

Creating Receipts Through the VRP API

Oct 8, 2026 · @Woody Halevy

Overview

One API call records a complete sale in VRP: you send the receipt as a JSON document and VRP returns the receipt number it saved.

A single call can create:

- **The receipt header** (store, date, register, cashier, transaction type) — always created.
- **Item lines** — one per UPC sold.
- **Payments** — one per tender (cash, card, gift card, etc.).
- **Tax collected** — per line and tax authority.
- **The customer** — matched to an existing VRP customer or created new, plus a **shipping address** for the order.

The whole receipt is saved in one step: if anything is wrong, nothing is saved and you get an error message explaining why.

Endpoint and authentication

Send an HTTP **GET** to `/CreateReceiptFromJson` with the receipt JSON in the `Json` query parameter, URL-encoded.

```
GET http://<server>:<port>/CreateReceiptFromJson?Json=<URL-encoded JSON>
Authorization: Basic <base64 of username:password>
```

Item	Value
Server and port	Provided by your VRP contact (the VRP API default port is 8080)
Protocol	<code>http</code> , or <code>https</code> if your VRP API is set up with SSL
Method	<code>GET</code>
Authentication	HTTP Basic, with the API username and password provided by your VRP contact

Item	Value
Query parameter	<code>Json</code> = the receipt document described below, URL-encoded

Two encoding rules matter:

1. **URL-encode the whole JSON** (for example with `encodeURIComponent` or `Uri.EscapeDataString`). Otherwise characters such as `+`, `&` and `#` are lost or cut the request short.
2. **Write apostrophes as `'` inside JSON text**, for example `"LastName": "O'Brien"`. VRP saves it as a normal apostrophe; a plain `'` makes the call fail.

Keep each request under about 8,000 characters after encoding, which covers a typical receipt with a customer and 30–40 lines. For larger receipts, ask your VRP contact about the body-based `/SQL/v2` endpoint.

Request body

The JSON has receipt fields at the top level, plus optional sections for lines, payments, tax and the customer. Only `StoreID` and `TRSDate` are required, and every receipt needs at least one item line or one payment.

```
{
  "StoreID": "01",
  "TRSDate": "2026-10-08T14:30:00",
  "Cashregister": "WEB",
  "RPTrsd": [ ... item lines ... ],
  "rpPay": [ ... payments ... ],
  "TaxCollectedDetails": [ ... optional tax detail ... ],
  "RPCust": { ... optional customer ... },
  "ShippingAddresses": { ... optional ship-to address ... },
  "ShippingReceipt": { ... optional shipping details ... }
}
```

Field names are not case-sensitive (`storeid` works the same as `StoreID`). Numbers can be sent as JSON numbers or as strings. Dates use the format `YYYY-MM-DD` or `YYYY-MM-DDTHH:MM:SS`.

Receipt fields

Field	Required	Type	If missing	Notes
StoreID	Yes	text, max 5	—	VRP store code
TRSDate	Yes	date or date-time	—	Sale date; its time is used as the sale time when TrsTime is missing
ReceiptNo	No	whole number	VRP assigns the next number	Must not already exist for the store
rpTransaction	No	text, max 2	S (sale)	CO (customer order) and LA (layaway) require a customer
TrsTime	No	time	Time from TRSDate , else the current time	e.g. 14:30:00
Cashregister	No	text, max 5	blank	Register code; also used when VRP assigns the receipt number
Cashier	No	text, max 5	blank	VRP user code
CustomerID	No	text, max 20	blank	An existing VRP customer; see Customer and shipping
TaxAmt	No	amount	Total of the tax collected	Total tax on the receipt
TRSNote	No	text	blank	Free-text note, such as your order number
GovReceiptNo	No	whole number	0	Your own order or reference number
ParentRecNo	No	whole number	0	Original receipt, for returns

Item lines, payments and tax

Item lines — `RPTrsd` (array)

One object per item sold. A UPC that is an alternate or child barcode is saved under its main VRP UPC automatically.

Field	Required	Type	If missing	Notes
<code>UPC</code>	Yes	text, max 20	—	Item barcode or SKU
<code>Quantity</code>	No	number	1	Use a negative quantity for a return
<code>Amount</code>	No	amount	0	Selling price
<code>Discount</code>	No	amount	0	Discount on the line
<code>LineTax</code>	No	amount	0	Tax charged on the line
<code>TaxRule</code>	No	whole number	0	VRP tax rule code
<code>Cost</code>	No	amount	Item's average cost in VRP	Leave out unless you track cost yourself
<code>Lineitem</code>	No	whole number	Position in the array (1, 2, 3...)	Must be unique within the receipt
<code>Usercode1</code> , <code>Usercode2</code>	No	text, max 5	blank	Salesperson codes
<code>CouponType</code>	No	text, max 100	blank	

Payments — `rpPay` (array)

One object per tender.

Field	Required	Type	If missing	Notes
<code>PayType</code>	Yes	text, max 2	—	VRP payment type code; your VRP contact provides the list

Field	Required	Type	If missing	Notes
<code>Amount</code>	No	amount	0	Amount paid with this tender
<code>PaymentNumber</code>	No	whole number	Position in the array	Must be unique per pay type
<code>Rate</code>	No	number	1	Currency exchange rate

Tax collected — `TaxCollectedDetails` (array, optional)

Usually you can leave this section out. Without it, VRP records tax from each line's `LineTax`, or from `TaxAmt` when no line has tax. Send it only to split tax by authority (for example state and city).

Field	Required	Type	If missing	Notes
<code>taxCollected</code>	Yes	amount	—	Rows with 0 are skipped
<code>lineitem</code>	No	whole number	Position in the array	The item line the tax belongs to
<code>Authority</code>	No	text, max 20	<code>SalesTax</code>	Tax authority name
<code>SoldAmount</code>	No	amount	The line's <code>Amount</code>	Taxable amount

Customer and shipping

All three sections are optional single objects (`{ }`, not arrays). Without any of them and without a `CustomerID`, the receipt is saved with no customer.

How the customer is chosen

1. `RPCust.CustomerID`, or the receipt's `CustomerID` — that customer is used. If the ID is new, VRP creates the customer with that ID.
2. No ID, but `RPCust.EmailAddress` — matched to an existing customer with the same email.
3. No email match — matched by phone (`PhoneNumber`, else `Telephone2`). Only the digits are compared, so `+1 (212) 555-1212` matches `212-555-1212`.

4. No match at all — VRP creates a new customer and assigns the ID.

The ID used is returned in the response. When several customers match, the most recently updated one is used.

Customer — `RPCust` (object)

Any VRP customer field can be sent; these are the common ones. For an **existing** customer, only the fields you send are updated. For a **new** customer, fields you leave out are saved blank.

Field	Type	Notes
<code>CustomerID</code>	text, max 20	Leave out to match by email or phone
<code>FirstName</code> , <code>LastName</code> , <code>CompanyName</code>	text	
<code>EmailAddress</code>	text	Used for matching
<code>PhoneNumber</code> , <code>Telephone2</code>	text	Used for matching
<code>Address1</code> , <code>Address2</code> , <code>City</code> , <code>StateOrProvince</code> , <code>PostalCode</code> , <code>Country</code>	text	Billing address
<code>Birthday</code>	date	
<code>Notes</code>	text	

Ship-to address — `ShippingAddresses` (object)

This needs a customer: a `CustomerID` or an `RPCust` section. If the customer already has the same address (same name, `Address1`, `Address2`, `City` and `Zip`), it is reused; otherwise a new address is added to the customer.

Field	Type
<code>FirstName</code> , <code>LastName</code> , <code>Company</code>	text
<code>Address1</code> , <code>Address2</code> , <code>City</code> , <code>State</code> , <code>Zip</code> , <code>Country</code>	text
<code>Phone1</code> , <code>Phone2</code> , <code>Mobile</code> , <code>eMail</code>	text

Field	Type
Note	text

Shipping details — ShippingReceipt (object)

Field	Type	If missing
ShippingServiceCode	whole number	0
Weight	number	0

Send `ShippingReceipt` without `ShippingAddresses` only to reuse a saved address: include its `Seq` number.

Response and errors

On success the API returns **HTTP 200** with the store, the receipt number and the customer used. All values are returned as strings, and `CustomerID` is blank when the receipt has no customer.

```
{"results":[{"StoreID":"01","ReceiptNo":"52717309450","CustomerID":"t300"}]}
```

Keep the `ReceiptNo`: it identifies the sale in VRP. On any error the API returns **HTTP 500** with the error message in the body, and nothing is saved. Fix the request and send it again.

Message contains	Cause
parameter is not valid JSON	The <code>Json</code> value is not valid JSON, or was cut short by missing URL encoding
StoreID is mandatory	<code>StoreID</code> missing or blank
TRSDate is mandatory	<code>TRSDate</code> missing or blank
every RPTrsd line must have a UPC	An item line has no <code>UPC</code>
every rpPay line must have a PayType	A payment has no <code>PayType</code>

Message contains	Cause
must have at least one RPTTrsd (UPC) line or one rpPay line	No item lines and no payments
must be JSON objects { }	RPCust , ShippingAddresses or ShippingReceipt sent as an array
need a customer	Shipping sent without CustomerID or RPCust
must have the Seq of an existing address	ShippingReceipt sent alone without Seq
rpTransaction C0 / LA requires a customer	Customer order or layaway without a customer
receipt ... already exists for store ...	The ReceiptNo you sent was already used; leave it out to let VRP assign one
Authentication Failed (HTTP 401)	Missing or wrong API username or password

Other messages come from the database, for example a value that is too long or not a number in a numeric field.

To avoid saving the same sale twice after a timeout, put your own order number in GovReceiptNo or TRSNote , so the sale can be found in VRP before you resend it.

Complete examples

A web order with a customer and shipping

Two items with tax, paid by credit card, shipped to the customer. VRP finds the customer by email, or creates them.

```
{
  "StoreID": "01",
  "TRSDDate": "2026-10-08T14:30:00",
  "Cashregister": "WEB",
  "Cashier": "API",
  "GovReceiptNo": 100245,
  "TRSNote": "Web order 100245",
  "RPTTrsd": [
```

```

    { "UPC": "039500207701", "Quantity": 2, "Amount": 6.38, "LineTax": 0.57
  },
  { "UPC": "041594869741", "Amount": 6.69, "LineTax": 0.59 }
],
"rpPay": [
  { "PayType": "V", "Amount": 14.23 }
],
"RPCust": {
  "FirstName": "Pat", "LastName": "O'Brien",
  "EmailAddress": "pat@example.com", "PhoneNumber": "212-555-1212"
},
"ShippingAddresses": {
  "FirstName": "Pat", "LastName": "O'Brien",
  "Address1": "275 W 96th Street", "City": "New York", "State": "NY",
  "Zip": "10025"
},
"ShippingReceipt": { "Weight": 1.5 }
}

```

A minimal walk-in sale

```

{ "StoreID": "01", "TRSDDate": "2026-10-08",
  "RPTrsd": [ { "UPC": "039500207701", "Amount": 3.19 } ],
  "rpPay": [ { "PayType": "C", "Amount": 3.19 } ] }

```

Calling the API

`curl (-G` with `--data-urlencode` does the URL encoding):

```

curl -G "http://<server>:<port>/CreateReceiptFromJson" \
  -u "<username>:<password>" \
  --data-urlencode "Json@receipt.json"

```

C#:

```

using var http = new HttpClient();
var auth = Convert.ToBase64String(Encoding.UTF8.GetBytes("<username>:
<password>"));
http.DefaultRequestHeaders.Authorization = new
AuthenticationHeaderValue("Basic", auth);

string json = File.ReadAllText("receipt.json");
string url = "http://<server>:<port>/CreateReceiptFromJson?Json=" +
Uri.EscapeDataString(json);

var resp = await http.GetAsync(url);
string body = await resp.Content.ReadAsStringAsync();
if (!resp.IsSuccessStatusCode) throw new Exception(body);    // error message
from VRP
// body: {"results":[{"StoreID":"01","ReceiptNo":"...","CustomerID":"..."}]}

```

JavaScript:

```

const receipt = { StoreID: "01", TRSDate: "2026-10-08",
  RPTrsd: [{ UPC: "039500207701", Amount: 3.19 }],
  rpPay: [{ PayType: "C", Amount: 3.19 }] };

// JSON.stringify does not escape apostrophes, so replace them with '
const json = JSON.stringify(receipt).replace(/'/g, "\\u0027");
const res = await fetch(
  "http://<server>:<port>/CreateReceiptFromJson?Json=" +
encodeURIComponent(json),
  { headers: { Authorization: "Basic " + btoa("<username>:<password>") } });
if (!res.ok) throw new Error(await res.text());
const { results } = await res.json();
console.log(results[0].ReceiptNo);

```