PH7X SYSTEMS®

0.0.0 · CHAPTER 4

Making and defending a recommendation

SharePoint Migration Field Guide · Chapter 4

pH7x Systems

2026-08-05

4. Making and defending a recommendation

> A recommendation should survive a product update. A feature comparison does not.

Chapter 3 ended with a list of destinations that survived your hard constraints. This chapter turns that list into a recommendation somebody else can read, question, and act on eighteen months from now.

That last clause is the whole difficulty. Recommendations are often defensible on the day they are made, in the room where they are made, to the people who were in it. Fewer survive the departure of the person who made them.

A recommendation should survive a product update

The usual way to justify a destination is a comparison table. Two columns, a list of capabilities, ticks down one side. It is easy to build and it is the wrong instrument, for a reason that has nothing to do with accuracy.

A comparison table is a photograph of two products on one day. Products change. Half the rows that decided your recommendation will read differently in a year, and when they do, nobody can tell whether the recommendation still holds, because the argument was never about the products. It was about your estate.

A recommendation built on constraints ages differently. "SharePoint Online is eliminated because two full-trust solutions are still required" stays true until somebody rewrites them. When that happens, the recommendation does not quietly rot. It becomes wrong in a visible, dateable way, and somebody reopens it on purpose.

That is the test to apply to every sentence you are about to write: **if this stopped being true, would anyone notice?**

Start with eliminated paths

The first half of a recommendation is not the destination you chose. It is the destinations you cannot have, and why.

This feels backwards to most people writing one. It reads as negative, and there is pressure to open with the answer. Resist it, because the eliminations carry the evidence and the answer does not. A reader who accepts your eliminations has already accepted most of your recommendation. A reader who is only shown the answer has been given a conclusion and asked to trust it.

Write each one in the form chapter 3 asked for: the option, the fact that removed it, and the date. Not "Subscription Edition: rejected". Something closer to:

```
SharePoint Online          eliminated
  by: two full-trust solutions still required by the finance sites
  cost to remove: quoted at N days, not funded this year
  recorded: 2026-08-05
```

Dates in the record use ISO 8601, `YYYY-MM-DD`, so that they sort correctly and stay unambiguous for readers who write dates differently from you.

An eliminated path with a cost attached is not a closed door. It is a priced one, and pricing it is what lets somebody reopen the decision later without repeating the analysis.

Constraints change at different speeds

Chapter 3 sorted constraints by weight: what removes an option, and what only changes the price. That question is answered and this chapter does not reopen it.

Here the axis is different, and it is the one that decides how long your recommendation lasts:

> **Who can change this constraint, and how quickly?**

CONSTRAINT	WHO CAN CHANGE IT	HOW QUICKLY
Full-trust solution that must keep running	Your organisation, through development work	Months, and it needs funding
Internal preference for a platform	Whoever holds the preference	A conversation, if the right person is in it
No budget in the current year	Finance, at a known moment	The next budget cycle, and the date is knowable
Data residency obligation in law	Nobody in the project	Legislative timescales, or never
Dependency with no supported equivalent	A vendor, on their roadmap	Unknown, and the honest answer is that you do not know

Read that table next to your own constraint list and something useful falls out. A recommendation resting entirely on the bottom two rows is stable, and you can say so. A recommendation resting on the third row has an expiry date somebody can look up. **We recommend writing that expiry date into the document**, because a recommendation that carries its own review date is far harder to leave rotting in a shared folder.

The row that causes the most argument is the second. An internal preference is a real constraint, in the sense that it will shape what happens, and it is not evidence. It belongs in the record, named as a preference, with the name of whoever holds it. Recording it is not a criticism. It is what stops it being smuggled in later as though it had been a technical finding.

When more than one destination remains

Sometimes the hard constraints leave exactly one destination standing. Those recommendations write themselves, and they are not the interesting case.

Sometimes two survive. That is the moment where most recommendations go quietly wrong, because the writer already has a preference and now has to justify it, and the easiest way to justify it is a feature table.

Here is the rule this chapter asks you to follow. **Until this point, no opinion has entered.** Evidence produced facts, hard constraints removed options, and none of that required a view. Now, with two destinations open, judgement enters, and it enters once, in the open.

What discriminates is the soft constraints. In chapter 3 they only changed the price. Here, with the field already narrowed, price is the question, and they get a vote:

- how much of the estate needs rebuilding at each destination;
- how much of that work sits on the critical path rather than beside it;
- how many other teams have to agree, and on whose release schedule;
- what happens to each option if the project slips by a quarter.

Answer those for both survivors, in writing, and the comparison you produce is about your estate rather than about two products. That is a comparison that still means something next year.

If after all of it the two are genuinely close, say so. "Both remain viable; we recommend the first for these three reasons, and the second would not be a mistake" is a stronger document than a manufactured margin. A recommendation that admits a near tie is easier to examine, because the reader can see that the uncertainty was not hidden.

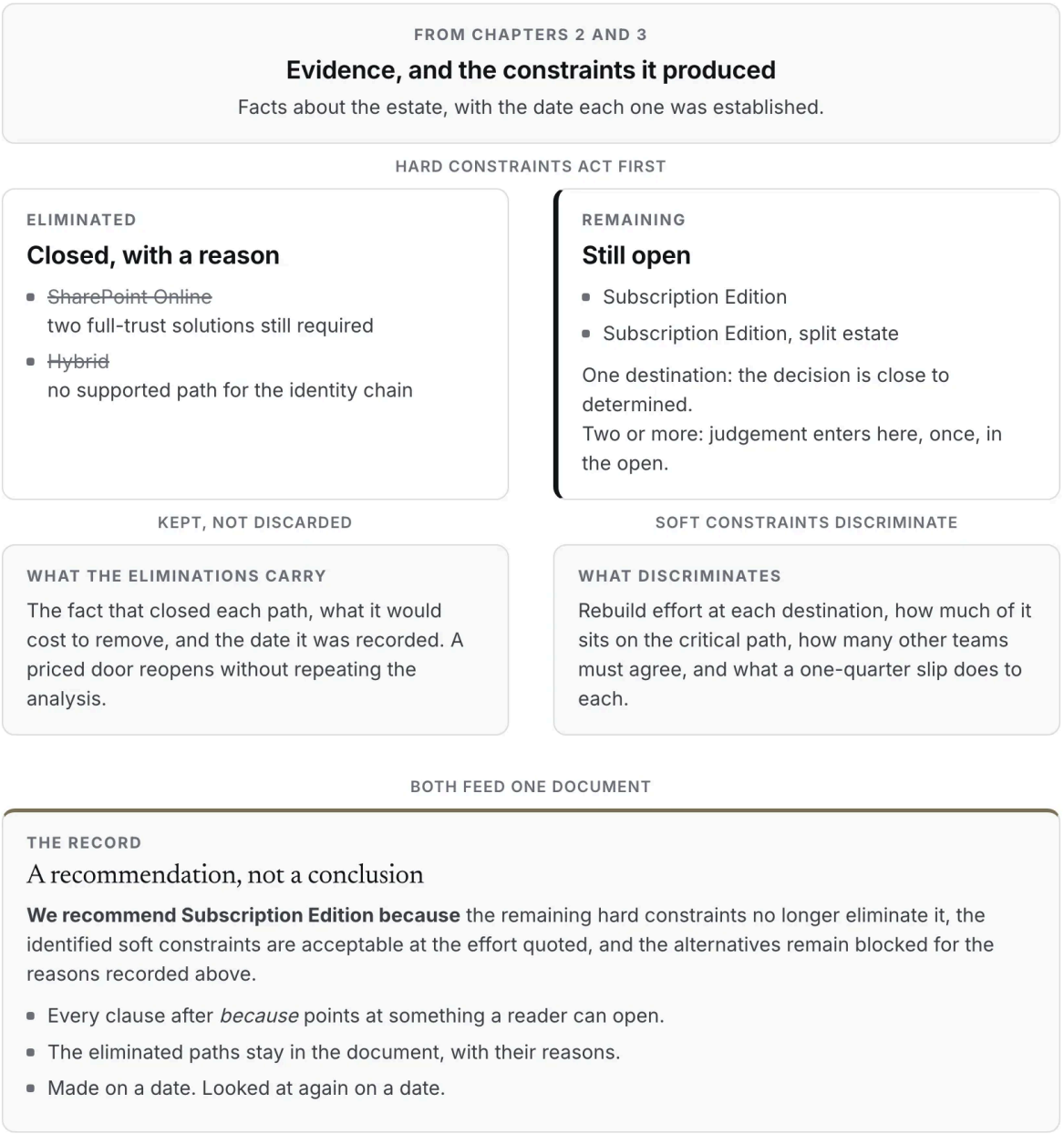

Figure 4.1. A recommendation is a record, not a conclusion. The eliminated paths stay in it.

Writing a recommendation that survives scrutiny

Compare two sentences.

> We recommend SharePoint Online because it offers better collaboration features and a lower administrative burden.

> We recommend SharePoint Online because the remaining hard constraints no longer eliminate it, the identified soft constraints are acceptable at the effort we have quoted, and the alternatives remain blocked for documented reasons.

The first is not false. It is unfalsifiable, which is worse. Nobody can check it, nobody can date it, and nobody can tell you a year from now whether it still holds.

The second points at four artefacts: your constraint list, your elimination record, your effort estimate and your review date. Every clause after

because

is a reference to something a reader can open and disagree with.

That is the shape to aim for. If a clause in your recommendation cannot be traced to a documented constraint, it is a preference, and it belongs in the preference section under somebody's name.

When a recommendation fails

Recommendations rarely fail because the destination was wrong. They fail because nobody can reconstruct why it was chosen. Four common failure modes:

It cites features instead of constraints. The product changes and the argument evaporates. Nobody notices, because there was no mechanism for noticing.

It records only the chosen path. Twelve months later somebody asks why the other option was not taken, and the honest answer is that nobody remembers. The debate restarts from nothing, and it restarts every year.

It carries no date. Without a date, a recommendation is timeless, and a timeless recommendation is either permanently valid or permanently suspect. It gets treated as the second.

It cannot be reconstructed by a stranger. The person who made it has left, and what remains is a slide with a conclusion. The conclusion may well be right. It cannot be defended, so it gets remade.

All four have the same fix, which is why this chapter has laboured the point: the recommendation is not the sentence. It is the sentence plus the eliminations plus the dates, kept together, in the project, as a document.

Out of scope

The cost model behind the recommendation. Turning effort into money, and money into an approved budget, is chapter 5, building the business case. This chapter produces the input to that work; it does not make the case itself.

Governance decisions at the destination. How the target environment should operate is chapter 6, governance before technology. Mixing it in here is how a destination decision turns into a redesign nobody approved.

How to migrate, and with what. Choosing migration tooling is chapter 29.

Whether the recommendation gets accepted. Approval depends on sponsorship, budget, risk appetite and organisational authority, and this chapter decides none of them. What it can do is make acceptance or rejection a conscious decision rather than the outcome of an argument nobody could reconstruct.

Before you continue

- [ ] Your recommendation names the destinations you eliminated, and the fact that eliminated each one
- [ ] Every clause after "because" points at a constraint you can show, not a feature
- [ ] Each constraint in the record says who can change it and how quickly
- [ ] Where two destinations remained, the soft constraints that discriminated between them are written down
- [ ] Any preference in the document is named as a preference, with the name of whoever holds it
- [ ] The document carries the date it was made and the date it should be looked at again
- [ ] Somebody who was not in the room could reconstruct the reasoning from what you wrote