# Billing RFP criteria & requirements guide

*An internal planning tool for evaluating billing systems*

## Document purpose

This document helps internal teams define and document their requirements for a billing system. It outlines common billing use cases, decision criteria and technical considerations often overlooked or underspecified. It is intended to aid in vendor evaluations with a specific focus on avoiding system limitations that can introduce unnecessary manual effort, delays, reconciliation issues or operational costs.

## Scope of this document

This document includes requirements relevant to the following billing models and scenarios:

* Recurring fixed‑fee subscriptions
* Usage‑based pricing (including tiered, volume and pooled models)
* One‑time and milestone billing
* Hybrid billing (combinations of the above)
* Contract amendments and renewals
* Revenue recognition mapping and automation
* Integration with CRM, CPQ and external systems

It does not define implementation methodologies, resourcing needs or project timelines. Those considerations should be addressed separately once a preferred solution is selected.

## Intended audience

* Finance and accounting leadership
* Revenue operations and quote‑to‑cash process owners
* IT or systems teams responsible for ERP and billing integration
* Procurement or vendor management stakeholders
* External consultants supporting technology selection or implementation

## How to use this document

The content is organized by billing capability area. Each section includes:

* A brief section overview highlighting the capability, billing scenario or process
* Functional requirements written in plain, operational language
* Considerations that may require further validation during vendor evaluation
* An open field for a vendor response

This document can be used as a standalone business requirements guide or incorporated into a formal RFP process. Any topics or sections that are not applicable may be deleted or left blank. Note that vendor names and commercial language have been deliberately excluded.

## Project information

* Prepared by [Insert team or department]
* Project name [e.g. Billing Transformation, QTC Modernization]
* Date [Insert date]
* Document version [e.g. v1.0]
* Stakeholders [Enter names and titles here]
* Target implementation [e.g. Quarter and year]

# Section 1: General requirements

There are several foundational system capabilities that affect how well a billing solution integrates into existing systems and workflows. The following requirements cover architectural fit, global support, user governance and the ability to adapt to change. Gaps here tend to create long‑term friction for IT, finance and operations.

| Ref | Requirement | Considerations | Vendor response |
| --- | --- | --- | --- |
| 1.1 | Runs natively inside the ERP, not externally integrated | Solutions that operate outside the ERP require middleware, sync jobs or external reconciliation, increasing IT cost, latency and the risk of data mismatches between systems. |  |
| 1.2 | Supports multiple subsidiaries, legal entities and business units | Systems that can’t manage intercompany billing or multi‑entity consolidation force teams to handle these workflows manually or duplicate customer records across orgs. |  |
| 1.3 | Supports multi‑currency billing, pricing and revenue recognition | Without native currency handling, finance must manage FX conversion outside the system or rely on bolt‑ons, adding delay and increasing reconciliation effort. |  |
| 1.4 | Offers role‑based permissions, audit logs and approval controls | If access, changes and approvals aren’t tracked at a granular level, it creates compliance risk and limits internal control over contract and billing data. |  |
| 1.5 | Integrates with CRM, CPQ, usage sources, tax and payment platforms | Systems with limited APIs or rigid data models may require custom integrations or external ETL pipelines – increasing time to value and introducing fragility. |  |
| 1.6 | Supports future changes to pricing, billing or revenue models without reimplementation | Platforms that tie logic to static product catalog definitions or hardcoded billing rules can’t adapt easily to new monetization strategies – requiring costly rework. |  |
| 1.7 | Provides a single system of record for invoice generation and lifecycle management | Handling invoices, adjustments, payments and cancellations in multiple systems (the ERP, CRM and billing system) creates duplicate records, closing delays, audit risks and reporting issues. |  |

# Section 2: Subscription and contract management

Recurring contracts often include mid-term changes, custom billing cadences and renewal logic that varies by customer. Billing systems must adapt to this complexity without fragmenting contract data or introducing manual steps that compromise lifecycle traceability.

| Ref | Requirement | Considerations | Vendor response |
| --- | --- | --- | --- |
| 2.1 | Defines subscription‑level terms including start date, end date, billing frequency and minimum commitment | If these terms are only defined at the charge level, teams must manually reconstruct contract timelines from individual billing components. This increases the risk of mismatched renewal dates, inaccurate minimum commitment calculations and inconsistent billing cadences across a single agreement. |  |
| 2.2 | Supports multiple charge lines within a single subscription, each with distinct billing schedules | When separate billing cadences require multiple subscription records, contract data becomes fragmented. Teams must merge invoices manually, reconcile billing terms across records and rebuild contract‑level views for reporting and revenue recognition. |  |
| 2.3 | Allows subscription modifications mid‑term (e.g. price, quantity, product) without cancelling and re‑creating the record | Cancel/rebook logic creates disconnected contract records, which require manual intervention to carry forward committed terms, realign revenue schedules and preserve historical billing context. |  |
| 2.4 | Applies effective dates for subscription changes that differ from approval or billing dates | Systems that lack effective dating force billing teams to issue off‑cycle credits and rebills or manage timing changes in spreadsheets – introducing delays, audit complications and the risk of billing customers incorrectly. |  |
| 2.5 | Maintains a full version history of subscription changes, including all amendments | When subscription changes overwrite prior data or aren’t linked to the original record, it becomes difficult to justify billing outcomes, respond to contract disputes or meet audit requirements for change traceability. |  |
| 2.6 | Supports renewal terms that differ from original contract terms (e.g. evergreen renewals after fixed initial term) | If renewal logic defaults to copying prior terms, teams must manually reconfigure records to reflect new structures. This slows down the renewal process and increases the risk of billing on outdated terms. |  |
| 2.7 | Automates creation of renewal records or renewal opportunities | Without automation, renewals depend on CRM workflows or manual calendar tracking. This can lead to lapsed contracts, revenue leakage or inconsistent renewal terms across customers. |  |
| 2.8 | Links subscription amendments to the original contract to preserve lifecycle traceability | When changes are tracked in standalone records without system linkage, contract history must be pieced together manually. This limits visibility for finance, increases audit effort and makes contract evolution hard to explain. |  |
| 2.9 | Supports policy‑driven contract ramps without cloning subscriptions | Without native ramp logic, finance teams must split a single multi‑year contract into multiple subscriptions or rely on manual change orders for each scheduled increase. This fragments data, complicates reporting on TCV and ARR, creates compliance risk in ARM and increases the chance of billing errors or missed uplifts. |  |
| 2.10 | Centralizes renewal uplift policies, including index-based, percentage and flat-dollar rules | If uplift policies aren’t automated and centralized, teams are forced to track and apply increases manually at renewal. This leads to revenue leakage, contract non‑compliance, and inconsistent pricing across customers. A key requirement here is the ability to reference authoritative CPI or index data sources. |  |

# Section 3: Usage‑based and complex billing

Usage billing introduces complexity around data ingestion, real-time rating, prepaid management and pooled consumption. Systems must handle these differences without forcing manual reconciliation, data flattening or compliance workarounds.

| Ref | Requirement | Considerations | Vendor response |
| --- | --- | --- | --- |
| 3.1 | Supports multiple usage sources per customer or subscription (e.g. time tracking, metering systems, external platforms) | If usage ingestion is limited to a single source or format, teams must manually consolidate usage data before upload, adding reconciliation effort and delaying billing cycles. |  |
| 3.2 | Accepts usage data in flexible formats and granularity (e.g. by hour, day, week, custom interval) | Rigid data input requirements increase preprocessing work and create risk of usage mismatches between the source system and billing platform. |  |
| 3.3 | Applies usage aggregation rules within the billing platform (e.g. sum, max, average, count) | If usage must be pre‑aggregated externally, the system can’t support real‑time rating or re‑rating – and finance must rely on non‑auditable transformation pipelines. |  |
| 3.4 | Supports usage‑based pricing models including tiered, volume, block, flat‑rate and hybrid structures | If pricing logic is limited or hardcoded, teams may need to configure workarounds for common models – increasing maintenance effort and the chance of misbilled usage. |  |
| 3.5 | Allows usage‑based charges and fixed‑fee charges to coexist on a single subscription or invoice | If the system requires separate subscriptions or billing runs for usage and fixed fees, teams must merge invoices manually and maintain duplicate contract records. |  |
| 3.6 | Enables per‑customer pricing logic for usage charges | Systems that require global usage pricing or SKU‑level definitions make it difficult to apply customer‑specific contracts, reducing flexibility and increasing reliance on exception handling. |  |
| 3.7 | Varies billing frequency by charge line within a single subscription (e.g. monthly fixed + quarterly usage) | If billing cadence is tied to the subscription record or contract frequency, teams can’t support hybrid billing scenarios, leading to off‑cycle invoices or misaligned revenue schedules. |  |
| 3.8 | Handles usage corrections and adjustments after a charge has been rated or billed | Without correction capabilities, teams must issue manual credits and re‑rates outside the system – increasing the risk of errors, disputes or audit complications. |  |
| 3.9 | Provides in‑platform usage mediation from any NetSuite record | If usage ingestion is limited to a single record type, teams must build ETL pipelines or prep data in spreadsheets, increasing reconciliation effort and delaying billing cycles. |  |
| 3.10 | Supports advanced transformations and windowed calculations before rating | If transformations like unit conversions or percentile logic must happen outside the ERP, billing teams rely on spreadsheets that lack audit trails and increase the risk of misbilled usage. |  |
| 3.11 | Offers prepaid, drawdown and minimum‑commit usage with carryforward rules | If prepaid usage defaults to straight-line recognition, finance must manually adjust revenue schedules to stay compliant with ASC 606 when consumption timing varies. |  |
| 3.13 | Enforces billing maximums or charge caps at the contract or line level | If charge caps aren’t enforced automatically, finance must monitor thresholds manually and issue credits when limits are exceeded. |  |
| 3.14 | Automatically tops-off prepaid credits or units when balances reach a defined threshold | If top-offs aren’t automated, teams must monitor balances manually and intervene before prepaid credits run out, increasing workload and risking service interruptions. |  |

# Section 4: Billing, invoicing and consolidation

Invoices must consolidate usage and subscription charges clearly, and support downstream processes like cash application and collections. When these functions are siloed from billing, finance teams face delays, manual effort and reconciliation gaps that slow cash flow.

| Ref | Requirement | Considerations | Vendor response |
| --- | --- | --- | --- |
| 4.1 | Generates a single invoice across multiple subscriptions or services for a customer | If each subscription generates its own invoice, customers receive multiple fragmented bills – increasing the chance of missed payments, confusion and billing disputes. |  |
| 4.2 | Consolidates invoices across entities or business units where appropriate (e.g. global customers) | When billing is restricted to a 1:1 relationship between customer and entity, teams must manually combine or reissue invoices to meet enterprise customer expectations. |  |
| 4.3 | Groups or formats invoice lines by configurable logic (e.g. product family, business unit, region) | If invoice formatting is rigid, finance teams can’t present billing in a customer‑friendly format – reducing clarity and increasing the chance of billing inquiries. |  |
| 4.4 | Includes usage‑based and fixed‑fee charges on the same invoice | Systems that require usage and fixed billing to be handled separately result in partial invoices, which increases reconciliation work and can slow collections. |  |
| 4.5 | Defines billing responsibilities and payment terms per charge or subscription line | When systems only allow a single bill‑to and payment term per customer, it becomes difficult to manage contracts with split payment responsibilities or shared services. Support multiple billing contacts per account or contract and allocate specific line items to different legal entities with separate invoicing. |  |
| 4.6 | Supports pooled usage across multiple users or accounts under a single customer or contract | Without pooled usage capabilities, customers with multiple users or locations must be billed separately, increasing invoice volume and misaligning with contract expectations. |  |
| 4.7 | Configures invoice timing independently of subscription start date or renewal cycle | If billing schedules are tied directly to subscription terms, teams must manipulate contract data to align with preferred invoice timing, which creates audit and reporting complications. |  |
| 4.8 | Previews invoices prior to issuing, with full charge‑level and tax detail | Lack of preview functionality increases the risk of errors reaching the customer, requiring credit/rebill cycles and delaying payment. |  |
| 4.9 | Manages cash application, dunning and collections using ERP-native tools with visibility to billing and usage | If collections aren’t managed on ERP-native records, teams must toggle between systems to reference invoice, payment and usage data, delaying follow-ups and increasing DSO. |  |
| 4.10 | Provides real-time visibility into invoice status, payments and customer balances within the ERP | If invoice and payment status aren’t visible in real time within the ERP, finance must reconcile data manually across systems, slowing close cycles and increasing reporting risk. |  |
| 4.11 | Supports distinct Bill To and Ship To addresses at both the contract and line-item level | If billing and fulfillment addresses cannot be managed separately, teams must create duplicate contracts or override records manually, increasing operational effort and risking errors in invoicing and delivery. |  |

# Section 5: Revenue recognition

Revenue must be recognized based on consumption, not invoice timing – especially in usage-based models. Billing systems must handle these distinctions automatically to avoid compliance issues, misstated revenue and manual accounting adjustments.

| Ref | Requirement | Considerations | Vendor response |
| --- | --- | --- | --- |
| 5.1 | Support ASC 606 or IFRS 15 revenue recognition out of the box | Without built-in compliance support, finance teams must build custom revenue recognition rules, increasing audit risk and creating long-term dependency on technical resources |  |
| 5.2 | Create many-to-one or one-to-many mappings between billing elements and revenue elements | When systems enforce a 1:1 relationship between charge lines and revenue schedules, revenue cannot be allocated accurately, especially for bundled contracts or ratable services |  |
| 5.3 | Recognize revenue based on usage, delivery or event-based triggers | Rigid revenue recognition rules tied only to fixed schedules prevent accurate recognition for milestone billing, variable usage or deferred revenue events |  |
| 5.4 | Realign revenue schedules automatically when contract terms change (e.g. quantity, pricing, duration) | Without this capability, teams must manually adjust revenue schedules every time a contract is amended, introducing reconciliation burden and audit complexity |  |
| 5.5 | Support revenue allocation across merged or amended contracts | If amended contracts create new revenue arrangements without linking to prior terms, teams must manually track cumulative revenue performance obligations |  |
| 5.6 | Tie revenue schedules to subscription or contract records, not just sales orders | Revenue arrangements created at the transaction level lose visibility into contract intent, making it harder to support complex amendments or provide audit traceability |  |
| 5.7 | Preview revenue schedules prior to posting | Without visibility into upcoming rev schedules, finance cannot proactively detect errors, anticipate revenue shifts or ensure compliance before month-end close |  |
| 5.8 | Align usage revenue to consumption periods, not invoice timing | If usage revenue is tied to invoice date instead of consumption period, finance must manually adjust revenue schedules to stay compliant with ASC 606. |  |
| 5.9 | Manage unbilled receivables automatically for variable billing | If unbilled receivables aren’t managed automatically, finance must calculate and post journal entries manually to reconcile earned vs. billed revenue. |  |

# Section 6: Integration and workflow automation

When billing logic is fragmented across CRM, ERP and external platforms, teams face integration failures, lost audit trails and manual cleanup. A billing system must support end-to-end continuity without requiring intermediary objects or duplicate records.

| Ref | Requirement | Considerations | Vendor response |
| --- | --- | --- | --- |
| 6.1 | Sync product and pricing configuration directly from CRM or CPQ | When billing systems rely on static product catalogs or manual entry, teams must duplicate configuration logic – leading to discrepancies between quoted and billed values |  |
| 6.2 | Reflect amendments made in CRM directly in billing records, without requiring change orders | If every amendment requires a new change order, teams face delays, rekeying and increased error risk, especially in high-velocity or high-touch sales environments |  |
| 6.3 | Accept usage data from multiple sources (e.g. product telemetry, APIs, files) in real time | Systems that only accept batch uploads or predefined formats cannot handle dynamic usage, forcing engineering or ops to build preprocessing pipelines |  |
| 6.4 | Allow external systems to trigger billing events via API | Without real-time event handling, billing lags behind service delivery or usage and delays invoicing, causing misalignment with customer expectations |  |
| 6.5 | Push billing data downstream to finance, analytics and data warehouse systems automatically | Manual exports or delayed syncs create reporting gaps, reconciliation work and stale data in revenue dashboards or close processes |  |
| 6.6 | Track and log changes to integrated records for audit and troubleshooting | Without a clear integration audit trail, teams struggle to diagnose issues when data mismatches occur, increasing the risk of delayed billing or incorrect revenue attribution |  |
| 6.7 | Prevent activation bottlenecks between CRM/CPQ and billing | If billing requires a separate activation step after CRM close, missed handoffs delay invoicing and force finance to backdate revenue and fix reporting manually. |  |
| 6.8 | Accept CRM/CPQ amendments without intermediary constructs | If amendments require change orders or billing accounts not used in CRM, integrations break on multi-part changes and contract history becomes fragmented. |  |
| 6.9 | Post full sub-ledger transactions to the ERP, not just summarized journal entries | If only summary journal entries post to the ERP, finance can’t drill into invoices or payments. This forces manual reconciliation and breaks audit traceability. |  |

# Section 7: Future‑proofing

Billing needs change as products evolve, pricing grows more complex, and teams adapt their GTM models. This section defines requirements that help the system keep pace – reducing the risk of future rework, custom development, or platform replacement as business needs shift.

| Ref | Requirement | Considerations | Vendor response |
| --- | --- | --- | --- |
| 7.1 | Support transitions between pricing models (e.g. fixed fee to usage‑based) without reimplementation | If pricing changes require reconfiguration of catalogs, contracts or billing logic from scratch, time‑to‑market slows and system cost increases. |  |
| 7.2 | Handle mixed billing models within a single contract (e.g. fixed, usage and milestone‑based charges) | Rigid systems that separate billing logic by model create fragmented invoicing, revenue misalignment and duplicate contract records. |  |
| 7.3 | Allow one billing account or master contract to cover multiple subsidiaries or customer entities | Without this, companies serving multi‑entity customers must manage multiple billing records manually, in turn increasing operational overhead and risking inconsistencies. |  |
| 7.4 | Adapt to channel models such as distributor, reseller or partner‑led billing | Systems that assume direct‑to‑customer billing can’t easily accommodate upstream discounting, downstream invoice generation or multi‑party settlements. |  |
| 7.5 | Enable custom contract logic or billing structures without source code changes | If business‑specific billing needs require custom development, agility is limited and cost of change increases with every update. |  |
| 7.6 | Configure new product lines or pricing models without global rework | Systems that tightly couple billing logic to global product catalog or revenue settings introduce unintended downstream changes during new launches. |  |

## Appendix

Use this space to add any organization‑specific scenarios, exceptions or measurement criteria that vendors should address in their responses.