TECO PGP Onboarding & Decryption/Encryption SOP

Author: Vishnu / RadiusPoint

Date: 2025-10-31

---

1. Purpose

This Standard Operating Procedure (SOP) describes step-by-step how to import TECO PGP keys, decrypt incoming `.pgp` EDI 810 files, run MapForce processing, create and encrypt 997 ACK files, and automate the process using FlowForce Server. It is written for the Windows environment using Gpg4win (gpg.exe) and PowerShell.

2. Scope

Applies to: RPAdmin Windows host that performs TECO EDI processing and FlowForce jobs.

This SOP covers: - Folder layout and permissions - PGP key import and verification - Manual decryption test - Automated PowerShell scripts for decrypt/encrypt - FlowForce job configuration notes - Troubleshooting and common errors - Security and hand-off notes

---

3. Prerequisites

1. Windows Server/Workstation where FlowForce / MapForce runs.
2. Gpg4win / GnuPG installed and gpg available in PATH.
3. RPAdmin account (or another service account) that will run FlowForce jobs and hold the GPG keyring.
4. TECO provided key pair (or vendor public key) files and passphrase if provided:
    - Private key file: `0xxxxxxxx-sec.asc` (example)
    - Public key file: `0xxxxxxxxx-pub.asc` (example)
5. Sample encrypted test file from TECO.

---

4. Folder Structure (create on RPAdmin desktop)

Create exact folders by running the PowerShell snippet below (run as RPAdmin):

```
C:\FlowForce\TECO\
├── incoming\
├── decrypted\
├── processed\
├── archive\
├── ack\
└── logs\
```

---

5. Key Import & Verification

1. Place key files into `C:\Users\RPAdmin\Desktop\EDI-TECO\Key`.

2. Import keys (run as RPAdmin):

```
cd "C:\Users\RPAdmin\Desktop\EDI-TECO\Key"
gpg --import "0xxxxxxxx-pub.asc"
gpg --import "0xxxxxxxxx-sec.asc"
```

3. Restart agent (fix timeouts):

```
gpgconf --kill gpg-agent
gpgconf --launch gpg-agent
```

4. Verify keys are present:

```
gpg --list-keys
gpg --list-secret-keys
gpg --fingerprint "ABC@radiuspoint.com"
```

Expected: secret key entry with key id `68DFB2E4FC0435` (or `0xFC0BA6B4`) should appear.

---

6. Confirm which key the vendor used (before decrypting)

To avoid `No secret key` errors, inspect the encrypted file:

```
gpg --list-packets "C:\Users\RPAdmin\Desktop\EDI-TECO\Incoming\Encrypted-Incoming\<filename>.pgp"
```

Look for the `keyid` line. It must match your key id (e.g. `68DFB2E4FC0BA6B4`). If it does not match, request the vendor re-encrypt using your public key.

---

7. Manual Decryption (one-time test)

1. Run this command (PowerShell or CMD). Use CMD if passphrase contains `%` or `?`.

```
# PowerShell:
gpg --output "C:\Users\RPAdmin\Desktop\EDI-TECO\Incoming\Decrypted-
Output\FileName.txt" `
--decrypt "C:\Users\RPAdmin\Desktop\EDI-TECO\Incoming\Encrypted-Incoming\
Filename.pgp"
```

2. If pinentry GUI blocks automation, use loopback (temporary for test):

```
# Replace <passphrase> with the private key passphrase
gpg --batch --yes --pinentry-mode loopback --passphrase "<passphrase>" `
--output "C:\Users\RPAdmin\Desktop\EDI-TECO\Incoming\Decrypted-
Output\<output>.txt" `
--decrypt "C:\Users\RPAdmin\Desktop\EDI-TECO\Incoming\Encrypted-
Incoming\<file>.pgp"
```

3. Confirm decrypted file exists and contains EDI (ISA*, GS*, ST* lines).

---

8. PowerShell Decrypt Script (check-and-decrypt_TECO.ps1)

Save under `C:\All_Scripts\Teco\check-and-decrypt_TECO.ps1` (or `Scripts\Decryption_Script`):

```
# check-and-decrypt_TECO.ps1
<#
   decrypt_auto_with_pass.ps1
   - Decrypts all .pgp files in $inDir using loopback pinentry and a supplied passphrase
   - Safe argument passing (array form) to avoid quoting issues
   - Logs to a daily logfile
#>


# ---------- CONFIG ----------
$inDir   = "C:\Users\RPAdmin\Desktop\EDI-TECO\Incoming\Encrypted-Incoming"
```

```
$outDir  = "C:\Users\RPAdmin\Desktop\EDI-TECO\Incoming\Decrypted-Output"
$passphrase = "x3XusS%RB?8ZbnCY"          # Your passphrase (protect file permissions)
$gpgExe = "C:\Program Files\GnuPG\bin\gpg.exe"
if (-not (Test-Path $gpgExe)) { $gpgExe = "C:\Program Files (x86)\GnuPG\bin\gpg.exe" }

$logDir = Join-Path (Split-Path $outDir -Parent) "Logs"
if (-not (Test-Path $logDir)) { New-Item -ItemType Directory -Path $logDir -Force | Out-Null }
$logFile = Join-Path $logDir ("decrypt_auto_" + (Get-Date -Format "yyyyMMdd") + ".log")

function Log($msg){
   $ts = (Get-Date).ToString("yyyy-MM-dd HH:mm:ss")
   "$ts  $msg" | Out-File -FilePath $logFile -Append -Encoding utf8
   Write-Host $msg
}

# Make sure directories exist
if (-not (Test-Path $inDir)) { Log "ERROR: Input folder not found: $inDir"; exit 1 }
if (-not (Test-Path $outDir)) { New-Item -ItemType Directory -Path $outDir -Force | Out-Null;
Log "Created output folder: $outDir" }

# Check gpg exists
if (-not (Test-Path $gpgExe)) { Log "ERROR: gpg.exe not found at expected locations."; exit 2 }

# Log current user and GnuPG homedir for debugging
$currentUser = [System.Security.Principal.WindowsIdentity]::GetCurrent().Name
Log "Running as user: $currentUser"

# Optional: show GPG home (may vary)
```

```
$gpgHome = (& $gpgExe --version 2>$null) | Select-String -Pattern "Home:" -SimpleMatch
if ($gpgHome) { Log "GnuPG version/home info: $($gpgHome.Line)" }

# Helper: attempt to ensure agent allows loopback (best-effort)
# This will only work if gpg-agent.conf is writable by this user and gpgconf is available.
try {
   $agentConfPath = Join-Path $env:APPDATA "gnupg\gpg-agent.conf"
   if (Test-Path $agentConfPath) {
      $confText = Get-Content $agentConfPath -ErrorAction SilentlyContinue
      if ($confText -notmatch 'allow-loopback-pinentry') {
         Log "Note: gpg-agent.conf exists but does not contain allow-loopback-pinentry. Please add it manually if needed: $agentConfPath"
      } else {
         Log "gpg-agent.conf already contains allow-loopback-pinentry"
      }
   } else {
      Log "gpg-agent.conf not found at $agentConfPath. Create it and add line: allow-loopback-pinentry"
   }
} catch {
   Log "Warning: could not inspect gpg-agent.conf: $($_.Exception.Message)"
}

# Process files
$files = Get-ChildItem -Path $inDir -Filter *.pgp -File -ErrorAction SilentlyContinue
if (-not $files -or $files.Count -eq 0) { Log "No .pgp files to process in $inDir."; exit 0 }

foreach ($f in $files) {
```

```
    $inFile = $f.FullName
    $outFile = Join-Path $outDir ($f.BaseName + ".dec")

    Log "Starting decrypt: $($f.Name)"

    # Build argument array (safer than one long string)
    $args = @(
        "--batch",
        "--yes",
        "--pinentry-mode", "loopback",
        "--passphrase", $passphrase,
        "--output", $outFile,
        "--decrypt", $inFile
    )

    # Run gpg
    $proc = Start-Process -FilePath $gpgExe -ArgumentList $args -NoNewWindow -Wait -
PassThru
    $exit = $proc.ExitCode

    if ($exit -eq 0) {
        Log "Decrypted OK -> $outFile"
        try {
            Remove-Item -Path $inFile -Force
            Log "Deleted source encrypted file: $inFile"
        } catch {
            Log "Warning: failed to delete source file: $($_. Exception. Message)"
        }
```

```
    } else {
        Log "ERROR: gpg exit code $exit for file $($f.Name)."
        # capture any stderr output - try running gpg with --verbose to debug if needed
    }
}
Log "Script finished."
```

Important: remove plaintext passphrase and use gpg-agent or FlowForce credential store for production. Replace `<REPLACE_WITH_SECURE_METHOD>` with secure mechanism.

---

9. PowerShell Encrypt Script (encrypt-997_TECO.ps1)

Save as `C:\Users\RPAdmin\Desktop\EDI-TECO\Scripts\Encryption_Script\encrypt-997_TECO.ps1`. (Full script included in the attached SOP file.)

```
# ============================
# Encrypt all 997 ACK files for TECO
# Source:  Ack\Decryption  →  Output: Ack\Encryption
# Uses vendor public key: 0xFC0BA6B4-pub.asc
# ============================
# --- Paths ---
$SourceFolder = "C:\Users\RPAdmin\Desktop\EDI-TECO\Ack\Decryption"
$TargetFolder = "C:\Users\RPAdmin\Desktop\EDI-TECO\Ack\Encryption"
$KeysFolder   = "C:\Users\RPAdmin\Desktop\EDI-TECO\Key"
$LogFolder    = "C:\Users\RPAdmin\Desktop\EDI-TECO\Incoming\Logs"

# Public key and passphrase
$PublicKeyFile = Join-Path $KeysFolder "0xFC0BA6B4-pub.asc"
$Passphrase = "x3XusS%RB?8ZbnCY."

# GPG executable
```

```
$gpgPath = "C:\Program Files\GnuPG\bin\gpg.exe"
if (-not (Test-Path $gpgPath)) { $gpgPath = "C:\Program Files (x86)\GnuPG\bin\gpg.exe" }


# --- Ensure folders exist ---
foreach ($p in @($TargetFolder, $LogFolder)) {
   if (-not (Test-Path $p)) { New-Item -ItemType Directory -Path $p -Force | Out-Null }
}
# --- Logging ---
$LogFile = Join-Path $LogFolder ("encrypt_997_teco_" + (Get-Date -Format "yyyyMMdd") +
".log")
function Log($m) {
   $ts = (Get-Date).ToString("yyyy-MM-dd HH:mm:ss")
   "$ts  $m" | Out-File -FilePath $LogFile -Append -Encoding utf8
   Write-Host $m
}
# --- Validate source folder ---
if (-not (Test-Path $SourceFolder)) {
   Log "ERROR: Source folder not found: $SourceFolder"
   exit 1
}
# --- Import public key if missing ---
$keyImported = $false
$keyID = "0xFC0BA6B4"
$existingKeys = & $gpgPath --list-keys $keyID 2>$null
if (-not $existingKeys) {
   if (Test-Path $PublicKeyFile) {
      Log "Importing TECO public key: $PublicKeyFile"
      & $gpgPath --import "$PublicKeyFile" 2>&1 | Out-Null
```

```
        $keyImported = $true
    } else {
        Log "ERROR: Public key file not found: $PublicKeyFile"
        exit 2
    }
}
Log "Using public key ID: $keyID"
if ($keyImported) { Log "Public key imported successfully." }

# --- Collect files to encrypt (skip existing .pgp/.gpg) ---
$files = Get-ChildItem -Path $SourceFolder -File | Where-Object { $_.Extension -notin @(".pgp", ".gpg") }
if (-not $files -or $files.Count -eq 0) {
    Log "No files to encrypt in $SourceFolder."
    exit 0
}

# --- Encrypt each file ---
foreach ($f in $files) {
    $inFile  = $f.FullName
    $outFile = Join-Path $TargetFolder ($f.BaseName + ".pgp")

    Log "Encrypting: $($f.Name) → $(Split-Path $outFile -Leaf)"

    & $gpgPath --batch --yes --trust-model always --recipient $keyID `
        --pinentry-mode loopback --passphrase "$Passphrase" `
        --output "$outFile" --encrypt "$inFile"
```

```
    if ($LASTEXITCODE -ne 0) {
        Log "ERROR: gpg failed with exit code $LASTEXITCODE for $($f.Name)."
        continue
    }

    Log "Encrypted successfully: $($f.Name) → $(Split-Path $outFile -Leaf)"
}

Log "Encryption run complete."
```

---

10. FlowForce Job Configuration (high-level)

Job 1 — TECO_Decrypt - Task: Execute program (powershell.exe) - Arguments: `-ExecutionPolicy Bypass -File "C:\All_Scripts\Teco\check-and-decrypt_TECO.ps1"` - Run as: RPAdmin (must have GPG keyring and gpg-agent) - Trigger: Folder trigger on `Incoming\Encrypted-Incoming`

Job 2 — MapForce_Process - Task: Execute MapForce CLI to map decrypted `.txt` to CSV

Job 3 — ACK_Encrypt_And_Send - Task: Run `encrypt-997_TECO.ps1` to create ACK `.pgp` - Task: SFTP put `.pgp` to vendor (use FlowForce credentials store)

---

11. Troubleshooting (common errors & fixes)

- `No secret key` → vendor encrypted with wrong key. Run `gpg --list-packets <file>` and confirm `keyid`. Request vendor re-encrypt to your public key.
- `Timeout` or pinentry hangs → restart agent: `gpgconf --kill gpg-agent` then `gpgconf --launch gpg-agent`.
- `can't open ... No such file or directory` → verify exact file path and filename; use `Get-ChildItem` to confirm.
- `gpg not found` → ensure Gpg4win installed and gpg path included; update scripts with full path.

- If automation requires no interactive passphrase: configure `gpg-agent` or use secure passphrase file with `--pinentry-mode loopback` (security risk — avoid unless approved).

---

12. Security & Best Practices

- Do not store passphrase in plaintext files. Use `gpg-agent` or FlowForce credential store.
- Share public keys over a verified out-of-band channel (email+phone).
- Apply tight NTFS ACLs to `Key` and `Scripts` folders (RPAdmin only).
- Log all errors and set FlowForce alerts for failures.

---

13. Email Template to Request Vendor Re-encrypt

```
Subject: Please re-encrypt EDI test file using our public key (Key ID:
0xFC0BA6B4)

Hi TECO,

Please re-encrypt the attached test EDI file using our public key (attached:
0xFC0BA6B4-pub.asc). We could not decrypt the previous file; it was encrypted
to a different key.

Regards,
Vishnu
```

---

14. Appendix: Useful Commands

- List keys: `gpg --list-keys`
- List secret keys: `gpg --list-secret-keys`
- Inspect packet/keyid: `gpg --list-packets <file>.pgp`
- Import keys: `gpg --import <keyfile.asc>`
- Decrypt file: `gpg --output <out.txt> --decrypt <file>.pgp`
- Encrypt file: `gpg --output <out.pgp> --encrypt -r <recipient>`

End of SOP