Configuration Module — Business Guide

ERP FLOW · Bilingual Documentation

English & বাংলা

Generated from `backend/Modules/Configuration/routes/api.php` · Code-verified

Part A — English

Configuration Module — Business Guide (English)

This guide explains the **Configuration** module of the ERPFLOW backend in everyday business language. It is written so that HR, Admin, and operations staff can understand it without technical knowledge.

> *বাংলা ভার্সন: **CONFIGURATION_MODULE_GUIDE_BN.md***

---

1. What is this module?

The Configuration module is the **shared "master data" centre** of the ERP. It is the dictionary file that many other modules (Employee, Attendance, Payroll, Onboarding, etc.) rely on.

Instead of each module keeping its own copy of information such as what the departments are, what job titles exist, or where people work, the Configuration module stores these once and lets everyone reuse them.

It is divided into two big areas:

Area	Purpose
Employee / HR masters	Employment types, salary structures, document types, designations, grades, teams, divisions, branches, work locations, sections, cost centers, employee ID card settings
System Configuration	Companies and departments (the tenant structure the whole system sits on)

Key idea: almost everything here is scoped to the **currently selected company** (tenant). A user only sees and manages master data belonging to the company they are currently working in.

> *All routes are served under the **`api/v1/configuration`** prefix and require a logged-in user.*

---

2. Who should use it?

Role	Typical use
System / Super Admin	Manage the companies (tenants) that the ERP serves
HR / Admin	Maintain departments, designations, job types, document types, grades, teams, divisions, branches, sections, cost centers, work locations
Payroll / Finance	Read the salary structures, ID card templates, work locations
Regular employees (read-only via pickers)	Choose a department, section, cost center, work location, document type, etc. when filling forms

Because of the `onboarding.access` and `onboarding.document` middleware, the access also depends on the person being logged in, having completed onboarding, and not having an expired document deadline. Because of the `permission:*` middleware, each operation additionally requires a specific permission (see **Section 6**).

> ***Access model (verified in `PermissionEngine::userCan`, `config/platform.php`, `config/actions.php`):** access is checked per permission key for the currently selected company. Users with user-type `developer` (default `bypass_user_types`) or holding the system role `administrator` inside the company get **full access** without role-matrix grants. Everyone else receives permissions from their roles (`role_permissions`) plus per-user allow/deny overrides; a **deny** override always wins.*

---

3. Before you start (prerequisites)

To create or use the data managed by this module, the following should exist:

1. A **company** (tenant) for all company-scoped masters, in an **active** status. 2. An **active employee user** account for the person who will be department head, and correct authorization. 3. Valid authentication (JWT via `auth:api`), a **completed onboarding**, and a **non-expired document deadline**. 4. The relevant role permissions such as `configuration.view`, `configuration.create`, `configuration.update`, `configuration.delete`, and `configuration.system-configuration-menu-view`.

---

4. What problems does it solve?

Business problem	How this module solves it
Duplicate or inconsistent "department" or "designation" names across the system	One company-scoped master record reused everywhere
No central place to configure a tenant/company structure	Company + department management under System Configuration
Hiring needs standard lists (job types, document types, grades, teams, branches)	Picker-style master data that Employee onboarding and HR forms reuse
Nobody is responsible for a department	Department management with a **department head** picker
Payroll depends on salary structures	Salary structures are maintained centrally and read here
HR cannot determine an employee's workplace / cost grouping	Work locations, sections, and cost centers as company-scoped masters

---

5. Main features (described in business language)

5.1 System health check

- **What it is:** a lightweight "is the module alive" signal for monitoring.
- **API:** `GET /api/v1/configuration/health`
- **Permission:** `configuration.view`
- **Controller/Service:** `ConfigurationController::health` → `ConfigurationService::getHealthStatus`
- **Business impact:** used by DevOps/monitoring; logs an activity event `configuration.health_viewed`.

5.2 Employment types (Full-time, Contract, Intern, etc.)

- **What it is:** the types of contract/relationship an employee can have.
- **API (read):** `GET /employment-types`, `GET /employment-types/{id}`
- **API (write):** `POST /system-configuration/employment-types`, `PUT /system-configuration/employment-types/{id}`, `DELETE /system-configuration/employment-types/{id}`
- **Controllers:** `EmploymentTypeController` + `EmploymentTypeService` + `EmploymentTypeRepository`, model `EmploymentType`; requests `Store/UpdateEmploymentTypeRequest`.
- **Dependencies:** belongs to a company; consumed by the Employee module (Employment section).
- **Business impact:** determines job classification used by HR and (indirectly) leave/payroll rules.

5.3 Salary structures

- **What it is:** named salary frameworks an employee gets assigned to.
- **API (read-only in this module):** `GET /salary-structures`, `GET /salary-structures/{id}`; also `GET /system-configuration/salary-structures`, `GET /system-configuration/salary-structures/{id}`
- **Controller:** `SalaryStructureController` (`index` / `show` only), model `SalaryStructure`.
- **Important note (verified in the route files):** the Configuration route file comment says *"Salary

structure writes moved to Payroll (`payroll.structure-manage`)"* — and `Modules/Payroll/routes/api.php` confirms it: `POST /salary-structures`, `PUT /salary-structures/{id}`, `PATCH .../activate`, `PATCH .../deactivate` and the `/components` endpoints all live in the Payroll module behind `permission:payroll.structure-manage`. The Configuration `SalaryStructureController` still contains `store/update/destroy` methods, but **no write route is registered for them in this module**. - **Dependencies:** owned by a company; consumed by Employee (Salary) and Payroll (which owns the writes). - **Business impact:** payroll and HR always see the same salary frameworks. ### 5.4 Document types - **What it is:** the "roll types" HR allows in employee files (NID, passport, offer letter, etc.), including sensitive ("restricted") and expiry-required flags. - **API:** `GET /document-types`, `GET /document-types/active`, `GET /document-types/{id}`; writes under `/system-configuration/document-types`. - **Controllers:** `DocumentTypeController` + `DocumentTypeService`, model `DocumentType` (fields `is_restricted`, `requires_expiry`). - **Dependencies (verified):** used by the Employee module — `EmployeeDocument`, `EmployeeIdentity` (via `EmployeeDocumentService`) — and by the Onboarding document-upload step (`DocumentUploadStepHandler`). The

employee ID card feature does **not** reference document types. - **Business impact:** controls what classified employees can upload; restricted documents need a special role.

5.5 Designations (job titles)

- **What it is:** the titles employees are hired into.
- **API:** `GET /system-configuration/designations`, `POST /system-configuration/designations`, `PUT /system-configuration/designations/{id}`, `DELETE /system-configuration/designations/{id}`
- **Controllers:** `DesignationController` (`listDesignations`, `storeDesignation`, `updateDesignation`, `destroyDesignation`), `DesignationService`, model `Designation`.
- **Dependencies:** consumed by the Employee module (Organization assignment).

5.6 Employee grades (grade / level)

- **What it is:** the grade/level ladder (e.g. Grade-1 … Grade-10) that ranks employees.
- **API:** `GET /grades`, `POST /grades`, `PUT /grades/{id}`, `DELETE /grades/{id}`
- **Controller:** `Grade/EmployeeGradeManagementController` + `GradeService`, model `Grade`.
- **Note:** delete soft-archives (`archived_at`) so historical assignments keep their names.
- **Dependencies:** belongs to a company (grade uniqueness is company-scoped); consumed by Employee.

5.7 Teams

- **What it is:** the team groupings people are organised into.
- **API:** `GET /teams`, `POST /teams`, `PUT /teams/{id}`, `DELETE /teams/{id}`
- **Controller:** `Team/TeamManagementController` + `TeamService`, model `Team`.

5.8 Company divisions

- **What it is:** the divisions / business units under a company.
- **API:** `GET /divisions`, `POST /divisions`, `PUT /divisions/{id}`, `DELETE /divisions/{id}`
- **Controller:** `Division/CompanyDivisionManagementController` + `DivisionService`, model `Division`.

5.9 Company branches

- **What it is:** the physical branches of the company.
- **API:** `GET /branches`, `POST /branches`, `PUT /branches/{id}`, `DELETE /branches/{id}`
- **Controller:** `Branch/EmployeeBranchManagementController` + `BranchService`, model `Branch`.

5.10 Employee ID card settings

- **What it is:** templates that control the employee ID card design (portrait/landscape, fields, validity).
- **API:** `GET /employee-id-card-settings`, `POST /employee-id-card-settings`, `PUT /employee-id-card-settings/{settingId}`, `DELETE /employee-id-card-settings/{settingId}`
- **Controller:** `EmployeeIdCard/EmployeeIdCardManagementController` + `EmployeeIdCardSettingService`, models `EmployeeIdCardSetting` (belongs to company) and `EmployeeIdCard`.

5.11 Sections, Cost centers, Work locations (the "flat org masters")

- **What they are:**
- **Sections**: a unit within the organization
- **Cost centers**: a budget / cost grouping (finance-oriented)
- **Work locations**: where employees physically work
- **API per row (list + active + show + store + update + destroy):**
- `GET /sections`, `GET /sections/active`, `GET /sections/{id}`, `POST /sections`, `PUT /sections/{id}`, `DELETE /sections/{id}`
- `GET /cost-centers`, `GET /cost-centers/active`, `GET /cost-centers/{id}`, `POST /cost-centers`, `PUT /cost-centers/{id}`, `DELETE /cost-centers/{id}`
- `GET /work-locations`, `GET /work-locations/active`, `GET /work-locations/{id}`, `POST /work-locations`, `PUT /work-locations/{id}`, `DELETE /work-locations/{id}`
- **Controllers:** `SectionController`, `CostCenterController`, `WorkLocationController` (each with a service + repository + model + request/resource).
- **Permission:** reads under `configuration.view`; writes under `configuration.create` / `.update` / `.delete`.
- **Business impact:** the `active` endpoints exist for the picker forms (only active values are shown when assigning an employee).

---

6. How access is controlled (verified from code)

The whole route group runs through three middleware layers:

1. `auth:api` — the caller must present a valid JWT for an active user. 2. `onboarding.access` + `onboarding.document` — new joiners must complete onboarding; an expired document deadline blocks access (verified in `EnsureOnboardingAccess` / `EnsureDocumentDeadline`). 3. `permission:*` — `App\Platform\Http\Middleware\EnsurePermission` calls `PermissionEngine::userCan()`, which grants a key only when the user is **active**, the request has a resolved **company**, and the key is either granted through roles/overrides or bypassed.

Where permission keys come from

Permission keys are generated from the action catalog `config/actions.php`. The Configuration module does not opt out of common verbs, so its published keys are **common actions ∪ module-specific actions**:

Key	Kind	Referenced in this route file?
`configuration.view`	common action `view`	✅ yes
`configuration.create`	common action `create`	✅ yes
`configuration.update`	common action `update`	✅ yes
`configuration.delete`	common action `delete`	✅ yes
`configuration.system-configuration-menu-view`	module-specific action	✅ yes
`configuration.system-configuration-company-view`	module-specific action	❌ catalogued, but **no route references it**
`configuration.approve` / `.reject` / `.assign` / `.export` / `.import` / `.print` / `.clone` / `.archive` / `.restore` / `.download` / `.upload` / `.menu-view`	remaining common actions	❌ not referenced by this route file

So exactly **five keys** gate every endpoint in this file:

Permission	What it allows (as wired in the routes)
`configuration.view`	Read the core picker lists (employment types, salary structures, document types, sections, cost centers, work locations) and the health endpoint
`configuration.system-configuration-menu-view`	Read/List System Configuration data (companies, departments, department-head candidates, designations, employment types, salary structures, document types, grades, teams, divisions, branches, employee ID card settings)
`configuration.create`	Create every master record and system entity (POST routes)
`configuration.update`	Edit existing records (PUT routes)
`configuration.delete`	Delete/archive master records (DELETE routes; soft archive via `archived_at` where the model defines it)

How a key becomes "granted" (verified in `PermissionEngine`)

1. `developer` user-type (`bypass_user_types`, default in `config/platform.php`) or the system role slug `administrator` (`bypass_role_slugs`) inside the current company → **all** active permissions. 2. Otherwise, the granted set = role-linked keys (`role_permissions`, only for roles of the current company) ∪ per-user **allow** overrides, minus per-user **deny** overrides (deny always wins). 3. The permission record itself must have `status = active`.

Approval wiring (verified)

`config/actions.php` marks Configuration `create`, `update`, `delete` as **requires_approval**. `ConfigurationApprovalSeeder` seeds the **"Configuration Create Approval"** workflow (one parallel step, resolver_type `role`, Administrator role, completion rule `any`) and a default setting with `approvalEnabled = false` for the demo company — i.e. the approval machinery exists and `configuration.create` is registered to it, but requests are **not blocked by default** in the demo environment.

Who is expected to use it (verified from demo seeders)

- `administrator` role / `developer` users → full access to everything (bypass).
- Demo role **`configuration-staff`** (`ConfigurationRoleSeeder`) → seeded with only `configuration.view`

(plus `employee.update-own-profile`), i.e. read-only picker access. - Any other role must be given the needed `configuration.*` keys through the Platform role-matrix screen; the route file does not hard-code a role list.

Company (tenant) isolation (verified)

`SetCompanyContext` resolves the company from the `X-Company-Id` header, falling back to the user's default membership. Non-platform users are confined to companies they belong to; platform-scoped users (developer / super-admin) may act in every **active** company. When no company resolves, tenant-scoped controllers throw `PermissionDeniedException('company.context')` — data is never silently read across tenants.

---

7. Dependencies on other modules

- **Employee module** — consumes designations, departments, employment types, salary structures, document types, grades, teams, divisions, branches, sections, cost centers, work locations (via pickers).
- **Payroll module** — owns salary-structure *write* operations (`payroll.structure-manage`); this module keeps the read/reference view.
- **Onboarding module** — gates new staff; the `onboarding.access` / `onboarding.document` middleware applies.
- **Auth / Platform** — provides `auth:api`, roles, and the `permission` middleware.
- **Tenancy layer** (`App\Core\Tenancy\TenantContext`) — provides the "current company" for every endpoint.
- **Core Companies** — the company model that departments, grades, teams, etc. belong to.

---

8. A typical business workflow

"Onboard a new employee in an existing company":

1. Admin opens **System Configuration** and confirms the **company** is **active**. 2. Admin/HR make sure the required **masters** exist and are active: employment type (Full-time), designation (Software Engineer), grade, team, division, branch, and the relevant **department** (with its head), **section**, **work location**, **cost center**. 3. HR creates a **document type** if the employee document folder needs it. 4. Front-end picker forms call the **active** endpoints (e.g. `/sections/active`) so only valid options show. 5. Payroll assigns a package by selecting the salary structure from the Payroll module; Configuration stays the single reference. 6. When an employee later leaves, `DELETE` on the masters soft-archives them so history keeps the names.

---

9. System / business impact

- **Single source of truth** — one company-scoped master per concept reused across modules.
- **Tenant isolation** — every object is scoped to the current company, so data never leaks between tenants.
- **Safe deletion** — most org masters soft-archive, preserving historical employee assignments.
- **Clear ownership** — Payroll owns salary writes, Configuration owns the reference/read, avoiding double maintenance.

---

10. Common problems (and what they usually mean)

What you see	Likely reason
401 — not logged in	Missing / invalid `auth:api` token
403 — forbidden	Onboarding not finished, document deadline expired, or the role lacks the `permission:*` key (and the user is not a `developer`/`administrator`, which bypass)
Department list empty	No company in context, or the company is not **active**
Cannot pick a value in a form	No **active** master exists; the `active` endpoints drive the pickers
Cannot create a salary structure here	Writes moved to Payroll (`payroll.structure-manage`) — not exposed in this route file
Deleting a department fails	The system blocks deleting a department that has child departments or active employees (data protection)

---

11. Summary

The Configuration module is the shared **administrative reference centre** of the ERP. It defines the companies and departments, plus all the HR / organization masters (employment types, salary structures, document types, designations, grades, teams, divisions, branches, sections, cost centers, work locations, and employee ID card settings) that every other module reuses. Because the data is company-scoped and protected by layered permissions and the onboarding gate, only authorised people can change it — but once defined, the same clean, consistent values flow through every form and every employee operation in the system.

Generated from `backend/Modules/Configuration/routes/api.php`. Written as a business guide; references controllers, services, models and middleware for accuracy.

অংশ বি — বাংলা গাইড

Configuration Module — ব্যবহারকারী গাইড (বাংলা)

এই গাইডে **Configuration** মডিউল সহজ ভাষায় ব্যাখ্যা করা হয়েছে। টেকনিক্যাল জ্ঞান ছাড়াই পড়ে বোঝা যাবে। এখানে API, Module, Role, Permission, Controller, Endpoint ইত্যাদি গুরুত্বপূর্ণ টেকনিক্যাল শব্দ ইংরেজিতে রাখা হয়েছে।

> *ইংরেজি ভার্সন: **CONFIGURATION_MODULE_GUIDE_EN.md***

---

১. এই মডিউল কী?

Configuration মডিউল হলো পুরো ERP-এর **সাধারণ "মাস্টার ডেটা" কেন্দ্র**। একে অভিধান ফাইল (Dictionary File) ভাবা যেতে পারে।

অনেক মডিউল (Employee, Attendance, Payroll, Onboarding ইত্যাদি) সিস্টেমে কী কী Department, পদবি (Designation), কর্মস্থান ইত্যাদি আছে — সেই তথ্য নিজের কাছে আলাদা করে রাখে না। Configuration মডিউল সেগুলো **একবার সেভ করে** সবাইকে ব্যবহারের সুযোগ দেয়।

দুটি বড় অংশ আছে:

অংশ	উদ্দেশ্য
Employee / HR masters	Employment types, Salary structures, Document types, Designations, Grades, Teams, Divisions, Branches, Work locations, Sections, Cost centers, Employee ID card settings
System Configuration	Companies (কোম্পানি/টেন্যান্ট) ও Departments (বিভাগ) — পুরো সিস্টেমকে আকাশ থাকে ভিত্তি

গুরুত্বপূর্ণ ধারণা: এখানে প্রায় সবকিছু **বর্তমান কোম্পানি (tenant)**-এর সাথে যুক্ত। ব্যবহারকারী শুধু সেই কোম্পানির মাস্টার ডেটা দেখে ও ম্যানেজ করতে পারে।

> *সব Route **`api/v1/configuration`** prefix-এর নিচে এবং লগ-ইন থাকা তা সাপেক্ষে চলে।*

---

২. কারা ব্যবহার করবে?

ভূমিকা	সাধারণ ব্যবহার
System / Super Admin	ERP-এর আওতাধীন প্রতিষ্ঠানসমূহ (কোম্পানি, tenant) পরিচালনা
HR / Admin	Department, Designation, Employment type, Document type, Grade, Team, Division, Branch, Section, Cost center, Work location পরিচালনা
Payroll / Finance	Salary structure, ID card template, Work location পড়া/ব্যবহার
সাধারণ কর্মচারী (read-only, picker)	ফর্ম পূরণের সময় Department, Section, Cost center, Work location ইত্যাদি বেছে নেওয়া

এছাড়া `onboarding.access` ও `onboarding.document` Middleware-র কারণে লগইন, onboarding সদৃশ, এবং document deadline- এর উপর নির্ভর করে access নিয়ন্ত্রিত হয়। `permission:*` Middleware-র কারণে প্রতিটি কাজের জন্য নির্দিষ্ট Permission আবশ্যক (দেখুন **নীচে অংশ ৬**)।

> ***Access model (কোডে যাচাই: `PermissionEngine::userCan`, `config/platform.php`, `config/actions.php`):** access প্রতি Permission key-এর জন্য, বর্তমান কোম্পানির ভেতরে যাচাই হয়। যাদের user-type `developer` (ডিফল্ট `bypass_user_types`) অথবা কোম্পানিতে system role `administrator` (`bypass_role_slugs`) আছে, তারা **full access** পায়। বাকি সবাই Role-এর মাধ্যমে (`role_permissions`) ও per-user allow/deny override-এর ভিত্তিতে Permission পায়; **deny** override সবসময় জেতে।*

---

৩. শুরু করার আগে যা থাকতে হবে

এই মডিউলের ডেটা তৈরি বা ব্যবহার করতে নিচেরগুলো থাকা দরকার:

1. একটি **active কোম্পানি (company/tenant)** — কারণ সব মাস্টার ডেটা কোম্পানি-নির্ভর। 2. Department head করার জন্য সেই ব্যক্তির **active employee user অ্যাকাউন্ট** ও সঠিক অনুমোদন। 3. বৈধ authentication (`auth:api`), **সম্পন্ন onboarding**, এবং **অ-মেয়াদোত্তীর্ণ document**

deadline। 4. প্রয়োজনীয় Role Permission: `configuration.view`, `configuration.create`, `configuration.update`, `configuration.delete` এবং `configuration.system-configuration-menu-view`।

---

৪. এই মডিউল কোন কোন সমস্যার সমাধান করে?

ব্যবসায়িক সমস্যা	সমাধান
সিস্টেমজুড়ে Department বা Designation-এর নাম নকল/অসঙ্গত হয়	একটিমাত্র কোম্পানি-নির্ভর master রেকর্ড যা সবাই reuse করে
কোম্পানি/টেন্যান্ট কাঠামো সাজানোর কেন্দ্রীয় জায়গা নেই	System Configuration-এ Company + Department ব্যবস্থাপনা
নিয়োগের জন্য মানসম্মত তালিকা লাগে (job type, document type, grade ইত্যাদি)	Employee/HR ফর্মে ব্যবহারযোগ্য picker-style master data
Department-এর দায়িত্বে কে — তা অস্পষ্ট	**Department head** নির্বাচন ব্যবস্থা
Payroll-এর জন্য salary structure দরকার	কেন্দ্রীয়ভাবে salary structure রক্ষা ও পড়া
কর্মীর কর্মস্থল/খরচ-গোষ্ঠী বোঝা সহজ নয়	Work location, Section, Cost center- কোম্পানি-নির্ভর master

---

৫. প্রধান ফিচারগুলো (সহজ ভাষায়)

5.1 System health check

- কী: মডিউলটি "জীবিত আছে কি না" দেখার হালকা সংকেত।
- **API:** `GET /api/v1/configuration/health`
- Permission: `configuration.view`
- Controller/Service: `ConfigurationController::health` → `ConfigurationService::getHealthStatus`
- ভূমিকা: monitoring-এ ব্যবহৃত; `configuration.health_viewed` অ্যাক্টিভিটি লগে সংরক্ষণ হয়।

5.2 Employment types (Full-time, Contract, Intern ইত্যাদি)

- কী: কর্মচারী কিসের ধরনের সম্পর্ক/চুক্তি থাকতে পারে।
- API (read): `GET /employment-types`, `GET /employment-types/{id}`
- API (write): `POST /system-configuration/employment-types`, `PUT /system-configuration/employment-types/{id}`, `DELETE /system-configuration/employment-types/{id}`
- Controller: `EmploymentTypeController` + `EmploymentTypeService` + `EmploymentTypeRepository`, model `EmploymentType`।
- নির্ভরতা: কোম্পানি-নির্ভর; Employee মডিউল (Employment অংশ) ব্যবহার করে।
- প্রভাব: HR এবং (পরোক্ষে) leave/payroll নিয়মে ব্যবহৃত চাকরি-শ্রেণী নির্ধারণ।

5.3 Salary structures (বেতন কাঠামো)

- কী: name করা বেতন-কাঠামো যা একজন কর্মীকে বরাদ্দ হয়।
- API (এই মডিউলে শুধু read): `GET /salary-structures`, `GET /salary-structures/{id}`; এবং `GET /system-configuration/salary-structures`, `GET /system-configuration/salary-structures/{id}`
- Controller: `SalaryStructureController` (শুধু `index` / `show`), model `SalaryStructure`।
- **গুরুত্বপূর্ণ নোট (Route ফাইলে যাচাই):** Route ফাইলের মন্তব্য অনুযায়ী — *"Salary structure writes moved

to Payroll (`payroll.structure-manage`)"* — এবং `Modules/Payroll/routes/api.php`-এও দেখা যায়: `POST /salary-structures`, `PUT /salary-structures/{id}`, `PATCH .../activate`, `PATCH .../deactivate` ও `/components` Endpoint-গুলো সবই Payroll মডিউলে, `permission:payroll.structure-manage`-এর ভেতরে। Configuration-এর `SalaryStructureController`-এ store/update/destroy মেথড আছে, কিন্তু **এই মডিউলে তাদের জন্য কোনো write route নিবন্ধিত নেই** — write Payroll মডিউলে হয়। - নির্ভর: কোম্পানি-নির্ভর; Employee (Salary) ও Payroll (যাদের write করবে) ব্যবহার করে। - প্রভাব: Payroll ও HR সবসময় একই বেতন-কাঠামো দেখে। ### 5.4 Document types - কী: Employee file-এ HR কী ধরনের ডকুমেন্ট রাখবে (NID, passport, offer letter ইত্যাদি); সাথে সংবেদনশীল ("restricted") ও নির্দিষ্ট মেয়াদ-প্রয়োজন ("requires expiry") চিহ্ন। - **API:** `GET /document-types`, `GET /document-types/active`, `GET /document-types/{id}`; write নিচের অংশে `/system-configuration/document-types`। - Controller: `DocumentTypeController` + `DocumentTypeService`, model `DocumentType` (ক্ষেত্র `is_restricted`, `requires_expiry`)। - নির্ভর: Employee মডিউল (documents), ID cards, এবং Onboarding document ধাপ ব্যবহৃত। - প্রভাব: কর্মচারী কী কী ফাইল আপলোড করতে পারবে তা নিয়ন্ত্রণ; restricted ডকুমেন্টের জন্য বিশেষ Role লাগে।

5.5 Designations (পদবি/চাকরির পদ)

- কী: কর্মচারী কোন পদবিতে নিয়োগ পাবে।
- **API:** `GET /system-configuration/designations`, `POST /system-configuration/designations`, `PUT /system-configuration/designations/{id}`, `DELETE /system-configuration/designations/{id}`

- Controller: `DesignationController` (`listDesignations`, `storeDesignation`, `updateDesignation`, `destroyDesignation`), `DesignationService`, model `Designation`।
- নির্ভর: Employee মডিউল (Organization assignment) ব্যবহার করে।

5.6 Grades (কর্মী পদমর্যাদা/লেভেল)

- কী: কর্মীকে ক্রমিক লেভেলে (যেমন Grade-1 … Grade-10) সাজায়।
- **API:** `GET /grades`, `POST /grades`, `PUT /grades/{id}`, `DELETE /grades/{id}`
- Controller: `Grade/EmployeeGradeManagementController` + `GradeService`, model `Grade`।
- নোট: delete-এ **soft-archive** হয় (`archived_at`), তাই পুরোনো বরাদ্দের নাম সংরক্ষিত থাকে।
- নির্ভর: কোম্পানি-নির্ভর (company-scoped uniqueness); Employee মডিউল ব্যবহৃত।

5.7 Teams

- কী: কর্মীদের সংগঠিত করার দল।
- **API:** `GET /teams`, `POST /teams`, `PUT /teams/{id}`, `DELETE /teams/{id}`
- Controller: `Team/TeamManagementController` + `TeamService`, model `Team`।

5.8 Company divisions

- কী: কোম্পানির অধীন ব্যবসায়িক ইউনিট (বিভাগ)।
- **API:** `GET /divisions`, `POST /divisions`, `PUT /divisions/{id}`, `DELETE /divisions/{id}`
- Controller: `Division/CompanyDivisionManagementController` + `DivisionService`, model `Division`।

5.9 Company branches

- কী: কোম্পানির ভৌত শাখা (শাখা অফিস)।
- **API:** `GET /branches`, `POST /branches`, `PUT /branches/{id}`, `DELETE /branches/{id}`
- Controller: `Branch/EmployeeBranchManagementController` + `BranchService`, model `Branch`।

5.10 Employee ID card settings

- কী: Employee ID card-এর ডিজাইন/টেমপ্লেট নিয়ন্ত্রণ (portrait/landscape, ক্ষেত্র, মেয়াদ)।
- **API:** `GET /employee-id-card-settings`, `POST /employee-id-card-settings`, `PUT /employee-id-card-settings/{settingId}`, `DELETE /employee-id-card-settings/{settingId}`
- Controller: `EmployeeIdCard/EmployeeIdCardManagementController` + `EmployeeIdCardSettingService`, models `EmployeeIdCardSetting` (কোম্পানি-নির্ভর) এবং `EmployeeIdCard`।

5.11 Sections, Cost centers, Work locations ("flat org masters")

- এগুলো কী:
- **Sections**: প্রতিষ্ঠানের ভেতরের একক/বিভাগ
- **Cost centers**: ব্যয়/বাজেট-গোষ্ঠী (finance-নির্ভর)
- **Work locations**: কর্মচারী শারীরিকভাবে কোথায় কাজ করে
- API (প্রতিটির জন্য list + active + show + store + update + destroy):
- `GET /sections`, `GET /sections/active`, `GET /sections/{id}`, `POST /sections`, `PUT /sections/{id}`, `DELETE /sections/{id}`
- `GET /cost-centers`, `GET /cost-centers/active`, `GET /cost-centers/{id}`, `POST /cost-centers`, `PUT /cost-centers/{id}`, `DELETE /cost-centers/{id}`
- `GET /work-locations`, `GET /work-locations/active`, `GET /work-locations/{id}`, `POST /work-locations`, `PUT /work-locations/{id}`, `DELETE /work-locations/{id}`
- Controller: `SectionController`, `CostCenterController`, `WorkLocationController` (প্রতিটির service + repository + model + request)।
- Permission: read-এর জন্য `configuration.view`; write-এর জন্য `configuration.create` / `.update` / `.delete`।
- প্রভাব: **active** Endpoint গুলো picker ফর্মের জন্য — কর্মী বরাদ্দের সময় শুধু সক্রিয় মান দেখানো হয়।

---

৬. কে কীভাবে access পায় (কোড থেকে যাচাইকৃত)

সমস্ত Route তিনটি Middleware লেয়ারের ভেতর দিয়ে চলে:

1. `auth:api` — কলকারীকে পাঠ্য active ইউজারের বৈধ JWT দিতে হবে। 2. `onboarding.access` + `onboarding.document` — নতুন জয়নকে অবশ্যই onboarding সম্পন্ন করতে হবে; document deadline মেয়াদোত্তীর্ণ হলে access আটকে যায় (যাচাই: `EnsureOnboardingAccess` / `EnsureDocumentDeadline`)। 3. `permission:*` — `App\Platform\Http\Middleware\EnsurePermission` → `PermissionEngine::userCan()`-কে ডাকে। এটি তখনই key দেয়, যখন ইউজার **active**, request-এর একটি **কোম্পানি** resolved, এবং key হয় Role/override দিয়ে grant অথবা bypass করা।

Permission key কোথা থেকে আসে

Permission key-গুলো ক্যাটালগ `config/actions.php` থেকে তৈরি হয়। Configuration মডিউল common verb বাদ দেয় না, তাই এর প্রকাশিত key = **common actions ∪ module-specific actions**:

Key	ধরন	এই Rou ফাই ব্যবহ
`configuration.view`	common action `view`	✅ হ
`configuration.create`	common action `create`	✅ হ
`configuration.update`	common action `update`	✅ হ
`configuration.delete`	common action `delete`	✅ হ
`configuration.system-configuration-menu-view`	module-specific action	✅ হ
`configuration.system-configuration-company-view`	module-specific action	❌ ক্যাট আছে কিন্তু **বে Rou ব্যবহ নয়*
`configuration.approve`/`.reject`/`.assign`/`.export`/`.import`/`.print`/`.clone`/`.archive`/`.restore`/`.download`/`.upload`/`.menu-view`	বাকি common actions	❌ Rou ফাই ব্যবহ নয়

অর্থাৎ এই ফাইলের প্রতিটি Endpoint ঠিক **৫টি key** দিয়ে নিয়ন্ত্রিত:

Permission	যা যা করতে দেয় (Route-এ যেভাবে জোড়া)
`configuration.view`	মূল picker তালিকা পড়তে দেয় (employment types, salary structures, document types, sections, cost centers, work locations) এবং health endpoint
`configuration.system-configuration-menu-view`	System Configuration-এর পড়া/তালিকা (companies, departments, department-head candidates, designations, employment types, salary structures, document types, grades, teams, divisions, branches, employee ID card settings)
`configuration.create`	সব master রেকর্ড ও সিস্টেম entity তৈরি (POST routes)
`configuration.update`	বিদ্যমান রেকর্ড সম্পাদনা (PUT routes)
`configuration.delete`	Master রেকর্ড ডিলিট/আর্কাইভ (DELETE routes; মডেল যেখানে `archived_at` soft-delete করে)

একটি key কীভাবে "granted" হয় (যাচাই: `PermissionEngine`)

1. `developer` user-type (`bypass_user_types` — `config/platform.php`-এ ডিফল্ট) অথবা বর্তমান কোম্পানিতে system role slug `administrator` (`bypass_role_slugs`) → **সব** active permission। 2. অন্যথায় granted set = বর্তমান কোম্পানির Role থেকে আসা key (`role_permissions`) ∪ per-user **allow** overrides − per-user **deny** overrides (deny-ই জেতে)। 3. Permission রেকর্ডটি নিজেও `status = active` হতে হবে।

Approval wiring (যাচাইকৃত)

`config/actions.php`-এ Configuration-এর `create`, `update`, `delete`-কে **requires_approval** চিহ্নিত করা হয়। `ConfigurationApprovalSeeder` একটি **"Configuration Create Approval"** workflow seed করে (একটি parallel ধাপ, resolver_type `role`, Administrator role, completion rule `any`) এবং demo কোম্পানির জন্য `approvalEnabled = false` ডিফল্ট। অর্থাৎ approval ব্যবস্থা আছে ও `configuration.create` তাতে নিবন্ধিত, তবে demo পরিবেশে ডিফল্টভাবে request আটকানো হয় না।

কে ব্যবহার করবে (demo seeder থেকে যাচাইকৃত)

- `administrator` role / `developer` ইউজার → সবকিছুতে **full access** (bypass)।
- Demo role **`configuration-staff`** (`ConfigurationRoleSeeder`) → শুধু `configuration.view`

(এবং `employee.update-own-profile`) — অর্থাৎ শুধু read-only picker access। - অন্য যেকোনো Role-কে Platform role-matrix স্ক্রিনে প্রয়োজনীয় `configuration.*` key দিতে হয়; Route ফাইলে কোনো Role-তালিকা হার্ড-কোড করা নেই।

কোম্পানি (tenant) isolation (যাচাইকৃত)

`SetCompanyContext` `X-Company-Id` header থেকে কোম্পানি resolves করে, না থাকলে ইউজারের default membership ব্যবহার করে। Non-platform ইউজার যেসব কোম্পানির সদস্য—শুধু সেগুলোতেই সীমাবদ্ধ; platform-scoped ইউজার (developer/super-admin) সব **active** কোম্পানিতে কাজ করতে পারে। কোম্পানি resolved না হলে tenant-scoped Controller-গুলো `PermissionDeniedException('company.context')` ছোঁড়ে — ডেটা কখনো অন্যের টেন্যান্ট থেকে পড়া হয় না।

---

৭. অন্য মডিউলের সাথে সম্পর্ক (Dependencies)

- **Employee মডিউল** — Designation, Department, Employment type, Salary structure, Document type, Grade, Team, Division, Branch, Section, Cost center, Work location ব্যবহার করে (picker-এর মাধ্যমে)।
- **Payroll মডিউল** — Salary structure-এর write কাজ নিজে করে (`payroll.structure-manage`); এই মডিউল শুধু read/reference রাখে।
- **Onboarding মডিউল** — নতুন কর্মীর জন্য gate; `onboarding.access` / `onboarding.document` Middleware এখানে গ্যালা।
- **Auth / Platform** — `auth:api`, Role ও `permission` Middleware সরবরাহ করে।
- **Tenancy লেয়ার** (`App\Core\Tenancy\TenantContext`) — প্রতি Endpoint-এর "বর্তমান কোম্পানি" নির্ধারণ।
- **Core Companies** — Company মডেল যার সাথে Department, Grade, Team ইত্যাদি যুক্ত।

---

৮. একটি সাধারণ ব্যবসায়িক workflow

Scenario: "বিদ্যমান কোম্পানিতে নতুন কর্মী নিয়োগ":

1. Admin **System Configuration** খুলে নিশ্চিত করে **কোম্পানি active** আছে। 2. Admin/HR নিশ্চিত করে প্রয়োজনীয় **masters** আছে ও active: Employment type (Full-time), Designation (Software Engineer), Grade, Team, Division, Branch, এবং প্রাসঙ্গিক **Department** (এর head-সহ), **Section**, **Work location**, **Cost center**। 3. প্রয়োজন হলে HR একটি **Document type** তৈরি করে (কর্মীর ডকুমেন্ট ফোল্ডারের জন্য)। 4. Front-end-এর picker ফর্ম **active** Endpoint ডাকে (যেমন `/sections/active`) — ফলে শুধু বৈধ অপশন দেখায়। 5. Payroll কর্মীর জন্য প্যাকেজ ঠিক করার সময় Payroll মডিউল থেকে salary structure বাছাই করে; Configuration-ই একক reference। 6. কর্মী চলে গেলে master-এ `DELETE` করলে soft-archive হয় — ইতিহাসে নাম থেকে যায়।

---

৯. সিস্টেম/ব্যবসায়িক প্রভাব

- **একক তথ্যের উৎস (Single source of truth)** — প্রতিটি ধারণার একটি মাত্র কোম্পানি-নির্ভর master, সব মডিউল তা reuse করে।
- **Tenant isolation** — প্রতিটি অবজেক্ট বর্তমান কোম্পানির মধ্যে সীমাবদ্ধ, তাই ভিন্ন ভিন্ন কোম্পানির ডেটা কখনো মেশে না।
- **নিরাপদ মুছে ফেলা** — বেশিরভাগ org master soft-archive হয়, ফলে পুরোনো কর্মী-বরাদ্দের নাম সংরক্ষিত থাকে।
- **দায়িত্বের স্বচ্ছতা** — Salary write Payroll করে, Configuration শুধু reference/read রাখে — দ্বৈত রক্ষণাবেক্ষণ নেই।

---

১০. সাধারণ সমস্যা (এবং সাধারণত যা বোঝায়)

যা দেখেন	সম্ভাব্য কারণ
401	লগইন নেই / `auth:api` token সঠিক নয়
403	Onboarding অসম্পূর্ণ, document deadline মেয়াদোত্তীর্ণ, অথবা Role-এ `permission:*` নেই (এবং ইউজার `developer`/`administrator` নয়, যারা bypass পায়)
Department তালিকা খালি	কোম্পানি context নেই, অথবা কোম্পানি **active** নয়
ফর্মে অপশন বাছাই করা যাচ্ছে না	কোনো **active** master নেই; picker-এ `active` Endpoint ব্যবহৃত হয়
এই মডিউলে salary structure তৈরি করা যাচ্ছে না	Write Payroll-এ স্থানান্তরিত (`payroll.structure-manage`) — এই Route ফাইলে নেই
Department ডিলিট হচ্ছে না	সিস্টেম child department যুক্ত মারাত বা active কর্মী আছে এমন department ডিলিট আটকে দেয়

---

১১. সংক্ষেপে

Configuration মডিউল হলো পুরো ERP-এর **প্রশাসনিক রেফারেন্স কেন্দ্র** — এখানেই Company, Department এবং সব HR/সংগঠন-সংক্রান্ত master (Employment type, Salary structure, Document type, Designation, Grade, Team, Division, Branch, Section, Cost center, Work location ও Employee ID card settings) তৈরি ও রক্ষণাবেক্ষণ হয়, যা বাকি প্রতিটি মডিউল ব্যবহার করে। যেহেতু ডেটা কোম্পানি-নির্ভর এবং layered Permission আর onboarding gate-এ সুরক্ষিত — তাই শুধু অনুমোদিত লোকই এটি বদলাতে পারবে; কিন্তু একবার সেট হলে সারা সিস্টেমের সব ফর্ম ও কর্মী সম্পর্কিত কাজে একই পরিষ্কার ও সঙ্গত মান প্রবাহিত হয়।

`backend/Modules/Configuration/routes/api.php` থেকে সংকলিত। ব্যবসায়িক ভাষার গাইড; যথার্থতার জন্য Controller, Service, Model ও Middleware পর্যালোচনা করা হয়েছে।