ChatGPT

Prime TS BD Automation – Final Project Artifacts

Project Timeline (Major Milestones & Decisions)

Date	Event Summary	Engine	Source (Chat/ Document)	Drive Folder Path
June 2025	**5-Engine Architecture Adopted:** Defined five core BD automation engines (Scraper, Program Mapping, Org Chart, Playbook Generator, Scoring) as the project's foundation [1]. This established an integrated pipeline for data-driven BD intelligence.	*Cross-Engine*	*Step 1 – Decisions Made* [1] [2]	Prime TS BD Automation/ Architecture & Planning
July 2025	**Scraper Approach Finalized:** Chose Apify (Puppeteer) actors for job data extraction (e.g. competitor job boards, LinkedIn) instead of custom HTTP scrapers [3]. Compiled a master list of target competitor career site URLs/ATS types [4] and implemented multi-keyword search loops (e.g. "Secret", "TS/SCI") with in-run de-duplication [5]. Set daily scrape schedule with proxy/throttle safeguards [6].	*Scraper Engine*	*Step 1 – Decisions Made* [3] [7]	Prime TS BD Automation/ Engine 1 – Scraper
July 2025	**Program Mapping Targets Set:** Identified **Top 20 DoD programs** (≥$100M subcontract spend) as primary mapping targets [8]. Created a "Top 20 Programs" list and developed program trackers for each, capturing primes, key roles, and notes [9]. Designed mapping logic to auto-tag scraped jobs with likely program names using curated keywords and FPDS/ USAspending data (ready to deploy once job data flows) [10].	*Program Mapping*	*Step 1 – Decisions Made* [8] [11]	Prime TS BD Automation/ Engine 2 – Program Mapping

Date	Event Summary	Engine	Source (Chat/ Document)	Drive Folder Path
July 2025	**Org Chart Blueprint Defined:** Outlined methodology to infer org structures from hiring data [12]. Decided to enrich internal Bullhorn contacts with external data (LinkedIn scraping via Apify actors) [13]. Mapped Bullhorn contact extraction schema (PMs, leads per program) to feed Org Chart Engine [14]. Deferred full build until Scraper/Mapping provide inputs, but established standard org-chart output format for consistency [15].	*Org Chart Engine*	*Step 1 – Decisions Made* [13] [16]	Prime TS BD Automation/ Engine 3 – Org Chart
Aug 2025	**Sales Playbook v1 Completed:** Compiled the **Prime TS Sales Prospecting Playbook** – a comprehensive outreach playbook with call scripts, email templates, rebuttals, and meeting frameworks tailored to cleared programs [17] [18]. This Playbook (v1, finalized) became the content foundation for the Playbook Generator engine [19]. Began integrating these materials so GPT-4 can auto-summarize and customize them per target program.	*Playbook Engine*	*Step 1 – Decisions Made* [17] [19]	Prime TS BD Automation/ Engine 4 – Playbook
Aug 2025	**Scoring Model Designed:** Developed a **weighted opportunity scoring model** with agreed factors (e.g. demand 30%, pain 20%, contract value 20%, past performance 20%, competition 10%) [20]. Set up a Postgres schema and fields to calculate scores (job counts, clearance difficulty, past performance, etc.) [21]. Created an Airtable “Program Priority” dashboard prototype showing example program scores [22]. Full automation of scoring to be implemented once data is populated.	*Scoring Engine*	*Step 1 – Decisions Made* [23] [22]	Prime TS BD Automation/ Engine 5 – Scoring

Date	Event Summary	Engine	Source (Chat/ Document)	Drive Folder Path
Aug 2025	**Initial Data Ingestion & Results:** Ran pilot scrapes (e.g. Fort Meade hub with clearance filters) yielding ~124 cleared job postings [24]. Expanded Scraper coverage from 18 to 26 metro hubs, including key DoD regions [25]. Consolidated all competitor job board URLs into a JSON config for daily n8n cron runs [26]. Began feeding results into Airtable and aligned fields with the planned database schema [27].	*Scraper Engine*	*Step 3 – Current State* [24] [28]	Prime TS BD Automation/ Engine 1 – Scraper
Aug 2025	**Full Stack Environment Setup:** Built out infrastructure for the integrated system. Created a GitHub repository scaffold ("Primetime BD IntelliRepo") with sub-folders for each engine and pre-populated README configs [29]. Prepared a unified *.env template* (API keys for Apify, LinkedIn, Airtable, etc.) [30] and a **Docker Compose** file with services for n8n, Postgres (with pgVector), PgAdmin, and optional Qdrant (vector DB) [31]. Successfully test-launched containers (n8n, DB) and custom scraper service.	*Cross-Engine*	*Step 1 – Decisions Made* [29] [32]	Prime TS BD Automation/⚙ Infrastructure
Aug 2025	**Competitive Tactics & Events Integrated:** Published the **Competitive Capture & Insertion Strategy** playbook (PDF) detailing sub-to-prime "wedge" tactics against key competitors [33]. Incorporated event-driven BD: aligned outreach efforts with major **Q3–Q4 2025 defense conferences** (e.g. AUSA, TechNet, DISA) [34] – flagged as must-attend for intel gathering. Adjusted Playbook Engine to leverage event calendars and warm introduction opportunities from these events [35] [36].	*Playbook Engine* <br/>(+ Mapping)	*Step 1 – Decisions Made* [34] [37]	Prime TS BD Automation/ Engine 4 – Playbook

Date	Event Summary	Engine	Source (Chat/ Document)	Drive Folder Path
Aug 2025	**Integrated AI Workflows:** Finalized the **Agent Mode Prompt** for ChatGPT/LLMs – a master JSON prompt that orchestrates analysis of all strategy docs to produce a BD "Attack Plan" [38]. Enabled an OpenAI GPT-4 node in n8n to generate summaries of call notes and competitor intel as a proof-of-concept for the Playbook Engine [39]. Established a HUMINT feedback loop where recruiters log meeting notes which the AI will later analyze for playbook refinements [40] [41] (automation of this analysis is pending).	*Cross-Engine*	*Step 1 – Decisions Made* [38] [42]	Prime TS BD Automation/ Agent Workflows
Late Aug 2025	**Consolidated Current State & Next Steps:** Completed a full project state review (Step 3) documenting each engine's status [43] [20]. At this stage, Scraper and Playbook engines are partially operational, Mapping is designed awaiting data, Org Chart is outlined (on hold for data), and Scoring is formulated on paper [44] [20]. Identified remaining blockers (e.g. need Bullhorn contact data, implement scoring code) and prepared for Phase 2 build-out (automation of org charts, scoring, and multi-engine agent orchestration).	*Cross-Engine*	*Step 3 – Current State* [44] [20]	Prime TS BD Automation/ Architecture & Planning

Sources: Major milestones and decisions compiled from **"Step 1 – Decisions Made"** [3] [7], **"Step 3 – Current State by Engine"** [24] [20], and the Prime TS chat logs [45] [46].

Master Inventory (By Engine and Artifact)

Below is a comprehensive inventory of all project artifacts, organized by engine. Each artifact includes its title, type, engine, purpose, how it's used/integrated, storage path, hyperlink (if applicable), tags, suggested retrieval keywords, workflow phase, and cross-references to related items. *("Workflow Phase" corresponds to the BD lifecycle stage: Target Identification, Credibility/Enrichment, Insertion/Outreach, Scoring/Prioritization, or Maintenance/Infrastructure.)* **All artifacts are listed, including drafts and superseded versions (marked accordingly), to ensure nothing is omitted per the knowledge base** [47] [48] **.**

Scraper Engine (Engine 1 – Data Collection)

This engine is responsible for daily data collection of cleared job postings and related BD intel (phase: **Target Identification**). Artifacts include scraping workflows, configurations, and reference documents:

Title	Type	Engine	**Purpose** (Use Case)	**Integration Usage** (How It's Used)	Folder Path	Hyperlink
20250804_142008.pdf	Document	Scraper	Unidentified reference document (auto-saved page/email – content unclear) [49]. Included for completeness.	*Not actively used* – archived for reference (purpose unknown) [50].	Engine 1 – Scraper/ References	(No link – internal
AI Automation Methodologies & Blueprint (Nate Herk & AI Automation Society).pdf	Document	Scraper	Overview of no-code AI automation methodologies & best practices (from community guide) – used as general design reference for our pipeline [51].	Reference only – informs design of n8n workflows, RAG (retrieval-augmented) pipelines, and multi-agent integration across engines [52].	Engine 1 – Scraper/ References	AI Automation Blueprint.pdf <sup sup>
Installing Puppeteer on Windows (Docker Desktop Setup).pdf	Document	Scraper	Technical setup guide for running Puppeteer headless browser on Windows via Docker [53]. Aids environment configuration for scrapers.	Used during development to configure the Puppeteer environment for web scraping tasks (ensured our scrapers run reliably on dev machines) [54].	Engine 1 – Scraper/ Setup Guides	Puppeteer Docker Guide.pdf <sup>*

Title	**Type**	**Engine**	**Purpose** (Use Case)	**Integration Usage** (How It's Used)	**Folder Path**	**Hyperlink**
Competitor Job Board URL Master Table	Document (Table)	Scraper	Master list of competitor companies (Tier 1–3 cleared firms) with their careers page URLs and ATS platform (Workday, iCIMS, Greenhouse, etc.) [55]. Serves as the master input for scraping all targets.	**Ingested as config** for the Scraper Engine: n8n iterates through this list to drive daily scrapes for each competitor site [56]. Centralizes all target URLs to ensure full coverage.	Engine 1 – Scraper/ Configurations	Competitor URLs Master.xlsx <sup>
Optimized Coverage Hubs (18 cities)	List (Text/ CSV)	Scraper	Curated list of ~18 metro hub locations (initial geographic focus for job searches) provided by user. Ensures key regions (e.g. NCR, Huntsville, Colorado Springs, etc.) are covered.	Used in scraper workflows to loop through target locations for search queries [57]. Ensures daily scrapes cover all priority cities (100-mi radii) in rotation.	Engine 1 – Scraper/ Configurations	Coverage Hubs.cs <sup>*</sup>

Title	**Type**	**Engine**	**Purpose** (Use Case)	**Integration Usage** (How It's Used)	**Folder Path**	**Hyperlink**
Apify Actor Input (Fort Meade)	JSON Config	Scraper	Example JSON configuration for an Apify actor to scrape Insight Global jobs around Fort Meade (with clearance keywords) [58]. Demonstrates the format for location-specific scrapers.	Runs on Apify platform (triggered via n8n HTTP calls). This JSON drives a daily Apify actor execution that searches jobs in the given area with clearance filters [59]. Integrated via n8n cron + Apify API for scheduled scraping.	Engine 1 – Scraper/ Workflows & Scripts	FortMeade_ApifyI <sup>*</sup>
Cleared Job Scraper (Example)	JSON Config	Scraper	**Draft** n8n workflow (JSON) for scraping a single source (e.g. TEKsystems jobs) and sending results to Airtable [60]. First prototype of the cleared jobs scraper.	Imported into n8n as a proof-of-concept workflow – scrapes one site and pushes to Airtable. **Superseded** by the multi-source workflow once logic expanded [61]. Retained for reference/ testing.	Engine 1 – Scraper/ Workflows & Scripts	ClearedJobScraper <sup>*</sup>

Title	**Type**	**Engine**	**Purpose** (Use Case)	**Integration Usage** (How It's Used)	**Folder Path**	**Hyperlink**
Cleared Jobs Workflow (Revised)	Workflow (n8n)	Scraper	**Revised end-to-end n8n workflow** for cleared job postings: scrapes multiple sources, enriches data, de-duplicates, and syncs results to Airtable [62] . This is the current main scraping workflow.	Implemented as an automated n8n workflow (triggered daily via cron). Iterates through all competitor job boards (using the config list), handles pagination & deduplication, then stores postings to Airtable [63] . This is the production scraper logic (v2).	Engine 1 – Scraper/ Workflows & Scripts	ClearedJobs_Work <sup>*</sup>
Scraper Config JSON	JSON Config	Scraper	Puppeteer/ Apify configuration JSON for a multi-keyword job search in one location (Fort Meade test config). Contains search terms, location, and crawler settings [64] .	Used by the Scraper Engine's Puppeteer script – defines parameters for test runs. Helps ensure search loops (multiple clearance keywords for one city) function correctly [65] . (This served as a template for building out the full workflow.)	Engine 1 – Scraper/ Configurations	SampleScraperCo <sup>*</sup>

Title	**Type**	**Engine**	**Purpose** (Use Case)	**Integration Usage** (How It's Used)	**Folder Path**	**Hyperlink**
Scraper Engine Config JSON	JSON Config	Scraper	**Master configuration file** mapping each target competitor to its careers page URL and ATS type (Greenhouse vs. Workday, etc.) [66] . This file drives dynamic routing in the scraper.	Consumed by the Scraper Engine's workflow to decide which scraping method to use per site [67] (e.g., use appropriate spider or actor for Workday vs. iCIMS). Central config ensures adding a new competitor only requires updating this JSON.	Engine 1 – Scraper/ Configurations	[ScraperEngine_Co]() <sup>*</sup>
Job Postings – Grid View.csv	CSV Data	Scraper	CSV export of job postings from Airtable ("Grid View"). Captures all fields of collected jobs in a flat file for schema alignment/ testing [27] .	Used as a reference to validate our job schema. This export was compared against our Postgres schema to ensure field consistency [68] . Also used for initial data loads before live pipeline was fully running.	Engine 1 – Scraper/ Data Exports	[Jobs_GridView.csv]() <sup>*</sup>

Title	**Type**	**Engine**	**Purpose** (Use Case)	**Integration Usage** (How It's Used)	**Folder Path**	**Hyperlink**
Jobs Records (Full)	Table (Excel)	Scraper	Complete set of job records parsed from a large exported text block (jobs portion of a placements document) [69]. Essentially a compiled list of all jobs for analysis.	Used for data loading/testing – this comprehensive table of job entries (from initial dumps) served as input to the engines (especially Program Mapping & Scoring) [70]. Ensured historical jobs were accounted for in the system.	Engine 1 – Scraper/ Data Exports	[Jobs_Full.xlsx]() <sup sup>
Jobs Table (Separate Doc)	Table (Excel)	Scraper	Job records parsed from a separately provided jobs document (a second source of job data) [71]. This was merged with the main "Jobs Records (Full)" to ensure no job entry was missed.	Supplementary input data merged into the **Jobs Records (Full)** table [72]. Ensured that jobs from multiple source documents were consolidated. Marked as duplicate since its content was merged.	Engine 1 – Scraper/ Data Exports	[Jobs_Additional.xl]() <sup>*</sup>

Title	Type	Engine	**Purpose** (Use Case)	**Integration Usage** (How It's Used)	**Folder Path**	**Hyperlink**
Tara Lopez.pdf	Document (PDF)	Scraper	Resume or LinkedIn profile PDF for an individual (Tara Lopez). Likely used as a sample for testing AI resume parsing or profile enrichment processes [73].	Used as a **test input** for the AI Resume Analyzer or profile-parsing workflow. Allowed validation that our parsing nodes (in n8n or Python) correctly extract info from a real resume [74]. Not part of production data flow, but helpful for QA of parsing logic.	Engine 1 – Scraper/ Test Data	TaraLopez_Profile. <sup>*</sup>

<small><sup>*</sup>Drive link placeholders – actual links accessible in Prime TS Google Drive.</small>

Program Mapping Engine (Engine 2 – Intelligence Tagging & Credibility)

Engine 2 tags scraped jobs with likely DoD programs and primes, building program intelligence for BD (**Target ID** and **Credibility** phases). Artifacts include program lists, past performance "credibility" docs, and data tables:

Title	**Type**	**Engine**	**Purpose** (Use Case)	**Integration Usage** (How It's Used)	**Folder Path**	**Hyperlink**
Bullhorn Contact Extraction Plan	Document	Program Mapping	Schema/plan for pulling Bullhorn CRM contacts for each target program (which roles to extract: PMs, tech leads, etc., by prime, program, location) [75]. Ensures we gather the right internal contacts for mapping.	Used to drive data pulls from Bullhorn (via API or export). Informs Org Chart/Mapping engines which contacts to import per program [76]. Essentially a spec so that when Bullhorn data is fetched, it's structured and filtered for our needs.	Engine 2 – Program Mapping/ Strategy Docs	[Bullhorn_ContactPlan]() <sup>*</sup>
Top 20 DoD Programs with High Subcontractor Staffing Spend.pdf	Document	Program Mapping	Analysis report of the **top 20 DoD programs** ranked by subcontractor staffing spend, including program descriptions and BD opportunities [77]. Defines the high-value programs Prime TS should pursue.	Used to prioritize BD targeting in the pipeline – the Mapping Engine tags new jobs with these program names and uses their priority rank to focus efforts [78]. Also serves as a reference list when scoring opportunities.	Engine 2 – Program Mapping/ Strategy Docs	[Top20_Programs.pdf]() sup>

Title	**Type**	**Engine**	**Purpose** (Use Case)	**Integration Usage** (How It's Used)	**Folder Path**	**Hyperlink**
Past Performance Made Easy Lists.pdf	Reference	Program Mapping	"Cheat sheet" lists of Prime TS's **past performance** highlights (contract names, success stories) for easy reference [79]. Aids credibility by quickly finding relevant experience to mention.	Referenced by Mapping & Playbook engines – provides ready-made past performance talking points. When generating proposals or outreach, the AI can pull relevant lines from these lists to bolster credibility [80].	Engine 2 – Program Mapping/ Credibility Docs	PastPerf_Lists.pdf <su
Prime NGC Past Performance.pdf	Battle Card	Program Mapping	Past performance "battle card" focused on **Northrop Grumman (NGC)** programs – lists key contracts and Prime TS successes related to NGC [81]. Demonstrates our relevant experience with this major prime.	Used by Playbook & Mapping engines for credibility insertion. When targeting a program involving NGC (either as incumbent or competitor), the system can pull tailored credibility blurbs from this card [82] – e.g. mentioning similar NGC projects Prime TS has delivered.	Engine 2 – Program Mapping/ Credibility Docs	NGC_BattleCard.pdf < sup>

Title	**Type**	**Engine**	**Purpose** (Use Case)	**Integration Usage** (How It's Used)	**Folder Path**	**Hyperlink**
Proposed Architecture for Prime TS BD Intelligence Automation.pdf	Document	Program Mapping	*Superseded* initial high-level design blueprint of the 5-engine BD automation system (conceptual architecture and data model) [83]. Created early in the project to visualize the solution.	**Superseded by final architecture.** Was used initially to guide development until updated plans emerged [84]. Retained for historical reference but replaced by the End-to-End Stack design.	Prime TS BD Automation/ Architecture & Planning	Proposed_Architectur <sup>*</sup>
End-to-End BD Automation Stack for Prime Technical Services.pdf	Document	Program Mapping	**Finalized architecture & tech stack** document for the integrated BD automation system [85]. Details how all five engines connect (data flow) and enumerates key technologies (n8n, OpenAI, Apify, Postgres, etc.) used.	**Primary reference for implementation** – guides how engines hand off data (Scraper → Mapping → Org Chart → Playbook → Scoring) and how each component is deployed [86]. Informs configuration of the overall pipeline and CI/CD. Supersedes the "Proposed Architecture" doc [87].	Prime TS BD Automation/ Architecture & Planning	Final_Architecture.pdf sup>

Title	**Type**	**Engine**	**Purpose** (Use Case)	**Integration Usage** (How It's Used)	**Folder Path**	**Hyperlink**
Hiring Trends and BD Target Analysis.pdf	Document	Program Mapping	"Heatmap" analysis of hiring trends and placement stats across agencies/ programs [88] . Identifies which programs and labor categories are hottest, highlighting gaps or opportunities for Prime TS.	Feeds both Mapping and Scoring: data from this analysis (e.g. surge in hiring for certain skillsets or clearance levels) is used to tag incoming jobs and also influences the Scoring Engine's prioritization [89] . Helps tailor outreach by showing where demand is high.	Engine 2 – Program Mapping/ Credibility Docs	HiringTrends_Analysis <sup>*</sup>
Engine Placements (Clean Headers)	Table (Excel)	Program Mapping	Placement records table with initial **standardized column headers** applied (cleaned field names per spec) [90] . First step of cleaning raw placement data.	Intermediate data artifact in the placement data pipeline. Provided a foundation for mapping placements to programs by ensuring consistent field names (e.g. normalized job_title, client_name) [91] . Used to verify header mapping before further cleanup.	Engine 2 – Program Mapping/ Data (Placements)	Placements_CleanHea <sup>*</sup>

Title	**Type**	**Engine**	**Purpose** (Use Case)	**Integration Usage** (How It's Used)	**Folder Path**	**Hyperlink**
Engine Placements (Remapped Headers)	Table (Excel)	Program Mapping	Placement records with headers remapped to the new schema (after deciding final field names) [92]. Reflects column name changes.	Another intermediate dataset showing the result of renaming columns. Used to ensure all fields now match the target schema for integration into Org Chart/ Scoring engines [93]. Basically a checkpoint confirming header alignment.	Engine 2 – Program Mapping/ Data (Placements)	Placements_Remappe <sup>*</sup>
Engine Placements (Clean Job Titles)	Table (Excel)	Program Mapping	Placement records after **removing recruiter names** from job titles (so titles are clean) [94]. This step produced a cleaned "Job Title" field.	Intermediate cleaning step – ensures job titles are standardized with no extraneous text. Important for Org Chart clustering (so that e.g. "Software Engineer" isn't treated differently due to a recruiter's name) [95].	Engine 2 – Program Mapping/ Data (Placements)	Placements_CleanTitl <sup>*</sup>

Title	**Type**	**Engine**	**Purpose** (Use Case)	**Integration Usage** (How It's Used)	**Folder Path**	**Hyperlink**
Engine Placements (Full Cleaned)	Table (Excel)	Program Mapping	**Full placement dataset, cleaned and normalized** with final schema applied (complete output of placement data cleansing) [96] [97] . This is the ready-to-use placement data.	**Primary cleaned dataset** for placements – used by Org Chart and Scoring engines as a key input [98] . Contains every placement with standardized fields, ready for analysis (e.g. count placements per program for credibility scoring) [99] .	Engine 2 – Program Mapping/ Data (Placements)	[Placements_FullClean]() <sup>*</sup>
Engine Placements (Full Parsed Block)	Table (Excel)	Program Mapping	Final structured table produced by parsing a raw pasted block of placement data (the entire dataset before cleaning) [100] . Essentially the initial parse of the placements document.	Baseline parsed placement data used at the **start of the cleaning pipeline** [101] . Contains all placement entries with original formatting; underwent header renaming and cleaning to become the "Full Cleaned" dataset. Serves as the raw input archive.	Engine 2 – Program Mapping/ Data (Placements)	[Placements_RawParse]() <sup>*</sup>

Title	**Type**	**Engine**	**Purpose** (Use Case)	**Integration Usage** (How It's Used)	**Folder Path**	**Hyperlink**
NTT Data Apps & BPS Tech Stack.pdf	Document	Program Mapping	Technical stack document detailing **NTT Data's Applications & BPS** offerings and technologies [102]. Provides insight into a competitor's capabilities.	Used for competitive intelligence in Mapping/Org Chart – offers context on NTT's tech focus areas [103]. Can inform program mapping by highlighting where Prime TS might differentiate or align capabilities when targeting programs involving NTT.	Engine 2 – Program Mapping/ Competitive Intel	NTT_TechStack.pdf <s sup>

Org Chart Engine (Engine 3 – Organization Mapping)

Engine 3 reconstructs organizational charts and key personnel relationships for target programs (phase: **Credibility** building, feeding into targeted outreach). Artifacts include methodology outlines, program-specific org info, contact lists, and data exports:

Title	**Type**	**Engine**	**Purpose** (Use Case)	**Integration Usage** (How It's Used)	**Folder Path**
Org Chart Engine – Org Design Outline	Document	Org Chart	Methodology outline for **reconstructing org hierarchies** from data [104]. Details how to cluster job titles, map contacts to programs, and infer reporting structure – blueprint for Org Chart Engine's approach.	Serves as the design spec for building the Org Chart Engine. Informs development of the custom n8n function or script that will group roles and link contacts to org units [105]. Used to guide coding of org-inference logic (role clustering rules, supervisor linking).	Engine 3 – Org Chart/ Strategy & Design

Title	**Type**	**Engine**	**Purpose** (Use Case)	**Integration Usage** (How It's Used)	**Folder Path**
GBSD BD Strategy.pdf	Strategy Deck	Org Chart	Business Development strategy deck for the **Sentinel/ GBSD program** – including org charts, key labor categories, subcontractor targets, and capture tactics [106]. Tailored strategic plan for this flagship program.	Used by Org Chart & Playbook engines when pursuing GBSD. Provides program-specific context (who's who, target roles, known pain points) [107]. The Playbook engine pulls talking points from this when generating outreach, and Org Chart uses it to validate inferred org structures.	Engine 3 – Org Chart/ Program-Specific Intel

Title	**Type**	**Engine**	**Purpose** (Use Case)	**Integration Usage** (How It's Used)	**Folder Path**
GBSD Notebook.pdf	Notebook	Org Chart	Collection of reference notes for the GBSD program – historical context, contract background, requirements, etc. [108]. Essentially a research notebook on GBSD.	Serves as a knowledge base for Org Chart/ Playbook engines. The AI can pull background facts from here to enrich playbooks or tailor conversations for GBSD [109]. Ensures outreach and strategy incorporate program history and context (improving credibility with clients).	Engine 3 – Org Chart/ Program-Specific Intel

Title	**Type**	**Engine**	**Purpose** (Use Case)	**Integration Usage** (How It's Used)	**Folder Path**
Sentinel GBSD Program Key Contacts Call Sheet.pdf	Call Sheet	Org Chart	One-page **call sheet of key contacts** for the GBSD program – lists names, titles, and notes for program PMs, hiring managers, etc. [110]. Guides BD reps on who to call and what to say.	Feeds the Org Chart engine's contact database for GBSD. These contacts are flagged as top outreach targets. The Playbook Generator uses this list to personalize scripts (e.g. referencing a contact's role or recent note in emails) [111]. Updated as new contacts are identified.	Engine 3 – Org Chart/ Program-Specific Intel
Ogden Org Overview.pdf	Document	Org Chart	Org structure overview for the GBSD program's **Ogden site** – describes teams or hierarchy at that location [112]. Provides insight into how that program is organized on-site.	Used to validate/ supplement the Org Chart Engine's output for GBSD-Ogden. When our engine infers an org chart from data, we compare against this reference to ensure accuracy [113]. Also helps identify any missing roles in our reconstruction.	Engine 3 – Org Chart/ Program-Specific Intel

Title	**Type**	**Engine**	**Purpose** (Use Case)	**Integration Usage** (How It's Used)	**Folder Path**
GBSD Ogden Contacts Job/ Function.pdf	Document	Org Chart	List of contacts on the GBSD program's Ogden teams – names, job titles, and possibly job functions [114]. Essentially the personnel roster for Ogden related to GBSD.	Ingested into the Org Chart Engine to enrich its contact list for GBSD. Ensures our system knows specific individuals at Ogden (so outreach can target them). The Playbook uses these names to personalize messaging (mentioning local team members) [115].	Engine 3 – Org Chart/ Program-Specific Intel

Title	**Type**	**Engine**	**Purpose** (Use Case)	**Integration Usage** (How It's Used)	**Folder Path**
Clearance Overview.docx	Guidance Doc	Org Chart	Overview of U.S. security clearance levels, terminology, and typical timelines (Public Trust vs. Secret/TS, reinvestigation periods, etc.) [116]. Used to educate team and inform processes.	Utilized by Org Chart & Playbook processes to inform conversations and planning [117]. E.g., the Playbook might insert notes about clearance timelines when discussing candidate availability; Org Chart uses it to consider clearance levels of contacts. Ensures BD staff/AI speak accurately about clearances.	Engine 3 – Org Chart/ References

Title	**Type**	**Engine**	**Purpose** (Use Case)	**Integration Usage** (How It's Used)	**Folder Path**
Docx Parsing Attempt (Placements)	Script (Py)	Org Chart	A one-off Python script that attempted to parse an initial placements .docx file (which was malformed) [118]. Early utility created to salvage data from the original placements document.	**One-time use** during data prep – tried extracting placement records via `python-docx`. Not part of the automated pipeline (since the doc was problematic), but it informed the team to obtain data in alternate formats [119]. Essentially a throwaway tool, kept for reference.	Engine 3 – Org Chart/ Data Processing Scripts
Engine_Placements_Updated.csv	CSV Data	Org Chart	Initial merged placement dataset (original export + first batch of new records integrated) [120]. Captures placements after first update cycle.	Intermediate combined data used to verify that newly added placements integrated correctly [121]. Served as input to subsequent cleaning steps; once the second update came, this was superseded by "Updated_Full".	Engine 3 – Org Chart/ Data (Placements)

Title	**Type**	**Engine**	**Purpose** (Use Case)	**Integration Usage** (How It's Used)	**Folder Path**
Engine_Placements_Updated_Full.csv	CSV Data	Org Chart	Extended placement dataset with additional records appended (after second update) [122] . Essentially all placements after incorporating another new batch.	Follow-up merged file capturing all placement records after the second integration of new data [123] . Used as a checkpoint before starting full data cleaning on the entire set. Marked duplicate since final cleaning moved to "All".	Engine 3 – Org Chart/ Data (Placements)
Engine_Placements_Cleaned.csv	CSV Data	Org Chart	Trimmed placement dataset with unnecessary columns removed (streamlined schema) [124] . A cleaned version focusing only on relevant fields.	Used to reduce noise before final integration. This cleaned CSV (with irrelevant columns dropped) made subsequent mapping and scoring more efficient [125] . It feeds into the final standardized dataset used by Org Chart & Scoring.	Engine 3 – Org Chart/ Data (Placements)

Title	**Type**	**Engine**	**Purpose** (Use Case)	**Integration Usage** (How It's Used)	**Folder Path**
Engine_Placements_Final.csv	CSV Data	Org Chart	Near-final placement dataset after schema alignment (almost complete; minor tweaks pending) [126]. Penultimate version of consolidated placements.	Used for final validation – checked that all fields match the target schema and data was ready for import to DB or scoring model [127]. Replaced by the fully consolidated "All" dataset once verification was done.	Engine 3 – Org Chart/ Data (Placements)
Engine_Placements_All.csv	CSV Data	Org Chart	**Fully consolidated final placement dataset** with all records under the standardized schema [128]. This is the authoritative placements data used in production.	Serves as the **master placement data source** for Org Chart and Scoring engines [129]. Contains every placement record, cleaned and normalized. Imported into the system's database or dataframes for use in org chart inference and scoring calculations (e.g. total placements per program) [130].	Engine 3 – Org Chart/ Data (Placements)

Title	**Type**	**Engine**	**Purpose** (Use Case)	**Integration Usage** (How It’s Used)	**Folder Path**
Cleared_Cable_Tech_Sourcing_Packet.pdf	Document	Org Chart	Example **sourcing packet** for a Cleared Cable Technician position (likely includes job req, candidate profiles, etc.) [131]. Demonstrates how candidates and requirements are presented for a cleared role.	Reference for sourcing strategies. Org Chart & Playbook engines may consult this to understand how technical roles are packaged – informing how to present candidates or craft outreach for similar positions [132]. Essentially a template for framing skill sets to clients.	Engine 3 – Org Chart/ References

Title	Type	Engine	**Purpose** (Use Case)	**Integration Usage** (How It's Used)	**Folder Path**
Michael Staff LinkedIn Activity (Export).pdf	Document	Org Chart	Export of LinkedIn activity feed for Michael Staff (target individual's posts, comments, etc.) [133]. Provides recent public activity of a key contact.	Input to the Playbook Engine's research step – an AI agent can summarize this feed to personalize outreach (e.g. referencing a topic the contact recently posted about) [134]. Essentially enriches the HUMINT playbook for that contact by adding timely talking points.	Engine 3 – Org Chart/ Program-Specific Intel

Playbook Generator (Engine 4 – Outreach Playbooks & HUMINT)

Engine 4 generates tailored prospecting playbooks and outreach content, combining data-driven insights with proven BD tactics (phase: **Insertion** – equipping BD reps for engagement). Artifacts include the master playbook document, competitive tactics, prompts, and agent outputs:

Title	**Type**	**Engine**	**Purpose** (Use Case)	**Integration Usage** (How It's Used)	**Folde**
Sales Prospecting Playbook.pdf	Document	Playbook	Comprehensive sales prospecting playbook – includes outreach scripts, cold-call openers, objection rebuttals, meeting cadence frameworks, etc., tailored to cleared BD [135]. Now enhanced with program-specific talking points.	Content **foundation for the Playbook Engine**. An OpenAI node uses this material to generate call/email scripts and talking points [136]. The Playbook Generator pulls from these proven templates when assembling program-specific playbooks, ensuring consistent messaging.	Engin Playb Maste Playb
Competitive Capture & Insertion Strategy for Prime Technical Services.pdf	Document	Playbook	Playbook/ strategy guide detailing tactics for inserting Prime TS into competitor-held contracts [137] – i.e. "wedge" strategies to subcontract under large primes, leveraging known competitor weaknesses or differentiators.	Used by Playbook & Mapping engines to plan targeted approaches. When a competitor is incumbent on a contract, the system references these tactics [138] (e.g. emphasize SDVOSB status, faster clearance processing) to suggest how Prime TS can position itself as a valuable partner or alternative.	Engin Playb Strate Guide

Title	**Type**	**Engine**	**Purpose** (Use Case)	**Integration Usage** (How It's Used)	**Folde**
Prime_TS_BD_Attack_Plan.pdf	Document	Playbook	**Draft BD "Attack Plan"** for Prime TS – a consolidated action plan output that includes a table of BD actions, narrative guidance, and a 30-day schedule of activities, synthesized from all engine outputs [139]. Essentially a master playbook generated by the AI agent.	This is the **combined plan** produced by the agent after analyzing all strategy artifacts. It distills targets, intel, and messaging into a concrete execution plan [140]. In practice, used in Maintenance phase: loaded into task trackers or shared with the team to ensure the recommended outreach steps are followed. (Draft status until fully validated.)	Engin Playb Agent Outpu
Agent Mode Prompt – Prime Technical BD Attack Plan	Prompt (JSON)	*Cross-Engine* <br/>(Playbook)	JSON-based **master prompt** that instructs the AI agent on how to analyze all BD strategy docs and generate a tailored Attack Plan [141]. It ensures the agent covers key areas (targets, pain points, action items) when producing the plan.	Executed in *Agent Mode* via the master prompting system. When triggered, the agent uses this prompt to read all relevant artifacts (docs, trackers) and output the Attack Plan (as seen in Prime_TS_BD_Attack_Plan.pdf) [142]. This prompt "template" ensures consistency and thoroughness in the plan generation.	Prime Autor Age Prom

Title	**Type**	**Engine**	**Purpose** (Use Case)	**Integration Usage** (How It's Used)	**Folde**
Master Prompt Template	Prompt	*Cross-Engine* <br/ >(Playbook)	Master instruction template for all Agent Mode chats, ensuring structured outputs and delta analyses across sessions [143]. Standardizes the format (Decisions, Outputs, Blockers, Next Steps) for chat summaries and comparisons.	Used in every agent-run conversation – it provides a consistent scaffold for the AI to report updates. Also enables *delta analysis* (Old vs New content changes) automatically by comparing updates [144]. Stored as a prompt file and referenced whenever an engine's chat summary is needed, so outputs remain uniform.	Prime Autor Age Prom
PTS_BD_Master_Trackers.xlsx	Spreadsheet	*Cross-Engine* <br/ >(Playbook)	Dual-sheet BD tracker for monitoring (1) **Target Programs Intelligence** and (2) **Prime TS Active Pursuits**. Consolidates program status, contacts, and active capture efforts in one Excel workbook.	Used as a living tracking tool (outside automation) – updated by BD team and potentially via automation sync. In Maintenance phase, agents or team review this to ensure no target is neglected [145]. Agents could read it to adjust priorities or prompt follow-ups on stalled pursuits [146].	Prime Autor Trac Sheet

Title	Type	Engine	**Purpose** (Use Case)	**Integration Usage** (How It's Used)	Folde
Q3–Q4 2025 DoD BD Events Calendar	Calendar	*Cross-Engine* <br/>(Playbook)	Chronological calendar of relevant DoD/IC industry events in Q3 & Q4 2025 (AUSA, TechNet, DISA Forecast, etc.), annotated with BD notes [147]. Helps align BD efforts with networking opportunities.	Referenced in Maintenance stage – ensures both the BD team and automation know upcoming conferences and industry days [148]. Agents might use it to suggest outreach timing (e.g., scheduling meetings around an event) or to plan intel-gathering trips.	Prime Autor Trac Sheet

Scoring Engine (Engine 5 – Opportunity Prioritization)

Engine 5 computes a priority score for each BD opportunity or program by aggregating data from all other engines (phase: **Scoring** – ranking where to focus). Artifacts include the scoring model design, integration notes, and any preliminary ranking reports:

Title	**Type**	**Engine**	**Purpose** (Use Case)	**Integration Usage** (How It's Used)	**Folder Path**	**Hyperlink**
Scoring Engine – Weighted Model Draft	Document	Scoring	Draft design of the weighted scoring model for BD opportunities – proposes weights for factors like demand (30%), pain (20%), contract value (20%), past performance fit (20%), competition (10%) [149] . Defines how scores should be calculated.	Guides future implementation of the Scoring Engine. Since no live scoring is running yet, this document serves as the blueprint for coding the scoring logic (likely in Python or SQL) once data is ready [150] . Essentially the agreed formula that will be translated into code.	Engine 5 – Scoring/ Design & Specs	ScoringModel_Draft.docx <sup>*</sup>
API Integration Notes for Scoring (USAspending & FPDS)	Document	Scoring	Notes on integrating external data sources (USAspending API, FPDS scrapes) to enhance scoring [151] . Covers how to incorporate contract award data (values, prime vs sub) into the scoring criteria.	Used during Scoring Engine planning – outlines how to fetch and use authoritative contract data so the scoring can account for program size/ value [152] . Informed the integration of our FPDS scraper and potential USAspending API calls into the scoring process.	Engine 5 – Scoring/ Design & Specs	Scoring_API_Notes.pdf <sup>*</sup>

Title	**Type**	**Engine**	**Purpose** (Use Case)	**Integration Usage** (How It's Used)	**Folder Path**	**Hyperlink**
Top Opportunities (Most to Least Strategic).pdf	Report	Scoring	Ranked list (report) of BD opportunities/ programs labeled from most strategic to least, presumably created by leadership as a subjective prioritization [153]. Serves as a human-derived ranking for comparison.	Serves as an initial input or sanity check for the automated Scoring Engine. We use this human-ranked list to calibrate our model – ensuring the engine's output aligns with leadership's priorities or making intentional deviations [154]. Helps validate that our scoring weights yield similar top targets.	Engine 5 – Scoring/ Reference Reports	Top_Opportunities_Ranking.p <sup>*</sup>

Cross-Engine & Infrastructure

Artifacts that span multiple engines or pertain to overall system infrastructure, including multi-engine agent tools, repository analyses, external workflow templates, etc.:

Title	Type	Engine	**Purpose** (Use Case)	**Integration Usage** (How It's Used)	Folder Path	Hyp
AI Automation Society N8n Templates – Categorization & Usage Guide.pdf	Document	Cross-Engine	Comprehensive mapping of community-contributed **n8n templates** from the AI Automation Society forum (Skool community), categorized by type and date [155]. Lists various AI workflow templates and explains their purpose with usage tips.	Reference library of automation patterns. Not executed directly in our pipeline, but provides inspiration and examples (multi-agent coordination, error handling, etc.) [156]. The dev team can import or mimic these templates to enhance our n8n workflows.	Prime TS BD Automation/ External References	AI_S <sup
NewsletterAgentTeam.json	Workflow (n8n)	Cross-Engine	Template: AI-generated newsletter workflow – delegates sub-tasks (news gathering, summarization, email draft) to a team of AI "sub-agents" [157]. Automates daily or periodic newsletter creation with multiple agents.	Can be imported to automate multi-agent content creation. Coordinates GPT-powered agents for each part of the workflow (one curates news, one writes summary, etc.) [158]. Useful as an example of parallel agent orchestration in n8n.	Prime TS BD Automation/ External Templates	New <sup

Title	Type	Engine	**Purpose** (Use Case)	**Integration Usage** (How It's Used)	Folder Path	Hyp
UltimateMediaAgent.json	Workflow (n8n)	Cross-Engine	Template: "Ultimate Media Agent" – automates a range of media content tasks (generate social posts, videos, graphics via AI, then post/schedule) [159]. A complex multi-function media workflow.	A starting point for comprehensive media automation. After import, one can configure API keys (image/video generation services, social APIs) to auto-generate and publish content across channels [160]. Demonstrates orchestrating several creative tasks in one flow.	Prime TS BD Automation/ External Templates	Ultin <sup
9Socials.json	Workflow (n8n)	Cross-Engine	Template: automates posting content to **nine social platforms** at once [161]. Simplifies multi-platform distribution by handling all API calls in one workflow.	Can be imported and configured with social media account credentials. Useful for broadcasting announcements or marketing content broadly with a single trigger [162]. Shows how to use multiple API nodes in parallel for different platforms.	Prime TS BD Automation/ External Templates	9Soc

Title	**Type**	**Engine**	**Purpose** (Use Case)	**Integration Usage** (How It's Used)	**Folder Path**	**Hyp**
ErrorHandling.json	Workflow (n8n)	Cross-Engine	Template: "Unlimited" error handling workflow for n8n. Sets up a global error catcher that other workflows can trigger to log errors, retry tasks, or send alerts [163]. Prevents crashes by handling errors centrally.	Meant to be embedded as a sub-workflow or called on error. Other workflows route their errors here (via error triggers) so this flow can log details, notify Slack/Email, or even attempt retries [164]. Provides robust 24/7 automation by catching failures.	Prime TS BD Automation/ External Templates	Unlir <sup
ElevenLabsVoice.json	Workflow (n8n)	Cross-Engine	Template: AI voice agent integration using ElevenLabs text-to-speech [165]. Allows an AI agent to generate spoken audio output from text (giving the agent a "voice").	Import to add voice capability to agents. Requires ElevenLabs API credentials; then an agent's text output can be converted to an audio file via this workflow [166]. Useful for voice assistants or generating audio briefs from text content.	Prime TS BD Automation/ External Templates	Eleve <sup

Title	**Type**	**Engine**	**Purpose** (Use Case)	**Integration Usage** (How It's Used)	**Folder Path**	**Hyp**
AgentSwarm.json	Workflow (n8n)	Cross-Engine	Template: AI "manager-agent" spawns and coordinates a **team of sub-agents** in parallel [167]. Demonstrates dynamic creation of multiple agents for parallel task execution under one supervising agent.	Import to implement multi-agent orchestration. The manager agent distributes tasks to sub-agents (which can be separate workflows), then aggregates results [168]. Useful for breaking complex jobs (research, writing, QC) into parallel subtasks handled by different AI agents simultaneously.	Prime TS BD Automation/ External Templates	Ager sup>
Metadata.json	Workflow (n8n)	Cross-Engine	Template: **Metadata enrichment** workflow. Automates extraction of metadata (keywords, tags, descriptions) from content and/or uses metadata to route info in workflows [169].	Adaptable utility – for example, connect to a file upload trigger to auto-tag files with AI-generated keywords, or use in a content pipeline to add descriptions. Improves searchability and organization by enriching raw data with metadata [170].	Prime TS BD Automation/ External Templates	Meta sup>

Title	**Type**	**Engine**	**Purpose** (Use Case)	**Integration Usage** (How It's Used)	**Folder Path**	**Hyp**
FirstRAGAgent.json	Workflow (n8n)	Cross-Engine	Template: basic **Retrieval-Augmented Generation (RAG) agent** workflow [171]. Agent can fetch info from a knowledge source (docs, DB) and use it to inform its answers. Demonstrates a simple search-then-answer pattern.	Use to build an AI agent that consults external data. E.g., connect to a document store or API: the workflow performs a search step to retrieve relevant text before the GPT node answers [172]. Ideal for Q&A bots or assistants that need up-to-date knowledge beyond the base model.	Prime TS BD Automation/ External Templates	First <sup
WebhookSecurity.json	Workflow (n8n)	Cross-Engine	Template focusing on securing incoming webhooks in n8n [173]. Implements verification (secret tokens or signatures) so only authorized sources trigger workflows.	Apply to any n8n HTTP Webhook trigger that needs authentication. The workflow checks a token or signature before processing payloads [174]. Protects against unauthorized or malicious triggers. Useful for any externally exposed automation endpoints.	Prime TS BD Automation/ External Templates	Web <sup

Title	Type	Engine	**Purpose** (Use Case)	**Integration Usage** (How It's Used)	Folder Path	Hyp
ZepMemory.json	Workflow (n8n)	Cross-Engine	Template integrating the **Zep long-term memory API** for AI agents [175]. Allows an AI agent to store and retrieve conversation memory or other long-term data via Zep, for persistent memory across sessions.	Import to give agents persistent memory. Configure with Zep API; agent can then save key info (conversation context, user data) and fetch it later [176]. Useful for agents that need to "remember" info over time (beyond a single chat session) without hitting external DBs directly.	Prime TS BD Automation/ External Templates	ZepM sup>
Parallelization.json	Workflow (n8n)	Cross-Engine	Template demonstrating how to run tasks in **parallel** within n8n [177]. Splits a process into multiple branches that execute concurrently to speed up automation.	Use to optimize workflows that can benefit from concurrency (e.g., processing multiple items or API calls at once). Shows usage of multiple branches or the Split-In-Batches node [178]. The accompanying PDF "n8n Subworkflows & Parallelization" provides further guidance [179].	Prime TS BD Automation/ External Templates	Para sup>

Title	Type	Engine	**Purpose** (Use Case)	**Integration Usage** (How It's Used)	Folder Path	Hyp
ResumeScreeningSystem.json	Workflow (n8n)	Cross-Engine	Template for an AI-driven **resume screening system** [180] . Automates intake of resumes, parsing of candidate info (skills, experience), and possibly scoring/ matching to job criteria.	Use to streamline recruiting workflows. Connect to a resume source (email, form upload), use AI parser nodes or APIs to extract structured data, then auto-rank or route candidates [181] . Likely requires configuring a document parser (e.g., OCR or an AI service). Great for automating initial candidate filtering.	Prime TS BD Automation/ External Templates	Resu <sup
RerankingRAG.json	Workflow (n8n)	Cross-Engine	Template for an advanced RAG agent that generates multiple answers and **re-ranks** them to choose the best one [182] . Adds a quality control layer on top of basic RAG.	Use when answer quality is critical. The template likely runs several prompt variations via GPT, then uses a secondary step (another AI or heuristic) to score and pick the best response [183] . Useful for research assistants or decision-support agents where you want high confidence by comparing multiple outputs.	Prime TS BD Automation/ External Templates	Rera <sup

Title	Type	Engine	**Purpose** (Use Case)	**Integration Usage** (How It's Used)	Folder Path	Hyp
YouTubeStrategist.json	Workflow (n8n)	Cross-Engine	Template for a **YouTube content strategy agent** [184]. Automates tasks like researching trending topics, generating video ideas or scripts via AI, and possibly scheduling content.	Import to assist with YouTube channel management. The agent can integrate with YouTube's API for analytics, use OpenAI to brainstorm ideas or write scripts, and help plan an upload schedule [185]. Useful for automating parts of content planning for video creators or marketing teams.	Prime TS BD Automation/ External Templates	YouT <sup
GlassFruitASMR.json	Workflow (n8n)	Cross-Engine	Template from a community challenge showcasing creative automation – generates **ASMR-style content** (audio/video) via AI [186]. A quirky example of unconventional use of AI in workflows.	Not likely used in our business context, but included as a demonstration of creative workflow assembly. If imported, one would configure any needed AI generation nodes (text-to-speech, sound generation, etc.) [187]. Inspires thinking outside the box for how to integrate unusual AI tools into n8n.	Prime TS BD Automation/ External Templates	Glas <sup

Title	Type	Engine	**Purpose** (Use Case)	**Integration Usage** (How It's Used)	Folder Path	Hyp
DeveloperAgent.json	Workflow (n8n)	Cross-Engine	Template for an **AI software development assistant** agent [188]. Can generate code snippets, review code, or turn natural language requests into pseudo-code or actual code using an LLM.	Use to augment developer workflows. For example, an agent can monitor a GitHub repo for new issues and propose code solutions, or respond to "How do I implement X in Python?" prompts [189]. Integrate with OpenAI Codex or similar for best results. Helps automate parts of development (with human oversight).	Prime TS BD Automation/ External Templates	Deve <sup
VideoAnalysis.json	Workflow (n8n)	Cross-Engine	Template automating **video content analysis** [190]. Downloads or accepts a video, then uses AI (vision API or speech-to-text + language model) to extract insights – e.g. generate a summary or detect key topics in the video.	Useful for media monitoring or content creators. Configure with video processing services (e.g., Google Video Intelligence or Whisper for transcription). The workflow likely outputs a text summary or keywords from any given video [191]. Helps to quickly index or get the gist of long videos.	Prime TS BD Automation/ External Templates	Vide sup>

Title	Type	Engine	**Purpose** (Use Case)	**Integration Usage** (How It's Used)	Folder Path	Hyp
BrowserAgent.json	Workflow (n8n)	Cross-Engine	Template for a **web-browsing AI agent** [192]. Enables an AI to navigate websites, take screenshots, and scrape content based on natural language instructions.	Use to give agents limited web browsing capability. Likely requires a headless browser integration (Puppeteer or similar). Once configured, the agent can be instructed to visit pages and retrieve info/screenshots [193]. Great for automating web research tasks via an AI, with caution to throttle requests [194].	Prime TS BD Automation/ External Templates	Brow <sup
FirecrawlSearchScrape.json	Workflow (n8n)	Cross-Engine	Template for "Firecrawl" agent – performs web search queries and scrapes data (including taking webpage screenshots) **without external proxies** [195]. Shows an internal search+scrape agent capability.	Import to build an internal web-scraping agent. Likely uses an API like Google CSE for search and an HTML extract or screenshot node for capture [196]. Useful when the AI needs to pull live data from the web as part of its tasks, complementing BrowserAgent.	Prime TS BD Automation/ External Templates	Firec <sup

Title	Type	Engine	**Purpose** (Use Case)	**Integration Usage** (How It's Used)	Folder Path	Hyp
FirecrawlExtract.json	Workflow (n8n)	Cross-Engine	Companion to Firecrawl – focuses on **extracting structured data** from known pages/ URLs [197] . After the search step, this iterates through given pages and pulls specific info (via HTML selectors or regex).	Use when you have a list of URLs to scrape in detail. Configure parsing logic for target data fields; decouples the search from extraction [198] . For example, feed it competitor press release URLs to extract key points from each. Complements FirecrawlSearchScrape by handling the deep extraction.	Prime TS BD Automation/ External Templates	Firec <sup
25n8nHacks.json	Workflow (n8n)	Cross-Engine	Template containing **25 productivity hacks or hidden features** demonstrations for n8n [199] . Essentially showcases 25 different tips/ tricks in one workflow (likely via example nodes or subflows).	Not an automation use-case per se, but a learning tool. Importing this provides a playground of 25 mini-examples of n8n capabilities [200] . Useful for training or discovering advanced techniques; one can copy patterns from here into real workflows.	Prime TS BD Automation/ External Templates	25_n sup>

Title	Type	Engine	**Purpose** (Use Case)	**Integration Usage** (How It's Used)	Folder Path	Hyp
ShortsMachine.json	Workflow (n8n)	Cross-Engine	Template for a 24/7 "**viral shorts**" generator – automates creating and posting short-form videos (YouTube Shorts/TikToks) continuously without human input [201] .	Examining this shows how to chain content creation in a loop. Likely generates video ideas, uses a video creation API (e.g., HeyGen avatars or FFmpeg) and schedules posts in a continuous cycle [202] . Good for understanding how to keep an agent running indefinitely on a schedule.	Prime TS BD Automation/ External Templates	Shor <sup
APIsforAgents.json	Workflow (n8n)	Cross-Engine	Template demonstrating how AI agents can utilize **external APIs** within workflows [203] . Contains examples of an agent fetching data from various services (weather, stock prices, etc.) and incorporating it into responses.	Use as a foundation to extend agents with real-world tools. Post-import, replace/add API calls relevant to BD (e.g., a CRM API to fetch client info, a calendar API to schedule meetings) [204] . This shows how to make agents more "actionable" by connecting to external services.	Prime TS BD Automation/ External Templates	APIs <sup

Title	**Type**	**Engine**	**Purpose** (Use Case)	**Integration Usage** (How It's Used)	**Folder Path**	**Hyp**
TrackAgentActions.json	Workflow (n8n)	Cross-Engine	Template for **logging and monitoring** AI agent usage and actions [205]. Tracks metrics like token counts, API calls, timestamps, etc., for each agent run.	Integrate this as a monitoring sub-process in AI workflows. Have each agent workflow trigger this logging flow (passing agent name, tokens used, etc.) [206]. It can write to Google Sheets, a database, or logs. Helps analyze usage patterns (e.g., detect a spike in token use) and control costs.	Prime TS BD Automation/ External Templates	Ager <sup
DynamicBrain.json	Workflow (n8n)	Cross-Engine	Template for an AI agent with a self-managed knowledge base ("**dynamic brain**") [207]. The agent can dynamically store info it learns (e.g., in a vector store) and retrieve it later, enabling learning over time.	Implement to give agents persistent adaptive knowledge. Likely uses a vector DB (Pinecone, Weaviate, etc.) for semantic search on stored Q&A. The agent decides when to save new info and when to use memory vs fresh API calls [208]. Great for agents that need to build up knowledge on a user or topic over repeated interactions.	Prime TS BD Automation/ External Templates	Dyna sup>

Title	Type	Engine	**Purpose** (Use Case)	**Integration Usage** (How It's Used)	Folder Path	Hyp
ProductVideography.json	Workflow (n8n)	Cross-Engine	Template automating **product videography** content creation [209]. Takes product images or info and uses AI to compile short promo videos (adds transitions, text overlays, voiceover).	Adapt for marketing automation. Provide product details (from a CMS or DB), then integrate an AI video generation service. The workflow likely generates a script via GPT and uses an API to create a slideshow or animated video [210]. Useful for quickly producing product demo videos at scale.	Prime TS BD Automation/ External Templates	Prod <sup
AIMarketingTeam.json	Workflow (n8n)	Cross-Engine	Template for an "**AI Marketing Team**" agent [211]. Simulates a full marketing department run by AI by breaking tasks (content creation, scheduling, graphic design) into sub-tasks within one workflow.	Automates various marketing tasks in tandem. Likely sequences content writing, image creation (via AI API), and scheduling posts via platform APIs [212]. Customize with your topics/branding. Useful for small businesses to automate weekly social posts, blog content, newsletters, etc.	Prime TS BD Automation/ External Templates	AI_M <sup

Title	Type	Engine	**Purpose** (Use Case)	**Integration Usage** (How It's Used)	Folder Path	Hyp
LinkedInPostGraphic.json	Workflow (n8n)	Cross-Engine	Template for generating LinkedIn posts with an **AI-generated graphic** [213]. Agent takes a topic, writes a LinkedIn post, and creates a simple image (via DALL-E or similar) to accompany it.	Ideal for content creators on LinkedIn. Provide a prompt or topic, and this workflow uses a text generation node for the post content and an image generation API for the graphic [214]. You can then post the combination manually or automatically. Speeds up making engaging posts with visuals.	Prime TS BD Automation/ External Templates	Link <sup
ErrorWorkflow.json	Workflow (n8n)	Cross-Engine	Earlier iteration of an error handling workflow in n8n [215]. Captures errors from other workflows and sends basic alerts (simpler predecessor to the "Unlimited ErrorHandling" template).	Can be used as a quick drop-in if the full error handling system isn't needed. Attach it to critical workflows via error triggers – it will send an email/Slack alert on failure [216]. Useful for basic failure notifications without complex retries.	Prime TS BD Automation/ External Templates	Basi <sup

Title	Type	Engine	**Purpose** (Use Case)	**Integration Usage** (How It's Used)	Folder Path	Hype
ThinkTool.json	Workflow (n8n)	Cross-Engine	Template incorporating Anthropic's "**Think**" chain-of-thought approach for AI reasoning [217]. Prompts an AI (Claude or similar) to reason step-by-step within an n8n workflow for more logical outputs.	Use this to improve your AI agents' reasoning. The workflow likely first asks the AI to outline its approach ("think") then executes the plan in a second step [218]. Good for complex tasks – it encourages the AI to break down problems. Helps debug agent decisions or handle multi-part queries by making the thought process explicit.	Prime TS BD Automation/ External Templates	Anth <sup
3AIWorkflows.json	Workflow (n8n)	Cross-Engine	Template bundle containing three beginner-friendly AI workflows (from a 44-min tutorial) [219]. Likely includes a basic Q&A chatbot, an image generator trigger, and a simple data processing flow as separate examples.	Quick-start pack for new users. Importing this creates multiple example workflows demonstrating simple AI integrations: one might call OpenAI for Q&A, another generates an image from text, etc. [220]. Great for learning/ training purposes to see how triggers and AI nodes connect.	Prime TS BD Automation/ External Templates	3AI_V <sup

Title	Type	Engine	**Purpose** (Use Case)	**Integration Usage** (How It's Used)	Folder Path	Hyp
LongFormContent.json	Workflow (n8n)	Cross-Engine	Template for **fully automated long-form content creation** (e.g. blog posts) [221]. Chains together research, outlining, drafting, and editing steps using AI to produce a full article with minimal human input.	Deploy to automate content marketing. Configure to use preferred info sources (RSS feeds, web search) and an OpenAI model for writing. The workflow likely gathers facts, creates an outline, then writes and refines the article [222]. Useful for generating first drafts of articles or reports on specified topics.	Prime TS BD Automation/ External Templates	Long <sup
DeepResearchReport.json	Workflow (n8n)	Cross-Engine	Template for in-depth research that outputs a formatted PDF report on any given topic [48]. Automates searching, compiling info, and formatting it into a PDF.	Use to have an agent do comprehensive research. Provide a query, and the workflow will perform multiple searches (web queries, knowledge base lookups), organize findings (with headings, bullets), and use a PDF node to generate a nicely formatted report [223]. Ideal for quickly briefing a new topic or automating market research reports.	Prime TS BD Automation/ External Templates	Deep <sup

<small><sup>*</sup>Drive link placeholders – actual links accessible in Prime TS Google Drive.</small>

Master_Inventory.json

The inventory above is also provided in structured JSON format, enabling integration with ChatGPT or other agents for quick reference and retrieval. Each artifact is represented as an object with standardized fields as follows:

- **canonical_title** – A unique, human-readable title for the artifact (often the file name or descriptive name).
- **bd_engine** – Which BD engine(s) the artifact is associated with (*Scraper*, *Program Mapping*, *Org Chart*, *Playbook*, *Scoring*, or *Cross-Engine*).
- **artifact_type** – Type/category of artifact (e.g. Document, Spreadsheet, JSON Config, Workflow, Prompt, Script, Table, etc.).
- **file_name** – Actual file name (including extension if applicable) as stored.
- **drive_path** – Suggested Google Drive folder path where the file resides.
- **integration_status** – Status of integration into the system (e.g. *Active* (in use), *Draft*, *Superseded*, *Reference Only*, *Archived*).
- **purpose** – Brief description of the artifact's purpose or contents (use case).
- **integration_usage** – How the artifact is or isn't used in the workflow (integration guidance).
- **workflow_phase** – Phase of the BD workflow lifecycle that this artifact supports (Target ID, Credibility, Insertion, Scoring, or Maintenance).
- **prompt_tags** – Key tags or labels useful for prompting or searching (topics, context keywords).
- **retrieval_keywords** – Additional keywords to help retrieve this artifact in a QA or agent context (synonyms or related terms someone might use to ask for it).
- **cross_references** – List of titles of related artifacts (e.g. an updated version, a data source, or a connected document).

Below is the JSON Master Inventory, grouped by engine:

```
{
  "Scraper": [
    {
      "canonical_title": "20250804_142008.pdf",
      "bd_engine": "Scraper",
      "artifact_type": "Document",
      "file_name": "20250804_142008.pdf",
      "drive_path": "Prime TS BD Automation/Engine 1 – Scraper/References",
      "integration_status": "Archived (Not Used)",
      "purpose": "Unidentified reference document (possibly auto-saved email or
web page). Content unclear.",
      "integration_usage": "Not actively used – kept only as an archived
reference (no known integration).",
      "workflow_phase": "Target ID",
      "prompt_tags": ["Archive", "Misc"],
      "retrieval_keywords": ["unknown reference file", "auto-saved document",
"unidentified PDF"],
      "cross_references": []
    },
    {
      "canonical_title": "AI Automation Methodologies & Blueprint.pdf",
      "bd_engine": "Scraper",
      "artifact_type": "Document",
      "file_name": "AI Automation Methodologies & Blueprint (Nate Herk & AI
Automation Society).pdf",
      "drive_path": "Prime TS BD Automation/Engine 1 – Scraper/References",
      "integration_status": "Reference",
      "purpose": "Overview of no-code AI automation methods and blueprint
(community guide).",
      "integration_usage": "Reference methodology document – informed design of
our n8n workflows, RAG pipelines, and multi-agent integration.",
      "workflow_phase": "Maintenance",
      "prompt_tags": ["Methodology", "Blueprint"],
      "retrieval_keywords": ["no-code automation guide", "AI Automation Society
blueprint", "best practices reference"],
      "cross_references": []
    },
    {
      "canonical_title": "Installing Puppeteer on Windows (Docker Desktop
Setup).pdf",
      "bd_engine": "Scraper",
      "artifact_type": "Document",
      "file_name": "Installing Puppeteer on Windows (Docker Desktop Setup).pdf",
      "drive_path": "Prime TS BD Automation/Engine 1 – Scraper/Setup Guides",
      "integration_status": "Reference",
      "purpose": "Tech setup guide for Puppeteer on Windows via Docker.",
      "integration_usage": "Used during development to configure the Puppeteer
environment for web scraping tasks.",
      "workflow_phase": "Maintenance",
      "prompt_tags": ["Setup", "Puppeteer"],
      "retrieval_keywords": ["Puppeteer install guide", "Docker Puppeteer
setup", "scraper environment config"],
      "cross_references": []
    },
    {
      "canonical_title": "Competitor Job Board URL Master Table",
      "bd_engine": "Scraper",
      "artifact_type": "Table (Document)",
      "file_name": "Competitor Job Board URL Master Table.xlsx",
      "drive_path": "Prime TS BD Automation/Engine 1 – Scraper/Configurations",
      "integration_status": "Active",
      "purpose": "Master list of competitor careers page URLs and ATS types
(Tier 1–3 cleared firms).",
      "integration_usage": "Ingested as config for Scraper Engine; n8n iterates
through this list to direct scrapes to each competitor site.",
      "workflow_phase": "Target ID",
      "prompt_tags": ["Targets", "ATS List"],
      "retrieval_keywords": ["competitor list", "ATS URLs", "job board list",
"scraping targets"],
      "cross_references": ["Scraper Engine Config JSON"]
    },
    {
      "canonical_title": "Optimized Coverage Hubs (18 cities)",
      "bd_engine": "Scraper",
      "artifact_type": "List (CSV)",
      "file_name": "Optimized Coverage Hubs.csv",
      "drive_path": "Prime TS BD Automation/Engine 1 – Scraper/Configurations",
      "integration_status": "Active",
      "purpose":
"List of ~18 metro hub locations to prioritize for job scraping (geographic
coverage targets).",
      "integration_usage": "Used in scraper workflows to loop through target
locations for job searches (ensures priority cities are covered).",
      "workflow_phase": "Target ID",
      "prompt_tags": ["Geo Coverage", "Locations"],
      "retrieval_keywords": ["metro hub list", "target cities for jobs",
"location list for scraping"],
      "cross_references": []
    },
    {
      "canonical_title": "Apify Actor Input (Fort Meade)",
      "bd_engine": "Scraper",
      "artifact_type": "JSON Config",
      "file_name": "Apify Actor Input - Fort Meade.json",
      "drive_path": "Prime TS BD Automation/Engine 1 – Scraper/Workflows",
      "integration_status": "Active",
      "purpose": "JSON config for an Apify actor to scrape Insight Global jobs
in Fort Meade (with clearance keywords).",
      "integration_usage": "Run via Apify in n8n (HTTP integration) – drives a
daily actor execution for that location with clearance filters.",
      "workflow_phase": "Target ID",
      "prompt_tags": ["Scraper Config", "Apify"],
      "retrieval_keywords": ["Apify actor config", "Fort Meade scraper config",
"clearance keywords config"],
      "cross_references": []
    },
    {
      "canonical_title": "Cleared Job Scraper (Example)",
      "bd_engine": "Scraper",
      "artifact_type": "Workflow (JSON)",
      "file_name": "Cleared Job Scraper v1.json",
      "drive_path": "Prime TS BD Automation/Engine 1 – Scraper/Workflows",
      "integration_status": "Superseded (Draft)",
      "purpose": "First draft n8n workflow for scraping a single source (e.g.
TEKsystems) and sending results to Airtable.",
      "integration_usage": "Prototype JSON imported into n8n – scrapes one
source and posts to Airtable. Superseded by revised multi-source workflow.",
      "workflow_phase": "Target ID",
      "prompt_tags": ["Workflow (Draft)"],
      "retrieval_keywords": ["initial scraper workflow", "single-source scraper
JSON", "Airtable upload scraper"],
      "cross_references": ["Cleared Jobs Workflow (Revised)"]
    },
    {
      "canonical_title": "Cleared Jobs Workflow (Revised)",
      "bd_engine": "Scraper",
      "artifact_type": "Workflow (JSON)",
      "file_name": "Cleared Jobs Workflow v2.json",
      "drive_path": "Prime TS BD Automation/Engine 1 – Scraper/Workflows",
      "integration_status": "Active",
      "purpose": "Revised end-to-end n8n workflow to scrape multiple sources,
enrich data, de-duplicate, and sync cleared job postings to Airtable.",
      "integration_usage": "Main scraping workflow in production (triggered via
n8n cron) – iterates through competitor list, processes postings (with
deduplication & multi-location support), then stores to Airtable.",
      "workflow_phase": "Target ID",
      "prompt_tags": ["Workflow (Prod)"],
      "retrieval_keywords": ["multi-source scraper workflow", "deduping scraper
JSON", "Airtable sync workflow"],
      "cross_references": ["Cleared Job Scraper (Example)"]
    },
    {
      "canonical_title": "Scraper Config JSON",
      "bd_engine": "Scraper",
      "artifact_type": "JSON Config",
      "file_name": "Scraper_Test_Config.json",
      "drive_path": "Prime TS BD Automation/Engine 1 – Scraper/Configurations",
      "integration_status": "Active",
      "purpose": "Puppeteer/Apify config JSON for a multi-keyword job scrape in
one location (Fort Meade test run).",
      "integration_usage": "Used by Scraper Engine’s Puppeteer script – defines
search keywords, location, and crawl settings for test runs (template for
broader scrapes).",
      "workflow_phase": "Target ID",
      "prompt_tags": ["Config", "Puppeteer"],
      "retrieval_keywords": ["scraper config JSON", "test scrape config",
"keyword search config"],
      "cross_references": []
    },
    {
      "canonical_title": "Scraper Engine Config JSON",
      "bd_engine": "Scraper",
      "artifact_type": "JSON Config",
      "file_name": "ScraperEngine_Config.json",
      "drive_path": "Prime TS BD Automation/Engine 1 – Scraper/Configurations",
      "integration_status": "Active",
      "purpose":
"Master configuration mapping competitor organizations to their job board URLs
and ATS types.",
      "integration_usage": "Ingested by Scraper workflow to dynamically route
scrapers based on each competitor’s ATS (ensures correct scraping method per
site).",
      "workflow_phase": "Target ID",
      "prompt_tags": ["Config", "Targets"],
      "retrieval_keywords": ["scraper master config", "competitor ATS mapping
JSON", "competitor config file"],
      "cross_references": ["Competitor Job Board URL Master Table"]
    },
    {
      "canonical_title": "Job Postings – Grid View.csv",
      "bd_engine": "Scraper",
      "artifact_type": "CSV Data",
      "file_name": "Job_Postings_GridView.csv",
      "drive_path": "Prime TS BD Automation/Engine 1 – Scraper/Data Exports",
      "integration_status": "Reference",
      "purpose": "CSV export of job postings from Airtable (Grid View), used to
align schema between Airtable and internal DB.",
      "integration_usage": "Used in data integration – served as reference to
ensure fields from scraping match Airtable and planned Postgres schema.",
      "workflow_phase": "Target ID",
      "prompt_tags": ["Data Export", "Schema"],
      "retrieval_keywords": ["jobs CSV export", "Airtable jobs export", "job
schema reference"],
      "cross_references": []
    },
    {
      "canonical_title": "Jobs Records (Full)",
      "bd_engine": "Scraper",
      "artifact_type": "Table (Excel)",
      "file_name": "Jobs_Records_Full.xlsx",
      "drive_path": "Prime TS BD Automation/Engine 1 – Scraper/Data Exports",
      "integration_status": "Active",
      "purpose":
"Complete set of job records parsed from an exported text block (all jobs
portion of placements doc).",
      "integration_usage": "Used for data loading/testing – this full table of
job entries (historical dump) served as input to engines (especially Mapping &
Scoring) to ensure baseline data coverage.",
      "workflow_phase": "Target ID",
      "prompt_tags": ["Data", "Historical"],
      "retrieval_keywords": ["all jobs dataset", "jobs master list",
"historical job records"],
      "cross_references": ["Jobs Table (Separate Doc)"]
    },
    {
      "canonical_title": "Jobs Table (Separate Doc)",
      "bd_engine": "Scraper",
      "artifact_type": "Table (Excel)",
      "file_name": "Jobs_Table_Additional.xlsx",
      "drive_path": "Prime TS BD Automation/Engine 1 – Scraper/Data Exports",
      "integration_status": "Duplicate (Merged)",
      "purpose": "Job records parsed from a separately provided jobs document
(additional source).",
      "integration_usage": "Supplementary input merged into Jobs Records (Full)
to ensure all job postings were captured. Now superseded by the merged full
dataset.",
      "workflow_phase": "Target ID",
      "prompt_tags": ["Data", "Supplement"],
      "retrieval_keywords": ["additional jobs data", "second jobs source",
"supplemental jobs records"],
      "cross_references": ["Jobs Records (Full)"]
    },
    {
      "canonical_title": "Tara Lopez.pdf",
      "bd_engine": "Scraper",
      "artifact_type": "Document",
      "file_name": "Tara Lopez.pdf",
      "drive_path": "Prime TS BD Automation/Engine 1 – Scraper/Test Data",
      "integration_status": "Test Data",
      "purpose": "Resume or LinkedIn profile of an individual (Tara Lopez) used
as a sample input.",
      "integration_usage": "Served as a test document for AI resume/profile
parsing – used to validate that our parsing workflow correctly extracts info
from a real resume. Not part of production data flow.",
      "workflow_phase": "Credibility",
      "prompt_tags": ["Sample Data", "Resume"],
      "retrieval_keywords": ["sample resume PDF", "test profile document",
"Tara Lopez resume"],
      "cross_references": []
    }
  ],
  "Program Mapping": [
    {
      "canonical_title": "Bullhorn Contact Extraction Plan",
      "bd_engine": "Program Mapping",
      "artifact_type": "Document",
      "file_name": "Bullhorn Contact Extraction Plan.docx",
      "drive_path": "Prime TS BD Automation/Engine 2 – Program Mapping/Strategy
Docs",
      "integration_status": "Active",
      "purpose": "Schema/plan for extracting key contacts from Bullhorn (roles,
by prime/program/location).",
      "integration_usage": "Used to drive Bullhorn data pulls – informs Org
Chart which contacts (PMs, leads) to import per program, enriching the program
maps with internal contacts.",
      "workflow_phase": "Credibility",
      "prompt_tags": ["Schema", "Contacts"],
      "retrieval_keywords": ["Bullhorn contacts schema", "contact extraction
plan", "BD contacts by program"],
      "cross_references": ["Engine Placements (Full Parsed Block)"]
    },
    {
      "canonical_title": "Top 20 DoD Programs (High Subcontractor Spend).pdf",
      "bd_engine": "Program Mapping",
      "artifact_type": "Document",
      "file_name": "Top 20 DoD Programs with High Subcontractor Staffing
Spend.pdf",
      "drive_path": "Prime TS BD Automation/Engine 2 – Program Mapping/Strategy
Docs",
      "integration_status": "Active",
      "purpose": "Analysis of 20 top DoD programs ranked by subcontractor
staffing spend, with descriptions & opportunities.",
      "integration_usage": "Used to prioritize targeting – Mapping engine tags
new jobs with these program names and uses their ranking to focus BD efforts on
high-value programs.",
      "workflow_phase": "Target ID",
      "prompt_tags": ["Programs", "Priority"],
      "retrieval_keywords": ["Top 20 DoD programs list", "high spend programs
analysis", "target program list"],
      "cross_references": []
    },
    {
      "canonical_title": "Past Performance Made Easy Lists.pdf",
      "bd_engine": "Program Mapping",
      "artifact_type": "Document",
      "file_name": "Past Performance Made Easy Lists.pdf",
      "drive_path": "Prime TS BD Automation/Engine 2 – Program Mapping/
Credibility Docs",
      "integration_status": "Active",
      "purpose": "Quick-reference lists of past performance examples and
contract highlights for credibility.",
      "integration_usage": "Referenced by Mapping & Playbook engines – provides
past performance talking points that can be inserted into proposals or outreach
to strengthen credibility.",
      "workflow_phase": "Credibility",
      "prompt_tags": ["Credibility", "Past Performance"],
      "retrieval_keywords": ["past performance cheat sheet", "contract
highlights list", "success story list"],
      "cross_references": ["Prime NGC Past Performance.pdf"]
    },
    {
      "canonical_title": "Prime NGC Past Performance.pdf",
      "bd_engine": "Program Mapping",
      "artifact_type": "Document",
      "file_name": "Prime NGC Past Performance.pdf",
      "drive_path": "Prime TS BD Automation/Engine 2 – Program Mapping/
Credibility Docs",
      "integration_status": "Active",
      "purpose": "Northrop Grumman past performance \"battle card\" (key
programs, contract examples, Prime TS experience).",
      "integration_usage": "Used by Playbook & Mapping engines – when a target
involves NGC, the system can pull credibility statements from this card to cite
relevant past successes.",
      "workflow_phase": "Credibility",
      "prompt_tags": ["Credibility", "Battle Card"],
      "retrieval_keywords": ["NGC past performance", "Northrop Grumman battle
card", "Prime TS NGC experience"],
      "cross_references": ["Past Performance Made Easy Lists.pdf"]
    },
    {
      "canonical_title": "Proposed Architecture (BD Intelligence
Automation).pdf",
      "bd_engine": "Program Mapping",
      "artifact_type": "Document",
      "file_name": "Proposed Architecture for Prime TS BD Intelligence
Automation.pdf",
      "drive_path": "Prime TS BD Automation/Architecture & Planning",
      "integration_status": "Superseded",
      "purpose": "Original high-level design blueprint of the five-engine BD
automation system (conceptual architecture).",
      "integration_usage": "Initially guided development of engines and n8n
workflow integration. Superseded by the updated final architecture doc.",
      "workflow_phase": "Maintenance",
      "prompt_tags": ["Architecture (Old)"],
      "retrieval_keywords": ["initial system design", "conceptual BD automation
architecture", "early architecture blueprint"],
      "cross_references": ["End-to-End BD Automation Stack for Prime TS.pdf"]
    },
    {
      "canonical_title": "End-to-End BD Automation Stack (Prime TS).pdf",
      "bd_engine": "Program Mapping",
      "artifact_type": "Document",
      "file_name": "End-to-End BD Automation Stack for Prime Technical
Services.pdf",
      "drive_path": "Prime TS BD Automation/Architecture & Planning",
      "integration_status": "Active",
      "purpose": "Finalized architecture & tech stack for the integrated BD
automation system (covering all 5 engines and tools).",
      "integration_usage": "Primary reference for implementation – defines data
flow between engines and technologies (OpenAI, Apify, Airtable, Postgres, etc.)
used at each stage. Guides environment setup and CI/CD for the whole stack.",
      "workflow_phase": "Maintenance",
      "prompt_tags": ["Architecture", "Tech Stack"],
      "retrieval_keywords": ["final architecture doc", "BD automation tech
stack", "system design final"],
      "cross_references": ["Proposed Architecture (BD Intelligence
Automation).pdf"]
    },
    {
      "canonical_title": "Hiring Trends and BD Target Analysis.pdf",
      "bd_engine": "Program Mapping",
      "artifact_type": "Document",
      "file_name": "Hiring Trends and BD Target Analysis.pdf",
      "drive_path": "Prime TS BD Automation/Engine 2 – Program Mapping/
Credibility Docs",
      "integration_status": "Active",
      "purpose": "\"Heatmap\" analysis of hiring trends, placements, and past
performance across agencies/programs – identifies BD opportunities and gaps.",
      "integration_usage": "Informs Mapping & Scoring engines – data points
(e.g. hot programs, clearance demands) feed into scoring models and outreach
prep to highlight market insights.",
      "workflow_phase": "Credibility",
      "prompt_tags": ["Analysis", "Trends"],
      "retrieval_keywords": ["hiring trends report", "BD target analysis doc",
"placement trends and gaps"],
      "cross_references": []
    },
    {
      "canonical_title": "Engine Placements (Clean Headers)",
      "bd_engine": "Program Mapping",
      "artifact_type": "Table (Excel)",
      "file_name": "Engine_Placements_CleanHeaders.xlsx",
      "drive_path": "Prime TS BD Automation/Engine 2 – Program Mapping/Data",
      "integration_status": "Duplicate",
      "purpose": "Placement records table with initial cleaned/standardized
column names.",
      "integration_usage": "Intermediate dataset used during placement data
cleaning – ensured consistent field names for mapping placements to programs.",
      "workflow_phase": "Credibility",
      "prompt_tags": ["Data", "Cleaning"],
      "retrieval_keywords": ["placements clean headers",
"standardized placement fields", "cleaned header table"],
      "cross_references": ["Engine Placements (Remapped Headers)"]
    },
    {
      "canonical_title": "Engine Placements (Remapped Headers)",
      "bd_engine": "Program Mapping",
      "artifact_type": "Table (Excel)",
      "file_name": "Engine_Placements_RemappedHeaders.xlsx",
      "drive_path": "Prime TS BD Automation/Engine 2 – Program Mapping/Data",
      "integration_status": "Duplicate",
      "purpose": "Placement records with headers remapped to the final schema
(after renaming columns).",
      "integration_usage": "Intermediate dataset showing result of header
renaming – verified all placement fields match expected schema before full
integration.",
      "workflow_phase": "Credibility",
      "prompt_tags": ["Data", "Cleaning"],
      "retrieval_keywords": ["placements remapped headers", "renamed columns
dataset", "schema-aligned placement data"],
      "cross_references": ["Engine Placements (Clean Headers)", "Engine
Placements (Full Cleaned)"]
    },
    {
      "canonical_title": "Engine Placements (Clean Job Titles)",
      "bd_engine": "Program Mapping",
      "artifact_type": "Table (Excel)",
      "file_name": "Engine_Placements_CleanJobTitles.xlsx",
      "drive_path": "Prime TS BD Automation/Engine 2 – Program Mapping/Data",
      "integration_status": "Duplicate",
      "purpose": "Placement records after stripping recruiter names from job
titles (cleaned job titles).",
      "integration_usage": "Intermediate cleaning step – ensured standardized
job titles (no extraneous text) for accurate role clustering in Org Chart
engine.",
      "workflow_phase": "Credibility",
      "prompt_tags": ["Data", "Cleaning"],
      "retrieval_keywords": ["placements clean job titles", "cleaned job titles
dataset", "recruiter names removed data"],
      "cross_references": ["Engine Placements (Full Cleaned)"]
    },
    {
      "canonical_title": "Engine Placements (Full Cleaned)",
      "bd_engine": "Program Mapping",
      "artifact_type": "Table (Excel)",
      "file_name": "Engine_Placements_FullCleaned_v2.xlsx",
      "drive_path": "Prime TS BD Automation/Engine 2 – Program Mapping/Data",
      "integration_status": "Active",
      "purpose": "Full placement dataset after all cleaning steps, with final
schema applied (complete cleaned output).",
      "integration_usage": "Consolidated placement data used by Org Chart &
```

```
Scoring engines – primary input for inferring team structures and calculating
credibility metrics (e.g. placements count by program).",
      "workflow_phase": "Credibility",
      "prompt_tags": ["Data", "Cleaned Master"],
      "retrieval_keywords": ["placement data final cleaned", "consolidated
placements dataset", "cleaned placements master file"],
      "cross_references": ["Engine Placements (Full Parsed Block)"]
    },
    {
      "canonical_title": "Engine Placements (Full Parsed Block)",
      "bd_engine": "Program Mapping",
      "artifact_type": "Table (Excel)",
      "file_name": "Engine_Placements_RawParsed.xlsx",
      "drive_path": "Prime TS BD Automation/Engine 2 – Program Mapping/Data",
      "integration_status": "Duplicate",
      "purpose":
"Structured table of placement data parsed from raw pasted text (full dataset
prior to cleaning).",
      "integration_usage":
"Baseline parsed placements used at start of cleaning pipeline. Contains all raw
placement entries which were then cleaned to produce the Full Cleaned dataset.",
      "workflow_phase": "Credibility",
      "prompt_tags": ["Data", "Raw Parse"],
      "retrieval_keywords": ["raw placement data table", "parsed placements
(uncleaned)", "initial placement dataset"],
      "cross_references": ["Engine Placements (Clean Headers)"]
    },
    {
      "canonical_title": "NTT Data Apps & BPS Tech Stack.pdf",
      "bd_engine": "Program Mapping",
      "artifact_type": "Document",
      "file_name": "NTT Data Apps & BPS Tech Stack.pdf",
      "drive_path": "Prime TS BD Automation/Engine 2 – Program Mapping/
Competitive Intel",
      "integration_status": "Active",
      "purpose": "Technical stack document detailing NTT Data’s Applications &
BPS offerings/technologies (competitor intel).",
      "integration_usage": "Used for competitive intelligence in Mapping/Org
Chart – provides insight into NTT’s capabilities, informing program targeting
(know competitor strengths) and differentiation points in strategy.",
      "workflow_phase": "Credibility",
      "prompt_tags": ["Competitive Intel"],
      "retrieval_keywords": ["NTT tech stack document", "NTT Data
capabilities", "competitor tech offerings PDF"],
      "cross_references": []
    }
  ],
  "Org Chart": [
    {
      "canonical_title": "Org Chart Engine – Org Design Outline",
      "bd_engine": "Org Chart",
      "artifact_type": "Document",
      "file_name": "Org Chart Engine – Org Design Outline.docx",
      "drive_path": "Prime TS BD Automation/Engine 3 – Org Chart/Strategy &
Design",
      "integration_status": "Active",
      "purpose": "Methodology outline for reconstructing org hierarchy & key
personnel from hiring data (Org Chart Engine blueprint).",
      "integration_usage": "Design spec guiding Org Chart Engine development –
informs how to cluster roles and link contacts to infer reporting structures in
workflow (basis for role clustering script in n8n).",
      "workflow_phase": "Credibility",
      "prompt_tags": ["Design", "Methodology"],
      "retrieval_keywords": ["Org Chart design doc", "org reconstruction
blueprint", "hierarchy inference outline"],
      "cross_references": []
    },
    {
      "canonical_title": "GBSD BD Strategy.pdf",
      "bd_engine": "Org Chart",
      "artifact_type": "Document",
      "file_name": "GBSD BD Strategy.pdf",
      "drive_path": "Prime TS BD Automation/Engine 3 – Org Chart/Program Intel",
      "integration_status": "Active",
      "purpose": "BD strategy for the Sentinel/GBSD program (org structure, key
roles, subcontractor targets, capture tactics).",
      "integration_usage": "Used by Org Chart & Playbook engines for GBSD
pursuit – provides program-specific context (org insights, target roles) to
tailor outreach and insertion tactics.",
      "workflow_phase": "Insertion",
      "prompt_tags": ["Program Strategy"],
      "retrieval_keywords": ["GBSD strategy deck", "Sentinel program strategy",
"GBSD capture plan"],
      "cross_references": ["GBSD Notebook.pdf", "Sentinel GBSD Program Key
Contacts Call Sheet.pdf"]
    },
    {
      "canonical_title": "GBSD Notebook.pdf",
      "bd_engine": "Org Chart",
      "artifact_type": "Document",
      "file_name": "GBSD Notebook.pdf",
      "drive_path": "Prime TS BD Automation/Engine 3 – Org Chart/Program Intel",
      "integration_status": "Active",
      "purpose": "Reference notes for the GBSD program (historical context,
requirements, contract background, etc.).",
      "integration_usage": "Knowledge base for Org Chart/Playbook – agent can
```

```
pull background info from here to enrich playbooks and outreach for GBSD (adds
historical context to communications).",
      "workflow_phase": "Credibility",
      "prompt_tags": ["Program Intel"],
      "retrieval_keywords": ["GBSD program notes", "Sentinel program
background", "GBSD historical context"],
      "cross_references": ["GBSD BD Strategy.pdf"]
    },
    {
      "canonical_title": "Sentinel GBSD Program Key Contacts Call Sheet.pdf",
      "bd_engine": "Org Chart",
      "artifact_type": "Document",
      "file_name": "Sentinel GBSD Program Key Contacts Call Sheet.pdf",
      "drive_path": "Prime TS BD Automation/Engine 3 – Org Chart/Program Intel",
      "integration_status": "Active",
      "purpose": "Call sheet listing key contacts for the GBSD program (names,
titles, contact info, notes for outreach).",
      "integration_usage": "Feeds Org Chart’s contact list for GBSD – these
contacts are prioritized for outreach. Playbook Engine uses this to craft
personalized call/email scripts for each contact.",
      "workflow_phase": "Insertion",
      "prompt_tags": ["Contacts", "Call Sheet"],
      "retrieval_keywords": ["GBSD key contacts list", "Sentinel call sheet",
"GBSD contact sheet"],
      "cross_references": ["GBSD BD Strategy.pdf"]
    },
    {
      "canonical_title": "Ogden Org Overview.pdf",
      "bd_engine": "Org Chart",
      "artifact_type": "Document",
      "file_name": "Ogden Org Overview.pdf",
      "drive_path": "Prime TS BD Automation/Engine 3 – Org Chart/Program Intel",
      "integration_status": "Active",
      "purpose": "Org structure overview for the GBSD program’s Ogden site
(teams and hierarchy at that location).",
      "integration_usage": "Provides a reference org chart for GBSD-Ogden. Org
Chart Engine uses it to validate its inferred org structure for that site and
ensure accuracy.",
      "workflow_phase": "Credibility",
      "prompt_tags": ["Program Org"],
      "retrieval_keywords": ["GBSD Ogden org chart", "Sentinel Ogden org
overview", "Ogden site org structure"],
      "cross_references": ["GBSD Ogden Contacts Job/Function.pdf"]
    },
    {
      "canonical_title": "GBSD Ogden Contacts Job/Function.pdf",
      "bd_engine": "Org Chart",
      "artifact_type": "Document",
      "file_name": "GBSD Ogden Contacts (Job-Function).pdf",
      "drive_path": "Prime TS BD Automation/Engine 3 – Org Chart/Program Intel",
      "integration_status": "Active",
      "purpose":
"List of contacts on GBSD (Ogden teams) with names, job titles, possibly job
functions.",
      "integration_usage": "Used to enrich Org Chart for GBSD – these contacts
are loaded into the engine’s dataset so outreach can target specific
individuals. Playbook uses these names to personalize messaging (mentioning
roles).",
      "workflow_phase": "Insertion",
      "prompt_tags": ["Contacts", "Program"],
      "retrieval_keywords": ["GBSD Ogden contacts list", "Ogden team contacts",
"Northrop Ogden personnel list"],
      "cross_references": ["Ogden Org Overview.pdf"]
    },
    {
      "canonical_title": "Clearance Overview.docx",
      "bd_engine": "Org Chart",
      "artifact_type": "Document",
      "file_name": "Clearance Overview.docx",
      "drive_path": "Prime TS BD Automation/Engine 3 – Org Chart/References",
      "integration_status": "Active",
      "purpose":
"Overview of US security clearance levels and timelines (Public Trust vs DoD
clearances, expirations, etc).",
      "integration_usage":
"Utilized by Org Chart & Playbook processes to inform conversations and
planning. Ensures BD reps/AI mention clearance facts accurately (e.g., timeline
for TS investigations).",
      "workflow_phase": "Credibility",
      "prompt_tags": ["Reference", "Clearances"],
      "retrieval_keywords": ["clearance levels guide", "security clearance
timeline doc", "Public Trust vs TS overview"],
      "cross_references": []
    },
    {
      "canonical_title": "Docx Parsing Attempt (Placements)",
      "bd_engine": "Org Chart",
      "artifact_type": "Script (Python)",
      "file_name": "parse_placements_docx.py",
      "drive_path": "Prime TS BD Automation/Engine 3 – Org Chart/Data Scripts",
      "integration_status": "Archived",
      "purpose": "Throwaway Python script attempted to parse an initial
placements .docx (malformed). Early attempt to extract placement data via
code.",
      "integration_usage": "One-off utility used during data prep – tried to
salvage placement data from a bad source. Not part of production pipeline;
```

```
indicated need to get data in better format after it failed.",
      "workflow_phase": "Credibility",
      "prompt_tags": ["Utility", "Script"],
      "retrieval_keywords": ["placements parsing script",
"python-docx placement extractor", "data salvage script"],
      "cross_references": ["Engine Placements (Full Parsed Block)"]
    },
    {
      "canonical_title": "Engine_Placements_Updated.csv",
      "bd_engine": "Org Chart",
      "artifact_type": "CSV Data",
      "file_name": "Engine_Placements_Updated.csv",
      "drive_path": "Prime TS BD Automation/Engine 3 – Org Chart/Data",
      "integration_status": "Duplicate",
      "purpose": "Initial merged placement dataset (original export + first
batch of new records).",
      "integration_usage": "Intermediate combined data used to verify first
integration of new placements. Used as input to subsequent cleaning steps;
superseded by 'Updated_Full'.",
      "workflow_phase": "Credibility",
      "prompt_tags": ["Data", "Merged"],
      "retrieval_keywords": ["placements update1 CSV", "initial merged
placements", "first update placement data"],
      "cross_references": ["Engine_Placements_Updated_Full.csv"]
    },
    {
      "canonical_title": "Engine_Placements_Updated_Full.csv",
      "bd_engine": "Org Chart",
      "artifact_type": "CSV Data",
      "file_name": "Engine_Placements_Updated_Full.csv",
      "drive_path": "Prime TS BD Automation/Engine 3 – Org Chart/Data",
      "integration_status": "Duplicate",
      "purpose": "Extended placement dataset after second batch of new records
appended (full set after updates).",
      "integration_usage":
"Follow-up merged file capturing all placements after second update. Used as
checkpoint before full data cleaning on entire set. Marked duplicate since final
cleaned data resides in 'All'.",
      "workflow_phase": "Credibility",
      "prompt_tags": ["Data", "Merged"],
      "retrieval_keywords": ["placements update2 CSV", "full merged placements
after updates", "extended placement data"],
      "cross_references": ["Engine_Placements_All.csv"]
    },
    {
      "canonical_title": "Engine_Placements_Cleaned.csv",
      "bd_engine": "Org Chart",
      "artifact_type": "CSV Data",
      "file_name": "Engine_Placements_Cleaned.csv",
      "drive_path": "Prime TS BD Automation/Engine 3 – Org Chart/Data",
      "integration_status": "Duplicate",
      "purpose": "Trimmed placement dataset with unnecessary columns removed
(streamlined schema).",
      "integration_usage": "Cleaned CSV used to reduce noise in placement data.
Only relevant fields retained, simplifying mapping of placements to programs/
contacts. Feeds into final standardized dataset.",
      "workflow_phase": "Credibility",
      "prompt_tags": ["Data", "Cleaned"],
      "retrieval_keywords": ["trimmed placement data",
"reduced fields placement CSV", "cleaned placement CSV"],
      "cross_references": ["Engine_Placements_Final.csv"]
    },
    {
      "canonical_title": "Engine_Placements_Final.csv",
      "bd_engine": "Org Chart",
      "artifact_type": "CSV Data",
      "file_name": "Engine_Placements_Final.csv",
      "drive_path": "Prime TS BD Automation/Engine 3 – Org Chart/Data",
      "integration_status": "Duplicate",
      "purpose": "Near-final placement dataset after schema alignment (almost
complete; pending minor tweaks).",
      "integration_usage": "Penultimate version used for final validation –
checked all fields match target schema, then replaced by fully consolidated
'All' dataset after verification.",
      "workflow_phase": "Credibility",
      "prompt_tags": ["Data", "Cleaned"],
      "retrieval_keywords": ["final check placement data", "pre-final placement
CSV", "near-final placements dataset"],
      "cross_references": ["Engine_Placements_All.csv"]
    },
    {
      "canonical_title": "Engine_Placements_All.csv",
      "bd_engine": "Org Chart",
      "artifact_type": "CSV Data",
      "file_name": "Engine_Placements_All_v3.csv",
      "drive_path": "Prime TS BD Automation/Engine 3 – Org Chart/Data",
      "integration_status": "Active",
      "purpose": "Fully consolidated final placement dataset with all records
under the standardized schema.",
      "integration_usage": "Authoritative placement data source for Org Chart &
Scoring – contains every placement record cleaned & normalized. Imported into
DB/dataframes for use in org inference and scoring (e.g. placements count by
program).",
      "workflow_phase": "Credibility",
      "prompt_tags": ["Data", "Master"],
      "retrieval_keywords": ["all placements master CSV", "final cleaned
placements data", "consolidated placements dataset"],
      "cross_references": ["Engine Placements (Full Cleaned)"]
    },
    {
      "canonical_title": "Cleared_Cable_Tech_Sourcing_Packet.pdf",
      "bd_engine": "Org Chart",
      "artifact_type": "Document",
      "file_name": "Cleared_Cable_Tech_Sourcing_Packet.pdf",
      "drive_path": "Prime TS BD Automation/Engine 3 – Org Chart/References",
      "integration_status": "Active",
      "purpose": "Example sourcing packet for a Cleared Cable Technician role
(job requirements + candidate profiles).",
      "integration_usage": "Reference for sourcing strategies. Org Chart &
Playbook engines may refer to it to see how talent is presented for similar
roles, informing how to frame candidates to clients in outreach.",
      "workflow_phase": "Credibility",
      "prompt_tags": ["Reference", "Sourcing"],
      "retrieval_keywords": ["sourcing packet example", "cleared role candidate
packet", "Cable Tech packet PDF"],
      "cross_references": []
    },
    {
      "canonical_title": "Michael Staff LinkedIn Activity (Export).pdf",
      "bd_engine": "Org Chart",
      "artifact_type": "Document",
      "file_name": "Michael Staff LinkedIn Activity - Export.pdf",
      "drive_path": "Prime TS BD Automation/Engine 3 – Org Chart/Program Intel",
      "integration_status": "Active",
      "purpose": "Exported LinkedIn activity feed for Michael Staff (target
individual’s posts & engagements).",
      "integration_usage": "Input to Playbook Engine’s research – agent
summarizes this feed to personalize outreach (e.g., referencing a topic the
person posted about). Provides HUMINT details to tailor messaging.",
      "workflow_phase": "Credibility",
      "prompt_tags": ["Contact Intel", "Social"],
      "retrieval_keywords": ["LinkedIn activity export", "contact social media
posts", "Michael Staff LinkedIn feed"],
      "cross_references": []
    }
  ],
  "Playbook": [
    {
      "canonical_title": "Sales Prospecting Playbook.pdf",
      "bd_engine": "Playbook",
      "artifact_type": "Document",
      "file_name": "Sales Prospecting Playbook.pdf",
      "drive_path": "Prime TS BD Automation/Engine 4 – Playbook/Master
Playbook",
      "integration_status": "Active",
      "purpose":
"Comprehensive sales prospecting playbook (scripts, strategies, rebuttals,
frameworks) tailored to cleared BD.",
      "integration_usage":
"Content foundation for Playbook Engine – OpenAI uses this to generate outreach
scripts and bullet points. The engine assembles program-specific playbooks by
drawing from these templates, ensuring consistent tactics.",
      "workflow_phase": "Insertion",
      "prompt_tags": ["Playbook", "Outreach"],
      "retrieval_keywords": ["Prime TS sales playbook", "prospecting scripts
PDF", "BD outreach playbook"],
      "cross_references": ["Competitive Capture & Insertion Strategy for
PTS.pdf"]
    },
    {
      "canonical_title": "Competitive Capture & Insertion Strategy for PTS.pdf",
      "bd_engine": "Playbook",
      "artifact_type": "Document",
      "file_name": "Competitive Capture & Insertion Strategy for Prime TS.pdf",
      "drive_path": "Prime TS BD Automation/Engine 4 – Playbook/Strategy
Guides",
      "integration_status": "Active",
      "purpose": "Playbook detailing tactics to insert Prime TS into competitor-
held contracts (sub-to-prime wedge strategies, competitor weaknesses).",
      "integration_usage": "Used by Playbook & Mapping engines for planning –
when a competitor is incumbent, the system references these tactics (e.g.,
leverage SDVOSB status, fast reaction) to suggest how Prime TS can position as a
subcontractor or alternative.",
      "workflow_phase": "Insertion",
      "prompt_tags": ["Strategy", "Competition"],
      "retrieval_keywords": ["competitor wedge strategy", "subcontractor
insertion tactics", "competitive capture playbook"],
      "cross_references": ["Sales Prospecting Playbook.pdf"]
    },
    {
      "canonical_title": "Prime_TS_BD_Attack_Plan.pdf",
      "bd_engine": "Playbook",
      "artifact_type": "Document",
      "file_name": "Prime_TS_BD_Attack_Plan.pdf",
      "drive_path": "Prime TS BD Automation/Engine 4 – Playbook/Agent Outputs",
      "integration_status": "Draft (AI Generated)",
      "purpose": "Draft BD “Attack Plan” for Prime TS – AI-generated plan with
BD actions, narratives, and 30-day schedule integrating all engine outputs.",
      "integration_usage":
"Consolidated action plan produced by the agent after analyzing strategy docs.
Used in maintenance to guide team execution – loaded into trackers or shared to
ensure all recommended tasks are addressed. (Draft until reviewed by team.)",
      "workflow_phase": "Insertion",
      "prompt_tags": ["Agent Output", "Plan"],
      "retrieval_keywords": ["AI-generated attack plan", "Prime TS BD action
plan", "30-day BD plan AI"],
      "cross_references": ["BD Target List", "Events Calendar",
"Org Charts (per program)"]
    },
    {
      "canonical_title": "Agent Mode Prompt – Prime Technical BD Attack Plan",
      "bd_engine": "Cross-Engine",
      "artifact_type": "Prompt (JSON)",
      "file_name": "AgentMode_PrimeTS_BD_AttackPlan.json",
      "drive_path": "Prime TS BD Automation/Agent Prompts",
      "integration_status": "Active",
      "purpose": "Configured GPT prompt that guides an AI agent to analyze all
BD strategy docs and output a tailored “Attack Plan”.",
      "integration_usage": "Executed in Agent Mode – when invoked, the agent
uses this prompt to read relevant artifacts and produce the comprehensive BD
Attack Plan (as a PDF). Ensures the plan covers targets, credibility points,
actions, timelines.",
      "workflow_phase": "Maintenance",
      "prompt_tags": ["AI Prompt", "Agent"],
      "retrieval_keywords": ["attack plan AI prompt", "BD plan generation
prompt", "agent instruction JSON"],
      "cross_references": ["Prime_TS_BD_Attack_Plan.pdf"]
    },
    {
      "canonical_title": "Master Prompt Template",
      "bd_engine": "Cross-Engine",
      "artifact_type": "Prompt (Text)",
      "file_name": "Master_Prompt_Template.txt",
      "drive_path": "Prime TS BD Automation/Agent Prompts",
      "integration_status": "Active",
      "purpose": "Master structured prompt template for agent chats
(standardizes outputs and includes delta analysis logic).",
      "integration_usage": "Used in all Agent Mode runs – provides a consistent
format (Decisions, Outputs, Blockers, Next Steps) for summarizing engine
updates, and instructions to perform Old vs New delta analysis of documents.
Ensures uniform reporting across chats.",
      "workflow_phase": "Maintenance",
      "prompt_tags": ["AI Prompt", "Template"],
      "retrieval_keywords": ["agent master prompt", "chatGPT structure
template", "delta analysis prompt template"],
      "cross_references": []
    },
    {
      "canonical_title": "PTS_BD_Master_Trackers.xlsx",
      "bd_engine": "Cross-Engine",
      "artifact_type": "Spreadsheet",
      "file_name": "PTS_BD_Master_Trackers.xlsx",
      "drive_path": "Prime TS BD Automation/Trackers & Sheets",
      "integration_status": "Active",
      "purpose":
"Dual-sheet BD tracker for (1) Target Programs Intelligence and (2) Prime TS
Active Pursuits (captures program statuses and pursuit progress).",
      "integration_usage": "Used as living tracker by BD team (and potentially
updated via Airtable or API). Agents refer to this in maintenance to identify
stale pursuits or to adjust priorities, ensuring no target is overlooked.",
      "workflow_phase": "Maintenance",
      "prompt_tags": ["Tracker", "Sheet"],
      "retrieval_keywords": ["BD master tracker", "target programs tracker",
"active pursuits spreadsheet"],
      "cross_references": []
    },
    {
      "canonical_title": "Q3–Q4 2025 DoD BD Events Calendar",
      "bd_engine": "Cross-Engine",
      "artifact_type": "Calendar (Spreadsheet)",
      "file_name": "2025_DOD_BD_Events_Calendar.xlsx",
      "drive_path": "Prime TS BD Automation/Trackers & Sheets",
      "integration_status": "Active",
      "purpose": "Calendar of major defense/IC industry events in Q3 & Q4 2025
(with notes for BD relevance).",
      "integration_usage": "Referenced in maintenance – informs scheduling of
outreach around key events and in-person intel gathering opportunities. Agents
might use it to recommend meeting timing or flag upcoming events for
engagement.",
      "workflow_phase": "Maintenance",
      "prompt_tags": ["Calendar", "Events"],
      "retrieval_keywords": ["2025 defense events calendar", "Q3 Q4 events
schedule", "DoD industry events list"],
      "cross_references": []
    }
  ],
  "Scoring": [
    {
      "canonical_title": "Scoring Engine – Weighted Model Draft",
      "bd_engine": "Scoring",
      "artifact_type": "Document",
      "file_name": "Scoring Engine Weighted Model Draft.docx",
      "drive_path": "Prime TS BD Automation/Engine 5 – Scoring/Design & Specs",
      "integration_status": "Active",
      "purpose": "Draft weighted scoring model for opportunities (proposed
weights: demand 30%, pain 20%, contract value 20%, past perf 20%, competition
10%).",
      "integration_usage": "Blueprint for Scoring Engine implementation.
```

```
Provides criteria & formula to be coded (e.g. in Python or SQL) once enough data
is available. Serves as agreed-upon model definition pre-implementation.",
      "workflow_phase": "Scoring",
      "prompt_tags": ["Design", "Model"],
      "retrieval_keywords": ["scoring model draft", "opportunity scoring
weights", "scoring criteria doc"],
      "cross_references": []
    },
    {
      "canonical_title": "API Integration Notes for Scoring (USAspending &
FPDS)",
      "bd_engine": "Scoring",
      "artifact_type": "Document",
      "file_name": "API Integration Notes for Scoring.pdf",
      "drive_path": "Prime TS BD Automation/Engine 5 – Scoring/Design & Specs",
      "integration_status": "Active",
      "purpose": "Notes on integrating external data (USAspending API, FPDS
scraper) to enhance scoring with contract award info.",
      "integration_usage": "Guides development of data feeds into Scoring –
outlines how to pull contract values, prime vs sub data, etc., so the scoring
model can factor financial metrics. Directly informs how FPDS scraper &
USAspending data tie into scoring calculations.",
      "workflow_phase": "Scoring",
      "prompt_tags": ["Design", "Integration"],
      "retrieval_keywords": ["scoring integration notes", "FPDS USAspending
integration", "external data for scoring"],
      "cross_references": []
    },
    {
      "canonical_title": "Top Opportunities (Most to Least Strategic).pdf",
      "bd_engine": "Scoring",
      "artifact_type": "Document",
      "file_name": "Top Opportunities (Most to Least Strategic).pdf",
      "drive_path": "Prime TS BD Automation/Engine 5 – Scoring/Reference
Reports",
      "integration_status": "Active",
      "purpose": "Human-ranked list of BD opportunities (programs) from most to
least strategic (leadership’s subjective ranking).",
      "integration_usage": "Used as input/validation for automated scoring –
ensures the Scoring Engine’s results align with or intelligently diverge from
leadership’s priorities. Helps calibrate weightings by comparing model output to
this list.",
      "workflow_phase": "Scoring",
      "prompt_tags": ["Reference", "Ranking"],
      "retrieval_keywords": ["strategic opportunities ranking", "leadership BD
priorities list", "most strategic programs report"],
      "cross_references": []
    }
```

```
  ],
  "Cross-Engine": [
    {
      "canonical_title":
"AI Automation Society N8n Templates – Categorization & Usage Guide.pdf",
      "bd_engine": "Cross-Engine",
      "artifact_type": "Document",
      "file_name":
"AI Automation Society N8n Templates – Categorization & Usage Guide.pdf",
      "drive_path": "Prime TS BD Automation/External References",
      "integration_status": "Active",
      "purpose": "Catalog of community N8n templates (from AI Automation
Society) by category, with explanations and usage recommendations.",
      "integration_usage": "Reference library of automation patterns. Not
executed directly, but provides ideas and best practices for building/enhancing
our n8n workflows (multi-agent coordination, error handling, etc.). Used as
knowledge base for dev team.",
      "workflow_phase": "Maintenance",
      "prompt_tags": ["Reference", "Templates"],
      "retrieval_keywords": ["AI Society N8n guide", "community templates
mapping", "automation templates usage guide"],
      "cross_references": ["All JSON templates listed below"]
    },
    {
      "canonical_title": "NewsletterAgentTeam.json",
      "bd_engine": "Cross-Engine",
      "artifact_type": "Workflow (n8n)",
      "file_name": "NewsletterAgentTeam.json",
      "drive_path": "Prime TS BD Automation/External Templates",
      "integration_status": "Template",
      "purpose": "N8n workflow template for an AI-generated newsletter team
(delegates news curation, summarization, email drafting to sub-agents).",
      "integration_usage": "Import into n8n to automate end-to-end newsletter
creation. Coordinates multiple GPT agents, each handling part of the process
(news gatherer, summarizer, email writer). Useful example of parallel agent
use.",
      "workflow_phase": "Maintenance",
      "prompt_tags": ["Template", "Workflow"],
      "retrieval_keywords": ["newsletter agent workflow", "multi-agent
newsletter automation", "AI sub-agent newsletter"],
      "cross_references": []
    },
    {
      "canonical_title": "UltimateMediaAgent.json",
      "bd_engine": "Cross-Engine",
      "artifact_type": "Workflow (n8n)",
      "file_name": "UltimateMediaAgent.json",
      "drive_path": "Prime TS BD Automation/External Templates",
      "integration_status": "Template",
      "purpose": "N8n workflow template (“Ultimate Media Agent”) for automating
multi-channel content creation (social posts, videos, graphics) and
distribution.",
      "integration_usage": "Use as a base for media automation. Requires
configuring image/video generation APIs and social media API credentials.
Demonstrates orchestrating various creative tasks (text, image, video) in one
workflow.",
      "workflow_phase": "Maintenance",
      "prompt_tags": ["Template", "Media"],
      "retrieval_keywords": ["media agent workflow", "AI content creation
pipeline", "social media automation template"],
      "cross_references": []
    },
    {
      "canonical_title": "9Socials.json",
      "bd_engine": "Cross-Engine",
      "artifact_type": "Workflow (n8n)",
      "file_name": "9Socials.json",
      "drive_path": "Prime TS BD Automation/External Templates",
      "integration_status": "Template",
      "purpose": "N8n workflow template to post content to 9 social platforms
simultaneously.",
      "integration_usage": "Import and configure with social platform API keys.
Enables broadcasting content across all major networks in one execution. Useful
for multi-platform announcements.",
      "workflow_phase": "Maintenance",
      "prompt_tags": ["Template", "Social"],
      "retrieval_keywords": ["multi-platform posting workflow", "social media
auto-post template", "9 socials automation JSON"],
      "cross_references": []
    },
    {
      "canonical_title": "ErrorHandling.json",
      "bd_engine": "Cross-Engine",
      "artifact_type": "Workflow (n8n)",
      "file_name": "UnlimitedErrorHandling.json",
      "drive_path": "Prime TS BD Automation/External Templates",
      "integration_status": "Template",
      "purpose": "N8n workflow template for a robust error handling system
(global catcher with retries & notifications).",
      "integration_usage": "Embed as a sub-workflow or global error handler.
Other workflows trigger this on failure to log error details, send alerts, and
optionally retry. Ensures automations continue running by handling errors
centrally.",
      "workflow_phase": "Maintenance",
      "prompt_tags": ["Template", "Error Handling"],
      "retrieval_keywords": ["global error handler workflow", "n8n unlimited
error handling template", "error catch and retry JSON"],
      "cross_references": ["ErrorWorkflow.json"]
    },
    {
      "canonical_title": "ElevenLabsVoice.json",
      "bd_engine": "Cross-Engine",
      "artifact_type": "Workflow (n8n)",
      "file_name": "ElevenLabsVoice.json",
      "drive_path": "Prime TS BD Automation/External Templates",
      "integration_status": "Template",
      "purpose": "N8n workflow for an AI voice agent using ElevenLabs text-to-
speech (gives AI a voice).",
      "integration_usage":
"Import to add voice output to agents. Configure with ElevenLabs API
credentials; agent can then convert text responses to audio files. Useful for
voice assistants or generating spoken updates.",
      "workflow_phase": "Maintenance",
      "prompt_tags": ["Template", "Voice"],
      "retrieval_keywords": ["AI voice agent template", "text-to-speech
workflow", "ElevenLabs n8n example"],
      "cross_references": []
    },
    {
      "canonical_title": "AgentSwarm.json",
      "bd_engine": "Cross-Engine",
      "artifact_type": "Workflow (n8n)",
      "file_name": "AgentSwarm.json",
      "drive_path": "Prime TS BD Automation/External Templates",
      "integration_status": "Template",
      "purpose": "N8n workflow enabling an AI manager-agent to spawn &
coordinate multiple sub-agents in parallel.",
      "integration_usage": "Import to implement multi-agent orchestration. The
manager agent delegates tasks to sub-agents (as separate processes) and
aggregates results. Useful for parallelizing parts of a complex task (research,
writing, checking).",
      "workflow_phase": "Maintenance",
      "prompt_tags": ["Template", "Multi-Agent"],
      "retrieval_keywords": ["multi-agent orchestration JSON", "agent swarm
workflow", "parallel agents template"],
      "cross_references": ["MultiAgentSystem.json"]
    },
    {
      "canonical_title": "Metadata.json",
      "bd_engine": "Cross-Engine",
      "artifact_type": "Workflow (n8n)",
      "file_name": "Metadata.json",
      "drive_path": "Prime TS BD Automation/External Templates",
      "integration_status": "Template",
      "purpose": "N8n workflow template for enriching or handling metadata
(extracts keywords, descriptions, tags from content or uses metadata for
routing).",
      "integration_usage": "Adapt to any process requiring metadata management.
E.g., trigger on file upload to auto-tag with AI-generated keywords, or
integrate into content pipeline to add descriptions. Improves downstream search
& organization by adding metadata to raw info.",
      "workflow_phase": "Maintenance",
      "prompt_tags": ["Template", "Utility"],
      "retrieval_keywords": ["metadata tagging workflow", "auto-tag files
template", "metadata extraction n8n example"],
      "cross_references": []
    },
    {
      "canonical_title": "FirstRAGAgent.json",
      "bd_engine": "Cross-Engine",
      "artifact_type": "Workflow (n8n)",
      "file_name": "FirstRAGAgent.json",
      "drive_path": "Prime TS BD Automation/External Templates",
      "integration_status": "Template",
      "purpose": "Basic Retrieval-Augmented Generation (RAG) agent workflow –
fetches info from a knowledge source then uses it to answer with GPT.",
      "integration_usage": "Use to build an AI agent that consults external
data. Connect to a document store or API: the workflow performs a search step
before the GPT node forms an answer, allowing grounded responses using up-to-
date info. Good foundation for Q&A bots that require knowledge lookup.",
      "workflow_phase": "Maintenance",
      "prompt_tags": ["Template", "RAG"],
      "retrieval_keywords": ["RAG agent workflow", "retrieval augmented
template", "search-then-answer agent JSON"],
      "cross_references": ["RerankingRAG.json"]
    },
    {
      "canonical_title": "WebhookSecurity.json",
      "bd_engine": "Cross-Engine",
      "artifact_type": "Workflow (n8n)",
      "file_name": "WebhookSecurity.json",
      "drive_path": "Prime TS BD Automation/External Templates",
      "integration_status": "Template",
      "purpose":
"N8n workflow template to secure incoming webhooks (uses secret tokens/
signatures to verify authenticity before proceeding).",
      "integration_usage": "Apply to any n8n webhook that needs protection. The
workflow checks a token or signature on incoming requests and only processes
them if valid. Prevents unauthorized or malicious triggers from executing
workflows.",
      "workflow_phase": "Maintenance",
      "prompt_tags": ["Template", "Security"],
      "retrieval_keywords": ["webhook security workflow", "verify webhook
template", "secure webhook trigger JSON"],
      "cross_references": []
    },
    {
      "canonical_title": "ZepMemory.json",
      "bd_engine": "Cross-Engine",
      "artifact_type": "Workflow (n8n)",
      "file_name": "ZepMemory.json",
      "drive_path": "Prime TS BD Automation/External Templates",
      "integration_status": "Template",
      "purpose": "N8n workflow integrating Zep long-term memory for AI agents
(store/retrieve conversational memory via Zep API).",
      "integration_usage": "Import to give AI agents persistent memory.
Configure with Zep API endpoint; agent can then save conversation context or Q&A
pairs to Zep and retrieve them in later interactions, enabling continuity across
sessions.",
      "workflow_phase": "Maintenance",
      "prompt_tags": ["Template", "Memory"],
      "retrieval_keywords": ["agent memory template", "Zep long-term memory
workflow", "persistent memory agent JSON"],
      "cross_references": []
    },
    {
      "canonical_title": "Parallelization.json",
      "bd_engine": "Cross-Engine",
      "artifact_type": "Workflow (n8n)",
      "file_name": "Parallelization.json",
      "drive_path": "Prime TS BD Automation/External Templates",
      "integration_status": "Template",
      "purpose": "N8n workflow template demonstrating parallel task execution
(multiple branches running concurrently to speed up processing).",
      "integration_usage": "Use to structure workflows that benefit from
concurrency. Shows how to use branches or Split In Batches for parallel
execution. Paired with a reference PDF on sub-workflows & parallelization for
optimization tips.",
      "workflow_phase": "Maintenance",
      "prompt_tags": ["Template", "Optimization"],
      "retrieval_keywords": ["parallel tasks template", "concurrent workflow
example", "n8n parallel execution JSON"],
      "cross_references": ["n8n Subworkflows & Parallelization.pdf"]
    },
    {
      "canonical_title": "ResumeScreeningSystem.json",
      "bd_engine": "Cross-Engine",
      "artifact_type": "Workflow (n8n)",
      "file_name": "ResumeScreeningSystem.json",
      "drive_path": "Prime TS BD Automation/External Templates",
      "integration_status": "Template",
      "purpose": "N8n workflow for an AI-driven resume screening system
(automates intake, parsing, and initial candidate evaluation).",
      "integration_usage": "Leverage for recruiting automation. Connect to
resume sources (inbox, form) and use an AI parser to extract skills/experience
into structured data. Workflow can then auto-rank or route candidates. Likely
needs an OCR or resume parsing API (like GPT) configured.",
      "workflow_phase": "Maintenance",
      "prompt_tags": ["Template", "HR"],
      "retrieval_keywords": ["resume screening template", "AI resume parser
workflow", "candidate filtering automation JSON"],
      "cross_references": []
    },
    {
      "canonical_title": "RerankingRAG.json",
      "bd_engine": "Cross-Engine",
      "artifact_type": "Workflow (n8n)",
      "file_name": "RerankingRAG.json",
      "drive_path": "Prime TS BD Automation/External Templates",
      "integration_status": "Template",
      "purpose":
"Advanced RAG agent workflow – agent generates multiple answers and re-ranks
them to choose the best response.",
      "integration_usage": "Use when answer quality must be high. Template
likely runs several GPT query variations, then uses an evaluation step (another
AI or heuristic) to score and select the best answer. Great for research or
support agents to ensure optimal answers.",
      "workflow_phase": "Maintenance",
      "prompt_tags": ["Template", "RAG"],
      "retrieval_keywords": ["RAG rerank workflow", "answer re-ranking agent
template", "multiple answers selection JSON"],
      "cross_references": ["FirstRAGAgent.json"]
    },
    {
      "canonical_title": "YouTubeStrategist.json",
      "bd_engine": "Cross-Engine",
      "artifact_type": "Workflow (n8n)",
      "file_name": "YouTubeStrategist.json",
      "drive_path": "Prime TS BD Automation/External Templates",
      "integration_status": "Template",
      "purpose": "N8n workflow for a YouTube content strategy agent (researches
trending topics, generates video ideas/scripts, possibly schedules content).",
      "integration_usage": "Import for automating YouTube channel planning.
Agent can integrate with YouTube API for data, use GPT to brainstorm or draft
scripts, then output content ideas or posting schedules. Useful for content
teams wanting AI-assisted strategy.",
      "workflow_phase": "Maintenance",
      "prompt_tags": ["Template", "Content"],
      "retrieval_keywords": ["YouTube strategy workflow", "video ideas AI
template", "YouTube content agent JSON"],
      "cross_references": []
    },
    {
      "canonical_title": "GlassFruitASMR.json",
      "bd_engine": "Cross-Engine",
      "artifact_type": "Workflow (n8n)",
      "file_name": "GlassFruitASMR.json",
      "drive_path": "Prime TS BD Automation/External Templates",
      "integration_status": "Template",
      "purpose": "N8n workflow example from a challenge – generates ASMR-style
content via AI (creative, unconventional use-case).",
      "integration_usage": "Primarily a creative demo. If imported, ensure
required AI generators (audio/visual) are configured. Shows integration of
unconventional AI tools in workflows. Not directly applicable to BD, but sparks
ideas for agent creativity.",
      "workflow_phase": "Maintenance",
      "prompt_tags": ["Template", "Creative"],
      "retrieval_keywords": ["ASMR content workflow", "creative AI challenge
template", "GlassFruit ASMR JSON"],
      "cross_references": []
    },
    {
      "canonical_title": "DeveloperAgent.json",
      "bd_engine": "Cross-Engine",
      "artifact_type": "Workflow (n8n)",
      "file_name": "DeveloperAgent.json",
      "drive_path": "Prime TS BD Automation/External Templates",
      "integration_status": "Template",
      "purpose": "N8n workflow for an AI software development assistant (code
snippet generation, code review, NL to code conversion).",
      "integration_usage": "Use to augment dev workflows. For example, set an
agent to watch a GitHub repo for new issues and suggest code, or answer dev
questions by generating code. Configure with an OpenAI Codex-like model for best
results. Provides partial automation for development tasks (with human
oversight).",
      "workflow_phase": "Maintenance",
      "prompt_tags": ["Template", "Coding"],
      "retrieval_keywords": ["developer assistant workflow", "AI coding agent
template", "code generation agent JSON"],
      "cross_references": []
    },
    {
      "canonical_title": "VideoAnalysis.json",
      "bd_engine": "Cross-Engine",
      "artifact_type": "Workflow (n8n)",
      "file_name": "VideoAnalysis.json",
      "drive_path": "Prime TS BD Automation/External Templates",
      "integration_status": "Template",
      "purpose": "N8n workflow automating video content analysis (downloads
video, uses AI to transcribe and summarize or tag key topics).",
      "integration_usage":
"Great for media monitoring. Configure a video source (URL or file) and
integrate with transcription (e.g., Whisper) and analysis (maybe GPT for
summarizing). Output could be a text summary or topic list. Helps quickly digest
long videos or index them for search.",
      "workflow_phase": "Maintenance",
      "prompt_tags": ["Template", "Video"],
      "retrieval_keywords": ["video analysis workflow", "AI video summary
template", "video content analyzer JSON"],
      "cross_references": []
    },
    {
      "canonical_title": "BrowserAgent.json",
      "bd_engine": "Cross-Engine",
      "artifact_type": "Workflow (n8n)",
      "file_name": "BrowserAgent.json",
      "drive_path": "Prime TS BD Automation/External Templates",
      "integration_status": "Template",
      "purpose": "N8n workflow for a web-browsing AI agent that can navigate
websites, take screenshots, and scrape content via natural language
instructions.",
      "integration_usage": "Use to give agents browsing capability. Requires
headless browser integration (Puppeteer or similar). Agent can be instructed to
go to specified URLs and retrieve info or screenshots. Ensure to include delays
and respect site rules to avoid detection. Useful for AI-driven web research
tasks.",
      "workflow_phase": "Maintenance",
      "prompt_tags": ["Template", "Web"],
      "retrieval_keywords": ["browser agent workflow", "AI web navigation
JSON", "web scraping agent template"],
      "cross_references": ["FirecrawlSearchScrape.json",
"FirecrawlExtract.json"]
    },
    {
      "canonical_title": "FirecrawlSearchScrape.json",
      "bd_engine": "Cross-Engine",
      "artifact_type": "Workflow (n8n)",
      "file_name": "FirecrawlSearchScrape.json",
      "drive_path": "Prime TS BD Automation/External Templates",
      "integration_status": "Template",
      "purpose":
"\"Firecrawl\" agent workflow – performs web search queries and scrapes data
(including screenshots) without external proxies.",
      "integration_usage": "Import to build internal search+scrape capability.
```

```
Likely uses Google Custom Search for web search and HTML extract or screenshot
nodes to capture page data. Useful when AI needs to pull live web info
autonomously as part of a workflow (complements BrowserAgent but focused on
search results).",
      "workflow_phase": "Maintenance",
      "prompt_tags": ["Template", "Web"],
      "retrieval_keywords": ["firecrawl search workflow", "search and scrape
agent JSON", "web scraping agent example"],
      "cross_references": ["BrowserAgent.json"]
    },
    {
      "canonical_title": "FirecrawlExtract.json",
      "bd_engine": "Cross-Engine",
      "artifact_type": "Workflow (n8n)",
      "file_name": "FirecrawlExtract.json",
      "drive_path": "Prime TS BD Automation/External Templates",
      "integration_status": "Template",
      "purpose":
"Companion workflow to FirecrawlSearchScrape – iterates through a list of known
URLs and extracts specific data from each page.",
      "integration_usage": "Use when you have URLs to scrape in detail.
Configure with parsing logic (CSS selectors, regex) for target fields. This
workflow can be fed URLs (e.g. from FirecrawlSearchScrape results or known
sources) to systematically extract structured info. Good for scraping sets of
pages after discovery.",
      "workflow_phase": "Maintenance",
      "prompt_tags": ["Template", "Web"],
      "retrieval_keywords": ["firecrawl extract workflow", "detailed scraper
template", "HTML extraction JSON"],
      "cross_references": ["FirecrawlSearchScrape.json"]
    },
    {
      "canonical_title": "25n8nHacks.json",
      "bd_engine": "Cross-Engine",
      "artifact_type": "Workflow (n8n)",
      "file_name": "25n8nHacks.json",
      "drive_path": "Prime TS BD Automation/External Templates",
      "integration_status": "Template",
      "purpose": "Workflow showcasing 25 N8n productivity hacks/hidden features
(mini-examples of various nodes and tricks).",
      "integration_usage": "Educational – import to see and test 25 distinct
examples of N8n capabilities in one workflow. Useful for training or discovering
techniques (these examples can be copied into real workflows where needed).
Essentially a tutorial compilation.",
      "workflow_phase": "Maintenance",
      "prompt_tags": ["Template", "Tutorial"],
      "retrieval_keywords": ["25 n8n hacks workflow", "n8n tips and tricks
examples", "n8n tutorial workflow JSON"],
      "cross_references": []
    },
    {
      "canonical_title": "ShortsMachine.json",
      "bd_engine": "Cross-Engine",
      "artifact_type": "Workflow (n8n)",
      "file_name": "ShortsMachine.json",
      "drive_path": "Prime TS BD Automation/External Templates",
      "integration_status": "Template",
      "purpose": "24/7 viral shorts generator workflow – automates continuous
creation and posting of short-form videos (YouTube Shorts/TikTok).",
      "integration_usage":
"Import to explore how to chain content creation in a loop. Likely uses AI for
idea generation, text-to-video APIs for content creation, and scheduling nodes
for posting repeatedly. Good for understanding indefinite agent runs and
scheduling within workflows.",
      "workflow_phase": "Maintenance",
      "prompt_tags": ["Template", "Content"],
      "retrieval_keywords": ["shorts generator workflow", "continuous content
loop JSON", "viral shorts automation example"],
      "cross_references": []
    },
    {
      "canonical_title": "APIsforAgents.json",
      "bd_engine": "Cross-Engine",
      "artifact_type": "Workflow (n8n)",
      "file_name": "APIsforAgents.json",
      "drive_path": "Prime TS BD Automation/External Templates",
      "integration_status": "Template",
      "purpose": "Workflow demo of AI agents using external APIs (examples:
fetch weather, stock prices, etc., and incorporate data into responses).",
      "integration_usage": "Use as template to extend agents with API tools.
After import, replace with domain-relevant API calls (e.g. CRM API for client
data, calendar API for scheduling). Shows how agents can become more actionable
by retrieving real-world data mid-process.",
      "workflow_phase": "Maintenance",
      "prompt_tags": ["Template", "Integration"],
      "retrieval_keywords": ["agents using APIs workflow", "external API call
agent JSON", "tool-using agent template"],
      "cross_references": []
    },
    {
      "canonical_title": "TrackAgentActions.json",
      "bd_engine": "Cross-Engine",
      "artifact_type": "Workflow (n8n)",
      "file_name": "TrackAgentActions.json",
      "drive_path": "Prime TS BD Automation/External Templates",
      "integration_status": "Template",
      "purpose": "Workflow for logging/monitoring AI agent usage (tracks tokens
used, API calls, timestamps for each run).",
      "integration_usage": "Integrate as monitoring for agent workflows. Each
agent execution triggers this with context (agent name, token count, etc.),
which then logs to a sheet/DB or sends alerts. Aids in analyzing usage patterns
(spikes in token use) and controlling costs or detecting runaway agents.",
      "workflow_phase": "Maintenance",
      "prompt_tags": ["Template", "Monitoring"],
      "retrieval_keywords": ["agent usage logging JSON", "AI agent monitor
workflow", "token usage tracker template"],
      "cross_references": []
    },
    {
      "canonical_title": "DynamicBrain.json",
      "bd_engine": "Cross-Engine",
      "artifact_type": "Workflow (n8n)",
      "file_name": "DynamicBrain.json",
      "drive_path": "Prime TS BD Automation/External Templates",
      "integration_status": "Template",
      "purpose": "Workflow for an AI agent with a dynamic knowledge base
(“brain”) – agent stores info it learns (e.g. to a vector DB) and retrieves it
later to improve future responses.",
      "integration_usage": "Implement for agents needing to accumulate
knowledge. Likely uses a vector store for semantic memory. Agent saves Q&A or
facts after interactions and uses them to answer similar questions later without
new API calls. Useful for agents that interact repeatedly on a topic or with a
user, building up context over time.",
      "workflow_phase": "Maintenance",
      "prompt_tags": ["Template", "Memory"],
      "retrieval_keywords": ["dynamic memory agent JSON", "agent vector memory
workflow", "self-learning agent template"],
      "cross_references": []
    },
    {
      "canonical_title": "ProductVideography.json",
      "bd_engine": "Cross-Engine",
      "artifact_type": "Workflow (n8n)",
      "file_name": "ProductVideography.json",
      "drive_path": "Prime TS BD Automation/External Templates",
      "integration_status": "Template",
      "purpose": "Workflow automating product videography creation – takes
product info and images to generate short promo videos using AI.",
      "integration_usage":
"Adapt for marketing automation. Provide product data (images, descriptions),
integrate with an AI video or slideshow generation service, and optionally an
audio voiceover. The workflow likely creates a script via GPT then calls a video
API to produce a video. Useful for scaling creation of product demo videos for
many products quickly.",
      "workflow_phase": "Maintenance",
      "prompt_tags": ["Template", "Media"],
      "retrieval_keywords": ["product video generator JSON", "AI product
videography workflow", "promo video automation template"],
      "cross_references": []
    },
    {
      "canonical_title": "AIMarketingTeam.json",
      "bd_engine": "Cross-Engine",
      "artifact_type": "Workflow (n8n)",
      "file_name": "AIMarketingTeam.json",
      "drive_path": "Prime TS BD Automation/External Templates",
      "integration_status": "Template",
      "purpose":
"Workflow for an “AI Marketing Team” agent – automates multiple marketing tasks
(content creation, scheduling, basic design) sequentially as if done by a
team.",
      "integration_usage": "Use to automate various marketing outputs together.
Template likely has one part writing a blog or social post, another creating an
image (via AI like DALL-E or Canva API), another scheduling posts via an API.
Customize topics and branding and let the agent produce multi-format content.
Good for small teams to cover content production needs across channels using
AI.",
      "workflow_phase": "Maintenance",
      "prompt_tags": ["Template", "Marketing"],
      "retrieval_keywords": ["AI marketing team workflow", "marketing tasks
automation JSON", "multi-step marketing agent template"],
      "cross_references": []
    },
    {
      "canonical_title": "LinkedInPostGraphic.json",
      "bd_engine": "Cross-Engine",
      "artifact_type": "Workflow (n8n)",
      "file_name": "LinkedInPostGraphic.json",
      "drive_path": "Prime TS BD Automation/External Templates",
      "integration_status": "Template",
      "purpose": "Workflow for generating a LinkedIn post with an AI-generated
graphic – agent writes a post and creates a simple accompanying image.",
      "integration_usage": "Ideal for LinkedIn content automation. Provide a
topic, the workflow uses GPT to draft the post text, then calls an image
generation API (like DALL-E) to produce a relevant graphic. The result can then
be posted. Demonstrates combining text and image generation in one agent flow.",
      "workflow_phase": "Maintenance",
      "prompt_tags": ["Template", "Content"],
      "retrieval_keywords": ["LinkedIn post generator JSON", "post + graphic AI
workflow", "social post image automation example"],
      "cross_references": []
    },
    {
      "canonical_title": "ErrorWorkflow.json",
      "bd_engine": "Cross-Engine",
      "artifact_type": "Workflow (n8n)",
      "file_name": "ErrorWorkflow_Basic.json",
      "drive_path": "Prime TS BD Automation/External Templates",
      "integration_status": "Template",
      "purpose":
"Earlier (simpler) error handling workflow – catches errors and sends
notifications (no advanced retry logic).",
      "integration_usage": "Drop-in basic error notifier. Attach via error
triggers on important workflows to get email/Slack alerts on failures. Useful if
the full error handling system is not implemented; provides immediate failure
visibility to the team.",
      "workflow_phase": "Maintenance",
      "prompt_tags": ["Template", "Error Handling"],
      "retrieval_keywords": ["basic error handler JSON", "error notification
workflow", "simple error alert template"],
      "cross_references": ["ErrorHandling.json"]
    },
    {
      "canonical_title": "ThinkTool.json",
      "bd_engine": "Cross-Engine",
      "artifact_type": "Workflow (n8n)",
      "file_name": "ThinkTool.json",
      "drive_path": "Prime TS BD Automation/External Templates",
      "integration_status": "Template",
      "purpose":
"Workflow using Anthropic’s “Think” chain-of-thought method for AI – prompts the
AI to outline its reasoning before final answer.",
      "integration_usage":
"Use to improve agent reasoning. The workflow has the AI first produce a step-
by-step plan (“think”) and then a second step to execute/answer. Good for
complex questions – encourages the AI to break down problems, which can yield
more logical answers and make its reasoning auditable.",
      "workflow_phase": "Maintenance",
      "prompt_tags": ["Template", "AI Reasoning"],
      "retrieval_keywords": ["Anthropic think chain template", "step-by-step
reasoning workflow", "Claude chain-of-thought example"],
      "cross_references": []
    },
    {
      "canonical_title": "3AIWorkflows.json",
      "bd_engine": "Cross-Engine",
      "artifact_type": "Workflow (n8n)",
      "file_name": "3AIWorkflows_Tutorial.json",
      "drive_path": "Prime TS BD Automation/External Templates",
      "integration_status": "Template",
      "purpose": "Bundle of three beginner-friendly AI workflows (from 44-min
tutorial): likely a Q&A chatbot, an image generator, and a simple data processor
as separate flows.",
      "integration_usage":
"Quickstart examples – import provides multiple basic AI workflow examples for
learning. Useful for new team members to see simple implementations (e.g., one
calls OpenAI for a question, another triggers on input to generate an image).
Acts as a mini library of patterns to build upon.",
      "workflow_phase": "Maintenance",
      "prompt_tags": ["Template", "Tutorial"],
      "retrieval_keywords": ["beginner AI workflows JSON",
"tutorial AI workflow examples", "44-min tutorial flows bundle"],
      "cross_references": []
    },
    {
      "canonical_title": "LongFormContent.json",
      "bd_engine": "Cross-Engine",
      "artifact_type": "Workflow (n8n)",
      "file_name": "LongFormContent.json",
      "drive_path": "Prime TS BD Automation/External Templates",
      "integration_status": "Template",
      "purpose": "Workflow for fully automated long-form content creation (blog
posts/articles) by chaining research, outline, drafting, and editing via AI.",
      "integration_usage": "Deploy for content marketing automation. After
import, configure info sources (RSS feeds, search APIs) and use GPT for writing.
The agent likely gathers facts, outlines, then drafts an article and optionally
revises tone/style. Good for generating first drafts of blogs or reports on
chosen topics with minimal human input.",
      "workflow_phase": "Maintenance",
      "prompt_tags": ["Template", "Content"],
      "retrieval_keywords": ["long-form content AI JSON", "blog generator
workflow", "automated article writing template"],
      "cross_references": []
    },
    {
      "canonical_title": "DeepResearchReport.json",
      "bd_engine": "Cross-Engine",
      "artifact_type": "Workflow (n8n)",
      "file_name": "DeepResearchReport.json",
      "drive_path": "Prime TS BD Automation/External Templates",
      "integration_status": "Template",
      "purpose": "Workflow for deep research that compiles info and outputs a
formatted PDF report on a given topic (searches, summarizes, then PDF).",
      "integration_usage": "Use to have an AI agent produce comprehensive
research briefs. Provide a query/topic, the workflow performs multiple searches
(web, internal docs), aggregates findings with structure (headings, bullets),
and uses a PDF generator to output a nicely formatted report. Ideal for quick
turnaround research docs or market intel reports generated by AI.",
      "workflow_phase": "Maintenance",
      "prompt_tags": ["Template", "Research"],
      "retrieval_keywords": ["deep research agent JSON", "AI report generator
workflow", "comprehensive research PDF automation"],
      "cross_references": []
    }
  ]
}
```

(The JSON above mirrors the Master Inventory, with each artifact's metadata in a structured form suitable for ChatGPT agents to use for zero-shot retrieval [48] [224]. Fields like prompt_tags and retrieval_keywords are included to help an agent intelligently locate the right artifact based on a user's query.)

GitHub Repo Matrix (Prime TS BD Automation Code Repositories)

The table below summarizes all relevant GitHub repositories analyzed for the Prime TS BD Automation project. Each row details the repository's purpose, its fit to one or more engines, how it integrates, installation steps, main scripts, status, dependencies, and recommended usage.

Repo Name	**Purpose / Functionality**	**Engine Fit**	**Integration Points** (How to Integrate)	**Install / Setup**
JobSpy	Python library for multi-source job search scraping (LinkedIn, Indeed, Google Jobs, etc.). Acts as an orchestrator returning normalized job listings [225] [226]. Handles proxies and unified schema output.	*Scraper* (Core)	Import as Python library in Scraper engine (via Python node or custom code). Can run broad multi-site job queries on schedule [227].	`pip install jobspy` (plus proxy settings). Configure targets via its API.

Repo Name	Purpose / Functionality	Engine Fit	Integration Points (How to Integrate)	Install / Setup
jobsparser	CLI tool built on JobSpy (wraps jobspy2). Supports scraping LinkedIn, Indeed, Glassdoor via command-line with parameters [229] [230]. Useful for scheduled scraping without coding.	*Scraper* (Core)	Run via N8n Execute Command or cron job. Writes scraped jobs to CSV for ingestion [231]. Essentially a command-line interface to JobSpy.	Python-based CLI. Install via Git (`git clone`) pip if available. Requires jobspy2.
JOB-POSTING-SCRAPER	Scrapy-based scraper for ATS job boards (Greenhouse, Lever, Ashby, etc.) [234] [235]. Contains spiders for major ATS platforms used by primes. Originally used for levergreen.dev aggregator.	*Scraper* (Core)	Run spiders via Python script or Docker. Integrate by invoking spiders from N8n (Execute Command) and consuming output JSON/CSV [236].	Clone repository. Install Python deps (Scrapy etc.). Configure target company board URLs in settings.

Repo Name	Purpose / Functionality	Engine Fit	Integration Points (How to Integrate)	Install / Setup
fpdsScraper	Python scraper for **FPDS.gov** (Federal contract awards) [239]. Fetches contract award data (JSON/CSV outputs) for mapping programs and identifying prime vs sub relationships.	*Program Mapping* (Core)	Run standalone via Python (or call from N8n Exec node). Outputs award data used to tag programs with prime, award amount, etc. [240].	Python script. Install required libs (requests, BeautifulSoup). Needs FPDS endpoint access o HTML parsing.
CrossLinked	Python tool for **employee name enumeration** via search engine dorks [242]. Finds LinkedIn profiles for a company by querying Google/Bing with role keywords. Outputs names & titles (CSV).	*Org Chart* (Core)	Run as Python script (with proxies) externally or via N8n. Accepts company name & keywords, returns CSV of potential employee names/titles [243].	Python script. Install deps (requests or SerpAPI etc.). Likely requires API keys if using search AP

Repo Name	Purpose / Functionality	Engine Fit	Integration Points (How to Integrate)	Install / Setup
apify-linkedin-profile-search-by-name	Node.js/Apify actor that searches LinkedIn profiles by name (with optional company filter) [245]. Returns LinkedIn profile data (name, headline, etc.) and attempts email discovery.	*Org Chart* (Core)	Deploy as Apify actor or run locally via Node. Trigger from N8n via HTTP to Apify API with query params [246]. Use LinkedIn cookie for authenticated scraping.	Use via Apify platform (no local install needed, just config). Or `npm install` and run `index.js` with env vars (cookies).
scrapedin	Node.js library using Puppeteer to scrape **LinkedIn profiles** (given login cookies) [248]. Extracts full profile data (experience, education, etc.). Maintenance stagnant since 2021.	*Org Chart* (Auxiliary)	Use as a Node library or executed via Node in N8n. Needs LinkedIn credentials (cookies) to log in and scrape profiles [249]. Can be integrated as a sub-module for profile enrichment.	`npm install scrapedin`. Use by importing and calling `scrapedin({cookies}).open(profileUrl`

Repo Name	Purpose / Functionality	Engine Fit	Integration Points (How to Integrate)	Install / Setup
linkout-scraper	Node.js Puppeteer tool for **LinkedIn automation** (visiting profiles, sending connection requests/ messages at scale) [251]. Actively maintained through ~2023. Can scrape profile data and perform outreach actions.	*Playbook* (Core) <br/ >(*Scraper* support)	Run via Node (script or container). Integrate by either containerizing it or calling it from N8n's Execute Command to perform LinkedIn actions [252]. Requires LinkedIn auth cookies.	`npm install` or use Docker. Configure LinkedIn credentials (cookies) and any proxies. Run provided script (likely `index.js` or simil
linvo-scraper	Node.js Puppeteer LinkedIn automation tool (similar to Linkout) [254]. Could send connection requests, scrape profiles. Open-source alternative, last updated 2022.	*Playbook* (Auxiliary)	Usage similar to linkout-scraper. Could be swapped in or used alongside Linkout for LinkedIn actions [255]. Integrate via Node script execution.	`npm install linvo-scraper`. Provide LinkedIn credentials. Use in code or CLI if provided.

Repo Name	Purpose / Functionality	Engine Fit	Integration Points (How to Integrate)	Install / Setup
linkedin-message-generator	Node.js app that generates personalized LinkedIn outreach messages using GPT and Google Custom Search [257] [258]. Inputs: prospect's role/company; outputs: a drafted intro message.	*Playbook* (Core)	Host as a service or container. Integrate via HTTP API: send person & context, get back a GPT-crafted message [259]. Could also be invoked via Node in n8n.	`npm install` (likely). Needs OpenAI API key and Google Custom Search API key configured [260]. Run `npm start` if it's a server, or call generator function if library.

Repo Name	Purpose / Functionality	Engine Fit	**Integration Points** (How to Integrate)	Install / Setup
linkedapi-node	Official Node.js SDK for LinkedAPI (third-party LinkedIn automation API). Provides functions to send connection requests, messages, fetch profiles via LinkedAPI.io service.	*Playbook* (Core)	Use as an npm library within Node environment or custom N8n function node. Requires LinkedAPI account & tokens. Instead of direct scraping, call LinkedAPI endpoints (e.g., sendConnectionRequest).	`npm install linkedapi-node`. Obtain API credentials from LinkedAPI.io and configure tokens. Then use library methods in code.

Repo Name	Purpose / Functionality	Engine Fit	**Integration Points** (How to Integrate)	Install / Setup
puppeteer-gpt	Experimental tool allowing GPT to control a headless browser via Puppeteer (natural language web automation) [264] [265]. AI can navigate and manipulate web as instructed.	*Cross-Engine* (Auxiliary)	Run standalone (Node + Puppeteer). Integration via API or by triggering the script from N8n with an instruction payload. Not directly in N8n due to its interactive nature; best run as a service.	Clone repo. `npm install`. Configure OpenA API key. Run `node puppeteer-gpt.js` with prompt that includes instructions.

Repo Name	**Purpose / Functionality**	**Engine Fit**	**Integration Points** (How to Integrate)	**Install / Setup**
AI-Resume-Analyzer-and-LinkedIn-Scraper-using-Generative-AI	Python Jupyter app that uses GPT to evaluate resumes and scrape LinkedIn profiles, outputting a candidate "score" or profile analysis [266] [267]. Could rank candidates or assess fit.	*Scoring* (Auxiliary)	Not directly in pipeline. Run offline or containerize. Could integrate results by importing generated CSV or via an API. Use to provide a "candidate quality" metric into Scoring Engine (if needed for internal hiring or talent mapping).	Python environment or Jupyter. Needs OpenAI API and LinkedIn cookie. Run notebooks or scripts provided.
PrimeTime-BD-Intel	*Monorepo* tying together all engines – contains data schemas, config JSONs, and documentation for the Prime TS BD Intelligence project. Meant as the central knowledge base repo.	*Cross-Engine* (Core)	Used as the main **config & documentation repository**. Could be integrated via Git sync to n8n (or CI pipeline) to pull latest JSON configs, prompts, etc. [268] [269]. Not a runtime code repo, more of a source-of-truth storage.	Clone repo. No build needed if mainly docs/JSC If it contains code (not likely), follow README.

Repo Name	Purpose / Functionality	Engine Fit	Integration Points (How to Integrate)	Install / Setup
primetime-agent-mode	Custom code (possibly Python) implementing the multi-engine **Agent Mode** logic. Orchestrates combined analyses across engines (the "glue" to produce outputs like the Attack Plan) [270] [271].	*Cross-Engine* (Core)	Likely run as a microservice or script that pulls data from all engines (DB, files) and uses LangChain or similar to generate combined analyses. Integrate via N8n HTTP node or CLI call when a consolidated report is needed.	Clone repo. If Python, set up venv, install requirements. Run main script (maybe `agent_mode.py`) with access to all artifact fil

Repo Name	Purpose / Functionality	Engine Fit	Integration Points (How to Integrate)	Install / Setup
mcp-agent	Possibly a Master Control Program (MCP) service for managing AI agents (Node or Python) [272] [273]. Manages agent instances, memory, and orchestrates continuous or long-lived agents.	*Cross-Engine* (Auxiliary)	Could run as a server that N8n or other services call to spawn/query agents. Integrate via its API if available. Not central to pipeline unless pursuing autonomous agents beyond single tasks.	Check repo README. If Node: `npm install` then `npm run` or build Docker. If Python: sim steps with requirements.
hdw-mcp-server	Legacy server component (possibly from a prior project) superseded by new agent/orchestration approach [274] [275]. Perhaps an older attempt at the MCP concept.	*Cross-Engine* (Archived)	Not used in current system. No integration – kept only for reference (may contain code for Airtable sync, cron jobs, etc., that have been replaced by n8n/Agent Mode).	Not applicable (archived). If needed, see READI for historical setup.

<small>**Legend:** Core vs Auxiliary vs Archived indicates whether a repo is central to the solution (core), optional/alternative (auxiliary), or deprecated (archived). Engine Fit shows which engine(s) benefit from the repo. Active/Stagnant reflects maintenance status (last commit, community activity). *Tags* highlight primary focus (e.g. scraping, LinkedIn, contracts, etc.).</small>

Sources: Repo information synthesized from structured repo analyses [276] [277] [278] and the GitHub Repos Analysis document [279] [275]. The matrix ensures we focus on maintained, high-value tools and notes those flagged as stagnant [280] or archived.

GitHub_Repo_Matrix.json

Below is a JSON representation of the GitHub Repo Matrix, including additional metadata to aid retrieval and agent integration (e.g. prompt keywords and an agent_mode_label for easy reference in prompts):

```
{
  "repositories": [
    {
      "repo_name": "JobSpy",
      "purpose": "Multi-source job search scraper (LinkedIn, Indeed, etc.) –
orchestrates searches and returns normalized job postings.",
      "bd_engine": "Scraper",
      "core_or_aux": "core",
      "integration_points": "Used as a Python library within the Scraper engine
(via Python node or script) to run broad job searches. Triggered on schedule in
n8n, outputs jobs in unified schema. [227] ",
      "install_steps": "pip install jobspy (and configure proxies for
LinkedIn).",
      "main_scripts": "Library function `scrape_jobs()` called from custom
code.",
      "tags": ["Python", "Scraping"],
      "status": "active",
      "dependencies": "Python 3, requests/BS4, proxy support modules.",
      "recommended_usage": "Best for broad web job scraping across general
boards; run 2-3x daily to feed open-web jobs [228] .",
      "prompt_keywords": ["JobSpy", "multi-source scraper", "job search
library"],
      "agent_mode_label": "Repo:JobSpy"
    },
    {
      "repo_name": "jobsparser",
      "purpose": "CLI tool wrapping JobSpy for multi-site job scraping via
command-line (LinkedIn, Indeed, Glassdoor).",
      "bd_engine": "Scraper",
      "core_or_aux": "core",
      "integration_points": "Executed via N8n command node or cron job. Outputs
```

```
CSV of scraped jobs for ingestion. [281] ",
      "install_steps": "Clone or pip install; ensure jobspy2 dependency
present.",
      "main_scripts": "Run `jobsparser` command with flags (no separate script
file, it's an installed CLI).",
      "tags": ["Python", "CLI"],
      "status": "active",
      "dependencies": "Python 3, jobspy2 library (fork of JobSpy).",
      "recommended_usage": "Ideal for scheduled scraping with one command.
Combine multiple site scrapes in one cron. Extend by adding site modules for
ClearanceJobs, etc. [282] .",
      "prompt_keywords": ["jobsparser", "CLI scraper", "jobspy CLI"],
      "agent_mode_label": "Repo:jobsparser"
    },
    {
      "repo_name": "JOB-POSTING-SCRAPER",
      "purpose": "Scrapy spiders for competitor ATS job boards (Greenhouse,
Lever, etc.) – deep scrape of prime contractors' career pages.",
      "bd_engine": "Scraper",
      "core_or_aux": "core",
      "integration_points": "Run via Python (Scrapy). Integrate by invoking
spiders from N8n or scheduler; read output JSON/CSV for ingestion. [236] ",
      "install_steps": "Clone; pip install -r requirements.txt (Scrapy).
Configure target company URLs in spiders or settings.",
      "main_scripts": "`run_job_scraper.py` to execute all spiders; individual
spider files per ATS.",
      "tags": ["Python", "ATS"],
      "status": "active",
      "dependencies": "Python 3, Scrapy framework.",
      "recommended_usage": "Use for thorough daily scraping of target
competitors' job boards [283] . Must customize spiders for each company’s careers
page and schedule runs (preferably containerized).",
      "prompt_keywords": ["job-board-scraper", "Scrapy spiders", "ATS scraper"],
      "agent_mode_label": "Repo:JOB-POSTING-SCRAPER"
    },
    {
      "repo_name": "fpdsScraper",
      "purpose": "Scraper for FPDS.gov to retrieve federal contract award data
(for mapping programs & prime contractors).",
      "bd_engine": "Program Mapping",
      "core_or_aux": "core",
      "integration_points":
"Standalone Python scraper. Run via N8n exec node or cron job; outputs JSON/CSV
with contract awards. [240] ",
      "install_steps": "Clone; install required libs (requests, etc.). Possibly
configure search parameters (agency, date range).",
      "main_scripts": "`fpds_scraper.py` (assuming this is the main script
name).",
      "tags": ["Python", "Contracts"],
      "status": "active",
      "dependencies": "Python 3, requests/BeautifulSoup or Selenium for FPDS
(depending on impl).",
      "recommended_usage": "Use on schedule to pull recent contract awards to
enhance Program Mapping with contract size & prime info [241] . Complements
USAspending API integration notes [152] .",
      "prompt_keywords": ["fpdsScraper", "contract data scraper", "FPDS
awards"],
      "agent_mode_label": "Repo:fpdsScraper"
    },
    {
      "repo_name": "CrossLinked",
      "purpose": "Tool for enumerating employees on LinkedIn via Google/Bing
search (finds names/titles by company).",
      "bd_engine": "Org Chart",
      "core_or_aux": "core",
      "integration_points": "Run as Python script externally or in N8n. Input
company & role keywords, outputs CSV of names/titles. [243] ",
      "install_steps":
"Clone; pip install requirements (likely requests + maybe SerpAPI). Set up
proxies or API keys for search.",
      "main_scripts": "`crosslinked.py` (with arguments for company name,
keywords).",
      "tags": ["Python", "OSINT"],
      "status": "active",
      "dependencies": "Python 3, search API or HTML scraping of Google results
(plus proxies to avoid blocks).",
      "recommended_usage":
"Use to generate initial contact lists for Org Charts when internal data is
lacking [284] . After run, filter relevant roles and feed into LinkedIn scraping
(Apify or scrapedin) for details.",
      "prompt_keywords": ["CrossLinked", "LinkedIn name enumerator", "Google
dork employees"],
      "agent_mode_label": "Repo:CrossLinked"
    },
    {
      "repo_name": "apify-linkedin-profile-search-by-name",
      "purpose":
"Apify actor to search LinkedIn profiles by name (optionally by company) and
return profile data.",
      "bd_engine": "Org Chart",
      "core_or_aux": "core",
      "integration_points": "Call via Apify API from N8n (HTTP request) with
person name and optional company; receives JSON profile results. [246] ",
      "install_steps": "No local install if using Apify cloud (just use actor
ID). For local: npm install, set LinkedIn cookie env var.",
      "main_scripts":
```

```
"Runs on Apify platform (no single script exposed, it's an actor).",
      "tags": ["Node.js", "Apify"],
      "status": "active",
      "dependencies": "Node.js (on Apify). Requires valid LinkedIn session
cookie; Apify SDK.",
      "recommended_usage":
"Use after obtaining names (e.g. from CrossLinked) to enrich with current
titles, profile URLs, maybe emails [247] . Reliable and maintained method to get
LinkedIn data with minimal dev effort (just config on Apify).",
      "prompt_keywords": ["Apify LinkedIn search", "profile search actor",
"LinkedIn by name Apify"],
      "agent_mode_label": "Repo:apify-linkedin-profile-search-by-name"
    },
    {
      "repo_name": "scrapedin",
      "purpose": "Node.js Puppeteer library to log in and scrape full LinkedIn
profiles (experience, education, etc.).",
      "bd_engine": "Org Chart",
      "core_or_aux": "auxiliary",
      "integration_points": "Use as Node module in custom script or N8n
Function. Provide LinkedIn creds (cookie) and profile URL to scrape profile
data. [249] ",
      "install_steps":
"npm install scrapedin (but note last update 2021). Might require patching for
current UI.",
      "main_scripts": "No standalone script; usage via
`scrapedin(credentials).openProfile(url)` in code.",
      "tags": ["Node.js", "Puppeteer"],
      "status": "stagnant",
      "dependencies":
"Node.js, Puppeteer (bundled in lib). Last commit 2021, so may not work without
fixes.",
      "recommended_usage":
"Use only if needed and willing to maintain a fork [250] . Provides deep profile
data. Good as backup if Apify or LinkedIn API options fail; must watch for
LinkedIn UI changes.",
      "prompt_keywords": ["scrapedin", "LinkedIn profile scraper lib", "Node
LinkedIn scraper"],
      "agent_mode_label": "Repo:scrapedin"
    },
    {
      "repo_name": "linkout-scraper",
      "purpose": "Node.js (Puppeteer) tool to automate LinkedIn actions (visit
profiles, send connect requests with notes, etc.) at scale.",
      "bd_engine": "Playbook",
      "core_or_aux": "core",
      "integration_points": "Use via Node (script or Docker). Integrate by
scheduling runs (e.g., daily via N8n Exec) to send connection requests or scrape
```

```
profile info. [252] ",
      "install_steps": "npm install or use provided Docker container. Provide
LinkedIn auth (cookie/email-pass) and config (message template, daily limits).",
      "main_scripts": "CLI or script provided by repo (e.g., `npm start` runs a
defined scenario like connect or message).",
      "tags": ["Node.js", "Puppeteer"],
      "status": "active",
      "dependencies": "Node.js, Puppeteer, LinkedIn cookies. Maintained through
late 2023.",
      "recommended_usage":
"Primary tool for **LinkedIn outreach automation** [285] . Use to execute the AI-
generated outreach at scale (sending invites/messages). Throttle to <20 actions/
day/account to avoid bans. Actively maintained, so more robust to LinkedIn
updates.",
      "prompt_keywords": ["linkout-scraper", "LinkedIn automation bot",
"Puppeteer LinkedIn actions"],
      "agent_mode_label": "Repo:linkout-scraper"
    },
    {
      "repo_name": "linvo-scraper",
      "purpose": "Alternative LinkedIn automation tool (Node/Puppeteer) with
overlapping functionality to linkout-scraper.",
      "bd_engine": "Playbook",
      "core_or_aux": "auxiliary",
      "integration_points": "Similar usage as linkout. Could swap in if needed;
integrate via Node script. Supports sending connection requests, etc. [255] ",
      "install_steps": "npm install linvo-scraper. Provide LinkedIn cookies and
config JSON.",
      "main_scripts": "Likely CLI usage (e.g., `npx linvo-scraper ...`) or an
example script to run.",
      "tags": ["Node.js", "Puppeteer"],
      "status": "active",
      "dependencies": "Node.js, Puppeteer. Last commit Nov 2022 (some
maintenance).",
      "recommended_usage": "Good backup if Linkout becomes unavailable or as an
open-source option [256] . Keep updated for LinkedIn UI changes. Use similarly to
connect & message, but watch maintenance gaps.",
      "prompt_keywords": ["linvo-scraper", "LinkedIn automation alt", "open
source LinkedIn bot"],
      "agent_mode_label": "Repo:linvo-scraper"
    },
    {
      "repo_name": "linkedin-message-generator",
      "purpose": "Node.js app that uses GPT (and Google Search) to generate
personalized LinkedIn outreach messages given a prospect's info.",
      "bd_engine": "Playbook",
      "core_or_aux": "core",
      "integration_points":
```

```
"Run as a service (Express API) or script. Integrate via HTTP calls from N8n:
send prospect name/role/company, get back a crafted message [259] .",
      "install_steps":
"npm install. Set OPENAI_API_KEY and Google Custom Search API key in env. `npm
start` to run server.",
      "main_scripts": "If server mode: index.js (Express endpoint). If
standalone: script that reads input and prints message.",
      "tags": ["Node.js", "OpenAI"],
      "status": "active",
      "dependencies": "Node.js, OpenAI API (GPT-4), Google Custom Search API,
possibly Express.",
      "recommended_usage": "Use to empower even junior reps with high-quality,
tailored intro messages at scale [261] . Great for prepping connection notes or
follow-ups using live info (news about prospect). Integrate before the LinkedIn
send step in Playbook engine.",
      "prompt_keywords": ["LinkedIn message generator", "GPT outreach text",
"personalized message AI"],
      "agent_mode_label": "Repo:linkedin-message-generator"
    },
    {
      "repo_name": "linkedapi-node",
      "purpose": "Official SDK for LinkedAPI.io (LinkedIn automation via API) –
safer, throttled LinkedIn actions through a paid service.",
      "bd_engine": "Playbook",
      "core_or_aux": "core",
      "integration_points": "Include as npm library. Call LinkedAPI methods in
Node (e.g. send connection request or fetch profile) instead of direct scraping
 [262] .",
      "install_steps": "npm install linkedapi-node. Obtain API token from
LinkedAPI.io and config in code.",
      "main_scripts": "No script; use library functions like
`client.sendConnectionRequest()` within custom code.",
      "tags": ["Node.js", "API"],
      "status": "active",
      "dependencies": "Node.js; LinkedAPI.io account (external service).",
      "recommended_usage": "Use for **scalable & compliant outreach** if budget
allows – offloads risk to LinkedAPI service [263] . Good for large-scale connection
efforts where maintaining cookies is burdensome. Keep API usage within limits to
control cost.",
      "prompt_keywords": ["linkedapi SDK", "LinkedIn API wrapper", "official
LinkedIn automation API"],
      "agent_mode_label": "Repo:linkedapi-node"
    },
    {
      "repo_name": "puppeteer-gpt",
      "purpose": "Experimental tool allowing GPT to control a browser (headless
Puppeteer) by natural language commands (AI-driven web automation).",
      "bd_engine": "Cross-Engine",
      "core_or_aux": "auxiliary",
      "integration_points": "Run outside n8n (Node process). Potential
integration via API or by triggering with an instruction prompt via N8n HTTP
node. [286] ",
      "install_steps": "Clone; npm install. Set up OpenAI API key. Run `node
puppeteer-gpt.js` (or provided CLI) with an instruction JSON.",
      "main_scripts": "`puppeteer-gpt.js` (listens for or contains the GPT-to-
browser loop).",
      "tags": ["Node.js", "Puppeteer", "GPT"],
      "status": "active (experimental)",
      "dependencies": "Node.js, Puppeteer, OpenAI API (GPT-4 likely).",
      "recommended_usage": "Use carefully for **one-off complex web tasks**
(e.g., instruct AI to log into a site and extract info) [265] . Great power but
needs constraints on instructions. Not in regular pipeline; use in controlled
scenarios requiring custom web navigation by AI.",
      "prompt_keywords": ["puppeteer-gpt", "AI web controller", "GPT browser
automation"],
      "agent_mode_label": "Repo:puppeteer-gpt"
    },
    {
      "repo_name": "AI-Resume-Analyzer-and-LinkedIn-Scraper-using-Generative-
AI",
      "purpose": "Python project (with Jupyter notebooks) that uses GPT to
evaluate resumes and scrape LinkedIn profiles, outputting scored profiles or
analysis.",
      "bd_engine": "Scoring",
      "core_or_aux": "auxiliary",
      "integration_points":
"Run offline (Jupyter) or containerize. Not directly integrated into n8n;
results (scored resumes or extracted skills) can be imported via CSV/API to
inform Scoring engine.",
      "install_steps": "Clone; set up Python env. Install OpenAI, Selenium/
Playwright for LinkedIn, etc. Run Jupyter notebook or script.",
      "main_scripts": "Jupyter Notebook (.ipynb) or scripts inside (e.g.,
`analyze_resume.py`).",
      "tags": ["Python", "OpenAI", "Scraper"],
      "status": "active",
      "dependencies": "Python 3, OpenAI API, LinkedIn scraping method (Selenium
or similar).",
      "recommended_usage": "Supplemental – can add a **“candidate quality”
dimension** to Scoring or help scan competitor talent [267] . Use for internal
analysis (e.g., evaluating incoming resumes vs reqs, or analyzing competitor
employees for recruitment). Not core to BD pipeline but useful for talent-
focused analytics.",
      "prompt_keywords": ["resume analyzer GPT", "LinkedIn scraper AI",
"candidate scoring tool"],
      "agent_mode_label": "Repo:AI-Resume-Analyzer"
    },
    {
      "repo_name": "PrimeTime-BD-Intel",
      "purpose": "Monorepo tying together all BD engines: holds config files
(JSON), schemas, prompts, and documentation for the project. Acts as master
knowledge base repo.",
      "bd_engine": "Cross-Engine",
      "core_or_aux": "core",
      "integration_points": "Used as main config/docs repo. Integrated via Git:
n8n or CI/CD can pull latest JSON configs, prompts, etc., to update automation
flows. [287] ",
      "install_steps": "Clone. No build needed (documentation). Optionally set
up a GitHub Action to deploy config changes.",
      "main_scripts": "No code to run; contains JSON, MD, etc. Possibly scripts
to sync with Airtable or DB if included.",
      "tags": ["Markdown", "JSON", "Docs"],
      "status": "active",
      "dependencies": "None (documentation). Possibly requires Git LFS if large
files.",
      "recommended_usage": "**Single source of truth** for configurations and
knowledge [269] . Keep updated whenever strategy docs or config files change. Use a
webhook or CI to propagate changes to the live system (e.g., refresh vector
index or n8n workflows).",
      "prompt_keywords": ["PrimeTime-BD-Intel repo", "master config repo", "BD
knowledge base repo"],
      "agent_mode_label": "Repo:PrimeTime-BD-Intel"
    },
    {
      "repo_name": "primetime-agent-mode",
      "purpose": "Code for the multi-engine Agent Mode orchestrator – combines
outputs from all engines using LLMs to produce analyses (e.g., BD Attack
Plan).",
      "bd_engine": "Cross-Engine",
      "core_or_aux": "core",
      "integration_points":
"Runs as a separate service or CLI. Invoked from N8n or schedule when a
consolidated multi-engine analysis is needed. Likely reads from engines' output
files or DB, uses LangChain to prompt GPT with all info. [288] [289] ",
      "install_steps": "Clone; install requirements (LangChain, OpenAI, etc.).
Configure file paths or DB connections to engine outputs. Run main orchestrator
script on demand.",
      "main_scripts": "`agent_mode.py` or similar (if Python), orchestrating
cross-engine analysis.",
      "tags": ["Python", "LangChain", "LLM"],
      "status": "active",
      "dependencies": "Python 3, OpenAI API, possibly vector DB or local JSON
files from PrimeTime-BD-Intel repo.",
      "recommended_usage": "Central to **combined insights generation** [271] . Use
it to produce the BD Attack Plan or any analysis that requires data from all
```

```
engines (jobs, programs, org contacts, playbook content). Trigger it
periodically (e.g., weekly strategy refresh) or after major data updates. Ensure
it has access to latest data (via PrimeTime-BD-Intel or DB).",
      "prompt_keywords": ["primetime-agent-mode", "multi-engine agent
orchestrator", "combined analysis agent"],
      "agent_mode_label": "Repo:primetime-agent-mode"
    },
    {
      "repo_name": "mcp-agent",
      "purpose": "Master Control Program for agents – likely manages agent
instances, state, and long-running agent tasks (continuous agents or multi-step
workflows).",
      "bd_engine": "Cross-Engine",
      "core_or_aux": "auxiliary",
      "integration_points": "Potentially run as a server that N8n can call via
API to spawn/query long-lived agents. Could handle things like an agent that
monitors channels or periodically acts without direct triggers. [290] [291] ",
      "install_steps": "Clone; if Node (TypeScript) run `npm install && npm
build && npm start`. If Python, install and run server script. Configure any
required DB for state.",
      "main_scripts": "If Node: a server (maybe Express) handling agent
lifecycle. If Python: similar concept via FastAPI/Flask.",
      "tags": ["Agent Management", "Server"],
      "status": "active",
      "dependencies": "If Node: TypeScript/Node, maybe Redis/DB for state. If
Python: similar with event loop or background scheduler.",
      "recommended_usage": "Not in initial deployment, but consider for
**autonomous agents** or continuous monitoring tasks in future. Could give our
system more autonomous decision-making and memory persistence outside N8n [273] .
Use when ready to experiment with long-running agent workflows (beyond stateless
triggers).",
      "prompt_keywords": ["mcp-agent server", "agent management service",
"continuous agent framework"],
      "agent_mode_label": "Repo:mcp-agent"
    },
    {
      "repo_name": "hdw-mcp-server",
      "purpose":
"Legacy MCP server from an earlier iteration (HDW project). Not used in current
architecture; kept for reference (contains older automation logic like cron
jobs, Airtable sync).",
      "bd_engine": "Cross-Engine",
      "core_or_aux": "archived",
      "integration_points": "None (deprecated). Superseded by updated agent/
orchestration components in current system. [292] [293] ",
      "install_steps": "N/A for production. (Historical: would have had its own
setup, possibly Docker-compose for DB/Airtable connection, but now
irrelevant.)",
      "main_scripts": "Legacy server main (could be index.js or app.py,
depending on stack). Not executed anymore.",
      "tags": ["Legacy", "Server"],
      "status": "archived",
      "dependencies": "Possibly Python (Django/Flask) or Node. Contains old
integration code (Airtable, etc.). Last commit Feb 2023.",
      "recommended_usage": "Do not use in new system – refer only if needed to
see how something was previously implemented [275] . Any useful functionality has
been ported to n8n or primetime-agent-mode. Safe to archive.",
      "prompt_keywords": ["hdw-mcp legacy", "old MCP server", "archived BD
automation server"],
      "agent_mode_label": "Repo:hdw-mcp-server"
    }
  ]
}
```

(In this JSON, each repository object includes fields like repo_name, purpose, bd_engine, core_or_aux (indicating if it's part of the core stack, an auxiliary tool, or archived), integration_points (how to integrate/use it in our system), install_steps, main_scripts, tags, status, dependencies, and recommended_usage. Additional fields prompt_keywords and agent_mode_label are added to help an AI agent quickly identify and retrieve details about the repo given a user query or for referencing this repo in prompts.)

Deliverables Catalog (All Artifacts by Engine & Type)

Finally, a consolidated catalog of all project deliverables (files) sorted by engine and artifact type. This is essentially a flattened list combining the Master Inventory across engines, highlighting the artifact title, engine, type, purpose, how it's used, storage location, link, cross-references, tags, retrieval keywords, and workflow phase:

Artifact Title	**Engine**	**Type**	**Purpose** (Description)	**Integration Usage** (How Used / Status)	**Folde** (Drive
20250804_142008.pdf	Scraper	Document (PDF)	Unidentified reference document (auto-saved content; unclear).	*Not used* – archived for completeness (no active role).	Engin Scrap Refer
AI Automation Methodologies & Blueprint.pdf	Scraper	Document (PDF)	No-code AI automation methods blueprint (Nate Herk/AI Society).	Reference design guide – informed pipeline & agent integration.	Engin Scrap Refer

Artifact Title	**Engine**	**Type**	**Purpose** (Description)	**Integration Usage** (How Used / Status)	**Folde** (Drive
Installing Puppeteer on Windows (Docker Setup).pdf	Scraper	Document (PDF)	Puppeteer + Docker Desktop setup on Windows (dev environment).	Reference during dev – configured headless browser environment.	Engin Scrap Guide
Competitor Job Board URL Master Table	Scraper	Table (Excel)	Master list of competitor job-board URLs & ATS types (Tier1-3 firms).	**Active config** – used by Scraper Engine to iterate daily scrapes.	Engin Scrap Config
Optimized Coverage Hubs (18 cities)	Scraper	List (Text/ CSV)	List of 18 metro hubs prioritized for job searches (geo focus).	Active – scraper loops through these locations for each keyword.	Engin Scrap Config
Apify Actor Input (Fort Meade)	Scraper	JSON Config	Apify actor JSON to scrape Insight Global jobs around Fort Meade.	Active – runs via Apify API (n8n HTTP node) daily with clearance terms.	Engin Scrap Workf
Cleared Job Scraper (Example)	Scraper	Workflow (JSON)	*Draft* single-source scraper (n8n JSON) – scrapes one site to Airtable.	**Superseded** prototype – kept for reference/ testing.	Engin Scrap Workf
Cleared Jobs Workflow (Revised)	Scraper	Workflow (JSON)	**Revised multi-source scraper** (n8n JSON) – scrapes multiple sources, enriches & de-dupes, outputs to Airtable.	**Active** – main scraping workflow (cron-triggered) in production.	Engin Scrap Workf
Scraper Config JSON	Scraper	JSON Config	Puppeteer/Apify config JSON for multi-keyword search (Fort Meade test).	Active – used in Scraper engine's script for test runs & template.	Engin Scrap Config

Artifact Title	**Engine**	**Type**	**Purpose** (Description)	**Integration Usage** (How Used / Status)	**Folde** (Drive
Scraper Engine Config JSON	Scraper	JSON Config	**Master scraping config** – maps competitors to job-board URLs & ATS types.	**Active** – ingested by Scraper workflow for dynamic site routing.	Engin Scrape Confi
Job Postings – Grid View.csv	Scraper	CSV Data	Export of scraped jobs from Airtable (grid view) – for schema alignment.	Reference – used to verify fields match between Airtable and DB.	Engin Scrape Expor
Jobs Records (Full)	Scraper	Table (Excel)	Complete set of job records parsed from placement docs (all jobs).	Active – used for initial data load/testing and cross-engine inputs.	Engin Scrape Expor
Jobs Table (Separate Doc)	Scraper	Table (Excel)	Additional job records from a separate doc (merged into main list).	Duplicate – merged into "Jobs Records (Full)" (kept as backup).	Engin Scrape Expor
Tara Lopez.pdf	Scraper	Document (PDF)	Sample resume/ LinkedIn profile (Tara Lopez) for AI parsing tests.	Test data – used to validate resume parsing (not in production flow).	Engin Scrape Data
Bullhorn Contact Extraction Plan	Program Mapping	Document (Docx)	Plan for extracting Bullhorn contacts (roles by program/ prime).	Active – guides which contacts to pull for Org Chart/Mapping.	Engin Progr Mapp Strate
Top 20 DoD Programs...Spend.pdf	Program Mapping	Document (PDF)	Analysis of top 20 DoD programs by subcontractor staffing spend.	Active – prioritizes target programs (tags jobs, guides focus).	Engin Progr Mapp Strate

Artifact Title	**Engine**	**Type**	**Purpose** (Description)	**Integration Usage** (How Used / Status)	**Folde** (Drive
Past Performance Made Easy Lists.pdf	Program Mapping	Document (PDF)	Quick "cheat sheet" lists of Prime TS past performance highlights.	Active – referenced for credibility points in Mapping & Playbook.	Engin Progr Mapp Credib Docs
Prime NGC Past Performance.pdf	Program Mapping	Document (PDF)	Past performance battle card – Prime TS success examples with NGC.	Active – used to inject NGC-related cred points into outreach.	Engin Progr Mapp Credib Docs
Proposed Architecture...Automation.pdf	Program Mapping	Document (PDF)	*Superseded* initial BD automation architecture blueprint (concept).	Superseded – early design (v1); replaced by final stack doc.	Prime Autom Archit Planni
End-to-End BD Automation Stack...Services.pdf	Program Mapping	Document (PDF)	**Final** integrated architecture & tech stack for BD automation (all engines).	Active – primary reference for system design and integrations.	Prime Autom Archit Planni
Hiring Trends and BD Target Analysis.pdf	Program Mapping	Document (PDF)	Hiring trends & placement analysis "heatmap" (identifies hot areas).	Active – informs Mapping (what programs to emphasize) and Scoring (inputs for demand/ pain).	Engin Progr Mapp Credib Docs
Engine Placements (Clean Headers)	Program Mapping	Table (Excel)	Placement records with initial standardized column headers.	Duplicate – intermediate cleaned data (field names aligned).	Engin Progr Mapp

Artifact Title	**Engine**	**Type**	**Purpose** (Description)	**Integration Usage** (How Used / Status)	**Folde** (Drive
Engine Placements (Remapped Headers)	Program Mapping	Table (Excel)	Placement records with headers renamed to final schema.	Duplicate – intermediate (columns remapped, pre-full clean).	Engin Progr Mapp
Engine Placements (Clean Job Titles)	Program Mapping	Table (Excel)	Placement records after removing recruiter names from titles.	Duplicate – intermediate (cleaned job titles for Org Chart clustering).	Engin Progr Mapp
Engine Placements (Full Cleaned)	Program Mapping	Table (Excel)	**Fully cleaned placement dataset** with final schema (complete set).	**Active** – master cleaned placement data used by Org Chart & Scoring.	Engin Progr Mapp
Engine Placements (Full Parsed Block)	Program Mapping	Table (Excel)	Parsed placement data from raw text (full dataset pre-cleaning).	Duplicate – baseline parsed data (input to cleaning pipeline).	Engin Progr Mapp
NTT Data Apps & BPS Tech Stack.pdf	Program Mapping	Document (PDF)	NTT Data technical stack overview (competitor capabilities doc).	Active – competitive intel, informs Mapping/Org Chart (know competitor strengths).	Engin Progr Mapp Comp Intel
Org Chart Engine – Org Design Outline	Org Chart	Document (Docx)	Methodology blueprint for Org Chart Engine (how to infer orgs).	Active – guides Org Chart development (role clustering logic).	Engin Chart/ & Des

Artifact Title	**Engine**	**Type**	**Purpose** (Description)	**Integration Usage** (How Used / Status)	**Folde** (Drive
GBSD BD Strategy.pdf	Org Chart	Strategy Deck (PDF)	BD strategy for **Sentinel/GBSD** program (org structure, targets, tactics).	Active – used in Org Chart & Playbook for GBSD pursuit context.	Engin Chart/ Progra
GBSD Notebook.pdf	Org Chart	Notebook (PDF)	Reference notes on GBSD program (history, requirements, context).	Active – knowledge base for GBSD; used to enrich outreach content.	Engin Chart/ Progra
Sentinel GBSD Program Key Contacts Call Sheet.pdf	Org Chart	Call Sheet (PDF)	Key contacts list for GBSD program (names, titles, notes).	Active – Org Chart uses to populate contacts; Playbook personalizes outreach per this list.	Engin Chart/ Progra
Ogden Org Overview.pdf	Org Chart	Document (PDF)	Org structure overview for GBSD Ogden site (teams, hierarchy).	Active – reference for Org Chart output validation (GBSD/Ogden).	Engin Chart/ Progra
GBSD Ogden Contacts Job.Function.pdf	Org Chart	Document (PDF)	Contacts list for GBSD Ogden teams (names, titles, functions).	Active – imported into Org Chart contacts DB; used in outreach targeting for Ogden.	Engin Chart/ Progra

Artifact Title	**Engine**	**Type**	**Purpose** (Description)	**Integration Usage** (How Used / Status)	**Folde** (Drive
Clearance Overview.docx	Org Chart	Guidance Doc (Word)	Overview of clearance levels & timelines (Public Trust vs Secret/TS, etc.).	Active – reference for Org Chart & Playbook to add clearance context in comms.	Engin Chart/ Refere
Docx Parsing Attempt (Placements)	Org Chart	Script (Python)	One-off Python script to parse a bad placements .docx (data salvage attempt).	Archived – used once during data prep; not in pipeline (informational).	Engin Chart/ Script
Engine_Placements_Updated.csv	Org Chart	CSV Data	Merged placement data after first update batch (initial + new records).	Duplicate – intermediate merge file (replaced by Updated_Full).	Engin Chart/
Engine_Placements_Updated_Full.csv	Org Chart	CSV Data	Merged placement data after second update (full set pre-cleaning).	Duplicate – checkpoint before cleaning entire set (replaced by final).	Engin Chart/
Engine_Placements_Cleaned.csv	Org Chart	CSV Data	Trimmed placement dataset (unnecessary columns removed).	Duplicate – intermediate cleaned data (for streamlined schema).	Engin Chart/
Engine_Placements_Final.csv	Org Chart	CSV Data	Near-final placement dataset after schema alignment (pre-“All”).	Duplicate – penultimate version for validation (replaced by All).	Engin Chart/
Engine_Placements_All.csv	Org Chart	CSV Data	**Final consolidated placement dataset** (all records, standardized schema).	**Active** – master placement data used by Org Chart & Scoring.	Engin Chart/

Artifact Title	**Engine**	**Type**	**Purpose** (Description)	**Integration Usage** (How Used / Status)	**Folde** (Drive
Cleared_Cable_Tech_Sourcing_Packet.pdf	Org Chart	Document (PDF)	Example sourcing packet for a cleared tech role (Cable Tech).	Active – reference for sourcing/ outreach style (Org Chart & Playbook).	Engin Chart/ Refere
Michael Staff LinkedIn Activity (Export).pdf	Org Chart	Document (PDF)	Export of LinkedIn posts/activity of Michael Staff (target contact).	Active – used by Playbook Engine to personalize outreach to that contact.	Engin Chart/ Progra
Sales Prospecting Playbook.pdf	Playbook	Document (PDF)	**Master Sales Playbook** – cold-call scripts, email templates, rebuttals, etc., with program-specific enhancements.	**Active** – core content for Playbook Engine; AI summarizes and adapts it per program.	Engin Playbo Maste Playbo
Competitive Capture & Insertion Strategy...pdf	Playbook	Document (PDF)	Playbook of tactics for inserting Prime TS under competitors (sub-to-prime strategies).	Active – used to inform targeted approaches in Playbook & Mapping (competitor-specific tactics).	Engin Playbo Strate Guide
Prime_TS_BD_Attack_Plan.pdf	Playbook	Document (PDF)	**AI-Generated BD Attack Plan** – consolidated action plan (30-day, narrative + table) produced by Agent.	Draft (v1) – used to coordinate BD team actions; output of Agent Mode combining all intel.	Engin Playbo Agent

Artifact Title	**Engine**	**Type**	**Purpose** (Description)	**Integration Usage** (How Used / Status)	**Folde** (Drive
Agent Mode Prompt – Prime TS BD Attack Plan	Cross-Engine (Playbook)	Prompt (JSON)	Master JSON prompt instructing AI to generate the BD Attack Plan from all docs.	Active – run in Agent Mode to produce the Attack Plan; ensures key areas are covered [308].	Prime Autom Agent Promp
Master Prompt Template	Cross-Engine (Playbook)	Prompt (Text)	Standardized prompt template for agent chats (structured summary + delta format).	Active – used in every agent session to enforce consistent output structure and do diff analysis [309] [144].	Prime Autom Agent Promp
PTS_BD_Master_Trackers.xlsx	Cross-Engine (Playbook)	Spreadsheet (Excel)	Master BD tracker (sheet1: Target Programs Intel, sheet2: Active Pursuits).	Active – living tracker (manual+auto updates); referenced in maintenance for scheduling and follow-ups [145].	Prime Autom Tracke Sheets
Q3–Q4 2025 DoD BD Events Calendar	Cross-Engine (Playbook)	Calendar (Excel)	Calendar of major defense/IC events in late 2025 with BD notes.	Active – referenced to align outreach timing and event attendance (maintenance phase) [148].	Prime Autom Tracke Sheets
Scoring Engine – Weighted Model Draft	Scoring	Document (Docx)	Draft weights & formula for opportunity scoring model (demand/ pain etc.).	Active (design) – blueprint for Scoring Engine implementation (to be coded) [149].	Engine Scorin Design Specs

Artifact Title	**Engine**	**Type**	**Purpose** (Description)	**Integration Usage** (How Used / Status)	**Folde** (Drive
API Integration Notes for Scoring (USAspending & FPDS)	Scoring	Document (PDF)	Notes on integrating contract data APIs (USAspending/FPDS) into scoring.	Active (reference) – used to plan data feeds for scoring factors (e.g. contract value) [152].	Engin Scorin Desig Specs
Strategic Ranking of DLA JETS 2.0...pdf (Top Opportunities)	Scoring	Report (PDF)	Human-created ranking of BD opportunities (most to least strategic).	Active (reference) – used to calibrate/ validate automated scoring results [312].	Engin Scorin Refere Repor
AI Automation Society N8n Templates Guide.pdf	Cross-Engine	Document (PDF)	Guide mapping AI Automation Society n8n templates (community workflows) by category.	Active (reference) – library of patterns for advanced workflow designs (not executed directly) [156].	Extern Refere Autom Societ
NewsletterAgentTeam.json	Cross-Engine	Workflow (n8n JSON)	Template: AI-driven newsletter team (multiple sub-agents create a daily newsletter).	Template (external) – importable for multi-agent content generation (not in core pipeline).	Extern Templ Societ
UltimateMediaAgent.json	Cross-Engine	Workflow (n8n JSON)	Template: “Ultimate Media Agent” for multi-platform content creation (social posts, videos).	Template – reference for complex media automation (outside core BD usage).	Extern Templ Societ

Artifact Title	**Engine**	**Type**	**Purpose** (Description)	**Integration Usage** (How Used / Status)	**Folde** (Drive
9Socials.json	Cross-Engine	Workflow (n8n JSON)	Template: post content to 9 social platforms simultaneously.	Template – shows multi-platform posting pattern (ext. reference).	Extern Templ Societ
ErrorHandling.json	Cross-Engine	Workflow (n8n JSON)	Template: robust global error handler for n8n (unlimited retries, alerts).	Template – used as example for building error handling into flows.	Extern Templ Societ
ElevenLabsVoice.json	Cross-Engine	Workflow (n8n JSON)	Template: integrate ElevenLabs TTS to give AI agents a voice (audio output).	Template – reference for adding voice output to agents (ext.).	Extern Templ Societ
AgentSwarm.json	Cross-Engine	Workflow (n8n JSON)	Template: manager-agent spawns/ coordinats multiple sub-agents in parallel.	Template – used as guide for multi-agent orchestration patterns.	Extern Templ Societ
Metadata.json	Cross-Engine	Workflow (n8n JSON)	Template: metadata extraction & enrichment workflow.	Template – provides pattern for auto-tagging content with AI metadata.	Extern Templ Societ
FirstRAGAgent.json	Cross-Engine	Workflow (n8n JSON)	Template: basic Retrieval-Augmented Generation agent workflow (search + answer).	Template – basis for building QA agents that use external data.	Extern Templ Societ

Artifact Title	**Engine**	**Type**	**Purpose** (Description)	**Integration Usage** (How Used / Status)	**Folde** (Drive
WebhookSecurity.json	Cross-Engine	Workflow (n8n JSON)	Template: secure an n8n webhook (token/signature verification before processing).	Template – example for implementing webhook security in flows.	Extern Templ Societ
ZepMemory.json	Cross-Engine	Workflow (n8n JSON)	Template: integrate Zep long-term memory API for persistent agent memory.	Template – reference for giving agents recall across sessions.	Extern Templ Societ
Parallelization.json	Cross-Engine	Workflow (n8n JSON)	Template: run tasks in parallel in n8n (branching for concurrent execution).	Template – demonstrates optimizing workflows with parallel branches.	Extern Templ Societ
ResumeScreeningSystem.json	Cross-Engine	Workflow (n8n JSON)	Template: AI-driven resume screening pipeline (parse resumes, rank candidates).	Template – could be adapted for internal recruiting automation (not core BD).	Extern Templ Societ
RerankingRAG.json	Cross-Engine	Workflow (n8n JSON)	Template: advanced RAG agent (generates multiple answers, then re-ranks best).	Template – shows method to improve answer quality via re-ranking.	Extern Templ Societ
YouTubeStrategist.json	Cross-Engine	Workflow (n8n JSON)	Template: AI agent for YouTube content strategy (trends research, idea generation).	Template – example for automating content planning (outside BD scope).	Extern Templ Societ

Artifact Title	**Engine**	**Type**	**Purpose** (Description)	**Integration Usage** (How Used / Status)	**Folde** (Drive
GlassFruitASMR.json	Cross-Engine	Workflow (n8n JSON)	Template: novelty workflow generating ASMR content via AI (creative example).	Template – included for creative inspiration (not enterprise use).	Extern Templ Societ
DeveloperAgent.json	Cross-Engine	Workflow (n8n JSON)	Template: AI software dev assistant agent (generates code, answers dev Qs).	Template – could automate code suggestions or issue responses (internal use).	Extern Templ Societ
VideoAnalysis.json	Cross-Engine	Workflow (n8n JSON)	Template: automates video analysis (transcribe & summarize videos via AI).	Template – useful for content indexing/ monitoring (not core BD).	Extern Templ Societ
BrowserAgent.json	Cross-Engine	Workflow (n8n JSON)	Template: web-browsing agent (AI navigates websites, screenshots, scrapes data).	Template – blueprint for giving agents browsing capabilities.	Extern Templ Societ
FirecrawlSearchScrape.json	Cross-Engine	Workflow (n8n JSON)	Template: “Firecrawl” agent – does web searches and scrapes results (with screenshots).	Template – demonstrates internal search+scrape without proxies.	Extern Templ Societ
FirecrawlExtract.json	Cross-Engine	Workflow (n8n JSON)	Template: Firecrawl companion – extracts structured data from specific URLs/pages.	Template – shows how to parse details from a list of URLs after search.	Extern Templ Societ

Artifact Title	**Engine**	**Type**	**Purpose** (Description)	**Integration Usage** (How Used / Status)	**Folde** (Drive
25n8nHacks.json	Cross-Engine	Workflow (n8n JSON)	Template: showcases 25 useful n8n hacks/hidden features in one workflow (tutorial).	Template – educational reference for advanced n8n techniques.	Extern Templ Societ
ShortsMachine.json	Cross-Engine	Workflow (n8n JSON)	Template: 24/7 "viral shorts" generator – continuously creates & posts short videos.	Template – shows looping content generation; not directly BD-related.	Extern Templ Societ
APIsforAgents.json	Cross-Engine	Workflow (n8n JSON)	Template: examples of AI agents calling external APIs (weather, stocks, etc.).	Template – demonstrates tool use within agent workflows.	Extern Templ Societ
TrackAgentActions.json	Cross-Engine	Workflow (n8n JSON)	Template: agent usage logger – tracks tokens, API calls, time per agent run.	Template – use to monitor agent activity and detect anomalies.	Extern Templ Societ
DynamicBrain.json	Cross-Engine	Workflow (n8n JSON)	Template: agent with self-building knowledge base (stores Q&A to a vector DB for memory).	Template – approach for agents to learn and reuse info over time.	Extern Templ Societ
ProductVideography.json	Cross-Engine	Workflow (n8n JSON)	Template: automates product promo video creation from images/descriptions via AI.	Template – content marketing automation example (not core BD).	Extern Templ Societ

Artifact Title	**Engine**	**Type**	**Purpose** (Description)	**Integration Usage** (How Used / Status)	**Folde** (Drive
AIMarketingTeam.json	Cross-Engine	Workflow (n8n JSON)	Template: “AI Marketing Team” – agent handles content, scheduling, design in one workflow.	Template – multi-step marketing automation showcase.	Extern Templ Societ
LinkedInPostGraphic.json	Cross-Engine	Workflow (n8n JSON)	Template: agent creates a LinkedIn post + an AI-generated graphic image.	Template – shows combining text generation and image generation.	Extern Templ Societ
ErrorWorkflow.json	Cross-Engine	Workflow (n8n JSON)	Template: earlier basic error handler (sends alerts on failures; predecessor to unlimited version).	Template – simple error notification workflow (for reference).	Extern Templ Societ
ThinkTool.json	Cross-Engine	Workflow (n8n JSON)	Template: uses Anthropic “Think” chain-of-thought prompting for better AI reasoning.	Template – improves agent logic by splitting plan vs execution steps.	Extern Templ Societ
3AIWorkflows.json	Cross-Engine	Workflow (n8n JSON)	Template bundle: 3 beginner AI workflows (chatbot Q&A, image generator, simple data flow).	Template – tutorial examples for new users learning AI + n8n.	Extern Templ Societ
LongFormContent.json	Cross-Engine	Workflow (n8n JSON)	Template: fully automated long-form content creation (blog/article via research->draft).	Template – shows multi-step content generation pipeline.	Extern Templ Societ

Artifact Title	Engine	Type	**Purpose** (Description)	**Integration Usage** (How Used / Status)	**Folde** (Drive
DeepResearchReport.json	Cross-Engine	Workflow (n8n JSON)	Template: agent conducts deep research and outputs a formatted PDF report.	Template – for automating comprehensive research reports with AI.	Extern Templ Societ

Note: "Drive Hyperlink" entries marked as *(Internal Archive)* or generic indicate items not meant for external sharing (e.g., internal reference or draft files). Google Drive links are provided for primary deliverables and reference documents where applicable. All artifacts, including intermediate and draft versions, are listed to maintain completeness [47] [224]. Tags and retrieval keywords are included to facilitate quick search and identification of relevant files in a ChatGPT context or via Drive search.

Google Drive Build Blueprint

Folder Structure & Naming Conventions:

To organize the Prime TS BD Automation deliverables, a structured Google Drive hierarchy is recommended:

- **Main Project Folder:** `Prime TS BD Automation (BD IntelliRepo)` – Top-level folder containing all project artifacts.
- **Engine 1 – Scraper** – Contains subfolders for Scraper Engine artifacts:
  - *Configurations* – e.g., JSON configs, target lists (Competitor URLs JSON, Coverage Hubs CSV).
  - *Workflows & Scripts* – e.g., n8n workflow JSONs (Cleared Jobs workflows) and any code scripts.
  - *Data Exports* – e.g., Jobs CSV exports or full job record tables.
  - *References* – e.g., setup guides, external PDFs related to scraping (Puppeteer setup, AI automation blueprint).
  - *Test Data* (if needed) – e.g., sample resumes or dev-only files.
- **Engine 2 – Program Mapping** – Subfolders:
  - *Strategy Docs* – e.g., Top 20 Programs PDF, Bullhorn Extraction Plan.
  - *Credibility Docs* – e.g., Past Performance lists, Battle Cards, Hiring Trends PDF.
  - *Competitive Intel* – e.g., competitor-specific references (NTT Tech Stack PDF, etc.).
  - *Data* – placement data tables (Full Parsed, Cleaned, etc.). Could further subdivide versions or mark superseded files with "[Archive]" or use a naming version (e.g., "v1, v2" as done in JSON).
- **Engine 3 – Org Chart** – Subfolders:
  - *Program-Specific Intel* – e.g., GBSD Strategy deck, GBSD Notebook, Org Overviews, Key Contacts sheets.
  - *Strategy & Design* – e.g., Org Design Outline doc.
  - *Data (Placements)* – the placement datasets (Updated, Final, All). Within, prefix file names with "Placements_" for clarity. The **final master** placement file should be clearly labeled (e.g., "Engine_Placements_All (Final Master).csv") to avoid confusion.

    - *References* – e.g., Clearance Overview doc, sourcing packet PDF, any general reference not tied to one program.
    - *Data Scripts* – e.g., the one-off Python parsing script (with file name prefix "Archive_" or "ZZ_" to denote it's not active).
- **Engine 4 – Playbook** – Subfolders:
    - *Master Playbook* – the finalized Sales Prospecting Playbook PDF (and Word, if exists).
    - *Strategy Guides* – e.g., Competitive Insertion Strategy PDF, any outreach tactic guides.
    - *Agent Outputs* – e.g., the AI-generated BD Attack Plan PDF. (Optionally store the prompt JSON here too, or in a Cross-Engine prompts folder.)
    - *Templates* – if any standard outreach templates (call scripts, email templates) exist as separate docs, store here.
- **Engine 5 – Scoring** – Subfolders:
    - *Design & Specs* – e.g., Scoring Model Draft doc, API Integration Notes PDF.
    - *Reference Reports* – e.g., the Strategic Opportunity Ranking PDF (human rankings).
    - *(No dynamic data yet, but if score outputs or dashboards are exported, a folder can be added.)*
- **Cross-Engine & Infrastructure** – Subfolders:
    - *Architecture & Planning* – e.g., Proposed Architecture PDF (archived), Final End-to-End Stack PDF, Master State docs if any.
    - *Agent Prompts* – e.g., JSON prompt files (Attack Plan master prompt, Master Prompt Template).
    - *Trackers & Sheets* – e.g., the Master Trackers Excel, Events Calendar Excel.
    - *External References* – e.g., AI Automation Society N8n Templates Guide PDF, any competitive strategy docs not engine-specific (like general AI community research).
    - *External Templates* – JSON workflows from AI Automation Society or others. These can be stored in a subfolder (perhaps organized further by category if many). Use a naming convention like "Template – [Name].json" for clarity (e.g., "Template – NewsletterAgentTeam.json"). Since these are reference only, consider prefixing with "Template -" or storing under a clearly labeled folder to prevent confusion with our own workflows.

File Naming Conventions:

- **Descriptive Titles:** Use descriptive file names that start with the artifact's purpose, followed by specifics. E.g., "Top_20_Programs_List.pdf" instead of a generic export name. For multi-word names, capitalizing Each Word or using underscores for spaces improves readability in Drive URLs.
- **Engine Prefix (optional):** Within each engine's folder, filenames already imply context, so prefixing with engine name is not strictly needed. However, for cross-engine references or when files are viewed in "Recent" with no folder context, a prefix can help. For example, "Scraper – Competitor_URLs.xlsx" or "Mapping – Top20 Programs.pdf". In practice, grouping by folder suffices, so we rely on folders for context.
- **Versioning:** For iterative artifacts, use simple version tags in filenames (v1, v2, Final). The inventory JSON shows version fields; apply those to file names where relevant. E.g., "Cleared_Jobs_Workflow_v2.json" (as already named in analysis) or "Engine_Placements_All_v3.csv" for the final placements (as was indicated).
- **Draft/Archive Labels:** Append "(Draft)" or "(Archive)" in the file name for clarity on superseded documents. E.g., "Proposed_Architecture_(Archive).pdf" or prefix older files with "ZZ_" to push to bottom. Alternatively, create an "Archive" subfolder within each engine for deprecated files to reduce clutter.

- **CamelCase vs Underscores:** Maintain consistency – many provided file names use spaces or underscores. Prefer **underscores or dashes** for multi-word file names to avoid issues. For example, "Master_Prompt_Template.txt" instead of "Master Prompt Template.txt". This matches JSON references.
- **Avoid Special Characters:** Use only alphanumeric, spaces, underscores, or dashes. (The provided names sometimes have parentheses, which are fine, e.g., "(Full Parsed Block)" but could be replaced by descriptive text.)
- **Distinctiveness:** Ensure each file name is unique across the entire Drive. E.g., differentiate "Placements_Final.csv" vs "Placements_All.csv" (perhaps keep just one – use "All" as final). The blueprint would choose one naming scheme for final placements – likely "Engine_Placements_All_Final.csv" as the authoritative dataset.

Example Folder Layout:

```
Prime TS BD Automation (BD IntelliRepo)/
├─ Engine 1 – Scraper/
│   ├─ Configurations/
│   │     ├─ Competitor_Job_Board_URLs.xlsx
│   │     ├─ Optimized_Coverage_Hubs.csv
│   │     └─ ScraperEngine_Config.json
│   ├─ Workflows & Scripts/
│   │     ├─ Cleared_Jobs_Workflow_v2.json
│   │     ├─ Cleared_Jobs_Workflow_v1.json (Archive Prototype)
│   │     └─ FortMeade_Apify_Actor_Input.json
│   ├─ Data Exports/
│   │     ├─ Jobs_Records_Full.xlsx
│   │     ├─ Jobs_Table_Additional.xlsx (Archive)
│   │     └─ Job_Postings_Grid_View.csv
│   └─ References/
│         ├─ AI_Automation_Methodologies_Blueprint.pdf
│         ├─ Puppeteer_Docker_Setup.pdf
│         └─ Tara_Lopez_SampleResume.pdf
├─ Engine 2 – Program Mapping/
│   ├─ Strategy Docs/
│   │     ├─ Top20_Programs_Priority_List.pdf
│   │     └─ Bullhorn_Contact_Extraction_Plan.docx
│   ├─ Credibility Docs/
│   │     ├─ Past_Performance_Lists.pdf
│   │     ├─ Prime_NGC_Past_Performance.pdf
│   │     └─ Hiring_Trends_Analysis.pdf
│   ├─ Competitive Intel/
│   │     └─ NTT_Data_Tech_Stack.pdf
│   └─ Data/
│         ├─ Engine_Placements_Full_Cleaned.xlsx (Final)
│         ├─ Engine_Placements_Full_Parsed.xlsx (Archive)
│         ├─ Engine_Placements_Clean_Headers.xlsx (Archive)
│         ├─ Engine_Placements_Remapped_Headers.xlsx (Archive)
│         ├─ Engine_Placements_Clean_Titles.xlsx (Archive)
│         ├─ Engine_Placements_All_Final.csv (Final consolidated, maybe same as
Full_Cleaned if not splitting)
│         └─ ... (any other data files)
├─ Engine 3 – Org Chart/
│    ├─ Program Intel/
│    │    ├─ GBSD_BD_Strategy.pdf
│    │    ├─ GBSD_Reference_Notes.pdf
│    │    ├─ GBSD_Key_Contacts_Call_Sheet.pdf
│    │    ├─ Ogden_Org_Overview.pdf
│    │    └─ Ogden_Contacts_List.pdf
│    ├─ Strategy & Design/
│    │    └─ Org_Chart_Design_Outline.docx
│    ├─ Data (Placements)/
│    │    ├─ Engine_Placements_All_Final.csv (or link to Mapping's final if
shared)
│    │    ├─ Engine_Placements_Updated_Full.csv (Archive)
│    │    ├─ Engine_Placements_Updated.csv (Archive)
│    │    ├─ Engine_Placements_Cleaned.csv (Archive)
│    │    └─ Engine_Placements_Final_Check.csv (Archive)
│    ├─ References/
│    │    ├─ Clearance_Levels_Overview.docx
│    │    └─ Cleared_CableTech_Sourcing_Packet.pdf
│    └─ Data Scripts/
│         └─ parse_placements_docx.py (Archive utility)
├─ Engine 4 – Playbook/
│    ├─ Master Playbook/
│    │    └─ Prime_TS_Sales_Prospecting_Playbook.pdf
│    ├─ Strategy Guides/
│    │    └─ Competitive_Capture_Insertion_Strategies.pdf
│    ├─ Agent Outputs/
│    │    └─ Prime_TS_BD_Attack_Plan_v1.pdf
│    └─ Templates/ (if needed for outreach templates or prompt files specific to
playbook)
├─ Engine 5 – Scoring/
│    ├─ Design & Specs/
│    │    ├─ Scoring_Model_Weights_Draft.docx
│    │    └─ Scoring_Data_Integration_Notes.pdf
│    └─ Reference Reports/
│         └─ Top_Opportunities_Human_Ranking.pdf
├─ ⚙ Architecture & Planning/
│    ├─ End-to-End_BD_Automation_Stack.pdf
│    ├─ Proposed_Architecture_Concept.pdf (Archived)
│    ├─ Master_State_Consolidated.docx (if exists)
│    └─ Google_Cloud_APIs_for_BD.pdf (if relevant to stack)
```

```
├─   Agent Prompts/
│    ├─ Master_Prompt_Template.txt
│    └─ Attack_Plan_Agent_Prompt.json
├─ 📊 Trackers & Sheets/
│    ├─ PTS_BD_Master_Trackers.xlsx
│    └─ 2025_Q3-Q4_BD_Events_Calendar.xlsx
├─   External References/
│    └─ AI_Automation_Society_N8n_Templates_Guide.pdf
└─   External Templates/ (AI Automation Society Templates)/
     ├─ Template – NewsletterAgentTeam.json
     ├─ Template – UltimateMediaAgent.json
     ├─ Template – 9Socials.json
     ├─ Template – UnlimitedErrorHandling.json
     ├─ Template – ElevenLabsVoice.json
     ├─ Template – AgentSwarm.json
     ├─ Template – Metadata.json
     ├─ Template – FirstRAGAgent.json
     ├─ Template – WebhookSecurity.json
     ├─ Template – ZepMemory.json
     ├─ Template – Parallelization.json
     ├─ Template – ResumeScreeningSystem.json
     ├─ Template – RerankingRAG.json
     ├─ Template – YouTubeStrategist.json
     ├─ Template – GlassFruitASMR.json
     ├─ Template – DeveloperAgent.json
     ├─ Template – VideoAnalysis.json
     ├─ Template – BrowserAgent.json
     ├─ Template – FirecrawlSearchScrape.json
     ├─ Template – FirecrawlExtract.json
     ├─ Template – 25n8nHacks.json
     ├─ Template – ShortsMachine.json
     ├─ Template – APIsforAgents.json
     ├─ Template – TrackAgentActions.json
     ├─ Template – DynamicBrain.json
     ├─ Template – ProductVideography.json
     ├─ Template – AIMarketingTeam.json
     ├─ Template – LinkedInPostGraphic.json
     ├─ Template – BasicErrorWorkflow.json
     ├─ Template – ThinkTool.json
     ├─ Template – 3AIWorkflows.json
     ├─ Template – LongFormContent.json
     └─ Template – DeepResearchReport.json
```

Placement & Labeling Rules:

- **Engine-specific content goes into its engine folder.** Cross-functional items (multi-engine or infrastructure related) go in the cross-engine sections (Architecture, Prompts, Trackers).
- **Canonical Naming:** Use clear, human-friendly names as above (capitalize words, no ambiguous abbreviations). The names in the inventory table are essentially the canonical titles to use in Drive (they are descriptive yet succinct). For agent ease, these match what an agent would reference (e.g., “Sales Prospecting Playbook.pdf” is clear and unique).
- **Folder Labels:** Include engine number and name in folder title (for clarity and ordering). Use emoji or icons (optional) as visual aids (as shown: or relevant emoji like 🗺 for mapping, but that’s aesthetic).
- **File Links in Inventory:** Ensure each file’s “Drive Hyperlink” either points to the actual Drive file or, if kept internal, note it’s an internal/archived item not meant to be accessed externally.

By following this blueprint, each artifact is uniquely identified and easy to locate. The canonical file names align with those used in the Master Inventory (JSON and Excel), ensuring that a ChatGPT-based agent or any team member can refer to a file by name and quickly find it in the Google Drive structure. This structure also facilitates **zero-shot retrieval** by an AI agent: the **engine-based grouping** and **descriptive naming** mean an agent can infer, for example, that "Top20_Programs_Priority_List.pdf" would be in Engine 2’s folder and contain what it needs [77]. All files, even drafts, are retained (with clear status labels) so that historical decisions and data transformations are traceable, but without cluttering the primary view (thanks to archive subfolders or naming). The blueprint above thus balances completeness with clarity, supporting both human users and AI assistants in navigating the Prime TS BD Automation knowledge base.

---

1 3 4 5 6 7 8 9 13 14 15 16 17 18 19 21 22 23 25 29 30 31 32 33 34 38 40 41 42
Step 1 – Decisions Made.pdf
https://drive.google.com/file/d/1q_nE2RLH9kowGRmnuTum41YiKC2e6ucD

2 45 46 Master State Google Doc
https://docs.google.com/document/d/1Sy3GEQfnhxa43kT-P26XIzXTXobIlPm9avaeNA61PIo

10 11 20 24 26 28 39 43 44 Step 3_ Consolidated Current State by Engine.pdf
https://drive.google.com/file/d/1TZfjpzk8DsYoz0N1fYBBSJJwW255QT8m

12 27 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 227 229 231 234 236 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330 Master Inventory (JSON).pdf
https://drive.google.com/file/d/1gb9dMJAps05CP1xVUbMNu5on-pRpxxsM

35 36 37 “Current State by Module/Engine”
https://docs.google.com/document/d/1_enrMOihNnURbMlcxwN5rnirpx0F2mjHoVnAXnscxwg

47 Prime TS Knowledge Base Inventory.pdf
https://drive.google.com/file/d/1qSYLRGRxkqZ8ydhVMcMOCIJuX_JwTNbA

226 228 230 232 233 235 237 238 260 282 283 Analysis of GitHub Repositories for BD Automation Engines.pdf
https://drive.google.com/file/d/1F3ZXDWh2SlwxK7MiG0yDMiowUl0L9nSH