AP Bill Worker – Gemini Parameters & Parsing Logic Reference

Reference for Gemini parameters, extraction schemas, and all parsing logic used in this project.

Gemini Configuration

Env Variable	Default	Description
`GEMINI_API_KEY`	*(required)*	Google AI API key
`GEMINI_MODEL`	`gemini-3.1-pro-preview`	Primary model for both passes
`GEMINI_FALLBACK_MODEL`	`gemini-2.5-pro`	Stable fallback if primary fails

Both passes use **structured JSON output** (`responseMimeType: "application/json"` + `responseSchema`).

Pass 1: Invoice Extraction

Function: `extractInvoiceWithGemini(ocrText, config, attachment)`

Input: OCR text + optional image bytes (for multi-modal accuracy).

Output schema – all fields below are extracted in a single Gemini call:

vendor

Field	Type	Description
`name`	string	Best-guess vendor/seller name
`confidence`	number	0–1 confidence score
`source`	string	Where the name was found: `header`, `body`, `atp_printer_box`, `unknown`

vendor_candidates

Array of up to 5 alternative vendor guesses, each with `name`, `confidence`, `source`.

vendor_details

Field	Type	Description
`tin`	string	Tax Identification Number (PH format)
`branch_code`	string	Branch code if present
`address`	string	Vendor address
`entity_type`	string	`corporation`, `sole_proprietor`, `individual`, `unknown`

Field	Type	Description
`trade_name`	string	Business/DBA name (e.g. "JORJEL LAUNDRY SHOP")
`proprietor_name`	string	Owner's personal name for sole proprietors (e.g. "JOCELYN E. SANTOS")

Entity type detection rules:

- `corporation` – name ends with Inc., Corp., Co., LLC, etc.
- `sole_proprietor` – both a trade name AND a personal owner name present (look for "Prop.", "Owner:")
- `individual` – vendor is clearly a person with no business name
- `unknown` – cannot determine

invoice

Field	Type	Description
`number`	string	Invoice/receipt number
`date`	string	Date in `YYYY-MM-DD` format
`date_confidence`	number	0–1
`currency`	string	ISO currency code (e.g. "PHP", "USD")

vat

Field	Type	Description
`classification`	string	`vatable`, `exempt`, `zero_rated`, `unknown`
`goods_or_services`	string	`goods`, `services`, `unknown`
`vatable_base`	number	Net amount before VAT
`vat_amount`	number	VAT amount
`exempt_amount`	number	VAT-exempt amount
`zero_rated_amount`	number	Zero-rated amount
`evidence`	string	Text snippet supporting the classification

VAT classification rules:

- `exempt` – receipt shows "VAT Exempt" or VAT-exempt amount
- `zero_rated` – receipt shows "Zero Rated", "0% ZR"
- `vatable` – shows VAT amount, "VAT Sales", "Vatable Sales", or 12% VAT
- `unknown` – none of the above

totals

Field	Type	Description

Field	Type	Description
`grand_total`	number	Final amount due (VAT-inclusive if applicable)
`net_total`	number	Amount BEFORE VAT. Computed as `grand_total / 1.12` if only inclusive total shown
`tax_total`	number	Tax amount
`amounts_are_vat_inclusive`	boolean	`true` if prices already include 12% VAT
`vat_exempt_amount`	number	Exempt portion
`zero_rated_amount`	number	Zero-rated portion

All totals have accompanying `*_confidence` fields (0–1).

line_items

Array of extracted line items:

Field	Type	Description
`description`	string	Item description
`quantity`	number	Qty
`unit_price`	number	Price per unit
`amount`	number	Line total
`unit_price_includes_vat`	boolean	Whether unit price is VAT-inclusive
`expense_category`	string	One of: `office_supplies`, `meals`, `repairs`, `rent`, `fuel`, `professional_fees`, `freight`, `utilities`, `inventory`, `other`

expense_account_hint

Field	Type	Description
`category`	string	Best-fit category (same options as line_items)
`suggested_account_name`	string	Human-friendly account name guess
`confidence`	number	0–1
`evidence`	string	Supporting snippet

amount_candidates

Array of all notable amounts found, each with `label`, `amount`, `confidence`, `snippet`.

warnings

Array of strings flagging potential issues (e.g. "Multiple totals found", "Low confidence vendor match").

PH-Specific Prompt Rules

- **ATP/Printer box exclusion**: Names found near "ATP", "BIR Permit", "Printer", "Accreditation" are excluded as vendor candidates.
- **VAT-inclusive detection**: Most PH receipts show prices including 12% VAT. The prompt instructs Gemini to detect this and compute `net_total = grand_total / 1.12`.
- **Sole proprietor detection**: Gemini is instructed to look for "Prop.", "Owner", or personal names near trade names to distinguish sole proprietors from corporations.

Pass 2: Account Assignment

Function: `assignAccountsWithGemini(extracted, expenseAccounts, config, industry)`

Input:

- Extracted data from Pass 1
- Full chart of expense accounts from Odoo (`account.account` where `account_type in [expense, expense_direct_cost]`)
- Company industry (auto-resolved from `res.company.x_studio_industry`)

Output schema:

Field	Type	Description
`assignments`	array	Per-line account picks
`assignments[].line_index`	number	0-based index into line_items
`assignments[].account_id`	number	Matched Odoo account ID
`assignments[].confidence`	number	0–1
`assignments[].reasoning`	string	Why this account was chosen
`bill_level_account_id`	number	Best single account for the whole bill
`bill_level_confidence`	number	0–1

Prompt Rules

1. **Specificity over generality**: NEVER pick "Admin Expense", "Miscellaneous", "General Expense" unless no specific account exists.
2. **Match by item description, not vendor**: What was bought matters, not who sold it.
3. **Industry-aware COGS vs OpEx**: If company industry is known, items directly used in production go to Cost of Revenue; back-office items go to Operating Expense.

Item-to-Account Examples (in prompt)

Item	Account
TABLE CLOTH	Supplies / Housekeeping Supplies

Item	Account
LPG REFILL 11KG	Fuel & Oil / Gas & Oil
BOND PAPER A4	Office Supplies / Stationery
TONER CARTRIDGE	Office Supplies / Printing
ELECTRICITY BILL	Utilities / Power & Light
WATER BILL	Utilities / Water
INTERNET	Communication / Telecom
JANITORIAL SUPPLIES	Janitorial / Cleaning Supplies
FOOD / MEALS	Meals & Entertainment
LEGAL FEES	Professional Fees
SHIPPING / DELIVERY	Freight / Shipping
RENT / LEASE	Rent Expense
GASOLINE / DIESEL	Fuel & Oil / Transportation
COURIER / GRAB	Freight / Delivery
PACKAGING / BOXES	Packaging Supplies (or COGS)
FABRIC / CLOTH	Raw Materials (or COGS for mfg)
DETERGENT / BLEACH	Janitorial (or COGS for laundry)
UNIFORM / WORKWEAR	Uniforms / Employee Benefits

Industry-Based COGS vs OpEx Examples

Industry	COGS Items	OpEx Items
Restaurant/food	Ingredients, condiments, packaging	Cleaning supplies, office supplies
Retail/trading	Merchandise, store supplies, bags	Office supplies
Manufacturing	Raw materials, factory supplies	Office supplies
Services/consulting	Subcontractor fees, project materials	Office rent, admin supplies
Laundry	Detergent, fabric softener	Store signage
Construction	Cement, lumber, rebar	Office supplies

Confidence Scale

Score	Meaning
0.9–1.0	Account name directly matches item

Score	Meaning
0.7–0.9	Clearly the right category
0.5–0.7	Reasonable guess, multiple could fit
0.3–0.5	Generic fallback
< 0.3	Wild guess

Vendor Resolution

Search Order (`findVendor`)

1. Search Odoo `res.partner` by **primary vendor name** (from `vendor.name`)
2. Search by **trade name** (from `vendor_details.trade_name`)
3. Search by **proprietor name** (from `vendor_details.proprietor_name`)

All searches: `name ilike <value>` + `supplier_rank > 0`.

Auto-Creation (`createVendorIfMissing`)

Conditions:

- Vendor confidence >= 0.9
- Not an ATP/printer vendor
- No existing match found (searches by name, trade name, and proprietor name)

Sole proprietor handling:

- If `entity_type` is `sole_proprietor` or `individual` AND `proprietor_name` is present → Odoo vendor is created with the **proprietor's personal name** (not the trade name)
- Trade name goes into `comment` field
- Sets `company_type: "person"` (falls back to `is_company: false` if field doesn't exist)

Corporation handling:

- Vendor created with the business name
- Sets `company_type: "company"`

Fields written:

- `name`, `supplier_rank: 1`, `street` (from address), `vat` (from TIN), `comment` (trade name / proprietor notes)

Tax & VAT Logic

Tax ID Selection (`pickTaxIds`)

Based on `vat.classification` from Gemini:

- `exempt`, `zero_rated`, `unknown` → **no tax IDs** (empty array)

- `vatable` → picks tax IDs from routing config based on `goods_or_services`

Tax ID sources (from routing sheet, auto-resolved from Odoo):

- `vat_purchase_tax_id_goods` – for goods purchases
- `vat_purchase_tax_id_services` – for service purchases
- `vat_purchase_tax_id_generic` – fallback

Tax Auto-Resolution (`pickVatTaxesForCompany`)

Queries `account.tax` where `type_tax_use = purchase` and 10% < amount < 14% (targets 12% PH VAT).

Filters out:

- Capital goods taxes (names containing "capital", "asset")
- Only picks regular goods/services/generic VAT

Tax Metadata (`getTaxMeta`)

Reads from Odoo for the selected tax ID:

- `price_include` – whether Odoo expects VAT-inclusive prices
- `amount` – the tax rate percentage

Price Adjustment (VAT-Inclusive)

Function: `adjustPriceForTax(price, invoiceVatInclusive, taxPriceInclude, taxRate)`

Handles the mismatch between what the invoice shows and what Odoo expects:

Invoice Price	Odoo Tax Config	Action
VAT-inclusive	`price_include = true`	No adjustment (pass-through)
VAT-inclusive	`price_include = false`	Divide by `(1 + rate/100)` to get net
VAT-exclusive	`price_include = true`	Multiply by `(1 + rate/100)` to get gross
VAT-exclusive	`price_include = false`	No adjustment (pass-through)

Currency Resolution

Function: `resolveCurrencyId(odoo, companyId, currencyCode)`

- If Gemini extracts a currency (e.g. "USD", "PHP"), searches `res.currency` in Odoo
- Returns the Odoo `currency_id` to set on the bill
- Returns 0 (Odoo default) if not found or empty

Expense Account Cascade

Function: `resolveExpenseAccountId(...)` – 5-tier resolution:

Tier 1: Vendor Default

Read `property_account_expense_id` from the matched `res.partner`. If set, use it immediately.

Tier 2: Gemini Pass 2

Use the `account_id` from `assignAccountsWithGemini` output. Validated against the real chart of accounts (must exist in `loadExpenseAccounts` result).

Tier 3: Sheet Mapping (AccountMapping tab)

Lookup in the `AccountMapping` Google Sheet by `category` + `company_id` + `target_db`. Falls back to company-only match, then global default.

AccountMapping columns: `target_db`, `company_id`, `category`, `account_id`, `account_name`

Tier 4: Fuzzy Name Match

Matches the line item description and category keywords against account names in the chart of accounts.

Scoring:

- Each matching token scores by length (specific words) or 1 point (generic words)
- Generic words penalized: "expense", "admin", "general", "miscellaneous", "other", "sundry"
- Accounts where >50% of name is generic get a 60% score penalty
- Minimum score of 4 required

Category keyword boost: Each category has associated keywords (e.g. `fuel` → "fuel", "gas", "oil", "lpg", "diesel", "petroleum", "gasoline", "petrol") that are added to the search tokens.

Tier 5: Environment Fallback

Uses `DEFAULT_EXPENSE_ACCOUNT_ID` from environment variables.

Tier 6: None

Returns `accountId: 0` – Odoo will use its own default.

Bill Construction

Function: `buildBillVals(...)`

Line Items vs Single Line

- If extracted `line_items` exist AND their total is within 5% of the invoice total → use **itemized lines**
- Otherwise → create a **single summary line**

Bill Values

Odoo Field	Source

Odoo Field	Source
`move_type`	Always `"in_invoice"` (vendor bill)
`partner_id`	Matched/created vendor ID
`company_id`	From routing config
`journal_id`	Auto-resolved purchase journal (prefers "Vendor Bills" journal)
`currency_id`	From Gemini currency extraction
`ref`	Invoice number from Gemini
`invoice_date`	Invoice date from Gemini
`invoice_line_ids`	Computed lines with tax, account, price adjustment

Per-Line Fields

Field	Source
`name`	Line item description (max 256 chars)
`quantity`	From extraction
`price_unit`	Adjusted via `adjustPriceForTax()`
`account_id`	From expense account cascade
`tax_ids`	`[[6, 0, taxIds]]` (Odoo many2many write format)

Document Linking

Function: `linkDocumentToBill(odoo, companyId, docId, billId, logger)`

1. Reads original `folder_id` from the document
2. Sets `res_model = "account.move"` and `res_id = billId` (creates Odoo smart button)
3. Sets `account_move_id` / `invoice_id` if those fields exist
4. **Explicitly preserves `folder_id`** – if Odoo automation moves the document, it's restored
5. Posts a clickable link to the document in the bill's chatter

File attachment: `attachFileToBillChatter` posts the original file as a **chatter attachment** (via `message_post` with `attachment_ids`), NOT as a direct `ir.attachment` on the bill. This prevents Odoo from creating a duplicate document in the "Purchases" folder.

Chatter Messages Posted to Bill

1. Source document link
2. Vendor extraction details (name, confidence, entity type, TIN, address, trade name, proprietor)
3. Account suggestions per line (code + name)
4. Extracted amounts (grand total, net total, tax, VAT-inclusive flag, currency)
5. Warnings (if vendor confidence < 0.9 or extraction warnings present)

Reprocessing (Deleted Bills)

How It Works

Each processed document has a **marker** written to `ir.attachment.description`:

```
BILL_OCR_PROCESSED|V1|<target_key>|doc:<doc_id>|bill:<bill_id>|...
```

Reprocess Flow

1. `processOneDocument` reads the marker and extracts the `bill_id`
2. Checks if that bill still exists in Odoo (`account.move` search)
3. If **bill was deleted**: clears the marker from the attachment, then reprocesses from scratch
4. If **bill exists**: skips (already processed)

`runOne` Stale Link Cleanup

When calling `/run-one` with a `doc_id` whose linked bill was deleted:

1. Tries to find the document (with fallback to broader search + archived docs)
2. If `res_model` points to a deleted `account.move`, clears `res_model`, `res_id`, `account_move_id`, `invoice_id`
3. Proceeds with normal processing

Batch `/run` Revisit

The batch run processes documents in two phases:

1. **New docs** (ID > last checkpoint) – normal forward processing
2. **Revisit older docs** (ID <= checkpoint) – checks if linked bills still exist, reprocesses if deleted

---

Retry & Fallback

Function: `geminiWithRetryAndFallback(config, body, options)`

Strategy

1. Try **primary model** (`GEMINI_MODEL`)
2. On HTTP 429/500/503: retry up to **2 times** with backoff (3s, 6s)
3. If primary exhausted: try **fallback model** (`GEMINI_FALLBACK_MODEL`) with same retry logic
4. Pass 1 (`extractInvoiceWithGemini`): throws on total failure
5. Pass 2 (`assignAccountsWithGemini`): returns `null` on failure (cascade continues without Gemini)

Retryable Status Codes

Code	Meaning
429	Rate limited

Code	Meaning
500	Server error
503	High demand / overloaded

Non-retryable errors (400, 401, 403, 404) fail immediately without retry.

Auto-Resolved Routing Fields

These fields in the ProjectRouting sheet are **auto-populated** from Odoo on each run:

Column	Source	Odoo Model/Field
`vat_purchase_tax_id_goods`	12% VAT for goods	`account.tax`
`vat_purchase_tax_id_services`	12% VAT for services	`account.tax`
`vat_purchase_tax_id_generic`	Generic 12% VAT	`account.tax`
`purchase_journal_id`	"Vendor Bills" journal	`account.journal` (type=purchase)
`ap_folder_id`	AP folder in Documents	`documents.folder`
`industry`	Company industry	`res.company.x_studio_industry`

These are refreshed every run and written back to the Google Sheet.