**Technical Documentation: Alpha/Beta Command Center**

**Overview**

The Product Operations Alpha/Beta Command Center is a web-based application designed to streamline various tasks related to client recruitment and communication for Alpha/Beta programs. It provides distinct tools accessed via a central landing page:

1. **List Generation**: Generates new client recruitment lists using natural language prompts processed by Snowflake Cortex AI.
2. **List Scrubbing/Filtering**: Filters existing client lists based on specified criteria, also leveraging Cortex AI.
3. **Marketing Content Generation**: Creates templated DOCX invite documents (e.g., Opt-In/Opt-Out invites) populated with data fetched from Snowflake based on a selected list ID.
4. **Qualtrics Survey Creation**: Automates the creation of standardized Opt-In/Opt-Out surveys in Qualtrics based on feature details, either manually entered or derived from generated lists.

The system integrates a Node.js backend server, HTML/JavaScript frontend, PowerShell and Python scripts for orchestration and specific tasks, and leverages Snowflake for data storage, processing, and AI capabilities. Background processes (Python jobs and Snowflake tasks) ensure underlying data tables are kept up to date. **User authentication is managed via server-side sessions initiated on the home page.**

**Main Components**

**1. Web Application (HTML + JavaScript)**

* **Purpose**: Provides user interfaces for the different tools.
* **Files**:
  + home.html: Central landing page linking to individual tools. **Includes a username input form for session initiation and a logout mechanism.**
  + list\_tools.html: UI for List Generation and List Scrubbing. Features interactive prompt building, context list input, file naming, filter selection, and a modal for scrubbing options (including a "Save to Downloads" toggle). **No longer requires username input directly; relies on active session.** Includes a client-side check to redirect to home if no session exists.
  + marketing\_content.html: UI for generating DOCX invites. **Dynamically loads/displays lists associated with the logged-in user's session.** Includes search functionality for other lists/features, and a modal for template selection. **No longer requires username input directly; relies on active session.** Includes a client-side check to redirect to home if no session exists.
  + qualtrics\_survey.html: UI for Qualtrics survey creation. **Dynamically loads/displays lists associated with the logged-in user's session.** Includes a manual form for providing feature details directly. **No longer requires username input directly; relies on active session.** Includes a client-side check to redirect to home if no session exists.
* **Functionality**:
  + Collects user inputs (prompts, file names, filters, feature details, etc.).
  + **Initiates user session on home.html via /set-username endpoint.**
  + **Checks for active session on tool pages via /check-session endpoint and redirects if inactive.**
  + Dynamically constructs prompts based on selected filters (List Tools).
  + Fetches data via Node.js backend (e.g., user lists for the session user, feature definitions).
  + Submits data via POST requests to specific Node.js endpoints for processing.
  + Displays results, confirmation messages, or error messages returned from the server.
  + Handles user logout via /logout endpoint.

**2. Node.js Server (server.js)**

* **Purpose**: Acts as the central backend API and orchestrator, bridging the frontend UI with backend scripts and database interactions. Manages user sessions.
* **Framework**: Express.js
* **Middleware**: express.static (serves HTML/JS/CSS), body-parser (handles form data), **express-session (manages user sessions via cookies)**, custom request logger (logs session username), requireUsername (custom middleware protecting specific API endpoints).
* **Configuration**:
  + Hardcoded Paths: Contains file paths for PowerShell/Python scripts, DOCX templates, and DOCX output directory. (NOTE: Consider making configurable).
  + Snowflake Credentials: Retrieves Snowflake credentials (API\_USERNAME, API\_PASSWORD) from process environment variables (process.env).
  + **Session Secret**: Reads SESSION\_SECRET from environment variables for signing session cookies. **CRITICAL for security and session persistence.**
* **Key API Endpoints**:
  + /set-username (POST): Receives username from home.html, validates it, and stores it in req.session.username. Creates the session cookie.
  + /check-session (GET): Checks if req.session.username exists and returns { loggedIn: true/false, username: ... }. Used by frontend pages to verify session status.
  + /logout (POST): Destroys the current session on the server and clears the session cookie on the client.
  + /run-powershell (POST): **Protected by requireUsername**. Spawns list\_generation\_power\_shell.ps1. **Passes username from req.session.username**, prompt, filename.
  + /run-list-filter (POST): **Protected by requireUsername**. Spawns list\_filter\_power\_shell.ps1. **Passes username from req.session.username**, prompt, filename, and saveToDownloads flag.
  + /generate-content (POST): **Protected by requireUsername**. Handles Marketing Content generation. Parses list ID, fetches feature data from cr\_alpha\_content (via getFeatureContentData), uses docxtemplater, saves DOCX, and logs the event (using req.session.username).
  + /get-lists (GET): **Protected by requireUsername**. Fetches metadata of lists generated by the user stored in req.session.username from cr\_user\_requests (via getAllListMetadata, **formats username to user@domain for query**).
  + /search-features-by-criteria (GET): Searches for feature definitions in cr\_alpha\_content. (Currently unprotected by session).
  + /get-feature-name (GET): Fetches client-facing name for a feature key. (Currently unprotected by session).
  + /create-qualtrics-survey (POST): **Protected by requireUsername**. Spawns create\_qualtrics\_survey.py. Passes survey type, feature details, and the **username from req.session.username (formatted without domain for Python script)**. Returns JSON from Python script.
* **Direct Snowflake Interaction**: Uses the odbc package for /generate-content, /get-lists, /search-features-by-criteria, /get-feature-name, and internal logging (via helper functions). Helper functions (getAllListMetadata, logContentCreation) ensure the correct username format (user@domain.com) is used for DB operations.

**3. PowerShell Script (list\_generation\_power\_shell.ps1) - List Generation**

* **Purpose**: Orchestrates new list generation.
* **Parameters**: $var1 (username, **now provided by Node.js session**), $var2 (prompt), $var3 (base filename).
* **Functionality**: Retrieves machine credentials, connects via ODBC, calls alpha\_beta\_list\_generation SP, processes response, modifies/wraps SQL, executes query, logs results to alpha\_beta\_list\_generation\_results, exports Excel to OneDrive, logs request to cr\_user\_requests.

**4. PowerShell Script (list\_filter\_power\_shell.ps1) - List Scrubbing/Filtering**

* **Purpose**: Orchestrates list scrubbing.
* **Parameters**: $var1 (username, **now provided by Node.js session**), $var2 (prompt/filters), $var3 (base filename), $SaveToDownloads (Switch).
* **Functionality**: Retrieves machine credentials, connects via ODBC, calls alpha\_beta\_list\_generation\_filter SP, processes response, modifies/wraps SQL, executes query, logs results to alpha\_beta\_list\_generation\_results, exports Excel (conditional location), logs request to cr\_user\_requests.

**5. Python Script (create\_qualtrics\_survey.py)**

* **Purpose**: Creates a Qualtrics survey and logs the event.
* **Execution**: Invoked by Node.js server via spawn.
* **Parameters (Command Line)**: Survey Type, Feature Number, Alpha/Beta Stage, Client-Facing Feature Name, **Requesting Username (provided by Node.js session, likely without domain)**.
* **Functionality**: Loads QSF template, gets Qualtrics/Snowflake env var credentials, updates QSF, calls Qualtrics API, **looks up Qualtrics User ID for sharing based on provided username**, attempts to share survey, logs details to cr\_opt\_in\_out\_surveys, prints JSON result (including sharingStatus) to stdout.

**6. Snowflake Stored Procedure (alpha\_beta\_list\_generation)**

* **Purpose**: Generates a full SQL query using Snowflake Cortex AI based on a natural language prompt for new list generation.
* **Parameters**: PROMPT\_INPUT, USER\_ENTERED.
* **Functionality**: Extracts forced context IDs, constructs detailed Cortex prompt (schema, filtering logic, limits, feature exclusions, sampling, UNIONs), executes SNOWFLAKE.CORTEX.COMPLETE, cleans SQL, returns JSON with generated\_sql and forcedContextIDs.

**7. Snowflake Stored Procedure (alpha\_beta\_list\_generation\_filter)**

* **Purpose**: Generates SQL filtering logic (WHERE clause) using Cortex AI for scrubbing.
* **Parameters**: PROMPT\_INPUT, USER\_ENTERED.
* **Functionality**: Constructs Cortex prompt focused on WHERE clause generation based on schema and user description, handles feature exclusion logic, executes SNOWFLAKE.Cortex.COMPLETE, cleans SQL, returns JSON with generated\_sql.

**8. Snowflake Database**

* **Purpose**: Central data warehouse.
* **Key Tables**:
  + alpha\_beta\_list\_generation: Source client/context data.
  + cr\_full\_invite\_list: Previous invite history.
  + alpha\_beta\_list\_generation\_results: Data output logs from PS scripts. **[Metadata Table]**
  + cr\_user\_requests: Request metadata logs from PS scripts. **[Metadata Table]**
  + cr\_alpha\_content: Feature content details (read by Node.js).
  + cr\_alpha\_beta\_content\_creation: Marketing DOCX creation logs (written by Node.js, **uses formatted user@domain username**). **[Metadata Table]**
  + cr\_opt\_in\_out\_surveys: Qualtrics survey creation logs (written by Python). **[Metadata Table]**
  + Staging Tables (e.g., CR\_ALPHA\_OPT\_INS, CR\_CLIENT\_LIST): Used by background processes.
  + UX\_PROD.SURVEY.qualtrics\_user: Qualtrics User ID lookup table (read by Python).

**9. Excel File Output (.xlsx)**

* **Purpose**: User-friendly export of generated/filtered lists and metadata.
* **Location**:
  + List Generation: Fixed OneDrive path (C:\...\Client List Pulls).
  + List Scrubbing: User's Downloads folder OR standard OneDrive path (based on flag).
* **Common Sheets**: Metadata, ALLCONTACTS, ExcludedContexts / ExcludedContextsDetails.

**10. DOCX File Output (.docx)**

* **Purpose**: Generated marketing invite documents.
* **Location**: Fixed OneDrive path (C:\...\Invite Drafts, defined by OUTPUT\_DIR\_PATH).
* **Generation**: Uses docxtemplater library in Node.js with templates, populated by data from cr\_alpha\_content.

**Authentication & Authorization**

* **Authentication**: Managed via **server-side sessions** (express-session).
  + User provides username on home.html.
  + Server verifies/stores username in req.session and issues a session ID cookie.
  + Subsequent requests include the cookie, allowing the server to retrieve the user's session data.
  + No passwords are currently stored or used.
* **Authorization**: Basic authorization is implemented via the requireUsername middleware in server.js.
  + This middleware protects specific API endpoints (/run-powershell, /run-list-filter, /generate-content, /get-lists, /create-qualtrics-survey).
  + If req.session.username is not present, the middleware rejects the request with a 401 status.
  + Client-side JavaScript on tool pages checks /check-session on load and redirects to home if not logged in.

**Metadata Logging Summary**

Metadata and operational logs are written to the following Snowflake tables:

* **cr\_user\_requests**: Captures list generation/scrubbing requests (User from session, Prompt, SQL, Filename). Written by PS scripts.
* **alpha\_beta\_list\_generation\_results**: Stores actual data rows returned by PS list/scrub queries (RUN\_BY=User from session, RUN\_AT). Written by PS scripts.
* **cr\_alpha\_beta\_content\_creation**: Logs DOCX generation events (User from session **formatted as user@domain**, Timestamp, Type, List ID). Written by Node.js.
* **cr\_opt\_in\_out\_surveys**: Logs Qualtrics survey creation details (Feature details, Type, Name, Qualtrics ID/URL, Requesting User from session, Sharing Status, Timestamp). Written by Python script.

**Main Flows (Updated for Session)**

1. **Session Initiation**:
   * User visits home.html.
   * JS checks /check-session. If not logged in, shows username form.
   * User enters username, submits form to /set-username.
   * Node.js validates username, stores it in req.session, sends session cookie back.
   * JS updates UI, shows welcome message and tool links.
2. **List Generation Flow**:
   * User navigates to list\_tools.html. JS checks /check-session, proceeds if logged in.
   * User builds prompt, specifies filename components.
   * Frontend JS sends POST to Node.js /run-powershell (no username in payload).
   * Node.js requireUsername middleware verifies session.
   * /run-powershell handler retrieves username from req.session.
   * Node.js spawns list\_generation\_power\_shell.ps1, passing **session username**.
   * (Rest of PS/SP flow as before)
   * Node.js sends result back to frontend.
3. **List Scrubbing/Filtering Flow**:
   * User navigates to list\_tools.html, opens Scrub modal.
   * User enters filename components, pastes contexts, selects filters, toggles save location.
   * Frontend JS sends POST to Node.js /run-list-filter (no username in payload).
   * Node.js requireUsername middleware verifies session.
   * /run-list-filter handler retrieves username from req.session.
   * Node.js spawns list\_filter\_power\_shell.ps1, passing **session username**.
   * (Rest of PS/SP flow as before)
   * Node.js sends result back to frontend.
4. **Marketing Content Generation Flow**:
   * User navigates to marketing\_content.html. JS checks /check-session. If logged in, calls /get-lists.
   * Node.js requireUsername verifies session for /get-lists. Handler retrieves **session username**, formats it (user@domain), queries DB. Lists are displayed.
   * User selects list/searches, clicks "Generate Doc", selects template.
   * Frontend JS sends POST to Node.js /generate-content (no username in payload).
   * Node.js requireUsername middleware verifies session.
   * /generate-content handler retrieves **session username**. Parses list ID, fetches content via getFeatureContentData.
   * Node.js generates DOCX, saves file, calls logContentCreation (passing **session username**, which gets formatted to user@domain).
   * Node.js sends result back to frontend.
5. **Qualtrics Survey Creation Flow**:
   * User navigates to qualtrics\_survey.html. JS checks /check-session. If logged in, calls /get-lists.
   * Node.js requireUsername verifies session for /get-lists. Handler retrieves **session username**, formats it (user@domain), queries DB. Lists are displayed.
   * Option 1 (List): User clicks button on list row. JS fetches feature name.
   * Option 2 (Manual): User fills form, JS fetches feature name.
   * Frontend JS sends POST to Node.js /create-qualtrics-survey (no username in payload).
   * Node.js requireUsername middleware verifies session.
   * /create-qualtrics-survey handler retrieves **session username**.
   * Node.js spawns create\_qualtrics\_survey.py, passing survey details and the **session username (formatted without domain)**.
   * (Python script flow as before, including sharing attempt based on passed username)
   * Node.js sends Python result (including sharing status) back to frontend.

**Background Processes**

*(Based on previous documentation - verify details if possible)*

1. **Python Job (Opt\_Ins\_and\_Opt\_Outs.py)**
   * Frequency: Runs periodically (e.g., 3 times daily via Task Scheduler).
   * Purpose: Processes raw data files from various OneDrive locations and uploads cleaned data to Snowflake staging tables.
   * Functionality: Reads Excel files, cleans data, generates DDL, uploads DataFrames using snowflake-connector-python to staging tables.
   * Output: Populates staging tables used by Snowflake tasks.
2. **Snowflake Tasks**
   * Frequency: Daily (e.g., 9 AM EST).
   * Tasks:
     + ALPHA\_BETA\_LIST\_GENERATION\_TASK: Calls SP to refresh alpha\_beta\_list\_generation table.
     + CR\_FULL\_INVITE\_LIST\_TASK: Calls SP to truncate/reload CR\_FULL\_INVITE\_LIST from staging tables.

**Configuration & Dependencies**

* **Node.js**:
  + Requires Node.js runtime.
  + Dependencies: express, body-parser, express-session, odbc, pizzip, docxtemplater, cheerio.
  + Environment Variables: API\_USERNAME, API\_PASSWORD (Snowflake), **SESSION\_SECRET (Critical)**.
  + Hardcoded Paths: Script paths, template dir, output dir in server.js.
* **PowerShell**:
  + Requires PowerShell environment.
  + Dependency: ImportExcel module.
  + Environment Variables: API\_USERNAME, API\_PASSWORD (read from Machine scope).
* **Python**:
  + Requires Python 3 runtime.
  + Dependencies: requests, snowflake-connector-python.
  + Environment Variables: QUALTRICS\_API\_TOKEN, QUALTRICS\_DATACENTER\_ID, API\_USERNAME, API\_PASSWORD.
  + Hardcoded Paths: Base template path in create\_qualtrics\_survey.py.
* **Snowflake**:
  + Requires configured Snowflake account, database, schema, warehouse, role, user.
  + ODBC Driver: SnowflakeDSIIDriver must be installed/configured on machine running PS scripts and Node.js server.

**Summary**

The Product Operations Alpha/Beta Command Center uses a Node.js/Express backend with server-side sessions (express-session) for user persistence. Users set their username once on the home page. This username is stored in a session cookie and used by the backend to authorize API requests and retrieve/log user-specific data. The frontend includes checks to redirect unauthenticated users to the home page. Core functionalities (list generation/scrubbing, content generation, Qualtrics creation) now leverage the session username instead of requiring manual input on each tool. The system continues to rely on PowerShell, Python, Snowflake (including Cortex AI), and background jobs for its operations. Proper configuration of environment variables (especially SESSION\_SECRET) and paths remains critical.