Calculation Formulas (Car Import Cost Calculator)

Language: RU (primary), brief EN notes inline for clarity.

1. Область применения / Scope

Расчет полной ориентировочной стоимости ввоза легкового автомобиля (категория M1, физическое лицо) из стран: Япония, Корея, ОАЭ, Китай. Формулы отражают текущие конфигурационные данные (YAML). Все ставки/тарифы могут обновляться через конфиги без изменения кода.

2. Входные данные (Inputs)

Поле	Описание
country	"japan"
year	Год выпуска ТС
engine_cc	Объем двигателя (см³) > 0
purchase_price	Цена закупки в валюте ввода
currency	Валюта цены закупки (ISO код)
freight_type	Тип фрахта (country-specific)
sanctions_unknown	Флаг (не используется в текущей формуле)

3. Возраст и возрастная категория

```
age_years = current_year - year
if age_years < 3 -> age_category = "lt3"
elif 3 <= age_years <= 5 -> age_category = "3_5"
else -> age_category = "gt5"
```

4. Проходная / непроходная (passing_category)

```
if age_category == "3_5": passing_category = "passing" else "non_passing"
```

(Используется для UI, не влияет на формулы прямо.)

5. Конвертация валют

Курсы объединяются: статические + (при включении) live ЦБ РФ.

```
rate(code) = currencies[ f"{code}_RUB" ]  (Decimal)
purchase_price_rub = purchase_price * rate(currency)
```

Все промежуточные значения держим в Decimal (precision=28), итоговые поля округляем до целых RUB (ROUND_HALF_UP).

6. Пошлина (Duty)

6.1. Возраст < 3 лет (lt3)

Имеется шкала value_brackets:

```
customs_value_eur = purchase_price_rub / EUR_RUB
Для каждого brackets (ordered):
  if customs_value_eur <= max_customs_value_eur (или последний без max):
     percent = bracket.percent
     min_rate = bracket.min_rate_eur_per_cc
     break

# Две альтернативы:
duty_eur_percent = customs_value_eur * percent
duty_eur_min = engine_cc * min_rate
if duty_eur_percent >= duty_eur_min:
    duty_mode = "percent"
    duty_eur = duty_eur_percent
else:
    duty_mode = "min"
    duty_eur = duty_eur_min

duty_rub = duty_eur * EUR_RUB
```

6.2. Возраст 3–5 (3_5) и >5 (gt5)

Таблица bands: первая запись, где engine_cc <= max_cc (или последняя open-ended):

```
rate_eur_per_cc = band.rate
duty_rub = engine_cc * rate_eur_per_cc * EUR_RUB
duty_mode = "per_cc"
```

7. Утилизационный сбор (Utilization Fee)

Таблица `utilization.personal_m1` (конфиг). Сегменты по объему:

```
key = (age_category == "lt3") ? "lt3" : "ge3"
iterate table in order:
  if segment.max_cc is None or engine_cc <= max_cc:
      utilization_fee_rub = segment[key]
      break
```

Значение уже в RUB, без доп. расчетов.

8. Расходы по стране (Country Expenses)

Япония

Таблица tiers по максимальной закупочной цене (в JPY). Берется первый подходящий tier.

```
if purchase currency != JPY -> warning (используем введенную стоимость как есть для подбора tier)
country_expenses = tier.expenses (JPY)
country_expenses_rub = country_expenses * JPY_RUB
```

Другие страны

```
base_expenses: сумма значений (в country_currency)
country_expenses_rub = sum(base_expenses) * rate(country_currency)
```

9. Фрахт (Freight)

Из конфигурации `fees[country].freight`:

```
if requested freight_type in freight: use it else first item
freight_rub = amount * rate(currency)
```

Если нет записей => 0.

10. Таможенные услуги (customs_services)

Фиксированная сумма по country из `rates.customs_services` (RUB).

11. Оборудование ЭРА-ГЛОНАСС

`era_glonass_rub` фиксированное поле (RUB).

12. Комиссия компании (Company Commission)

```
thresholds: ordered list
for th in thresholds:
    if th.max_price is None or purchase_price_rub <= th.max_price:
        commission_rub = th.amount
        break
```

ВАЖНО: Используем исходную закупочную стоимость в рублях (без duty). Это зафиксировано для предсказуемости и прозрачности.

13. Итоговая сумма

```
total_rub = purchase_price_rub
          + duties_rub
          + utilization_fee_rub
          + customs_services_rub
          + era_glonass_rub
          + freight_rub
          + country_expenses_rub
          + company_commission_rub
```

Все компоненты перед суммированием округляются только в финальном выводе (расчеты внутри в Decimal до общего итогового rounding).

14. Округления / Precision

- Внутри: Decimal, quantize при нужных промежуточных шагах до 4 знаков.
- Вывод: int RUB через ROUND_HALF_UP.
- meta.eur_rate_used хранит строку: "<курс>:<источник>".
- meta.duty_formula_mode: one of {"percent","min","per_cc", None}.

15. Warnings

Код	Случай
WARN_JAPAN_TIER_CURRENCY	Японский расчет, но валюта не JPY
WARN_NO_DUTY_RATE	Не найдено подходящей ставки (конфиг ошибка)

16. Псевдокод Pipeline

```
def calculate(req):
    load configs
    rates = merge(static, live)
    purchase_price_rub = convert(req.purchase_price, req.currency)
    age_category = classify(year)
    duty, duty_mode = compute_duty(age_category, engine_cc, purchase_price_rub, EUR_RUB)
    utilization = utilization_fee(age_category, engine_cc)
    country_exp = country_expenses(country, purchase_price, currency)
    freight = select_freight(country, req.freight_type)
    customs = customs_services[country]
    era = era_glonass
    commission = commission(purchase_price_rub)
    total = sum(all)
    round each to int (RUB)
    return breakdown + meta(duty_mode, eur_rate)
```

17. Примеры

Пример A (lt3, min duty):

Допустим: engine 1800cc, purchase 7,000 EUR, EUR_RUB=100:

```
 purchase_price_rub = 7000 * 100 = 700000
customs_value_eur = 700000 / 100 = 7000
bracket: <=8500 => percent=0.54, min=2.5
percent_branch: 7000 * 0.54 = 3780 EUR
min_branch: 1800 * 2.5 = 4500 EUR -> duty_mode = min
 duty_rub = 4500 * 100 = 450000
```

(Далее добавляются прочие расходы из конфигов.)

Пример B (lt3, percent duty):

engine 1000cc, purchase 80,000 EUR, EUR_RUB=100:

```
 purchase_rub = 8,000,000
customs_value_eur = 80,000
bracket (16,700–42,300) actually: value 80k => bracket 42,300–84,500 -> percent=0.48 min=7.5
percent = 80,000 * 0.48 = 38,400 EUR
min = 1000 * 7.5 = 7,500 EUR
mode=percent -> duty_rub = 38,400 * 100 = 3,840,000
```

18. Добавление новой страны (Checklist)

1. fees.yml: country_currency, base_expenses или tiers, freight.
2. rates.yml: customs_services entry (если отличается).
3. commissions.yml: обычно общий для всех – проверить.
4. tests/test_data/cases.yml: добавить кейс.
5. При необходимости – локальные правила (например особый warning) в engine.

19. Расширение (Future)

- Разные формулы комиссий (от суммы после duty).
- Дополнительные типы ТС (pickup, bus) -> отдельные утиль таблицы.
- Переключение стратегии округления (банковское).

20. Краткие EN Notes

- Duties for <3y: max(percent * customs_value_eur, min_rate * cc)
- For >=3y: cc * band_rate * EUR_RUB
- All amounts Decimal; output ints.

Документ версионируется по commit hash в README (см. Config hash в /api/health).