Productization Path

The Productization Path spec converts a capability registered in c10-capability-inventory from its current form — often a bespoke internal tool or consulting deliverable — into a repeatable, sellable offer. Productization is distinct from "building the feature": it is the disciplined removal of one-off consulting dependencies, the encoding of tacit expert judgment into standard interfaces, and the definition of what the product delivers every time without heroics. This spec relies on e4-workflow-artifacts as the workflow substrate: every capability that is productizable has at its core a workflow that can be modeled, versioned, and handed off. Without that encoding, the capability is a service dependent on individuals, not a product. The output of this spec feeds c10-revenue-model-design (what is being priced) and c10-defensibility-map (which productization investments become moat / defensibility).

Productization Readiness Criteria

Before writing the path spec, assess the capability against these readiness criteria. A capability scoring fewer than four of six is not yet productization-ready and should remain in explore tier on c10-capability-inventory.

#	Criterion	Assessment method
1	The core workflow is documented and executable without the original author present	Walk a new operator through the workflow without coaching; success = correct output
2	The top-80% input cases are handled by deterministic logic; only edge cases require judgment	Classify a sample of 20 real inputs; count deterministic vs. judgment steps
3	The output schema is stable (no per-customer customization required for the core output)	Check whether the last 3 deliveries produced the same output structure
4	The evaluation rubric exists and a passing threshold has been defined in s2-scoring-system	Confirm a rubric exists; confirm it has passed at least one real run
5	The unit cost of delivery is known within ±30% (from e6-unit-economics-worksheet)	Pull the relevant row; flag if unpopulated
6	At least one non-founder customer has used or piloted the capability	Cite the pilot account or flag as unmet

Spec Template

```
productization_id: <kebab-case>
capability_ref: <capability_id from c10-capability-inventory>
version: <semver>
last_reviewed: <ISO date>

# ■■ Repeatability Analysis ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
repeatability_analysis:
  core_workflow_ref: <e4-workflow-artifacts instance id>
```

```
  deterministic_coverage_pct: <0–100; share of real input cases handled by deterministic
logic>
  judgment_residual_cases:
    - <describe each case type requiring human or expert judgment>
  judgment_handling: <escalate | HITL-gate | config-option | out-of-scope>
  one_off_dependencies:
    - dependency: <name of consulting dependency, custom script, or expert person>
      removal_plan: <how it is replaced with a standard interface or documented rule>
      removal_target: <version or date>

# ■■ Standard Interfaces ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
standard_interfaces:
  input_contract:
    schema_ref: <pointer to the canonical input schema>
    accepted_formats: [<format list>]
    validation_rule: <what happens on invalid input — reject, warn, transform>
  output_contract:
    schema_ref: <pointer to the canonical output schema>
    output_types: [<type list from acceptable_outputs in s1-operating-context-canvas>]
    variability_budget: >
      <how much output variation is permitted per run on identical inputs;
      if zero, the step is deterministic; if non-zero, state the tolerance>
  configuration_surface:
    - param: <name>
      scope: <per-tenant | per-run | global>
      description: <what it controls>
      default: <value>

# ■■ Consulting Dependency Elimination ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
consulting_elimination:
  eliminated_dependencies: []  # fill after removal_plan execution
  residual_service_layer:
    description: >
      <any remaining service work that is intentional — e.g., onboarding,
      rule-layer configuration, annual rule-set updates — and is productized
      as a defined SKU rather than bespoke consulting>
    sku_name: <e.g., "Annual Rule Maintenance Subscription">
    sku_priced_separately: <yes | no>

# ■■ Packaging ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
packaging:
  tiers:
    - tier_name: <e.g., Starter | Professional | Enterprise>
      included_capabilities: [<capability_id list>]
      excluded_capabilities: [<capability_id list>]
      configuration_scope: <what the buyer can configure at this tier>
      support_level: <self-serve | email | dedicated-CSM>
  modular_add_ons:
    - add_on_name: <e.g., Multi-Jurisdiction Rule Layer>
      description: <what it adds>
      dependency: <capability_id or workflow artifact it builds on>

# ■■ Productization Milestones ■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■
milestones:
  - milestone: <short name>
    definition_of_done: <observable, verifiable outcome>
    target_date: <ISO date or sprint label>
    owner: <role>
    blocking_artifact: <artifact id that must exist before this milestone>
```

Worked Example — GROW Library Productization Path

This example productizes the grow-library-builder-skill capability from c10-capability-inventory — the GROW library itself as a commercializable builder toolkit. Figures marked (illustrative) are placeholders.

```
productization_id: grow-library-builder-toolkit
capability_ref: grow-library-builder-skill
version: 0.1.0
last_reviewed: 2026-05-29

repeatability_analysis:
  core_workflow_ref: e4-workflow-artifacts
  deterministic_coverage_pct: 85
  judgment_residual_cases:
    - Selecting the correct evaluation tier (demo/operational/high-stakes) for a
      novel agent type not covered by existing s2-scoring-system rubrics
    - Choosing moat_source classification for capabilities without a clear analog
      in the glossary
  judgment_handling: HITL-gate
  one_off_dependencies:
    - dependency: >
        Artifact build requires a practitioner who has internalized the house
        style and the C1–C12 contract surface; no new practitioner can onboard
        from the library alone without guided examples.
      removal_plan: >
        Add a scaffolded onboarding sequence (Artifact 0: Build Your First
        Agent Canvas) that walks a practitioner through s1-operating-context-canvas
        using a real case, with inline rubric feedback against s2-scoring-system.
      removal_target: v0.3.0
    - dependency: >
        Cross-cluster wiring (reciprocal upstream_deps / downstream_consumers)
        currently reconciled centrally by the library owner.
      removal_plan: >
        Automate via a lint script that validates all canonical ids in
        upstream_deps and downstream_consumers resolve to real artifact files.
      removal_target: v0.2.0

standard_interfaces:
  input_contract:
    schema_ref: shared-artifact-schema
    accepted_formats: [markdown, yaml-frontmatter]
    validation_rule: >
      Reject files with missing or malformed frontmatter fields; warn on
      unresolvable canonical cross-reference ids.
  output_contract:
    schema_ref: shared-artifact-schema
    output_types: [template, register, policy, spec, checklist, matrix, playbook, map]
    variability_budget: >
      Artifact body is intentionally variable (practitioner adapts examples);
      frontmatter fields and cross-reference ids are zero-variability (must
      match canonical spec exactly).
  configuration_surface:
    - param: owner_ecosystem
      scope: per-tenant
      description: >
        The organization name and worked-example domain substituted into the
        fillable blocks (e.g., "GoodSam.ai / South Walton County" vs.
        "Acme Corp / Chicago Transit").
      default: GROW Library
    - param: evaluation_tier
      scope: per-run
      description: >
        The quality tier (demo / operational / high-stakes) applied by
        s2-scoring-system when evaluating artifacts produced in a build session.
      default: operational

consulting_elimination:
  eliminated_dependencies:
    - Guided onboarding (resolved via scaffolded Artifact 0 — pending v0.3.0)
  residual_service_layer:
    description: >
      Annual library maintenance (rule updates, new cluster additions, contract
      schema bumps) and custom-domain worked-example packs (e.g., a
      stormwater-specific pack, a DAO-governance pack) are intentional service
```

```
      SKUs that extend the library without requiring custom consulting.
    sku_name: GROW Library Annual Maintenance + Domain Pack
    sku_priced_separately: yes

packaging:
  tiers:
    - tier_name: Practitioner
      included_capabilities: [grow-library-builder-skill]
      excluded_capabilities: []
      configuration_scope: owner_ecosystem substitution only
      support_level: self-serve
    - tier_name: Team
      included_capabilities: [grow-library-builder-skill, govtech-compliance-agent]
      excluded_capabilities: []
      configuration_scope: >
        owner_ecosystem, evaluation_tier, and custom domain-pack inclusion
      support_level: email
    - tier_name: Enterprise
      included_capabilities:
        - grow-library-builder-skill
        - govtech-compliance-agent
        - stormwater-permit-prescreener
      excluded_capabilities: []
      configuration_scope: full configuration surface + custom jurisdiction rule layers
      support_level: dedicated-CSM
  modular_add_ons:
    - add_on_name: Stormwater Jurisdiction Rule Pack
      description: >
        Pre-validated FDEP rule mapping for FL special districts; drops into
        t7-govtech-package as a jurisdiction overlay; maintained annually.
      dependency: t7-govtech-package
    - add_on_name: DAO Governance Pack
      description: >
        PP DAO-derived governance charter and decision-rights templates
        pre-configured for a c9-dao-governance build.
      dependency: c9-governance-purpose-charter

milestones:
  - milestone: Lint automation live
    definition_of_done: >
      CI script validates all canonical ids in upstream_deps and
      downstream_consumers resolve to real files; zero false positives on
      the current 70-artifact library.
    target_date: "2026-Q3"
    owner: library-maintainer
    blocking_artifact: shared-artifact-schema
  - milestone: Practitioner onboarding pack published
    definition_of_done: >
      New practitioner completes s1-operating-context-canvas for a real
      agent using Artifact 0 scaffold; output scores >= 4.0 on s2-scoring-system
      rubric without coaching.
    target_date: "2026-Q3"
    owner: library-maintainer
    blocking_artifact: s2-scoring-system
  - milestone: First external pilot
    definition_of_done: >
      One non-GoodSam team completes a full Course 1 build (7 artifacts)
      using only the library and the Practitioner onboarding pack; proof-of-value
      scorecard published.
    target_date: "2026-Q4"
    owner: library-maintainer
    blocking_artifact: c10-gtm-architecture
```

Productization Anti-Patterns

Three failure modes appear repeatedly when teams rush productization. First, renaming a consulting engagement as a "product" without removing the judgment dependencies — the delivery still requires the same expert, now invisibly embedded in a support contract. Detect this by auditing one_off_dependencies: if none are listed, the analysis is incomplete. Second, over-configuring the product surface to accommodate every prospect's request — configuration explosion is the symptom of an ICP that is not narrow enough; resolve by tightening c10-customer-market-framing before expanding configuration. Third, shipping before the e6-unit-economics-worksheet row is populated — cost surprises discovered post-contract are moat-destroying. The anti-vaporware rule in c10-revenue-model-design enforces this: no pricing hypothesis without a populated unit-cost row.