VOTS WORKFORCE LEARNING

Scope Creep

When “One Small Change” Becomes a Different Project

Recognizing, Documenting, Evaluating, Approving, and Controlling Changes

PROGRAM & PROJECT MANAGEMENT • SOURCE TRAINING MANUAL

Change Control • Scope Management • Governance • Execution Discipline

Veterans of the Storm (VOTS)

Purpose and Use

Scope creep is often described as a project-management nuisance: extra requests appear, the schedule slips, and the team gets frustrated. That description is too shallow. Scope creep is a control problem. It happens when the project cannot reliably distinguish approved work from requested work, cannot measure the impact of change, or cannot enforce who has authority to alter the baseline.

This manual teaches a practical change-control system for recognizing changes early, capturing them without losing work, evaluating their real impact, routing them to the correct decision-maker, updating the project baseline when approved, and preventing rejected or deferred work from quietly entering execution.

CENTRAL PRINCIPLE

A change is not “small” because the request sounds small. A change is small only after its effect on scope, schedule, cost, quality, risk, dependencies, and downstream work has been evaluated.

Learning Objectives

- Recognize the difference between clarification, correction, defect repair, enhancement, and true scope change.
- Identify how informal requests bypass project controls.
- Create a usable change request rather than relying on chat, email, or memory.
- Evaluate change impact across scope, schedule, cost, resources, quality, risk, dependencies, and benefits.
- Define approval authority and prevent unauthorized commitments.
- Maintain a change log and preserve an auditable decision trail.
- Update project baselines after approved changes so the plan remains truthful.
- Control cumulative “small” changes that create large aggregate impact.
- Handle urgent changes without abandoning governance.
- Close projects with clear acceptance and a record of approved, rejected, deferred, and unimplemented changes.

The Change Control Cycle

1. Recognize the change.
2. Capture the request.
3. Classify the request.
4. Evaluate impact.
5. Route to the proper decision authority.
6. Approve, reject, defer, or request more information.
7. Update the baseline and communicate the decision.
8. Implement and verify the approved change.
9. Close the change record.

WHY THIS MATTERS

Without a defined cycle, the project treats requests as conversations. Conversations are easy to forget, reinterpret, or act on before the consequences are understood.

Contents

Part I — Understanding Scope Creep

1. What Scope Creep Really Is
2. Change Is Not the Enemy
3. How Small Changes Compound
4. The Hidden Channels Where Scope Enters

Part II — Establishing a Baseline Worth Protecting

5. Scope Must Be Testable
6. Requirements and Acceptance Criteria
7. Assumptions, Constraints, and Dependencies
8. Ownership and Change Authority

Part III — Recognizing and Capturing Change

9. Clarification vs. Change
10. The Change Request
11. The Change Log
12. Informal Requests and Commitment Risk

Part IV — Evaluating Impact

13. The Eight-Dimension Impact Review
14. Schedule and Dependency Effects
15. Cost and Resource Effects
16. Quality, Risk, and Benefit Tradeoffs
17. Cumulative Change Load

Part V — Decision and Approval

18. Approval Thresholds
19. Approve, Reject, Defer, or Split
20. Emergency and Time-Critical Changes
21. Communicating the Decision

Part VI — Implementing Approved Change

22. Rebaseline the Project
23. Prevent Partial or Shadow Implementation

24. Verify Completion and Acceptance

25. Version Control and Traceability

Part VII — Preventing Recurrence

26. The Behaviors That Create Scope Creep

27. Client and Stakeholder Pressure

28. Gold Plating and Internal Scope Creep

29. Recovering a Project Already Overrun by Change

30. Closing the Project Cleanly

Applied Case Studies

Field Tools

Application Exercises

Final Knowledge Check and Answer Key

PART I

Understanding Scope Creep

Scope creep is not simply “more work.” It is uncontrolled change to the project’s agreed boundaries. The word uncontrolled matters. Projects can absorb major changes successfully when those changes are visible, evaluated, authorized, funded, scheduled, and incorporated into the plan. A much smaller change can be more damaging when it enters informally and forces unrecognized tradeoffs.

1. What Scope Creep Really Is

A project baseline represents the approved understanding of what will be delivered, under what constraints, and by when. Scope creep occurs when work expands or changes without corresponding approval and adjustment to the baseline. The project begins doing one thing while the official plan still describes another.

- **Controlled change.** The change is documented, evaluated, approved by the correct authority, and reflected in the plan.
- **Scope creep.** The work changes but the baseline, authority, schedule, budget, or acceptance expectations do not change with it.
- **Requirement discovery.** A missing requirement is discovered; it may still become a change if the approved baseline must be altered.
- **Defect correction.** Work is brought back into conformance with an already-approved requirement; this is not automatically new scope.

DIAGNOSTIC QUESTION

Did the approved obligation change? If yes, treat it as a change until proven otherwise. Do not let wording such as “quick,” “minor,” or “while you’re in there” make the decision for you.

2. Change Is Not the Enemy

Projects exist in changing environments. Users learn more, regulations change, technology shifts, leaders make new decisions, and assumptions prove wrong. A change-control system should not punish useful learning. Its purpose is to stop unexamined change from becoming invisible work.

Strong change control lets the organization say yes intelligently. It also lets the organization say no, not now, or only if another commitment moves.

Healthy Change Management	Uncontrolled Change
Request is captured before work begins.	Request is acted on from a conversation or message.
Impact is evaluated before approval.	Impact is discovered during execution.

Authority is explicit.	Whoever asks loudest appears to have authority.
Baseline changes when the work changes.	Original deadline and budget remain unchanged.
Decision is traceable.	Team relies on memory and email archaeology.

3. How Small Changes Compound

A single two-hour change may truly be small. Ten two-hour changes are no longer small. More importantly, small changes can alter sequencing, trigger rework, consume contingency, delay reviews, or create new acceptance conditions. Their impact is not limited to the estimated build time.

Cumulative change load is the combined burden of approved and unapproved changes on the project. Teams often evaluate each request in isolation and miss the aggregate effect.

COMPOUNDING EFFECT

Five “one-day” changes do not necessarily equal five days. If each change interrupts planned work, requires review, affects another deliverable, or creates retesting, the total schedule effect can be much larger.

4. The Hidden Channels Where Scope Enters

- Meetings where someone says, “Can we also…?” and the team begins working before a decision is recorded.
- Executive comments treated as automatic approval even when the executive did not see the impact analysis.
- Direct requests from customers to individual team members.
- Chat messages and email threads that bypass the project owner.
- Design reviews where preferences are added as if they were defects.
- Vendor conversations that change an interface, assumption, or dependency.
- Internal “while we are here” improvements added by the team.
- Temporary workarounds that become permanent deliverables.

The common pattern is not bad intent. It is a missing boundary between request, evaluation, decision, and implementation.

PART II

Establishing a Baseline Worth Protecting

5. Scope Must Be Testable

You cannot control scope that cannot be described. Statements such as “improve the customer experience,” “modernize the process,” or “make the portal easier” may be useful objectives, but they are not sufficient scope boundaries. The team needs observable deliverables, exclusions, acceptance conditions, and interfaces.

- **In scope.** What the project is explicitly responsible for delivering.
- **Out of scope.** Related work the project is explicitly not responsible for.
- **Deliverables.** The concrete outputs the project must produce.
- **Acceptance.** The evidence required to consider each deliverable complete.
- **Interfaces.** Where the project hands off to or receives work from another team, system, or vendor.

CONTROL STARTS HERE

If the baseline is vague, every later disagreement can be framed as a clarification. Clear scope turns “I thought that was included” into a question that can actually be resolved.

6. Requirements and Acceptance Criteria

Requirements define obligations. Acceptance criteria define the evidence that the obligation has been met. Weak criteria create late-stage scope disputes because stakeholders can continue changing their interpretation of “done.”

- Use observable language.
- Separate must-have requirements from preferences.
- Name the approver for acceptance.
- Record regulatory, legal, accessibility, security, and performance requirements explicitly.
- Link each major requirement to a deliverable or test.

WEAK CRITERION

“The dashboard should be user-friendly.”

STRONGER CRITERION

“A first-time user can locate the three required reports from the landing page without administrative assistance; labels and navigation match the approved prototype.”

7. Assumptions, Constraints, and Dependencies

Scope creep is often blamed on stakeholders when the real cause is an untested assumption. If the project assumed a data source existed, assumed a vendor would perform a task, or assumed an approval was not required, the later correction can look like "new work" even though it was created by the original planning gap.

Type	Question	Example
Assumption	What are we treating as true without proof?	Existing data can be reused.
Constraint	What fixed limit must the project operate within?	Go-live cannot move beyond fiscal year end.
Dependency	What must another party or event provide?	Security review must be completed before production release.

Record these items because a change to any of them can create a scope, schedule, or cost change even if no deliverable name changes.

8. Ownership and Change Authority

One of the fastest ways to create scope creep is allowing people to request work without defining who can commit the project to it. Request authority and approval authority are different.

- **Requester.** May identify a need and submit a change.
- **Evaluator.** Analyzes impact and options.
- **Approver.** Has authority to change the baseline within defined thresholds.
- **Sponsor or governance body.** Decides changes above the project manager's delegated authority.
- **Implementer.** Performs approved work but should not be expected to authorize the commitment.

AUTHORITY RULE

No team member should have to decide whether a stakeholder is "important enough" to bypass change control. Define approval authority before pressure arrives.

PART III

Recognizing and Capturing Change

9. Clarification vs. Change

Not every conversation that changes wording changes scope. The practical distinction is whether the request changes the approved obligation, constraints, acceptance conditions, interfaces, or effort required to meet the obligation.

Clarification	Change
Explains an existing approved requirement without adding obligation.	Adds, removes, replaces, or materially alters an approved obligation.
Resolves ambiguous wording while preserving the same acceptance result.	Changes the acceptance result or what the stakeholder will receive.
Corrects a misunderstanding about the approved baseline.	Requires new work, rework, cost, time, or risk beyond the baseline.

When uncertain, log the request. Logging a question does not mean approving a change. It preserves the decision trail until classification is complete.

10. The Change Request

A useful change request is brief enough to complete but complete enough to support a decision. It should separate the requested outcome from the requester's preferred solution.

- Request ID and date
- Requester
- Current baseline or requirement affected
- Requested change
- Reason / business need
- Desired date
- Consequence of not making the change
- Known dependencies or constraints
- Initial classification
- Attachments or evidence

CAPTURE THE NEED, NOT JUST THE SOLUTION

"Add a new export button" is a proposed solution. "Finance needs a monthly extract in CSV for reconciliation" states the need. Evaluating the need may reveal a lower-impact option.

11. The Change Log

The change log is the project's memory. It shows what was requested, what was decided, when, by whom, and whether the approved change reached completion. A change log also makes cumulative load visible.

- Unique ID
- Short title
- Status
- Requester
- Date submitted
- Impact summary
- Decision
- Decision authority
- Decision date
- Implementation owner
- Target date
- Verification / closure date

Statuses should be simple and mutually understandable: Submitted, Under Evaluation, Awaiting Decision, Approved, Rejected, Deferred, Implementing, Verified, Closed.

12. Informal Requests and Commitment Risk

The project manager's hardest job is often not documenting formal requests; it is intercepting informal commitments before they become expectations. A polite "we can look at that" can later be remembered as "you agreed to do it."

SAFE RESPONSE PATTERN

"I'll capture that as a change request so we can see the schedule, cost, and dependency impact before we commit to it."

This response does not reject the request. It protects both the requester and the project from a decision made without enough information.

PART IV

Evaluating Impact

13. The Eight-Dimension Impact Review

A change should be evaluated across more than effort hours. Use eight dimensions so the decision-maker sees the tradeoff rather than only the requested feature.

10. Scope: What deliverables or requirements change?
11. Schedule: What dates, milestones, reviews, or float are affected?
12. Cost: What labor, vendor, licensing, procurement, or operational cost changes?
13. Resources: Which people or specialized skills must shift?
14. Quality: Does the change alter quality criteria, testing, or acceptance?
15. Risk: What new risks are introduced or existing risks increased?
16. Dependencies: What upstream or downstream work changes?
17. Benefits: What value does the change create, protect, or enable?

IMPACT IS THE DECISION PRODUCT

The evaluator's job is not to say yes or no. The evaluator's job is to make the consequences legible enough that the authorized decision-maker can choose deliberately.

14. Schedule and Dependency Effects

The direct build time of a change is often the smallest part of schedule impact. A request may reopen design, trigger procurement, force retesting, require another legal or security review, or delay a dependent team.

- Work that must stop or be resequenced
- Milestones that move
- Review windows that must reopen
- External dependencies
- Lead times and procurement
- Retesting and regression work
- Lost contingency or schedule float
- Impact on committed delivery dates

SCHEDULE QUESTION

If this change is approved, what must move, wait, repeat, or happen earlier?

15. Cost and Resource Effects

Changes consume more than money. They consume finite attention and capacity. A "free" internal change may still displace planned work, overload a specialist, or require overtime.

- Direct labor hours
- Contractor or vendor cost
- Licensing, equipment, or procurement
- Opportunity cost of displaced work
- Specialist availability
- Overtime or surge capacity
- Training and support burden
- Long-term maintenance cost

When resources are fixed, approving new work is also a prioritization decision. The change decision should identify what gives way if capacity does not increase.

16. Quality, Risk, and Benefit Tradeoffs

A change can improve functionality while reducing schedule resilience or increasing operational risk. Decision-makers need to see the full trade, not only the requested benefit.

- Does quality improve, decline, or require new tests?
- Does the change create security, privacy, compliance, safety, or reliability risk?
- Does it introduce a new single point of failure?
- Does it make the solution harder to maintain?
- Does the expected benefit justify the added complexity?
- Does the change protect a required outcome or merely add preference?

17. Cumulative Change Load

Evaluate the portfolio of changes, not just the latest request. A project can remain “on track” after each individual approval while collectively consuming all schedule float and contingency.

- Number of approved changes
- Total added effort
- Total schedule movement
- Contingency consumed
- New dependencies created
- Open changes still awaiting implementation
- Changes awaiting decision
- Rework caused by prior changes

GOVERNANCE TRIGGER

Set thresholds that force a broader review, such as when cumulative approved change exceeds 10% of planned effort, consumes more than half of schedule contingency, or alters a major project outcome. The exact threshold should fit the project.

PART V

Decision and Approval

18. Approval Thresholds

Not every change requires an executive committee. Change authority should scale with impact. Clear thresholds prevent both bureaucracy and unauthorized commitments.

Level	Example Authority	Typical Trigger
Low	Project manager	No milestone, budget, or acceptance impact; small effort within contingency.
Medium	Sponsor / product owner	Affects milestone, priority, resource allocation, or approved requirement.
High	Steering/governance body	Material budget, contractual, regulatory, benefits, or delivery-date impact.

Thresholds should be documented before the project is under pressure. They can be based on dollars, effort, schedule movement, risk classification, requirement type, or contractual impact.

19. Approve, Reject, Defer, or Split

A mature change-control process has more than yes and no. The decision can be structured to protect the project while preserving value.

- **Approve.** The value justifies the impact and authority is sufficient.
- **Reject.** The change does not justify the impact or conflicts with project objectives.
- **Defer.** The need may be valid but belongs in a later release, phase, or backlog.
- **Split.** Approve a lower-impact subset and defer or reject the rest.
- **Return for information.** The request cannot be decided until impact or need is clearer.

NO SILENT APPROVAL

If work begins before the decision is recorded, the organization has effectively approved the change without knowing who made the decision.

20. Emergency and Time-Critical Changes

Urgency may justify an accelerated process, not the absence of process. Emergency changes should still record the reason, authority, minimum impact review, implementation owner, and retrospective review.

18. Identify why normal timing is impossible.
19. Confirm who has emergency authority.
20. Evaluate immediate safety, legal, operational, schedule, and cost impact.
21. Record the decision before or immediately after action if circumstances truly prevent prior documentation.
22. Perform a retrospective review and update the baseline and change log.

A project that labels every late request “urgent” has a governance problem, not an emergency-change process.

21. Communicating the Decision

A change decision is not complete until affected people know what changed. Communication should identify the decision, rationale, new commitments, affected dates, and any work that is no longer expected.

- Decision and status
- Effective date
- Baseline items changed
- New or removed deliverables
- New owner or decision rights if applicable
- Schedule or milestone change
- Budget or resource change
- Risks and dependencies
- What teams should stop doing

STOP-WORK CLARITY

When a change replaces work, say what is no longer required. Otherwise teams may execute both the old and new versions.

PART VI

Implementing Approved Change

22. Rebaseline the Project

Approval is not the end of change control. The project plan must become truthful again. If the work changes but the schedule, scope statement, backlog, requirements, budget, and risk register do not, the team is now executing against contradictory sources.

- Update scope and requirement records
- Update deliverables and acceptance criteria
- Update schedule and milestones
- Update budget and resource assignments
- Update risk and dependency registers
- Update decision logs and governance records
- Update stakeholder communications
- Update contracts or statements of work where required

BASELINE RULE

There should be one current answer to “What are we delivering now?” If different systems show different approved scope, the change is not fully implemented administratively.

23. Prevent Partial or Shadow Implementation

Sometimes a request is formally rejected but pieces of it still enter the product through helpful employees, design tweaks, or untracked follow-up. This creates shadow scope: work that exists without a corresponding approved obligation.

- Link work items to approved requirements or change IDs.
- Review backlog additions for authorization.
- Do not let rejected changes remain assigned as “nice to have” work inside the current project.
- Watch for “quick fixes” that permanently alter support or maintenance obligations.

24. Verify Completion and Acceptance

An approved change should close only after its implementation is verified against the approved request. This prevents “approved version A, delivered version B” drift.

23. Confirm the implemented result matches the approved change.
24. Confirm related acceptance criteria and tests were updated.
25. Confirm affected documentation is current.
26. Confirm stakeholders accept the result where required.
27. Record closure date and evidence.

25. Version Control and Traceability

Traceability connects the original baseline, change request, decision, implementation artifact, test, and acceptance record. The goal is not paperwork for its own sake. It is the ability to reconstruct why the project is different today from what was approved earlier.

- Versioned scope or requirements
- Change ID linked to work items
- Decision record
- Updated schedule / baseline
- Test or acceptance evidence
- Release note or implementation record

AUDIT QUESTION

Can a reasonable reviewer answer: what changed, why, who approved it, what did it affect, and how do we know it was implemented?

PART VII

Preventing Recurrence and Recovering Control

26. The Behaviors That Create Scope Creep

Scope creep is usually a human-and-system interaction. The process may exist on paper while behavior routes around it.

- Avoiding difficult “not yet” conversations
- Fear of disappointing senior stakeholders
- Rewarding responsiveness more than disciplined delivery
- Treating every idea as immediately actionable
- Team members proving value by adding extras
- Leaders making requests in meetings without clarifying whether they are ideas or directives
- Project managers protecting relationships by absorbing change silently

The correction is not to become rigid. It is to normalize the sentence: “That may be valuable; let’s evaluate the change before committing.”

27. Client and Stakeholder Pressure

Stakeholders may experience change control as resistance if the process is slow, opaque, or framed as bureaucracy. A good process explains the tradeoff quickly and gives options.

BETTER CONVERSATION

“We can add this. The current estimate is three additional weeks because it reopens security review. Option A moves launch three weeks. Option B keeps launch and defers this to release two. Option C removes the lower-priority reporting feature to keep the date. Which tradeoff do you want to authorize?”

This shifts the conversation from “the project team refuses” to “the organization chooses among consequences.”

28. Gold Plating and Internal Scope Creep

Not all scope creep comes from clients. Gold plating occurs when the team adds features, polish, analysis, documentation, or technical sophistication that was not required or approved. The intent may be quality, innovation, or pride, but the project still pays the cost.

- Extra features added without a requirement
- Higher technical complexity than the outcome needs
- Additional reports or dashboards nobody requested
- Redesign work based on personal preference
- Expanded documentation beyond regulatory or operational need

QUALITY VS. GOLD PLATING

Meeting the agreed quality standard is required. Adding unapproved functionality or sophistication beyond that standard is a change, even when the team believes it is better.

29. Recovering a Project Already Overrun by Change

When a project is deep in scope creep, trying to reconstruct every historical conversation may be impossible. Recovery should establish a new truthful baseline.

28. Freeze new discretionary changes temporarily.
29. Inventory what the team is actually building or performing now.
30. Compare current work with the last approved baseline.
31. Classify differences: required correction, approved change, unapproved change, unknown.
32. Quantify schedule, cost, resource, quality, and risk impact.
33. Escalate unresolved scope decisions to the correct authority.
34. Approve a recovery baseline.
35. Restart change control from that point forward.

RECOVERY PRINCIPLE

Do not preserve a false baseline for the sake of appearing on plan. A truthful revised plan is more useful than an “on-time” schedule nobody believes.

30. Closing the Project Cleanly

Project closeout should reconcile change history so open requests do not quietly become post-project obligations.

- List all approved changes and confirm implementation.
- List deferred changes and identify where they were transferred.
- Close rejected changes so they do not remain ambiguous.
- Confirm no shadow work remains undocumented.
- Reconcile final scope, schedule, budget, and acceptance.
- Capture lessons about where change entered and which controls worked or failed.

FINAL CONTROL TEST

The project should end with a clear answer to three questions: What did we commit to? What did we actually deliver? What approved decisions explain the difference?

Six Workplace Cases

Case 1 — Government Portal: The Executive Comment

During a steering meeting, an executive says, “It would be useful if this also supported external partners.” The technical team interprets the comment as direction and begins redesigning authentication. Two weeks later, the sponsor is surprised by a schedule slip.

- Visible failure: schedule delay.
- Hidden control failure: an idea was treated as authorization.
- Better response: log the request, evaluate identity/security impact, and route to sponsor/governance approval.
- Lesson: status does not equal change authority.

Discussion question: At what point should the change-control cycle have intercepted this situation, and what artifact would have prevented ambiguity?

Case 2 — Healthcare Workflow: The “Tiny” Field

A clinical stakeholder asks for one additional required field in an intake workflow. The field affects database schema, reporting, training materials, interface mapping, and regulatory validation.

- Visible request: one field.
- Actual impact: multiple downstream artifacts and tests.
- Better response: evaluate the entire dependency chain before estimating.
- Lesson: size must be measured by system impact, not screen appearance.

Discussion question: At what point should the change-control cycle have intercepted this situation, and what artifact would have prevented ambiguity?

Case 3 — Technology Release: Ten Small Enhancements

A product team approves ten requests, each estimated at one day. The release slips three weeks because every change requires regression testing and two reopen UX review.

- Visible failure: estimates appear wrong.
- Hidden control failure: cumulative change load was never reviewed.
- Better response: track aggregate effort, retest burden, and contingency consumed.
- Lesson: individual approvals can create collective failure.

Discussion question: At what point should the change-control cycle have intercepted this situation, and what artifact would have prevented ambiguity?

Case 4 — Training Development: Helpful Gold Plating

An instructional designer adds interactive scenarios, extra downloadable guides, and a second quiz to improve quality. None were requested. The course misses its review date.

- Visible failure: late delivery.

- Hidden control failure: internal scope grew without approval.
- Better response: distinguish required quality from optional enhancement.
- Lesson: good intent does not make unapproved work free.

Discussion question: At what point should the change-control cycle have intercepted this situation, and what artifact would have prevented ambiguity?

Case 5 — Vendor Integration: Rejected but Implemented

A change request for a new data element is rejected for the current release. A developer later includes it because the field is "easy." The vendor begins relying on it.

- Visible failure: unsupported interface commitment.
- Hidden control failure: rejected work entered through implementation.
- Better response: link work items to approved change IDs and review backlog authorization.
- Lesson: rejection must be enforced in execution, not only documented.

Discussion question: At what point should the change-control cycle have intercepted this situation, and what artifact would have prevented ambiguity?

Case 6 — Program Recovery: Nobody Knows Current Scope

After six months of informal changes, the schedule, contract notes, backlog, and stakeholder expectations all disagree. The team spends more time arguing about commitments than completing work.

- Visible failure: confusion and delay.
- Hidden control failure: the baseline stopped being authoritative.
- Better response: temporary change freeze, scope inventory, gap classification, impact review, sponsor-approved recovery baseline.
- Lesson: a truthful reset is better than defending a fictional plan.

Discussion question: At what point should the change-control cycle have intercepted this situation, and what artifact would have prevented ambiguity?

Practical Change-Control Job Aids

These tools are designed for direct use on live projects. Use only the level of documentation appropriate to the project's risk, size, and contractual environment.

Tool 1 — Scope Boundary Sheet

Define the baseline before requests arrive.

Prompt	Working Response
Project objective	
In-scope outcomes	
Out-of-scope items	
Major deliverables	
Acceptance authority	
Key interfaces	

USE IT WELL

Use explicit exclusions for adjacent work stakeholders may otherwise assume is included.

Tool 2 — Change Recognition Checklist

Decide whether a request may alter the baseline.

Prompt	Working Response
Does the request add or remove an obligation?	
Does it change acceptance criteria?	
Does it require rework?	
Does it affect schedule, cost, resources, risk, or dependencies?	
Does it change an assumption or constraint?	
Classification: clarification / correction / defect / change / uncertain	

USE IT WELL

If uncertain, log it. Classification can change after evaluation.

Tool 3 — Change Request Form

Capture the request without committing to it.

Prompt	Working Response
Request ID / date	
Requester	

Affected baseline item	
Requested change	
Business reason	
Desired date	
Consequence if not approved	
Proposed solution, if any	
Evidence / attachments	

USE IT WELL

Separate the need from the proposed solution.

Tool 4 — Change Log

Maintain the project's decision memory.

Prompt	Working Response
Change ID	

Short title	
Requester	
Status	
Date submitted	
Decision	
Decision authority	
Decision date	
Owner	
Target / closure date	

USE IT WELL

Keep statuses simple and current.

Tool 5 — Eight-Dimension Impact Review

Evaluate the complete effect of a request.

Prompt	Working Response
Scope impact	
Schedule impact	
Cost impact	
Resource impact	
Quality impact	
Risk impact	
Dependency impact	
Benefit / value impact	

USE IT WELL

Do not reduce impact to estimated build hours.

Tool 6 — Schedule Impact Worksheet

Trace timing effects beyond direct effort.

Prompt	Working Response
New work duration	
Rework required	
Reviews reopened	
Dependencies affected	
Lead times	
Retesting	
Milestones moved	
Contingency consumed	

USE IT WELL

Ask what must move, wait, repeat, or happen earlier.

Tool 7 — Resource and Cost Impact Sheet

Make displacement and opportunity cost visible.

Prompt	Working Response
Direct labor	
Vendor / contractor cost	
Licensing / procurement	
Specialist capacity	
Overtime / surge needs	
Work displaced	
Long-term support cost	

USE IT WELL

Internal work is not free when capacity is finite.

Tool 8 — Risk and Quality Impact Sheet

Evaluate non-schedule consequences.

Prompt	Working Response

New risks	
Existing risks increased	
Compliance / privacy / security effect	
Quality criteria changed	
New tests required	
Maintainability effect	
Operational support effect	

USE IT WELL

A feature benefit can be real while risk also increases.

Tool 9 — Cumulative Change Dashboard

Show aggregate change load.

Prompt	Working Response
Approved changes count	

Open changes count	
Total added effort	
Schedule movement	
Contingency consumed	
New dependencies	
Rework attributable to change	

USE IT WELL

Set a threshold that triggers portfolio-level review.

Tool 10 — Change Authority Matrix

Define who can approve what.

Prompt	Working Response
Change category	
Project manager authority	

Sponsor authority	
Governance authority	
Dollar / schedule / risk threshold	
Contractual review required?	

USE IT WELL

Do this before the project is under pressure.

Tool 11 — Decision Record

Make approval traceable.

Prompt	Working Response
Change ID	
Decision: approve / reject / defer / split	
Decision-maker	
Date	

Rationale	
Conditions	
Effective date	
Follow-up action	

USE IT WELL

Avoid decisions that live only in email or memory.

Tool 12 — Emergency Change Record

Preserve governance under urgency.

Prompt	Working Response
Reason normal process cannot be used	
Emergency authority	
Immediate impact	
Decision	

Implementation owner	
Date/time	
Retrospective review date	
Baseline updates required	

USE IT WELL

Urgent is a faster path, not a no-documentation path.

Tool 13 — Rebaseline Checklist

Make the official plan match the approved work.

Prompt	Working Response
Scope updated	
Requirements updated	
Schedule updated	
Budget updated	

Resources updated	
Risks/dependencies updated	
Contracts/SOW updated	
Stakeholders notified	

USE IT WELL

One current answer to what the project is delivering.

Tool 14 — Implementation Traceability Sheet

Connect approval to delivered work.

Prompt	Working Response
Change ID	
Work item / ticket	
Owner	
Version / release	

Test evidence	
Acceptance evidence	
Closure date	

USE IT WELL

Useful for complex, contractual, regulated, or multi-team work.

Tool 15 — Rejected / Deferred Change Control

Prevent shadow implementation.

Prompt	Working Response
Change ID	
Decision	
Reason	
Backlog / future destination	
Current-project work removed?	

Affected stakeholders notified?	
Owner of future consideration	

USE IT WELL

Deferred does not mean “do it quietly if there is time.”

Tool 16 — Stakeholder Change Conversation Planner

Prepare a tradeoff-based response.

Prompt	Working Response
Requested outcome	
Impact summary	
Option A	
Option B	
Option C	
Recommended option	

Decision needed from stakeholder	

USE IT WELL

Give options and consequences rather than hiding behind process.

Tool 17 — Gold-Plating Audit

Identify internally generated scope growth.

Prompt	Working Response
Work item	
Approved requirement?	
Quality requirement?	
Change ID?	
Why team wants it	
Impact	
Keep / submit change / remove	

USE IT WELL

Apply the same discipline to “helpful” internal additions.

Tool 18 — Scope Recovery Inventory

Recover a project with conflicting versions of scope.

Prompt	Working Response
Current work being performed	
Last approved baseline item	
Difference	
Approved change?	
Required correction?	
Unknown / disputed?	
Decision needed	

USE IT WELL

Build a truthful inventory before trying to rebuild the plan.

Tool 19 — Change Freeze Plan

Create temporary control during recovery.

Prompt	Working Response
Freeze start	
What changes are blocked	
What emergency path remains open	
Who can authorize exceptions	
Backlog location	
Review date	
Conditions to lift freeze	

USE IT WELL

A freeze is temporary stabilization, not permanent rigidity.

Tool 20 — Project Change Closeout

Reconcile change history at project end.

Prompt	Working Response

Approved changes verified	
Rejected changes closed	
Deferred changes transferred	
Open changes resolved	
Final scope accepted	
Final baseline archived	
Lessons captured	

USE IT WELL

End with a clean record of why actual delivery differs from original plans.

APPLICATION

Scenario-Based Exercises

Exercise 1 — Clarification or Change?

Classify five requests: a wording correction, a new report, a defect against an approved requirement, a new security rule, and a stakeholder preference discovered during review. Explain what evidence you need before deciding.

Working notes:

Exercise 2 — Evaluate the “One-Day” Change

A request is estimated at one developer day. Map every review, dependency, test, documentation update, handoff, and approval it could trigger. Re-estimate schedule impact.

Working notes:

Exercise 3 — Build Approval Thresholds

Design low, medium, and high change-authority thresholds for a project you know. Include schedule, cost, risk, and contractual triggers.

Working notes:

Exercise 4 — Handle Executive Pressure

A senior leader asks for a change in a meeting and says, “Just make it happen.” Write a response that respects authority but preserves impact evaluation and traceability.

Working notes:

Exercise 5 — Cumulative Change Review

Assume eight approved changes have consumed 60% of schedule contingency and added two external dependencies. Decide whether individual change approval remains sufficient or broader governance review is required.

Working notes:

Exercise 6 — Recover Shadow Scope

A project contains work that is not traceable to requirements or approved changes. Create a process to classify, stop, approve, defer, or remove the work.

Working notes:

Exercise 7 — Rebaseline

Take an approved change and list every project artifact that must be updated so the baseline is truthful again.

Working notes:

Knowledge Check

1. Which statement best defines scope creep?

- A. Any change requested after kickoff
- B. Uncontrolled change to approved project obligations or boundaries
- C. Any schedule delay
- D. Any stakeholder disagreement

2. What makes a change “small”?

- A. The requester calls it small
- B. It takes less than one day to build
- C. Its evaluated effect on scope, schedule, cost, quality, risk, resources, and dependencies is small
- D. It does not require executive approval

3. Which is usually NOT a scope change?

- A. Adding a new report
- B. Correcting a defect so work meets an already-approved requirement
- C. Changing an acceptance criterion
- D. Adding an external user group

4. Why is a vague scope baseline dangerous?

- A. It makes the project look too simple
- B. It allows later disagreements to be treated as clarification because boundaries are not testable
- C. It prevents meetings
- D. It increases vendor cost automatically

5. What is the safest response to an informal request?

- A. Start the work and document later
- B. Reject it immediately
- C. Capture it for impact evaluation before commitment
- D. Ask the team to vote

6. What is the primary purpose of a change log?

- A. Replace the project schedule
- B. Record request, status, decision, authority, implementation, and closure
- C. Track employee performance
- D. Store meeting minutes

7. Which item belongs in an impact review?

- A. Only build hours
- B. Scope, schedule, cost, resources, quality, risk, dependencies, and benefits

- C. Requester seniority only
- D. Team preference

8. Why can ten one-day changes delay a project by more than ten days?

- A. Teams always estimate badly
- B. Changes can cause interruption, rework, review, retesting, and dependency effects
- C. Managers dislike change
- D. One-day changes are never allowed

9. If resources are fixed, approving new work also means:

- A. Nothing else changes
- B. Some other work may need to move, shrink, or stop
- C. The manager works overtime
- D. Acceptance criteria disappear

10. What is cumulative change load?

- A. The latest approved change
- B. The combined burden of all changes on project capacity and commitments
- C. The number of stakeholders
- D. The change budget only

11. What should approval thresholds accomplish?

- A. Send every decision to executives
- B. Match decision authority to impact and risk
- C. Eliminate documentation
- D. Keep stakeholders from requesting changes

12. What does “defer” mean?

- A. Quietly implement if time allows
- B. Reject permanently
- C. Move consideration or implementation to a later phase/release/backlog
- D. Approve without funding

13. How should emergency changes be handled?

- A. With no documentation
- B. Through an accelerated path with authority, minimum impact review, and retrospective update
- C. Exactly like routine changes regardless of urgency
- D. By any employee available

14. Why must the project be rebaselined after approval?

- A. To make the official plan reflect the work the organization actually authorized
- B. To create more paperwork
- C. To remove the change log
- D. To hide the original plan

15. What is shadow implementation?

- A. Testing at night
- B. Rejected, deferred, or unapproved work entering execution anyway
- C. Emergency change control
- D. Vendor work outside office hours

16. What is traceability meant to answer?

- A. Who worked the most hours
- B. What changed, why, who approved it, what it affected, and how implementation was verified
- C. Which stakeholder complained
- D. How many meetings occurred

17. What is gold plating?

- A. Meeting required quality standards
- B. Team-added unapproved functionality or sophistication beyond required scope
- C. Approved enhancement work
- D. Vendor profit

18. What is the first useful step when a project is already overrun by informal change?

- A. Keep adding changes until the next milestone
- B. Temporarily stabilize and inventory actual work against the last approved baseline
- C. Delete the change log
- D. Replace the whole team

19. Which question best tests whether closeout is clean?

- A. Did everyone enjoy the project?
- B. Can we explain the difference between original commitments and final delivery through approved decisions?
- C. Did we avoid all changes?
- D. Did the project manager attend every meeting?

20. What is the manual's central principle?

- A. Never change scope
- B. A request is small if it sounds small
- C. Evaluate the real impact of change before commitment, then keep the baseline truthful
- D. Only executives can request changes

Answer Key

1. B — Scope creep is uncontrolled change; controlled change can be large and still be well governed.

2. C — The true size of a request comes from evaluated impact, not wording or direct build time.

3. B — Bringing defective work back into conformance with an approved requirement does not automatically add scope.

4. B — If scope cannot be tested, later requests can masquerade as clarification.

5. C — Capture and evaluate first; this protects both the requester and the project.

6. B — The log preserves the decision trail and implementation status.

7. B — Impact is multidimensional; effort alone is insufficient.

8. B — Changes create secondary work and sequencing effects.

9. B — Finite capacity makes every approval a prioritization choice.

10. B — Aggregate change burden can exhaust contingency even when each individual request looked manageable.

11. B — Authority should scale to the consequence of the decision.

12. C — Deferred work is intentionally moved outside the current commitment.

13. B — Urgency accelerates governance but does not erase authority or traceability.

14. A — An approved change that is not reflected in the baseline leaves the team executing conflicting plans.

15. B — Formal rejection is meaningless if the work still enters the product or service.

16. B — Traceability makes project evolution reconstructable.

17. B — Gold plating is internally generated scope growth beyond the approved obligation.

18. B — Recovery begins by making actual work and the last approved plan comparable.

19. B — Clean closeout reconciles planned and actual delivery through documented decisions.

20. C — Change control is about deliberate tradeoffs and a truthful baseline, not resisting all change.

The Change-Control Standard

Scope control is not the art of saying no. It is the discipline of ensuring that every yes has a visible consequence, an authorized decision, and a plan that reflects the new reality.

A healthy project can change significantly and still remain controlled. An unhealthy project can drift through dozens of tiny requests while insisting nothing important changed. The difference is not the amount of change. The difference is whether the organization can see, evaluate, authorize, and trace it.

> *Recognize the change. Capture it. Measure the tradeoff. Put the decision in the right hands. Then make the plan tell the truth.*