RefactorOps

Scope Starter Pack

The standardized way to define, defend, and control scope — from first draft to change requests — so delivery stays predictable.

By Michael Smith — Founder, RefactorOps
“Turning engineering chaos into profit.”

Why this matters

Most scope pain is self-inflicted: assumptions never written down, exclusions left implicit, and changes absorbed quietly. This pack fixes that. You’ll write scope the same way every time, track every change, and align owners so decisions have one **Accountable** person.

- Clear, reusable structure for SOWs and statements.
- Explicit out-of-scope lists to kill surprises.
- Scope Ledger + SCRs so creep becomes a business decision, not a gift.

> Rule of thumb: **Write the boundaries before the estimates.** If it isn’t written, it doesn’t exist.

How to use this

1. Fill out the **Scope Summary** and the **Standard Scope Framework**.
2. Create a shared **Scope Ledger** on day one.
3. Require an **SCR** for any change to $/timeline/deliverables.
4. Run the **Alignment Checklist** before kickoff.
5. Publish the **Scope Policy** so no one can claim ignorance later.

1) Scope Summary

Project Name

Date / Version

Prepared By

Reviewed By

Objective

Acceptance Criteria (high-level)

In-Scope Items

Out-of-Scope Items

Risks / Constraints

Assumptions

Dependencies

2) Standard Scope Framework

Use this outline to define scope the same way every time. It becomes your SOW backbone.

BUSINESS & TECHNICAL CONTEXT

FUNCTIONAL & TECHNICAL DELIVERABLES

Explicitly Out of Scope

Assumptions & Dependencies

Risk & Mitigation

Post-Launch Expectations

ACCEPTANCE CRITERIA

3) Scope Ledger

Log every request. Decide transparently. Prevent creep. Keep this visible to the whole team.

DATE	REQUESTOR	CHANGE SUMMARY	IMPACT (HRS/$)	DECISION	NOTES

4) Scope Change Request (SCR)

Use when a change impacts budget, timeline, or major deliverables. CE/PM/Tech align before client escalation.

Request Title

Date

Requested By

Reference (Ledger Row / Ticket #)

REQUEST SUMMARY

Estimated Impact (Hours)

Estimated Cost ($)

Schedule Impact (Days)

DECISION & APPROVALS

ROLE	NAME	DECISION	DATE	NOTES
Engineering				
Delivery (PM/BA)				
Client Engagement				
Client				

5) “Yes, but ...” Responses & Say No Examples

REQUEST	SUGGESTED RESPONSE
“Can we add this one more feature?”	“Yes, but we’ll need to push something else or move the launch date. Which is more important?”
“Can we keep this old system too?”	“Yes, but we’ll need additional budget to maintain both versions.”
“Can we create three different designs?”	“Yes, but we must choose one before development starts or we’ll delay the build.”
“Can we scope that later?”	“Yes, but that puts us at risk of rework. Want to do a 30-minute workshop now?”
“Can we make it dynamic?”	“Yes, but QA must test all variations and we’ll need more creative support.”

SAY NO — EXAMPLES

EXAMPLE REQUEST	WHAT TO SAY	WHY?
“Add Google oAuth too?”	“Sounds good. Let’s log it in the Scope Ledger to ensure it’s accounted for.”	Adds a new auth provider; likely not in original scope.
“We’ll sync this to Salesforce now, right?”	“Let’s double check. I think our scope assumed no external integrations.”	Breaks an assumption; scope needs re-evaluation.
“Build a reporting dashboard while we’re at it.”	“That sounds big. Let’s flag it in the Scope Ledger for review.”	Brand new request; likely out of original scope.
“Let users upload files here.”	“May require resources we didn’t plan for. Logging for review.”	Introduces security/perf/storage risk and extra work.
“Personalize content in this launch.”	“We’ll log it and evaluate how it affects current priorities.”	Late change likely pushes current priorities.

6) Scope Alignment Checklist

ITEM	OWNER	DONE
Business objective clearly defined	_____	☐
Success metrics documented	_____	☐
Assumptions validated with Tech Lead	_____	☐
Explicit out-of-scope items reviewed	_____	☐
RACI confirmed (one “A” per decision)	_____	☐
Scope Ledger created and shared	_____	☐
Timeline reviewed with client	_____	☐
Scope sign-off received	_____	☐

7) Scope Policy (RefactorOps Standard)

1. All projects must use this Scope Framework with explicit exclusions.
2. Every new idea/change is logged in the Scope Ledger.
3. Engineering, Delivery, and Client Engagement co-own scope control.
4. SCRs are mandatory when budget/timeline/deliverables are impacted beyond tolerance.
5. Scope is reviewed at least bi-weekly during delivery.

Turning engineering chaos into profit.