Architectural Engineering of Autonomous Strategic Directives: A Technical Design for Endgame Screeps Systems

The transition from a localized, script-based colony to a high-throughput, multi-room empire within the Screeps execution environment represents a fundamental shift from simple heuristic programming to professional-grade systems engineering. In the strictly deterministic, tick-based lifecycle of this distributed simulation, the primary constraints for an automated agent are not merely in-game resources like energy or minerals, but the rigid limits of Central Processing Unit (CPU) cycles, Heap persistence, and the computational overhead of memory serialization.[1] To achieve true imperial sovereignty, a bot must move beyond the legacy role-based logic of first-generation scripts and adopt a kernel-driven architecture that treats the empire as a distributed operating system.

The endgame architecture is predicated on the implementation of specialized "Directives"—functional wrappers for strategic objectives that anchor geographic intents and manage complex lifecycles.[1] While the initial stages of development focus on the HarvestDirective for resource extraction, the progression toward Global Control Level (GCL) dominance necessitates a more sophisticated suite of directives: the ColonizeDirective for automated expansion, the SiegeDirective for high-security bunker penetration, the GuardDirective for choke point defense, and the PowerBankDirective for highway resource acquisition.[1] These directives are not isolated scripts but integrated modules that interact with a global logistics broker, a science overlord, and a power creep registry to maximize the return on investment for every CPU millisecond.[1]

The Kernel-Driven Operating System: Foundations of Scalability

The physical architecture of the Screeps server dictates the design of modern bot frameworks. User code executes within a Node.js virtual machine isolate, managed by isolated-vm, where the progression of time is measured in discrete game ticks. The standard execution model involves a calculation stage followed by a command processing phase, necessitating an architecture that can accurately predict the state of the world multiple ticks into the future.

The Serialization Wall and Heap Persistence

A critical vulnerability in early-stage development is the assumption that the environment is

stateless between ticks. In reality, the V8 engine's Heap persists across ticks unless the server explicitly tears down the isolate. Legacy architectures rely heavily on the Memory object, which is a JSON string parsed every tick via JSON.parse(). As a colony expands, a monolithic reliance on Memory inflates the baseline CPU floor; a 2MB memory file can consume upwards of 15ms merely to enter the loop, leaving insufficient time for advanced pathfinding or military coordination.

To achieve tick-perfect performance, the architecture must transition to a Heap-first model. The system operates entirely within the global cache, storing CostMatrices, pathing arrays, and class instances in the Heap, serializing only critical state transitions to Memory.

State Storage Feature	**Legacy Approach (Procedural)**	**Modern Industrial Protocol (Kernel-Driven)**	**Performance Impact**
Primary Data Store	Memory object parsed every tick.	Global Heap Cache accessed instantly.	10x CPU reduction in parsing.
Data Lifecycle	Re-instantiated sequentially every tick.	Persistent classes updated via .refresh().	Minimizes V8 garbage collection.
Capacity Constraints	2MB hard cap on Memory string.	2MB Memory + ~256MB Heap + 100MB Segments.	Enables complex historical analysis.
Spatial Sensing	Direct Room.find() calls per creep.	Centralized Registry with $O(1$ lookups.	Eliminates redundant spatial searches.

Inversion of Control and the Overlord Hierarchy

The kernel treats the bot as a managed environment where work is encapsulated into "Processes"—classes with unique Process IDs (PIDs) and integer priorities. This hierarchy employs an Inversion of Control (IoC) pattern: the Overmind root object provides analytics; the Overseer scans for developmental milestones; the Directive defines a strategic goal; and the Overlord manages the spawning and tasking of Zerg units.[1] Unlike legacy creeps that "decide" their own actions, Zerg are stripped of independent decision-making, strictly executing commands provided by their governing Overlord.[1]

ColonizeDirective: The Lifecycle of Imperial Birth

Expansion is the primary mechanism for increasing GCL and CPU limits. The ColonizeDirective manages the automated transition of a target room from a neutral state to a functional Room Control Level (RCL) 3 outpost.[1] This directive follows a linear, state-driven lifecycle designed to mitigate the risk of "Colony Crash" through a surrogate relationship with an RCL 8 parent room.[1]

Phase I: Scouting and Environmental Assessment

Triggered by the placement of a flag, the directive enters a SCOUTING state if vision is not available.[1] The parent colony spawns a fast, low-cost scout creep (usually [MOVE]) to gain vision. During this phase, the Architect module executes a Distance Transform algorithm to map the room's topography.[1] This algorithm assigns an integer representing the Manhattan distance to the nearest wall for every walkable tile.[1] By filtering for values ≥ 3 or ≥ 5, the architect identifies a sufficiently large footprint for core base infrastructure, such as the Bunker Stamp.[1]

Phase II: The Legal Claim and Resource Doubling

Once the room is validated, the directive transitions to the CLAIMING state, spawning a "Lawyer" creep equipped with a CLAIM part.[1] Successful claiming immediately changes the room's status to "Owned," which is critical for local resource yield. A standard energy source in a neutral room regenerates 1,500 energy every 300 ticks (5 energy per tick). Upon ownership, this yield doubles to 3,000 energy per cycle, or 10 energy per tick.[1]

Phase III: The Pioneer Influx and Metabolic Stabilization

At RCL 1, a room lacks the energy capacity (limited to 300) to build its first Spawn, which costs 15,000 energy.[1] The directive enters the BOOTSTRAPPING state, where the parent room spawns "Pioneer" creeps.[1] These massive 50-part workers carry starting energy and possess high WORK counts to accelerate building.[1]

Pioneer Type	Body Composition	Cost	Capabilities
All-Terrain	25 WORK, 25 MOVE	3,750	125 build pts/tick; moves 1/tick on plains.[1]
Heavy Builder	30 WORK, 10 CARRY, 10 MOVE	4,000	150 build pts/tick; requires roads for

			speed.[1]
Remote Settler	15 WORK, 10 CARRY, 25 MOVE	2,750	75 build pts/tick; optimized for swamp navigation.[1]

Preventing the Death Spiral: The Surplus Calculus

Expansion must be authorized by a mathematical verification of the parent's Net Energy Surplus ($E_{surplus}$) [1]:

$$E_{surplus} = \sum(I_{local} + I_{remote}) - (X_{spawn} + X_{repair} + X_{upgrade,min})$$

Where I is income, X_{spawn} is replacement cost per lifespan, X_{repair} is upkeep for containers and ramparts, and $X_{upgrade,min}$ is the 15 units per tick required to prevent RCL 8 controller downgrade.[1] Expansion is only sustainable if $E_{surplus} > C_{expansion\_rate}$, and the parent's storage remains above a buffer threshold (e.g., 100,000 energy).[1]

SiegeDirective: Engineering the Coordinated Assault

The tactical landscape of RCL 8 warfare is dominated by concentrated defensive firepower. A defender possessing six towers and reinforced ramparts can deliver up to 3,600 damage per tick within range 5.[1] The SiegeDirective counters this through the deployment of "Quads"—synchronized 2×2 squads that act as a single high-performance entity.[1]

Geometric Dilation and Pathfinding Integrity

The fundamental challenge of a quad is that the native PathFinder calculates routes for single-tile entities. To ensure the quad's cohesive movement, the architect must manipulate the CostMatrix using geometric dilation.[1] For every unwalkable tile at (x, y) , the matrix must mark four tiles as impassable (255) in the quad's specific matrix: (x, y) , $(x-1, y)$, $(x, y-1)$, and $(x-1, y-1)$.[1] This ensures that if the anchor (top-left creep) is on a walkable tile, all members are guaranteed to be on walkable terrain.

Kinetic Synchronization and Collective Fatigue

A quad's speed is dictated by the member with the highest fatigue. If one member accummulates fatigue—perhaps by entering a swamp—the squad will split unless synchronization is enforced.[1] The optimal solution is the creep.pull() mechanism, where the

anchor creep pulls its followers.[1] Fatigue generated by pulled creeps is transferred to the puller, allowing for specialized "Engine" creeps with high MOVE counts to compensate for "Heavy" units laden with WORK or HEAL parts.[1]

The collective fatigue reduction is calculated as:

$$R_{quad} = \sum_{i=1}^{4} MOVE_i \times 2$$

For 1-tile-per-tick velocity, R_{quad} must be $\geq$ the total fatigue generated by all non-move parts and terrain factors.[1]

Predictive Triage and Structural Resilience

Surviving tower fire requires proactive rather than reactive healing. Intents in Screeps process in batches; healing intents generally precede damage, but the total health is calculated at the end of the tick.[1] Top-tier sieges use "Pre-healing," where healers call heal() on members every tick regardless of current health, anticipating the defender's tower AI.[1]

The triage algorithm identifies quad members whose projected health deficit ($Hits + PredictedHeal - PredictedDamage$) is below their hitsMax. Priority is given to units likely to be "alpha-striked".[1] Boosted TOUGH parts are essential; a T3 boosted TOUGH part reduces damage by 70%, effectively tripling the hit points of the creep's armor layers.[1]

Part Type	Unboosted Effect	T3 Boosted Effect (XGHO2,XLHO2,XUH2O)	Efficiency Multiplier
TOUGH	100 Hits	0.3x Damage Taken	3.33x Effective HP.[1]
HEAL (Range 1)	12 HP	48 HP	4.0x.[1]
HEAL (Range 3)	4 HP	16 HP	4.0x.[1]
WORK (Dismantle)	50 Hits	200 Hits	4.0x.[1]

Formation Transitions: The Snake formation

RCL 8 bunkers often include 1-wide "honeycomb" corridors to break quad formations.[1] The

SiegeDirective must detect pathing failures in the dilated matrix and trigger a transition to a 1x4 "Snake" or "Worm" mode.[1] This involves Lead-Follower serialization, where a FIFO queue of the last four positions occupied by the leader is used to buffer movement.[1] The squad returns to 2×2 formation only once it reaches a tile with sufficient clearance, resuming dilated pathfinding.[1]

GuardDirective: Choke Point Defense and Algorithmic Fortification

The GuardDirective focuses on the creation of a mathematical perimeter to protect room infrastructure with minimal energy expenditure. Unlike mobile defenders, stationary guards are designed to camp specific ramparts, utilizing the defensive bonuses of the room layout.[1]

Min-Cut Wall Placement

The "Min-Cut" algorithm is the industry standard for determining optimal rampart locations. It seeks the minimum number of rampart tiles needed to completely isolate the base core from the exits.[7] This is modeled as an s-t cut problem in a flow network, typically solved using the Edmonds-Karp or Dinic algorithm.[7]

In this model, each room tile is split into two nodes (source and sink) connected by an edge with a capacity equal to the tile's weight. All other connections between adjacent tiles have infinite capacity.[7] The resulting "cut" through the nodes with the lowest capacities identifies the ideal rampart coordinates.[7] Best practices dictate a 3-tile buffer between ramparts and internal structures to prevent RangedMassAttack damage from penetrating the base.[1]

The 1-HP Rampart Shield Strategy

A unique mechanical quirk of the Screeps engine is that a rampart with even 1 hit point will absorb any single attack intent, regardless of the damage volume.[11] A GuardDirective can coordinate with a dedicated worker to continuously "repair-build" a rampart construction site on a tile where the rampart has just been destroyed.[11] This technique can reduce incoming damage by up to 33% at a cost of only 1 energy per three ticks, effectively neutralizing boosted attackers that would otherwise destroy reinforced structures in seconds.[11]

Quantitative Defensive Thresholds and Body Ratios

High-tier defense is a game of energy efficiency. The Threat Analyst process calculates the "Survivability Threshold" of hostile units before firing towers.[1] If a hostile creep's boosted heal-per-tick (HPT) exceeds the combined damage-per-tick (DPT) of all towers at their current ranges, the towers enter a "Hold Fire" state.[1] This prevents attackers from "draining" the

defender's energy reserves.[1]

Stationary defenders benefit from the lack of MOVE requirements. A guard placed on a rampart before a siege begins can utilize the full 50-part limit for combat actions.[13]

Defender Body Pattern	Composition	Strategic Intent
Heavy Melee Guard	40 ATTACK, 10 HEAL	Max damage "back-power" against melee attackers.[5]
Mass Attack Guard	30 RANGED_ATTACK, 20 HEAL	Area-of-effect damage to punish squads.[5]
Paladin	10 TOUGH, 20 ATTACK, 20 HEAL	Tanking frontline unit; requires T3 boosts.[1]

PowerBankDirective: Highway Resource Acquisition

The PowerBankDirective automates the complex process of cracking Power Banks on highway rooms to obtain the power resources required for Global Power Level (GPL) advancement.[2] These structures contain between 500 and 10,000 power and possess 2,000,000 hit points.[2]

Cracking Duos and Reflect Damage Math

Power Banks reflect 50% of incoming damage back to the attacker.[2] Because a single creep cannot perform a melee attack and a melee heal on the same tick, players utilize "Duo" teams of separate attack and heal creeps.[14]

To crack a 2M hit structure efficiently, the duo must maintain constant DPS. An attacker with 20 ATTACK parts deals 600 damage per tick and receives 300 reflect damage.[5] To survive this, the companion healer must possess enough parts to restore 300 HP per tick. Since one HEAL part restores 12 HP at range 1, the duo requires:

$$N_{heal} = \frac{Damage_{reflect}}{12} = \frac{300}{12} = 25 \text{ parts}$$

The optimal duo configuration for a cracking operation is:

- **Attacker**: 20 ATTACK, 20 MOVE, 10 TOUGH (Costs 2,100 energy).
- **Healer**: 25 HEAL, 25 MOVE (Costs 7,500 energy).

Power Hauler Part-Count Balancing

Retrieving the payload requires precise logistics. Power has a weight density of 1 unit per CARRY part.[15] The hauler fleet must be spawned based on the observed power amount in the bank. For a bank containing 5,000 power, the system needs 100 CARRY parts.[16] Using part-count balancing, the PowerBankDirective calculates hauler requirements based on the round-trip distance (D) from the nearest Terminal:

$$N_{haulers} = \frac{Power_{amount}}{50 \times (1 - \frac{2D}{1500})}$$

The hauler fleet is dispatched to arrive at the highway room precisely as the hits of the Power Bank reach zero, ensuring the power is collected before other players can "snipe" the ground drops.[14]

Highway Pathing and Multi-Room Logistics

Highway pathing utilizes the Path Analyst to identify safe routes across sector boundaries.[1] Quads and cracking duos are given top priority in the movement engine, utilizing "Traffic Shoving" to push civilian creeps out of the way.[1] This ensures a constant velocity of one tile per tick, which is critical for maintaining the tight arrival windows required for power harvesting.

The Science Overlord: Mastering Chemical Sovereignty

Mastery of the endgame directives is impossible without the mastery of the laboratory complex. The Science Overlord is a process manager responsible for the entire chemical lifecycle of the room, from extractor mining to T3 boost distribution.[1]

Recursive Stoichiometry and DAG Reaction Logic

High-tier chemistry involves multi-tiered reaction trees. Producing Catalyzed Ghodium Acid (XGH_2O) requires a Directed Acyclic Graph (DAG) of precursor compounds.[1] The stoichiometry is generally a 1:1 ratio, but reaction cooldowns scale from 5 ticks for base minerals to 80 ticks for T3 products.[1]

The Science Overlord utilizes a recursive formulation function that decomposes a target product into its immediate reagents using the REACTIONS global object.[1] The system checks the current Terminal and Storage inventory to prevent redundant production; if 500 units of Ghodium Acid already exist, the system skips precursor steps and proceeds to the final catalyzed reaction.[1]

Reagent Reservation and Military Synchronization

Modern bots use a "Request" protocol for boosting.[1] When a military unit is spawned for a SiegeDirective, its logic registers a request with the room's Science Overlord.[1] The Overlord reserves the necessary minerals and energy in the designated labs, pivoting the room's economy from routine production to high-priority combat support in a single tick.[1]

The Scientist Creep FSM

The Scientist is a specialized logistics agent driven by a Finite State Machine (FSM) to ensure deterministic behavior [1]:

1. **Renew State**: Triggered when TTL falls below a safety threshold to prevent dropping expensive minerals.[1]
2. **Supply Reagents**: Withdraws ingredients from the Terminal and deposits them into input labs.[1]
3. **Product Collection**: Returns finished minerals to the Terminal once they reach a threshold (e.g., 100 units).[1]
4. **Cleanup (Flushing)**: The most critical state. If an order changes or a "Lab Jam" occurs, the Scientist purges the labs of unauthorized resources.[1]

Global Logistics: The Gale-Shapley Broker

A 20-room empire requires a shift from local storage logic to a global "Quota and Delta" architecture.[1] The Terminal Overlord treats the collective storage of all rooms as a single, distributed inventory.[1]

Stable Matching and the Effective Store

The logistics economy is algorithmically modeled as a bipartite matching problem between "Providers" (containers, miners) and "Requesters" (spawns, towers, upgraders).[1] The industry standard for resolution is the Gale-Shapley algorithm, which finds a stable matching where no hauler and no request mutually prefer another assignment.[1]

To prevent "Energy Racing," the broker implements a predictive "Effective Store" (S_{eff}) ledger [1]:

$$S_{eff} = CurrentStore + IncomingReservations - OutgoingReservations$$

When a hauler is matched to a requester, the broker immediately updates S_{eff} . Subsequent matching iterations observe that the structure's needs are theoretically met, even if the hauler is still 10 ticks away from execution.[1]

Terminal Logistics and Quota Management

Each room's memory defines target quantities for resources based on its role (e.g., Military Hub, Industrial Refinery).[1] The system calculates the "Delta" (Δ) and matching Suppliers ($\Delta > 0$) with Consumers ($\Delta < 0$) using a greedy algorithm prioritizing shorter linear distances to minimize transaction energy fees.[1]

Resource Category	Quota Priority	Delivery Urgency
Energy	Critical	Immediate (Tier 0).[1]
Base Minerals	High	Sustained (Tier 1).[1]
Commodities	Medium	Just-in-Time (Tier 2).[1]
Market Surpluses	Low	Opportunistic (Tier 3).[1]

Power Creep Integration: The Catalyst of Intensive Growth

In the absolute endgame, the bottleneck shifts from land acquisition to CPU efficiency and per-room economic intensification. Power Creeps, specifically the Operator class, serve as the technical catalyst for this evolution.[1]

Automation of OPERATE_EXTENSION and Burst Spawning

OPERATE_EXTENSION is an instant-fill mechanic that pulls energy from a target container or storage to fill all extensions in a room.[2] At level 5, this power refills 100% of capacity.[2] This power is synchronized with the SpawnManager: when a massive creep begins spawning, the Power Creep positions itself near the Storage and triggers the power as soon as the first creep finishes, instantly refilling for the next unit in the queue.[1] This coordination eliminates the need for dozens of small courier creeps, reducing CPU and clearing pathfinding congestion.[1]

ROI and the Intensification Ratio

The relationship between power and energy efficiency is modeled by the intensification ratio (I):[1]

$$I = \frac{RoomOutput_{with,PC}}{RoomOutput_{standard}}$$

At high levels of OPERATE_SPAWN (80% reduction) and OPERATE_LAB, I can exceed 3.0, meaning one powered room produces as much value as three standard rooms while consuming significantly less CPU.[1]

Lifecycle Management and Renewal Protocols

The 5,000-tick lifespan of a Power Creep is a significant hurdle, as death triggers a mandatory 8-hour lockout penalty.[1] The kernel implements a proactive "Safe Renewal Threshold" (T_{safe}) based on the distance to the nearest PowerSpawn [1]:

$$T_{safe} = Path(PC, PowerSpawn) + T_{buffer}$$

When TTL falls below T_{safe} , the FSM transitions to the RENEW state, abandoning all tasks to reach a renewal point.[1] In the "Emergency" TTL phase (< 200 ticks), the PC uses force-pathing, ignoring traffic and blocking to ensure survival.[1]

Synthesis of the Autonomous Empire

The successful implementation of Colonize, Siege, Guard, and PowerBank directives within a kernel-driven architecture marks the pinnacle of Screeps AI development. By treating the bot as an operating system, the architect ensures that high-level strategic goals—such as the 4-phase expansion lifecycle or the geometric dilation required for quads—are executed with deterministic precision and absolute CPU efficiency.[1]

Through the mastery of mathematical models—from the Gale-Shapley matching in logistics to the Min-Cut Edmonds-Karp algorithm in defense—an autonomous agent transcends the limitations of human oversight.[1] The integration of the Science Overlord provides the chemical hegemony needed to sustain sieges, while Power Creeps act as force multipliers that intensify the economic output of every room.[1] In the persistent, competitive landscape of the world map, those who treat their codebase not as a collection of game scripts but as an industrial-grade engineering system will ultimately dictate the boundaries of territory and the flow of the global market.[1] The final goal is not just growth, but intensification: a singular, distributed intelligence capable of projecting power across any sector while maintaining perfect metabolic stability within its core colonies.

Works cited

1. Screeps Science Overlord Design.pdf
2. Power | Screeps Documentation, accessed February 19, 2026, https://docs.screeps.com/power.html
3. My task-based screeps implementation. - GitHub, accessed February 19, 2026, https://github.com/sscholl/screeps

4. Overhealing vs. strictly healing past damage: fixing inconsistencies based off of intent order | Screeps Forum, accessed February 19, 2026, https://screeps.com/forum/topic/319/overhealing-vs-strictly-healing-past-damage-fixing-inconsistencies-based-off-of-intent-order
5. Combat - Screeps Wiki, accessed February 19, 2026, https://wiki.screepspl.us/Combat/
6. Defending your room | Screeps Documentation, accessed February 19, 2026, https://docs.screeps.com/defense.html
7. Finding the minimum weighted rampart locations for the game Screeps, with the altered Edmonds–Karp algorithm. - GitHub Gist, accessed February 19, 2026, https://gist.github.com/clarkok/25b3e6e2c7cde42f9678d05db498fbee
8. Josef37/screeps-min-cut-wall: Minimize the amount of walls necessary by graph theory!, accessed February 19, 2026, https://github.com/Josef37/screeps-min-cut-wall
9. Defensive Structures - Screeps Wiki, accessed February 19, 2026, https://wiki.screepspl.us/Defensive_Structures/
10. Find minimum s-t cut in a flow network - GeeksforGeeks, accessed February 19, 2026, https://www.geeksforgeeks.org/dsa/minimum-cut-in-a-directed-graph/
11. Rampart with 1 hit point absorbs any single attack | Screeps Forum, accessed February 19, 2026, https://screeps.com/forum/topic/192/rampart-with-1-hit-point-absorbs-any-single-attack
12. Offensive Strategy : r/screeps - Reddit, accessed February 19, 2026, https://www.reddit.com/r/screeps/comments/7d6w2t/offensive_strategy/
13. Attacking & healing don't use energy - why? | Screeps Forum, accessed February 19, 2026, https://screeps.com/forum/topic/601/attacking-healing-don-t-use-energy-why
14. Power - Screeps Wiki, accessed February 19, 2026, https://wiki.screepspl.us/Power/
15. Questions on roles | Screeps Forum, accessed February 19, 2026, https://screeps.com/forum/topic/913/questions-on-roles
16. Power Creeps update | Screeps Forum, accessed February 19, 2026, https://screeps.com/forum/post/10564
17. Solving the minimum cut problem for undirected graphs - Hacker News, accessed February 19, 2026, https://news.ycombinator.com/item?id=40064027