

Opter TextEDI API

Integration Guide

Version 1.3.2 • 2026

for integration partners

1. Overview

The Opter TextEDI API provides a single unified endpoint for all transport communication. Whether you are booking orders, reporting shipment status, confirming deliveries, or revoking orders, every message goes through the same endpoint.

Endpoint: POST /api/textedi

Content-Type: application/json

The payload key inside the JSON body determines what operation is performed. Order bookings are sent as a single JSON object. Event messages (status, package scans, freight-bill scans, proof-of-delivery, revocations, deviations, price items, attachments) are sent as a JSON array containing one object.

⚠ Important: Opter is a transport management system used independently by many different carriers. There is no central Opter API platform. When you integrate using this guide, you are connecting directly to a specific carrier's own Opter instance.

This means:

- **All codes are carrier-specific.** Service codes, vehicle codes, customer codes, hub codes, and any transporter-specific fields are configured individually by each carrier in their own Opter system. A code that works with one carrier will not work with another.
- **Your API key is carrier-specific.** It grants access only to that carrier's instance.
- **Configuration must be agreed with your carrier before starting integration.** Do not assume any codes or features are available without confirmation.

If you integrate with multiple carriers, each integration is separate and will have its own endpoint, API key, and set of configured codes.

Payload key	Message types	Description
order	new, update, merge, pricerequest, revoke	Create, modify, merge, price-check, or cancel transport orders
status	departure, arrival and four terminal codes (see 4.3)	Order-level status with all packages and GPS coordinates
packagescan	18 codes incl. terminal, external terminal, floorcheck and undo variants (see 4.4)	Individual package scanned at a terminal
freightbillscan	Same 18 codes as packagescan	Freight-bill barcode scanned; requires freightBillNumber
pod	arrival, partialdelivery	Proof-of-delivery with optional signature data
revoke	orderRevoked	Order revocation / cancellation
deviation	Carrier-configured deviation code	Deviation / exception during transport
priceitem	Carrier-configured price item code	Price line item on an order
attachment	cmr, freightbill, awb, pod, deviation, image, customsdocument, edi, packagelabel	File attachment (URL or base64)
response	imported, accepted, failed, rejected	Returned by Opter as the answer to an order (see 5)

Every message shares a common envelope with these fields:

Field	Description
schema	Required. JSON Schema URI defining the message structure (see 8).
version	Required. Schema version (currently "1.0").
eventCreatedDateTime	Required. Timestamp when the message was created (YYYY-MM-DDTHH:MM:SS, local time, no timezone offset).
eventSender	Optional. Free-text identifier of the message sender, usually agreed during integration setup. Not validated — used for debugging and tracing only.

Field	Description
eventReceiver	Optional. Free-text identifier of the message receiver, usually agreed during integration setup. Not validated — used for debugging and tracing only.

i eventSender and eventReceiver are not mandatory. They are plain free text. Opter does not validate them, does not match them against any configuration, and does not route on them. Their only purpose is to make it possible to tell where a message came from when tracing or debugging a flow, so pick values that are meaningful to you and your carrier and keep them stable. Leaving them empty is valid.

1.1 Mapping Codes and Customer Codes

Before sending anything, it is worth knowing which values in this API are free text and which must already exist in the carrier's Opter instance.

Any field whose name ends in MappingCode is a mapping code. The value you send is not stored as text — it is looked up against a mapping configured in the carrier's Opter system. That covers serviceMappingCode, vehicleMappingCode, optionMappingCode, looseItemMappingCode, priceitemMappingCode, transporterHubMappingCode, and any other field following the same pattern. payingCustomerCode works the same way even though it does not carry the suffix: it must match a customer that already exists in the carrier's Opter instance.

What happens on a mismatch depends on which field it is, and the difference matters:

- **Rejecting:** serviceMappingCode, vehicleMappingCode, and payingCustomerCode. If one of these is wrong or missing and the carrier has no default configured, the whole message is rejected and nothing is created or updated. These are the errors listed in 5.2.
- **Warning only:** every other mapping code — optionMappingCode, looseItemMappingCode, priceitemMappingCode, transporterHubMappingCode, regionMappingCode, and the rest. The order or update still goes through; the unmatched value is simply not applied and you get a warning back. Read response.warnings (see 5.1) — a successful import can still mean part of your data was ignored.

Field	Must match	If it doesn't match
serviceMappingCode, vehicleMappingCode	A service / vehicle type configured in the carrier's Opter instance.	Message rejected, unless the carrier has configured a default.
payingCustomerCode	An existing customer (freight payer) in the carrier's Opter instance.	Message rejected.
other *MappingCode	A mapping configured in the carrier's Opter instance — agreed with the carrier before use.	Warning only. The order or update is processed; the value is not applied.
transporterCustomerCode	A known customer address in Opter (see 3.3).	The address is left empty.
shipmentId	An existing leg of the order — looked up on the statistics endpoint of the Opter API V1 (see 3.9).	The leg assignment does not land.
resourceCode, internalVehicleCode	An existing subcontractor or vehicle — looked up on the subcontractor and vehicle endpoints of the Opter API V1 (see 3.9).	The assignment does not land.

Everything else — references, messages, driver instructions, goodsType, packageTypeCode, eventSender, eventReceiver — is free text and is not validated against carrier configuration. Fields that are not part of the schema at all are rejected.

⚠ RULE OF THUMB

If a value was not agreed with your carrier, do not send it.

Mapping codes are carrier-specific. A code that works with one carrier will not work with another.

A rejection creates nothing. A warning means the message went through with part of your data ignored — so always read response.warnings, not just the HTTP status.

1.2 Change History

Version 1.3.2

Section	Change
3.1, 3.2, 4.2, 7.1	estimatedTransportCost is accepted on the order object (number or null) — 1.3.1 wrongly said it was rejected. Do not populate it unless your carrier asks.
3.3, 7.2	Country-specific zipCode lengths extended: DE, EE, FI, FR, and SE exactly 5 characters; DK and NO exactly 4.
3.5	Documented opterId on packages — present only when Opter exports an order.
3.7	documentType is required on every document.
3.11, 6.17	New order-level goodsItems array for customs goods lines, with a cross-border booking example.
4.3, 4.5, 4.8, 6.7, 6.8, 6.14	Breaking: POD and deviation image fields (podSignatureUrl, podSignatureImageBase64encoded, deviationUrl, deviationImageBase64encoded) are now arrays holding several images. Wrap a single value in an array.
8	Booking schema also published as opterOrderBookingjsonSchema1.0.json (lower-case j) — the same file.

Version 1.3.1

Section	Change
1, 3.1, 3.2	Envelope fields schema, version, and eventCreatedDateTime marked as required. Overview table now lists all ten payload keys.
3.1, 3.2, 7.1	estimatedTransportCost removed from the order fields — it is returned in responses and events and is rejected on the order object.
3.2, 3.3	Corrected maximum lengths: customerOrderReference 100; office messages and order-level driverInstructions 2048; address-level driverInstructions 255. Added country-specific zipCode lengths.
3.5, 3.6	quantity must be 1 (enforced by the JSON Schema). Added tunnelRestriction and transportIndex to dangerous goods.
3.8	Clarified which transporter-specific fields exist at package level, the 255-character limit for package-level extralids, and that unmatched mapping codes raise a warning (consistent with 1.1).
4	opterOrderReference is required on every event. Full code lists for status and packagescan; added podComment; revoke codes are carrier-specific. New sections 4.7–4.10 for freightbillscan, deviation, priceitem, and attachment.
5, 5.2, 7.3, 8	Events and responses share one schema, opterEventJsonSchema1.0.json (opterEventResponseSchema1.0.json is the same file). Error list wording; validate against the published JSON Schemas, which now enforce the rules in this guide.
6.13–6.16	New examples for freight-bill scan, deviation, price item, and attachment events.

Version 1.3

Section	Change
1, 1.1	New section explaining mapping codes and customer codes, including which mismatches reject a message and which only raise a warning. Clarified that eventSender and eventReceiver are optional free text used for tracing only.
3.2, 3.9	New order fields shipmentId, resourceCode, and internalVehicleCode for assigning a vehicle or resource to a specific leg of an order.
3.5	Clarified that packageTypeCode is free text, optionally mapped by the carrier.
3.8	Price items documented in full: 41 fields across identification, pricing, flags, subcontractor invoicing, material site, and custom slots. Corrected the path to transporterSpecificFields.priceitems.
3.10	New order-level looseItems array (looseItemMappingCode, startingQuantity, endingQuantity).
6.11, 6.12	New examples: internal-vehicle and subcontractor leg assignments, and an order with loose items.
7.1, 8	Guidance on leg assignment, loose items, and Opter API V1 access.

2. Authentication

All API requests require an API key passed in the request header:

```
X-API-Key: your_api_key_here
```

Contact your carrier, who will coordinate with the Opter EDI team to provide your API key. Keep this key secure and never expose it in client-side code.

3. Order Booking

Order bookings use the `order` payload key and are sent as a single JSON object (not an array). The `messageType` field determines the operation.

3.1 Message Types

messageType	Description	Notes
new	Create a new transport order	Default operation
update	Modify an existing order	Requires <code>opterOrderReference</code> or <code>customerOrderReference</code>
merge	Merge packages into an existing order	Requires carrier configuration. Requires <code>opterOrderReference</code> or <code>customerOrderReference</code>
pricerequest	Request a price quote	Requires carrier agreement
revoke	Cancel an existing order	Some carriers prefer manual cancellation. Requires <code>opterOrderReference</code> or <code>customerOrderReference</code>

⚠ CRITICAL RULES FOR ORDER BOOKINGS

Exactly 2 addresses required for new and pricerequest: one pickup and one delivery. Multi-stop is not supported.

Use EITHER `orderDimensions` OR package-level dimensions — never both.

`payingCustomerCode` must match a valid code configured in Opter.

`serviceMappingCode` and `vehicleMappingCode` must be pre-configured with your carrier.

Do NOT populate `estimatedTransportCost` unless your carrier instructs you to — pricing is calculated by the carrier.

3.2 Order Fields

All fields below are within the `order` object. Only `messageType` is required; all others are optional unless noted. The `order` object does not accept fields that are not listed in this guide.

Field	Type	Description
<code>messageType</code>	enum	Required. One of: new, update, merge, pricerequest, revoke
<code>customerOrderReference</code>	string	Your order reference / ID. Used to correlate responses and for update/merge/revoke if <code>opterOrderReference</code> is unavailable. Max 100 chars.
<code>opterOrderReference</code>	string	Opter's internal order number, returned after successful creation. Use for update/merge/revoke. Preferred over <code>customerOrderReference</code> . Max 50 chars.
<code>shipmentId</code>	integer	Opter's internal ID of a single leg (shipment) of the order. Use on update to target one leg, typically when assigning a vehicle or resource. Value must be fetched from the statistics endpoint of the Opter API V1 — see 3.9. Requires carrier agreement.
<code>payingCustomerCode</code>	string	Freight payer code in Opter. Must match a pre-configured value. Max 50 chars.
<code>serviceMappingCode</code>	mappingEnum	Maps to a service type in Opter (e.g., standard, express). Must be pre-configured. Max 50 chars.
<code>vehicleMappingCode</code>	mappingEnum	Maps to a vehicle type in Opter (e.g., van, truck). Must be pre-configured. Max 50 chars.
<code>estimatedTransportCost</code>	number	Estimated transport cost, usually in the carrier's local currency (nullable). Do not populate unless your carrier explicitly instructs you to — pricing is calculated by the carrier, and a value sent here is not a price agreement. Opter also returns its own estimate on responses and events (see 4.2).

Field	Type	Description
resourceCode	string	Subcontractor to assign to the leg identified by shipmentId, when that leg is bought externally. Mutually exclusive with internalVehicleCode; must be sent with shipmentId. Value must be fetched from the subcontractor/resource endpoint of the Opter API V1 — see 3.9. Requires carrier agreement. Max 50 chars.
internalVehicleCode	string	The carrier's own vehicle to assign to the leg identified by shipmentId. Mutually exclusive with resourceCode; must be sent with shipmentId. Not the same as vehicleMappingCode, which selects a vehicle type. Value must be fetched from the vehicle endpoint of the Opter API V1 — see 3.9. Requires carrier agreement. Max 50 chars.
transportDate	string	Planned transport date (YYYY-MM-DD). Address time windows provide more precise scheduling.
freightBillNumber	string	Freight bill / consignment number. Max 50 chars.
customerProject	string	Project name for grouping/reporting/invoicing. Max 50 chars.
projectNumber	string	Project number for grouping/reporting/invoicing. Max 50 chars.
sendersReference	string	Sender's reference for this shipment. Max 50 chars.
receiversReference	string	Receiver's reference for this shipment. Max 50 chars.
goodsType	string	General goods type description. Max 50 chars.
officeMessage	string	Internal message for carrier office staff. Max 2048 chars.
officeExtraMessage	string	Additional internal message. Max 2048 chars.
officeExternalMessage	string	External message (may appear on documents). Max 2048 chars.
driverInstructions	string	General instructions for the driver. Max 2048 chars.

3.3 Addresses

Every new and pricerequest order must have exactly one pickup address and one delivery address. No other address types are supported, and multi-stop routes require separate bookings.

Field	Type	Description
addressType	enum	Required. pickup or delivery.
earliestTransportDateTime	string	Earliest acceptable time (YYYY-MM-DDTHH:MM:SS). Empty if no constraint.
latestTransportDateTime	string	Latest acceptable time (YYYY-MM-DDTHH:MM:SS). Empty if no constraint.
transporterCustomerCode	string	Requires carrier agreement. Only use if this address is a known customer address in Opter. Opter autofills address details. If the code doesn't match, the address will be empty.
transporterHubMappingCode	mappingEnum	Carrier-configured hub/terminal code. Max 50 chars.
name	string	Company or person name. Max 50 chars.
careOf	string	Care-of name (c/o). Max 50 chars.
streetAddress	string	Street name and building number. Max 50 chars.
addressLine2	string	Additional address line (floor, suite). Max 50 chars.
addressLine3	string	Additional address line. Max 50 chars.
zipCode	string	Postal / ZIP code. Max 50 chars. Country-specific length: DE, EE, FI, FR, SE exactly 5 characters; DK, NO exactly 4 characters.
city	string	City name. Max 50 chars.
countryCode	string	ISO 3166-1 alpha-2 code (e.g., SE, NO, DK). Exactly 2 uppercase letters.

Field	Type	Description
longitude	number	GPS longitude (WGS 84). Optional but improves routing.
latitude	number	GPS latitude (WGS 84). Optional but improves routing.
entreCode	string	Door / entrance code/contact name for the driver. Max 50 chars.
phoneNumber	string	Contact phone. Recommended for driver coordination. Max 50 chars.
emailAddress	string	Contact email. Max 50 chars.
driverInstructions	string	Address-specific driver instructions. Max 255 chars.

3.4 Dimensions

⚠ CHOOSE ONE LEVEL ONLY

Option 1 (PREFERRED): Provide dimensions at package level in the packages array. Omit orderDimensions.

Option 2: Provide dimensions at order level in orderDimensions. Omit package dimensions.

Using both simultaneously will cause calculation errors.

Order-level dimensions (use only when package details are unavailable):

Field	Type	Description
orderPackageQuantity	integer	Total number of packages
orderWeightKgm	number	Total weight in kilograms
orderVolumeCbm	number	Total volume in cubic meters
orderEuropalletSpace	number	Total europallet spaces
orderLoadmeterMtr	number	Total load meters

3.5 Packages

When providing packages, include dimensions at the package level (preferred). If the packages array is empty or absent on a new or pricerequest order, orderDimensions must be provided instead.

⚠ ONE BARCODE = ONE PACKAGE OBJECT

Every physical package must be its own object in the packages array, even if multiple packages share identical dimensions.

Always set quantity to 1 per package object. Do NOT group multiple packages into one object with a higher quantity — any other value fails schema validation.

Each package object can have its own unique barcode. The barcode is used for individual package tracking through terminal scans and proof-of-delivery.

Example: 3 identical boxes = 3 separate package objects, each with quantity: 1 and its own barcode (if applicable).

Field	Type	Description
opterId	integer	Opter's internal ID of the package row (nullable). Only present when Opter exports an order — leave it out when sending a booking.
customerReference	string	Your reference for this package. Max 50 chars. (Can be set to the same as barcode unless specific IDs are needed.)
barcode	string	One unique barcode per package object. Used for tracking and actual scanning of labels. Max 50 chars.
packageTypeCode	string	Package type (e.g., EUR, PC, BOX). Free text — despite the name this is not a mapping code and Opter does not validate it. A carrier may choose to map the values you send onto the package types configured in their own setup, but whether that mapping exists is up to the carrier. Agree the values you will use before relying on them. Max 50 chars.

Field	Type	Description
text	string	Free-text description of contents. Max 50 chars.
goodsType	string	Goods type classification. Max 50 chars.

Package-level dimensions:

Field	Type	Description
quantity	integer	Always 1. Each physical package is its own object in the array.
weightKgm	number	Weight in kilograms
volumeCbm	number	Volume in cubic meters
europalletSpace	number	Europallet spaces consumed
loadmeterMtr	number	Load meters consumed
widthMtr	number	Width in meters
heightMtr	number	Height in meters
depthMtr	number	Depth (length) in meters

3.6 Dangerous Goods

Proper dangerous goods declaration is required by law when shipping hazardous materials. Always use official UN numbers and proper shipping names.

Field	Type	Description
unNumber	integer	UN number (1–9999) identifying the hazardous substance
properShippingName	string	Official proper shipping name per IMDG/ADR. Max 50 chars.
class	string	Hazard class (e.g., "1.3", "2", "3")
packingGroup	string	I (highest danger), II, or III
amount	number	Quantity of the substance
unit	string	Unit of measure (e.g., kg, liter). Max 50 chars.
tunnelRestriction	string	Tunnel restriction code for the dangerous goods. Max 50 chars.
transportIndex	string	Transport index of the dangerous goods. Max 50 chars.
barcodes	array	Barcodes of packages containing this item. At least one; no duplicates.

3.7 Documents

Attach documents using either base64-encoded data (inline) or a URL (external link). Provide one, not both.

Field	Type	Description
documentType	string	Required. freightbill, packagelabel, or edi
dataBase64encoded	string	Base64-encoded file content. Use this OR url.
url	string	URL to download the document. Use this OR dataBase64encoded. Max 255 chars. Note: the endpoint cannot require authentication that cannot be provided in the URL (i.e. OAuth2 is not supported).
fileName	string	File name with extension (e.g., freightbill_123.pdf). Max 50 chars.
comment	string	Description or comment. Max 255 chars.

3.8 Transporter-Specific Fields

⚠ CARRIER AGREEMENT REQUIRED

ALL fields in this section require prior agreement and configuration with your carrier.

Fields not listed in this guide are rejected. Unmatched mapping codes (e.g. `priceitemMappingCode`, `regionMappingCode`) are not applied and are reported in `response.warnings` — see 1.1.

Contact your carrier before using any of these fields.

`transporterSpecificFields` exists at order level and at package level. At package level it holds only the custom fields `extraId1–5` and `extraDimension1–5`; the control flags, `region/trail/query-set` codes, and price items are order level only. All of them require prior agreement with your carrier — an unagreed value is either rejected or not applied, never silently honoured.

Custom Fields

`extraId1–5` and `extraDimension1–5` are generic, carrier-defined slots — they have no fixed meaning in Opter itself. Each carrier decides what to map them to. For example, one carrier might use `extraId1` for an invoice reference, while another carrier uses `extraId1` for something completely unrelated. Always confirm the intended meaning of each slot with your carrier before populating it — do not assume a mapping based on another integration.

Field	Type	Description
<code>extraId1 – extraId5</code>	string	Generic custom text field. Meaning is defined per carrier (e.g. invoice reference for one carrier, something else entirely for another). Max 50 chars at order level, 255 chars at package level.
<code>extraDimension1 – extraDimension5</code>	decimal	Generic custom numeric field. Meaning is defined per carrier (e.g. an extra measurement or count).
<code>regionMappingCode</code>	mappingEnum	Carrier-configured region code for routing. Order level only. Max 50 chars.
<code>trailMappingCode</code>	mappingEnum	Carrier-configured trail/route code. Order level only. Max 50 chars.
<code>querySetMappingCode</code>	mappingEnum	Carrier-configured query set code. Order level only. Max 50 chars.

Control Flags

Boolean switches are available at order level. The precise effect of each flag is described below; note that these fields are almost never sent in most integrations. An exception might be integrations between a carrier and its internal systems.

Field	Type	Description
<code>readyForInvoicing</code>	boolean	Marks the order as ready to be invoiced by the carrier.
<code>readyForDispatching</code>	boolean	Marks the order as ready for the carrier to dispatch operationally.
<code>notToBeInvoiced</code>	boolean	Excludes the order from the carrier's normal invoicing flow.
<code>hideUntilScanned</code>	boolean	Keeps the order hidden in the carrier's system until the first scan event is registered against it.

Price Items

Price items attach one or more pre-configured price/cost lines to an order. Each item is an object inside the `priceitems` array, which sits inside `transporterSpecificFields` — not directly on the order.

⚠ EXACT PATH AND CASING

`order.transporterSpecificFields.priceitems` — lower-case "i" in `priceitems`.

`priceItems` directly on the order is rejected. The order object does not accept unknown fields.

⚠ Recommended SEND state: "UPDATE"

Send state: "update" for price items unless instructed otherwise or you can keep track of state internally.

update acts as an upsert — it creates the price item if it doesn't exist yet, and updates it if it does (matching on customerReference + priceitemMappingCode).

state defaults to "new" when omitted.

Identification and lifecycle

Field	Type	Description
priceitemMappingCode	mappingEnum	Maps to a specific price item configured in the carrier's Opter instance. Must be agreed and pre-configured with your carrier; an unmatched code is not applied and raises a warning (see 1.1). Max 50 chars.
state	enum	new, update, or delete. Defaults to new when omitted. In practice, send update — including the first time a given price item is sent for an order (create-or-update behaviour). delete removes the line.
opterId	string	Opter's internal price item ID. Use to target an existing line on update or delete. Max 50 chars.
customerReference	string	Your reference for this price item line. Max 50 chars. Use when creating/ updating a price item so you can update values later.
customerGuid	string	Your GUID reference for this line, as an alternative to customerReference. Max 50 chars.
description	string	Free-text description of the price item. Max 1024 chars.
currencyCode	string	Currency of the amounts on this line. Must be configured in the carrier's Opter instance. Max 50 chars.
resourceCode	string	Resource / subcontractor code the resource-side amounts on this line apply to. Max 50 chars.

Quantities and pricing

Numeric amounts accept up to 18 digits with 6 decimals. Customer-side fields are what the paying customer is charged; the matching Resource fields are the cost side, what the resource or subcontractor is paid.

Field	Type	Description
quantity	number	Quantity for the line.
unit1quantity	number	Quantity for the first pricing unit.
unit1name	string	Name of the first pricing unit (e.g. "kg", "hour"). Max 50 chars.
unit2quantity	number	Quantity for the second pricing unit.
unit2name	string	Name of the second pricing unit. Max 50 chars.
unitPrice	number	Customer-side price per unit.
unitPriceResource	number	Resource-side (cost) price per unit.
fixedPrice	number	Customer-side fixed price for the line, independent of quantity.
fixedPriceResource	number	Resource-side (cost) fixed price for the line.
discountAmount	number	Discount as an absolute amount.
discountFactor	number	Discount as a factor (e.g. 0.10 for 10%).
kickbackAmount	number	Kickback as an absolute amount.
kickbackFactor	number	Kickback as a factor.

Flags

Field	Type	Description
invoice	boolean	The line should be included on the customer invoice.
credit	boolean	The line is a credit rather than a debit.
noFuelExtra	boolean	No fuel surcharge on the customer-side amounts of this line.
noFuelExtraResource	boolean	No fuel surcharge on the resource-side amounts of this line.

Subcontractor and invoicing

Field	Type	Description
contractorExpectedCost	number	Expected cost from the subcontractor for this line.
contractorInvoiceNo	string	Subcontractor's invoice number for this line. Max 50 chars.
contractorInvoiceDate	string	Date of the subcontractor's invoice (YYYY-MM-DD).
contractorInvoiceDone	boolean	Subcontractor invoicing is complete for this line.
invoiceNo	string	Invoice number associated with this price item. Max 50 chars.
billNo	string	Bill number associated with this price item. Max 50 chars.

Material site and custom slots

Field	Type	Description
materialSiteName	string	Name of the material site this line relates to (bulk / materials handling). Max 50 chars.
materialSiteCode	string	Code of the material site. Must be configured in the carrier's Opter instance. Max 50 chars.
extraId1 – extraId8	string	Carrier-defined custom text slots for the price item line. Meaning is defined per carrier — confirm before use. Max 50 chars each.

3.9 Leg Assignment — shipmentId, internalVehicleCode, resourceCode

An order in Opter is carried out in one or more legs (shipments). A leg is the part of the transport performed by one vehicle or resource — for example, a pickup leg into a terminal and a distribution leg out of it are two separate legs of the same order. `shipmentId` is Opter's internal identifier for one such leg.

You need `shipmentId` whenever an update must act on a specific leg rather than on the order as a whole. The typical case is assigning a vehicle or a subcontractor: `internalVehicleCode` or `resourceCode` without a `shipmentId` is invalid (the JSON Schema rejects it), because it would be undefined which leg the assignment belongs to. This is the main reason the field exists — in AI-assisted or otherwise automated dispatching, where an external system decides which vehicle should take which leg and writes that decision back into Opter, the leg has to be addressed explicitly.

⚠ THESE VALUES COME FROM THE OPTER API V1

`shipmentId`, `resourceCode`, and `internalVehicleCode` are NOT returned by the TextEDI API — neither in order responses nor in event messages.

They all come from the Opter API V1: one separate API, with its own documentation and credentials, that groups several endpoints. Three of those endpoints matter here:

- Statistics endpoint — leg identifiers, i.e. `shipmentId` and `deliveryId`.
- Vehicle endpoint — the values for `internalVehicleCode`.
- Subcontractor / resource endpoint — the values for `resourceCode`.

All three fields require prior agreement and configuration with your carrier — request Opter API V1 access at the same time as your TextEDI API key.

⚠ ONE LEG IS EITHER INTERNAL OR SUBCONTRACTED — NEVER BOTH

internalVehicleCode and resourceCode are two different ways of performing the same leg: the carrier's own vehicle, or a subcontractor taking the job.

Send only one of the two per message. A message carrying both is invalid and fails schema validation.

Different legs of the same order can be assigned differently — leg 1 to an internal vehicle, leg 2 to a subcontractor. Each assignment is its own update message with its own shipmentId (see 6.11).

Field	Type	Description
shipmentId	integer	Opter's internal ID of a single leg (shipment) of the order. Identifies which leg an update applies to. Fetched via the statistics endpoint of the Opter API V1.
internalVehicleCode	string	The internal vehicle — one of the carrier's own — to assign to the leg. Mutually exclusive with resourceCode. Fetched via the vehicle endpoint of the Opter API V1. Max 50 chars.
resourceCode	string	The subcontractor to assign to the leg, when the leg is bought externally rather than driven in-house. Despite the generic name, this is a subcontractor code. Mutually exclusive with internalVehicleCode. Fetched via the subcontractor/resource endpoint of the Opter API V1. Max 50 chars.

i internalVehicleCode is not vehicleMappingCode. The two look similar and are often confused, but they answer different questions: what kind of vehicle does this order need, and which vehicle actually drives this leg. They are not interchangeable, and one does not substitute for the other.

	vehicleMappingCode	internalVehicleCode
What it selects	A vehicle <i>type</i> (van, truck, trailer).	One concrete vehicle in the carrier's fleet.
Scope	The order as a whole.	A single leg, identified by shipmentId.
Where the value comes from	A mapping pre-configured with your carrier (see 1.1).	The vehicle endpoint of the Opter API V1 (see above).
When it is sent	On the booking, with messageType new.	Afterwards, on a messageType update carrying the shipmentId.
If the value doesn't match	The message is rejected, unless the carrier has a default configured.	The assignment does not land on the leg.

They are not alternatives and do not conflict: an order normally carries a vehicleMappingCode from the booking, and later gets an internalVehicleCode on one of its legs when that leg is assigned. Setting one neither sets nor overrides the other. If a leg is subcontracted instead, it gets a resourceCode rather than an internalVehicleCode — the order's vehicleMappingCode is unaffected either way.

See section 6.11 for a leg assignment example.

3.10 Loose Items

looseItems is an order-level array, structured the same way as the options array. Each entry references a carrier-configured loose item code and reports its quantity at the start and at the end of the transport — for example to keep track of load carriers handed over and handed back.

⚠ CARRIER AGREEMENT REQUIRED

Every looseItemMappingCode must be pre-configured in the carrier's Opter instance.

The meaning of each code is defined by the carrier — confirm it before populating the array.

Field	Type	Description
looseItemMappingCode	mappingEnum	Maps to a loose item configured in the carrier's Opter instance. Must be agreed and pre-configured with your carrier. Max 50 chars.
startingQuantity	number	Quantity at the start of the transport.

Field	Type	Description
endingQuantity	number	Quantity at the end of the transport.

```

"looseItems": [
  {
    "looseItemMappingCode": "string",
    "startingQuantity": 0,
    "endingQuantity": 0
  }
]

```

3.11 Goods Items

goodsItems is an order-level array of goods lines for customs declaration. Each line describes one commodity in the shipment — what it is, its commodity code, how much of it there is, what it weighs, what it is worth, and where it comes from.

Goods items do not replace packages. A package is a physical unit with a barcode and dimensions; a goods item is a customs line. Keep sending the packages as usual, and link each goods line to the package it travels in with packageId.

⚠️ AGREE THE USE WITH YOUR CARRIER

Send goodsItems when the shipment needs a customs declaration and your carrier has agreed to receive it.

Agree which package identifier goes in packageId before relying on the link between goods lines and packages.

Field	Type	Description
quantity	integer	Number of units in this goods item.
commodityCode	string	HS/CN/TARIC commodity code (e.g. "84713000"). Max 50 chars.
description	string	Plain-text description of the goods. Max 255 chars.
netWeightKgm	number	Net weight in kilograms.
grossWeightKgm	number	Gross weight in kilograms.
amount	number	Monetary value of the goods item.
currency	string	ISO 4217 currency code for the amount (e.g. SEK). Max 50 chars.
countryOfOrigin	string	ISO 3166-1 alpha-2 country of origin (e.g. CN). Max 50 chars.
packageId	string	Identifier of the package this goods item belongs to. Agree with your carrier which package identifier to use. Max 50 chars.

See section 6.17 for a cross-border booking with goods items.

4. Transport Events

Transport events report the status and progress of shipments. They use the same `POST /api/textedi` endpoint as order bookings, but are sent as a JSON **array** containing exactly one event object (not a single object).

Each event contains the common envelope fields plus exactly one payload key that identifies the event type. The `eventMappingCode` within the payload key specifies the exact event.

⚠ EVENT IDENTIFICATION RULES

`opterOrderReference` is required on every event and identifies the order. `customerOrderReference` is echoed back for your own correlation.

Package-level events (`packagescan`) **MUST** also carry the package barcode — the barcode identifies the package and is mandatory for package scans.

`freightbillscan` events **MUST** carry `freightBillNumber` in the envelope.

For updates (order message type "update"), use either `opterOrderReference` (preferred) or `customerOrderReference` to identify the order to modify.

4.1 All Event Types at a Glance

Payload key	eventMappingCode	Description
status	departure	Vehicle departed from pickup location. Contains all packages + GPS.
status	arrival	Vehicle arrived at delivery location. Contains all packages + GPS. May carry POD fields.
packagescan	departureTerminal	Package scanned departing origin terminal. Single package object.
packagescan	arrivalTerminal	Package scanned arriving at destination terminal. Single package object.
packagescan	departureDistributionTerminal	Package scanned departing distribution terminal for last mile. Single package object.
packagescan	other codes (see 4.4)	External terminals, floorcheck, and undo variants of every scan.
freightbillscan	same codes as packagescan	Freight-bill barcode scanned. Requires <code>freightBillNumber</code> .
pod	arrival	Full delivery — all packages delivered to consignee. Optional signature.
pod	partialdelivery	Partial delivery — some packages not delivered. Optional signature.
revoke	orderRevoked	Order was cancelled/removed. Lists all packages.
deviation	carrier-configured	Deviation / exception during transport (damage, delay, refusal, ...).
priceitem	carrier-configured	Price line item on an order.
attachment	cmr, freightbill, awb, pod, deviation, image, customsdocument, edi, packagelabel	File attachment, by URL or base64.

i When Opter exports status events, `eventMappingCode` is not limited to departure / arrival — it can also carry the terminal codes `departureTerminal`, `arrivalTerminal`, `departureDistributionTerminal`, or `arrivalDistributionTerminal` at the order level. Treat these the same as their `packagescan` counterparts (see 4.4), just reported against the whole order rather than a single package.

4.2 Common Event Envelope Fields

In addition to the shared envelope (`schema`, `version`, `eventCreatedDateTime`, `eventSender`, `eventReceiver`), `event` messages include:

Field	Type	Description
opterOrderReference	string	Required. Opter's internal order number. Use to correlate with your orders.
customerOrderReference	string	Your order reference, echoed back for correlation. Max 100 chars.
freightBillNumber	string	Freight bill number associated with this order (nullable). Required for freightbillscan. Max 50 chars.
estimatedTransportCost	number	Estimated transport cost, if available (nullable). Opter's own estimate; the field is also accepted on orders (see 3.2).
estimatedTransportCo2Emission	number	Estimated CO ₂ emission in grams, if available (nullable).

4.3 Status Events

Payload key: status — Reports order-level status changes with GPS coordinates and all packages.

Field	Type	Description
eventDateTime	string	When the event actually occurred (may differ from eventCreatedDateTime)
eventMappingCode	string	One of: departure, arrival, departureTerminal, arrivalTerminal, departureDistributionTerminal, arrivalDistributionTerminal (see note in 4.1)
eventDescription	string	Human-readable description. Max 50 chars.
eventLocation	string	City or location name. For terminal codes, a hub code agreed with the carrier. Max 50 chars.
eventLatitude	number	GPS latitude (nullable)
eventLongitude	number	GPS longitude (nullable)
transporterResource	string	Vehicle or driver identifier (may be empty). Max 50 chars.
packages	array	All packages in the order (opterId, customerReference, barcode)
podSigneeName, podComment, podSignatureUrl, podSignatureImageBase64encoded	string / array	POD fields — only allowed when eventMappingCode is arrival. Same meaning and types as in 4.5: the two image fields are arrays.

4.4 Package Scan Events

Payload key: packagescan — Reports an individual package scanned at a terminal. **Barcode is mandatory** — the barcode is the identifier for package-level events. Contains a single package object (not an array).

Field	Type	Description
eventDateTime	string	When the scan occurred
eventMappingCode	string	One of: departure, arrival, departureTerminal, arrivalTerminal, departureDistributionTerminal, arrivalDistributionTerminal, departureExternalTerminal, arrivalExternalTerminal, undoDeparture, undoArrival, undoDepartureTerminal, undoArrivalTerminal, undoDepartureDistributionTerminal, undoArrivalDistributionTerminal, undoDepartureExternalTerminal, undoArrivalExternalTerminal, floorcheck, undoFloorcheck. The *Terminal codes use eventLocation as a hub code agreed with the carrier; the *ExternalTerminal codes use it as free text; undo* codes reverse a previous scan.
eventDescription	string	Human-readable scan description. Max 50 chars.
eventLocation	string	Terminal / location name. Max 50 chars.
eventLatitude	number	GPS latitude (nullable)
eventLongitude	number	GPS longitude (nullable)
package	object	Required. Single object with opterId, customerReference, barcode. Barcode is mandatory — it is the identifier for package-level events.

4.5 Proof of Delivery (POD)

Payload key: pod — Reports delivery completion with optional signature data.

Field	Type	Description
eventDateTime	string	When the delivery occurred
eventMappingCode	string	arrival (full delivery) or partialdelivery. Lower case is translated to upper case on import.
eventDescription	string	Human-readable description (e.g., "received by the consignee"). Max 50 chars.
eventLatitude	number	GPS latitude (nullable)
eventLongitude	number	GPS longitude (nullable)
packages	array	Packages included in this delivery (opterId, customerReference, barcode)
podSigneeName	string	Name of person who signed (empty if not captured). Max 50 chars.
podComment	string	Free-text comment captured at delivery (empty if none). Max 50 chars.
podSignatureUrl	array	URLs to POD images — one or more. Each URL max 255 chars. Opter exports at most one URL; the images themselves are in podSignatureImageBase64encoded. Empty array or null if none.
podSignatureImageBase64encoded	array	Base64-encoded POD images — one or more. The drawn signature comes first when there is one, followed by delivery photos. Empty array or null if none. Can be combined with podSignatureUrl.

4.6 Revocation Events

Payload key: revoke — Indicates that an order has been cancelled/removed.

Field	Type	Description
eventDateTime	string	When the revocation occurred
eventMappingCode	string	Commonly orderRevoked; carrier-specific codes are also accepted.
eventDescription	string	Reason or description (e.g., "order was removed"). Max 50 chars.
packages	array	All packages from the revoked order (opterId, customerReference, barcode)

4.7 Freight Bill Scan Events

Payload key: freightbillscan — Reports a scan of the freight-bill barcode rather than an individual package. The envelope **must** carry freightBillNumber. Contains a packages array.

Field	Type	Description
eventDateTime	string	When the scan occurred
eventMappingCode	string	Same 18 codes and semantics as packagescan (see 4.4).
eventDescription	string	Human-readable scan description. Max 50 chars.
eventLocation	string	Terminal / location name. Max 50 chars.
eventLatitude	number	GPS latitude (nullable)
eventLongitude	number	GPS longitude (nullable)
packages	array	Packages associated with the freight bill (opterId, customerReference, barcode)

4.8 Deviation Events

Payload key: `deviation` — Reports a problem encountered during transport (damage, delay, missed or refused delivery, etc.). Deviation codes are configured by the carrier — agree them before use.

Field	Type	Description
<code>eventDateTime</code>	string	When the deviation occurred
<code>eventMappingCode</code>	string	Carrier-configured deviation code. Max 50 chars.
<code>eventDescription</code>	string	Human-readable description. Max 50 chars.
<code>eventLatitude</code>	number	GPS latitude (nullable)
<code>eventLongitude</code>	number	GPS longitude (nullable)
<code>packages</code>	array	Packages affected by the deviation (<code>opterId</code> , <code>customerReference</code> , <code>barcode</code>)
<code>deviationEvent</code>	string	Sub-event under <code>eventMappingCode</code> (e.g. a specific deviation type). Max 50 chars.
<code>deviationReason</code>	string	Sub-event under <code>deviationEvent</code> (e.g. a specific reason). Max 50 chars.
<code>deviationComment</code>	string	Free-text comment. Max 50 chars.
<code>deviationUrl</code>	array	URLs to deviation evidence (images, documents) — one or more. Each URL max 255 chars.
<code>deviationImageBase64encoded</code>	array	Base64-encoded deviation evidence images — one or more. Opter exports deviation images here, not in <code>deviationUrl</code> . Can be combined with <code>deviationUrl</code> .

4.9 Price Item Events

Payload key: `priceitem` — Communicates one price line on an order. The fields mirror the order-level price items in 3.8. Lower-case `eventMappingCode` is translated to upper case on import.

Field	Type	Description
<code>eventDateTime</code>	string	When the price item event was created
<code>eventMappingCode</code>	string	Carrier-configured price item code. Max 50 chars.
<code>eventDescription</code>	string	Description of the price item. Max 1024 chars.
<code>opterId</code>	integer	Opter internal price item ID.
<code>customerReference</code>	string	Your reference for this line (nullable). Max 50 chars.
<code>state</code>	enum	new or update.
<code>quantity, unit1quantity, unit2quantity</code>	number	Quantities. Up to 18 digits, 6 decimals (nullable).
<code>unit1name, unit2name</code>	string	Names of the pricing units. Max 50 chars.
<code>unitPrice, unitPriceResource, fixedPrice, fixedPriceResource</code>	number	Customer-side and resource-side prices, as in 3.8 (nullable).
<code>resourceCode</code>	string	Resource / subcontractor code. Max 50 chars.
<code>contractorExpectedCost</code>	number	Expected cost from the subcontractor (nullable).
<code>contractorInvoiceNo</code>	string	Subcontractor's invoice number. Max 50 chars.
<code>contractorInvoiceDate</code>	string	Date of the subcontractor's invoice (YYYY-MM-DD).
<code>contractorInvoiceDone</code>	boolean	Subcontractor invoicing is complete for this line.
<code>extraId1 – extraId8</code>	string	Carrier-defined custom text slots. Max 50 chars each.

4.10 Attachment Events

Payload key: attachment — Attaches a file to an order. Provide either attachmentUrl or attachmentBase64encoded.

Field	Type	Description
eventDateTime	string	When the attachment was created
eventMappingCode	enum	Type of attachment: cmr, freightbill, awb, pod, deviation, image, customsdocument, edi, or packagelabel.
eventDescription	string	Description of the attachment. Max 255 chars.
attachmentComment	string	Comment. Max 255 chars.
attachmentFileName	string	File name with extension. Max 50 chars.
attachmentUrl	string	URL to download the attachment. Max 255 chars.
attachmentBase64encoded	string	Base64-encoded file content.

5. Response Structure

The API always responds with a JSON array containing one element. On successful order import, the array contains a response object. On validation failure, it contains an error string. Responses use the same envelope and the same schema as events (opterEventJsonSchema1.0.json).

5.1 Successful Response

A successful response includes:

Field	Description
opterOrderReference	Store this value. Opter's internal order number for future update/merge/revoke operations.
response.eventMappingCode	Result code: imported, accepted, failed, or rejected.
response.packages	Packages with Opter-assigned IDs (opterId, customerReference, barcode)
response.errors	Error array. Non-empty means the order was NOT processed.
response.warnings	Warning array. Order processed but with caveats — e.g. an unmatched mapping code that was not applied.
response.freightBillBase64encoded	Base64-encoded freight bill PDF (empty if not generated)
response.packageLabelBase64encoded	Base64-encoded package label (empty if not generated)

i The response does not contain shipmentId, resourceCode, or internalVehicleCode. Those values are read from the Opter API V1 — see 3.9.

5.2 Error Responses

Error strings returned in the response array. The first three are the only mapping-related failures that reject a message — any other unmatched mapping code produces a warning instead and the order is still processed (see 1.1). NOT IMPLEMENTED means the requested feature is not available.

Error	Cause
EXCEPTION_Service type incorrect (and no default)	serviceMappingCode does not match any configured service
EXCEPTION_Vehicle type incorrect (and no default)	vehicleMappingCode does not match any configured vehicle
EXCEPTION_Incorrect customer code	payingCustomerCode does not match any customer in Opter
NOT IMPLEMENTED	The requested feature is not available

6. Examples

6.1 Simple Order Booking (package-level dimensions)

```
{
  "schema": "https://download.opter.com/json/opterOrderBookingJsonSchema1.0.json",
  "version": "1.0",
  "eventCreatedDateTime": "2025-02-13T14:00:00",
  "eventSender": "CustomerSystem",
  "eventReceiver": "OpterSystem",
  "order": {
    "messageType": "new",
    "customerOrderReference": "CUST-2025-001",
    "payingCustomerCode": "CUSTOMER123",
    "serviceMappingCode": "STANDARD",
    "vehicleMappingCode": "VAN",
    "transportDate": "2025-02-14",
    "driverInstructions": "Call 15 minutes before arrival",
    "addresses": [
      {
        "addressType": "pickup",
        "earliestTransportDateTime": "2025-02-14T08:00:00",
        "latestTransportDateTime": "2025-02-14T12:00:00",
        "name": "Warehouse AB",
        "streetAddress": "Industrivägen 10",
        "zipCode": "12345",
        "city": "Stockholm",
        "countryCode": "SE",
        "phoneNumber": "+46701234567"
      },
      {
        "addressType": "delivery",
        "earliestTransportDateTime": "2025-02-14T13:00:00",
        "latestTransportDateTime": "2025-02-14T17:00:00",
        "name": "Customer Corp",
        "streetAddress": "Kungsgatan 25",
        "zipCode": "54321",
        "city": "Gothenburg",
        "countryCode": "SE",
        "phoneNumber": "+46709876543"
      }
    ],
    "packages": [
      {
        "customerReference": "PKG-001",
        "barcode": "1000000001",
        "packageTypeCode": "BOX",
        "dimensions": { "quantity": 1, "weightKgm": 25.0, "volumeCbm": 0.125 }
      }
    ]
  }
}
```

6.2 Status Event — Departure

```
[
  {
    "schema": "https://download.opter.com/json/opterEventJsonSchema1.0.json",
    "version": "1.0",
    "eventCreatedDateTime": "2025-04-11T12:05:15",
    "eventSender": "ReceiverID",
    "eventReceiver": "SenderID",
    "opterOrderReference": "9595",
    "customerOrderReference": "CustomerOrderRef123",
    "freightBillNumber": "FreightBill123",
    "estimatedTransportCost": null,
    "estimatedTransportCo2Emission": null,
    "status": {
      "eventDateTime": "2025-04-11T12:05:05",
      "eventMappingCode": "departure",
      "eventDescription": "departure",
      "eventLocation": "cityname",
      "eventLatitude": 55.683895,
      "eventLongitude": 12.574008,
      "transporterResource": "",
      "packages": [
        {
          "opterId": 9943,
          "customerReference": "PackageRef1",
          "barcode": "Barcode1"
        },
        {
          "opterId": 9944,
          "customerReference": "PackageRef2",
          "barcode": "Barcode2"
        }
      ]
    }
  }
]
```

6.3 Status Event — Arrival

```
[
  {
    "schema": "https://download.opter.com/json/opterEventJsonSchema1.0.json",
    "version": "1.0",
    "eventCreatedDateTime": "2025-04-11T12:05:55",
    "eventSender": "ReceiverID",
    "eventReceiver": "SenderID",
    "opterOrderReference": "9595",
    "customerOrderReference": "CustomerOrderRef123",
    "freightBillNumber": "FreightBill123",
    "estimatedTransportCost": null,
    "estimatedTransportCo2Emission": null,
    "status": {
      "eventDateTime": "2025-04-11T12:05:53",
      "eventMappingCode": "arrival",
      "eventDescription": "arrival",
      "eventLocation": "cityname2",
      "eventLatitude": 65.079422,
      "eventLongitude": 13.316355,
      "transporterResource": "",
      "packages": [
        {
          "opterId": 9943,
          "customerReference": "PackageRef1",
          "barcode": "Barcode1"
        },
        {
          "opterId": 9944,
          "customerReference": "PackageRef2",
          "barcode": "Barcode2"
        }
      ]
    }
  }
]
```

6.4 Package Scan Event — Departure Terminal

```
[
  {
    "schema": "https://download.opter.com/json/opterEventJsonSchema1.0.json",
    "version": "1.0",
    "eventCreatedDateTime": "2025-04-11T12:05:35",
    "eventSender": "ReceiverID",
    "eventReceiver": "SenderID",
    "opterOrderReference": "9595",
    "customerOrderReference": "CustomerOrderRef123",
    "freightBillNumber": "FreightBill123",
    "estimatedTransportCost": null,
    "estimatedTransportCo2Emission": null,
    "packagescan": {
      "eventDateTime": "2025-04-11T12:05:33",
      "eventMappingCode": "departureTerminal",
      "eventDescription": "departure terminal",
      "eventLocation": "malmö",
      "eventLatitude": 55.626362,
      "eventLongitude": 13.0362,
      "package": {
        "opterId": 9943,
        "customerReference": "PackageRef1",
        "barcode": "Barcode1"
      }
    }
  }
]
```

6.5 Package Scan Event — Arrival Terminal

```
[
  {
    "schema": "https://download.opter.com/json/opterEventJsonSchema1.0.json",
    "version": "1.0",
    "eventCreatedDateTime": "2025-04-11T12:05:45",
    "eventSender": "ReceiverID",
    "eventReceiver": "SenderID",
    "opterOrderReference": "9595",
    "customerOrderReference": "CustomerOrderRef123",
    "freightBillNumber": "FreightBill123",
    "estimatedTransportCost": null,
    "estimatedTransportCo2Emission": null,
    "packagescan": {
      "eventDateTime": "2025-04-11T12:05:41",
      "eventMappingCode": "arrivalTerminal",
      "eventDescription": "arrival terminal",
      "eventLocation": "årsta",
      "eventLatitude": 59.29369,
      "eventLongitude": 18.03087,
      "package": {
        "opterId": 9943,
        "customerReference": "PackageRef1",
        "barcode": "Barcode1"
      }
    }
  }
]
```

6.6 Package Scan Event — Departure Distribution Terminal

```
[
  {
    "schema": "https://download.opter.com/json/opterEventJsonSchema1.0.json",
    "version": "1.0",
    "eventCreatedDateTime": "2025-04-11T12:05:55",
    "eventSender": "ReceiverID",
    "eventReceiver": "SenderID",
    "opterOrderReference": "9595",
    "customerOrderReference": "CustomerOrderRef123",
    "freightBillNumber": "FreightBill123",
    "estimatedTransportCost": null,
    "estimatedTransportCo2Emission": null,
    "packagescan": {
      "eventDateTime": "2025-04-11T12:05:47",
      "eventMappingCode": "departureDistributionTerminal",
      "eventDescription": "departure distribution terminal",
      "eventLocation": "Årsta",
      "eventLatitude": 59.29369,
      "eventLongitude": 18.03087,
      "package": {
        "opterId": 9943,
        "customerReference": "PackageRef1",
        "barcode": "Barcode1"
      }
    }
  }
]
```

6.7 Proof of Delivery — Full Delivery

The images array carries the drawn signature first, then delivery photos. Base64 values are shortened here.

```
[
  {
    "schema": "https://download.opter.com/json/opterEventJsonSchema1.0.json",
    "version": "1.0",
    "eventCreatedDateTime": "2025-04-09T14:33:21",
    "eventSender": "Transporter company name",
    "eventReceiver": "Transporter's customers name",
    "opterOrderReference": "101337",
    "customerOrderReference": "CustomerOrderRef123",
    "freightBillNumber": "FreightBill123",
    "estimatedTransportCost": null,
    "estimatedTransportCo2Emission": null,
    "pod": {
      "eventDateTime": "2025-04-09T14:33:12",
      "eventMappingCode": "arrival",
      "eventDescription": "received by the consignee",
      "packages": [
        { "opterId": 55788, "customerReference": "21548", "barcode": "" }
      ],
      "podSigneeName": "Test Persson",
      "podComment": "",
      "podSignatureUrl": [
        "https://example.com/pod/101337.png"
      ],
      "podSignatureImageBase64encoded": [
        "iVBORw0KGgoAAAANSUhEUgAA...",
        "/9j/4AAQSkZJRgABAQ..."
      ]
    }
  }
]
```

6.8 Proof of Delivery — Partial Delivery

```
[
  {
    "schema": "https://download.opter.com/json/opterEventJsonSchema1.0.json",
    "version": "1.0",
    "eventCreatedDateTime": "2025-04-09T14:33:21",
    "eventSender": "Transporter company name",
    "eventReceiver": "Transporter's customers name",
    "opterOrderReference": "101337",
    "customerOrderReference": "CustomerOrderRef123",
    "freightBillNumber": "FreightBill123",
    "estimatedTransportCost": null,
    "estimatedTransportCo2Emission": null,
    "pod": {
      "eventDateTime": "2025-04-09T14:33:12",
      "eventMappingCode": "partialdelivery",
      "eventDescription": "",
      "eventLatitude": null,
      "eventLongitude": null,
      "packages": [
        { "opterId": 55788, "customerReference": "21548", "barcode": "" }
      ],
      "podSigneeName": "",
      "podComment": "",
      "podSignatureUrl": [],
      "podSignatureImageBase64encoded": []
    }
  }
]
```

partialdelivery is used when some packages could not be delivered. eventDescription, eventLatitude, eventLongitude, and podSigneeName may be empty or null when that information wasn't captured for the partial delivery.

6.9 Order Revocation

```
[
  {
    "schema": "https://download.opter.com/json/opterEventJsonSchema1.0.json",
    "version": "1.0",
    "eventCreatedDateTime": "2025-04-11T10:38:21",
    "eventSender": "ReceiverID",
    "eventReceiver": "SenderID",
    "opterOrderReference": "9592",
    "customerOrderReference": "CustomerOrderRef123",
    "freightBillNumber": "FreightBill123",
    "estimatedTransportCost": null,
    "estimatedTransportCo2Emission": null,
    "revoke": {
      "eventDateTime": "2025-04-11T10:38:21",
      "eventMappingCode": "orderRevoked",
      "eventDescription": "order was removed",
      "packages": [
        {
          "opterId": 9935,
          "customerReference": "PackageRef1",
          "barcode": "Barcode1"
        },
        {
          "opterId": 9936,
          "customerReference": "PackageRef2",
          "barcode": "Barcode2"
        }
      ]
    }
  }
]
```

6.10 Price Request

Uses messageType: "pricerequest" on the order object — requires prior carrier agreement (see 3.1). This is a quote-only request; no order is created in Opter. The response uses the same structure as a normal order booking response (see 5.1), but any references it returns are for the quote only and should not be reused for update, revoke, or status matching — confirm the exact behavior with your carrier.

```
{
  "schema": "https://download.opter.com/json/opterOrderBookingJsonSchema1.0.json",
  "version": "1.0",
  "eventCreatedDateTime": "2025-02-13T09:00:00",
  "eventSender": "CustomerSystem",
  "eventReceiver": "OpterSystem",
  "order": {
    "messageType": "pricerequest",
    "customerOrderReference": "CUST-2025-QUOTE-001",
    "payingCustomerCode": "CUSTOMER123",
    "serviceMappingCode": "STANDARD",
    "vehicleMappingCode": "VAN",
    "transportDate": "2025-02-14",
    "addresses": [
      {
        "addressType": "pickup",
        "zipCode": "12345",
        "city": "Stockholm",
        "countryCode": "SE"
      },
      {
        "addressType": "delivery",
        "zipCode": "54321",
        "city": "Gothenburg",
        "countryCode": "SE"
      }
    ],
    "packages": [
      {
        "customerReference": "PKG-001",
        "dimensions": { "quantity": 1, "weightKgm": 25.0, "volumeCbm": 0.125 }
      }
    ],
    "transporterSpecificFields": {
      "priceitems": [
        {
          "priceitemMappingCode": "FUEL_SURCHARGE",
          "customerReference": "PI-001",
          "state": "update",
          "description": "Fuel surcharge",
          "currencyCode": "SEK",
          "quantity": 1,
          "unitPrice": 45.50,
          "unitPriceResource": 38.00,
          "invoice": true
        }
      ]
    }
  }
}
```

6.11 Update — Assign a Vehicle or a Subcontractor to a Leg

Each message identifies the order with `opterOrderReference` and one leg with `shipmentId`, then assigns that leg to either an internal vehicle or a subcontractor — never both (see 3.9). The two messages below target the same order: leg 40122 is driven in-house, leg 40123 is bought from a subcontractor. All values are looked up beforehand on the Opter API V1 (see 3.9). Requires carrier agreement.

Leg 40122 — internal vehicle

```
{
  "schema": "https://download.opter.com/json/opterOrderBookingJsonSchema1.0.json",
  "version": "1.0",
  "eventCreatedDateTime": "2026-02-13T14:20:00",
  "eventSender": "CustomerSystem",
  "eventReceiver": "OpterSystem",
  "order": {
    "messageType": "update",
    "opterOrderReference": "9595",
    "customerOrderReference": "CUST-2025-001",
    "shipmentId": 40122,
    "internalVehicleCode": "VEH-114"
  }
}
```

Leg 40123 — subcontractor

```
{
  "schema": "https://download.opter.com/json/opterOrderBookingJsonSchema1.0.json",
  "version": "1.0",
  "eventCreatedDateTime": "2026-02-13T14:21:00",
  "eventSender": "CustomerSystem",
  "eventReceiver": "OpterSystem",
  "order": {
    "messageType": "update",
    "opterOrderReference": "9595",
    "customerOrderReference": "CUST-2025-001",
    "shipmentId": 40123,
    "resourceCode": "SUB-021"
  }
}
```

6.12 Order Booking with Loose Items

Loose items are reported at order level, alongside options. Each entry needs a pre-configured looseItemMappingCode (see 3.10).

```
{
  "schema": "https://download.opter.com/json/opterOrderBookingJsonSchema1.0.json",
  "version": "1.0",
  "eventCreatedDateTime": "2026-02-13T14:00:00",
  "eventSender": "CustomerSystem",
  "eventReceiver": "OpterSystem",
  "order": {
    "messageType": "new",
    "customerOrderReference": "CUST-2026-014",
    "payingCustomerCode": "CUSTOMER123",
    "serviceMappingCode": "STANDARD",
    "vehicleMappingCode": "VAN",
    "transportDate": "2026-02-14",
    "looseItems": [
      {
        "looseItemMappingCode": "EURPALLET",
        "startingQuantity": 12,
        "endingQuantity": 10
      },
      {
        "looseItemMappingCode": "ROLLCAGE",
        "startingQuantity": 4,
        "endingQuantity": 4
      }
    ],
    "addresses": [
      {
        "addressType": "pickup",
        "name": "Warehouse AB",
        "streetAddress": "Industrivägen 10",
        "zipCode": "12345",
        "city": "Stockholm",
        "countryCode": "SE"
      },
      {
        "addressType": "delivery",
        "name": "Customer Corp",
        "streetAddress": "Kungsgatan 25",
        "zipCode": "54321",
        "city": "Gothenburg",
        "countryCode": "SE"
      }
    ],
    "packages": [
      {
        "customerReference": "PKG-001",
        "barcode": "1000000014",
        "packageTypeCode": "EUR",
        "dimensions": { "quantity": 1, "weightKgm": 320.0, "europalletSpace": 1 }
      }
    ]
  }
}
```

6.13 Freight Bill Scan Event

The envelope carries `freightBillNumber`, which is required for this payload key (see 4.7).

```
[
  {
    "schema": "https://download.opter.com/json/opterEventJsonSchema1.0.json",
    "version": "1.0",
    "eventCreatedDateTime": "2025-04-11T12:06:10",
    "eventSender": "ReceiverID",
    "eventReceiver": "SenderID",
    "opterOrderReference": "9595",
    "customerOrderReference": "CustomerOrderRef123",
    "freightBillNumber": "FreightBill123",
    "estimatedTransportCost": null,
    "estimatedTransportCo2Emission": null,
    "freightbillscan": {
      "eventDateTime": "2025-04-11T12:06:05",
      "eventMappingCode": "arrivalTerminal",
      "eventDescription": "freight bill arrived at terminal",
      "eventLocation": "malmö",
      "eventLatitude": 55.626362,
      "eventLongitude": 13.0362,
      "packages": [
        {
          "opterId": 9943,
          "customerReference": "PackageRef1",
          "barcode": "Barcode1"
        },
        {
          "opterId": 9944,
          "customerReference": "PackageRef2",
          "barcode": "Barcode2"
        }
      ]
    }
  }
]
```

6.14 Deviation Event

Deviation codes and sub-codes are carrier-configured (see 4.8). Base64 values are shortened here.

```
[
  {
    "schema": "https://download.opter.com/json/opterEventJsonSchema1.0.json",
    "version": "1.0",
    "eventCreatedDateTime": "2025-04-11T13:00:00",
    "eventSender": "ReceiverID",
    "eventReceiver": "SenderID",
    "opterOrderReference": "9595",
    "customerOrderReference": "CustomerOrderRef123",
    "freightBillNumber": "FreightBill123",
    "estimatedTransportCost": null,
    "estimatedTransportCo2Emission": null,
    "deviation": {
      "eventDateTime": "2025-04-11T12:58:00",
      "eventMappingCode": "damageOnArrival",
      "eventDescription": "package damaged on arrival",
      "eventLatitude": 59.29369,
      "eventLongitude": 18.03087,
      "packages": [
        {
          "opterId": 9943,
          "customerReference": "PackageRef1",
          "barcode": "Barcode1"
        }
      ],
      "deviationEvent": "corner_crushed",
      "deviationReason": "handling",
      "deviationComment": "Visible crush damage on top-left corner",
      "deviationImageBase64encoded": [
        "/9j/4AAQSkZJRgABAQ...",
        "/9j/4AAQSkZJRgABAg..."
      ]
    }
  }
]
```

6.15 Price Item Event

```
[
  {
    "schema": "https://download.opter.com/json/opterEventJsonSchema1.0.json",
    "version": "1.0",
    "eventCreatedDateTime": "2025-04-11T13:30:00",
    "eventSender": "ReceiverID",
    "eventReceiver": "SenderID",
    "opterOrderReference": "9595",
    "customerOrderReference": "CustomerOrderRef123",
    "freightBillNumber": "FreightBill123",
    "estimatedTransportCost": 312.5,
    "estimatedTransportCo2Emission": 1842,
    "priceitem": {
      "eventDateTime": "2025-04-11T13:29:55",
      "eventMappingCode": "fuelSurcharge",
      "eventDescription": "Fuel surcharge applied for April 2025",
      "opterId": 88123,
      "customerReference": "FuelSurchargeQ2",
      "state": "new",
      "quantity": 1,
      "unitQuantity": 1,
      "unitName": "ea",
      "unitPrice": 42.5,
      "fixedPrice": 42.5,
      "resourceCode": "FUEL",
      "contractorInvoiceNo": "INV-2025-0411-001",
      "contractorInvoiceDate": "2025-04-11",
      "contractorInvoiceDone": false,
      "extraId1": "Q2",
      "extraId2": "2025"
    }
  }
]
```

6.16 Attachment Event

```
[
  {
    "schema": "https://download.opter.com/json/opterEventJsonSchema1.0.json",
    "version": "1.0",
    "eventCreatedDateTime": "2025-04-11T14:00:00",
    "eventSender": "ReceiverID",
    "eventReceiver": "SenderID",
    "opterOrderReference": "9595",
    "customerOrderReference": "CustomerOrderRef123",
    "freightBillNumber": "FreightBill123",
    "estimatedTransportCost": null,
    "estimatedTransportCo2Emission": null,
    "attachment": {
      "eventDateTime": "2025-04-11T13:59:50",
      "eventMappingCode": "cmr",
      "eventDescription": "CMR document for order 9595",
      "attachmentComment": "Signed CMR",
      "attachmentFileName": "cmr_9595.pdf",
      "attachmentUrl": "https://example.com/docs/cmr_9595.pdf",
      "attachmentBase64encoded": ""
    }
  }
]
```

6.17 Order Booking with Goods Items

A Sweden → Germany booking carrying customs goods lines (see 3.11). The packages still carry the physical dimensions and barcodes; each goods line points at its package through `packageId`. German zip codes are exactly 5 characters.

```
{
  "schema": "https://download.opter.com/json/opterOrderBookingJsonSchema1.0.json",
  "version": "1.0",
  "eventCreatedDateTime": "2026-10-20T09:00:00",
  "eventSender": "CustomerSystem",
  "eventReceiver": "OpterSystem",
  "order": {
    "messageType": "new",
    "customerOrderReference": "CUST-2026-207",
    "payingCustomerCode": "CUSTOMER123",
    "serviceMappingCode": "STANDARD",
    "vehicleMappingCode": "TRUCK",
    "transportDate": "2026-10-21",
    "addresses": [
      {
        "addressType": "pickup",
        "name": "Warehouse AB",
        "streetAddress": "Industrivägen 10",
        "zipCode": "12345",
        "city": "Stockholm",
        "countryCode": "SE",
        "phoneNumber": "+46701234567"
      },
      {
        "addressType": "delivery",
        "name": "Kunde GmbH",
        "streetAddress": "Hafenstraße 4",
        "zipCode": "20095",
        "city": "Hamburg",
        "countryCode": "DE",
        "phoneNumber": "+49401234567"
      }
    ],
    "packages": [
      {
        "customerReference": "PKG-207-1",
        "barcode": "1000000207",
        "packageTypeCode": "EUR",
        "dimensions": { "quantity": 1, "weightKgm": 96.5, "europalletSpace": 1 }
      },
      {
        "customerReference": "PKG-207-2",
        "barcode": "1000000208",
        "packageTypeCode": "EUR",
        "dimensions": { "quantity": 1, "weightKgm": 41.0, "europalletSpace": 1 }
      }
    ],
    "goodsItems": [
      {
        "quantity": 40,
        "commodityCode": "84713000",
        "description": "Portable computers",
        "netWeightKgm": 88.0,
        "grossWeightKgm": 96.5,
        "amount": 182000,
        "currency": "SEK",
        "countryOfOrigin": "CN",
        "packageId": "1000000207"
      },
      {
        "quantity": 120,
        "commodityCode": "85044090",
        "description": "Power adapters",
        "netWeightKgm": 36.0,
        "grossWeightKgm": 41.0,
        "amount": 21600,
        "currency": "SEK",
        "countryOfOrigin": "CN",
        "packageId": "1000000208"
      }
    ]
  }
}
```

7. Best Practices

7.1 Integration Rules

Topic	Guidance
Dimensions	Always prefer package-level dimensions. Use orderDimensions only when package details are unavailable. Never use both.
Addresses	Provide exactly two addresses — one pickup and one delivery. No multi-stop routes.
Updates	Use either opterOrderReference or customerOrderReference to identify orders on update, merge, and revoke. Store opterOrderReference from the creation response.
Carrier config	Do not use transporter-specific fields, price items, loose items, leg assignment fields, merge, or pricerequest without prior carrier agreement.
Pricing	Do not populate estimatedTransportCost unless your carrier instructs you to — the order object accepts it, but pricing is calculated by the carrier. Carriers calculate pricing automatically unless fixed pricing is agreed.
Event correlation	opterOrderReference is present on every event — use it to match events to orders; customerOrderReference is echoed for convenience. For package-level events (packagescan), the barcode is mandatory and serves as the package identifier.
Leg assignment	Always send shipmentId together with internalVehicleCode or resourceCode — without it the message fails schema validation and the assignment is not tied to a leg. Send only one of the two per leg: a leg is either driven internally or subcontracted, never both. Fetch all three values from the Opter API V1 — shipmentId on its statistics endpoint, the codes on its vehicle and subcontractor endpoints; none of them are returned by this API.
Loose items	Send startingQuantity and endingQuantity for each looseItemMappingCode. Confirm the meaning of every code with your carrier before use.
Goods items	Send goodsItems for shipments that need a customs declaration, alongside the packages — not instead of them. Agree with your carrier which package identifier goes in packageId.

7.2 Data Quality

Topic	Guidance
Addresses	Provide full street addresses with building numbers.
Phone numbers	Include phone numbers for both pickup and delivery to enable driver coordination.
Dimensions	Provide accurate weights and dimensions at the package level whenever possible.
Time windows	Set realistic pickup and delivery windows that account for business hours and travel time.
Country codes	Use ISO 3166-1 alpha-2 codes (SE, NO, DK, FI, etc.) — always 2 uppercase letters.
Zip codes	DE, EE, FI, FR, and SE zip codes are exactly 5 characters; DK and NO exactly 4. No spaces.

7.3 Error Prevention

Topic	Guidance
Validate first	Validate every message against the published JSON Schemas (see 8) before sending. They enforce the rules in this guide — address types, zip lengths, dimensions, leg assignment, and event structure — which the OpenAPI spec cannot express.
Store references	Always save opterOrderReference returned when creating orders.
Verify codes	Confirm serviceMappingCode, vehicleMappingCode, and payingCustomerCode are configured with your carrier.
Refresh lookups	Re-read leg IDs, vehicle codes, and resource codes from the Opter API V1 rather than caching them indefinitely — legs and fleet data change.
Handle errors	Implement proper error handling for all response types. A 200 status does not always mean success.

Topic	Guidance
Read warnings	Only serviceMappingCode, vehicleMappingCode, and payingCustomerCode reject a message. Any other unmatched mapping code is reported in response.warnings and not applied — log and act on warnings or you will lose data without noticing.
Check arrays	Event payloads are JSON arrays with exactly one object; order payloads are objects. Don't mix them up.

8. Support

For API access, integration setup, and carrier-specific configuration:

Contact	Details
API access	Contact your carrier, who will coordinate with the Opter EDI team.
Configuration	Discuss service codes, vehicle codes, customer codes, and available features with your carrier before starting integration.
Opter API V1	Required for leg identifiers (statistics endpoint), vehicles, and subcontractors/resources — see 3.9. A separate API from TextEDI, with its own documentation and credentials — request access through your carrier.
JSON Schemas	The source of truth for message structure, at https://download.opter.com/json/ . Two schemas: <code>opterOrderBookingJsonSchema1.0.json</code> for order bookings (JSON object), and <code>opterEventJsonSchema1.0.json</code> for every JSON-array message — all events and Opter's responses, including the plain-string errors. <code>opterEventResponseSchema1.0.json</code> is the same file under its earlier name, and <code>opterOrderBookingjsonSchema1.0.json</code> (lower-case j) is the same file as the booking schema. Validate against them: they enforce rules the OpenAPI spec cannot express (addresses, zip lengths, dimensions, leg assignment, event payload).
OpenAPI spec	An OpenAPI 3.0.3 YAML specification (<code>OpterTextEdi.yaml</code>) and a Postman collection are available at the same address for importing into Postman, Swagger UI, or other API tools.

This guide reflects the actual API constraints and requirements. Always verify carrier-specific configurations before using advanced features.