FedEx | CUSTOM CRITICAL · DATA & ANALYTICS

INTERNAL REFERENCE GUIDE

Legacy System Archival

Do's and Don'ts for Vendors and Project Teams

A field-tested guide for archiving legacy databases, file-based systems, and vendor-managed platforms — distilled from six completed projects across the FedEx and TNT estate.

OWNER
FedEx Custom Critical — Data & Analytics

VERSION
April 2026

SCOPE
All legacy TNT/FedEx systems

AUDIENCE
Vendors and Project Teams

INTERNAL REFERENCE GUIDE

Legacy System Archival — Do's and Don'ts

A practitioner's guide for vendors and project teams.

OWNER
FedEx Custom Critical · Data & Analytics

APPLIES TO
All legacy TNT/FedEx systems

VERSION
April 2026

BASIS
Lessons learned from 6+ completed archival projects

1 Planning & Scoping

✓ DO

- **Inventory the data source before starting.** Count tables, views, stored procedures, and total size. For file-based systems (FoxPro DBF, Firebird FDB), count files, total size, and identify memo/index files. This sets expectations and helps estimate effort.
- **Identify the system type and database technology early.** Each technology has different extraction tools and challenges:
  - **SQL Server** (SEAGHA, CHARISMA, SAGE): use standard SQL tools, pyodbc, SSMS
  - **Firebird** (FINAWIN): may require specialized drivers or raw binary parsing if custom collations block SQL
  - **Visual FoxPro DBF** (TAAVI): use dbfread (Python), handle CP1257/CP1250 Baltic/CEE encodings
  - **Vendor-managed platforms** (MICROSCOP/Mikrocop): no direct database access — must engage vendor
- **Identify the record types in the system** (e.g., Import/Export/Transit declarations, payroll/HR/contracts, invoices). Each type may have its own table structure, views, and output format.
- **Get sample exports early.** Obtain Excel, CSV, or PDF files that represent the expected output format. These are your specification — the extraction must reproduce them exactly. For payroll systems, get sample payslips; for customs, get sample declaration exports.
- **Identify the date field used for filtering.** In customs systems this is typically the acceptance date (`DatAanv`), not the creation or modification date. In payroll systems, it is the pay period. In HR systems, it is the contract date or employment start date.
- **Document data volumes per type and per year upfront.** This determines whether you can export everything at once or need to work in chunks.
- **Check if the database has existing views or stored procedures that produce report formats.** These are your building blocks — do not reinvent them. SEAGHA had 164 views that were essential; FINAWIN had 650 stored procedures; CHARISMA had complex salary calculation procedures.
- **Identify character encoding requirements immediately.** Legacy systems in Europe frequently use non-standard encodings:
  - Latvian: UNICODE_LV custom collation (FINAWIN)
  - Estonian: CP1257 Baltic Windows codepage (TAAVI)
  - Slovenian: EBCDIC mainframe encoding (MICROSCOP)
  - Romanian: Windows-1250 Central European (CHARISMA)
  - Always verify that diacritics and special characters survive extraction.
- **Map the complete data flow.** Understand where data originates, where it is processed, and where it ends up. Some data may only exist at a vendor (MICROSCOP: Mikrocop held the full

10-year archive, not FedEx).

- **Identify regulatory retention periods by country** before starting:
  - Belgium: customs declarations — 7 years
  - Estonia: employment contracts — employment + 10 years; payroll — 7 years
  - Latvia: payroll and personnel records — 75 years for some categories
  - Slovenia: invoices — 10 years
  - Romania: payroll records — 50 years for pension-relevant data
  - Always confirm with local legal/compliance — do not guess.
- **Engage the business stakeholders early.** They know which data matters, which reports they actually use, and what questions auditors ask. They are the specification, not the database schema.

DON'T

- Don't start writing extraction code without a sample output file. You will waste time guessing column order, data transformations, and join logic.
- Don't assume the database schema documentation is complete or current. Always verify against the actual DDL (`CREATE TABLE/VIEW` statements) or the live database structure.
- Don't underestimate the size. A 20 GB SQL Server database can produce 5+ GB of text output. A 2 GB Firebird database with BLOBs can produce hundreds of extracted files. Plan for storage and transfer.
- Don't plan to export the entire database in one query or one pass. Break it down by type, by sheet/report, and optionally by date range.
- Don't assume all data is in-house. Vendor-managed systems (MICROSCOP) may hold the authoritative archive externally. Discover this in planning, not during extraction.
- Don't assume someone on the team knows the system. Legacy system knowledge often resides with departed employees. Document everything from the first conversation.

2 Database Access & Setup

✓ DO

- **Restore the database backup on a separate server** (e.g., DWH/staging), never on the production application server.
- **Verify the data range immediately after restore.** The backup may not contain the most recent years. The SEAGHA backup only had data through 2020 — supplementary Excel exports were needed for 2021–2022. Plan to fill gaps from other sources.
- **Script the full schema** (tables, views, stored procedures) to a `.sql` file early. This becomes your primary reference for understanding the data model. For FINAWIN, this meant extracting 650 stored procedures, 60 functions, and 331 triggers.
- **Use read-only access.** Archival is extraction, not modification.
- **Use `WITH (NOLOCK)` hints** for large read-only queries on SQL Server to avoid blocking.
- **For non-SQL databases, verify driver compatibility before committing to an approach:**
  - Firebird: test that `fdb` / `firebirdsql` Python drivers can read all text columns. Custom collations (UNICODE_LV) can cause all text to return NULL.
  - FoxPro: test that `dbfread` handles the codepage correctly. Check for memo files (`.fpt`) that contain supplementary data.
  - If standard drivers fail, be prepared for alternative extraction methods (raw binary parsing worked for FINAWIN when SQL failed).
- **Preserve the original data files.** Keep the original `.bak` , `.fdb` , `.dbf` files as-is alongside any extracted output. These are your insurance policy.

✕ DON'T

- Don't use `INSERT` , `UPDATE` , `ALTER` , or `DELETE` statements. This is a `SELECT` /extraction project, not data loading.
- Don't modify the source database. Treat it as read-only.
- Don't skip the schema export. Without it, you're navigating blind when the application is decommissioned.
- Don't discard original database files after extraction. Keep them archived alongside the output.
- Don't assume a driver that connects successfully is reading data correctly. Verify text fields contain actual characters, not NULLs or garbled encoding.

3 Extraction Development

✓ DO

- **Use the existing database views as building blocks.** They contain the correct join logic, data transformations, and column aliases that match the report format.
- **Preserve the exact column order** to match the expected output template. Column position matters — the business users know their data by column position.
- **Build parameterized queries** with `@StartDate` / `@EndDate` and `@AllRecords` flag. This gives flexibility for testing (small range) and full extraction.
- **Add a `@TestMode`** with a few known record IDs for verification against sample data.
- **Handle data type conversions carefully:** `ISNUMERIC()` checks before `CAST`, date year checks (≤1970 = NULL), string cleanup (`REPLACE` , `RTRIM`).
- **Handle character encoding explicitly in your extraction scripts:**
  - Specify encoding when opening files: `open(file, encoding='utf-8')` or `encoding='cp1257'`
  - Use a fallback chain: try UTF-8 → CP1257 → CP1250 → Latin-1
  - Verify diacritics manually: check that names like "Jansone-Birsnece" (Latvian) or "Poljakova" (Estonian) appear correctly
  - For Firebird with custom collations: consider raw binary page-level parsing as a fallback
- **Extract BLOBs and binary content separately.** Employee photos, report templates, PDF payslips, XML files — these need special handling:
  - Store BLOBs as individual files in a structured directory (`blobs/table_name/row_id.ext`)
  - Create a metadata table linking BLOB files back to their source rows
  - Identify BLOB types: images, PDFs, XML templates, RTF documents
- **Document each query/script:** what output it produces, how many columns/rows expected, which source tables it reads, what the output filename should be.
- **Test with a small set of known records first.** Compare output column-by-column against the sample export file.

DON'T

- Don't hardcode record ID ranges. Always use date-based filtering through the appropriate date field.
- Don't skip columns, even if they appear empty. The column structure is the contract with downstream consumers.
- Don't confuse column aliases in views with actual database column names (e.g., `StatVal` in Excel may be `StatAdju` in the table).
- Don't ignore version/revision logic. Many systems keep multiple versions of a record — views often select `MAX(SubID) WHERE Function='9'` to get the latest.
- Don't mix header-level and detail-level data. Documents, parties, and financial info may exist at both levels with different join keys.
- Don't ignore credential/password tables. Note their existence but flag them for review — they may contain hashed passwords (SHA-512 in TAAVI) that should be excluded from the public archive.

4 Data Export & Output Formats

✓ **DO**

- **Export to multiple formats based on the audience:**
  - **Tab-delimited text** (`.txt`): most portable intermediate format, works with any tool
  - **Parquet**: compressed columnar format for large datasets (SEAGHA: 22 GB → 650 MB Parquet)
  - **SQLite**: preserves relational structure, queryable, self-contained (FINAWIN, TAAVI)
  - **CSV**: Excel-ready for non-technical users (one CSV per table)
  - **Excel** (`.xlsx`): final delivery format with formatting, auto-filters, freeze panes
- **Use consistent, descriptive filenames:** `Type_SheetName.txt` (e.g., `Import_Main.txt` , `Export_Detail.txt`) or `TableName.csv` .
- **Separate output per sheet/report type:** Main, Detail, Parties, Documents, etc. Each becomes a sheet in the final Excel or a separate CSV.
- **Provide two Excel views when applicable:**
  - `per_year` : all sheets for one year (good for audits of a specific period)
  - `per_type` : all years for one sheet type (good for trend analysis)
  - Users need both depending on their query.
- **Create a `Master_Index.xlsx` or `README.md`** with clickable links to all files and a description of what each contains.
- **Extract and preserve non-database content:**
  - PDF payslips (TAAVI: 1,676 salary slip PDFs)
  - SEPA XML payment files (TAAVI: 43 files)
  - Tax declaration files (TAAVI: 104 files)
  - Report templates (FINAWIN: 225 FastReport XML templates)
  - Stored procedure source code (FINAWIN: 650 procedures)
- **Build a web-based viewer** for non-technical users. A simple Node.js + Express + better-sqlite3 application has been proven across multiple projects:
  - Searchable table list, paginated data grids
  - Row detail panel, BLOB viewer (inline images, PDFs)
  - SQL query runner with preset queries
  - No build step, runs on localhost

DON'T

- Don't try to export the entire database at once. Work by type, by year, and by chunk.
- Don't use Excel as the intermediate format. It has row limits (1M rows), is slow to write, and loses precision on large numbers.
- Don't forget to handle NULL values. NULLs should appear as empty cells, not as the string `'NULL'`.
- Don't neglect number formatting. Use `#,##0.00` for amounts and plain integer format for IDs (no thousands separator).
- Don't deliver only one format. SQLite for queries + CSV for Excel users + web viewer for browsing = maximum accessibility.

5 Validation & Quality Assurance

DO

- **Verify row counts:** the number of rows exported must match the source query count. Document counts per table.
- **Spot-check specific records:** pick 5 known records and verify every column against the original application or sample export.
- **Verify column count:** the output must have exactly the expected number of columns (e.g., 140 for Import Main in SEAGHA).
- **Verify character encoding:** check that special characters (Latvian: ā, č, ē, ģ, ī, ķ, ļ, ņ, š, ū, ž; Estonian: ä, ö, ü, õ; Slovenian: č, š, ž; Romanian: ă, â, î, ș, ț) appear correctly in the output.
- **Check for unknown/orphan records:** rows with NULL dates go into `*_Unknown` files — verify these are expected.
- **Document any data gaps:** if a backup only covers 2003–2020 and supplementary exports cover 2022, note the gap (2021). Be explicit about what is missing.
- **Keep the extraction scripts, schema DDL, and pipeline code** as part of the archival package. The data is useless without the context of how it was extracted.
- **Verify BLOB extraction:** spot-check that extracted images open correctly, PDFs are readable, XML templates parse.
- **Test the web viewer** with a non-technical user. Can they find an employee record? A specific declaration? A payslip?

DON'T

- Don't assume the extraction is correct just because the query runs without errors. Always validate against sample data.
- Don't delete the intermediate files (TXT, Parquet) after generating Excel. They may be needed for re-extraction or verification.
- Don't lose the schema DDL file. It is the only documentation of the database structure once the server is decommissioned.
- Don't sign off on extraction without encoding verification. A database that looks correct in row counts but has garbled text is a failed archival.

6 Vendor & Stakeholder Engagement

DO

- **Identify all external vendors early** who hold data or provide services. In MICROSCOP, the vendor (Mikrocop) held the authoritative 10-year archive — not FedEx.
- **Establish vendor contact through the business**, not through IT alone. Business contacts know the vendor relationship; IT may not have the right access or context.
- **Request vendor data export capabilities** and pricing upfront. Some vendors charge per-record; others provide bulk exports. Know before you budget.
- **Get access to vendor frontends/portals** for self-service verification. You need to confirm that the exported data matches what the vendor shows.
- **Document the vendor relationship:** contract terms, contact persons, data formats, SLAs, and any retention commitments.
- **Brief the business on what data will be archived and what will be lost.** They must sign off on scope — you cannot make compliance decisions for them.
- **Schedule regular check-ins** with business stakeholders during extraction. Show them sample output early. Their feedback is the specification.
- **Document knowledge transfer explicitly.** When the person who knows the system leaves, the archival becomes exponentially harder. Record everything in the first meeting.

DON'T

- Don't assume a vendor will cooperate on your timeline. Vendor engagement can stall for months (MICROSCOP: 2+ months waiting for contact).
- Don't skip the business stakeholder. They know which queries auditors run, which reports are legally required, and which data is actually used.
- Don't rely on a single person's knowledge. If only one person knows the system, document what they know immediately — they may leave before archival is complete.
- Don't assume active vendor processes can be retired immediately. MICROSCOP was still actively receiving daily invoice files — you cannot archive and retire simultaneously without a transition plan.

7 Delivery & Handoff

DO

- **Upload all files to a shared location** (SharePoint, network drive) with a clear folder structure.
- **Provide a README or Master Index** explaining the folder structure, file naming convention, and how to find specific data.
- **Include a technical reference document** with: database technology, server details, schema statistics, data volumes, extraction scripts, and all queries used.
- **Deliver extraction scripts** as standalone files that can be re-executed if the database is still available.
- **Note the data retention period** and any legal/compliance requirements that drove the archival, including country-specific regulations.
- **Include the web viewer application** with setup instructions (`npm install` , `npm start`). This makes the archive immediately usable without SQL skills.
- **Provide preset queries** for the most common lookups:
  - "Find employee by name"
  - "Show all declarations for reference number X"
  - "List payroll entries for period Y"
  - "Export contract details for employee Z"
- **Document the archival date and scope:** what was archived, from which system, which date range, and by whom. This metadata is essential for future compliance inquiries.

DON'T

- Don't deliver data without documentation. A folder of Excel files without context is barely better than no archive at all.
- Don't assume the recipient knows the database schema. Provide enough context for someone with SQL skills but no domain knowledge to navigate the data.
- Don't keep the archive in only one location. Ensure there is a backup of the backup.
- Don't forget to include the original database backup/files alongside the extracted output. Future needs may require re-extraction with different parameters.

8 Common Pitfalls — Lessons from Real Projects

PITFALL	PROJECT	WHAT HAPPENED	HOW TO AVOID
Custom database collation	FINAWIN	Latvian UNICODE_LV collation caused all text columns to return NULL via standard SQL	Test text extraction immediately; have a fallback plan (raw binary parsing)
Incomplete backup	SEAGHA_A	Database backup only contained data through 2020, missing 2021–2022	Verify date range immediately after restore; locate supplementary exports
Data held by vendor	MICROSCOP	Mikrocop (external vendor) held the full 10-year invoice archive, not FedEx	Map the complete data flow in planning phase; identify external data holders
Knowledge departure	MICROSCOP	Business contact moved to a different department; knowledge chain became fragile	Document everything in the first meeting; don't defer
Encoding loss	TAAVI	FoxPro files used CP1257 Baltic encoding; default UTF-8 readers produced garbled Estonian names	Always detect and specify encoding explicitly; verify diacritics
Active process retirement	MICROSCOP	The invoice process was still actively running — couldn't just archive and shut down	Understand full lifecycle before planning retirement

PITFALL	PROJECT	WHAT HAPPENED	HOW TO AVOID
BLOB content ignored	FINAWIN	Employee photos, report templates, and audit logs were stored as BLOBs — would have been lost without explicit extraction	Always check for BLOB/binary columns; extract and catalog them
Credential exposure	TAAVI	User tables contained SHA-512 hashed passwords	Flag credential tables; exclude from public-facing outputs

9 Archival Project Checklist

Use this checklist to track progress on each legacy system archival.

DISCOVERY

- [ ] Data source located (database backup, file system, vendor platform)
- [ ] System type and technology identified (SQL Server, Firebird, FoxPro, vendor-managed)
- [ ] Schema/structure exported (tables, views, stored procedures, file listings)
- [ ] Table/view/SP counts documented
- [ ] Data size documented (original + estimated output)
- [ ] Record types identified (declarations, employees, payroll, invoices, etc.)
- [ ] Sample export files obtained from business or application
- [ ] Date field for filtering identified
- [ ] Data range verified (which years are present)
- [ ] Data gaps identified and documented
- [ ] Character encoding identified and tested
- [ ] External vendor dependencies identified
- [ ] Regulatory retention periods confirmed with legal/compliance
- [ ] Business stakeholders identified and briefed
- [ ] Knowledge holders documented (names, roles, what they know)

DEVELOPMENT

- [ ] Extraction scripts/queries written for each data type
- [ ] Scripts handle character encoding correctly
- [ ] Scripts parameterized for testing vs. full extraction
- [ ] Scripts tested against sample records
- [ ] Column order verified against sample exports
- [ ] BLOB/binary content extraction planned
- [ ] Output filenames and folder structure documented

EXTRACTION

- [ ] All queries/scripts executed for full extraction
- [ ] Output files saved with headers
- [ ] Row counts verified against expected volumes
- [ ] Character encoding verified (diacritics, special characters)
- [ ] BLOB content extracted and cataloged
- [ ] Intermediate formats created (Parquet, SQLite)
- [ ] Final delivery formats created (Excel, CSV)
- [ ] Master Index or README created
- [ ] Web viewer built and tested (if applicable)

VALIDATION

- [ ] 5+ records spot-checked against original system/sample
- [ ] Row counts match source
- [ ] Column counts match specification
- [ ] Encoding verified for local-language content
- [ ] BLOBs verified (images open, PDFs readable)
- [ ] Data gaps documented explicitly
- [ ] Web viewer tested with non-technical user

DELIVERY

- [ ] Files uploaded to SharePoint/shared location
- [ ] README/documentation included
- [ ] Technical reference document included (DB details, scripts, schema)
- [ ] Original database files/backups preserved alongside output
- [ ] Extraction scripts included and documented
- [ ] Web viewer included with setup instructions
- [ ] Preset queries included for common lookups
- [ ] Email notification sent to stakeholders
- [ ] Data retention period and compliance requirements documented
- [ ] Vendor contacts and relationships documented (if applicable)
- [ ] Archival date, scope, and responsible person documented

10 Technology Quick Reference

DATABASE TYPE	EXTRACTION TOOL	ENCODING WATCH	OUTPUT FORMAT
SQL Server	pyodbc, SSMS, sqlcmd	Windows-1252, UTF-8	TXT → Parquet → Excel
Firebird	fdb driver, raw binary parser	Custom collations (ICU)	SQLite + CSV
Visual FoxPro (DBF)	dbfread (Python)	CP1257, CP1250	SQLite + CSV
Vendor platform	Vendor API/export	EBCDIC, platform-specific	PDF, CSV (vendor-dependent)

Proven Tech Stack

- **Extraction:** Python (`fdb` , `dbfread` , `pyodbc` , `pandas`)
- **Transformation:** Python (`sqlite3` , `pyarrow` /parquet, `openpyxl` / `xlsxwriter`)
- **Storage:** SQLite, Parquet, CSV, Excel (`.xlsx`)
- **Viewer:** Node.js + Express + better-sqlite3

- **Frontend:** Vanilla JS (no framework — simple, no build step)
- **Documentation:** Markdown + PDF export

> **About this guide.** *This guide is based on real-world experience from the following FedEx/TNT legacy archival projects: SAGE (01), CHARISMA (02), FINAWIN (03), MICROSCOP (04), TAAVI (05), and SEAGHA_A (50). It should be treated as a living document and updated as new projects are completed.*