Q3 — Product Installation Guide

This guide is written for the technician installing VSMS Connect for the first time on a customer site. Follow the steps in order. When every service reports healthy and the application loads to the welcome screen, hand the system over to the business administrator — they take over from Q4 (Configuration Manual) to register the business and connect an accounting provider.

Out of scope — This document does not cover Vanuatu network or hardware procurement, V-SDC certificate procurement, or accounting-provider portal setup (Xero / Sage / MYOB developer apps). Those are pre-requisites the business is expected to arrange before the technician arrives.

1. System requirements

Component	Minimum version	Notes
Node.js	v20 LTS	Used by the backend and consumer services.
Yarn	1.22 (Classic)	Workspace manager — npm is not supported.
Microsoft SQL Server	2019	Used by the services/db microservice for persistent storage.
Redis	6.x	Used for Redis Streams (fiscalisation job dispatch) and rate-
Operating system	Windows 10 / 11	macOS and Linux are also supported.
Browser (web client)	Chromium-based or	SafariRequired for the web build. Mobile native builds use untim

2. Repository layout

The VSMS Connect codebase is a Yarn workspace with four runtime services and three shared packages.

Path	Purpose
apps/app	Expo / React Native client (web + iOS + Android).
services/backend	Express HTTP API — the BFF that the app and external integrations talk to
services/consumer	Fiscalisation job consumer — polls the Redis Stream and submits to V-SDC
services/db	MS SQL microservice — single ingress point for every SQL query.
services/salesAgent	On-premise Sales Agent service.
services/printAgent	Receipt printer agent service.
packages/dtos	Shared DTOs and API contracts.
packages/logger	Shared structured logger (@code200/logger) — git submodule.
packages/utils	Shared helpers (@code200/utils) — git submodule.

3. Preparing the environment

Clone the repository, initialise submodules, install dependencies, and build the shared packages before any service is started:

```
1. git clone <repo>; cd vanuatu.taxcore.vsmsconnect
2. git submodule update --init --recursive
3. yarn install
4. (cd packages/utils && yarn build)
5. (cd packages/logger && yarn build)
6. (cd packages/dtos && yarn build)
```

Build order matters: utils and logger have no internal dependencies, but dtos imports both. Services then resolve to the dist/ folders of each package via tsconfig paths.

4. Configuring environment files

Each service ships an .env.example file next to its package.json. Copy each example to .env and fill in the per-deployment values:

Service	Required values
apps/app/.env	EXPO_PUBLIC_ENVIRONMENT, EXPO_PUBLIC_BACKEND_BASEURL,
services/backend/.env	DB_SERVICE_URL, DB_SERVICE_ACCESS_TOKEN, DB_SERVICE_RE AGENT_BINARIES_DIR — Path to the directory containing Sales Agent ar
services/salesAgent/.env	VSMS_BACKEND_URL, VSMS_BUSINESS_ID, VSMS_API_KEY, VSMS_
services/printAgent/.env	VSMS_BACKEND_URL, VSMS_SETUP_TOKEN (one-time token generat
services/consumer/.env	DB_SERVICE_URL, REDIS_URL, TAXCORE_VSDC_URL, CERT_STORA
services/db/.env	MS SQL connection string, S2S JWT signing keys.

Secrets — Never commit a populated .env file. All .env files are listed in .gitignore. Generate strong, distinct encryption keys for DB_ENCRYPTION_KEY, XERO_TOKEN_ENCRYPTION_KEY, SAGE_TOKEN_ENCRYPTION_KEY, and MYOB_TOKEN_ENCRYPTION_KEY using: `node -e "console.log(require('crypto').randomBytes(32).toString('hex'))"`

5. Bringing up each service

Start the services in the order below so dependencies are available before consumers attach. Each command is run in its own terminal in a fresh dev environment.

1. **Database microservice —** cd services/db && yarn dev. Listens on http://localhost:3233. This is the single ingress point for every SQL query; both the backend and the consumer call it.
2. **Backend API —** cd services/backend && yarn dev. Listens on http://localhost:3000. Migrations run automatically once the database connection succeeds.
3. **Fiscalisation consumer —** cd services/consumer && yarn dev. No HTTP listener; it polls the Redis Stream taxcore:fiscalise and submits jobs to the V-SDC.
4. **Application —** cd apps/app && yarn web (for the web client) or yarn start (for Expo Go on a phone). Listens on http://localhost:8081.

6. Database migrations

Migrations live in services/db/migrations/. They are applied automatically on startup of the db service. Verify by querying sys.tables after the first boot — the expected core tables include

Users, Businesses, Invoices, InvoiceLineItems, InvoicePayments, FiscalisationJobs, AuditLogs, XeroConnections, SageConnections, MyobConnections, TaxRateMappings, and UnmappedTaxTypes.

Locations, Certificates, Tills, PrintJobs, ApiKeys, EmailVerifications, InviteTokens, LicenceKeys, TaxcoreTaxRates, McpTokens — Added in migrations 042–052. Required for multi-location support, receipt printing, on-premise agent auth, and MCP access.

7. Verifying service health

- **Backend —** GET http://localhost:3000/health should return { "status": "ok" } once the DB service is reachable.
- **Consumer —** log line "[fiscalisation-job] consumer started" with the configured Redis stream name.
- **DB microservice —** log line "[db] listening on :3233" plus a successful JWT signing-key load.
- **App —** open http://localhost:8081 in a browser. The Vanuatu Sales Monitoring System welcome screen must render with no console errors.

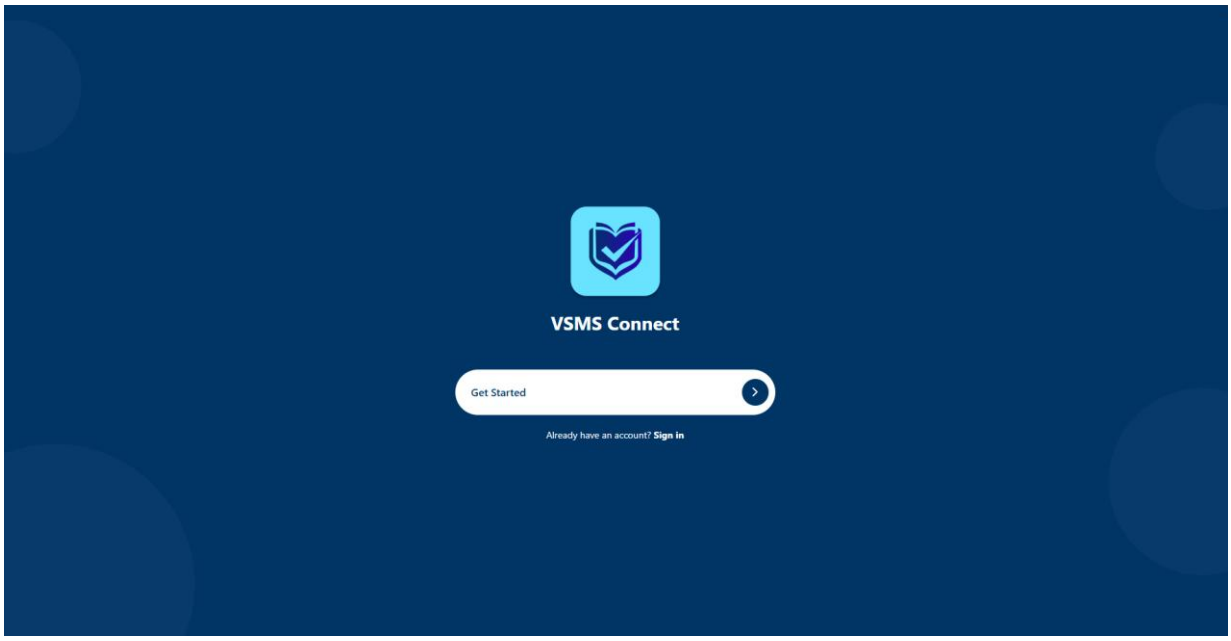

Figure 1 — Final handover check: the welcome screen renders cleanly with no console errors.

8. Handover checklist

- All four services are running and report ready in their logs.
- The browser loads the welcome screen.
- The .env files are committed to the deployment vault, not to git.
- CERT_STORAGE_DIR is on a shared volume readable by both the backend and theconsumer.
- Redis is configured with persistence enabled (appendonly or RDB snapshots) so streamssurvive a restart.
- A backup schedule for MS SQL has been agreed with the customer.

When every item is ticked, hand the system to the business administrator. The next document is Q4 — Configuration Manual.

Additional checklist items for on-premise deployments: salesAgent running and pushing invoices; printAgent running and till paired; agent binaries accessible at /api/v1/agents/; AGENT_BINARIES_DIR env var set in backend.

9. Installing the On-Premise Agents

This section covers installing the Sales Agent and Print Agent on customer machines — separate from the server installation above.

9.1 Sales Agent — Windows

The Sales Agent is a background service installed on the computer running the on-premise accounting software (Amicus, Sage 100, Sage 200 Evolution, Sage 300, or Atrex). Before installing, generate an API Key in the VSMS Connect app (Businesses → Integrations → [provider] → API Keys → Create) and note the Backend URL, Business ID, and API Key.

Installation steps:

1. Download the Sales Agent installer from the VSMS Connect app or from /api/v1/agents/ onthe backend.
2. Run the installer and follow the setup wizard.
3. Select your accounting provider from the dropdown.
4. Enter the database connection details for your local accounting software.
5. Enter the VSMS Connect Backend URL, Business ID, and API Key.
6. Complete the wizard — the agent installs as a background service and starts automatically.

> **Note — Watermark tracking & reliability**
> The Sales Agent uses watermark-based tracking to read only new sales and refunds since its last run, preventing duplicate submissions. It includes automatic retry logic with exponential backoff and local failure logging for troubleshooting stalled syncs.

9.2 Sales Agent — Linux/macOS

Installation steps:

1. Download the Sales Agent binary for Linux/macOS from your VSMS Connect portal.
2. Make it executable: chmod +x salesagent.
3. Run the setup command: ./salesagent setup — enter your Backend URL, API Key, providertype (AMICUS, SAGE_200_EVOLUTION, etc.), and database connection details when prompted.
4. Install as a system service: ./salesagent install. On Linux, this registers a systemd service. On macOS, this registers a launchd service.
5. Start the service: ./salesagent start. The agent connects to your local SQL Server databaseand begins syncing invoices automatically.

9.3 Print Agent — Windows

The Print Agent is a background service installed on the computer connected to the receipt printer at each till. Before installing, generate a Setup Token in the VSMS Connect app (Businesses → Receipt Printers → New Printer → Generate Setup Token) and note the Setup Token (valid for 24 hours) and the Backend URL.

Installation steps:

1. Download the Print Agent installer from the VSMS Connect app.
2. Run the installer.
3. Enter the VSMS Connect Backend URL.
4. Enter the one-time Setup Token.
5. Enter a name for this till (e.g. "Till 1 — Main Counter").
6. Complete the wizard — the agent installs as a background service, claims the setup token,and registers as a named till.

Supported printers: any receipt printer compatible with CUPS (Linux/macOS) or Windows printer drivers. The agent does not require a specific printer brand or model.

9.4 Print Agent — Linux/macOS

Installation steps:

1. Download the Print Agent binary for Linux/macOS from your VSMS Connect portal.
2. Make it executable: chmod +x printagent.
3. Run setup: ./printagent setup — enter your Backend URL and Setup Token when prompted.
4. Name this till (e.g. Till 1 — Main Counter).
5. Install as a service: ./printagent install. The Print Agent uses CUPS for printer discovery onLinux/macOS.
6. Start: ./printagent start. The till appears as Online in the Receipt Printers list.