

StoreFlow — A1.6 API Specification & Endpoint Contract

1. API Overview & Principles

This document defines the API surface for StoreFlow, focusing on:

- Public storefront APIs (used by customer-facing frontends)
- Authenticated dashboard APIs (used by merchant/staff UI via Inertia)
- Internal service endpoints (for AJAX-style JSON actions within dashboard)

Architectural principles:

- Use clean, resource-oriented REST where it makes sense.
- Prefer POST for operations that mutate or compute sensitive state.
- All dashboard calls are authenticated and scoped to `merchant_id` + `store_id`.
- Public APIs are minimal and carefully rate-limited.
- All responses use JSON for machine-readable data; HTML is served via Inertia for full-page loads.

Versioning:

- Public APIs are namespaced under `/api/v1/...`
- Dashboard endpoints may remain unversioned and server-driven initially (as Inertia routes).

2. Authentication & Headers

Authentication Types:

1) Merchant Dashboard (staff/owner/manager)

- Session-based (Laravel 'web' guard)
- CSRF-protected for state-changing requests
- No bearer tokens required for internal dashboard calls

2) Customer Storefront

- Public APIs used for checkout, shipping quote, tracking
- Optional session-based login for customers (separate 'customer' guard)
- Public tracking uses a public_id token, not customer_id

HTTP Headers:

- Content-Type: application/json for API calls
- Accept: application/json for API endpoints
- X-CSRF-TOKEN: required for authenticated POST/PUT/PATCH/DELETE
- Authorization: (reserved for future bearer/token-based APIs)

Error format (JSON):

```
{  
  "message": "Human readable error",  
  "errors": {  
    "field_name": ["Validation error 1", "Validation error 2"]  
  }  
}
```

3. Public Storefront API – Shipping Quote

ENDPOINT: POST /api/v1/stores/{store_id}/shipping/quote

Description:

Calculate available shipping options for a given destination and cart contents.

Request Body (JSON):

```
{
  "country": "AU",
  "state": "VIC",
  "postcode": "3182",
  "items": [
    { "product_id": 101, "quantity": 2 },
    { "product_id": 202, "quantity": 1 }
  ]
}
```

Validation:

- store_id must be a valid, active store.
- All product_ids must belong to the same merchant/store.
- Quantities must be ≥ 1 .

Response 200 (JSON):

```
{
  "options": [
    {
      "id": 5,
      "method_id": 12,
      "label": "Standard Shipping",
      "price_cents": 1299,
      "eta_text": "2-5 business days"
    },
  ],
}
```

```
{  
  "id": 6,  
  "method_id": 13,  
  "label": "Express Shipping",  
  "price_cents": 1999,  
  "eta_text": "1-2 business days"  
}  
]  
}
```

Response 422 (validation error):

- Missing or invalid fields.

Response 409 (no shipping possible):

```
{  
  "message": "Delivery is not available to this postcode for your cart."  
}
```

4. Public Storefront API – Checkout (Order Creation)

ENDPOINT: POST /api/v1/stores/{store_id}/checkout

Description:

Create a new order (pickup or shipping). In v1, payment is treated as successful.

Request Body (JSON):

```
{
  "fulfilment_type": "shipping", // or "pickup"
  "contact": {
    "first_name": "Josh",
    "last_name": "Walkerden",
    "email": "josh@example.com",
    "mobile": "+61412345678"
  },
  "shipping_address": {
    "country": "AU",
    "state": "VIC",
    "postcode": "3182",
    "line1": "Unit 5 107 Westbury Street",
    "line2": "",
    "city": "Balaclava"
  },
  "shipping_method_id": 12, // required for shipping
  "pickup_time": null, // optional for pickup
  "items": [
    {
      "product_id": 101,
      "quantity": 2,
      "customizations": [
```

```
{ "option_id": 301, "quantity": 1 },
{ "option_id": 302, "quantity": 2 }
]
},
"customer_account": {
  "create_account": true,
  "password": "StrongPass123!" // optional; ignored if false
}
```

Server Actions:

- Validate inputs, ensure shipping_method_id is valid for the destination if shipping.
- Resolve or create customer record (guest or registered).
- Calculate items_total_cents from product + options.
- Calculate shipping_cost_cents via shipping engine.
- Compute total_cents.
- Create order, order_items, order_item_options.
- Assign public_id token.
- Queue events and broadcasts.

Response 201 (JSON):

```
{
  "order_id": 555,
  "public_id": "SF-2K9Q3L",
  "status": "pending",
  "total_cents": 3499
}
```

Errors:

- 422 – validation failure.
- 409 – shipping configuration mismatch or unavailable.

5. Public Storefront API – Order Tracking

ENDPOINT: GET /api/v1/orders/{public_id}

Description:

Retrieve public information about an order by its public token.

Response 200 (JSON):

```
{
  "public_id": "SF-2K9Q3L",
  "status": "shipped",
  "fulfilment_type": "shipping",
  "placed_at": "2025-12-02T10:15:00Z",
  "items": [
    {
      "name": "Chicken Burger",
      "quantity": 2,
      "total_cents": 1999
    }
  ],
  "totals": {
    "items_total_cents": 2999,
    "shipping_cost_cents": 500,
    "discount_cents": 0,
    "tax_cents": 0,
    "total_cents": 3499
  },
  "shipping": {
    "shipping_method": "Standard Shipping",
    "shipping_status": "shipped",
    "tracking_code": "ABC123456",
    "tracking_url": "https://carrier.example/track/ABC123456"
```

```
},  
"store": {  
  "name": "Josh's Store",  
  "contact_phone": "+61...",  
  "address": "..."  
}  
}
```

Security:

- This endpoint is read-only.
- No customer_id or internal IDs exposed.
- Throttled per IP + per public_id to mitigate abuse.

Errors:

- 404 – no order with that public_id.

6. Public Storefront API – Customer Login & Registration (Optional)

These endpoints are optional early on but useful for loyalty and account-based UX.

ENDPOINT: POST /api/v1/customers/login

Body:

```
{  
  "merchant_slug": "storeflow-merchant-a",  
  "email": "josh@example.com",  
  "password": "StrongPass123!"  
}
```

Response:

- 200 on success, sets customer session cookie.
- 422 on validation errors.
- 401 on invalid credentials.

ENDPOINT: POST /api/v1/customers/register

Body:

```
{  
  "merchant_slug": "storeflow-merchant-a",  
  "first_name": "...",  
  "last_name": "...",  
  "email": "...",  
  "mobile": "...",  
  "password": "..."  
}
```

Behavior:

- Creates a customers row with password_hash.
- Optionally links prior guest orders by email/mobile.

Note:

- Exact implementation can be deferred; these contracts are reserved so Claude can wire them in later without breaking changes.

7. Dashboard API – Orders (Operations Panel)

While the dashboard primarily uses Inertia for page loads, many operations are AJAX-like JSON calls.

ENDPOINT: GET /dashboard/orders/active

Auth: Merchant session (owner/manager/staff)

Query Parameters:

- status[] = pending,accepted,... (optional)
- fulfilment_type = pickup|shipping (optional)

Response 200 (JSON):

```
{
  "orders": [
    {
      "id": 555,
      "public_id": "SF-2K9Q3L",
      "status": "accepted",
      "fulfilment_type": "pickup",
      "customer_name": "Josh Walkerden",
      "placed_at": "2025-12-02T10:15:00Z",
      "total_cents": 3499
    },
    ...
  ]
}
```

ENDPOINT: PATCH /dashboard/orders/{id}/status

Body:

```
{
  "status": "ready",
  "tracking_code": null,
  "tracking_url": null
}
```

```
}
```

Behavior:

- Validates allowed status transitions.
- Updates timestamps accordingly.
- Emits OrderStatusUpdated (+ ShippingStatusUpdated if relevant).
- Writes audit log.

Response 200:

```
{  
  "id": 555,  
  "status": "ready"  
}
```

Errors:

- 403 if user not authorized for this store.
- 409 if invalid state transition.

8. Dashboard API – Products & Customizations

ENDPOINT: GET /dashboard/products

Auth: owner/manager for store

Query:

- search
- is_active
- page, per_page

Response:

- Paginated product list with minimal fields.

ENDPOINT: POST /dashboard/products

Body:

```
{
  "name": "...",
  "description": "...",
  "price_cents": 1299,
  "is_active": true,
  "is_shippable": true,
  "weight_grams": 200,
  "length_cm": 0,
  "width_cm": 0,
  "height_cm": 0,
  "customization_groups": [
    {
      "name": "Size",
      "min_select": 1,
      "max_select": 1,
      "required": true,
      "options": [
        { "name": "Small", "price_delta_cents": 0 },
```

```
{ "name": "Large", "price_delta_cents": 200 }  
]  
}  
]  
}
```

Behavior:

- Creates product and nested customization records in a transaction.
- Emits audit log.

PATCH /dashboard/products/{id}

- Same payload subset for partial updates.

DELETE /dashboard/products/{id}

- Typically soft-delete; enforced in business logic.

9. Dashboard API – Shipping Configuration

ENDPOINT: GET /dashboard/shipping/zones

- Returns all zones for current store.

ENDPOINT: POST /dashboard/shipping/zones

Body:

```
{  
  "name": "Metro Melbourne",  
  "country": "AU",  
  "state": "VIC",  
  "postcode_from": 3000,  
  "postcode_to": 3207  
}
```

ENDPOINT: POST /dashboard/shipping/methods

Body:

```
{  
  "name": "Standard",  
  "type": "weight",  
  "is_active": true  
}
```

ENDPOINT: POST /dashboard/shipping/rates

Body:

```
{  
  "method_id": 12,  
  "zone_id": 3,  
  "min_weight_grams": 0,  
  "max_weight_grams": 5000,  
  "min_cart_total_cents": null,  
  "max_cart_total_cents": null,  
  "base_price_cents": 999,  
}
```

```
"price_per_kg_cents": 300
```

```
}
```

All endpoints:

- Require manager/owner role.
- Log changes in audit_logs.

10. Dashboard API – Loyalty, Customers & Audit Logs

LOYALTY CONFIG:

ENDPOINT: GET /dashboard/loyalty

ENDPOINT: POST /dashboard/loyalty

Body:

```
{  
  "points_per_dollar": 1,  
  "threshold": 1000,  
  "reward_json": { "type": "fixed_discount", "amount_cents": 1000 }  
}
```

CUSTOMERS:

ENDPOINT: GET /dashboard/customers

Query:

- search (by name/email/mobile)
- page, per_page

Response:

```
{  
  "data": [  
    {  
      "id": 123,  
      "first_name": "Josh",  
      "last_name": "Walkerden",  
      "email": "josh@example.com",  
      "mobile": "+614...",  
      "orders_count": 5,  
      "points_balance": 2300  
    }  
  ]  
}
```

```
}
```

AUDIT LOGS:

ENDPOINT: GET /dashboard/audit-logs

Query:

- entity (optional)
- user_id (optional)
- date_from, date_to (optional)

Response:

```
{  
  "data": [  
    {  
      "id": 1,  
      "entity": "order",  
      "entity_id": 555,  
      "action": "status_changed",  
      "meta_json": { "from_status": "pending", "to_status": "accepted" },  
      "created_at": "2025-12-02T10:20:00Z",  
      "user": { "id": 10, "username": "manager1" }  
    }  
  ]  
}
```

11. Error Handling, Status Codes & Conventions

Status Code Conventions:

- 200 OK – Successful GET/PUT/PATCH/DELETE
- 201 Created – Successful resource creation (POST)
- 204 No Content – Successful delete without body
- 400 Bad Request – Malformed payload
- 401 Unauthorized – Not authenticated
- 403 Forbidden – Authenticated but not authorized
- 404 Not Found – Resource not accessible in current tenant context
- 409 Conflict – State conflict (e.g., invalid status transition, shipping unavailable)
- 422 Unprocessable Entity – Validation error

Validation Response Example (422):

```
{  
  "message": "The given data was invalid.",  
  "errors": {  
    "email": ["The email field is required."],  
    "items.0.quantity": ["The quantity must be at least 1."]   
  }  
}
```

Conflict Example (409):

```
{  
  "message": "Order cannot transition from ready to pending."  
}
```

All errors should be deterministic and documented so that frontends, including Claude-generated code, can react predictably.