Re-using a barcode or SKU fails with a conflict that names the current owner:



409 Conflict

```
{
  "statusCode": 409,
  "code": "BARCODE_TAKEN",
  "message": "This barcode is already assigned to another product",
  "details": { "code": "6281000616970", "productId": "51f771bf-cfb5-4814-8e79-55b990be69c9", "sku": "SKU-2026-000001" },
  "requestId": "5ecd4f0d-256e-4818-afef-23dd38d3bc1d"
}
```

Update a product

`PATCH /products/{id}` — scope `products:edit`. Send only what changes. When you do send `units` or `barcodes`, the list **replaces** the existing one; to add a single barcode use `POST /products/{id}/barcodes`.

curl

```
# change the price only – fields you leave out are not touched
curl -X PATCH "https://erp.sa4s.com/api/v1/products/51f771bf-cfb5-4814-8e79-55b990be69c9" \
-H "Authorization: Bearer $SA4S_API_KEY" -H "Content-Type: application/json" \
-d '{ "sellingPrice": "1.25" }'

# add one more barcode without resending the list
curl -X POST "https://erp.sa4s.com/api/v1/products/51f771bf-cfb5-4814-8e79-55b990be69c9/barcodes" \
-H "Authorization: Bearer $SA4S_API_KEY" -H "Content-Type: application/json" \
-d '{ "code": "6281020618633" }'
```

JavaScript

```
await sa4s('PATCH', `/products/${productId}`, { sellingPrice: '1.25' });
await sa4s('POST', `/products/${productId}/barcodes`, { code: '6281020618633' });
```

Python

```
sa4s("PATCH", f"/products/{product_id}", {"sellingPrice": "1.25"})
sa4s("POST", f"/products/{product_id}/barcodes", {"code": "6281020618633"})
```

PHP

```
<?php
sa4s('PATCH', "/products/$productId", ['sellingPrice' => '1.25']);
sa4s('POST', "/products/$productId/barcodes", ['code' => '6281020618633']);
```

Look up a product by barcode

`GET /products/lookup?code=...` — scope `products:view`. Matches a barcode first, then a SKU. The answer tells you which unit the code sells (a carton barcode returns the carton unit and its factor) and the stock, optionally for one `warehouseId`.

curl

```
curl "https://erp.sa4s.com/api/v1/products/lookup?code=6281010616977&warehouseId=1ae3935b-6fae-4c7f-803a-bf405dbc0ea0" \
-H "Authorization: Bearer $SA4S_API_KEY"
```



JavaScript

```
try {
  const hit = await sa4s('GET', `/products/lookup?code=${encodeURIComponent(scanned)}`);
  console.log(hit.matchedBy, hit.product.nameEn, hit.unit?.code, hit.unit?.factor, hit.stock.available);
} catch (e) {
  if (e.code === 'BARCODE_NOT_FOUND') console.log('unknown code'); else throw e;
}
```

Python

```
from sa4s import sa4s, Sa4sError

try:
    hit = sa4s("GET", "/products/lookup", params={"code": scanned})
    print(hit["matchedBy"], hit["product"]["nameEn"], hit["unit"]["code"], hit["stock"]["available"])
except Sa4sError as e:
    if e.code != "BARCODE_NOT_FOUND":
        raise
```

PHP

```
<?php
try {
    $hit = sa4s('GET', '/products/lookup?code=' . urlencode($scanned));
    echo $hit['matchedBy'], ' ', $hit['product']['nameEn'], ' ', $hit['unit']['code'], "\n";
} catch (Sa4sError $e) {
    if (($e->body['code'] ?? '') != 'BARCODE_NOT_FOUND') throw $e;
}
```

200 OK (abridged)

```
{
  "matchedBy": "barcode",
  "product": { "id": "51f771bf-cfb5-4814-8e79-55b990be69c9", "sku": "SKU-2026-000001", "nameEn": "Mineral water 330
mL", "sellingPrice": "1.15", "vatCategory": "S" },
  "barcode": { "id": "7f6457f4-1571-4fe4-aca7-3b8927691c48", "code": "6281010616977", "symbology": "EAN13", "unitId":
"b82dbc97-8f7b-43c0-8b73-8b4aed1ebdb4", "isPrimary": false },
  "unit": { "unitId": "b82dbc97-8f7b-43c0-8b73-8b4aed1ebdb4", "code": "BOX", "nameAr": "علبة", "nameEn": "Box",
"factor": "24", "sellingPrice": "25", "purchasePrice": "14.4" },
  "stock": { "onHand": "282", "reserved": "0", "available": "282" }
}
```

404 Not Found

```
{
  "statusCode": 404,
  "code": "BARCODE_NOT_FOUND",
  "message": "No product matches this code",
  "details": { "code": "000000000000" },
  "requestId": "d7c8deca-f7ac-4f0f-9d28-effe98b2a121"
}
```

Recipes — customers

Create a customer

`POST /customers` — scope `customers:create`. Only `name` is required. A customer with a `vatNumber` (15 digits, starting and ending with 3) gets tax (B2B) invoices by default. The national-address fields are used on tax invoices. `code` is numbered automatically (`CUS-2026-000001`).

curl

```
curl -X POST "https://erp.sa4s.com/api/v1/customers" \
-H "Authorization: Bearer $SA4S_API_KEY" -H "Content-Type: application/json" \
-d '{
  "type": "COMPANY",
  "name": "شركة الأفق للتجارة",
  "nameEn": "Horizon Trading Co",
  "vatNumber": "310123456700003",
  "crNumber": "1010123456",
  "phone": "+966500000001",
  "email": "ap@horizon.example",
  "city": "Riyadh", "district": "Al Olaya", "street": "King Fahd Rd",
  "buildingNumber": "1234", "postalCode": "12211", "additionalNumber": "5678",
  "creditLimit": "50000.00",
  "paymentTermsDays": 30
}'
```

JavaScript

```
const customer = await sa4s('POST', '/customers', {
  type: 'COMPANY', // INDIVIDUAL (default) | COMPANY | GOVERNMENT
  name: 'شركة الأفق للتجارة', // primary (Arabic) name – required
  nameEn: 'Horizon Trading Co',
  vatNumber: '310123456700003', // 15 digits, starts and ends with 3 → invoices default to TAX (B2B)
  crNumber: '1010123456',
  phone: '+966500000001',
  email: 'ap@horizon.example',
  city: 'Riyadh', district: 'Al Olaya', street: 'King Fahd Rd', buildingNumber: '1234', postalCode: '12211',
  additionalNumber: '5678',
  creditLimit: '50000.00',
  paymentTermsDays: 30,
});
console.log(customer.code); // CUS-2026-000001
```

Python

```
customer = sa4s("POST", "/customers", {
    "type": "COMPANY",
    "name": "شركة الأفق للتجارة",
    "nameEn": "Horizon Trading Co",
    "vatNumber": "310123456700003",
    "phone": "+966500000001",
    "email": "ap@horizon.example",
    "city": "Riyadh", "district": "Al Olaya", "street": "King Fahd Rd",
    "buildingNumber": "1234", "postalCode": "12211", "additionalNumber": "5678",
    "creditLimit": "50000.00",
    "paymentTermsDays": 30,
})
print(customer["id"], customer["code"])
```



PHP

```
<?php
$customer = sa4s('POST', '/customers', [
    'type' => 'COMPANY',
    'name' => 'شركة الأفق للتجارة',
    'nameEn' => 'Horizon Trading Co',
    'vatNumber' => '310123456700003',
    'phone' => '+966500000001',
    'email' => 'ap@horizon.example',
    'city' => 'Riyadh', 'district' => 'Al Olaya', 'street' => 'King Fahd Rd',
    'buildingNumber' => '1234', 'postalCode' => '12211', 'additionalNumber' => '5678',
    'creditLimit' => '50000.00',
    'paymentTermsDays' => 30,
]);
echo $customer['code'], "\n";
```

C#

```
var customer = await Sa4s.Send(HttpMethod.Post, "customers", new
{
    type = "COMPANY",
    name = "شركة الأفق للتجارة",
    nameEn = "Horizon Trading Co",
    vatNumber = "310123456700003",
    phone = "+966500000001",
    city = "Riyadh", district = "Al Olaya", street = "King Fahd Rd", buildingNumber = "1234", postalCode = "12211",
    creditLimit = "50000.00",
    paymentTermsDays = 30,
});
Console.WriteLine(customer.GetProperty("code"));
```

201 Created

```
{
  "id": "fab57752-483f-4b1a-8d80-8c0401809ffe",
  "code": "CUS-2026-000001",
  "type": "COMPANY",
  "name": "شركة الأفق للتجارة",
  "nameEn": "Horizon Trading Co",
  "vatNumber": "310123456700003",
  "crNumber": "1010123456",
  "phone": "+966500000001",
  "email": "ap@horizon.example",
  "country": "SA",
  "city": "Riyadh",
  "district": "Al Olaya",
  "street": "King Fahd Rd",
  "buildingNumber": "1234",
  "postalCode": "12211",
  "additionalNumber": "5678",
  "creditLimit": "50000",
  "paymentTermsDays": 30,
  "openingBalance": "0",
  "balance": "0",
  "isWalkIn": false,
  "isActive": true,
  "notes": null,
  "createdAt": "2026-10-06T17:01:01.916Z",
  "updatedAt": "2026-10-06T17:01:01.916Z",
}
```



```
"deletedAt": null,
"availableCredit": "50000",
"overCreditLimit": false
}
```

409 Conflict

```
{
  "statusCode": 409,
  "code": "VAT_NUMBER_TAKEN",
  "message": "Another record already uses this VAT number",
  "details": { "vatNumber": "310123456700003" },
  "requestId": "175ebe07-cc59-447f-a30e-2106c84d503f"
}
```

Find or update a customer

GET /customers?search=... (scope `customers:view`) and PATCH /customers/{id} (scope `customers:edit`, send only the changed fields). A common pattern for stores is *find by phone or VAT number, create when missing*:

curl

```
# search matches name, English name, phone, VAT number, code, e-mail and CR number
curl "https://erp.sa4s.com/api/v1/customers?search=%2B966551234567&pageSize=5" \
-H "Authorization: Bearer $SA4S_API_KEY"
```

JavaScript

```
// find a store customer by phone (or VAT number), create them when missing
async function upsertCustomer({ name, phone, email, vatNumber }) {
  const key = vatNumber ?? phone;
  const found = await sa4s('GET', `/customers?search=${encodeURIComponent(key)}&pageSize=5`);
  const match = found.items.find((c) => (vatNumber ? c.vatNumber === vatNumber : c.phone === phone));
  if (match) return sa4s('PATCH', `/customers/${match.id}`, { email }); // keep contact data fresh
  return sa4s('POST', '/customers', { name, phone, email, vatNumber, type: vatNumber ? 'COMPANY' : 'INDIVIDUAL' });
}
```

Python

```
def upsert_customer(name, phone, email=None, vat_number=None):
    key = vat_number or phone
    found = sa4s("GET", "/customers", params={"search": key, "pageSize": 5})["items"]
    match = next((c for c in found if (c["vatNumber"] == vat_number if vat_number else c["phone"] == phone)), None)
    if match:
        return sa4s("PATCH", f"/customers/{match['id']}", {"email": email})
    body = {"name": name, "phone": phone, "email": email, "type": "COMPANY" if vat_number else "INDIVIDUAL"}
    if vat_number:
        body["vatNumber"] = vat_number
    return sa4s("POST", "/customers", body)
```

PHP

```
<?php
function upsert_customer(string $name, string $phone, ?string $email = null, ?string $vat = null): array
{
    $found = sa4s('GET', '/customers?' . http_build_query(['search' => $vat ?? $phone, 'pageSize' => 5]))['items'];
    foreach ($found as $c) {
        if ($vat ? $c['vatNumber'] === $vat : $c['phone'] === $phone) {
            return sa4s('PATCH', "/customers/{$c['id']}", ['email' => $email]);
        }
    }
}
```



```
return sa4s('POST', '/customers', array_filter(['name' => $name, 'phone' => $phone, 'email' => $email, 'vatNumber' => $vat, 'type' => $vat ? 'COMPANY' : 'INDIVIDUAL']));
}
```

curl

```
curl -X PATCH "https://erp.sa4s.com/api/v1/customers/b4382d00-46be-436d-af45-c3dcd8b0222c" \
-H "Authorization: Bearer $SA4S_API_KEY" -H "Content-Type: application/json" \
-d '{ "phone": "+966551234568", "city": "Jeddah" }'
```

JavaScript

```
await sa4s('PATCH', `/customers/${customerId}`, { phone: '+966551234568', city: 'Jeddah' });
```

Python

```
sa4s("PATCH", f"/customers/{customer_id}", {"phone": "+966551234568", "city": "Jeddah"})
```

PHP

```
<?php
sa4s('PATCH', "/customers/$customerId", ['phone' => '+966551234568', 'city' => 'Jeddah']);
```

Customer statement

GET /customers/{id}/statement?from=...&to=... — scope `customers:view`: opening balance, every invoice and payment with a running balance, and the ageing of open items.

curl

```
curl "https://erp.sa4s.com/api/v1/customers/fab57752-483f-4b1a-8d80-8c0401809ffe/statement?from=2026-10-01&to=2026-10-31" \
-H "Authorization: Bearer $SA4S_API_KEY"
```

JavaScript

```
const st = await sa4s('GET', `/customers/${customerId}/statement?from=2026-10-01&to=2026-10-31`);
console.log(st.openingBalance, st.closingBalance, st.outstanding.aging);
```

Python

```
st = sa4s("GET", f"/customers/{customer_id}/statement", params={"from": "2026-10-01", "to": "2026-10-31"})
```

PHP

```
<?php
$st = sa4s('GET', "/customers/$customerId/statement?from=2026-10-01&to=2026-10-31");
```

200 OK (abridged)

```
{
  "kind": "CUSTOMER",
  "party": { "id": "fab57752-483f-4b1a-8d80-8c0401809ffe", "code": "CUS-2026-000001", "name": "شركة الأفق للتجارة", "vatNumber": "310123456700003" },
  "currency": "SAR",
  "from": "2026-10-01",
  "to": "2026-10-31",
```



```
"openingBalance": "0.00",
"lines": [
  { "date": "2026-10-06T17:01:02.289Z", "type": "INVOICE", "refType": "SALES_INVOICE", "number": "INV-2026-000002",
    "debit": "270.25", "credit": "0.00", "balance": "270.25" },
  { "date": "2026-10-06T17:01:02.330Z", "type": "PAYMENT", "refType": "PAYMENT", "number": "RCV-2026-000002",
    "description": "BANK_TRANSFER · TRX-55120", "debit": "0.00", "credit": "100.00", "balance": "170.25" },
  { "date": "2026-10-06T17:01:02.393Z", "type": "INVOICE", "refType": "SALES_INVOICE", "number": "INV-2026-000003",
    "debit": "230.00", "credit": "0.00", "balance": "400.25" }
],
"totals": { "debit": "500.25", "credit": "300.25" },
"closingBalance": "200.00",
"outstanding": {
  "total": "200.00",
  "aging": { "current": "200.00", "d1_30": "0.00", "d31_60": "0.00", "d61_90": "0.00", "d90plus": "0.00" },
  "openItems": [ { "type": "INVOICE", "number": "INV-2026-000003", "dueDate": "2026-11-05T00:00:00.000Z",
    "daysOverdue": 0, "outstanding": "200.00" } ]
}
```

Recipes — sales

`POST /sales/invoices` needs `sales.invoice:create`. `status` decides what happens: `DRAFT` (default) only saves; `ISSUED` numbers the invoice, moves stock, posts the ledger, records the `payments` and queues the e-invoice — all in one transaction. `type` defaults to `TAX` when the customer has a VAT number and `SIMPLIFIED` otherwise. The price of a line defaults to the product (or unit) selling price.

Simplified (B2C) invoice, VAT-inclusive, paid by card

curl

```
curl -X POST "https://erp.sa4s.com/api/v1/sales/invoices" \
-H "Authorization: Bearer $SA4S_API_KEY" -H "Content-Type: application/json" \
-d '{
  "type": "SIMPLIFIED",
  "status": "ISSUED",
  "pricesIncludeVat": true,
  "clientId": "09c3d295-3127-400b-ab9c-819ab51b011c",
  "notes": "Store order #10045",
  "lines": [
    { "productId": "51f771bf-cfb5-4814-8e79-55b990be69c9", "quantity": "6" },
    { "productId": "6fe0f6fa-ed4d-4c4b-81ca-c8095504658f", "quantity": "1", "unitPrice": "40.00" }
  ],
  "payments": [ { "method": "CARD", "amount": "46.90", "reference": "PSP-7781" } ]
}'
```

JavaScript

```
import { randomUUID } from 'node:crypto';

const invoice = await sa4s('POST', '/sales/invoices', {
  type: 'SIMPLIFIED',           // B2C — no customer needed
  status: 'ISSUED',            // number, stock, ledger and the ZATCA queue happen now
  pricesIncludeVat: true,       // unit prices below already contain 15 % VAT
  clientId: randomUUID(),       // store it with your order BEFORE sending, reuse it on every retry
  notes: 'Store order #10045',
  lines: [
    { productId: WATER_ID, quantity: '6' },           // price taken from the product (1.15)
    { productId: DATES_ID, quantity: '1', unitPrice: '40.00' }, // or sent explicitly
  ],
  payments: [{ method: 'CARD', amount: '46.90', reference: 'PSP-7781' } ],
});
console.log(invoice.number, invoice.total, invoice.paymentStatus); // INV-2026-000007 46.9 PAID
```

Python

```
import uuid

invoice = sa4s("POST", "/sales/invoices", {
  "type": "SIMPLIFIED",
  "status": "ISSUED",
  "pricesIncludeVat": True,
  "clientId": str(uuid.uuid4()),
  "notes": "Store order #10045",
  "lines": [
    {"productId": WATER_ID, "quantity": "6"},
    {"productId": DATES_ID, "quantity": "1", "unitPrice": "40.00"},
  ],
  "payments": [{"method": "CARD", "amount": "46.90", "reference": "PSP-7781"}],
})
```



```
    ],
    "payments": [{"method": "CARD", "amount": "46.90", "reference": "PSP-7781"}],
  })
  print(invoice["number"], invoice["total"], invoice["paymentStatus"])
```

PHP

```
<?php
$invoice = sa4s('POST', '/sales/invoices', [
    'type' => 'SIMPLIFIED',
    'status' => 'ISSUED',
    'pricesIncludeVat' => true,
    'clientId' => $order['sa4s_client_uuid'], // generated once and saved with the order
    'notes' => 'Store order #10045',
    'lines' => [
        ['productId' => $waterId, 'quantity' => '6'],
        ['productId' => $datesId, 'quantity' => '1', 'unitPrice' => '40.00'],
    ],
    'payments' => [['method' => 'CARD', 'amount' => '46.90', 'reference' => 'PSP-7781']],
]);
echo $invoice['number'], ' ', $invoice['total'], "\n";
```

C#

```
var invoice = await Sa4s.Send(HttpMethod.Post, "sales/invoices", new
{
    type = "SIMPLIFIED",
    status = "ISSUED",
    pricesIncludeVat = true,
    clientId = order.Sa4sClientId, // Guid saved with the order before the first attempt
    notes = "Store order #10045",
    lines = new object[]
    {
        new { productId = waterId, quantity = "6" },
        new { productId = datesId, quantity = "1", unitPrice = "40.00" },
    },
    payments = new[] { new { method = "CARD", amount = "46.90", reference = "PSP-7781" } },
});
Console.WriteLine($"{invoice.GetProperty("number")} {invoice.GetProperty("total")}");
```

201 Created (abridged)

```
{
  "id": "442ea37d-b9ac-4535-b726-b3ad69832afe",
  "number": "INV-2026-000007",
  "type": "SIMPLIFIED",
  "status": "ISSUED",
  "paymentStatus": "PAID",
  "customerId": null,
  "branchId": "73afa37e-46d6-4c51-9a13-549d92cad970",
  "warehouseId": "1ae3935b-6fae-4c7f-803a-bf405dbc0ea0",
  "clientId": "09c3d295-3127-400b-ab9c-819ab51b011c",
  "uuid": "a14f90a9-de63-4d97-a093-6834846ba8d5",
  "issueDate": "2026-10-06T17:17:09.161Z",
  "pricesIncludeVat": true,
  "currency": "SAR",
  "subtotal": "40.78",
  "discountTotal": "0",
  "taxableTotal": "40.78",
  "vatTotal": "6.12",
  "total": "46.9",
  "paidAmount": "46.9",
  "zatcaStatus": "NOT_REQUIRED",
  "items": [
```



```
{
  "lineNo": 1, "productId": "51f771bf-cfb5-4814-8e79-55b990be69c9", "sku": "SKU-2026-000001", "unitName": "قطعة",
  "quantity": "6", "baseQuantity": "6", "unitPrice": "1.15", "vatCategory": "S", "vatRate": "15", "taxableAmount": "6",
  "vatAmount": "0.9", "lineTotal": "6.9" },
  { "lineNo": 2, "productId": "6fe0f6fa-ed4d-4c4b-81ca-c8095504658f", "sku": "SKU-2026-000003", "unitName": "قطعة",
  "quantity": "1", "baseQuantity": "1", "unitPrice": "40", "vatCategory": "S", "vatRate": "15", "taxableAmount": "34.78",
  "vatAmount": "5.22", "lineTotal": "40" }
],
"payments": [
  { "id": "36f6d10d-95f4-40b7-8fd5-28cae04ccaa6", "number": "RCV-2026-000007", "direction": "IN", "method": "CARD",
  "amount": "46.9", "reference": "PSP-7781", "accountId": "0835961d-0dca-4896-99b4-11fa96c82eda" }
],
"zatca": null,
"amountDue": "0"
}
```

Tax (B2B) invoice, VAT-exclusive, partially paid

Payments may be split across methods (`CASH` , `CARD` , `BANK_TRANSFER` , `CHEQUE` , `WALLET`); whatever is not paid stays on the customer's account (`paymentStatus` `PARTIALLY_PAID` / `UNPAID` , `amountDue`). `CREDIT` marks a sale on account explicitly.

curl

```
curl -X POST "https://erp.sa4s.com/api/v1/sales/invoices" \
-H "Authorization: Bearer $SA4S_API_KEY" -H "Content-Type: application/json" \
-d '{
  "type": "TAX",
  "status": "ISSUED",
  "customerId": "fab57752-483f-4b1a-8d80-8c0401809ffe",
  "pricesIncludeVat": false,
  "dueDate": "2026-11-05",
  "clientUuid": "fe66bdfc-781f-442d-96d0-3fa50ca52a0a",
  "lines": [
    { "productId": "51f771bf-cfb5-4814-8e79-55b990be69c9", "unitId": "b82dbc97-8f7b-43c0-8b73-8b4aed1ebdb4",
    "quantity": "10", "unitPrice": "22.00" },
    { "productId": "7c6db833-d278-4537-87ed-86ef13f3801f", "quantity": "1" }
  ],
  "payments": [ { "method": "BANK_TRANSFER", "amount": "100.00", "reference": "TRX-55120" } ]
}'
```

JavaScript

```
const invoice = await sa4s('POST', '/sales/invoices', {
  type: 'TAX', // B2B: needs a customer (with a VAT number for ZATCA clearance)
  status: 'ISSUED',
  customerId: HORIZON_ID,
  pricesIncludeVat: false, // prices are net; the server adds 15 % VAT
  dueDate: '2026-11-05',
  clientUuid: order.clientUuid,
  lines: [
    { productId: WATER_ID, unitId: BOX_UNIT_ID, quantity: '10', unitPrice: '22.00' }, // 10 boxes = 240 pieces out of stock
    { productId: DELIVERY_SERVICE_ID, quantity: '1' }, // a SERVICE product (no stock)
  ],
  payments: [{ method: 'BANK_TRANSFER', amount: '100.00', reference: 'TRX-55120' }], // the rest stays on the customer account
});
console.log(invoice.total, invoice.paidAmount, invoice.amountDue); // 270.25 100 170.25
```

**Python**

```
invoice = sa4s("POST", "/sales/invoices", {
    "type": "TAX",
    "status": "ISSUED",
    "customerId": HORIZON_ID,
    "pricesIncludeVat": False,
    "dueDate": "2026-11-05",
    "clientId": order["client_uuid"],
    "lines": [
        {"productId": WATER_ID, "unitId": BOX_UNIT_ID, "quantity": "10", "unitPrice": "22.00"},
        {"productId": DELIVERY_SERVICE_ID, "quantity": "1"},
    ],
    "payments": [{"method": "BANK_TRANSFER", "amount": "100.00", "reference": "TRX-55120"}],
})
print(invoice["total"], invoice["amountDue"])
```

PHP

```
<?php
$invoice = sa4s('POST', '/sales/invoices', [
    'type' => 'TAX',
    'status' => 'ISSUED',
    'customerId' => $horizonId,
    'pricesIncludeVat' => false,
    'dueDate' => '2026-11-05',
    'clientId' => $order['sa4s_client_uuid'],
    'lines' => [
        ['productId' => $waterId, 'unitId' => $boxUnitId, 'quantity' => '10', 'unitPrice' => '22.00'],
        ['productId' => $deliveryServiceId, 'quantity' => '1'],
    ],
    'payments' => [['method' => 'BANK_TRANSFER', 'amount' => '100.00', 'reference' => 'TRX-55120']],
]);
echo $invoice['total'], ' / due ', $invoice['amountDue'], "\n";
```

C#

```
var invoice = await Sa4s.Send(HttpMethod.Post, "sales/invoices", new
{
    type = "TAX",
    status = "ISSUED",
    customerId = horizonId,
    pricesIncludeVat = false,
    dueDate = "2026-11-05",
    clientId = order.Sa4sClientId,
    lines = new object[]
    {
        new { productId = waterId, unitId = boxUnitId, quantity = "10", unitPrice = "22.00" },
        new { productId = deliveryServiceId, quantity = "1" },
    },
    payments = new[] { new { method = "BANK_TRANSFER", amount = "100.00", reference = "TRX-55120" } },
});
Console.WriteLine($"{invoice.GetProperty("total")} due {invoice.GetProperty("amountDue")}");
```

201 Created (abridged)

```
{
  "id": "1086f27d-620a-4e60-9dfe-0720fee48398",
  "number": "INV-2026-000002",
  "type": "TAX",
  "status": "ISSUED",
  "paymentStatus": "PARTIALLY_PAID",
  "customerId": "fab57752-483f-4b1a-8d80-8c0401809ffe",
```



```
{
  "dueDate": "2026-11-05T00:00:00.000Z",
  "pricesIncludeVat": false,
  "subtotal": "235",
  "taxableTotal": "235",
  "vatTotal": "35.25",
  "total": "270.25",
  "paidAmount": "100",
  "paymentMethod": "BANK_TRANSFER",
  "zatcaStatus": "NOT_REQUIRED",
  "items": [
    { "lineNo": 1, "sku": "SKU-2026-000001", "unitName": "علبة", "quantity": "10", "baseQuantity": "240", "unitPrice":
"22", "vatRate": "15", "taxableAmount": "220", "vatAmount": "33", "lineTotal": "253" },
    { "lineNo": 2, "sku": "SKU-2026-000002", "unitName": "قطعة", "quantity": "1", "baseQuantity": "1", "unitPrice":
"15", "vatRate": "15", "taxableAmount": "15", "vatAmount": "2.25", "lineTotal": "17.25" }
  ],
  "customer": { "id": "fab57752-483f-4b1a-8d80-8c0401809ffe", "name": "شركة الأفق للتجارة", "vatNumber": "310123456700003",
"balance": "170.25", "creditLimit": "50000" },
  "payments": [ { "number": "RCV-2026-000002", "method": "BANK_TRANSFER", "amount": "100", "reference": "TRX-55120" }
],
  "amountDue": "170.25"
}
```

Draft first, issue when the total matches

When your own system has a total (an online order), create a draft, compare the server's `total` with yours, then issue with the payment. Issuing twice is harmless: the second call answers `409 INVOICE_NOT_ISSUABLE`.

curl

```
# 1) draft – nothing is numbered, no stock moves, nothing reaches ZATCA
curl -X POST "https://erp.sa4s.com/api/v1/sales/invoices" \
-H "Authorization: Bearer $SA4S_API_KEY" -H "Content-Type: application/json" \
-d '{ "pricesIncludeVat": true, "clientId": "6f0c9a3e-2d61-4f7e-9a0b-5b1c2d3e4f50",
      "lines": [ { "productId": "6fe0f6fa-ed4d-4c4b-81ca-c8095504658f", "quantity": "2", "unitPrice": "40.00" } ] }'

# 2) issue it with the payment = the total the server computed
curl -X POST "https://erp.sa4s.com/api/v1/sales/invoices/44d7970d-e564-42fc-a122-f70657e7dc17/issue" \
-H "Authorization: Bearer $SA4S_API_KEY" -H "Content-Type: application/json" \
-d '{ "payments": [ { "method": "CASH", "amount": "80.00" } ] }'
```

JavaScript

```
const draft = await sa4s('POST', '/sales/invoices', { pricesIncludeVat: true, clientId, lines });
if (Math.round(Number(draft.total) * 100) !== Math.round(Number(order.total) * 100)) throw new Error(`total mismatch
${draft.total} ≠ ${order.total}`);
try {
  await sa4s('POST', `/sales/invoices/${draft.id}/issue`, { payments: [{ method: 'CARD', amount: draft.total }] });
} catch (e) {
  if (e.code !== 'INVOICE_NOT_ISSUABLE') throw e; // already issued by an earlier attempt → fine
}
```

Python

```
draft = sa4s("POST", "/sales/invoices", {"pricesIncludeVat": True, "clientId": client_uuid, "lines": lines})
try:
    sa4s("POST", f"/sales/invoices/{draft['id']}/issue", {"payments": [{"method": "CARD", "amount": draft["total"]}]}))
except Sa4sError as e:
    if e.code != "INVOICE_NOT_ISSUABLE":
        raise
```



PHP

```
<?php
$draft = sa4s('POST', '/sales/invoices', ['pricesIncludeVat' => true, 'clientId' => $clientId, 'lines' => $lines]);
try {
    sa4s('POST', "/sales/invoices/{$draft['id']}/issue", ['payments' => [['method' => 'CARD', 'amount' =>
    $draft['total']]]]);
} catch (Sa4sError $e) {
    if (($e->body['code'] ?? '') !== 'INVOICE_NOT_ISSUABLE') throw $e;
}
```

Retry an invoice safely

Send the same body with the same `clientId` as often as needed — see [Idempotency](#) for the full example. The replay returns `201` with the original invoice and `"idempotentReplay": true`.

Download the invoice PDF

`GET /sales/invoices/{id}/pdf` — scope `sales.invoice:print`. `format=a4` (default) or `receipt` (80 mm thermal), `lang=ar`, `en` or `both`, `download=true` for an attachment instead of inline. The PDF carries the ZATCA QR code once the invoice has one.

curl

```
curl "https://erp.sa4s.com/api/v1/sales/invoices/442ea37d-b9ac-4535-b726-b3ad69832afe/pdf?
format=a4&lang=ar&download=true" \
-H "Authorization: Bearer $SA4S_API_KEY" \
-o INV-2026-000007.pdf
```

JavaScript

```
import { writeFile } from 'node:fs/promises';

const pdf = await sa4s('GET', `/sales/invoices/${invoiceId}/pdf?format=a4&lang=both`); // Buffer (application/pdf)
await writeFile(`${invoiceNumber}.pdf`, pdf);
```

Python

```
pdf = sa4s("GET", f"/sales/invoices/{invoice_id}/pdf", params={"format": "receipt", "lang": "ar"}) # bytes
with open(f"{invoice_number}.pdf", "wb") as f:
    f.write(pdf)
```

PHP

```
<?php
$pdf = sa4s('GET', "/sales/invoices/$invoiceId/pdf?format=a4&lang=en");
file_put_contents("$invoiceNumber.pdf", $pdf);
```

C#

```
var bytes = await Sa4s.Http.GetByteArrayAsync($"sales/invoices/{invoiceId}/pdf?format=a4&lang=ar");
await File.WriteAllBytesAsync($"{invoiceNumber}.pdf", bytes);
```

**HTTP**

```

HTTP/1.1 200 OK
Content-Type: application/pdf
Content-Length: 17138
Content-Disposition: attachment; filename="INV-2026-000007.pdf"
Cache-Control: private, no-store

```

ZATCA status and QR code

Issued invoices are sent to ZATCA by SA4S automatically (reporting for simplified invoices, clearance for tax invoices) once the company has completed e-invoicing onboarding in the ERP. Keys can read the outcome but not manage onboarding:

zatcaStatus	Meaning
NOT_REQUIRED	Draft, or e-invoicing is not active for the company yet
PENDING / SUBMITTED	Queued / sent, waiting for ZATCA
REPORTED / CLEARED	Accepted (simplified / tax invoice)
REJECTED	Refused — <code>zatca.errors</code> explains why; the ERP shows it to the accountant

GET `/sales/invoices/{id}` returns `zatcaStatus` and a `zatca` block (`status`, `flow`, `icv`, `qrCode` as base64 TLV, `warnings`, `errors`, `submittedAt`) — `null` until the invoice is processed. Filter lists with `zatcaStatus=...`, or subscribe to the `zatca.status` [webhook](#).

curl

```

# one invoice: zatcaStatus + the zatca block (QR, flow, warnings)
curl "https://erp.sa4s.com/api/v1/sales/invoices/442ea37d-b9ac-4535-b726-b3ad69832afe" -H "Authorization: Bearer $SA4S_API_KEY"

# everything ZATCA rejected this month
curl "https://erp.sa4s.com/api/v1/sales/invoices?zatcaStatus=REJECTED&from=2026-10-01&to=2026-10-31" -H "Authorization: Bearer $SA4S_API_KEY"

```

JavaScript

```

const inv = await sa4s('GET', `/sales/invoices/${invoiceId}`);
console.log(inv.zatcaStatus); // NOT_REQUIRED | PENDING | SUBMITTED | CLEARED | REPORTED | REJECTED
if (inv.zatca?.qrCode) renderQr(inv.zatca.qrCode); // base64 TLV – print it on your own receipt
if (inv.zatcaStatus === 'REJECTED') console.warn(inv.zatca.errors);

```

Python

```

inv = sa4s("GET", f"/sales/invoices/{invoice_id}")
zatca = inv.get("zatca") or {}
print(inv["zatcaStatus"], zatca.get("flow"), zatca.get("qrCode"))

```

PHP

```

<?php
$inv = sa4s('GET', "/sales/invoices/$invoiceId");
echo $inv['zatcaStatus'], ' ', $inv['zatca']['qrCode'] ?? '(no QR yet)', "\n";

```

**200 OK — shape of the zatca block after processing (illustrative values)**

```
{
  "id": "442ea37d-b9ac-4535-b726-b3ad69832afe",
  "number": "INV-2026-000007",
  "type": "SIMPLIFIED",
  "uuid": "a14f90a9-de63-4d97-a093-6834846ba8d5",
  "zatcaStatus": "REPORTED",
  "zatca": {
    "status": "REPORTED",
    "flow": "reporting",
    "icv": 7,
    "qrCode": "<base64 TLV of the ZATCA QR>",
    "warnings": [],
    "errors": [],
    "submittedAt": "2026-10-06T17:17:12.402Z"
  }
}
```

Refund or return: credit note

`POST /sales/invoices/{id}/credit-note` — scope `sales.invoice:refund`. Credit some lines (`originalItemId` + `quantity`) or everything (`full: true`), optionally put the goods back in stock (`restock`) and pay the money back (`refund`). The note gets its own number (`CN-...`) and goes to ZATCA like any invoice.

curl

```
curl -X POST "https://erp.sa4s.com/api/v1/sales/invoices/1086f27d-620a-4e60-9dfe-0720fee48398/credit-note" \
-H "Authorization: Bearer $SA4S_API_KEY" -H "Content-Type: application/json" \
-d '{
  "reason": "Damaged boxes returned",
  "lines": [ { "originalItemId": "2cfc59f1-ae9f-4868-8920-0b264ac071b2", "quantity": "2" } ],
  "restock": true,
  "refund": { "method": "BANK_TRANSFER", "reference": "RF-1001" },
  "clientId": "d99f7cd1-d612-40bf-8a74-aeebad8c9e9c"
}'
```

JavaScript

```
const original = await sa4s('GET', `/sales/invoices/${invoiceId}`);
const line = original.items.find((i) => i.productId === WATER_ID);
const note = await sa4s('POST', `/sales/invoices/${invoiceId}/credit-note`, {
  reason: 'Damaged boxes returned',
  lines: [{ originalItemId: line.id, quantity: '2' }], // in the unit of the original line (boxes)
  restock: true, // goods go back to stock at the original cost
  refund: { method: 'BANK_TRANSFER', reference: 'RF-1001' }, // amount defaults to the note total (capped by what was paid)
  clientId: refund.clientId,
});
console.log(note.number, note.total); // CN-2026-000001 50.6
```

Python

```
original = sa4s("GET", f"/sales/invoices/{invoice_id}")
line = next(i for i in original["items"] if i["productId"] == WATER_ID)
note = sa4s("POST", f"/sales/invoices/{invoice_id}/credit-note", {
    "reason": "Damaged boxes returned",
    "lines": [{"originalItemId": line["id"], "quantity": "2"}],
})
```



```
"restock": True,
"refund": {"method": "BANK_TRANSFER", "reference": "RF-1001"},
"clientId": refund_uuid,
})
```

PHP

```
<?php
// full refund of everything still creditable on the invoice
$note = sa4s('POST', "/sales/invoices/$invoiceId/credit-note", [
    'reason' => 'Order returned',
    'full' => true,
    'restock' => true,
    'refund' => ['method' => 'CARD', 'reference' => $pspRefundId],
    'clientId' => $refundUuid,
]);
```

201 Created (abridged)

```
{
  "id": "d6111f48-8d38-4375-8786-805627995cb5",
  "number": "CN-2026-000001",
  "type": "CREDIT_NOTE",
  "status": "ISSUED",
  "paymentStatus": "PAID",
  "originalInvoiceId": "1086f27d-620a-4e60-9dfe-0720fee48398",
  "noteReason": "Damaged boxes returned",
  "taxableTotal": "44",
  "vatTotal": "6.6",
  "total": "50.6",
  "paidAmount": "50.6",
  "items": [ { "originalItemId": "2cfc59f1-ae9f-4868-8920-0b264ac071b2", "unitName": "علبة", "quantity": "2",
"baseQuantity": "48", "unitPrice": "22", "lineTotal": "50.6" } ]
}
```

Cancel an invoice

`POST /sales/invoices/{id}/cancel` with a `reason` — scope `sales.invoice:cancel`. Only for invoices without payments or notes that have not reached ZATCA; otherwise issue a credit note (`INVOICE_HAS_PAYMENTS`, `ZATCA_REPORTED_USE_CREDIT_NOTE`).

curl

```
curl -X POST "https://erp.sa4s.com/api/v1/sales/invoices/015727de-9146-41eb-839b-fd477dc9e9a8/cancel" \
-H "Authorization: Bearer $SA4S_API_KEY" -H "Content-Type: application/json" \
-d '{ "reason": "Customer cancelled the order" }'
```

JavaScript

```
await sa4s('POST', `/sales/invoices/${invoiceId}/cancel`, { reason: 'Customer cancelled the order' });
```

Python

```
sa4s("POST", f"/sales/invoices/{invoice_id}/cancel", {"reason": "Customer cancelled the order"})
```

PHP

```
<?php
sa4s('POST', "/sales/invoices/$invoiceId/cancel", ['reason' => 'Customer cancelled the order']);
```



Recipes — payments

Record a customer receipt

`POST /payments/receipts` — scope `payments:create`. Allocate the money to one or more open invoices (`allocations`), let SA4S allocate it to the oldest open invoices (`autoAllocate: true`), or leave it unallocated as an advance. `accountId` picks the cash drawer or bank account (default: the first active cash account for `CASH`, the first bank account otherwise); list the accounts with `GET /financial-accounts` (scope `cash:view`).

curl

```
curl -X POST "https://erp.sa4s.com/api/v1/payments/receipts" \
-H "Authorization: Bearer $SA4S_API_KEY" -H "Content-Type: application/json" \
-d '{
  "customerId": "fab57752-483f-4b1a-8d80-8c0401809ffe",
  "method": "BANK_TRANSFER",
  "amount": "170.25",
  "reference": "TRX-55190",
  "clientId": "c8b5212a-1b50-4864-9d45-7b562575f3fa",
  "allocations": [ { "invoiceId": "1086f27d-620a-4e60-9dfe-0720fee48398", "amount": "170.25" } ]
}'
```

JavaScript

```
const receipt = await sa4s('POST', '/payments/receipts', {
  customerId: HORIZON_ID,
  method: 'BANK_TRANSFER', // CASH | CARD | BANK_TRANSFER | CHEQUE | WALLET
  amount: '170.25',
  reference: 'TRX-55190', // your bank / PSP reference
  clientId: transfer.clientId, // idempotency key: one receipt per bank transfer
  allocations: [{ invoiceId: INVOICE_ID, amount: '170.25' }], // or autoAllocate: true (oldest open invoices first)
});
console.log(receipt.number, receipt.allocations.map((a) => `${a.invoiceId}=${a.amount}`));
```

Python

```
receipt = sa4s("POST", "/payments/receipts", {
    "customerId": HORIZON_ID,
    "method": "BANK_TRANSFER",
    "amount": "170.25",
    "reference": "TRX-55190",
    "clientId": transfer["client_uuid"],
    "allocations": [{"invoiceId": INVOICE_ID, "amount": "170.25"}],
})
print(receipt["number"])
```

PHP

```
<?php
$receipt = sa4s('POST', '/payments/receipts', [
    'customerId' => $horizonId,
    'method' => 'BANK_TRANSFER',
    'amount' => '170.25',
    'reference' => 'TRX-55190',
```



```
'clientId' => $transfer['client_uuid'],
'allocations' => [['invoiceId' => $invoiceId, 'amount' => '170.25']],
]);
echo $receipt['number'], "\n";
```

C#

```
var receipt = await Sa4s.Send(HttpMethod.Post, "payments/receipts", new
{
    customerId = horizonId,
    method = "BANK_TRANSFER",
    amount = "170.25",
    reference = "TRX-55190",
    clientId = transfer.ClientUuid,
    allocations = new[] { new { invoiceId, amount = "170.25" } },
});
Console.WriteLine(receipt.GetProperty("number"));
```

201 Created

```
{
  "id": "f7e14210-3cbc-43ba-8b39-ae5c2b207898",
  "number": "RCV-2026-000004",
  "direction": "IN",
  "partyType": "CUSTOMER",
  "partyId": "fab57752-483f-4b1a-8d80-8c0401809ffe",
  "method": "BANK_TRANSFER",
  "amount": "170.25",
  "date": "2026-10-06T17:01:02.845Z",
  "accountId": "0835961d-0dca-4896-99b4-11fa96c82eda",
  "reference": "TRX-55190",
  "notes": null,
  "branchId": null,
  "clientId": "c8b5212a-1b50-4864-9d45-7b562575f3fa",
  "journalEntryId": "b42f4c6e-154d-4a8a-b04e-62dca3465d57",
  "cancelledAt": null,
  "createdAt": "2026-10-06T17:01:02.846Z",
  "allocations": [
    { "id": "9152b27c-05d7-4c9a-9989-1e0fe31b6e3a", "invoiceType": "SALES_INVOICE", "invoiceId": "1086f27d-620a-4e60-9dfe-0720fee48398", "amount": "170.25" }
  ]
}
```

400 Bad Request

```
{
  "statusCode": 400,
  "code": "ALLOCATION_EXCEEDS_PAYMENT",
  "message": "Allocations exceed the payment amount",
  "requestId": "54c3a873-10f7-4dc1-97e4-52de241723c0"
}
```

List unpaid invoices

GET /sales/invoices?status=ISSUED&paymentStatus=UNPAID (or PARTIALLY_PAID) — scope sales.invoice:view . List rows carry balanceDue ; the single-invoice endpoint calls it amountDue .

**curl**

```
curl "https://erp.sa4s.com/api/v1/sales/invoices?customerId=fab57752-483f-4b1a-8d80-8c0401809ffe&status=ISSUED&paymentStatus=UNPAID&sort=issueDate&order=asc" \
-H "Authorization: Bearer $SA4S_API_KEY"
# partially paid ones: paymentStatus=PARTIALLY_PAID (run both to get every open invoice)
```

JavaScript

```
const open = [];
for (const paymentStatus of ['UNPAID', 'PARTIALLY_PAID']) {
  const q = new URLSearchParams({ customerId, status: 'ISSUED', paymentStatus, pageSize: '200', sort: 'issueDate',
    order: 'asc' });
  open.push(...(await sa4s('GET', `/sales/invoices?${q}`)).items);
}
const due = open.reduce((sum, inv) => sum + Math.round(Number(inv.balanceDue) * 100), 0) / 100; // compare in halalas
```

Python

```
from decimal import Decimal

open_invoices = []
for status in ("UNPAID", "PARTIALLY_PAID"):
    open_invoices += sa4s("GET", "/sales/invoices", params={
        "customerId": customer_id, "status": "ISSUED", "paymentStatus": status, "pageSize": 200, "sort": "issueDate",
        "order": "asc",
    })["items"]
total_due = sum(Decimal(i["balanceDue"]) for i in open_invoices)
```

PHP

```
<?php
$open = [];
foreach (['UNPAID', 'PARTIALLY_PAID'] as $ps) {
    $q = http_build_query(['customerId' => $customerId, 'status' => 'ISSUED', 'paymentStatus' => $ps, 'pageSize' =>
    200, 'sort' => 'issueDate', 'order' => 'asc']);
    $open = array_merge($open, sa4s('GET', "/sales/invoices?$q")['items']);
}
$due = array_reduce($open, fn ($s, $i) => bcadd($s, $i['balanceDue'], 2), '0');
```

C#

```
var open = new List<JsonElement>();
foreach (var ps in new[] { "UNPAID", "PARTIALLY_PAID" })
{
    var page = await Sa4s.Send(HttpMethod.Get, $"sales/invoices?customerId={customerId}&status=ISSUED&paymentStatus={ps}&pageSize=200&sort=issueDate&order=asc");
    open.AddRange(page.GetProperty("items").EnumerateArray());
}
decimal due = open.Sum(i => decimal.Parse(i.GetProperty("balanceDue").GetString()),
System.Globalization.CultureInfo.InvariantCulture));
```

200 OK (abridged)

```
{
  "items": [
    {
      "id": "46f95e34-0748-4b23-a08b-8765c6ad5d58",
      "number": "INV-2026-000003",
      "type": "TAX",
      "status": "ISSUED",
      "paymentStatus": "UNPAID",
```



```

    "issueDate": "2026-10-06T17:01:02.393Z",
    "dueDate": "2026-11-05T00:00:00.000Z",
    "total": "230",
    "paidAmount": "0",
    "paymentMethod": "CREDIT",
    "customer": { "id": "fab57752-483f-4b1a-8d80-8c0401809ffe", "name": "شركة الأفق للتجارة", "nameEn": "Horizon Trading Co", "vatNumber": "310123456700003" },
    "balanceDue": "230"
  }
],
"total": 1,
"page": 1,
"pageSize": 25
}

```

Pay a supplier

POST /payments/disbursements — scope **payments:create**; allocations point to posted purchase invoices. The paying account must hold enough money unless the company allows negative cash.

curl

```

curl -X POST "https://erp.sa4s.com/api/v1/payments/disbursements" \
-H "Authorization: Bearer $SA4S_API_KEY" -H "Content-Type: application/json" \
-d '{
  "supplierId": "3788b574-54f8-4ed9-9a30-f7d38199ff05",
  "method": "BANK_TRANSFER",
  "amount": "100.00",
  "clientUuid": "ed9df166-3fee-44b6-8a0d-d006e00e427e",
  "allocations": [ { "invoiceId": "274f3053-5728-4d84-931f-a42ef0ebdc55", "amount": "100.00" } ]
}'

```

JavaScript

```

const payment = await sa4s('POST', '/payments/disbursements', {
  supplierId: SUPPLIER_ID, method: 'BANK_TRANSFER', amount: '100.00', clientUuid: payout.clientUuid,
  allocations: [{ invoiceId: PURCHASE_INVOICE_ID, amount: '100.00' }],
});
console.log(payment.number); // PAY-2026-000001

```

Python

```

payment = sa4s("POST", "/payments/disbursements", {
    "supplierId": SUPPLIER_ID, "method": "BANK_TRANSFER", "amount": "100.00", "clientUuid": payout["client_uuid"],
    "allocations": [{"invoiceId": PURCHASE_INVOICE_ID, "amount": "100.00"}],
})

```

PHP

```

<?php
$payment = sa4s('POST', '/payments/disbursements', [
    'supplierId' => $supplierId, 'method' => 'BANK_TRANSFER', 'amount' => '100.00', 'clientUuid' =>
    $payout['client_uuid'],
    'allocations' => [['invoiceId' => $purchaseInvoiceId, 'amount' => '100.00']],
]);

```

**409 Conflict**

```
{
  "statusCode": 409,
  "code": "INSUFFICIENT_FUNDS",
  "message": "The balance of \"Main Bank Account\" is lower than the amount",
  "details": { "accountId": "f8aa56c8-9796-455f-b072-967a40ef7954", "accountName": "Main Bank Account", "balance":
"0.00", "amount": "100.00" },
  "requestId": "f40c8732-1f1a-4be3-9c16-7714b34c865e"
}
```

curl

```
# receipts (IN) of October, newest first
curl "https://erp.sa4s.com/api/v1/payments?direction=IN&from=2026-10-01&to=2026-10-31&sort=date&order=desc" \
-H "Authorization: Bearer $SA4S_API_KEY"
```

JavaScript

```
const { items } = await sa4s('GET', '/payments?direction=IN&from=2026-10-01&to=2026-10-31&sort=date&order=desc');
```

Python

```
items = sa4s("GET", "/payments", params={"direction": "IN", "from": "2026-10-01", "to": "2026-10-31", "sort": "date",
"order": "desc"})["items"]
```

PHP

```
<?php
$items = sa4s('GET', '/payments?direction=IN&from=2026-10-01&to=2026-10-31&sort=date&order=desc')['items'];
```

Recipes — purchasing

Create a purchase invoice

`POST /purchases/invoices` — scope `purchases.invoice:create`. A standalone supplier invoice receives the goods into `warehouseId` when it is posted; it can also be built from a goods receipt (`receiptId`) or an order (`orderId`). `unitPrice` is required on purchase lines. Posting (`POST /purchases/invoices/{id}/post` , or `post: true` on create) needs `purchases.invoice:approve`: it moves stock at cost, books input VAT and the supplier payable. A `supplierInvoiceNo` already used for the same supplier is refused (`DUPLICATE_SUPPLIER_INVOICE`), which makes retries safe.

curl

```
curl -X POST "https://erp.sa4s.com/api/v1/purchases/invoices" \
-H "Authorization: Bearer $SA4S_API_KEY" -H "Content-Type: application/json" \
-d '{
  "supplierId": "3788b574-54f8-4ed9-9a30-f7d38199ff05",
  "warehouseId": "1ae3935b-6fae-4c7f-803a-bf405dbc0ea0",
  "supplierInvoiceNo": "NWF-88231",
  "pricesIncludeVat": false,
  "items": [
    { "productId": "51f771bf-cfb5-4814-8e79-55b990be69c9", "unitId": "b82dbc97-8f7b-43c0-8b73-8b4aed1ebdb4",
    "quantity": "20", "unitPrice": "14.40" },
    { "productId": "6fe0f6fa-ed4d-4c4b-81ca-c8095504658f", "quantity": "50", "unitPrice": "28.00" }
  ]
}'

# post it (stock in at cost, input VAT, supplier payable) — needs purchases.invoice:approve
curl -X POST "https://erp.sa4s.com/api/v1/purchases/invoices/274f3053-5728-4d84-931f-a42ef0ebdc55/post" \
-H "Authorization: Bearer $SA4S_API_KEY"
```

JavaScript

```
const bill = await sa4s('POST', '/purchases/invoices', {
  supplierId: SUPPLIER_ID,
  warehouseId: WAREHOUSE_ID,           // where the goods are received
  supplierInvoiceNo: 'NWF-88231',      // the supplier's own number (keep it unique on your side)
  pricesIncludeVat: false,
  items: [
    { productId: WATER_ID, unitId: BOX_UNIT_ID, quantity: '20', unitPrice: '14.40' }, // unitPrice is required on
    purchases
    { productId: DATES_ID, quantity: '50', unitPrice: '28.00' },
  ],
  // post: true → create and post in one call
});
const posted = await sa4s('POST', `/purchases/invoices/${bill.id}/post`);
console.log(posted.number, posted.status, posted.total, posted.balanceDue); // PI-2026-000001 POSTED 1941.2 1941.2
```

Python

```
bill = sa4s("POST", "/purchases/invoices", {
    "supplierId": SUPPLIER_ID,
    "warehouseId": WAREHOUSE_ID,
    "supplierInvoiceNo": "NWF-88231",
    "pricesIncludeVat": False,
    "items": [
        {"productId": WATER_ID, "unitId": BOX_UNIT_ID, "quantity": "20", "unitPrice": "14.40"},
        {"productId": DATES_ID, "quantity": "50", "unitPrice": "28.00"},
    ]
})
```



```
    ],
    "post": True,
  })
  print(bill["number"], bill["status"])
```

PHP

```
<?php
$bill = sa4s('POST', '/purchases/invoices', [
    'supplierId' => $supplierId,
    'warehouseId' => $warehouseId,
    'supplierInvoiceNo' => 'NWF-88231',
    'pricesIncludeVat' => false,
    'items' => [
        ['productId' => $waterId, 'unitId' => $boxUnitId, 'quantity' => '20', 'unitPrice' => '14.40'],
        ['productId' => $datesId, 'quantity' => '50', 'unitPrice' => '28.00'],
    ],
    'post' => true,
]);
echo $bill['number'], ' ', $bill['status'], "\n";
```

201 Created (abridged)

```
{
  "id": "274f3053-5728-4d84-931f-a42ef0ebdc55",
  "number": "PI-2026-000001",
  "supplierInvoiceNo": "NWF-88231",
  "supplierId": "3788b574-54f8-4ed9-9a30-f7d38199ff05",
  "warehouseId": "1ae3935b-6fae-4c7f-803a-bf405dbc0ea0",
  "status": "DRAFT",
  "date": "2026-10-06T00:00:00.000Z",
  "dueDate": "2026-11-05T00:00:00.000Z",
  "pricesIncludeVat": false,
  "subtotal": "1688",
  "taxableTotal": "1688",
  "vatTotal": "253.2",
  "total": "1941.2",
  "paidAmount": "0",
  "receivesStock": true,
  "items": [
    { "productId": "6fe0f6fa-ed4d-4c4b-81ca-c8095504658f", "unitId": "0757ef16-15fa-4b58-996d-ebb3d8bb5fcb",
      "quantity": "50", "baseQuantity": "50", "unitPrice": "28", "vatCategory": "S", "vatRate": "15", "taxableAmount":
      "1400", "vatAmount": "210", "lineTotal": "1610" },
    { "productId": "51f771bf-cfb5-4814-8e79-55b990be69c9", "unitId": "b82dbc97-8f7b-43c0-8b73-8b4aed1ebdb4",
      "quantity": "20", "baseQuantity": "480", "unitPrice": "14.4", "vatCategory": "S", "vatRate": "15", "taxableAmount":
      "288", "vatAmount": "43.2", "lineTotal": "331.2" }
  ],
  "balanceDue": "1941.2"
}
```

Create a purchase order

POST /purchases/orders — scope `purchases.order:create` ; the order starts as `DRAFT` .

curl

```
curl -X POST "https://erp.sa4s.com/api/v1/purchases/orders" \
-H "Authorization: Bearer $SA4S_API_KEY" -H "Content-Type: application/json" \
-d '{
  "supplierId": "3788b574-54f8-4ed9-9a30-f7d38199ff05",
  "warehouseId": "1ae3935b-6fae-4c7f-803a-bf405dbc0ea0",
  "expectedDate": "2026-10-20",
```



```
"items": [ { "productId": "51f771bf-cfb5-4814-8e79-55b990be69c9", "unitId": "b82dbc97-8f7b-43c0-8b73-8b4aed1ebdb4",
"quantity": "30", "unitPrice": "14.40" } ]
}'
# → 201 { "number": "PO-2026-000001", "status": "DRAFT", "total": "496.8", ... }
# then POST /purchases/orders/{id}/submit (DRAFT → PENDING) and POST /purchases/orders/{id}/approve (→ APPROVED)
```

JavaScript

```
const po = await sa4s('POST', '/purchases/orders', {
  supplierId: SUPPLIER_ID, warehouseId: WAREHOUSE_ID, expectedDate: '2026-10-20',
  items: [{ productId: WATER_ID, unitId: BOX_UNIT_ID, quantity: '30', unitPrice: '14.40' }],
});
await sa4s('POST', `/purchases/orders/${po.id}/submit`); // needs purchases.order:edit
await sa4s('POST', `/purchases/orders/${po.id}/approve`); // needs purchases.order:approve
```

Python

```
po = sa4s("POST", "/purchases/orders", {
    "supplierId": SUPPLIER_ID, "warehouseId": WAREHOUSE_ID, "expectedDate": "2026-10-20",
    "items": [{"productId": WATER_ID, "unitId": BOX_UNIT_ID, "quantity": "30", "unitPrice": "14.40"}],
})
```

PHP

```
<?php
$po = sa4s('POST', '/purchases/orders', [
    'supplierId' => $supplierId, 'warehouseId' => $warehouseId, 'expectedDate' => '2026-10-20',
    'items' => [['productId' => $waterId, 'unitId' => $boxUnitId, 'quantity' => '30', 'unitPrice' => '14.40']],
]);
```



Recipes — inventory

Stock balances by warehouse

GET `/inventory/balances` — scope `inventory:view`. One row per product × warehouse with `onHand`, `reserved` and `available = onHand - reserved`, all in base units; `avgCost` and `value` appear only with `inventory.cost:view`.

curl

```
curl "https://erp.sa4s.com/api/v1/inventory/balances?warehouseId=1ae3935b-6fae-4c7f-803a-bf405dbc0ea0&page=1&pageSize=200&sort=sku&order=asc" \
-H "Authorization: Bearer $SA4S_API_KEY"
# other filters: productId, categoryId, search (SKU / name), status=all/low/over/out/dead/in_stock
```

JavaScript

```
// full snapshot of one warehouse: { sku → available }
async function stockSnapshot(warehouseId) {
  const out = new Map();
  for (let page = 1; ; page++) {
    const res = await sa4s('GET', `/inventory/balances?warehouseId=${warehouseId}&page=${page}&pageSize=200&sort=sku&order=asc`);
    for (const row of res.items) out.set(row.product.sku, row.available); // decimal strings, base units
    if (page * res.pageSize >= res.total) return out;
  }
}
```

Python

```
def stock_snapshot(warehouse_id):
    out, page = {}, 1
    while True:
        res = sa4s("GET", "/inventory/balances", params={"warehouseId": warehouse_id, "page": page, "pageSize": 200, "sort": "sku", "order": "asc"})
        out.update({row["product"]["sku"]: row["available"] for row in res["items"]})
        if page * res["pageSize"] >= res["total"]:
            return out
        page += 1
```

PHP

```
<?php
function stock_snapshot(string $warehouseId): array
{
    $out = [];
    for ($page = 1; ; $page++) {
        $res = sa4s('GET', '/inventory/balances?' . http_build_query(['warehouseId' => $warehouseId, 'page' => $page, 'pageSize' => 200, 'sort' => 'sku', 'order' => 'asc']));
        foreach ($res['items'] as $row) $out[$row['product']['sku']] = $row['available'];
        if ($page * $res['pageSize'] >= $res['total']) return $out;
    }
}
```

**C#**

```
var stock = new Dictionary<string, string>();
for (int page = 1; ; page++)
{
    var res = await Sa4s.Send(HttpMethod.Get, $"inventory/balances?warehouseId={warehouseId}&page={page}&pageSize=200&sort=sku&order=asc");
    foreach (var row in res.GetProperty("items").EnumerateArray())
        stock[row.GetProperty("product").GetProperty("sku").GetString()!] = row.GetProperty("available").GetString();
    if (page * res.GetProperty("pageSize").GetInt32() >= res.GetProperty("total").GetInt32()) break;
}
```

200 OK (abridged)

```
{
  "items": [
    {
      "warehouseId": "1ae3935b-6fae-4c7f-803a-bf405dbc0ea0",
      "warehouse": { "id": "1ae3935b-6fae-4c7f-803a-bf405dbc0ea0", "code": "WH-MAIN", "nameAr": "المستودع الرئيسي",
"nameEn": "Main Warehouse" },
      "productId": "51f771bf-cfb5-4814-8e79-55b990be69c9",
      "product": { "id": "51f771bf-cfb5-4814-8e79-55b990be69c9", "sku": "SKU-2026-000001", "nameAr": "مياه معدنية 330 مل",
"nameEn": "Mineral water 330 ml", "type": "PHYSICAL", "trackBatch": false, "baseUnit": "PCS" },
      "onHand": "282",
      "reserved": "0",
      "available": "282",
      "minStock": "48",
      "maxStock": null,
      "reorderLevel": null,
      "flags": { "low": false, "over": false, "out": false },
      "lastMovementAt": "2026-10-06T17:01:02.314Z",
      "updatedAt": "2026-10-06T17:01:02.284Z",
      "avgCost": "0.6",
      "value": "169.20"
    }
  ],
  "total": 2,
  "page": 1,
  "pageSize": 200,
  "summary": { "rows": 2, "totalOnHand": "325", "totalValue": "1373.20" }
}
```

curl

```
# one product across every warehouse (+ batches when the product tracks them)
curl "https://erp.sa4s.com/api/v1/inventory/products/51f771bf-cfb5-4814-8e79-55b990be69c9/stock" -H "Authorization: Bearer $SA4S_API_KEY"
# → { "product": { "sku": "SKU-2026-000001", ... }, "warehouses": [ { "warehouseId": "...", "onHand": "282", "reserved": "0", "available": "282" } ],
#    "totals": { "onHand": "282", "reserved": "0", "available": "282" } }
```

JavaScript

```
const s = await sa4s('GET', `/inventory/products/${productId}/stock`);
console.log(s.product.sku, s.totals.available, s.warehouses.map((w) => `${w.warehouse.code}=${w.available}`));
```

Python

```
s = sa4s("GET", f"/inventory/products/{product_id}/stock")
```



PHP

```
<?php
$s = sa4s('GET', "/inventory/products/$productId/stock");
```

Stock movements since a date

GET /inventory/movements — scope `inventory:view`. The immutable stock ledger: every sale, purchase, return, adjustment, transfer and count as a signed `qtyChange` in base units with the resulting `balanceAfter` and the source document (`refType` , `refId` , `note`). Types: `OPENING` , `PURCHASE_RECEIPT` , `PURCHASE_RETURN` , `SALE` , `SALE_RETURN` , `ADJUSTMENT_IN` , `ADJUSTMENT_OUT` , `TRANSFER_OUT` , `TRANSFER_IN` , `DAMAGE` , `EXPIRED` , `COUNT_ADJUSTMENT` , `COMPOSITE_CONSUME` .

curl

```
curl "https://erp.sa4s.com/api/v1/inventory/movements?from=2026-10-06&warehouseId=1ae3935b-6fae-4c7f-803a-bf405dbc0ea0&order=asc&page=1&pageSize=200" \
-H "Authorization: Bearer $SA4S_API_KEY"
# other filters: to, productId, type (SALE, PURCHASE_RECEIPT, ...), refType, refId, batchId
```

JavaScript

```
// every movement since a date, oldest first; skip the ones you already processed
async function movementsSince(day, seen) {
  const changed = new Set();
  for (let page = 1; ; page++) {
    const res = await sa4s('GET', `/inventory/movements?from=${day}&order=asc&page=${page}&pageSize=200`);
    for (const m of res.items) if (!seen.has(m.id)) { seen.add(m.id); changed.add(`${m.productId}:${m.warehouseId}`); }
    if (page * res.pageSize >= res.total) return changed;
  }
}
```

Python

```
def movements_since(day, seen):
    changed, page = set(), 1
    while True:
        res = sa4s("GET", "/inventory/movements", params={"from": day, "order": "asc", "page": page, "pageSize": 200})
        for m in res["items"]:
            if m["id"] not in seen:
                seen.add(m["id"])
                changed.add((m["productId"], m["warehouseId"]))
        if page * res["pageSize"] >= res["total"]:
            return changed
        page += 1
```

PHP

```
<?php
function movements_since(string $day, array &$amp;seen): array
{
    $changed = [];
    for ($page = 1; ; $page++) {
        $res = sa4s('GET', '/inventory/movements?' . http_build_query(['from' => $day, 'order' => 'asc', 'page' => $page, 'pageSize' => 200]));
        foreach ($res['items'] as $m) {
            if (isset($seen[$m['id']])) continue;
            $seen[$m['id']] = true;
            $changed["{$m['productId']}:{m['warehouseId']}"] = true;
        }
    }
}
```



```
    }  
    if ($page * $res['pageSize'] >= $res['total']) return array_keys($changed);  
  }  
}
```

200 OK (abridged)

```
{  
  "items": [  
    {  
      "id": "3002b923-48af-4d7a-b5b3-325a4fbaec75",  
      "createdAt": "2026-10-06T17:01:02.058Z",  
      "type": "PURCHASE_RECEIPT",  
      "warehouseId": "1ae3935b-6fae-4c7f-803a-bf405dbc0ea0",  
      "productId": "51f771bf-cfb5-4814-8e79-55b990be69c9",  
      "product": { "sku": "SKU-2026-000001", "nameAr": "مياه معدنية 330 مل", "nameEn": "Mineral water 330 ml" },  
      "batchId": null,  
      "qtyChange": "480",  
      "balanceAfter": "480",  
      "refType": "PURCHASE_INVOICE",  
      "refId": "274f3053-5728-4d84-931f-a42ef0ebdc55",  
      "note": "PI-2026-000001",  
      "unitCost": "0.6",  
      "value": "288.00"  
    },  
    {  
      "id": "16d83e5d-d936-4dff-a48a-b2e059637c92",  
      "createdAt": "2026-10-06T17:01:03.681Z",  
      "type": "SALE",  
      "warehouseId": "1ae3935b-6fae-4c7f-803a-bf405dbc0ea0",  
      "productId": "6fe0f6fa-ed4d-4c4b-81ca-c8095504658f",  
      "qtyChange": "-1",  
      "balanceAfter": "41",  
      "refType": "SALES_INVOICE",  
      "refId": "015727de-9146-41eb-839b-fd477dc9e9a8",  
      "note": "INV-2026-000006",  
      "unitCost": "28",  
      "value": "-28.00"  
    }  
  ],  
  "total": 8,  
  "page": 1,  
  "pageSize": 200  
}
```



Recipes — reports

Report catalogue

`GET /reports` — scope `reports:view`. Lists the reports the key may run with their filters, columns and formats. Each report also needs the scope of its data (e.g. `sales.invoice:view` for sales reports, `accounting:view` for financial statements).

curl

```
# only the reports this key may run, with their filters and columns
curl "https://erp.sa4s.com/api/v1/reports" -H "Authorization: Bearer $SA4S_API_KEY"
```

JavaScript

```
const { items } = await sa4s('GET', '/reports');
console.log(items.map((r) => `${r.name}: ${r.filters.map((f) => f.name).join(', ')}`));
```

Python

```
for r in sa4s("GET", "/reports")["items"]:
    print(r["name"], [f["name"] for f in r["filters"]])
```

PHP

```
<?php
foreach (sa4s('GET', '/reports')['items'] as $r) echo $r['name'], PHP_EOL;
```

200 OK (abridged)

```
{
  "groups": [
    { "key": "sales", "labelAr": "المبيعات", "labelEn": "Sales", "reports": ["sales-daily", "sales-monthly", "sales-by-product"] }
  ],
  "items": [
    {
      "name": "sales-daily",
      "group": "sales",
      "titleAr": "المبيعات اليومية",
      "titleEn": "Daily sales",
      "permissions": ["reports:view", "sales.invoice:view"],
      "paginated": true,
      "dateMode": "range",
      "defaultSort": { "key": "date", "order": "asc" },
      "filters": [ { "name": "dateFrom", "type": "date", "labelAr": "من تاريخ", "labelEn": "From date" }, { "name": "dateTo", "type": "date" }, { "name": "branchId", "type": "uuid" } ],
      "columns": [ { "key": "date", "labelAr": "التاريخ", "labelEn": "Date", "type": "date" } ],
      "formats": ["json", "csv", "xlsx", "pdf"]
    }
  ],
  "canExport": true,
  "canSeeCost": true
}
```



Run a report as JSON

`GET /reports/{name}` with the report's filters (`dateFrom` , `dateTo` , `asOf` , `period` , `branchId` , `warehouseId` , `customerId` , ...; `page` , `pageSize` up to 500). `format=csv` , `xlsx` or `pdf` returns a file and needs `reports:export` . Available names include `sales-daily` , `sales-monthly` , `sales-by-product` , `sales-by-customer` , `sales-by-payment-method` , `product-profitability` , `stock-balance` , `stock-movement` , `low-stock` , `expiring-batches` , `purchases-by-supplier` , `customer-balances` , `receivables-aging` , `profit-loss` , `balance-sheet` , `trial-balance` and `cash-flow` .

curl

```
curl "https://erp.sa4s.com/api/v1/reports/sales-daily?dateFrom=2026-10-01&dateTo=2026-10-31" \
-H "Authorization: Bearer $SA4S_API_KEY"

# the same report as a file (needs reports:export): format=csv | xlsx | pdf, lang=ar | en | both
curl "https://erp.sa4s.com/api/v1/reports/sales-daily?dateFrom=2026-10-01&dateTo=2026-10-31&format=xlsx&lang=en" \
-H "Authorization: Bearer $SA4S_API_KEY" -o sales-october.xlsx
```

JavaScript

```
const r = await sa4s('GET', '/reports/sales-daily?dateFrom=2026-10-01&dateTo=2026-10-31');
for (const row of r.rows) console.log(row.date, row.invoices, row.netSales, row.vat, row.total);
console.log('month total', r.totals.total);
```

Python

```
r = sa4s("GET", "/reports/stock-balance", params={"warehouseId": WAREHOUSE_ID, "status": "low", "pageSize": 500})
for row in r["rows"]:
    print(row["sku"], row["onHand"], row["minStock"])
```

PHP

```
<?php
$r = sa4s('GET', '/reports/sales-by-product?' . http_build_query(['dateFrom' => '2026-10-01', 'dateTo' => '2026-10-31']));
foreach ($r['rows'] as $row) echo json_encode($row, JSON_UNESCAPED_UNICODE), "\n";
```

200 OK (abridged)

```
{
  "report": { "name": "sales-daily", "group": "sales", "titleAr": "المبيعات اليومية", "titleEn": "Daily sales" },
  "filters": { "dateFrom": "2026-10-01", "dateTo": "2026-10-31", "branchId": null, "warehouseId": null },
  "currency": "SAR",
  "generatedAt": "2026-10-06T17:01:03.112Z",
  "columns": [ { "key": "date", "labelAr": "التاريخ", "labelEn": "Date", "type": "date" } ],
  "rows": [
    { "date": "2026-10-06", "invoices": 4, "creditNotes": 0, "quantity": "254", "grossSales": "511.08", "returns": "0.00", "netSales": "511.08", "vat": "76.67", "total": "587.75", "cost": "343.60", "profit": "167.48", "marginPercent": "32.77" },
  ],
  "totals": { "invoices": 4, "netSales": "511.08", "vat": "76.67", "total": "587.75", "profit": "167.48" },
  "summary": { "netSales": "511.08", "total": "587.75", "invoices": 4, "creditNotes": 0 },
  "page": 1,
  "pageSize": 50,
  "total": 1,
  "pages": 1
}
```



Webhooks

Webhooks push events to your HTTPS endpoint as they happen, so you do not have to poll.

Subscribe

1. In the ERP open **Settings** → **API & Integrations** → **Webhooks** → **New endpoint** (webhooks are managed in the ERP, not through API keys).
2. Enter your HTTPS URL, pick the events (or `*` for all) and save. Up to 20 endpoints per company.
3. Copy the signing secret (`whsec_...`) — it is shown once — and store it next to your receiver.
4. Press **Send test**: a signed `webhook.test` event is delivered immediately and the ERP shows your answer.

Delivery format

Each delivery is a `POST` with a JSON body `{ id, event, createdAt, data }` and these headers:

Header	Content
<code>X-SA4S-Event</code>	Event name, e.g. <code>invoice.issued</code>
<code>X-SA4S-Delivery</code>	Delivery id — equals the body <code>id</code> , stays the same across retries: de-duplicate on it
<code>X-SA4S-Timestamp</code>	Unix time (seconds) of this attempt
<code>X-SA4S-Signature</code>	<code>sha256=</code> + hex HMAC-SHA256 of <code>timestamp + "." + raw body</code> , keyed with the endpoint secret

`data` holds the ids plus a small snapshot of the record, so most consumers need no follow-up call; fetch the full record with the API when you need more. Events are delivered **at least once** and may arrive out of order — make handlers idempotent and, when order matters, re-read the current state.

HTTP

```
POST /sa4s/webhooks HTTP/1.1
Host: shop.example.com
Content-Type: application/json
User-Agent: SA4S-Webhooks/1.0
X-SA4S-Event: invoice.issued
X-SA4S-Delivery: 2a695809-856b-43c4-ae7e-ba876cb05daf
X-SA4S-Timestamp: 1791305956
X-SA4S-Signature: sha256=bf28b879951e61d1a313f917e5f07f7bdd884e7e98c46620c724a639165f3cd5

{"id":"2a695809-856b-43c4-ae7e-ba876cb05daf","event":"invoice.issued","createdAt":"2026-10-06T17:01:03.551Z","data":{...}}
```



Event catalogue

Event	When	data
<code>invoice.issued</code>	A sales invoice, credit note or debit note was issued	<code>invoiceId</code> , <code>invoice</code> snapshot
<code>invoice.cancelled</code>	An issued sales invoice was cancelled	<code>invoiceId</code> , <code>invoice</code> snapshot
<code>payment.recorded</code>	A customer receipt or supplier payment was recorded — or cancelled (<code>cancelled: true</code>)	<code>paymentId</code> , <code>direction</code> , optional <code>invoiceId</code> / <code>amount</code> , <code>payment</code> snapshot
<code>stock.changed</code>	The on-hand quantity of a product in a warehouse changed	<code>productId</code> , <code>warehouseId</code> , <code>balance</code> (<code>onHand</code> , <code>reserved</code>)
<code>zatca.status</code>	The ZATCA status of an invoice changed	<code>invoiceId</code> , <code>status</code> , <code>invoice</code> snapshot
<code>webhook.test</code>	You pressed Send test in the ERP	<code>message</code> , <code>endpointId</code> , <code>triggeredBy</code>

invoice.issued

```
{
  "id": "2a695809-856b-43c4-ae7e-ba876cb05daf",
  "event": "invoice.issued",
  "createdAt": "2026-10-06T17:01:03.551Z",
  "data": {
    "invoiceId": "aafe08be-d24b-47a4-9058-61fce56397b6",
    "invoice": {
      "id": "aafe08be-d24b-47a4-9058-61fce56397b6",
      "number": "INV-2026-000005",
      "type": "SIMPLIFIED",
      "status": "ISSUED",
      "paymentStatus": "PAID",
      "customerId": null,
      "branchId": "73afa37e-46d6-4c51-9a13-549d92cad970",
      "issueDate": "2026-10-06T17:01:03.475Z",
      "total": "40"
    }
  }
}
```

invoice.cancelled

```
{
  "id": "57109e0d-53a4-463e-816b-59fc33da1f36",
  "event": "invoice.cancelled",
  "createdAt": "2026-10-06T17:01:03.787Z",
  "data": {
    "invoiceId": "015727de-9146-41eb-839b-fd477dc9e9a8",
    "invoice": {
      "id": "015727de-9146-41eb-839b-fd477dc9e9a8",
      "number": "INV-2026-000006",
      "type": "SIMPLIFIED",
      "status": "CANCELLED",
      "paymentStatus": "UNPAID",
      "customerId": "b4382d00-46be-436d-af45-c3dcd8b0222c",
      "branchId": "73afa37e-46d6-4c51-9a13-549d92cad970",
      "issueDate": "2026-10-06T17:01:03.635Z",
      "total": "40"
    }
  }
}
```

**payment.recorded**

```
{
  "id": "c6ff3cf3-3f1b-4564-bcb7-91181e19eece",
  "event": "payment.recorded",
  "createdAt": "2026-10-06T17:01:03.551Z",
  "data": {
    "paymentId": "1662f93b-01fc-4abc-a45d-96aeda8d6f60",
    "direction": "IN",
    "invoiceId": "aafe08be-d24b-47a4-9058-61fce56397b6",
    "amount": "40",
    "payment": {
      "id": "1662f93b-01fc-4abc-a45d-96aeda8d6f60",
      "number": "RCV-2026-000006",
      "direction": "IN",
      "partyType": "CUSTOMER",
      "partyId": null,
      "method": "CASH",
      "amount": "40",
      "date": "2026-10-06T17:01:03.518Z",
      "reference": null
    }
  }
}
```

stock.changed

```
{
  "id": "48fb6802-8ac4-47f7-87de-b2483fd0e19b",
  "event": "stock.changed",
  "createdAt": "2026-10-06T17:01:03.552Z",
  "data": {
    "productId": "6fe0f6fa-ed4d-4c4b-81ca-c8095504658f",
    "warehouseId": "1ae3935b-6fae-4c7f-803a-bf405dbc0ea0",
    "balance": { "onHand": "42", "reserved": "0" }
  }
}
```

The `zatca.status` payload has the same shape as `invoice.issued` plus the new `status` (values below are illustrative):

zatca.status

```
{
  "id": "0d3c1f7e-5b52-4c8a-9a51-3f8f1d3e2b10",
  "event": "zatca.status",
  "createdAt": "2026-10-06T17:17:12.410Z",
  "data": {
    "invoiceId": "442ea37d-b9ac-4535-b726-b3ad69832afe",
    "status": "REPORTED",
    "invoice": {
      "id": "442ea37d-b9ac-4535-b726-b3ad69832afe",
      "number": "INV-2026-000007",
      "type": "SIMPLIFIED",
      "status": "ISSUED",
      "paymentStatus": "PAID",
      "customerId": null,
      "branchId": "73afa37e-46d6-4c51-9a13-549d92cad970",
      "issueDate": "2026-10-06T17:17:09.161Z",
    }
  }
}
```



```
    "total": "46.9"
  }
}
```

webhook.test

```
{
  "id": "7a8dad24-9ac7-48d7-82d3-62b396f7bdbf",
  "event": "webhook.test",
  "createdAt": "2026-10-06T16:59:16.516Z",
  "data": { "message": "This is a test delivery from SA4S", "endpointId": "b07d0348-c275-41e7-81ae-831247444a00",
  "triggeredBy": "Owner" }
}
```

Verify the signature

Compute the HMAC over the **raw bytes** you received — parsing and re-serialising the JSON changes the bytes and breaks the signature. Compare in constant time.

Node.js

```
import { createHmac, timingSafeEqual } from 'node:crypto';

/** rawBody: the exact bytes received (Buffer) - never a re-serialised JSON object */
export function verifySa4s(headers, rawBody, secret, toleranceSec = 300) {
  const ts = headers['x-sa4s-timestamp'];
  const given = String(headers['x-sa4s-signature'] ?? '');
  if (!ts || Math.abs(Date.now() / 1000 - Number(ts)) > toleranceSec) return false; // replay window
  const expected = 'sha256=' + createHmac('sha256', secret).update(`${ts}.`).update(rawBody).digest('hex');
  return given.length === expected.length && timingSafeEqual(Buffer.from(given), Buffer.from(expected));
}
```

Python

```
import hashlib, hmac, time

def verify_sa4s(headers, raw_body: bytes, secret: str, tolerance=300) -> bool:
    ts = headers.get("X-SA4S-Timestamp", "")
    if not ts.isdigit() or abs(time.time() - int(ts)) > tolerance:
        return False
    expected = "sha256=" + hmac.new(secret.encode(), ts.encode() + b"." + raw_body, hashlib.sha256).hexdigest()
    return hmac.compare_digest(expected, headers.get("X-SA4S-Signature", ""))
```

PHP

```
<?php
function verify_sa4s(string $rawBody, string $secret, int $tolerance = 300): bool
{
    $ts = $_SERVER['HTTP_X_SA4S_TIMESTAMP'] ?? '';
    if (!ctype_digit($ts) || abs(time() - (int) $ts) > $tolerance) return false;
    $expected = 'sha256=' . hash_hmac('sha256', $ts . '.' . $rawBody, $secret);
    return hash_equals($expected, $_SERVER['HTTP_X_SA4S_SIGNATURE'] ?? '');
}

$raw = file_get_contents('php://input');
if (!verify_sa4s($raw, getenv('SA4S_WEBHOOK_SECRET'))) { http_response_code(401); exit; }
$event = json_decode($raw, true);
```



C#

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

static bool VerifySa4s(HttpRequest req, byte[] rawBody, string secret, int toleranceSec = 300)
{
    string ts = req.Headers["X-SA4S-Timestamp"].ToString();
    if (!long.TryParse(ts, out var unix) || Math.Abs(DateTimeOffset.UtcNow.ToUnixTimeSeconds() - unix) > toleranceSec)
        return false;
    using var hmac = new HMACSHA256(Encoding.UTF8.GetBytes(secret));
    byte[] signed = [.. Encoding.UTF8.GetBytes(ts + "."), .. rawBody];
    string expected = "sha256=" + Convert.ToHexString(hmac.ComputeHash(signed)).ToLowerInvariant();
    return CryptographicOperations.FixedTimeEquals(Encoding.UTF8.GetBytes(expected),
        Encoding.UTF8.GetBytes(req.Headers["X-SA4S-Signature"].ToString()));
}
```

Java

```
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import java.nio.charset.StandardCharsets;
import java.security.MessageDigest;
import java.util.HexFormat;

static boolean verifySa4s(String timestamp, String signature, byte[] rawBody, String secret) throws Exception {
    if (timestamp == null || !timestamp.matches("\\d+")) return false;
    if (Math.abs(System.currentTimeMillis() / 1000 - Long.parseLong(timestamp)) > 300) return false;
    Mac mac = Mac.getInstance("HmacSHA256");
    mac.init(new SecretKeySpec(secret.getBytes(StandardCharsets.UTF_8), "HmacSHA256"));
    mac.update((timestamp + ".").getBytes(StandardCharsets.UTF_8));
    String expected = "sha256=" + HexFormat.of().formatHex(mac.doFinal(rawBody));
    return signature != null && MessageDigest.isEqual(expected.getBytes(StandardCharsets.UTF_8),
        signature.getBytes(StandardCharsets.UTF_8));
}
```

Go

```
import (
    "crypto/hmac"
    "crypto/sha256"
    "encoding/hex"
    "math"
    "net/http"
    "strconv"
    "time"
)

func VerifySa4s(r *http.Request, rawBody []byte, secret string) bool {
    ts := r.Header.Get("X-SA4S-Timestamp")
    unix, err := strconv.ParseInt(ts, 10, 64)
    if err != nil || math.Abs(float64(time.Now().Unix()-unix)) > 300 {
        return false
    }
    mac := hmac.New(sh256.New, []byte(secret))
    mac.Write([]byte(ts + "."))
    mac.Write(rawBody)
    expected := "sha256=" + hex.EncodeToString(mac.Sum(nil))
    return hmac.Equal([]byte(expected), []byte(r.Header.Get("X-SA4S-Signature")))
}
```

Replay protection

The timestamp is part of the signed content, so it cannot be altered. Reject deliveries whose `X-SA4S-Timestamp` is more than **5 minutes** away from your clock (the examples above do), and remember processed `X-SA4S-Delivery` ids (a unique index is enough): an attacker who captured a delivery can neither replay it later nor make you process it twice. Keep your server clock synced (NTP).

Retries and redelivery

Answer with any **2xx within 10 seconds** — acknowledge first, then do the work in the background. Redirects are not followed. Anything else (an error status, a timeout, a connection failure) is retried with exponential backoff — after about 30 seconds, then 1, 2, 4 and 8 minutes — **6 attempts** in total, each freshly signed with a new timestamp and the same delivery id. After the last attempt the delivery is marked *dead*. The ERP (**Deliveries**) shows every delivery with its status, attempts and your last response, and lets you **Redeliver** any of them. An endpoint whose deliveries keep dying (25 in a row) is paused automatically; fix it and re-enable it in the ERP.

Node.js (Express)

```
import express from 'express';
import { verifySa4s } from './verify.mjs';

const app = express();
const seen = new Set(); // use your DB (unique index on the delivery id) in production

app.post('/sa4s/webhooks', express.raw({ type: 'application/json' })), (req, res) => {
  if (!verifySa4s(req.headers, req.body, process.env.SA4S_WEBHOOK_SECRET)) return res.sendStatus(401);
  const delivery = req.get('X-SA4S-Delivery');
  if (seen.has(delivery)) return res.sendStatus(200); // retries re-send the same delivery id
  seen.add(delivery);
  const event = JSON.parse(req.body.toString('utf8'));
  queue.push(event); // answer within 10 s - do the real work asynchronously
  res.sendStatus(204);
});

async function handle(event) {
  switch (event.event) {
    case 'invoice.issued': return onInvoice(event.data.invoiceId); // fetch GET /sales/invoices/{id} for
    // full details
    case 'payment.recorded': return onPayment(event.data.paymentId, event.data.cancelled === true);
    case 'stock.changed': return onStock(event.data.productId, event.data.warehouseId, event.data.balance);
    case 'zatca.status': return onZatca(event.data.invoiceId, event.data.status);
    default: return; // ignore unknown events (new ones may be added)
  }
}

app.listen(3000);
```

Python (Flask)

```
import json, os
from flask import Flask, request

app = Flask(__name__)
SECRET = os.environ["SA4S_WEBHOOK_SECRET"]

@app.post("/sa4s/webhooks")
def sa4s_webhook():
    raw = request.get_data() # exact bytes
    if not verify_sa4s(request.headers, raw, SECRET):
        return "", 401
    delivery = request.headers["X-SA4S-Delivery"]
```



```
if already_processed(delivery):           # unique key in your DB
    return "", 200
enqueue(json.loads(raw))                 # process asynchronously, answer fast
mark_processed(delivery)
return "", 204
```

Testing locally

Your endpoint must be reachable over public HTTPS. During development, expose your local receiver through an HTTPS tunnel, use the **Send test** button, and replay captured bodies with a freshly computed signature:

Shell

```
# 1) run your receiver locally
node receiver.mjs           # listens on http://localhost:3000/sa4s/webhooks

# 2) expose it over HTTPS with any tunnel you trust, e.g.
cloudflared tunnel --url http://localhost:3000
#   → https://<random>.trycloudflare.com (use https://<random>.trycloudflare.com/sa4s/webhooks as the endpoint URL)

# 3) ERP → Settings → API & Integrations → Webhooks → Send test: a signed "webhook.test" is delivered immediately

# 4) replay a captured body against your local receiver with a fresh signature (no SA4S needed)
TS=$(date +%s)
BODY='{ "id": "7a8dad24-9ac7-48d7-82d3-62b396f7bdbf", "event": "webhook.test", "createdAt": "2026-10-06T16:59:16.516Z", "data": {} }'
SIG=$(printf '%s.%s' "$TS" "$BODY" | openssl dgst -sha256 -hmac "$SA4S_WEBHOOK_SECRET" -hex | sed 's/^.* //')
curl -X POST http://localhost:3000/sa4s/webhooks -H "Content-Type: application/json" \
  -H "X-SA4S-Event: webhook.test" -H "X-SA4S-Delivery: 7a8dad24-9ac7-48d7-82d3-62b396f7bdbf" \
  -H "X-SA4S-Timestamp: $TS" -H "X-SA4S-Signature: sha256=$SIG" -d "$BODY"
```

End-to-end: online store orders → SA4S invoices

A complete, idempotent order sync for a WooCommerce- or Salla-style store. Trigger it from the store's "order paid" webhook or from a scheduled job over recent paid orders — running it twice for the same order never creates a second invoice.

Flow

```
store "order paid" event
├─ map every line SKU → SA4S productId    GET  /products/lookup?code={sku}      (cached)
├─ B2B order? find / create the customer  GET  /customers?search={vat} · POST /customers
├─ clientId = UUIDv5(store URL + order id) (same order → same key on every retry / duplicate event)
├─ draft invoice, server computes VAT      POST /sales/invoices          { pricesIncludeVat: true, clientId,
lines }
├─ compare draft.total with the store total (±0.05) – stop and flag the order when they differ
├─ issue + record the store payment         POST /sales/invoices/{id}/issue { payments: [{ method: "CARD", amount:
draft.total }] }
└─ save invoice number / PDF on the order   GET  /sales/invoices/{id}/pdf
```

Design choices that keep it robust:

- **Deterministic** `clientId` — UUID v5 of the store URL + order id: duplicate store webhooks, crashes and parallel workers all map to the same invoice.
- **VAT-inclusive store prices** (`pricesIncludeVat: true`) and **final unit prices** (discounts already applied, so the key needs no `sales.discount:manage` scope); shipping as a line of a *SERVICE* product.
- **Draft** → **compare** → **issue** — a price or VAT-category mismatch stops the order for review instead of issuing a wrong invoice.
- B2B orders with a VAT number get a customer and a tax invoice; consumer orders a simplified invoice.

JavaScript

```
// order-sync.mjs – paid store orders → issued SA4S invoices. Idempotent: safe to re-run for the same order.
import { createHash } from 'node:crypto';
import { sa4s, Sa4sError } from './sa4s.mjs';

const SHOP = 'https://shop.example.com';
const SHIPPING_SKU = 'SRV-DELIVERY'; // a SERVICE product created once in SA4S for shipping fees
const URL_NAMESPACE = '6ba7b811-9dad-11d1-80b4-00c04fd430c8'; // RFC 4122

/** RFC 4122 version-5 UUID: deterministic, so the same order always maps to the same clientId */
export function uuidV5(name, namespace = URL_NAMESPACE) {
  const h = createHash('sha1').update(Buffer.from(namespace.replace(/-/g, ''), 'hex')).update(name, 'utf8').digest();
  h[6] = (h[6] & 0x0f) | 0x50;
  h[8] = (h[8] & 0x3f) | 0x80;
  const x = h.subarray(0, 16).toString('hex');
  return `${x.slice(0, 8)}-${x.slice(8, 12)}-${x.slice(12, 16)}-${x.slice(16, 20)}-${x.slice(20)}`;
}

const productIds = new Map();
async function productIdOf(sku) {
  if (!productIds.has(sku)) productIds.set(sku, (await sa4s('GET', `/products/lookup?code=${encodeURIComponent(sku)}`).product.id);
  return productIds.get(sku);
}

async function customerIdOf(billing) {
  if (!billing.vatNumber) return undefined; // B2C → simplified invoice without a customer
```

```
const { items } = await sa4s('GET', `/customers?search=${billing.vatNumber}&pageSize=5`);
const found = items.find((c) => c.vatNumber === billing.vatNumber);
if (found) return found.id;
return (await sa4s('POST', '/customers', {
  type: 'COMPANY', name: billing.company, vatNumber: billing.vatNumber, phone: billing.phone, email: billing.email,
  city: billing.city,
})).id;
}

export async function syncOrder(order) {
  const clientId = uuidV5(`${SHOP}/orders/${order.id}`);
  const lines = [];
  for (const item of order.items) {
    // discounts need a permission API keys do not have: send the final (VAT-inclusive) unit price instead
    lines.push({ productId: await productIdOf(item.sku), quantity: String(item.quantity), unitPrice:
item.finalUnitPrice, description: item.name });
  }
  if (Number(order.shippingTotal) > 0) lines.push({ productId: await productIdOf(SHIPPING_SKU), quantity: '1',
unitPrice: order.shippingTotal });
  const customerId = await customerIdOf(order.billing);

  const draft = await sa4s('POST', '/sales/invoices', {
    type: customerId ? 'TAX' : 'SIMPLIFIED', customerId, pricesIncludeVat: true, clientId, notes: `Store order
#${order.number}`, lines,
  });
  if (draft.status !== 'DRAFT') return draft; // issued by an earlier run (idempotent replay)
  if (Math.abs(Number(draft.total) - Number(order.total)) > 0.05) {
    throw new Error(`order ${order.number}: SA4S total ${draft.total} ≠ store total ${order.total} – check prices / VAT
categories`);
  }
  try {
    return await sa4s('POST', `/sales/invoices/${draft.id}/issue`, { payments: [{ method: 'CARD', amount: draft.total,
reference: order.paymentId }] });
  } catch (e) {
    if (e instanceof Sa4sError && e.code === 'INVOICE_NOT_ISSUABLE') return sa4s('GET', `/sales/invoices/${draft.id}`);
    // a parallel run won
    throw e;
  }
}
```

Python

```
# order_sync.py – paid store orders → issued SA4S invoices (idempotent)
import uuid
from decimal import Decimal
from sa4s import sa4s, Sa4sError

SHOP = "https://shop.example.com"
SHIPPING_SKU = "SRV-DELIVERY"
_product_ids = {}

def product_id_of(sku):
    if sku not in _product_ids:
        _product_ids[sku] = sa4s("GET", "/products/lookup", params={"code": sku})["product"]["id"]
    return _product_ids[sku]

def customer_id_of(billing):
    vat = billing.get("vat_number")
    if not vat:
        return None
    found = [c for c in sa4s("GET", "/customers", params={"search": vat, "pageSize": 5})["items"] if c["vatNumber"] ==
vat]
    if found:
        return found[0]["id"]
    return sa4s("POST", "/customers", {"type": "COMPANY", "name": billing["company"], "vatNumber": vat, "phone":
```



```

billing["phone"]}}]["id"]

def sync_order(order):
    client_uuid = str(uuid.uuid5(uuid.NAMESPACE_URL, f"{SHOP}/orders/{order['id']}"))
    lines = [{"productId": product_id_of(i["sku"]), "quantity": str(i["quantity"]), "unitPrice": i["final_unit_price"]}
    for i in order["items"]]
    if Decimal(order["shipping_total"]) > 0:
        lines.append({"productId": product_id_of(SHIPPING_SKU), "quantity": "1", "unitPrice": order["shipping_total"]})
    customer_id = customer_id_of(order["billing"])
    body = {"type": "TAX" if customer_id else "SIMPLIFIED", "pricesIncludeVat": True, "clientId": client_uuid,
           "notes": f"Store order #{order['number']}", "lines": lines}
    if customer_id:
        body["customerId"] = customer_id
    draft = sa4s("POST", "/sales/invoices", body)
    if draft["status"] != "DRAFT":
        return draft
    if abs(Decimal(draft["total"]) - Decimal(order["total"])) > Decimal("0.05"):
        raise ValueError(f"order {order['number']}: SA4S {draft['total']} != store {order['total']}")
    try:
        return sa4s("POST", f"/sales/invoices/{draft['id']}/issue",
                    {"payments": [{"method": "CARD", "amount": draft["total"], "reference": order["payment_id"]}]}
    except Sa4sError as e:
        if e.code == "INVOICE_NOT_ISSUABLE":
            return sa4s("GET", f"/sales/invoices/{draft['id']}")
        raise

```

PHP

```

<?php
// order-sync.php - paid store orders → issued SA4S invoices (idempotent)
require 'sa4s.php';

function uuid_v5(string $name, string $ns = '6ba7b811-9dad-11d1-80b4-00c04fd430c8'): string
{
    $h = sha1(hex2bin(str_replace('-', '', $ns)) . $name);
    $h = substr($h, 0, 12) . '5' . substr($h, 13, 3) . dechex(hexdec($h[16]) & 0x3 | 0x8) . substr($h, 17, 15);
    return vsprintf('%s-%s-%s-%s-%s', [substr($h, 0, 8), substr($h, 8, 4), substr($h, 12, 4), substr($h, 16, 4),
    substr($h, 20, 12)]);
}

function sync_order(array $order): array
{
    static $productIds = [];
    $pid = function (string $sku) use (&$productIds) {
        return $productIds[$sku] ??= sa4s('GET', '/products/lookup?code=' . urlencode($sku))['product']['id'];
    };
    $lines = array_map(fn ($i) => ['productId' => $pid($i['sku']), 'quantity' => (string) $i['quantity'], 'unitPrice'
=> $i['final_unit_price']], $order['items']);
    if ((float) $order['shipping_total'] > 0) {
        $lines[] = ['productId' => $pid('SRV-DELIVERY'), 'quantity' => '1', 'unitPrice' => $order['shipping_total']];
    }
    $draft = sa4s('POST', '/sales/invoices', [
        'type' => 'SIMPLIFIED',
        'pricesIncludeVat' => true,
        'clientId' => uuid_v5("https://shop.example.com/orders/{$order['id']}"),
        'notes' => "Store order #{$order['number']}",
        'lines' => $lines,
    ]);
    if ($draft['status'] !== 'DRAFT') return $draft;
    if (abs((float) $draft['total'] - (float) $order['total']) > 0.05) {
        throw new RuntimeException("order {$order['number']}: SA4S {$draft['total']} != store {$order['total']}");
    }
    try {
        return sa4s('POST', "/sales/invoices/{$draft['id']}/issue", ['payments' => [['method' => 'CARD', 'amount' =>
$draft['total'], 'reference' => $order['payment_id']]]]);
    }
}

```



```
    } catch (Sa4sError $e) {  
        if (($e->body['code'] ?? '') === 'INVOICE_NOT_ISSUABLE') return sa4s('GET', "/sales/invoices/${$draft['id']}");  
        throw $e;  
    }  
}
```

Afterwards store the invoice `number` on the order and offer the PDF (`GET /sales/invoices/{id}/pdf`) as the customer's tax invoice. For refunds use a `credit note` with its own deterministic `clientId` (e.g. UUID v5 of the refund id).

End-to-end: keep stock in sync

Publish SA4S stock to the store in three layers: a full snapshot, real-time webhooks, and a periodic reconciliation that repairs anything a missed or delayed event left behind.

Flow

- A. first run / nightly GET /inventory/balances?warehouseId={wh}&pageSize=200 (all pages) → set every store quantity by SKU
- B. real time webhook stock.changed { productId, warehouseId, balance.onHand, balance.reserved }
 → available = onHand - reserved → update that one SKU
- C. safety net (15 min) GET /inventory/movements?from={yesterday}&order=asc → products you have not seen
 → GET /inventory/products/{productId}/stock → update those SKUs

JavaScript

```
// stock-sync.mjs - keep store quantities equal to the SA4S "available" stock of one warehouse
import { sa4s } from './sa4s.mjs';

const WAREHOUSE_ID = process.env.SA4S_WAREHOUSE_ID;
const skuCache = new Map();
const skuOf = async (productId) => {
  if (!skuCache.has(productId)) skuCache.set(productId, (await sa4s('GET', `/products/${productId}`).sku);
  return skuCache.get(productId);
};
const toStoreQty = (decimal) => Math.max(0, Math.floor(Number(decimal))); // base units, never negative

// A) full sync
export async function fullSync(store) {
  for (let page = 1; ; page++) {
    const res = await sa4s('GET', `/inventory/balances?warehouseId=${WAREHOUSE_ID}&page=${page}&pageSize=200&sort=sku&order=asc`);
    for (const row of res.items) await store.setQuantity(row.product.sku, toStoreQty(row.available));
    if (page * res.pageSize >= res.total) break;
  }
}

// B) webhook handler for stock.changed (see "Webhooks")
export async function onStockChanged(store, { productId, warehouseId, balance }) {
  if (warehouseId !== WAREHOUSE_ID || !balance) return;
  await store.setQuantity(await skuOf(productId), toStoreQty(Number(balance.onHand) - Number(balance.reserved)));
}

// C) reconciliation - catches anything a missed / delayed webhook did not deliver
export async function reconcile(store, seen = new Set()) {
  const yesterday = new Intl.DateTimeFormat('en-CA', { timeZone: 'Asia/Riyadh' }).format(new Date(Date.now() - 86_400_000));
  const touched = new Set();
  for (let page = 1; ; page++) {
    const res = await sa4s('GET', `/inventory/movements?from=${yesterday}&warehouseId=${WAREHOUSE_ID}&order=asc&page=${page}&pageSize=200`);
    for (const m of res.items) if (!seen.has(m.id)) { seen.add(m.id); touched.add(m.productId); }
    if (page * res.pageSize >= res.total) break;
  }
  for (const productId of touched) {
    const s = await sa4s('GET', `/inventory/products/${productId}/stock`);
    const row = s.warehouses.find((w) => w.warehouseId === WAREHOUSE_ID);
    await store.setQuantity(s.product.sku, toStoreQty(row?.available ?? '0'));
  }
}
```



Python

```
# stock_sync.py
import math, os
from datetime import datetime, timedelta
from zoneinfo import ZoneInfo
from sa4s import sa4s

WAREHOUSE_ID = os.environ["SA4S_WAREHOUSE_ID"]
to_store_qty = lambda d: max(0, math.floor(float(d)))

def full_sync(store):
    page = 1
    while True:
        res = sa4s("GET", "/inventory/balances", params={"warehouseId": WAREHOUSE_ID, "page": page, "pageSize": 200,
"sort": "sku", "order": "asc"})
        for row in res["items"]:
            store.set_quantity(row["product"]["sku"], to_store_qty(row["available"]))
        if page * res["pageSize"] >= res["total"]:
            return
        page += 1

def on_stock_changed(store, data):
    # webhook "stock.changed"
    if data["warehouseId"] != WAREHOUSE_ID or "balance" not in data:
        return
    sku = sa4s("GET", f"/products/{data['productId']}")["sku"]
    store.set_quantity(sku, to_store_qty(float(data["balance"]["onHand"]) - float(data["balance"]["reserved"])))

def reconcile(store, seen):
    yesterday = (datetime.now(ZoneInfo("Asia/Riyadh")) - timedelta(days=1)).strftime("%Y-%m-%d")
    touched, page = set(), 1
    while True:
        res = sa4s("GET", "/inventory/movements", params={"from": yesterday, "warehouseId": WAREHOUSE_ID, "order":
"asc", "page": page, "pageSize": 200})
        for m in res["items"]:
            if m["id"] not in seen:
                seen.add(m["id"])
                touched.add(m["productId"])
        if page * res["pageSize"] >= res["total"]:
            break
        page += 1
    for product_id in touched:
        s = sa4s("GET", f"/inventory/products/{product_id}/stock")
        row = next((w for w in s["warehouses"] if w["warehouseId"] == WAREHOUSE_ID), {"available": "0"})
        store.set_quantity(s["product"]["sku"], to_store_qty(row["available"]))
```

PHP

```
<?php
// stock-sync.php - webhook part (A and C work like the JavaScript tab)
require 'sa4s.php';

$raw = file_get_contents('php://input');
if (!verify_sa4s($raw, getenv('SA4S_WEBHOOK_SECRET'))) { http_response_code(401); exit; }
$event = json_decode($raw, true);
if ($event['event'] === 'stock.changed' && $event['data']['warehouseId'] === getenv('SA4S_WAREHOUSE_ID')) {
    $d = $event['data'];
    $sku = sa4s('GET', "/products/{d['productId']}")['sku'];
    $qty = max(0, (int) floor((float) $d['balance']['onHand'] - (float) $d['balance']['reserved']));
    store_set_quantity($sku, $qty); // your store's API
}
http_response_code(204);
```

**Note**

Quantities are in base units. If the store sells cartons, divide by the unit `factor` from the product. Use `available` (on hand minus reserved), never `onHand`, for what can be sold.



Changelog and versioning

Edition	Date	Changes
1.1.0	2026-10-06	Guide rewritten from one source for the ERP and the PDF; recipes for every module with real responses; webhook payloads for every event; signature verification in six languages; end-to-end store examples; corrected filters (reports use <code>dateFrom</code> / <code>dateTo</code> , PDF <code>format=a4 receipt</code>).
1.0.0	2026-10	Developer API launched: API keys (live/sandbox, scopes, rotation, IP allow-list, rate limits), webhooks, OpenAPI reference.

The API version stays **v1** while changes are additive (see [Base URL and versioning](#)). The current OpenAPI document is always at `GET https://erp.sa4s.com/api/v1/developer/openapi.json`.



Support

- Start with the error `code` and the table in [Error codes](#); most problems are a missing scope or a validation message.
- When you contact support, include the `requestId`, the endpoint, the time (with time zone) and the key **prefix** — never the full secret.
- Support channel: the contact details published by SA4S for your subscription (ask your account owner).

Generated from the same source as the in-app guide. The interactive reference is in Settings → API & Integrations. ·

<https://erp.sa4s.com/api/v1/developer/openapi.json>