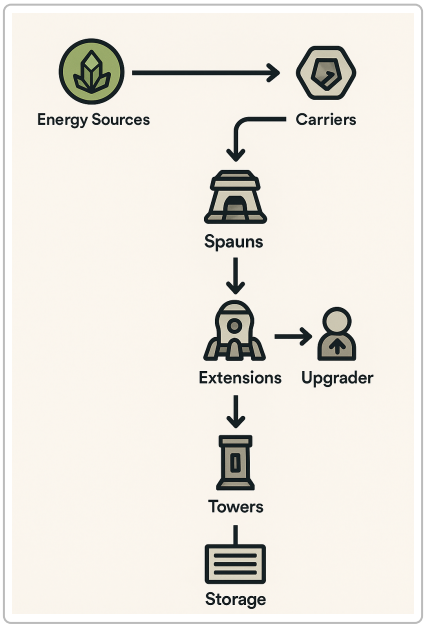
ChatGPT

Energy Collection and Distribution Guide for Screeps

Introduction

Energy is the lifeblood of a Screeps colony. It fuels nearly every action in the game: **spawning creeps, building structures, powering towers and upgrading the room controller** [1]. Energy is harvested from **sources** or acquired through other means, moved by creeps with `CARRY` parts, stored in various structures and consumed by workers and buildings. Efficient energy collection and distribution determines how quickly a room grows and how well it can defend itself.



The diagram above summarizes the ideal energy pipeline: miners harvest from sources into a container, carriers bring energy to spawns and extensions, then supply towers, and finally deposit surplus in storage. The sections below explore every component of this pipeline, provide a recommended priority order for distribution, and identify anti-patterns to avoid.

Energy Sources and Collection

Natural sources

- **Sources** – glowing spots that generate 3 000 energy every 300 ticks. A creep adjacent to a source with a `WORK` part can harvest 2 energy per tick and the energy is automatically stored in its `CARRY` parts. Any harvested energy that does not fit in the creep's `CARRY` parts is dropped on the ground [2].
- **Remote sources** – sources located in other rooms that your creeps can harvest. Remote mining increases income but requires extra creeps and infrastructure.

- **Ruins and tombstones** – destroyed structures and dead creeps often contain leftover energy which decays over time. A scavenger role can withdraw from ruins or pick up dropped energy before it disappears.

Harvesting strategies

1. **Generic workers (early game)** – At RCL1 you cannot build extensions or containers. Use simple `[WORK, CARRY, MOVE]` creeps to harvest and immediately deliver energy to the spawn. The Screeping tutorial notes that a base creep needs at least one `WORK`, one `MOVE` and one `CARRY` part to be useful [3].
2. **Static miners + haulers** – Once you can build containers (RCL2), place a container on each source. Build a "miner" creep with 5 `WORK` parts and minimal `MOVE` / `CARRY`. The miner stands on the container and harvests continuously, automatically depositing energy into the container. A separate "hauler" creep with mostly `CARRY` / `MOVE` withdraws from the container and transports the energy.
3. **Roads and paths** – Build roads between sources, storage and frequent destinations. Roads reduce fatigue and speed up logistics.
4. **Links and power creeps** – From RCL5 you can build links near sources and near storage/spawns. Links teleport energy between rooms at the cost of 3 % of the transmitted energy. Use them for long-distance transfers where haulers would be inefficient.
5. **Recycling and market** – Unused creeps can be recycled by a spawn to recover some energy [4], and energy can be purchased from the market at RCL6+ [5] when your room needs a boost.

Collecting from non-source energy

- **Dropped energy** – Decays quickly (at `ceil(amount/1000)` per tick [6]). A dedicated scavenger creep should pick it up and deposit it into storage or containers.
- **Tombstones & ruins** – Remnants from dead creeps and destroyed structures can hold significant energy. Scavengers should withdraw from them and deposit energy into storage.
- **Hostile structures** – In some cases you can dismantle hostile containers or extract energy from enemy structures after a fight.

Structures for Storage and Distribution

Structure	Purpose & capacity
Container	Holds 2 000 units. Can be built at any RCL. Ideal for placing next to sources and the controller. Containers are walkable and automatically collect dropped energy if the container has capacity [2].
Storage	Available at RCL4. Stores up to 1 000 000 resources [7]. Use it as central storage to stockpile energy and other resources.
Extension	Increases the energy capacity available for spawning. The spawn plus its extensions must contain all the energy needed when spawning begins [8].
Spawn	Creates creeps. Holds 300 energy by default; spawns cannot be bigger than 300 energy unless extensions are filled [8].
Tower	Defensive structure that attacks, heals or repairs at a cost of **10 energy per action** [9]. Towers are critical during attacks and must be kept fueled.

Structure	Purpose & capacity
Terminal	Available at RCL6. Sends resources to other rooms or trades on the market at an energy cost.
Lab / Power spawn / Nuker	High-level structures that consume energy for reactions, processing power and launching nukes.
Controller container	A container placed adjacent to the controller to supply upgraders. No other role should use or deposit energy here [10].

Priority Order for Energy Distribution

When a hauler has energy to deposit, it must decide where to deliver it. The **priority order** depends on the room's state (peaceful vs. under attack) and on your infrastructure. The following recommendations synthesize best practices from community guides and experience.

Default (peace time) priority

1. **Spawns and extensions:** Without energy in these structures you cannot spawn new creeps. Extensions effectively raise the spawn's available energy capacity; they must be full at the start of the spawn [8]. Always deliver energy to spawns and extensions first until they are fully charged.
2. **Towers (below threshold):** Towers consume 10 energy per attack/heal/repair action [9]. In peace time, keep towers at roughly **half** capacity to conserve energy but ensure they can respond immediately to small threats. Only fill towers completely when hostiles are nearby.
3. **Labs, power spawn and factory:** Mid- to late-game structures like labs and power spawns consume energy. If you are running reactions or processing power, feed these structures next.
4. **Storage and terminals:** Deposit excess energy into your main storage and terminal. Storage acts as a buffer to supply future creep production or to sell on the market. Avoid overfilling mining containers or the controller container.
5. **Controller container:** This container is reserved for upgraders. **Do not deposit** energy here unless your upgraders cannot keep it full. The Screeping guide warns that the upgrader container should never become a dump for generic energy [10].
6. **Miscellaneous sinks:** Factories, nukers and other special structures can be filled once the above priorities are satisfied.

Defensive (under attack) priority

When enemy creeps enter your room, priorities shift. Energy is precious in battle; deposit energy in the order recommended by the Screeping guide:

1. **Towers:** During combat, towers become the highest priority. They should be kept near full so they can continuously attack, heal and repair. The Screeping article notes that when there are enemies, getting energy to towers is "pretty dang important" [11].
2. **Spawns and extensions:** Keep spawns and extensions charged so you can replace losses quickly. In battle, spawns may die; having energy to respawn defenders is crucial [12].
3. **Central storage (storage and generic containers):** Top up your main storage to maintain reserves. If towers and spawns are full, deposit here [12].
4. **Controller container:** Only deposit here if everything else is full. Upgrading the controller can wait until after the fight [12].

These priorities can be implemented in code by maintaining a sorted list of potential targets and iterating until a valid structure with free capacity is found.

Best Practices for Energy Logistics

1. **Separate roles:** Avoid large "do-everything" creeps. Use specialized **miners**, **haulers**, **builders**, **upgraders**, and **scavengers**. Mixing harvesting and building in one creep wastes travel time and CPU. The Screeps forum example shows that combining hauler and filler logic leads to loops where the creep withdraws from and deposits back into the same container [13].
2. **Reserve containers:** Place one container next to each source and one next to the controller. Only miners should fill the source containers; only upgraders should draw from the controller container [10]. Other creeps should not deposit or withdraw from these containers.
3. **Cache target lists:** Searching for energy targets each tick is expensive. Cache the IDs of spawns, extensions, towers, storage and the upgrader container in `Memory` and reuse them. Recompute the list only when construction changes. The Screeping article shows how to cache and reuse energy targets to reduce CPU [14].
4. **Use roads and avoid wasted movement:** Build roads along frequent paths to reduce fatigue. Use `moveTo` with `range:1` to minimize pathfinding calls. Place towers and spawns so that carriers can fill multiple structures from a central position [15].
5. **Links and chain transfers:** At higher RCLs, install **links** to transport energy between distant points. Use them to send energy from remote mining containers to storage or from storage to upgrade sites. For long corridors, a chain of haulers transferring energy between themselves (a "bucket chain") can reduce decay and travel costs [16].
6. **Monitor supply and demand:** Track energy input (harvesting) and output (spawning, building, upgrading, repairs). Spawn additional miners when input falls below ~80 % of maximum capacity and spawn additional haulers or upgraders when output is low [17]. Always ensure there are enough carriers to keep the pipeline flowing.
7. **Adapt to room level:** At low RCLs, energy throughput is small and simple logic works. As your room approaches RCL7-8, you will need multiple carriers, remote mining, links and more complicated spawn queues to maintain spawn uptime [18].

Anti-Patterns and Test Cases

To ensure your code remains efficient, identify and avoid the following anti-patterns. Each anti-pattern includes a unit test idea (in pseudo-code) that can be implemented using the Screeps simulator or a testing framework.

1. Creeps withdrawing energy from spawns or extensions

Problem: Using spawns or extensions as an energy source starves the spawn of energy needed to produce new creeps. Since all energy for a creep must be present in spawns and extensions at the start of spawning [8], withdrawing from them slows or stops production.

Test idea: Instantiate a test world with a spawn, some extensions and a creep carrying no energy. Call your harvest logic for one tick and assert that the creep does **not** call `withdraw` on structures of type `STRUCTURE_SPAWN` or `STRUCTURE_EXTENSION`.

```
it('should not withdraw from spawns or extensions', () => {
  const creep = Game.creeps['tester'];
```

```
  // simulate tick where creep needs energy
  runHarvestLogic(creep);
  const intents = creep.memory.intents;
  // assert no withdraw intents target spawn/extension
  expect(intents.withdrawTargets).not.toContain(target => {
    const struct = Game.getObjectById(target);
    return struct.structureType === STRUCTURE_SPAWN ||
           struct.structureType === STRUCTURE_EXTENSION;
  });
});
```

2. Creeps withdrawing from towers

Problem: Towers are expensive to refill (10 energy per shot [9]) and are critical for defense. Creeps should never take energy from towers to perform other tasks.

Test idea: Place a tower with some energy and a creep needing energy. Run the creep's energy-acquisition logic and verify that the creep never calls `withdraw` on `STRUCTURE_TOWER`.

```
it('should not withdraw from towers', () => {
  const creep = Game.creeps['tester'];
  runHarvestLogic(creep);
  const intents = creep.memory.intents;
  expect(intents.withdrawTargets).not.toContain(id =>
Game.getObjectById(id).structureType === STRUCTURE_TOWER);
});
```

3. Using the upgrader container as general storage

Problem: A container near the controller should be reserved for upgraders; depositing or withdrawing energy from it for other purposes slows controller upgrades and can strand upgraders without fuel [10].

Test idea: Simulate a room with a controller container and a generic hauler carrying energy. After executing the hauler's deposit logic, assert that the controller container's store has not increased.

```
it('haulers should not fill the controller container', () => {
  const hauler = Game.creeps['hauler'];
  const controllerContainer = getControllerContainer();
  const before = controllerContainer.store[RESOURCE_ENERGY];
  runDepositLogic(hauler);
  const after = controllerContainer.store[RESOURCE_ENERGY];
  expect(after).toBe(before);
});
```

4. Mixed harvest/build roles in mid- to late-game

Problem: Creeps that both harvest and build/upgrader waste time switching tasks and moving around. A specialized miner can harvest continuously and a hauler can deliver energy far more efficiently.

Test idea: In simulation, spawn a mixed-role creep and separate miner/hauler creeps. Compare the total energy delivered to the controller over 500 ticks. The test should assert that the specialized pair delivers more energy.

5. Haulers depositing energy into mining containers

Problem: Source containers exist solely as a deposit point for miners. If haulers dump surplus energy back into a source container, it may overflow and waste harvest output or mislead miners about available capacity.

Test idea: In a test room with a source container and central storage, run hauler logic and verify that energy is only deposited into storage/valid targets, not source containers.

```
it('should not deposit into source containers', () => {
  const hauler = Game.creeps['hauler'];
  runDepositLogic(hauler);
  expect(hauler.memory.depositTarget).not.toBe(sourceContainer.id);
});
```

6. Over-filling towers while spawns are empty

Problem: Continuously filling towers to maximum capacity while spawns/extensions are empty leaves you unable to spawn new defenders or workers, even though towers might never fire.

Test idea: Simulate a room where towers are nearly full and spawns/extensions are empty. Run deposit logic for a hauler and assert that the chosen target is a spawn/extension rather than a tower.

```
it('should fill spawns/extensions before overfilling towers', () => {
  const hauler = Game.creeps['hauler'];
  const tower = Game.getObjectById('tower');
  tower.store[RESOURCE_ENERGY] = tower.store.getCapacity(RESOURCE_ENERGY) -
10;
  runDepositLogic(hauler);
  expect(hauler.memory.depositTarget).not.toBe(tower.id);
});
```

Conclusion

Energy management in Screeps is a complex but rewarding challenge. By understanding where energy comes from, how to harvest and transport it efficiently, and how to prioritize its distribution, you can ensure that your rooms grow quickly and remain defended. Spawns and extensions must always be kept full to maintain spawn uptime [8]; towers should be fueled when under threat [11]; and storage should act as a buffer for future needs. Avoiding common anti-patterns—such as withdrawing from spawns or towers or misusing the controller container—will keep your logistics clean and efficient. Implement the recommended priority order and write unit tests for the anti-patterns to catch mistakes early. With these guidelines, your Screeps AI will collect and distribute energy optimally and be well-prepared for expansion and defense.

---

1 2 4 5 6 7 16 Energy
https://wiki.screepspl.us/Energy/

3 RC1.0 Design: Basic Harvesting | Screeping
https://screeping.wordpress.com/2021/05/01/episode-4-1-rc1-basic-harvesting/

8 Creeps | Screeps Documentation
https://docs.screeps.com/creeps.html

9 15 StructureTower
https://wiki.screepspl.us/StructureTower/

10 11 12 14 Where to Get or Store Energy | Screeping
https://screeping.wordpress.com/2016/12/17/where-to-get-or-store-energy/

13 Need some help with "hauler" logic | Screeps Forum
https://screeps.com/forum/topic/2387/need-some-help-with-hauler-logic

17 Screeps #4: Pipeline Optimization | Field Journal
https://jonwinsley.com/notes/screeps-pipeline-optimization

18 Screeps #18: Spawn Uptime | Field Journal
https://jonwinsley.com/notes/screeps-spawn-uptime