Scan Scope — Full-Site Default (Updated)

PR #494 · feat/monitoring-scan-scope · commit 4858b654 · snayyar00/webability-platform · verified 2026-07-19

This supersedes the earlier scan-scope-backend.pdf — the default changed from `homepage` to `full_site`. Everything below is a fresh verification run against the current PR head, not a re-hash of the old document.

1. What changed

Product decision: paid customers should get full-site monitoring by default, not as an opt-in. Commit `4858b654` ("feat(monitoring): default scan_scope to full_site, not homepage") flips the schema column default. This is safe because the existing scan-time eligibility check (`isSiteEligibleForDeepScan`, called from `decideScanExecution` in `jobs/scheduledMonitoring.ts`) already defensively downgrades any site without an active, non-AppSumo, non-trial paid plan back to homepage-only *regardless of the schema default* — so non-paid accounts see zero behavior change.

2. Real test run — 33 / 33 passing

Executed in a fresh isolated git worktree checked out at PR #494's current head (`4858b654`, confirmed via `git log --oneline -3` before running anything):

```
npx tsx --env-file=.env --test test/monitoring-scan-scope.test.ts
```

Full unedited stdout:

```
[INFO] == decideScanExecution: homepage (explicit choice, single-page scan) ==
[INFO]   ✓ homepage -> fullSiteScan: false
[INFO]   ✓ homepage -> effectiveScope stays homepage
[INFO]   ✓ homepage -> never downgraded (nothing to downgrade from)
[INFO]   ✓ homepage -> unaffected by eligibility flag
[INFO] == Default scan scope for a brand-new site (no explicit scope choice) ==
[INFO]   ✓ DEFAULT_SCAN_SCOPE is full_site
[INFO]   ✓ new site, no explicit scope, active paid plan -> full_site scan by default
[INFO]   ✓ new site, no explicit scope, AppSumo/trial/free plan -> defensively downgraded
to homepage despite the full_site default
[INFO] == decideScanExecution: full_site (eligible) ==
[INFO]   ✓ full_site + eligible -> fullSiteScan: true
[INFO]   ✓ full_site + eligible -> effectiveScope stays full_site
[INFO]   ✓ full_site + eligible -> not downgraded
[INFO]   ✓ full_site + eligible -> no note (nothing unusual happened)
[INFO] == decideScanExecution: full_site (plan lapsed since scope was set) ==
[INFO]   ✓ full_site + ineligible -> fullSiteScan: false (defensive downgrade)
[INFO]   ✓ full_site + ineligible -> effectiveScope downgrades to homepage
[INFO]   ✓ full_site + ineligible -> downgraded: true
```

```
[INFO]   ✓ full_site + ineligible -> note explains why
[INFO] == decideScanExecution: specific_pages (eligible, no page-list executor yet) ==
[INFO]   ✓ specific_pages + eligible -> fullSiteScan: true (superset fallback)
[INFO]   ✓ specific_pages + eligible -> effectiveScope reported as full_site (honest about
the fallback)
[INFO]   ✓ specific_pages + eligible -> NOT flagged as downgraded (still ran the more
expensive scan, just not the exact requested scope)
[INFO]   ✓ specific_pages + eligible -> note documents the fallback + TODO
[INFO] == decideScanExecution: specific_pages (plan lapsed) ==
[INFO]   ✓ specific_pages + ineligible -> downgrades to homepage, same as full_site
[INFO] == normalizeSpecificPages: valid input ==
[INFO]   ✓ valid https pages pass through
[INFO]   ✓ http (not just https) is accepted
[INFO]   ✓ duplicates (incl. whitespace-only differences) are deduped to one entry
[INFO]   ✓ surrounding whitespace is trimmed
[INFO]   ✓ exactly the cap (25) is allowed, not rejected
[INFO] == normalizeSpecificPages: rejects empty ==
[INFO]   ✓ empty array throws
[INFO]   ✓ null throws
[INFO]   ✓ array of only blank strings throws (nothing usable after trim)
[INFO] == normalizeSpecificPages: rejects over the page cap ==
[INFO]   ✓ 26 pages (one over the cap) throws
[INFO]   ✓ error message names the cap
[INFO] == normalizeSpecificPages: rejects invalid URLs ==
[INFO]   ✓ a non-URL string throws
[INFO]   ✓ a non-http(s) protocol (e.g. ftp) throws
[INFO]   ✓ a javascript: URL throws (protocol allowlist, not blocklist)

33 passed, 0 failed
```

33 / 33 passing

3. Real migration — quoted verbatim

From `api/migrations/20260720_add_scan_scope_to_site_monitoring_settings.ts`, checked out at PR head:

```
export async function up(knex: Knex): Promise<void> {
  const hasScanScope = await knex.schema.hasColumn('site_monitoring_settings',
'scan_scope')
  if (!hasScanScope) {
    await knex.raw(`
      ALTER TABLE site_monitoring_settings
      ADD COLUMN scan_scope ENUM('homepage', 'full_site', 'specific_pages') NOT NULL
DEFAULT 'full_site' AFTER frequency
    `)
  }
  ...
}
```

The comment block above it (also quoted verbatim) explains the reasoning:

```
Default is 'full_site', not 'homepage': product wants full-site monitoring
to be what customers get out of the box, not an opt-in. This is safe
despite 'full_site' being the plan-gated, more-expensive scope, because
the gate is enforced independently of this column's default — see
isSiteEligibleForDeepScan (allowedSites.service.ts) called from
decideScanExecution (jobs/scheduledMonitoring.ts) at scan time, every
run, for every site, regardless of what scan_scope is stored.
```

4. Live evidence — two real calls against the real staging DB

Both calls were made in-process (service functions imported and invoked directly with `tsx`, exercising the exact code path a resolver call would use) against the real staging MySQL instance (`platform/api/.env`), no mocking.

4a. New paid-plan site → full_site by default

Every one of the 588 active/paid/non-AppSumo/non-trial sites already present in staging already had a `site_monitoring_settings` row, so a genuinely never-configured site was needed to prove the default. A throwaway test site + active paid plan was created, used, and deleted within the same script run — no existing customer data was touched.

Step	Result
Created test site	`allowed_sites.id = 1714`, owner = existing staging dogfood account (user 723)
Created test plan	`sites_plans.id = 2522` — `is_appsumo=0, is_trial=0, stripe_status='active'`
`site_monitoring_settings` row before call	none (never configured, as intended)
`getSiteMonitoringSettings(1714, 723)` result	`scan_scope: "full_site"`, `id: 0` (placeholder, not persisted)
`site_monitoring_settings` row after call	still none — read-only, no write occurred
Cleanup	test plan + test site deleted; re-queried independently afterward and confirmed both are gone

```
[INFO] DEFAULT_SCAN_SCOPE export => full_site
[INFO] [a] created throwaway test site id=1714 url=verify-fullsite-default-
1784509650134.example.com
[INFO] [a] created throwaway active paid sites_plans id=2522 for site 1714
[INFO] [a] site_monitoring_settings row for new site BEFORE call: (none — never configured,
as intended)
[INFO] [a] getSiteMonitoringSettings(newSite, owner) => {
  "id": 0,
  "site_id": 1714,
  "frequency": "monthly",
  "scan_scope": "full_site",
  "scan_scope_pages": null,
  ...
```

```
}
[INFO] [a] RESULT: scan_scope === 'full_site' ? true  (placeholder row id=0, 0 = never
persisted)
[INFO] [a] site_monitoring_settings row for new site AFTER call: (still none — confirms
read-only placeholder, no write occurred)
[INFO] [a] cleaned up: deleted sites_plans id=2522 and allowed_sites id=1714
```

4b. Same AppSumo site 1696 → still defensively downgraded to homepage

Re-verification of the exact site used in the earlier scan-scope-backend.pdf verification: `allowed_sites.id = 1696` (`appsumoclaim11783668750.example.com`), whose `sites_plans` row has `is_appsumo = 1`, `stripe_status = 'active'` — a real, currently-active AppSumo plan.

```
[INFO] [b] isSiteEligibleForDeepScan(1696) => false
[INFO] [b] decideScanExecution(DEFAULT_SCAN_SCOPE='full_site', eligible=false) => {
  "fullSiteScan": false,
  "effectiveScope": "homepage",
  "downgraded": true,
  "note": "scan_scope=\"full_site\" requires an active paid plan; plan is no longer
eligible, downgrading this run to homepage"
}
[INFO] [b] RESULT: effectiveScope === 'homepage' despite full_site default ? true /
downgraded === true ? true
```

Even with the schema default now `full_site`, an AppSumo site with no explicit scope choice is still scanned homepage-only at execution time — the two mechanisms (stored default vs. scan-time eligibility gate) are independent, and the gate wins.

5. Net effect

Site type	scan_scope default	What actually runs (decideScanExecution)
New site, active non-AppSumo/non-trial paid plan	`full_site`	`full_site` — genuinely full-site monitoring, no opt-in required
New site, AppSumo / trial / free / lapsed plan	`full_site` (stored, unused)	`homepage` — defensively downgraded, zero cost regression

Verified in an isolated git worktree off PR #494 head (commit 4858b654) · test run + migration source + both live DB verifications captured directly in this session · staging DB only, no production reads or writes · throwaway test site/plan created and deleted for §4a, deletion independently re-confirmed after the fact.