Xero ↔ Web App Field Mapping & Data Ownership

Overview

This document defines the **deterministic bidirectional mapping** between Xero entities and our web application’s data models, establishing clear ownership rules to prevent data conflicts and ensure safe synchronization.

---

1. Contacts (Clients & Vendors)

Data Ownership

Primary Owner: Xero
Sync Strategy: Pull-dominant (Xero → App), with controlled push for new records created in app

Field Mapping

Xero Field	App Field (Client)	App Field (Supplier)	Direction	Notes
ContactID	xeroContactId	xeroContactId	BOTH	UUID from Xero, immutable
Name	name	name	PULL	Xero is authoritative for name
ContactNumber	clientNumber	supplierNumber	BOTH	Auto-generated in app if missing
EmailAddress	email	email	PULL	First email from Xero
Phones[].PhoneNumber	phone	phone	PULL	First phone number
Addresses[].AddressLine1-4	address	address	PULL	Concatenated address lines
Addresses[].City	city	city	PULL	
Addresses[].Region	state	state	PULL	
Addresses[].PostalCode	postalCode	postalCode	PULL	
Addresses[].Country	country	country	PULL	Default ‘Singapore’ if empty
ContactPersons[0].FirstName + LastName	contactPerson	contactPerson	PULL	Concatenated from first contact person
IsCustomer	clientType != null	-	PULL	If true, create/update Client record
IsSupplier	-	supplierType != null	PULL	If true, create/update Supplier record
IsCustomer + IsSupplier	BOTH	BOTH	PULL	Create records in both tables

Xero Field	App Field (Client)	App Field (Supplier)	Direction	Notes
AccountNumber	xeroAccountNumber	xeroAccountNumber	PULL	Xero's internal account number
TaxNumber	companyReg	companyReg	BOTH	Can be updated from either side
BankAccountDetails	xeroBankAccountDetails	xeroBankAccountDetails	PULL	Stored as JSON
Website	website	website	PULL	
UpdatedDateUTC	xeroUpdatedDateUTC	xeroUpdatedDateUTC	PULL	For change detection
ContactStatus	xeroContactStatus	xeroContactStatus	PULL	ACTIVE, ARCHIVED, etc.

Additional Stored Fields (JSON)

- `xeroPhones` : Full array of phone objects from Xero
- `xeroAddresses` : Full array of address objects from Xero
- `xeroContactPersons` : Full array of contact person objects from Xero

Contact Classification Logic

```
if (xeroContact.IsCustomer && xeroContact.IsSupplier) {
  // Create/update in BOTH Client AND Supplier tables
  // Keep xeroContactId consistent across both records
} else if (xeroContact.IsCustomer) {
  // Create/update ONLY in Client table
} else if (xeroContact.IsSupplier) {
  // Create/update ONLY in Supplier table
}
```

Edge Cases

1. **Xero contact has no IsCustomer or IsSupplier flags**:
   - Skip import (log as warning)
   - If manually created in app first, do not push to Xero until user confirms classification

2. **Contact exists in app but not in Xero**:
   - Manual push only (requires explicit user action)
   - App assigns clientNumber/supplierNumber following existing pattern

3. **Duplicate detection**:
   - Match by xeroContactId (primary)
   - If no xeroContactId, match by email (case-insensitive)
   - If duplicate email found, log conflict for manual resolution

---

2. Invoices (Client Invoices)

Data Ownership

Primary Owner: Web App (for new invoices created in app)
Secondary Owner: Xero (for invoices created in Xero first)
Sync Strategy: Controlled push (App → Xero) on explicit user action; Pull for payments/status updates

Field Mapping

Xero Field	App Field	Direction	Notes
InvoiceID	xeroInvoiceId	BOTH	UUID from Xero after push
InvoiceNumber	invoiceNumber	PUSH	App generates, Xero stores
Contact.ContactID	Client.xeroContactId	PUSH	Must exist in sync mappings
Type	‘ACCREC’	PUSH	Always Accounts Receivable
Status	status	PULL	Draft, Submitted, Authorised, Paid, Voided
LineAmountTypes	‘Exclusive’ or ‘Inclusive’	PUSH	Based on app tax settings
LineItems[].Description	ClientInvoiceItem.description	PUSH	
LineItems[].Quantity	ClientInvoiceItem.quantity	PUSH	
LineItems[].UnitAmount	ClientInvoiceItem.unitPrice	PUSH	
LineItems[].LineAmount	ClientInvoiceItem.subtotal	PUSH	Quantity × UnitPrice
LineItems[].TaxAmount	ClientInvoiceItem.taxAmount	PUSH	Calculated from taxRate
LineItems[].TaxType	‘GST on Income’ (default)	PUSH	Singapore standard rate
LineItems[].AccountCode	From item category mapping	PUSH	Map ItemCategory to Xero account code
SubTotal	subtotal	BOTH	Sum of line items
TotalTax	taxAmount	BOTH	Sum of tax amounts
Total	totalAmount	BOTH	SubTotal + TotalTax
Date	issueDate	PUSH	Invoice issue date
DueDate	dueDate	PUSH	Payment due date

Xero Field	App Field	Direction	Notes
AmountPaid	-	PULL	Track payment status from Xero
AmountDue	-	PULL	Remaining balance
CurrencyCode	currency	PUSH	Default ‘SGD’
Reference	quotationId (if linked)	PUSH	Link to source quotation
UpdatedDateUTC	lastXeroSync	PULL	For change detection

Item Category → Xero Account Code Mapping

```
const CATEGORY_TO_ACCOUNT_CODE: Record<ItemCategory, string> = {
  SERVICES: '200',      // Sales - Services
  MATERIALS: '210',     // Sales - Materials
  LABOR: '220',         // Sales - Labour
  EQUIPMENT: '230',     // Sales - Equipment Rental
  GENERAL: '200'        // Default to services
}
```

Push Workflow (App → Xero)

1. **Validate invoice**:
   - Must have status = SENT or APPROVED
   - Client must have xeroContactId in sync mappings
   - All line items must be valid

2. **Create invoice in Xero**:
   - Use Xero API to create invoice
   - Store returned InvoiceID in ClientInvoice.xeroInvoiceId
   - Create sync mapping record

3. **Handle errors**:
   - Xero validation errors → return detailed message to user
   - Network errors → retry with exponential backoff
   - Conflict errors → log for manual resolution

Pull Workflow (Xero → App)

1. **Status updates only**:
   - Never overwrite invoice line items from Xero
   - Only update: status, amountPaid, paidDate

2. **Payment reconciliation**:
   - If Xero shows invoice as PAID, update app status to PAID
   - Create Payment record if payment exists in Xero

Edge Cases

1. **Invoice modified in Xero after push**:
   - App wins for line items (do not pull changes)
   - Xero wins for status and payments (always pull)

2. **Invoice deleted in Xero**:
   - Do not delete from app
   - Set isXeroSynced = false, log warning

3. **Duplicate invoice numbers**:
   - Xero enforces uniqueness
   - If push fails due to duplicate, suffix with -R1, -R2, etc.

---

3. Payments

Data Ownership

Primary Owner: Xero
Sync Strategy: Pull-only (Xero → App)

Field Mapping

Xero Field	App Field	Direction	Notes
PaymentID	xeroPaymentId	PULL	UUID from Xero
Invoice.InvoiceID	ClientInvoice.xeroInvoiceId	PULL	Link to invoice
Amount	amount	PULL	Payment amount
Date	paymentDate	PULL	Date payment received
Reference	reference	PULL	Payment reference/notes
CurrencyCode	currency	PULL	Payment currency
Status	status	PULL	Authorised, Deleted

Pull Workflow

1. **Fetch payments for mapped invoices only**:
   - Query Xero for payments where InvoiceID exists in sync mappings
   - Do not import payments for unmapped invoices

2. **Create or update Payment record**:
   - Match by xeroPaymentId

- Update linked invoice status to PAID if fully paid
- Create sync mapping record

3. **Reconciliation logic**:

```
typescript
  if (xeroPayment.Amount >= invoice.totalAmount) {
    invoice.status = 'PAID'
    invoice.paidDate = xeroPayment.Date
  } else {
    invoice.status = 'PARTIAL'
  }
```

Edge Cases

1. **Payment deleted in Xero**:
   - Mark Payment as CANCELLED in app
   - Revert invoice status to SENT or PARTIAL

2. **Multiple payments for one invoice**:
   - Create multiple Payment records
   - Sum all authorised payments to determine invoice status

---

4. Sync Metadata Schema

XeroSyncMapping Table

Tracks the relationship between app records and Xero records.

Field	Type	Description
id	UUID	Primary key
entity_type	ENUM	‘CONTACT’, ‘INVOICE’, ‘PAYMENT’
local_id	UUID	App record ID (Client, Supplier, ClientInvoice, Payment)
xero_id	String	Xero entity ID (ContactID, InvoiceID, PaymentID)
last_synced_at	DateTime	Last successful sync timestamp
sync_direction	ENUM	‘PULL’, ‘PUSH’, ‘BOTH’
change_hash	String	MD5 hash of last synced payload (for change detection)
status	ENUM	‘ACTIVE’, ‘ARCHIVED’, ‘ERROR’, ‘CONFLICT’
last_error	Text	Last error message (nullable)
created_at	DateTime	Record creation timestamp
updated_at	DateTime	Record update timestamp

Indexes

- `idx_entity_local` on (entity_type, local_id)
- `idx_entity_xero` on (entity_type, xero_id)
- `idx_status` on (status)
- `idx_last_synced` on (last_synced_at)

Change Hash Calculation

```
function calculateChangeHash(data: any): string {
  const relevantFields = extractSyncableFields(data)
  const normalized = JSON.stringify(relevantFields, Object.keys(relevantFields).sort()
)
  return crypto.createHash('md5').update(normalized).digest('hex')
}
```

---

5. Conflict Resolution Strategy

Conflict Detection

A conflict occurs when:
1. **Same record modified in both systems** (different change_hash)
2. **Duplicate records** (same email/name but different IDs)
3. **Validation errors** (data doesn't meet Xero requirements)

Resolution Flow

1. **Automatic resolution** (where ownership is clear):
   - Contacts: Xero wins (always pull)
   - Invoice line items: App wins (never pull changes)
   - Invoice status/payments: Xero wins (always pull)

2. **Manual resolution** (logged in XeroSyncConflict table):
   - Duplicate contacts with different emails
   - Invoice number conflicts
   - Validation errors requiring data correction

3. **Admin UI for conflict resolution**:
   - Show side-by-side diff of conflicting data
   - Allow admin to choose: "Use Xero", "Use App", "Merge", "Skip"
   - Log resolution decision and who made it

---

6. Sync Safety Rules

Never Sync

- Archived/deleted records (unless explicitly requested)
- Records with status = DRAFT (unless explicitly pushed)
- Incomplete records missing required fields

Always Log

- All sync operations (success, warning, error)
- Change detection (what changed, when, by whom)
- Conflict resolution decisions

Rate Limiting

- Max 60 requests per minute (Xero's limit)
- Implement exponential backoff on 429 errors
- Batch operations where possible (e.g., create 50 contacts at once)

Transaction Safety

- Use database transactions for all sync operations
- Rollback on any error in a batch
- Idempotent operations (re-running doesn't create duplicates)

---

7. Testing Scenarios

Contacts

- [ ] Pull 100 Xero contacts (mix of Customers, Suppliers, Both)
- [ ] Update contact in Xero, verify app updates on next sync
- [ ] Create new contact in app, push to Xero, verify xeroContactId stored
- [ ] Handle contact with no IsCustomer/IsSupplier flags
- [ ] Handle duplicate email addresses

Invoices

- [ ] Push new invoice to Xero, verify InvoiceID returned
- [ ] Update invoice status in Xero, verify app pulls status change
- [ ] Try to push invoice without xeroContactId mapping (should fail gracefully)
- [ ] Push invoice with special characters in line items
- [ ] Handle invoice number conflicts

Payments

- [ ] Pull payment for mapped invoice, verify status updated to PAID
- [ ] Pull partial payment, verify status = PARTIAL
- [ ] Pull multiple payments for one invoice
- [ ] Handle payment deletion in Xero

Edge Cases

- [ ] Xero API returns 429 (rate limit), verify retry logic
- [ ] Token expires mid-sync, verify refresh and resume
- [ ] Network timeout during sync, verify no partial data
- [ ] Large sync (1000+ records), verify performance

---

8. Rollback Plan

Pre-Sync Snapshot

Before any major sync operation:
1. Create database snapshot (pg_dump)
2. Export current sync mappings to JSON
3. Log pre-sync counts (clients, invoices, payments)

Rollback Steps

1. Stop all sync operations
2. Restore database from snapshot
3. Clear sync mappings for affected entity type
4. Re-enable sync after root cause fixed

Verification After Rollback

- Compare record counts before/after
- Verify no orphaned sync mappings

- Check for data integrity issues

Version History

- **v1.0** (2025-10-07): Initial field mapping and ownership rules defined