That's the most challenging part of the project—integrating the complex logic. Building on Phase 1, these instructions for an AI platform like Jules focus on implementing the advanced mechanics, the PvP environment, and the final logic gates.

Phase 2: Advanced Mechanics and Logic Implementation

GOAL: Implement multi-threading, advanced command execution, and the dynamic PvP environment with victory conditions.

Module 4: Advanced Scripting and Object Interaction

GOAL: Enable complex, high-level code execution and resource management.

Component	Instruction	Output Requirement
Parallel Processing	Implement the thread spawn command. Use the underlying execution environment (Python's multiprocessing or JS web workers) to simulate parallel script execution, respecting the cpu_max limit of the player's VC.	A cmd_thread_spawn(module_name) function that runs a script module asynchronously.
Resource Monitoring	Integrate a resource check into the run_script loop. Scripts that exceed the ram_max property of the VC_State must be terminated with an "Out of Memory" error.	Logic to dynamically calculate and check a script's simulated RAM usage against the VC_State before and during execution.
Exploit Object	Define the Exploit_Object class. Include the critical deploy(target_ip, target_service) method. The method's success must be determined by a **Boolean check** against the target_service.version property.	The Exploit_Object class and the logic for the deploy method returning TRUE only if version numbers match.
Knowledge Synthesis Logic	Refine the **K-Map** update logic. A command's state should change from DISCOVERED to **INTEGRATED** only after the player has provided a minimum of \mathbf{3} unique fragments and successfully compiled them into a single, valid .kns file.	A function synthesize_knowledge(cmd_name) that validates the three Boolean prerequisite states before updating the K-Map.

Module 5: PvP and Dynamic Network Environment

GOAL: Create the temporary, strategic multiplayer battlegrounds and victory conditions.

Component	Instruction	Output Requirement
Dynamic LAN Generation	Create a generate_pvp_lan(team_list) function. This function must assign random, non-internet-routable IP addresses to all participating player VCs and set up a temporary network topology.	A dictionary representing the temporary LAN, mapping player IDs to their assigned IP and making them discoverable by the scan command.
Mission File Logic	Implement the logic to scatter and track mission objective files. Randomly place \mathbf{3} unique files (file_A, file_B, file_C) on the filesystems of the players of the opposing team.	A server-side object that tracks the current location and ownership of the \mathbf{3} objective files.
Acquisition/Recovery	Implement the copy command for remote file transfer. A successful copy of an objective file from an opposing VC is an **Acquisition**. If a player copy's their *own* file back, it's a **Recovery** (both count toward the victory condition).	Logic within the cmd_copy function to update the central objective file tracker.
Victory Condition	Implement the core PvP victory check. The game session ends when the first team's **Boolean Goal** evaluates to **TRUE**: \mathbf{\text{WINNER} = (\text{TEAM_ACQUIRED_FILES} \ge 3)}.	A persistent check loop that monitors the file tracker and triggers the end-of-round sequence.

Module 6: VIP Quality-of-Life Features

GOAL: Finalize the features that provide value to the subscription model without enabling pay-to-win.

Component	Instruction	Output Requirement
Automated Defense	Implement the VIP command vc.auto_defend on. When this flag is set and an attack is detected while the player is offline, the system must automatically execute the player's highest-ranked defense script once.	A server-side check that triggers a script execution based on the vc.auto_defend flag and an incoming connect or exploit attempt.
Trace Mode (VIP)	Enhance the run command for VIP users: run $module --trace.	Conditional logic inside the run_script loop that only

Component	Instruction	Output Requirement
	This should print the value of all active variables after **every** line of script execution, providing a powerful, time-saving debugging tool.	executes the detailed variable logging if the is_vip flag is **TRUE**.