Partner Gateway API Guide (SpecBooks)

Version 1.1 | Namespace: `/api/specbooks`

1) Overview

- SpecBooks integrates only with this Gateway API.
- All command endpoints are asynchronous: `202 + jobId` then poll job status.
- Direct Acumatica access is deprecated effective `TODO: INSERT DATE`.

2) Base URL, Versioning, Auth

- Base URL: `https://<gateway-host>`
- Versioning: current contract namespace is `/api/specbooks`. Breaking changes will move to a new namespace (for example, `/api/specbooks/v2`).
- Required header on every request: `X-SPECBOOKS-API-KEY: <api-key>`
- POST/PATCH require `Content-Type: application/json`

Required headers summary

Header	Required for	Notes
`X-SPECBOOKS-API-KEY`	All endpoints	Shared partner key
`Content-Type: application/json`	POST, PATCH	JSON body required
`Idempotency-Key`	POST /opportunities	Required; dedupes creates

3) Supported Endpoints (Only 5)

Method	Path	Purpose	Immediate response
`GET`	`/api/specbooks/customers/{customerId}`	Queue customer fetch from Acumatica	`202 { \"jobId\": \"...\" }`
`GET`	`/api/specbooks/opportunities/{opportunityId}`	Queue opportunity fetch from Acumatica (expanded products)	`202 { \"jobId\": \"...\" }`
`POST`	`/api/specbooks/opportunities`	Queue create opportunity	`202 { \"jobId\": \"...\" }`
`PATCH`	`/api/specbooks/opportunities/{opportunityId}`	Queue update opportunity (coalesced)	`202 { \"jobId\": \"...\" }`
`GET`	`/api/specbooks/jobs/{jobId}`	Read job status/result/error	`200`

4) Async Job Lifecycle

- Statuses: `queued -> processing -> succeeded | failed`
- All command endpoints return: `{ "jobId": "..." }`
- Poll endpoint: `GET /api/specbooks/jobs/{jobId}`
- Job retention: jobs are retained for a minimum of `30 days`, then may be purged.
- After purge, polling a removed job returns `404`.

Recommended polling strategy

- Attempt 1..5: every 1 second
- Attempt 6..20: every 2 seconds
- Then every 5 seconds up to a client timeout of 2 minutes
- If timeout is reached, surface "pending" to user and allow manual refresh

5) Idempotency Rules

- Applies to `POST /opportunities`.
- Required header: `Idempotency-Key` (recommend UUID).
- Scope: unique per vendor (`specbooks`), per key string.
- Current behavior: if key already exists, gateway returns the existing `jobId` and does not enqueue a duplicate.
- Current retention: keys are retained for a minimum of `90 days` (may be retained longer).
- Failed-job behavior: same key returns the same prior jobId; client should use a new key for a new create attempt.

6) Request Field Allowlist

POST /opportunities (create)

- Top-level: `Subject, ClassID, BusinessAccount, Location, Owner, Products, ContactInformation, Address, Hold`
- `Products[]` create fields: `InventoryID` (required), `Quantity`, `UOM`
- Canonical quantity field for create is `Quantity` (not `Qty`).
- Unknown fields are rejected with 400 (strict recursively for top-level and nested objects).

PATCH /opportunities/{opportunityId}

- Top-level allowed: `Subject, ClassID, BusinessAccount, Location, Owner, Products, ContactInformation, Address, Hold`
- **Forbidden:** top-level `OpportunityID` (URL path is authoritative)
- `Products[]` update fields: `id`, `OpportunityProductID`, `InventoryID`, `Qty`, `Quantity`, `UOM`, `Warehouse`, `delete`
- Canonical quantity field for patch is `Qty`; `Quantity` is accepted as alias. Do not send both in the same line item.
- Unknown fields are rejected with 400 (strict recursively for top-level and nested objects).

Product line semantics for PATCH

Action	How to send
Add line	Line without `id`, with `InventoryID`
Update line	Line with `id` from GET opportunity result
Delete line	Line with `id` and `"delete": true`

7) Endpoint Examples

GET customer command response

```
{ "jobId": "eb605359-6789-44cf-b3d2-020a4c11bf6f" }
```

GET opportunity command response

```
{ "jobId": "757080cb-1be8-4ef3-9812-f7da2258a8cd" }
```

POST create request example

```
{
  "Subject": { "value": "New Project" },
  "Products": [
    { "InventoryID": { "value": "SKU-100" }, "Quantity": { "value": 1 } }
  ]
}
```

PATCH update + add example

```
{
  "Products": [
    {
      "id": "aa252933-2909-f111-9fbe-6045bda28239",
      "Qty": { "value": 2 },
      "Warehouse": { "value": "SALT LAKE APPLIANCES" }
    },
    {
      "InventoryID": { "value": "ROOM" },
      "Qty": { "value": 1 },
      "Warehouse": { "value": "SALT LAKE APPLIANCES" }
    }
  ]
}
```

PATCH delete line example

```
{
  "Products": [
    {
      "id": "88898d89-2909-f111-9fbe-6045bda28239",
      "delete": true
    }
  ]
}
```

8) Job Result Shape Examples

Result shape is passthrough JSON from Acumatica. In current GET customer/opportunity usage it is typically an array. Create/update may return an object. Clients should treat `result` as generic JSON.

GET /jobs/{jobId} (succeeded customer)

```
{
  "jobId": "...",
  "vendorId": "specbooks",
  "type": "GET_CUSTOMER",
  "status": "succeeded",
  "result": [ { "CustomerID": { "value": "BA0001318" }, "CustomerName": { "value": "..." } } ],
  "error": null,
  "createdAt": "...",
```

```
  "updatedAt": "..."
}
```

GET /jobs/{jobId} (succeeded opportunity)

```
{
  "jobId": "...",
  "vendorId": "specbooks",
  "type": "GET_OPPORTUNITY",
  "status": "succeeded",
  "result": [ {
    "OpportunityID": { "value": "OP11995" },
    "Products": [
      {
        "id": "aa252933-2909-f111-9fbe-6045bda28239",
        "OpportunityProductID": { "value": 4 },
        "InventoryID": { "value": "DF48650G/S/P" },
        "Qty": { "value": 1 },
        "UOM": { "value": "EACH" },
        "Warehouse": { "value": "SALT LAKE APPLIANCES" }
      },
      {
        "id": "88898d89-2909-f111-9fbe-6045bda28239",
        "OpportunityProductID": { "value": 7 },
        "InventoryID": { "value": "822113" },
        "Qty": { "value": 1 },
        "UOM": { "value": "EACH" },
        "Warehouse": { "value": "SALT LAKE APPLIANCES" }
      }
    ]
  } ],
  "error": null,
  "createdAt": "...",
  "updatedAt": "..."
}
```

GET /jobs/{jobId} (succeeded create/update)

```
{
  "jobId": "...",
  "vendorId": "specbooks",
  "type": "CREATE_OPPORTUNITY or UPDATE_OPPORTUNITY",
  "status": "succeeded",
  "result": { "...": "Acumatica response payload" },
  "error": null,
  "createdAt": "...",
  "updatedAt": "..."
}
```

GET /jobs/{jobId} (failed)

```
{
  "jobId": "...",
  "vendorId": "specbooks",
```

```
  "type": "UPDATE_OPPORTUNITY",
  "status": "failed",
  "result": null,
  "error": "Acumatica request failed: 400 ...",
  "createdAt": "...",
  "updatedAt": "..."
}
```

9) Error Envelope and Status Codes

Standard error envelope

```
{
  "error": "Validation failed | Unauthorized | ...",
  "issues": [
    { "path": "Products.0.InventoryID", "message": "Required" }
  ]
}
```

Status	Meaning	Action
400	Validation error	Fix payload/headers, retry after fix
401	Invalid API key	Fix auth key
404	Unknown jobId	Verify jobId or retention window
413	Payload too large	Reduce body size
429	Rate limit exceeded	Use `Retry-After`, exponential backoff + jitter
500/503	Server/upstream issue	Retry with backoff

10) Throughput Protection

- Gateway route limits (defaults): GET 30 rpm, write 20 rpm.
- Worker limits (defaults): vendor 8 concurrency / 90 rpm; global 12 concurrency / 200 rpm.
- PATCH coalescing: per `opportunityId`, latest update wins inside debounce window (default 5000ms).
- If a PATCH is already pending for an opportunity, gateway returns the existing pending `jobId` and updates buffered payload to the newest request.

11) Partner Integration Checklist

- Implement async job polling pattern.
- Store line `id` values from GET opportunity for future line updates/deletes.
- Use idempotency key for create retries.
- Implement retry policy for `429/5xx`.
- Handle validation errors using `issues[]` paths.