Universal Commerce Resolver – Refund Flow Walkthrough

1. Customer Message Arrives

“Hi, my order #1025 arrived with a broken bowl. Can I get a refund?”

2. Ticket Ingestion

Ticket from Gorgias/Zendesk/Shopify Inbox stored in Supabase.

3. Ticket Parser → Task Object

Extracts intent and entities:
{intent: 'refund_request', entities: {order_id: '1025', issue_type: 'damaged_item', refund_reason: 'broken bowl'}, confidence: 0.96}

4. Playbook Selection

Playbook 'refund_order' selected with steps: get_order_status → check_eligibility → create_return_label → issue_refund → send_email → close_ticket.

5. Tool Execution (via Connector Layer)

Sequential execution using connectors (Shopify, Shippo, etc.)
- get_order_status → delivered
- check_eligibility → eligible
- create_return_label → Shippo label generated
- issue_refund → success
- send_email → confirmation sent
- verify_refund → processed

6. Human-in-the-Loop

If risk > 7 (e.g., refund > $500), system pauses for approval before proceeding.

7. Logs & Analytics

Every step recorded for traceability and learning.
Example:
12:00:03 tool_call get_order_status delivered
12:00:06 issue_refund success

8. Why It’s Revolutionary

Understands, plans, acts, verifies, and learns autonomously. Closed-loop support automation.

9. Expandability

Same architecture works for WooCommerce, Stripe, Amazon FBA, or ERP systems – just add connectors.

■ Summary: Refund Flow Blueprint

Layer	Example Component	Purpose
Input	Customer message	Capture issue
Parser	Intent + entities extractor	Convert to structured goal
Planner	Playbook selector	Define steps/tools
Executor	Tool invocations via connectors	Perform actions
Verifier	Post-check	Confirm success
Notifier	Email/SMS	Customer update
Auditor	Event logs	Traceability
Policy Engine	Approval checks	Safety
Learning	Aggregates successful runs	Improves playbooks