POS API Guide — Tenants, Branches, Catalog, Menu, Inventory & Orders

Written for frontend devs — explain like I’m 5 (with simple pictures & Postman-style calls).

Key idea: you sell things at different shop locations (branches). Customers place orders at a branch POS. All data (categories, items, modifiers, menus, inventory) is connected by IDs.

1) The Mental Model (Very Simple)

- Tenant = A company/brand (e.g., "Extraction").
- Branch = One shop/location of that tenant (e.g., Extraction DHA store). Each branch is one POS.
- Catalog = What you sell: categories → items (with sizes) → optional modifiers (like milk type).
- Menu = Which parts of the catalog are visible at a specific branch right now.
- Advanced Catalog = Extra business rules like combos, taxes, and service charges.
- Inventory = Raw materials and stock (e.g., beans, milk). Recipes link sold items to stock consumption.
- Orders/KDS/Payments = The POS creates orders, kitchen prepares, inventory deducts when accepted, then payment is taken.

```
[ Tenant: Extraction ]
        |
   +----+---------------------------+
   |                                |
[Branch: DHA POS]            [Branch: Gulberg POS]
      |                             |
   (Menu for DHA)               (Menu for Gulberg)
      |                             |
   shows Catalog (Coffee, Pastry...) with Items & Modifiers
      |
   POS builds Orders → KDS (kitchen) → Inventory deducts on "accepted"
```

2) Headers & Variables (used in almost every call)

Header	Value / Notes
Authorization	Bearer {{tenantToken}} (from tenant login)
x-tenant-id	{{tenantSlug}} (e.g., extraction)
x-branch-id	Only for branch-specific actions (orders, KDS, some status updates)
Content-Type	application/json

Frontend should also keep these IDs once created: tenantToken, branchId, categoryId, modifierGroupId + optionIds, itemId + variantIds, menuId, invItemId, poId, orderId.

3) Full Flow (Step-by-Step, Postman Style)

3.1 Tenant Login (Owner/Manager)

POST /t/auth/login

```
Headers:
  x-tenant-id: {{tenantSlug}}
  Content-Type: application/json

Body:
{
  "email": "owner@extraction.local",
  "password": "P@ssw0rd123"
}

Response (store):
  result.token  -> {{tenantToken}}
```

3.2 Branches (each branch is one POS)

Create a branch

```
POST /t/branches
Headers:
  Authorization: Bearer {{tenantToken}}
  x-tenant-id: {{tenantSlug}}
  Content-Type: application/json

Body:
{
  "name": "Extraction – DHA",
  "code": "dha",
  "timezone": "Asia/Karachi",
  "currency": "PKR",
  "address": { "city": "Lahore" }
}

Response (store):
  result._id -> {{branchId}}
```

Repeat to add more branches (e.g., Gulberg).

3.3 Catalog (Categories → Modifiers → Items)

Create Category

```
POST /t/catalog/categories
Headers: Authorization, x-tenant-id, Content-Type

Body:
{ "name": "Coffee", "slug": "coffee", "isActive": true }

Response:
  result._id -> {{catId_coffee}}
```

Create Modifier Group (e.g., Milk Options)

```
POST /t/catalog/modifiers
Body:
{
  "name": "Milk Options",
  "key": "milk-options",
  "selection": "single",
  "min": 0,
  "max": 1,
  "options": [
    { "name": "Whole Milk" },
    { "name": "Skim Milk" },
    { "name": "Oat Milk", "price": 60 },
    { "name": "Almond Milk", "price": 60 }
  ]
}

Response:
  result._id                  -> {{modId_milk}}
  result.options[i]._id       -> optionIds (store all)
```

Create Item (attach categories & modifiers, define variants)

```
POST /t/catalog/items
Body:
{
  "name": "Latte",
  "slug": "latte",
  "categoryIds": ["{{catId_coffee}}"],
  "variants": [
    { "name": "Small",  "price": 480 },
    { "name": "Medium", "price": 560 },
    { "name": "Large",  "price": 640 }
  ],
  "modifiers": [
    { "groupId": "{{modId_milk}}", "required": false, "min": 0, "max": 1 }
  ],
  "isActive": true
}

Response:
  result._id -> {{itemId_latte}}
  result.variants[i]._id -> variantIds (store all)
```

3.4 Menu (what each branch shows)

Create & Publish Menu for a Branch

```
POST /t/catalog/menus
Body:
{
  "branchId": "{{branchId}}",
  "title": "DHA Menu",
  "status": "draft",
  "sections": [
    { "name": "Coffee", "categoryId": "{{catId_coffee}}", "itemIds": [], "sortIndex"
: 10 }
  ]
}

Response:
  result._id -> {{menuId}}

POST /t/catalog/menus/{{menuId}}/publish

GET /t/catalog/menus/branch/{{branchId}}/pos
  -> use result.items[] and result.categories[] to render POS grid
```

3.5 Advanced Catalog (optional but common)

```
POST /t/catalog/taxes
{ "name": "Standard VAT", "code": "vat16", "rate": 0.16, "inclusive": false, "countr
y": "PK" }

POST /t/catalog/service-charges
{
  "name": "Dine-In Service",
  "code": "svc-dinein",
  "type": "percentage",
  "value": 0.05,
  "applyOn": ["dinein"],
  "branchIds": ["{{branchId}}"]
}
```

3.6 Inventory (stock items, recipes, POs)

Create Inventory Items

```
POST /t/inventory/items
{ "name": "Espresso Beans", "sku": "BEANS-ESP", "uom": "g", "minThreshold": 500 }

Response:
  result._id -> {{invItemId_beans}}
```

Link Catalog Item → Inventory Ingredients (Recipe for consumption)

```
POST /t/inventory/recipes
{
  "catalogItemId": "{{itemId_latte}}",
  "catalogVariantId": "",
  "yieldQty": 1,
  "ingredients": [
    { "itemId": "{{invItemId_beans}}", "uom": "g", "qty": 18 }
  ],
  "notes": "Latte base recipe"
}
```

Receive Stock (Purchase Order)

```
POST /t/inventory/purchase-orders
{
  "supplierName": "Bean Supply Co.",
  "branchId": "{{branchId}}",
  "lines": [
    { "itemId": "{{invItemId_beans}}", "uom": "g", "qty": 5000, "unitCost": { "amoun
t": 0.02, "currency": "PKR" } }
  ]
}

Response:
  result._id -> {{poId}}

POST /t/inventory/purchase-orders/{{poId}}/receive
  -> Increments on-hand and writes ledger rows
```

3.7 POS Orders (build cart, submit order, accept, pay)

Create Order

```
POST /t/orders
Headers: x-branch-id: {{branchId}}
Body:
{
  "branchId": "{{branchId}}",
  "channel": "dinein",
  "lines": [
    {
      "itemId": "{{itemId_latte}}",
      "variantId": "{{variantId_medium}}",
      "quantity": 1,
      "appliedModifiers": [
        { "groupId": "{{modId_milk}}", "optionIds": ["{{optId_oat_milk}}"] }
      ]
    }
  ],
  "notes": "No sugar"
}

Response:
  result._id -> {{orderId}}
```

Move to 'accepted' (inventory deduction happens here)

```
POST /t/orders/{{orderId}}/status
Headers: x-branch-id: {{branchId}}
Body:
{ "status": "accepted" }
```

Kitchen Display System (KDS) Board

```
GET /t/kds/board?branchId={{branchId}}&statuses;=accepted,preparing,ready
```

Record Payment

```
POST /t/orders/{{orderId}}/pay
{ "method": "cash", "amount": 560 }
```

4) How Things Link (Explain Like I’m 5)

• Category is like a shelf: “Coffee Shelf”, “Pastry Shelf”.

• Item sits on a shelf: “Latte” sits on Coffee Shelf, “Croissant” sits on Pastry Shelf.

• Item sizes (variants): Small/Medium/Large (each has its own price and ID).

• Modifiers are choices: Milk type for Latte (Whole, Skim, Oat…). Some can add extra price.

• Menu chooses which shelves to show in a branch. Publish the menu so POS can see it.

• Inventory is the kitchen store: you say “One Latte uses 18g beans”. When order is accepted, 18g is deducted.

```
Catalog:
  Category: Coffee
    Item: Latte
      Variants: Small, Medium, Large
      Modifiers:
        - Milk Options (single select)
            Whole / Skim / Oat(+60) / Almond(+60)
  Category: Pastry
    Item: Croissant

Menu (for DHA Branch):
  Section "Coffee" -> shows all items from Category: Coffee
  Section "Bakery" -> shows all items from Category: Pastry

POS:
  Barista taps "Latte (Medium) + Oat Milk" -> creates an order line with itemId, var
iantId, and modifier optionId.
```

That’s it. Build once (catalog, menu), pick a branch, sell items (orders), kitchen prepares, inventory deducts, take payment.