Procurement Notes

SINCE 2025

Breaking Free from IT License Lock-in: A Strategic Guide for Procurement Professionals

Why software vendors become hard to leave, what that costs, and how procurement can keep exit options open before and after signing.

By **Pierluigi Peggiani**

PN-2025-03 18 SEPTEMBER 2025 5 MIN READ

Organizations now run hundreds of SaaS applications and devote a large share of their IT budgets to software licensing. Hidden inside those purchases is a major strategic risk: license lock-in. This is what happens when moving away from a vendor becomes so hard, costly, or risky that staying put feels inevitable—even when prices rise or service declines. What begins as a sensible technology choice can turn into a long, expensive obligation that slows innovation and limits options.

Lock-in shows up in several ways. Sometimes the technology itself makes leaving difficult: data sits in proprietary formats, integrations depend on vendor-specific APIs, and even simple changes ripple through the stack. Sometimes the economics do the trapping: by the time you add up migration work, retraining, process redesign, and lost productivity, the switching bill can equal two to four years of license fees. Contracts can tighten the grip with auto-renewals, narrow termination windows, and steep penalties. And when many products from the same vendor are linked together—ERP, CRM, analytics, collaboration, security—the dependency becomes an ecosystem. Replacing one piece means touching many.

The path into lock-in is often gradual. A "simple" system is rolled out and, over a few years, grows into a web of custom workflows, automations, and integrations tied to that vendor's way of doing things. Teams are trained, reports are built, and new projects stack on top of old ones. By the time prices jump or a better option appears, the cost and disruption of switching can look overwhelming.

Real-world stories make the risk clear. A global manufacturer customized an ERP across dozens of systems and thousands of users. When the vendor raised prices by 40 percent, the company discovered that leaving would require more than two years of work, heavy consulting spend, lost productivity, and rebuilding critical automations—roughly a fifty-million-dollar problem. In the cloud, organizations find that data "egress" fees, proprietary databases, and vendor-specific AI or event services turn migrations into long rebuilds. One enterprise estimated tens of millions to exit a single provider and re-platform most of its e-commerce. Database choices can have the same effect: vendor-specific SQL and stored procedures force major code rewrites and extensive retesting. Even office productivity suites can create friction through macros, templates, and workflows embedded across the business, making a switch a multi-month, multi-million-dollar effort. A fast-growing fintech aiming for multi-cloud resilience learned that most of its services relied on proprietary features; achieving portability required large code changes, parallel operations, and significant cost and time.

Once lock-in takes hold, the business impact spreads. Financially, prices tend to rise faster for trapped customers than for those with credible alternatives. Negotiating leverage drops because the threat of leaving is no longer believable, and hidden costs—paid upgrades, compatibility work, professional services—accumulate beyond base licenses. Operationally, innovation slows. Integrating new tools or capabilities that competitors adopt quickly can take months and cost millions. Support quality can slip as well, with longer response times and less context on your environment. Strategically, rigidity becomes expensive: regulatory changes, M&A integration, or new customer expectations turn into long projects instead of quick adaptations. Risk also increases. A vendor acquisition or product sunset can force emergency migrations. Compliance and security may lag your needs if the vendor's roadmap is slow. And running older technology for too long creates security gaps, efficiency losses, and hiring challenges.

The best place to address lock-in is before you buy. Start by looking at total cost of ownership over seven to ten years, including migration work, training, productivity impacts, and plausible price-increase scenarios. Don't stop at list price; model what happens at years three, five, and seven under different dependency levels. Evaluate lock-in risk explicitly: test real data exports, try the APIs with your own engineers, and verify claims of open standards with hands-on proof. Score vendors on data portability, API openness, standards compliance, customization reversibility, and ecosystem dependencies. Keep multiple credible vendors in play until signature; visible competition reliably improves terms and portability commitments.

Contract terms should preserve your exit rights. Spell out data portability in detail: complete exports in common formats, all metadata and relationships, audit trails, integrity checks, and a period of migration support at no extra cost, backed by meaningful penalties for non-performance. Prefer shorter initial terms with performance-based renewals and caps on annual increases. Aim for licensing models that flex with your business—quarterly true-ups, modular entitlements, geographic flexibility, hardship provisions, and automatic access to new features offered to peers at similar tiers.

Technical diligence matters just as much. Favor open standards you can validate without the vendor's proprietary tools. Review API documentation for authentication, rate limits, error handling, versioning, and backward compatibility, and test it with developers who haven't used the platform before. Design integrations with abstraction layers, middleware, and adapter patterns so that vendor-specific elements are isolated. The goal is to make switching a project, not a crisis.

If you are already locked in, you still have options. Treat it as a program, not a one-off negotiation. Start by mapping dependencies and targeting changes that reduce risk without disrupting core operations: reporting layers, user interfaces, and integration hubs are common early wins. Build hybrid architectures that let new capabilities run on alternative vendors while the core remains stable; the short-term cost increase buys future flexibility. Pilot alternatives in low-risk areas to prove technical feasibility, measure adoption, and build internal skills. In parallel, manage the relationship with structure and data. Schedule regular reviews, share your dependency analysis and competitive benchmarks, document commitments, and use user groups and peer networks to amplify requests for portability and fair terms. Track vendor behavior—price patterns, support quality, roadmap delivery—and keep a current view of alternatives and migration estimates so you can move when conditions change.

Over the long term, diversify where it makes sense. Maintain multi-vendor strategies for key domains, choose best-of-breed when requirements allow, and adopt multi-cloud approaches where feasible. Invest in internal capabilities to reduce reliance on vendor experts, keep thorough documentation of configurations and customizations, and strengthen change-management so transitions are predictable. Stay informed about market shifts, new entrants, and emerging technologies that offer different paths to the same outcomes, and compare notes with peers to avoid learning the hard way.

The landscape is still evolving. Cloud and SaaS make portability both easier and harder: infrastructure is more elastic, but data gravity, workflow ties, and API dependencies can be strong. Regulators are watching market concentration and may push for clearer data-portability rights. Mature open-source options are expanding the toolkit, though support and total cost need honest evaluation.

In the end, license lock-in is less a technical problem than a strategic one. It inflates costs, slows innovation, and narrows your choices—often without obvious warning signs until the commitment is deep. Guard against it by treating portability as a requirement, building credible alternatives into your plans, and designing architectures and contracts that preserve your ability to change course. If you are already entangled, reduce risk in stages while rebuilding leverage. Flexibility isn't a luxury; it is the foundation for sustainable cost control and timely innovation.

PP

ABOUT THE AUTHOR

Pierluigi Peggiani is a procurement and supply chain professional who has held strategic positions across Europe, the Middle East and Asia. His writing focuses on how organisations source and buy: selecting suppliers, shaping contracts, managing day-to-day purchasing and keeping materials flowing through the supply chain.

CONTACT pierluigi@procurementnotes.com