***Script Documentation Table of Contents***

[Projets.py documentation………………………………………………………………………………………3](#ProjectsPy)

[1.) Overview](#Overview_P)

[2.) Dependencies and prerequisites](#Depend_P)

[3.) Network Endpoints & Fixed Paths](#Network_P)

[4.) Operational Notes](#Operation_P)

[5.) Logging and Observability](#Logging_P)

[6.) Why this script only runs on a designated desktop](#Why_P)

[MasterProposals.py……………………………………………………………………………………………….5](#ProposalsPY)

[1.) Overview](#Overview_MP)

[2.) Dependencies and Prerequisites](#Depend_MP)

[3.) Network Endpoints & Fixed Paths](#Network_MP)

[4.) Operational Notes](#Operation_MP)

[5.) Logging and Observability](#Logging_MP)

[6.) Why this script only runs on designated Desktops](#Why_MP)

[Azure-Signin-Monitor.py………………………………………………………………………………………….9](#AzurePy)

[1.) Overview](#Overview_A)

[2.) Dependencies and Prerequisites](#Depend_A)

[3.) Network Endpoints & Fixed Paths](#Network_A)

[4.) Operational Notes](#Operation_A)

[5.) Logging and Observability](#Logging_A)

[6.) Why this script only runs on designated Desktops](#Why_A)

[RegularMasterProposals.py…………………………………………………………………………………..12](#RegularPy)

[1.) Overview](#Overview_R)

[2.) Dependencies and Prerequisites](#Depend_R)

[3.) Network Endpoints & Fixed Paths](#Network_R)

[4.) Operational Notes](#Operational_R)

[5.) Logging and Observability](#Logging_R)

[6.) Why this script only runs on designated Desktops](#Why_R)

[MasterProjects.py………………………………………………………………………………………………..15](#MProjectsPy)

[1.) Overview](#Overview_MProject)

[2.) Dependencies and Prerequisites](#Depend_MProject)

[3.) Network Endpoints & Fixed Paths](#Network_MProject)

[4.) Operational Notes](#Operation_MProject)

[5.) Logging and Observability](#Logging_MProject)

[6.) Why this script only runs on designated Desktops](#Why_MProject)

[MasterOpportunity.py…………………………………………………………………………………………..19](#MoppPy)

[1.) Overview](#Operation_MO)

[2.) Dependencies and Prerequisites](#Depend_MO)

[3.) Network Endpoints & Fixed Paths](#Network_MO)

[4.) Operational Notes](#Operation_MO)

[5.) Logging and Observability](#Logging_MO)

[6.) Why this script only runs on designated Desktops](#Why_MO)

[LucidEgnyteLnk.py……………………………………………………………………………………………….22](#LucidPy)

[1.) Overview](#Overview_L)

[2.) Dependencies and Prerequisites](#Depend_L)

[3.) Network Endpoints & Fixed Paths](#Network_L)

[4.) Operational Notes](#Operational_L)

[5.) Logging and Observability](#Logging_L)

[6.) Why this script only runs on designated Desktops](#Why_L)

[Recommendations for Scripts………………………………………………………………………………27](#Rec)

[Requirements.text………………………………………………………………………………………………30](#text)

***[Projects.py Documentation](#Main)***

**[1.) Overview](#Main)**

This script keeps your Egnyte project folders and Acumatica “EgnyteFolder” field in sync for Bennett & Pless Leicht (BPL) projects. Every 15 minutes, it authenticates to Egnyte and Acumatica, finds all active BPL projects in Acumatica whose IDs start with the current year prefix (e.g., 25) and whose EgnyteFolder is empty or “Pending,” checks if the corresponding project folder exists in Egnyte, copies a standardized template when it doesn’t, and then updates the Acumatica project’s EgnyteFolder value to “CD”. It then logs off Acumatica and repeats.

**[2) Dependencies and prerequisites](#Main)**

The script uses only standard libraries plus requests and python-dotenv.

**Python packages**

* requests (HTTP calls, timeouts, retryable exceptions)
* python-dotenv (loads secrets from an .env file)

**Standard library modules used (no install required)**

* logging, time, os, json, urllib.parse (only quote is imported), exceptions from requests

**[3.) Network endpoints & fixed paths](#Main)**

* Acumatica token: **https://bennett-pless.acumatica.com/identity/connect/token**
* Acumatica API base: **https://bennett-pless.acumatica.com**
* Egnyte API base: **https://bennettpless.egnyte.com/pubapi/v1**
* Egnyte OAuth: **https://bennettpless.egnyte.com/puboauth/token**
* Egnyte template folder (source): **/Shared/Team Member Resources/IT/ProjectFolderTemplates/BPL New Project**
* Egnyte destination base (target): **/Shared/Bennett & Pless Leicht/projects/2025/{folder\_name}**

**Local files created**

* **acumatica\_token\_cache.json** — stores access\_token, session id, and expiration timestamp
* **egnyte\_token\_cache.json** — stores access\_token and a timestamp for simple freshness checks
* **application.log** — standard log file (INFO level)

**[4.) Operational notes](#Main)**

* The main loop sleeps for 900 seconds (15 minutes) between iterations.
* **Timeouts:** Acumatica logoff uses a 30s timeout; most Egnyte/Acumatica requests use 30s.
* **Backoff:** Egnyte auth waits 5s up front and honors Retry-After on 429 (rate limit). Acumatica **retrieve\_project\_data** retries up to 5 times with 10s delay. Logoff has exponential-ish backoff.
* **Year prefix:** **current\_year\_prefix** = '25'. Update this value each calendar year, or make it dynamic as needed.
* **Safety:** If Egnyte folder existence can’t be confirmed (non-404/200), the project update is skipped.

**[5.) Logging & observability](#Main)**

* All major actions print to stdout and also write to application.log with timestamps and levels.
* Failures during per-project processing are caught and logged without terminating the loop.

**[6.) Why this script “only runs on a designated desktop”](#Main)**

**Absolute, machine-specific file path for secrets**

* The script loads:  
  c:/scripts/BPL Projects/config.env  
  That path must exist, be readable, and contain valid credentials. On another PC, that path likely doesn’t exist or is blocked by permissions or different drive mappings.

**Local token/cache & log write permissions**

* It writes **acumatica\_token\_cache.json**, **egnyte\_token\_cache.json**, and **application.log** to the current working directory.
* The designated box is typically the one that:
  + Runs 24/7 (Task Scheduler/service)
  + Has guaranteed write permissions in that directory
  + Is monitored/backed up appropriately

**Long-running scheduling model**

* The script is a daemon-style loop (while True with 15-minute sleeps).
* The designated desktop likely has Windows Task Scheduler/NSSM/service wrappers set up to auto-start on boot, restart on crash, and keep it alive.

**Preconfigured credentials & least-privilege**

* The .env on that desktop likely holds production **service credentials** with the right scopes/roles in Acumatica and Egnyte.
* Replicating those safely to “any desktop” is a **security risk** and often restricted by policy.

**File system conventions inside Egnyte**

* Paths like /Shared/Bennett & Pless Leicht/projects/2025 and the template folder assume **specific Egnyte permissions** for the runtime identity. The designated desktop is set up so that the process identity has those rights.

**[MasterProposals.py Documentation](#Main)**

**[1.) Overview](#Main)**  
This script provisions and maintains **Egnyte project folders** for non-BPL projects and keeps Acumatica’s **EgnyteFolder** field in sync. On a 5-minute loop it:

* Authenticates to **Acumatica** and **Egnyte** (with simple local token caching).
* Pulls all projects, then filters to **Active**, **ProjectID starts with ’25’**, **Branch != BPL**, and **EgnyteFolder != 'Created'**.
* Builds the **destination folder path** from a department code in the ProjectID (e.g., 25.04… → "25.04.000 - Knoxville").
* Chooses a **template** (from a CSV mapping, with special handling for “Series Projects” and Telecom) and **copies** it into Egnyte if the folder doesn’t exist.
* Sets Acumatica **EgnyteFolder = "CD"** after confirming/creating the folder.
* Sends a **single alert email** if a folder copy fails (per folder name).
* Logs off Acumatica and repeats.

**[2.) Dependencies and prerequisites](#Main)**  
The script uses only standard libraries plus requests and python-dotenv.

**Python packages**

* requests (HTTP calls to Acumatica/Egnyte; timeouts; error handling)
* python-dotenv (loads secrets from config.env)

**Standard library modules used (no install required)**

* logging, time, os, json, csv, email, smtplib, re, threading/concurrent (imports), urllib.parse, etc.

**Environment keys expected in config.env**

* **Acumatica:** ACUMATICA\_CLIENT\_ID, ACUMATICA\_CLIENT\_SECRET, ACUMATICA\_USERNAME, ACUMATICA\_PASSWORD
* **Egnyte:** EGNYTE\_CLIENT\_ID, EGNYTE\_CLIENT\_SECRET, EGNYTE\_USERNAME, EGNYTE\_PASSWORD
* **Email (alerts):** SMTP\_SERVER, SMTP\_PORT (default 587), SMTP\_PASSWORD, SENDER\_EMAIL

**[3.) Network endpoints & fixed paths](#Main)**  
**Endpoints**

* Acumatica token: **https://bennett-pless.acumatica.com/identity/connect/token**
* Acumatica API base: **https://bennett-pless.acumatica.com**
* Egnyte API base: **https://bennettpless.egnyte.com/pubapi/v1**
* Egnyte OAuth: **https://bennettpless.egnyte.com/puboauth/token**

**Egnyte folders**

* Template base (source): **/Shared/Team Member Resources/IT/ProjectFolderTemplates/<TemplateName>**
  + Special cases: Series Projects Child, Niche Tower Project Template
* Destination base (target): **/Shared/Projects/2025/{Department}/{FolderName}**
  + Department is resolved from the ProjectID’s middle segment via the built-in department\_mapping.

**Local/Windows paths the script assumes exist**

* Template map CSV**:** **C:/Scripts/Regular Projects/TemplateGuide.csv** (used by determine\_template\_path)

**Local files created (working directory)**

* **acumatica\_token\_cache.json** — access token, session id, expiration timestamp
* **egnyte\_token\_cache.json** — Egnyte access token + timestamp
* **application.log** — log output (INFO level)
* **projects.csv** — raw export of project data from Acumatica (overwritten on each run of retrieval)

**[4.) Operational notes](#Main)**

* **Loop cadence:** sleeps 300 seconds (5 min) between iterations.
* **Filtering before processing:**  
  status == Active ∧ ProjectID startswith '25' ∧ branch != 'BPL' ∧ EgnyteFolder != 'Created'.
* **Template selection:**
  + Main logic uses **TemplateGuide.csv** (ConditionSet → TemplateName).
  + Special cases:
    - **Series projects** (AddToExistingSeries=True): copy Series Projects Child under the master project folder.
    - **Telecom** (Infrastructure + Telecom Structures): copy Niche Tower Project Template.
* **Folder naming:**
  + **Master project:** {Department}/{ProjectID} - {ProjectName}
  + **Standard project:** {Department}/{ProjectID} - {ClientCustomerID} - {ProjectName}
* **Retries & timeouts:**
  + Egnyte **auth**: waits 5s up front; respects Retry-After on **429**; max 3 attempts.
  + Folder **copy**: 30s timeout; retries once on timeout; sanitizes a **trailing period** in names if Egnyte rejects ("File names cannot contain leading or trailing spaces"), then retries once.
* **State/caching:**
  + Acumatica and Egnyte tokens cached to local JSON files; Egnyte cache considered fresh for ~3500s.
* **Acumatica field update:** always sets **EgnyteFolder = "CD"** after detecting/creating the folder.
* **BPL branch:** intentionally **skipped** here (handled by separate BPL-specific job).

**[5.) Logging & observability](#Main)**

* Logs to **stdout** and application.log.
* Writes a one-time **alert email** per failing folder (prevents spam) to support@bennett-pless.com.
* Exports current project payload to **projects.csv** for troubleshooting/traceability.

**[6.) Why this script “only runs on a designated desktop”](#Main)**

* **Absolute, machine-specific pathing:**  
  Requires C:/Scripts/Regular Projects/TemplateGuide.csv. On other machines that path/drive or file won’t exist or be accessible.
* **Local secrets and token caches:**  
  Expects a config.env in the working directory (or a known location) with production credentials. The designated box is where those are securely staged and protected; duplicating them to “any desktop” is a security risk.
* **Network egress and allowlists:**  
  Outbound access to **Acumatica**, **Egnyte**, and **SMTP** may be IP-allowlisted, behind corporate proxies/TLS inspection, or tied to a managed device posture. A random desktop typically lacks those network exemptions.
* **Long-running service expectations:**  
  It’s a daemon-style loop (while True + sleeps). The designated desktop/server is configured with **Task Scheduler/service wrappers**, restart-on-boot, log rotation, monitoring, and backups for the working directory.
* **Filesystem permissions & conventions:**  
  The process must write application.log, the token caches, and projects.csv in the current working directory. The designated host guarantees those permissions and includes the required folder structure.

**[Azure-Signin-Monitor.py Script Documentation](#Main)**

**[1.) Overview](#Main)**  
This script monitors **Azure AD sign-ins** and flags potential risks. It pulls sign-in logs from **Microsoft Graph**, builds per-user **whitelists** (usual locations + typical hours), and then—on each polling cycle—detects: logins from **unusual locations**, **unusual hours**, and **impossible travel** between logins. It can **suppress alerts** from trusted devices/IPs, **email** a CSV report, and (optionally) post a **Teams** notification. It supports a **training mode** to build initial whitelists from historical data, then switches to standard monitoring and repeats on a schedule.

**[2.) Dependencies and prerequisites](#Main)**  
The script uses only standard libraries plus requests, python-dotenv, msal, pytz, and geopy.

**Python packages**

* requests (HTTP calls, Graph API, Teams webhook)
* python-dotenv (loads secrets from a config.env file)
* msal (auth for Microsoft Graph)
* pytz (timezone conversions)
* geopy (geodesic distance for “impossible travel”)

**Standard library modules used (no install required)**

* os, json, csv, datetime, time, signal, sys, smtplib, email.\*,  
  uuid, logging, re, secrets, threading, concurrent.futures,  
  collections, math, traceback

**[3.) Network endpoints & fixed paths](#Main)**

* Graph sign-ins (read): **https://graph.microsoft.com/v1.0/auditLogs/signIns**
* Graph auth authority (tenant): **https://login.microsoftonline.com/{tenant\_id}**
* Teams (optional, incoming webhook): **your TEAMS\_WEBHOOK\_URL**
* Teams file upload (Graph): uses the channel’s filesFolder and an upload session  
  (**e.g., .../teams/{group\_id}/channels/{channel\_id}/filesFolder → drives/{driveId}/items/{itemId}:/{file}:/createUploadSession**)
* SMTP: your **SMTP\_SERVER:SMTP\_PORT** from env

**Fixed/relative directories (created if missing)**

* ./Data
* ./Polling Reports

**Local files created (relative to the script directory)**

* **Data/login\_history.csv** — rolling store of processed sign-ins
* **Data/last\_fetch.json** — watermark for incremental polling
* **Data/user\_whitelists.json** — per-user safe locations & work hours
* **Data/device\_fingerprint\_whitelist.json** — trusted device fingerprints
* **Data/training\_data\_suspicious\_report\_<timestamp>.csv** — training-mode report
* **Polling Reports/suspicious\_signin\_report\_<start>\_to\_<end>.csv** — per-poll report
* **Data/excluded\_alerts\_log.csv** — alerts suppressed due to device/IP exclusions

**Config file (env)**

* The class loads config.env **from the script’s folder** by default (path is resolved relative to \_\_file\_\_).

**[4.) Operational notes](#Main)**

* Polling interval: configurable via **POLLING\_PERIOD\_MINUTES** (default **5 minutes**). Sleep is broken into ≤10-second chunks for responsive shutdown.
* **Modes:**
  + **Training mode** (TRAINING\_MODE=True): fetches historical window (default 30 days), builds login\_history.csv, creates whitelists, then switches to standard mode.
  + **Standard mode:** incremental fetch using last\_fetch.json (+15-minute lookback buffer), processes, updates CSV, applies whitelists, generates/sends reports.
* **Checks/toggles (env):**  
  CHECK\_LOCATION, CHECK\_UNUSUAL\_HOURS, CHECK\_IMPOSSIBLE\_TRAVEL (off by default), plus exempt app lists for location/impossible-travel checks.
* **Device/IP exclusions:**  
  EXCLUDE\_ENROLLED\_DEVICES, EXCLUDE\_WHITELISTED\_FINGERPRINTS, EXCLUDE\_WHITELISTED\_IPS with WHITELISTED\_IPS.  
  Fingerprints are derived from device/user agent/app attributes and can be auto-whitelisted by frequency/day-span thresholds.
* **Rate limiting & backoff:**
  + A sliding-window **RateLimitManager** (default **2000 req/min**, waits at 80%/95%).
  + Graph calls honor **429 Retry-After** when present; otherwise exponential backoff (capped at 120s).
  + **MSAL** token acquisition also retries with exponential backoff.
* **Timezones:** sign-in timestamps are converted to **US/Eastern** for reporting and “hours” analysis.
* **Impossible travel:** uses **geopy** great-circle distance, compares against time delta; flags if computed mph exceeds MAX\_TRAVEL\_SPEED\_MPH (default **900**) and distance ≥ MIN\_TRAVEL\_DISTANCE\_MILES (default **50**).
* **Teams notifications (optional):** posts an Adaptive Card to the incoming webhook; can also **upload the CSV** to the channel via Graph upload session.
* **Graceful shutdown:** handles SIGINT/SIGTERM; finishes current work, then exits.

**[5.) Logging & observability](#Main)**

* Progress and errors are printed to stdout; CSV/JSON artifacts in Data/ + Polling Reports/ serve as durable audit outputs.
* Suppressed alerts (due to device/IP exclusions) are logged to Data/excluded\_alerts\_log.csv.
* Each suspicious batch produces a timestamped CSV report; email and/or Teams posts summarize counts and details.

**[6.) Why this script “only runs on a designated desktop”](#Main)**  
**Secrets & app credentials on that machine**

* config.env (client ID/secret, tenant, SMTP creds, Teams webhook, IDs) resides alongside the script and is trusted on that box. Replicating to “any desktop” expands your attack surface and may violate policy.

**Corporate network posture**

* **Graph/Teams/SMTP** calls often require the corporate VPN, outbound firewall allowlists, and trusted root CAs/proxies installed on managed machines. An arbitrary desktop may fail TLS inspection or egress filtering.

**Long-running service with local state**

* The monitor maintains **local state** (last\_fetch.json, whitelists, history CSVs, fingerprint whitelist) and runs continuously on a stable host (Task Scheduler/service). That host is set up for **24/7 uptime**, monitoring, and backups.

**Least-privilege & allowlists**

* The Azure app registration, Teams channel, and SMTP relay may be **IP-restricted** or scoped to a managed device identity. Running elsewhere can fail auth or violate security controls.

**Filesystem expectations**

* The script **writes** to ./Data and ./Polling Reports. The designated desktop has known **write permissions** and expected folder layout tied to ops/backup routines.

**[RegularMastersProposal.py Documentation](#Main)**

**[1.) Overview](#Main)**

This script keeps Egnyte proposal folders and the Acumatica **Proposals.EgnyteFolder** field in sync for non-BPL proposals. Every cycle it:

* Authenticates to **Egnyte** and **Acumatica** (with simple JSON token caches).
* Pulls proposals from Acumatica.
* Filters to statuses **Prepared** or **Draft**, proposal numbers starting with **P.25**, **Branch != BPL**, and **EgnyteFolder != "Created"**.
* For each, builds a folder name:  
  "{ProposalNbr} - {BusinessAccount} - {ProjectDescription}".
* Checks Egnyte for /Shared/proposals/2025/{folder\_name}; if missing it **copies a template** folder and then **updates Acumatica** to set **EgnyteFolder = "Created"**.
* Logs off Acumatica and repeats.

**[2.) Dependencies & prerequisites](#Main)**

Uses standard library plus two third-party packages.

**Python packages**

* requests – HTTP calls
* python-dotenv – loads secrets from a config.env

**Standard library used (no install)**  
logging, os, time, json, csv, threading/concurrency imports, urllib.parse.quote

**Environment variables (in config.env)**

* Acumatica: ACUMATICA\_CLIENT\_ID, ACUMATICA\_CLIENT\_SECRET, ACUMATICA\_USERNAME, ACUMATICA\_PASSWORD
* Egnyte: EGNYTE\_CLIENT\_ID, EGNYTE\_CLIENT\_SECRET, EGNYTE\_USERNAME, EGNYTE\_PASSWORD

**[3.) Network endpoints & fixed paths](#Main)**

**Endpoints**

* Acumatica token: https://bennett-pless.acumatica.com/identity/connect/token
* Acumatica API base: https://bennett-pless.acumatica.com
* Egnyte API base: https://bennettpless.egnyte.com/pubapi/v1
* Egnyte OAuth: https://bennettpless.egnyte.com/puboauth/token

**Egnyte folders**

* Template (source): /Shared/Team Member Resources/IT/ProjectFolderTemplates/P24-XXXX - Proposal Example
* Destination base (target): /Shared/proposals/2025/{folder\_name}

**Local files created (relative to the script path)**

* proposals.log – rotating run log (INFO level)  
  *(The script resolves this beside the .py file.)*
* egnyte\_token\_cache.json – Egnyte access token + timestamp
* acumatica\_token\_cache.json – Acumatica access token, ASP.NET session id, expiration

**[4.) Operational notes](#Main)**

* **Loop cadence:** perpetual loop; sleeps **600s (10 min)** between iterations.
* **Filters:** Status in {"prepared","draft"}, ProposalNbr startswith "P.25", Branch != "BPL", EgnyteFolder != "Created".
* **Template choice:** currently hard-wired to P24-XXXX - Proposal Example (update when your template changes).
* **Token caching:**
  + Egnyte token considered fresh for ~3500s; otherwise re-auths (with a 5s pre-sleep to reduce rate limits; obeys Retry-After on 429).
  + Acumatica token/session cached with an expiration guard; explicit **logoff** after each pass.
* **Resilience:** JSON-decode guards and retry loops when fetching proposals; errors on a single proposal won’t stop the loop.
* **CSV export:** proposals.csv contains the latest fetched proposal info (including EgnyteFolder).

**[5.) Logging & observability](#Main)**

* Writes **proposals.log** next to the script with timestamps and levels.
* Prints progress to stdout (start/end of cycle, counts, per-proposal actions).
* Key actions logged: folder existence checks, template copies, Acumatica updates, HTTP failures, and decode errors.

**[6.) Why this “only runs on a designated desktop”](#Main)**

**Absolute, machine-specific file path**

* The template guide is read from **c:/Scripts/Regular Proposals/ProposalTemplateGuide.csv**. That Windows path must exist with the expected CSV. On other machines it likely doesn’t.

**Local secrets and caches**

* The process expects a valid **config.env** discoverable by find\_dotenv('config.env'). The designated box holds the production secrets.
* It writes egnyte\_token\_cache.json, acumatica\_token\_cache.json, and proposals.log beside the script—requires write permission there.

**Long-running service model**

* Designed to run indefinitely (while-True + sleeps). The designated PC/server usually has Task Scheduler/service wrappers to restart on boot/crash and keep it alive.

**Network & identity assumptions**

* Corporate firewalls/allowed-IP lists and machine credentials may be pre-approved for Egnyte/Acumatica. Running from any random desktop could fail auth or hit additional MFA/conditional access.

**Egnyte / filesystem conventions**

* Hardcoded Egnyte paths like /Shared/proposals/2025 and the template folder assume a specific tenancy and permissions assigned to the service identity used on that host.

**[MasterProjects.py Documentation](#Main)**

**[1.) Overview](#Main)**  
This script provisions and maintains Egnyte project folders for non-BPL projects and keeps Acumatica’s EgnyteFolder field in sync. On a 5-minute loop it:

• Authenticates to Acumatica and Egnyte (with simple local token caching).  
• Pulls all projects, then filters to **Active**, **ProjectID** starts with **’25’**, **Branch != BPL**, and **EgnyteFolder != 'Created'**.  
• Builds the destination folder path from a department code in the **ProjectID** (e.g., 25.04… → "25.04.000 - Knoxville").  
• Chooses a template (from a CSV mapping, with special handling for **Series Projects** and **Telecom**) and copies it into Egnyte if the folder doesn’t exist.  
• Sets Acumatica **EgnyteFolder = "CD"** after confirming/creating the folder.  
• Sends a single alert email if a folder copy fails (per folder name).  
• Logs off Acumatica and repeats.

**[2.) Dependencies and prerequisites](#Main)**  
The script uses only standard libraries plus **requests** and **python-dotenv**.

**Python packages**

• requests (HTTP calls to Acumatica/Egnyte; timeouts; error handling)  
• python-dotenv (loads secrets from config.env)

**Standard library modules used (no install required)**

• logging, time, os, json, csv, email, smtplib, re, concurrent.futures, urllib.parse, functools, threading (via concurrent), etc.

**Environment keys expected in config.env**

• **Acumatica:** ACUMATICA\_CLIENT\_ID, ACUMATICA\_CLIENT\_SECRET, ACUMATICA\_USERNAME, ACUMATICA\_PASSWORD

• **Egnyte:** EGNYTE\_CLIENT\_ID, EGNYTE\_CLIENT\_SECRET, EGNYTE\_USERNAME, EGNYTE\_PASSWORD

• **Email (alerts):** SMTP\_SERVER, SMTP\_PORT (default 587), SMTP\_PASSWORD, SENDER\_EMAIL

**[3.) Network endpoints & fixed paths](#Main)**

**Endpoints**

• Acumatica token: https://bennett-pless.acumatica.com/identity/connect/token  
• Acumatica API base: https://bennett-pless.acumatica.com  
• Egnyte API base: https://bennettpless.egnyte.com/pubapi/v1  
• Egnyte OAuth: https://bennettpless.egnyte.com/puboauth/token

**Egnyte folders**

• **Template base (source):** /Shared/Team Member Resources/IT/ProjectFolderTemplates/<TemplateName>

◦ Special cases: **Series Projects Child**, **Niche Tower Project Template**

• **Destination base (target):** /Shared/Projects/2025/{Department}/{FolderName}

◦ Department is resolved from the ProjectID’s middle segment via the built-in department\_mapping.

**Local/Windows paths the script assumes exist**

• Template map CSV: C:/Scripts/Regular Projects/TemplateGuide.csv (used by determine\_template\_path)

**Local files created (working directory)**

• acumatica\_token\_cache.json — access token, session id, expiration timestamp  
• egnyte\_token\_cache.json — Egnyte access token + timestamp  
• application.log — log output (INFO level)  
• projects.csv — raw export of project data from Acumatica (overwritten on each retrieval)

**[4.) Operational notes](#Main)**  
**Loop cadence**  
Runs every **5 minutes** (sleeps **300s** between iterations).

**Template selection**

* **Primary rule:** Use TemplateGuide.csv (maps ConditionSet → TemplateName).
* **Special cases:**
  + **Series projects** (AddToExistingSeries = True): copy **“Series Projects Child”** under the master project folder.
  + **Telecom** (Market = Infrastructure **and** SubMarket = Telecom Structures): copy **“Niche Tower Project Template.”**

**Folder naming**

* **Master project:**  
  {Department}/{ProjectID} - {ProjectName}
* **Standard project:**  
  {Department}/{ProjectID} - {ClientCustomerID} - {ProjectName}

**Retries & timeouts**

* **Egnyte auth:** wait **5s** before auth; honor **Retry-After** on **429**; max **3** attempts.
* **Folder copy:** **30s** timeout; retry **once** on timeout; if Egnyte rejects for trailing space/period, sanitize and retry **once**.

**State & caching**

* Acumatica and Egnyte tokens cached in local JSON files.
* Egnyte token considered fresh for ~**3500s**.

**Acumatica update**

* After detecting/creating the folder, set EgnyteFolder = "CD".

**BPL branch**

* **Skipped** here (handled by a separate BPL-specific job).

**[5.) Logging & observability](#Main)**  
• Logs to stdout and application.log.  
• Writes a **one-time alert email per failing folder** (prevents spam) to **support@bennett-pless.com**.  
• Exports current project payload to **projects.csv** for troubleshooting/traceability.

**[6.) Why this script “only runs on a designated desktop”](#Main)**  
• **Absolute, machine-specific pathing:**  
Requires C:/Scripts/Regular Projects/TemplateGuide.csv. On other machines that path/drive or file won’t exist or be accessible.

• **Local secrets and token caches:**  
Expects a config.env in the working directory (or a known location) with production credentials. The designated box is where those are securely staged and protected; duplicating them to “any desktop” is a security risk.

• **Network egress and allowlists:**  
Outbound access to Acumatica, Egnyte, and SMTP may be IP-allowlisted, behind corporate proxies/TLS inspection, or tied to a managed device posture. A random desktop typically lacks those network exemptions.

• **Long-running service expectations:**  
It’s a daemon-style loop (while True + sleeps). The designated desktop/server is configured with Task Scheduler/service wrappers, restart-on-boot, log rotation, monitoring, and backups for the working directory.

• **Filesystem permissions & conventions:**  
The process must write application.log, the token caches, and projects.csv in the current working directory. The designated host guarantees those permissions and includes the required folder structure.

Want me to export this into a Word doc with a clickable TOC in the same style as your other docs?

**[MasterOpportunities.py Script Documentation](#Main)**

**[1.) Overview](#Main)**  
This script provisions and maintains Egnyte folders for Opportunities and keeps Acumatica’s EgnyteFolder field in sync. On a 10-minute loop it:

• Authenticates to Acumatica and Egnyte (simple local token caching).  
• Pulls Opportunities, then filters to Status in {"New","Open"}, OpportunityID starts with "O.25", and EgnyteFolder != "Created".  
• Builds the folder name as {OpportunityID} - {BusinessAccount} - {Subject}.  
• Copies the Opportunities template into Egnyte (if the folder doesn’t exist).  
• Sets Acumatica EgnyteFolder = "Created" after confirming/creating the folder.  
• Logs off Acumatica and repeats.

**[2.) Dependencies and prerequisites](#Main)**  
The script uses only standard libraries plus requests and python-dotenv.

**Python packages**

• requests (HTTP calls to Acumatica/Egnyte; timeouts; error handling)  
• python-dotenv (loads secrets from config.env)

**Standard library modules used (no install required)**

• logging, time, os, json, csv, urllib.parse, concurrent.futures, functools, etc.

**Environment keys expected in config.env**

• Acumatica: ACUMATICA\_CLIENT\_ID, ACUMATICA\_CLIENT\_SECRET, ACUMATICA\_USERNAME, ACUMATICA\_PASSWORD

• Egnyte: EGNYTE\_CLIENT\_ID, EGNYTE\_CLIENT\_SECRET, EGNYTE\_USERNAME, EGNYTE\_PASSWORD  
*(No email settings are required for this script.)*

**[3.) Network endpoints & fixed paths](#Main)**  
**Endpoints**

• Acumatica token: https://bennett-pless.acumatica.com/identity/connect/token  
• Acumatica API base: https://bennett-pless.acumatica.com  
• Egnyte API base: https://bennettpless.egnyte.com/pubapi/v1  
• Egnyte OAuth: https://bennettpless.egnyte.com/puboauth/token

**Egnyte folders**

• Template base (source): /Shared/Team Member Resources/IT/ProjectFolderTemplates/Opportunities  
• Destination base (target): /Shared/Opportunities/2025/{FolderName}

**Local/Windows paths the script may reference**

• Optional template map CSV (not required): C:/Scripts/Opportunities/OpportunitiesTemplateGuide.csv

**Local files created (working directory)**

• acumatica\_token\_cache.json — access token, session id, expiration timestamp  
• egnyte\_token\_cache.json — Egnyte access token + timestamp  
• opportunities.log — log output (INFO level)

**[4.) Operational notes](#Main)**  
**Loop cadence**

• Perpetual loop; sleeps 600s (10 min) between iterations.

**Folder naming**

• {OpportunityID} - {BusinessAccount} - {Subject}

**Template selection**

• Copies from a single Opportunities template folder (no CSV-driven mapping or department logic).

**Retries & timeouts**

• Egnyte auth: 5s pre-sleep; honors Retry-After on 429; max 3 attempts.  
• Opportunity fetch: basic retry on empty bodies/JSON decode errors.  
• Folder copy: uses Egnyte copy API; if the destination exists, creation is skipped. *(No advanced name sanitation in this script.)*

**State & caching**

• Acumatica and Egnyte tokens cached in local JSON files; Egnyte token considered fresh for ~3500s.

**Acumatica update**

• After detecting/creating the folder, sets EgnyteFolder = "Created" on the Opportunity.

**[5.) Logging & observability](#Main)**

• Logs to stdout and opportunities.log.  
• No email alerts and no CSV exports in this script.  
• API errors are printed and the loop continues on next cycle.

**[6.) Why this script “only runs on a designated desktop”](#Main)**

**Local secrets & token caches:**

Expect a config.env and writes JSON token caches in its working directory; those credentials live on the managed host.

**Network egress/allowlists:**

Outbound access to Acumatica/Egnyte may be IP-allowlisted or behind corporate proxies—granted to the designated machine.

**Daemon-style operation:**

It’s a long-running loop intended to be wrapped by Task Scheduler/service tooling on that host (restart-on-boot, log handling, backups).

**Filesystem permissions & structure:**

The process must write logs/caches and assumes the Egnyte share layout and optional local script folders exist on that box.

**[LucidEgnyteLnk.py Documentation](#Main)**

**[1.) Overview](#Main)**

This script watches **Egnyte project city folders** on Z:\Shared\Projects\2025\… and automatically provisions the **matching LucidLink project folder** on L:\Revit\2\_PROJECTS\2025\… when a new project folder appears. For each newly created project folder that matches the naming convention, it:

* Detects the new folder (live **watchdog** events plus a **5-minute polling** safety net).
* Figures out which **city/department** the folder belongs to from its path.
* **Copies a Revit project template** into LucidLink if the destination doesn’t exist yet.
* **Copies General Notes** content into the new project’s 4\_General Notes subfolder.
* Creates **mutual Windows shortcuts (.lnk)** between the Egnyte and LucidLink locations.
* Logs activity and continues running as a background daemon until stopped.

Folder names must match: YY.DD.DDD - … (regex: \d{2}\.\d{2}\.\d{3} - .+).

**[2.) Dependencies and prerequisites](#Main)**

The script uses the standard library plus one third-party package.

**Python packages**

* watchdog (filesystem event monitoring)

**Standard library modules (no install required)**

* logging, time, os, shutil, subprocess, re, threading, pathlib-style ops via os.path

**System & environment requirements**

* **Windows-only** (uses PowerShell + COM to create .lnk shortcuts).
* Mapped drives **L:\** (LucidLink) and **Z:\** (Egnyte).
* Permissions to read from Z:\, and read/write to L:\.
* Ability to run PowerShell commands and COM automation (WScript.Shell).

**[3.) Network endpoints & fixed paths](#Main)**

**Endpoints**

* **None**. This script operates on **mapped drives** only (no HTTP APIs).

**Egnyte (source)**

* City roots monitored: Z:\Shared\Projects\2025\{CityCode} (per city\_map).

**LucidLink (destination)**

* City roots: L:\Revit\2\_PROJECTS\2025\{CityCode} (per city\_map).

**Template & content**

* **Project template (copied to new Lucid project):**  
  L:\Revit\2\_PROJECTS\2024\24.08.000 - Raleigh\24.08.### - Arch name - Project name
* **General Notes source (copied into the project):**  
  L:\Revit\1\_BP\_REVIT\BP\_REVIT GENERAL NOTES\GEN NOTES - EXCEL
* **General Notes destination:**  
  {LucidProjectFolder}\4\_General Notes

**Logging**

* Log directory:  
  C:\Scripts\LucidEgnyteLNK\v3 - Triggers on Creation with Polling\LucidEgnyteLnk.log

**City mapping (sample)**

* "25.08.000 - Raleigh" → L:\Revit\2\_PROJECTS\2025\25.08.000 - Raleigh (and similar for other cities)

**[4.) Operational notes](#Main)**

**Startup**

* Configures file + console logging.
* **wait\_for\_drives**: up to **10 minutes** for L:\ and Z:\ to become available (checks every 10s).

**Monitoring strategy**

* **Watchdog observers**: one per city Egnyte folder (non-recursive) to catch **new directory** creations instantly.
* **Polling thread**: every **300s (5 min)**, scans the Egnyte city folders and simulates a creation event for anything not yet processed—covers missed events or service hiccups.
* **De-duplication**: a processed\_folders set prevents double work if both the watcher and poller see the same folder.

**Creation pipeline (on a new project folder)**

1. Verify the folder name matches \d{2}\.\d{2}\.\d{3} - .+.
2. Determine city by matching the **city code segment** in the folder’s path.
3. Build the **Lucid project folder path** and check if it exists.
4. If missing, **copy the template** (shutil.copytree), tolerant of race conditions (FileExistsError is logged and ignored).
5. **Copy General Notes** contents into 4\_General Notes (files and subfolders).
6. Create **both shortcuts** using PowerShell + COM:
   * Egnyte → .lnk inside the new Lucid folder
   * Lucid → .lnk inside the new Egnyte folder  
     (Shortcut attributes set to Archive,ReparsePoint.)
7. Log a success line; any unexpected exceptions are caught and logged without killing the thread.

**Resilience**

* Broad try/except blocks in event and poll loops **prevent thread crashes**.
* Safe handling of already-existing template copies and missing paths.
* Continues the infinite loop until **Ctrl+C**; on exit, all observers are stopped and joined.

**[5.) Logging & observability](#Main)**

* Writes to **console** and a rotating **file log** at the configured path.
* Logs key milestones: drive readiness, watcher start, detected folders, template/notes copy, shortcut creation, and polling cycles.
* Errors include stack traces where helpful (exc\_info=True) to aid troubleshooting.

**[6.) Why this script “only runs on a designated desktop”](#Main)**

* **Mapped drives & paths**: Hard-coded L:\ and Z:\ must exist and remain mounted; the designated host guarantees this.
* **Windows integration**: Depends on **PowerShell** and **COM** (WScript.Shell) to create .lnk files—**Windows-only**.
* **Long-running service**: Intended to run continuously via **Task Scheduler** or a service wrapper with restart-on-boot, log retention, and monitoring.
* **Permissions & performance**: Needs reliable read/write on both shares and low-latency access to copy templates and notes quickly.

[Recommendations for Scripts](#Main)

**Rich timestamps & phase timing**

* Add start/finish logs for every major phase (auth, fetch, filter, copy, update) with durations.
* Replace bare print() with structured logging at INFO/DEBUG/ERROR.
* Example:
  + INFO Fetch: starting proposals…
  + INFO Fetch: completed in 12.4s (records=187)
  + INFO Copy: 25.08.123 – Raleigh – Acme HQ (attempt=1, template=Series Projects Child)

**“Still working” / progress visibility**

* Emit a **heartbeat** line every 60s while a long step runs: INFO Heartbeat: still processing, elapsed=300s, processed=41/187.
* Add a **stall warning** if a step exceeds a threshold (e.g., fetching proposals > 120s): WARN Fetch appears slow (elapsed=129s)….
* Keep a lightweight status.json (updated each pass) with: last step, last record id, processed counts, last error.

**Concurrency audit**

* Verify whether ThreadPoolExecutor (or any multiprocessing) is in use; if not, either:
  + Remove the unused imports, **or**
  + Parallelize safe, I/O-bound work (e.g., Egnyte copy calls) with a small worker pool (e.g., 4–8 workers) and per-host rate limits.
* Add per-task contextual logging so parallel work is traceable.

**Errors → logs and email**

* Standardize errors to include: record id, client name, folder path, template, attempt #, HTTP status/text, elapsed.
* On ERROR, log full stack (exc\_info=True) **and** send a concise alert to **support@bennett-pless.com** with pertinent fields:
  + Subject: Egnyte provision failed: 25.08.123 – Raleigh – Acme HQ
  + Body: record id, client, project/proposal info, target folder, template used, error text, next retry plan.
* Deduplicate emails per folder (one alert per unique folder per run window) to avoid spam.

**Egnyte naming guardrails (invalid characters)**

* Document and enforce a **bad-character policy** (to be confirmed with Egnyte):
  + Likely invalid anywhere: < > : " / \ | ? \*
  + Disallow **leading/trailing** spaces or periods.
  + Normalize double spaces and weird Unicode (optional).
* Maintain a single sanitizer:
  + Strip leading/trailing spaces/periods.
  + Replace invalid chars with \_ (or remove).
  + Collapse repeated separators/spaces.
* Log the **original → sanitized** name when a change occurs.

**Specific case: P.25.1147**

* Periods **inside** names are usually okay; **trailing** periods are not.
* Implement two-stage handling:
  1. Try with the original name.
  2. On 400/422 with a known Egnyte message (e.g., “File names cannot contain leading or trailing spaces” or “Invalid characters”):
     + Run sanitizer, **retry once**, and include “sanitized-retry” in the log context.
     + If still failing, raise and alert with full diagnostics.

**Pre-scan vs. on-failure policy**

* Add a config switch: NAME\_VALIDATION\_MODE = pre\_validate | on\_failure | off.
  + **pre\_validate**: sanitize *before* any Egnyte call (least noisy).
  + **on\_failure**: attempt clean name only *after* the first failure (most transparent).
  + **off**: no automatic fix; just error + email.
* Always record whether the name was altered and why.

**Make liveness obvious**

* On each loop, emit a banner: ===== pass start (ts=2025-02-07T14:03:22Z) =====
* At loop end: ===== pass end (elapsed=2m31s, processed=17, created=12, skipped=5, errors=0) =====
* Optional: write/update a last\_run.txt with a human-readable summary so ops can check at a glance.

**Known disallowed characters (to confirm)**

* Add a **doc table** listing each disallowed character and the replacement/removal rule.
* Action item: confirm with Egnyte documentation/support and update the sanitizer + docs accordingly.

**Nice-to-have automation**

* If a bad character is detected (per policy), either:
  + **Auto-fix & continue** (log + optional email “auto-fixed”), **or**
  + **Skip & alert** (create a ticket / send email with all metadata so IT can correct at the source).
* Consider a small retry budget with backoff for Egnyte copy (e.g., timeout=30s, max 2 tries) with clear “Attempt n/2” logs.

[Requirements. Text](#Main)

# HTTP client

requests~=2.31.0

# Load environment variables from .env

python-dotenv~=1.0.1

# Microsoft Graph auth (include only if you actually use Graph)

msal~=1.25.0

# Time zones (legacy pytz; fine if your code relies on it)

pytz>=2023.3

# Geo utilities (e.g., great-circle distance)

geopy~=2.4.0

# File system watcher + CLI (watchmedo)

watchdog[watchmedo]~=3.0.0