COENNECT

ENTERPRISE FIELD GUIDE
EDITION 1.0 · 2026

SALESFORCE OPERATIONAL INTELLIGENCE

# The Salesforce Event Monitoring Playbook

How enterprise Salesforce teams turn event logs into security, operational, and adoption intelligence.

CONTENTS

# The Salesforce Event Monitoring Playbook

FOREWORD

# Why Event Monitoring Has Become Essential for Enterprise Salesforce

RESEARCH CURRENT THROUGH JULY 28, 2026

Salesforce Event Monitoring is most valuable when it is treated as operational telemetry, not as a collection of CSV files. The platform records how users, applications, integrations, automation, and increasingly AI agents interact with Salesforce. Those records can reveal security exposure, failing integrations, slow experiences, underused capabilities, and emerging operational risk. The opportunity is substantial because the telemetry already exists. The challenge is turning it into decisions.

Raw event data rarely delivers that value on its own. Event types have different schemas, delivery cadences, retention rules, identifiers, and limitations. A high event count can be normal for one business process and alarming for another. A failed login can be harmless user error, an expired integration credential, or part of a coordinated attack. A slow page can be caused by the browser, a Lightning component, Apex, Flow, a report, an integration, or a downstream system. Context determines meaning.

This playbook is built around a simple operating model:

Event data + business and technical context + a defined response = operational intelligence.

A mature Event Monitoring program must answer five questions consistently:

1. What happened?
2. Was it expected for this user, system, process, and time?
3. What business service, data, or customer experience was affected?

4. What action should be taken, by whom, and with what urgency?
5. Did the response reduce risk or improve performance, adoption, or value?

The book therefore does not attempt to reproduce Salesforce documentation or list every field in every event type. Salesforce adds event types and changes schemas over time. A durable monitoring program validates current schemas, preserves raw evidence, and focuses its interpretation on operational questions that remain stable even as the platform evolves.

## THE VALUE OF EVENT MONITORING

Event Monitoring is often purchased for security or compliance. Those are important use cases, but they represent only part of its enterprise value. The same telemetry can help a Salesforce team:

- Detect suspicious access, exports, permission changes, and API behavior.
- Diagnose automation, UI, report, and integration performance.
- Understand which applications, features, reports, and channels people actually use.
- Measure whether investments are producing adoption and business outcomes.
- Create evidence for governance, audits, incident response, and platform planning.
- Monitor the growing activity of Agentforce and other agentic clients without confusing traditional event logs with full agent-session observability.

## THE PRINCIPLES USED THROUGHOUT THIS PLAYBOOK

Start with decisions, not event types. Define what the organization needs to detect, explain, or improve before choosing data sources.

Use the correct telemetry channel. Historical Event Log Files, Real-Time Event Monitoring, Event Log Objects, Transaction Security, Agentforce tracing, and external observability tools solve different problems. They are complementary.

Preserve evidence before transforming it. Keep the raw event payload, source schema, file identity, sequence, timestamp, and ingestion history so findings can be reproduced.

Add context before assigning risk. Enrich events with user role, license, permissions, application, connected client, integration owner, metadata dependencies, deployment windows, data sensitivity, and business criticality.

Measure rates and impact, not only volume. Event counts without denominators or baselines are often misleading. A useful metric explains frequency, affected population, severity, trend, and business consequence.

Make every alert actionable. A detection should have a named owner, evidence, severity, recommended response, and closure criteria. Alerts that create no decision should not exist.

Treat monitoring as a continuous operating discipline. Dashboards are useful, but value is created through ownership, playbooks, response, remediation, and measurement.

## WHO THIS PLAYBOOK IS FOR

This guide is written for enterprise Salesforce administrators, architects, platform owners, security teams, integration leaders, Centers of Excellence, and executives responsible for Salesforce outcomes. It assumes familiarity with Salesforce but does not require the reader to be a log engineer.

Each chapter is intentionally concise. The goal is not to maximize the number of pages. The goal is to give teams the frameworks, distinctions, and practical guidance required to unlock the full value of Salesforce Event Monitoring.

The chapters begin with the foundations of Event Monitoring, progress through security, operations, adoption, governance, and AI-assisted detection, and conclude with a practical framework for measuring success and realizing ROI. The central message remains consistent throughout: Event Monitoring becomes a high-value investment when organizations stop asking how to download the logs and start asking what the evidence should cause them to do.

CHAPTER 01

# Why Every Enterprise Should Care About Event Monitoring

## THE OPERATING REALITY

An enterprise Salesforce org is not a static application. It is a living operating environment shaped by thousands of users, integrations, automations, permission changes, deployments, packages, and business processes. Metadata tells a team what has been configured. Event Monitoring shows how that configuration is behaving in production.

That distinction matters. Two orgs can have nearly identical metadata and operate very differently. One may have healthy adoption, predictable integrations, and stable performance. The other may contain abandoned applications, excessive exports, retry storms, failing automation, and users who have quietly moved critical work outside Salesforce. Architecture alone cannot explain those differences. Behavioral telemetry can.

## THE FOUR OUTCOMES THAT JUSTIFY THE INVESTMENT

### 1. Stronger security

Event data provides evidence of who accessed Salesforce, from where, through which client, and what they attempted to do. Login, API, report, list view, file, permission, session, and transaction-security events can help identify suspicious access, unusual data movement, privilege activity, and policy violations. The value is not simply finding an event. It is reducing the time between risky behavior and a controlled response.

### 2. More reliable operations

A Salesforce platform team is accountable for user experience and business continuity, even when the root cause crosses multiple layers. Lightning performance, UI telemetry, Apex execution, unexpected exceptions, reports, API activity, callouts, queued work, and other events provide a common evidence base for investigating slowness, failures, and capacity pressure.

### 3. Better adoption and product decisions

License assignment and metadata inventory show what users could use. Event data shows what they actually use. That evidence helps teams evaluate application adoption, report engagement, search behavior, mobile usage, feature utilization, and the effect of training or release changes. It also prevents a common mistake: declaring a feature successful because it was deployed rather than because it changed behavior.

### 4. Clearer return on Salesforce investment

Event Monitoring can connect platform activity to the outcomes leaders care about: reduced security exposure, fewer operational incidents, faster investigation, higher adoption, improved experience, better integration reliability, and more defensible spending decisions. It turns platform conversations from opinion into evidence.

## WHY ENTERPRISES STRUGGLE

Most organizations do not lack data. They lack an operating model for the data.

Common failure patterns include:

- Downloading logs only after an incident.
- Building dashboards without assigning response ownership.
- Treating every event as equally important.
- Using fixed thresholds without accounting for seasonality, user population, or process volume.
- Separating event data from metadata, permissions, integrations, and business ownership.
- Measuring total activity when the meaningful question is rate, failure, exposure, or outcome.
- Sending raw alerts to teams that cannot interpret or remediate them.

- Assuming Event Monitoring captures every user action or every layer of an agent interaction.

A mature program replaces those patterns with defined questions, trusted telemetry, contextual analysis, and repeatable playbooks.

## A PRACTICAL STARTING FRAMEWORK

Begin with critical services, not with the full event catalog. Select five to ten business capabilities whose failure, abuse, or underuse would matter materially. Examples include customer onboarding, case routing, order processing, payment integration, regulated report access, partner portals, and agent-driven service actions.

For each capability, document:

- Business owner and technical owner.
- Users, integration identities, agents, and external clients involved.
- Sensitive data and privileged actions in scope.
- Expected activity pattern and business calendar.
- Events and metadata required to observe the capability.
- Conditions that require notification, investigation, blocking, or remediation.
- Service-level objectives and success measures.

Then classify each monitoring requirement by latency:

**Immediate control:** The action may need to be blocked or challenged. Use Real-Time Event Monitoring and Transaction Security where supported.

**Rapid investigation:** The team needs evidence within minutes or hours. Use streaming, stored real-time events, hourly Event Log Files, or an external monitoring platform.

**Trend and optimization:** The team needs daily, weekly, or monthly analysis. Use Event Log Files, Event Log Objects, normalized history, and curated dashboards.

### ENTERPRISE EXAMPLE

A global financial-services company sees API usage rise sharply overnight. A volume-only alert suggests possible abuse. Context changes the diagnosis.

The requests originate from a known integration identity and approved client. API events show repeated calls against the same resource. Named Credential and Apex execution events correlate through the request identifier. A deployment record shows that a new retry mechanism was released earlier that evening. The downstream service is returning intermittent errors, and the integration is retrying without sufficient backoff.

The real risk is not unauthorized access. It is an integration design defect that is consuming limits, delaying overnight processing, and degrading the next morning's user experience.

The monitoring response should therefore include:

- Contain the retry storm.
- Restore the previous integration behavior or apply controlled backoff.
- Validate downstream availability.
- Measure affected transactions and business delay.
- Add a playbook that correlates API growth, error rate, repeated resources, deployment changes, and integration ownership.

Event data created the evidence. Context produced the correct decision.

### WHAT GOOD LOOKS LIKE

An effective Event Monitoring program can explain its coverage in business terms. It knows which critical services are monitored, which risks can be detected, which actions can be controlled in real time, who owns each response, and whether the program is improving outcomes.

It does not attempt to alert on everything. It prioritizes signals that are material, explainable, and actionable.

**KEY TAKEAWAYS**

- Metadata explains what exists; Event Monitoring explains what is happening.
- The highest-value use cases span security, reliability, adoption, and investment decisions.
- Start with critical business capabilities and decisions, not a list of event types.
- Select telemetry based on required response latency.
- Enrich events before assigning meaning or severity.
- The measure of success is not how many logs are collected. It is how consistently evidence produces a better decision.

CHAPTER 02

# What Event Monitoring Actually Captures

## A useful architecture, not a single log source

"Event Monitoring" refers to a family of capabilities rather than one uniform dataset. Enterprise teams get better results when they choose the right access pattern for the operational question.

### 1. Event Log Files

Event Log Files are CSV-formatted files represented by the EventLogFile object. They provide broad historical coverage across many event types and are well suited to investigations, trend analysis, warehousing, and normalized analytics. Paid Event Monitoring customers can receive hourly files, but hourly does not mean immediate. Delivery can lag, and the files should not be used as the only control for an action that must be stopped in real time.

The EventLogFile object exposes the event type, interval, log date, file length, sequence, API version, field names, and field types. Access requires appropriate permissions, including View Event Log Files and API Enabled for API-based collection. A dedicated least-privileged collection identity is preferable to using a broad administrator account.

### 2. Real-Time Event Monitoring

Real-Time Event Monitoring publishes supported security and access events through a streaming model. Subscribers can consume them through Pub/Sub API, and event relays can deliver supported events. Streaming replay data is retained for up to three days.

Real-Time Event Monitoring also supports stored event data and Transaction Security policies. Transaction Security can notify, block an action, or require stronger authentication for supported conditions. This is the correct layer when the business requirement is prevention or immediate

intervention rather than next-day analysis.

### 3. Event Log Objects

Event Log Objects expose supported event data as queryable Salesforce objects rather than CSV files. They provide a useful SOQL-based path for targeted analysis. As of 2026, availability is limited to Hyperforce customers, only a subset of Event Log File coverage is represented, and retention is up to 30 days. They are useful for interactive queries and Salesforce-native analysis, but they do not replace a durable enterprise history.

### 4. Event Monitoring Analytics App

The Event Monitoring Analytics App provides prebuilt CRM Analytics dashboards and is an effective starting point for organizations that need visibility quickly. It is not a complete enterprise operating model. The app covers a subset of event types and depends on dataflow configuration, refresh timing, permissions, and data volume. Treat it as an accelerator, not as proof that every important condition is monitored.

### 5. External analytics, SIEM, or control-plane platforms

Large organizations commonly move Event Monitoring data into a security platform, data lake, warehouse, observability tool, or Salesforce control plane. This supports longer retention, cross-org comparisons, enrichment, anomaly detection, case management, and correlation with non-Salesforce telemetry. The architecture should preserve Salesforce identifiers and source fidelity so an analyst can trace a finding back to its evidence.

## THE INFORMATION MODEL

Although each event type has its own schema, most operational analysis depends on a common set of concepts:

- **Event:** What occurred.
- **Time:** When it occurred, normalized to a consistent time zone.
- **Actor:** User, automated process, integration identity, guest, agent, or internal Salesforce service.
- **Client:** Browser, mobile app, Connected App, External Client App, API client, package, or agentic client.
- **Session:** Login key, session key, device session, or agent session where available.

- **Request:** REQUEST_ID and related trace identifiers that connect work performed within one transaction.
- **Target:** Object, record, report, file, page, endpoint, action, or other resource.
- **Outcome:** Success, failure, denial, exception, block, or unknown.
- **Volume:** Rows, bytes, records, calls, queries, or other scale measure.
- **Duration:** End-to-end time and component times where available.
- **Context:** Org, app, business capability, owner, data classification, permissions, deployment, and dependency.

A durable normalized model keeps these common concepts while preserving every source field in the raw payload. Do not force every event into a false universal schema. A missing status field is not the same as success, and a zero duration may reflect how the event was generated rather than perfect performance.

## THE INGESTION CONTRACT

A production collection process should meet seven requirements.

### 1. Inventory before download

Query the EventLogFile records first. Record the file ID, event type, interval, log date, sequence, API version, created date, and length. This creates an auditable manifest and allows the pipeline to distinguish “no activity” from “file not collected.”

### 2. Collect every sequence

High-volume hourly event data can be split across multiple files. The Sequence field increments when more than one file is generated for the same hour. Missing a sequence produces silent gaps.

### 3. Validate the schema every release

Salesforce explicitly notes that event schemas can change. Read LogFileFieldNames and LogFileFieldTypes for each EventType and compare them with the expected ingestion contract. Add new fields without dropping the raw payload, and fail visibly when a breaking change occurs.

### 4. Preserve raw evidence

Store the source file, checksum, ingestion timestamp, schema, and original row. Normalization should be reproducible. Investigators must be able to show what Salesforce produced before enrichment or scoring.

### 5. Normalize identifiers and time

Use 18-character Salesforce IDs when possible, preserve the source ID, normalize timestamps to UTC, and retain org, event type, interval, and sequence. Map client IDs, user IDs, report IDs, object IDs, named credentials, packages, and other identifiers to current metadata without overwriting historical meaning.

### 6. Correlate transactions

REQUEST_ID is one of the most valuable fields in the ecosystem because multiple events in one transaction can share it. For API integrations, clients can set the X-SFDC-REQUEST-ID header with a trace-compatible identifier and then search Event Monitoring data for that value. Also use LOGIN_KEY, SESSION_KEY, device identifiers, bot or agent identifiers, and origin request IDs when available.

### 7. Monitor the monitor

Track ingestion freshness, expected file arrival, row counts, duplicate rate, parse failures, schema drift, missing sequences, and enrichment failures. A monitoring program that cannot detect gaps in its own evidence is not audit-ready.

## IMPORTANT LIMITATIONS

- Lightning Interaction is best-effort telemetry. Salesforce states that it is not intended to satisfy privacy or security audit requirements, that not all interactions or CRUD operations are tracked, and that data loss can occur.
- Composite API requests can be represented only by the parent request in EventLogFile. Do not assume one logged row for every child operation.
- Some event types are generated only when qualifying activity occurs. Absence of a file can mean zero events, a configuration issue, delayed generation, or failed collection.
- Event Monitoring does not replace Setup Audit Trail, metadata history, debug logs, Apex job monitoring, Flow error channels, infrastructure monitoring, or Agentforce session tracing. Use each source for the evidence it can reliably provide.

## CHOOSING THE RIGHT PATH

Use Transaction Security when an action must be blocked or challenged.

Use real-time streaming when security or operations need rapid external consumption.

Use Event Log Objects when a Salesforce-native SOQL investigation is sufficient and the supported coverage meets the need.

Use Event Log Files for broad historical analysis, correlation, and enterprise trend reporting.

Use a normalized external or multi-org platform when scale, retention, enrichment, cross-org governance, or automated playbooks exceed the native starting point.

**KEY TAKEAWAYS**

- Event Monitoring is a set of complementary telemetry channels.
- Delivery speed, retention, schema, and supported events vary by channel.
- Reliable collection requires a manifest, sequence handling, schema validation, raw preservation, and pipeline health monitoring.
- REQUEST_ID and session identifiers are central to meaningful correlation.
- Know the limitations of each event type before using it as audit evidence or a control.

CHAPTER 03

# The Salesforce Event Monitoring Maturity Model

## Maturity is the ability to act

Event Monitoring maturity is not determined by how many event types an organization collects. It is determined by how reliably the organization converts evidence into prevention, investigation, remediation, and measurable improvement.

Maturity should be assessed by use case. A company may be advanced at monitoring report exports while remaining immature at diagnosing integration failures or measuring Agentforce activity. A single enterprise-wide score can hide those gaps.

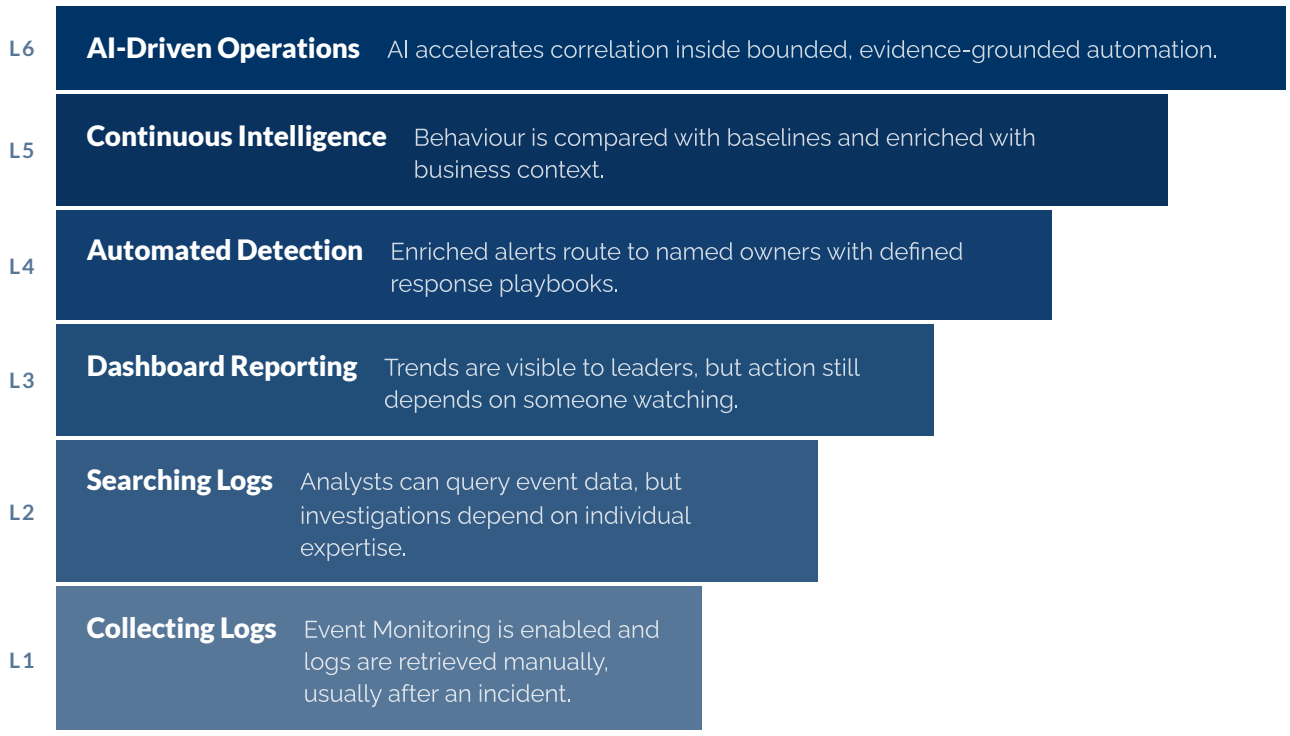


FIGURE 1 — THE SALESFORCE EVENT MONITORING MATURITY MODEL. MATURITY IS ASSESSED PER USE CASE, NOT ORG-WIDE.

## LEVEL 1 Collecting Logs

### TYPICAL BEHAVIOR

- Event Monitoring is enabled.
- Logs are downloaded manually or retained in Salesforce.
- Activity is reviewed mainly after an incident or audit request.
- Knowledge is concentrated in one or two administrators.

### PRIMARY RISK

Evidence exists but is incomplete, short-lived, or inaccessible when needed. The team cannot prove collection coverage or identify missing files.

### EXIT CRITERIA

- Named owner for the collection process.
- Verified access and retention settings.
- Automated file inventory and download.
- Collection heartbeat, missing-sequence detection, and raw preservation.

### BEST NEXT MOVE

Establish a reliable evidence pipeline before building more dashboards.

## LEVEL 2 Searching Logs

### TYPICAL BEHAVIOR

- Administrators can query EventLogFile or use the ELF Browser.
- Analysts know several important event types and fields.
- Investigations are still manual and depend on individual expertise.
- Findings are documented inconsistently.

### PRIMARY RISK

The organization can answer known questions, but it cannot detect unknown issues or respond consistently under pressure.

### EXIT CRITERIA

- Standard investigation queries for priority use cases.
- Event-to-user, client, report, object, integration, and request mapping.
- Repeatable evidence package for incidents and audits.
- Defined limitations for each event type.

### BEST NEXT MOVE

Convert recurring investigations into curated datasets and standard views.

## LEVEL 3 Dashboard Reporting

### TYPICAL BEHAVIOR

- Dashboards show logins, API usage, exports, performance, and other trends.
- Leaders receive periodic reports.
- Teams can identify outliers, but action usually depends on someone watching the dashboard.
- Metrics emphasize activity volume more than risk or outcome.

### PRIMARY RISK

Visibility improves without creating accountability. Important conditions can remain unresolved because no workflow begins when a metric changes.

### EXIT CRITERIA

- Every dashboard has a defined audience and decision.
- Metrics include denominators, baselines, affected scope, and business context.
- Findings link to owners and remediation work.

### BEST NEXT MOVE

Turn the highest-value dashboard conditions into automated detections and playbooks.

## LEVEL 4 Automated Detection

### TYPICAL BEHAVIOR

- Rules identify suspicious, failing, or materially changing behavior.
- Alerts contain enriched evidence and route to an owner.
- Transaction Security is used for selected real-time controls.
- Response playbooks define triage, containment, escalation, and closure.

### PRIMARY RISK

Poorly designed rules create alert fatigue. Fixed thresholds generate noise when they ignore seasonality, user populations, deployment windows, and business cycles.

### EXIT CRITERIA

- Materiality and severity standards.
- Alert ownership and response service levels.
- False-positive, duplicate, and closure measurement.
- Multi-signal logic for critical detections.
- Regular playbook testing.

### BEST NEXT MOVE

Add behavioral baselines, trend analysis, and cross-source correlation.

## LEVEL 5 Continuous Intelligence

### TYPICAL BEHAVIOR

- Current activity is compared with historical baselines and peer groups.
- Event data is enriched with metadata, permissions, ownership, data sensitivity, deployments, and business criticality.
- Monitoring outcomes influence architecture, training, licensing, and roadmap decisions.

### PRIMARY RISK

Complex scoring can become opaque. Teams may trust a risk score without understanding the evidence or uncertainty behind it.

### EXIT CRITERIA

- Explainable findings with source evidence and calculation logic.
- Confidence, evidence coverage, and known limitations.
- Feedback from investigations improves future detections.
- Executive metrics connect monitoring to security, reliability, adoption, and value.

### BEST NEXT MOVE

Use AI selectively to accelerate investigation, summarize evidence, and detect complex patterns while preserving human accountability.

### LEVEL 6 AI-Driven Operations

**TYPICAL BEHAVIOR**

- AI assists with pattern discovery, correlation, root-cause hypotheses, evidence summaries, and recommended actions.
- Natural-language investigation is grounded in governed event and metadata evidence.
- Agent activity is monitored across Event Monitoring, session tracing, platform tracing, action execution, integrations, and business outcomes.
- Automated responses are bounded by policy, reversibility, confidence, and human approval requirements.

**PRIMARY RISK**

AI can create convincing but unsupported explanations, hide uncertainty, or recommend an action outside the evidence. Automation can expand the blast radius of a false conclusion.

**EXIT CRITERIA**

- Every answer cites governed evidence.
- The system abstains when evidence is insufficient.
- Recommendations disclose confidence, limitations, and the rule or model used.
- Consequential actions require approved controls and human oversight.
- AI quality, unsupported-claim rate, and outcome accuracy are measured.

**BEST NEXT MOVE**

Continuously evaluate whether automation is improving decisions rather than merely producing more analysis.

## THE MATURITY ASSESSMENT

Score each priority use case against these ten questions:

1. Do we know which source events and fields are required?
2. Can we prove that the evidence is collected completely and on time?
3. Is the raw evidence retained and reproducible?
4. Is the activity enriched with ownership, permission, metadata, and business context?
5. Do we know the expected baseline and denominator?
6. Is the detection material, explainable, and tested?

7. Is there a named response owner and service level?
8. Can the team contain, remediate, and close the condition?
9. Do we measure false positives, recurrence, and business impact?
10. Does feedback improve the monitoring logic?

A "no" identifies the next capability to build. It is usually more valuable to advance five critical use cases to Level 4 than to collect fifty event types at Level 1.

## ROI GUIDANCE

Do not skip foundational levels. AI cannot repair missing files, broken schemas, unidentified clients, or absent ownership. It can only accelerate analysis of the evidence that exists.

Funding should follow operational value:

- First, ensure complete and trustworthy collection.
- Second, standardize investigations and dashboards.
- Third, automate material detections and playbooks.
- Fourth, add contextual baselines and cross-org intelligence.
- Finally, apply AI where it measurably improves speed, accuracy, or coverage.

### KEY TAKEAWAYS

- Maturity is use-case specific and measured by the ability to act.
- Collection and dashboards are foundations, not the end state.
- Automated detection requires context, ownership, and playbooks to avoid alert fatigue.
- Continuous intelligence combines event behavior with metadata and business context.
- AI-driven operations must remain evidence-grounded, explainable, and bounded.
- Advance the most critical use cases deeply before expanding coverage broadly.

CHAPTER 04

# Detecting Security Risks Before They Become Incidents

## An event is evidence, not a risk rating

Security monitoring fails when normal Salesforce activity is labeled suspicious without context. A report export, API call, login from a new location, or permission change can be legitimate. The same event can also be the first visible signal of account compromise, data exfiltration, or unauthorized administration.

Risk is determined by the combination of behavior, identity, privilege, data sensitivity, business purpose, timing, and deviation from an expected pattern.

A useful security finding should answer:

- What happened and which source events support it?
- Who or what performed the activity?
- What access did the actor have at that moment?
- Which data, configuration, or business process was exposed?
- Was the behavior expected for that actor and client?
- Is the activity still in progress?
- What containment action is appropriate and reversible?

## THE SIX SECURITY MONITORING DOMAINS

### 1. Authentication and Access

Relevant signals include login success and failure, LoginAs activity, geography, IP address, authentication method, client, user type, session start, and privileged-user status.

Higher-risk patterns include:

- Repeated failures followed by a successful login.
- A new country, network, device, or client for a privileged user.
- Concurrent or rapidly changing geographies that are inconsistent with normal travel.
- Activity by a dormant, terminated, or service identity.
- Interactive login by an identity expected to be integration-only.
- LoginAs activity without a support case or approved window.
- Authentication behavior that changes immediately after a permission or credential update.

Do not assign risk from geography alone. Corporate VPNs, mobile networks, Salesforce infrastructure, and outsourced operations can create legitimate variation.

### 2. Data Access and Exposure

Relevant signals include Report, ReportExport, ListView, API, Bulk API, file, search, and record-access events. Enrich them with object and field sensitivity, report ownership, row counts, bytes, user role, business unit, and historical behavior.

Higher-risk patterns include:

- Unusually large exports relative to the user's baseline.
- Repeated access to reports or objects outside the user's normal responsibilities.
- Export activity shortly after privilege elevation.
- High-volume data access by a new or unapproved API client.
- Access to sensitive data during off-hours or from an unfamiliar session.
- Multiple retrieval methods used against the same sensitive dataset.
- Bulk extraction followed by account deactivation or employment termination.

A row count is not a risk by itself. Ten thousand public product records and ten thousand regulated customer records have different consequences.

## 3. Administrative and Configuration Changes

Relevant evidence can include PermissionUpdate, package installation, sandbox activity, Transaction Security execution, Setup Audit Trail, metadata snapshots, deployment history, and current permission assignments.

### Higher-risk patterns include:

- Granting Modify All Data, View All Data, Manage Users, Customize Application, or other critical permissions outside an approved change.
- Changes to profiles or permission sets used by agents, integrations, or external users.
- Creation of a new admin or broad service identity without an owner.
- Installation or activation of an unapproved package or client.
- Changes to security controls, session settings, connected clients, named credentials, or transaction-security policies without separation of duties.
- Permission elevation followed by immediate export, API, or setup activity.

Event Monitoring should not be treated as the sole source for configuration history. Setup Audit Trail and metadata change evidence remain essential.

## 4. API and Integration Security

Relevant signals include API Total Usage, REST API, SOAP API, Bulk API, Apex REST, Named Credential, external callout, client ID, API family, resource, user, IP, request identifier, response, rows, and duration.

The API Total Usage event can distinguish requests from Agentforce agents, external applications, Lightning UI, Salesforce services, and unknown categories. That classification is valuable, but it should be enriched with the approved application registry and current client ownership.

### Higher-risk patterns include:

- Unknown or unapproved clients.
- Shared integration users serving unrelated systems.
- Interactive and API activity mixed under one identity.
- New API families or resources for an established integration.
- Abnormal data volume, call rate, or failure behavior.

- Use of sensitive named credentials by an unexpected package, user, bot, or planner.
- Requests from obsolete clients, excessive OAuth scopes, or weak authentication patterns.
- Agentic API activity that invokes privileged Apex, Flow, or external actions outside the agent's approved purpose.

### 5. Session and Device Hygiene

Session identifiers, login keys, device identifiers, browser, operating system, client IP, application, and user agent help analysts understand continuity. Useful detections include token reuse from inconsistent locations, sessions that outlive expected work patterns, abnormal device changes, and multiple high-risk actions inside one session.

Session evidence must be handled carefully. Some identifiers are feature-specific, and privacy requirements may limit how device and behavior data can be retained or used. Define a documented purpose and retention policy before building detailed user surveillance.

### 6. Anomalies and Threat Indicators

The strongest threat detections combine signals. Examples include:

- New geography + privileged identity + sensitive export.
- Permission elevation + report export + account deactivation.
- New API client + bulk data access + unusual IP range.
- Login failure burst + successful authentication + session change + high-risk setup action.
- Named Credential usage by an unrecognized package + unusual callout destination behavior.
- Agentforce client activity + high-privilege action + missing human approval.

A multi-signal finding is usually more actionable than several disconnected alerts.

## WHEN TO USE TRANSACTION SECURITY

Transaction Security is appropriate when the organization knows in advance that a supported action must be blocked, challenged, or immediately surfaced. Examples include:

- Blocking or notifying on a high-volume report or list-view export.
- Requiring stronger authentication for selected high-risk behavior.

- Controlling API, report, list-view, or file activity under defined conditions.
- Monitoring critical permission changes where supported.

Policies should be narrow, tested, and owned jointly by Salesforce and security teams. Overlapping policies can create unpredictable execution order, and an aggressive block can interrupt a legitimate business process. Begin with notification or a limited population when the operational effect is uncertain.

## A PRACTICAL SEVERITY MODEL

### Critical

Active or highly credible compromise, large sensitive-data exposure, unauthorized privileged action, or a condition requiring immediate containment.

### High

Material risk with strong evidence, broad access, sensitive data, or significant policy violation, but no confirmed ongoing compromise.

### Medium

Meaningful deviation that requires investigation, limited exposure, or a control gap that could become material.

### Low

Minor deviation, hygiene issue, or evidence requiring trend monitoring rather than urgent response.

### Informational

Expected activity retained for context, reporting, or future correlation.

Severity should consider impact and confidence separately. A high-impact hypothesis with weak evidence may require urgent validation, but it should not be presented as a confirmed incident.

## THE INVESTIGATION EVIDENCE PACKAGE

Every material finding should preserve:

- Source event rows and raw files.
- Org, time range, event type, file sequence, and ingestion status.
- User, client, session, request, device, and network context.

- Permissions and role effective at the time of activity where reconstructable.
- Object, report, file, endpoint, or data-sensitivity context.
- Related configuration and deployment changes.
- Baseline comparison and detection logic.
- Actions taken, approver, timestamps, and outcome.

### ENTERPRISE EXAMPLE

A regional sales leader exports a large customer report from a familiar location. The event alone appears legitimate. Enrichment shows that the report includes regulated identifiers, the user has never exported more than a small subset, and a broad permission set was assigned two hours earlier through an emergency request with no recorded approval.

The correct response is not simply "large export." It is a correlated finding: unusual sensitive-data movement following unexplained privilege elevation.

The playbook should verify the change request, preserve the export evidence, suspend or remove the elevated access if unauthorized, assess the data scope, notify the data owner and security team, and determine whether the activity was transmitted outside approved channels.

### KEY TAKEAWAYS

- Treat events as evidence and assign risk only after enrichment.
- Organize security monitoring across authentication, data exposure, administration, integrations, sessions, and multi-signal anomalies.
- Use data sensitivity, privilege, baseline, client approval, and business purpose to determine severity.
- Use Transaction Security for narrow, tested conditions that require immediate control.
- Correlated findings are more valuable than disconnected event alerts.
- Preserve a reproducible evidence package for every material investigation.

CHAPTER 05

# Monitoring Platform Health

## Event Monitoring is a diagnostic layer, not a complete APM system

Salesforce performance problems cross boundaries. A slow experience can originate in the browser, Lightning components, Apex, Flow, reports, database work, API calls, callouts, middleware, or a downstream service. Event Monitoring provides valuable evidence across many of those layers, but it does not replace debug logs, Apex job monitoring, Flow error channels, external application performance monitoring, or infrastructure telemetry.

The goal is to use Event Monitoring to identify scope, pattern, and correlation quickly enough to direct the deeper investigation.

### THE FOUR PERFORMANCE LAYERS

#### 1. User Interface and Front-End Experience

Relevant event types can include Lightning Performance, Lightning Interaction, URI, UI Telemetry Navigation Timing, UI Telemetry Resource Timing, and custom Lightning Logger events.

Useful dimensions include:

- Application and page route.
- Page entity type and record context.
- Browser, version, operating system, device, and connection type.
- Geography and network timing.
- Page load, rendering, resource, redirect, and interaction duration.

- Standard or custom component context where available.
- User population and session.

Monitor percentiles rather than only averages. The median describes the typical experience; p95 and p99 reveal the users who experience severe degradation. Segment before concluding that the entire org is slow. A problem limited to one browser version, mobile form factor, geography, app, or page is a different remediation than an org-wide issue.

Lightning Interaction is best-effort telemetry and can lose data. Use it for behavioral and diagnostic trends, not as complete proof that a user did or did not perform a specific action.

The Logger API can instrument custom Lightning web components and emit Lightning Logger events. Use it deliberately for business and diagnostic signals that standard telemetry cannot provide. Avoid logging secrets, regulated data, or uncontrolled message payloads.

## 2. Application and Automation Layer

Relevant events can include ApexExecution, ApexUnexpectedException, FlowExecution, queued execution, time-based workflow, continuation, and callout-related events.

ApexExecution can provide execution time, run time, CPU time, SOQL counts, DML activity, callouts, long-running indicators, entry point, quiddity, planner identifier, and request identifiers. Those fields make it possible to distinguish a synchronous Lightning request from scheduled Apex, queueable work, a bulk process, an invocable action, Apex REST, or an Agentforce-related execution path.

### High-value patterns include:

- Rising p95 execution time for a stable entry point.
- Increased CPU or query use without corresponding business-volume growth.
- Long-running requests concentrated in one release, user population, or quiddity.
- Repeated unexpected exceptions after deployment.
- Increasing callout time or failure in an otherwise stable transaction.
- Agent or planner executions invoking expensive or failing Apex actions.

FlowExecution confirms that a flow ran and can support correlation, but it is not a universal or complete Flow-error feed. Use Flow error emails, platform events where configured, debug evidence, failed interviews, Apex exceptions, and user-facing error telemetry as additional sources. Do not promise complete Flow failure monitoring from one event type.

### 3. Data, Query, and Analytics Layer

Report, report export, search, list view, and related events can reveal expensive or highly repeated activity. Relevant measures include duration, rows, report or object identity, user, client, format, and frequency.

Look for:

> - Reports whose runtime or failure rate is increasing.
> - High-cost reports run repeatedly by many users.
> - Exports that compete with peak transactional workloads.
> - Search patterns that indicate users cannot find the intended data or process.
> - Analytics assets that are heavily used but have no accountable owner.
> - Large result sets that should be redesigned, scheduled, cached, or moved to a more appropriate data pattern.

An expensive report used once per quarter may be acceptable. A moderately expensive report loaded hundreds of times each hour can create greater operational cost. Combine cost per execution with frequency and business criticality.

### 4. API and Dependency Layer

API, Named Credential, external data callout, Apex callout, continuation, and related events expose the performance of integrations and downstream services.

Separate Salesforce processing time from external response time where the event fields allow it. A high end-to-end duration can be caused by Salesforce preparation, network transfer, external execution, result fetching, retry behavior, or payload size. That distinction prevents the Salesforce team from optimizing the wrong component.

## THE INVESTIGATION METHOD

### 1. Define the user-visible symptom

State the page, action, process, report, integration, or outcome that is slow or failing. Avoid beginning with a generic “Salesforce performance” incident.

### 2. Measure affected scope

Quantify users, transactions, business units, applications, geographies, clients, and time range. Determine whether the issue is isolated, segmented, or systemic.

### 3. Compare with a valid baseline

Use the same page, process, entry point, volume range, and business period. Account for month-end, seasonal demand, batch windows, and releases.

### 4. Correlate across layers

Use REQUEST_ID, LOGIN_KEY, SESSION_KEY, origin request identifiers, timestamps, user, page, entry point, and client to connect front-end, Apex, API, and callout evidence.

### 5. Correlate with change

Review deployments, metadata changes, permission changes, package upgrades, integration releases, data growth, browser changes, and Salesforce releases.

### 6. Validate the hypothesis

Use debug logs, traces, query plans, Flow diagnostics, external APM, downstream logs, and controlled reproduction. Event Monitoring narrows the search; deeper evidence confirms the root cause.

### 7. Confirm recovery

Measure the same percentile, failure rate, user population, and business outcome after remediation. Closing the technical task is not evidence that the experience improved.

## DESIGNING USEFUL PERFORMANCE METRICS

Use metrics that preserve the denominator and affected population:

- p50, p95, and p99 duration by page, entry point, report, or endpoint.
- Error rate per transaction, not only total errors.
- Long-running request rate by quiddity and entry point.
- CPU, SOQL, DML, callout, and row-use trend per successful transaction.
- External fetch time versus Salesforce execution time.
- Percentage of active users affected by degraded experience.
- Repeat incident rate after remediation.
- Business completion rate for the monitored process.

Filter test executions, deployment activity, known batch workloads, and synthetic monitoring where appropriate. They can distort production baselines.

### ENTERPRISE EXAMPLE

Service Console users report intermittent slowness after a release. The average page-load metric remains acceptable. p95 Lightning performance shows degradation only on one case workspace page, primarily in Chrome for users loading a custom component. Lightning Logger events from the component indicate repeated data refreshes. ApexExecution shows the backing method now performs additional queries, and REQUEST_ID correlation links the slow UI sessions to the changed entry point.

The root cause is a release that introduced duplicate refresh behavior and increased query work. The remediation removes the duplicate call, reduces the query scope, and adds a regression metric for p95 component duration and query count.

### WHAT GOOD LOOKS LIKE

A mature platform-health program can move from a user symptom to affected scope, likely layer, related change, validated cause, and measured recovery without relying on intuition. It uses Event Monitoring to create a shared evidence model across administrators, developers, architects, integration teams, and support.

### KEY TAKEAWAYS

- Use Event Monitoring to identify scope and correlation, then use deeper diagnostics to prove root cause.
- Monitor user-interface, application, data, and dependency layers together.
- Use percentiles, rates, and affected population rather than averages and raw counts.
- Treat Flow and Lightning interaction coverage honestly; neither is complete evidence for every failure or action.
- Correlate performance with deployments, metadata, clients, and downstream systems.
- Confirm recovery with the same metric and business outcome that defined the incident.

CHAPTER 06

# Integration Monitoring

## Monitor the business service, not only the API count

An enterprise Salesforce integration is a business service delivered through identities, clients, APIs, credentials, middleware, automation, endpoints, schedules, and downstream systems. Monitoring only total API consumption can show that traffic changed, but it rarely explains whether orders, payments, cases, customer data, or agent actions completed correctly.

Effective integration monitoring answers six questions:

- **Availability:** Is the integration running when it should?
- **Correctness:** Are requests and business transactions succeeding?
- **Performance:** Is latency within the agreed objective?
- **Capacity:** Is volume sustainable within platform and downstream limits?
- **Security:** Is the approved identity, client, credential, and scope being used?
- **Ownership:** Can a qualified team respond when the pattern changes?

### BUILD THE INTEGRATION REGISTRY FIRST

Event data becomes actionable when each integration is mapped to an approved registry containing:

- Business purpose and criticality.
- Business owner and technical owner.
- Source and target systems.
- Production and nonproduction environments.

- Integration user, agent user, client ID, application, and authentication method.
- API family, version, resources, named or external credentials, and endpoints.
- Data classification and permitted operations.
- Expected schedule, throughput, payload size, and seasonal pattern.
- Service-level objectives for success, latency, freshness, and recovery.
- Retry, idempotency, timeout, replay, and duplicate-handling design.
- Support team, escalation path, and decommission date.

An unknown client cannot be classified confidently, and an alert without an owner cannot be resolved reliably.

## THE MAIN TELEMETRY PATHS

### Inbound API activity

API Total Usage, REST API, SOAP API, Bulk API, Apex REST, and related events can reveal the user, client category, client ID, API family, resource, version, rows, bytes, status, duration, and request identifiers available for the call.

API Total Usage can classify activity from Agentforce agents, external applications, Lightning UI, Salesforce services, and unknown sources. Use that field as a starting point, then map it to the actual approved client and business purpose.

### Outbound callouts and external data

Named Credential, Apex callout, continuation, External OData Callout, External Data Source Callout, External Custom Apex Callout, and related events can show which credential or adapter was used, the calling package or planner, execution time, fetch time, rows, response size, status, and request context.

Where timing fields separate Salesforce execution from external fetching, use that distinction. It tells the team whether the bottleneck is local processing, network transfer, an external service, or result volume.

### Authentication and identity

Login, session, OAuth, client, and credential evidence helps identify expired credentials, unexpected interactive use, shared identities, unapproved clients, and changes in authentication behavior. An integration user should exhibit a narrow and predictable pattern. A sudden browser login or access to unrelated objects is material even if every API call succeeds.

### Asynchronous and event-driven processing

Bulk jobs, queued execution, Platform Events, Change Data Capture, Pub/Sub subscribers, middleware queues, and downstream consumers require telemetry beyond EventLogFile. Monitor replay position, duplicate handling, ordering assumptions, dead-letter conditions, backlog, and consumer health in the systems that own those responsibilities.

## THE INTEGRATION HEALTH MODEL

### Success

Measure completed business transactions, not only HTTP or API success. A 200 response can still contain rejected records, partial failures, duplicate work, or an incorrect business outcome.

Recommended measures:

- End-to-end business success rate.
- Salesforce request success rate.
- Partial-failure rate.
- Records accepted, rejected, retried, and dead-lettered.
- Reconciliation difference between source and target.

### Latency

Measure p50, p95, and p99 latency for the same operation and payload class. Separate Salesforce processing, external execution, queue wait, and end-to-end business completion where possible.

### Capacity

Track calls, rows, bytes, concurrency, batch size, retry volume, and limit consumption relative to business demand. Normalize by transactions processed so growth is interpreted correctly.

### Freshness

For scheduled or data-synchronization integrations, monitor the age of the last successful business result. “No errors” is not evidence of health when the job never started.

### Security

Track unexpected users, clients, IP ranges, API families, resources, named credentials, package namespaces, planner identifiers, scopes, and data volumes. Separate approved Agentforce activity from external application traffic and from unknown clients.

### Ownership

Measure integrations with no current owner, outdated documentation, shared identities, unrotated credentials, missing service levels, or no tested recovery procedure.

## HIGH-VALUE DETECTION PATTERNS

- Success rate falls while request volume remains normal.
- Throughput stops during an expected processing window.
- API activity rises without corresponding business-volume growth.
- A stable client begins using a new API resource or family.
- Authentication failures increase before successful calls resume.
- Retry volume grows faster than primary transaction volume.
- The same request or business key is processed repeatedly.
- External fetch time increases while Salesforce execution time remains stable.
- A new package, planner, bot, or user invokes a sensitive named credential.
- An integration user performs interactive UI activity.
- An Agentforce client invokes an API or Apex REST action outside the agent's approved capability registry.
- A critical integration has not produced a successful business outcome within its freshness objective.

## CORRELATION AND TRACEABILITY

REQUEST_ID connects events within a Salesforce transaction. For external API clients, set X-SFDC-REQUEST-ID to a trace-compatible identifier and propagate the same trace through middleware and downstream systems where possible.

A useful end-to-end trace can include:

- Business transaction ID.
- External trace ID.
- X-SFDC-REQUEST-ID and Salesforce REQUEST_ID.
- Integration identity and client ID.
- Middleware message or job ID.
- Named credential and endpoint.
- Apex, Flow, agent action, and downstream request.
- Final business result and reconciliation status.

Remember that composite API requests can be represented only by the parent request in EventLogFile. Use API response details and client-side telemetry when child-operation evidence is required.

## RETRY DESIGN IS A MONITORING REQUIREMENT

A retry is not automatically a recovery. Poorly designed retries can multiply load, duplicate transactions, exhaust limits, and hide a persistent failure.

Every critical integration should define:

- Which failures are retryable.
- Maximum attempts and exponential backoff.
- Idempotency key and duplicate behavior.
- Timeout and circuit-breaker conditions.
- Dead-letter or manual-recovery path.
- Alerting on retry growth and repeated business keys.
- Reconciliation after partial failure.

### ENTERPRISE EXAMPLE

An overnight customer synchronization shows a 99.8 percent API success rate, yet downstream users report missing account updates. Event data shows that API requests succeeded, but batch sizes fell sharply and the integration stopped before processing the complete source population. The job scheduler marked the run successful because no final exception occurred.

The meaningful controls are population reconciliation, last-successful-record time, expected row range, and source-to-target difference. API success remains useful, but it is not the business outcome.

### WHAT GOOD LOOKS LIKE

A mature integration-monitoring program can identify which business service is affected, whether the issue is authentication, Salesforce processing, external latency, data quality, retries, capacity, or orchestration, and which owner must respond. It measures recovery through a reconciled business result.

### KEY TAKEAWAYS

- Build an integration registry before relying on event alerts.
- Monitor availability, correctness, performance, capacity, security, and ownership together.
- Distinguish technical request success from business-transaction success.
- Use REQUEST_ID and propagated trace identifiers for end-to-end investigation.
- Treat missing activity, retry behavior, and reconciliation as first-class signals.
- Monitor Agentforce and other agentic clients as governed integrations, not as anonymous API traffic.

CHAPTER 07

# Understanding User Behavior

## Measure adoption without turning monitoring into surveillance

Event Monitoring can show how people access and experience Salesforce, but activity data must be used for a defined operational purpose. The goal is to improve adoption, usability, support, training, and investment decisions. It is not to score individuals based on incomplete click data or to create hidden employee surveillance.

Document the purpose, permitted users, retention, privacy review, and reporting level before analyzing detailed user behavior. Prefer aggregated and role-based insights unless an investigation requires user-level evidence.

### THE FOUR LEVELS OF ADOPTION EVIDENCE

#### 1. Access

Can the intended population reach Salesforce and the relevant capability?

Useful evidence includes active-user status, license and permission assignment, login frequency, authentication failures, channel, mobile usage, and application access.

Access is a prerequisite, not adoption. A user who logs in once a month may still be fully effective in a seasonal role, while a user who logs in every day may complete the real work in spreadsheets or another system.

### 2. Engagement

Is the intended population using the application, feature, report, search experience, or process repeatedly?

Relevant events can include URI, Lightning Interaction, Search, Search Click, Report, ListView, mobile, file, and page events. Lightning Interaction is best-effort and incomplete, so use it for trends rather than exact audit claims.

### 3. Effectiveness

Can users complete the intended task efficiently and successfully?

Combine engagement with page performance, errors, abandoned searches, repeated navigation, report runtime, support cases, Flow or Apex failures, and process completion. High activity can signal friction rather than success.

### 4. Outcome

Did usage improve a business result?

Examples include faster case resolution, improved opportunity hygiene, fewer manual handoffs, increased self-service containment, reduced integration rework, better forecast completeness, or successful agent escalation. Outcome measurement often requires business objects and process data in addition to Event Monitoring.

## START WITH THE ELIGIBLE POPULATION

Every adoption metric needs a denominator. Define who should use the capability before measuring who did.

Useful segmentation includes:

- Role, profile, permission set, and license.
- Business unit, department, region, and manager hierarchy.
- Application, persona, channel, and device.
- New hire, training cohort, pilot group, and release wave.
- Internal, partner, customer, guest, integration, and agent users.
- Assigned feature or process ownership.

Without eligibility, a 30 percent adoption rate may be excellent, poor, or meaningless.

## HIGH-VALUE ADOPTION METRICS

### Activation rate

Eligible users who complete the first meaningful action within the expected time after access is granted.

### Active penetration

Eligible users who perform the defined meaningful action during the reporting period.

### Repeat-use rate

Activated users who return and perform the action in a later period. This separates trial from sustained adoption.

### Feature retention

Users who continue using the capability across multiple periods after launch or training.

### Time to first value

Elapsed time from license or permission assignment to the first completed business outcome.

### Workflow completion rate

Started processes that reach the intended terminal state without error, abandonment, or manual workaround.

### Usage concentration

Share of activity generated by the top users, teams, reports, or applications. A capability can appear widely adopted when a small expert group generates most of the activity.

### Search effectiveness

Searches that lead to a useful click or task completion, repeated searches, no-result patterns, and queries that indicate missing navigation, content, or terminology.

### Report dependence

Frequency and breadth of report use, repeated exports, and concentration around manually maintained assets. Heavy report usage can indicate value, but heavy export activity can also indicate an offline workflow.

### Experience quality

p95 page or action performance, error rate, and affected users for the same capability. Adoption and performance must be read together.

### License utilization

Assigned users who exhibit the behavior that requires the premium license or permission. This is evidence for review, not an automatic recommendation to remove access. Account for seasonality, leave, backup duties, and infrequent high-value work.

## AVOID THE COMMON MISREADS

Logins equal adoption

They do not. Logins show access, not value or completion.

### Clicks equal success

They do not. Repeated clicks can indicate confusion, latency, or failure.

### Low activity means the feature is unnecessary

It can also mean poor discoverability, missing permissions, bad training, slow performance, incomplete data, or a broken process.

### High exports mean strong report adoption

Exports can indicate useful reporting or a workflow that immediately leaves Salesforce.

### One population should have one baseline

Different roles, regions, and process frequencies require different expectations.

### Current metadata explains historical behavior

Permissions, apps, navigation, reports, and roles change. Use point-in-time context when evaluating historical activity.

## THE ADOPTION INVESTIGATION METHOD

1. Define the intended behavior in business terms.
2. Identify the eligible population.
3. Select the event signals and business outcome.

4. Establish a pre-change baseline.
5. Segment by persona, cohort, channel, and experience quality.
6. Identify where users stop, repeat, fail, or leave Salesforce.
7. Validate the interpretation with user research and process owners.
8. Implement a focused change.
9. Measure the same behavior and outcome after the change.

#### ENTERPRISE EXAMPLE

A company launches a guided selling application and reports strong adoption because 85 percent of sellers log in weekly. Event data shows that only 32 percent open the new application, 18 percent complete the guided workflow, and most successful completions come from one pilot region. Search and navigation patterns show that many users cannot find the application, while page-performance data identifies slow load times for mobile users.

The correct response is not a broad retraining campaign. The evidence supports three targeted changes: improve navigation, address mobile performance, and replicate the pilot region's process and coaching. Adoption is then measured through activation, repeat completion, and opportunity-quality outcomes.

## AGENT AND DIGITAL-LABOR ADOPTION

Agentforce adoption requires a different denominator and outcome model. Measure eligible users and channels, sessions initiated, contained outcomes, human escalations, action success, rework, latency, and cost per successful result. A high session count can reflect curiosity or repeated failure. Pair session tracing and Agent Analytics with Event Monitoring evidence for API, Apex, named credentials, permissions, and downstream actions.

### WHAT GOOD LOOKS LIKE

A mature adoption program can explain who was expected to use a capability, what meaningful behavior occurred, whether the experience was successful, and which business outcome changed. It can distinguish lack of access, lack of awareness, friction, technical failure, and weak process value.

**KEY TAKEAWAYS**

- Define a legitimate operational purpose and privacy controls for behavior analysis.
- Separate access, engagement, effectiveness, and outcome.
- Use the eligible population as the denominator.
- Treat Lightning interaction and click evidence as directional, not complete audit evidence.
- Combine adoption with performance, errors, support, and business outcomes.
- Use event data to target improvements, then measure whether the intended behavior changed.

CHAPTER 08

# Using Event Monitoring for Governance

## Governance is a system of evidence and decisions

Salesforce governance is often documented as standards, review boards, and approval processes. Those controls matter, but they are not sufficient. Effective governance can show whether approved behavior occurred, whether exceptions were controlled, and whether the platform is drifting from its intended operating model.

Event Monitoring provides behavioral evidence. Metadata, Setup Audit Trail, permission history, deployment records, client registries, and business ownership provide the context. Together they turn governance from a periodic assertion into a continuous control system.

### THE CONTROL LIFECYCLE

#### 1. Define

State the control objective in clear operational terms. Example: "Only approved integration identities may extract customer data through Bulk API."

#### 2. Map evidence

Identify the event types, fields, metadata, identities, clients, data classifications, and approval records required to test the objective.

### 3. Monitor

Evaluate activity continuously or at the cadence required by the risk. Use Transaction Security where supported and where immediate intervention is appropriate.

### 4. Respond

Route exceptions to a named owner with severity, evidence, due date, and approved containment or remediation actions.

### 5. Attest

Retain proof that the control operated, exceptions were reviewed, and remediation was completed.

### 6. Improve

Review false positives, missed conditions, recurring exceptions, and changes in the platform or business process.

## THE SIX GOVERNANCE DOMAINS

### 1. Change Governance

Monitor deployments, package installation, sandbox activity, setup changes, permission updates, and configuration drift. Event Monitoring can contribute evidence, but Setup Audit Trail and metadata snapshots remain necessary because no single event source captures every configuration change.

High-value controls include:

- Changes outside approved release windows.
- Production changes with no work item or approver.
- Direct changes to protected profiles, permissions, clients, credentials, or security settings.
- Sandbox-to-production drift that invalidates testing assumptions.
- Package installation or upgrade without ownership and impact review.
- Emergency changes that remain undocumented after the incident.

### 2. Access Governance

Test whether actual activity aligns with approved role and purpose.

Examples include:

> - Privileged users who perform no administrative work but retain broad access.
> - Service users that log in interactively.
> - Agent users or integration users with unnecessary data access.
> - Sensitive exports by users outside the approved population.
> - Permission elevation without timely removal.
> - LoginAs use without support authorization.

Access review should combine entitlements with behavior. Unused privilege is still risk, while legitimate high-volume activity may be expected for a narrowly scoped service identity.

## 3. Integration and Client Governance

Maintain an approved registry of Connected Apps, External Client Apps, integration users, OAuth scopes, named credentials, endpoints, packages, agents, and owners. Compare observed API and callout behavior with the registry.

Governance findings include unknown clients, shared identities, unapproved resources, obsolete API versions, unused clients that remain authorized, unowned credentials, and agentic tools that appear without review.

## 4. Data-Usage Governance

Monitor how sensitive data is viewed, searched, reported, exported, accessed through APIs, or delivered to agents and external systems.

Controls should consider:

- Data classification and purpose.
- Approved user, client, agent, and channel.
- Volume and frequency relative to the role.
- Export, file, API, and report behavior.
- Retention and evidence requirements.
- Legal or privacy restrictions.

A governance control should not block normal work merely because data is sensitive. It should ensure that sensitive use is authorized, proportionate, and traceable.

## 5. Operational Governance

Define service objectives and ownership for integrations, automation, reports, user experience, monitoring, and critical business processes.

Useful controls include:

- Critical services with no current owner or service level.
- Repeated failures with no durable remediation.
- Monitoring gaps for high-impact processes.
- Alert queues without response ownership.
- Outdated runbooks or untested recovery procedures.
- Technical debt that repeatedly contributes to incidents.

## 6. Policy Enforcement

Transaction Security can enforce selected controls in real time through notification, blocking, or stronger authentication. Use it where the condition is well understood, the supported event is reliable for the purpose, and the operational effect has been tested.

Policy enforcement should include an exception process. An undocumented workaround is not an exception process.

---

### EXCEPTION MANAGEMENT

No enterprise control operates without exceptions. A mature exception record includes:

- Control and requirement being waived.
- Business justification.
- Scope, data, users, clients, and environments covered.
- Risk owner and approver.
- Compensating controls.
- Start date, expiration date, and review cadence.
- Evidence that the exception is being used as approved.
- Remediation or retirement plan.

Use event data to verify whether the observed behavior remains within the approved exception. Expired exceptions should automatically return to noncompliant status.

## MULTI-ORG GOVERNANCE

Large enterprises should standardize the control objective, evidence model, severity, and reporting while allowing justified local variation.

Cross-org comparison can reveal:

- One org with materially weaker authentication or session behavior.
- Different privileged-access patterns for the same role.
- Integration identities reused across business units.
- Inconsistent export, API, or agent controls.
- Different monitoring coverage and response performance.
- Local exceptions that have become permanent architecture.

Do not rank orgs by raw event volume. Normalize for users, transactions, business model, criticality, and feature footprint.

## AUDIT READINESS

Audit-ready governance can reproduce:

- The control objective and owner.
- The evidence sources and collection health.
- The version of the detection logic.
- Findings and exceptions during the audit period.
- Investigation and remediation records.
- Approval and closure evidence.
- Known limitations and gaps.

Retain the raw source evidence and point-in-time context. A current permission set does not prove what access existed six months earlier.

### ENTERPRISE EXAMPLE

A production permission set is updated to grant broad object access during a release. PermissionUpdate and Setup Audit Trail evidence show the change, while deployment records contain no approved item. Event data then shows that the affected integration user began querying the newly accessible object.

The governance finding is not only an unauthorized permission change. It is a control-chain failure involving release approval, access design, and observed use. Remediation must remove unnecessary access, determine whether data was transferred, correct the deployment process, and add a control that correlates critical permission changes with approved releases and subsequent activity.

### WHAT GOOD LOOKS LIKE

A mature governance program can state the control, show the evidence, identify the exception, assign the owner, prove the response, and demonstrate improvement across time and orgs. It uses Event Monitoring to test whether policy is operating in practice.

### KEY TAKEAWAYS

- Governance requires continuous evidence, not only written standards.
- Combine Event Monitoring with metadata, permissions, Setup Audit Trail, deployments, and ownership.
- Define controls in operational terms and map each to reliable evidence.
- Manage exceptions with scope, approval, expiration, and compensating controls.
- Normalize cross-org comparisons before drawing conclusions.
- Preserve point-in-time evidence so controls remain defensible in audits and investigations.

CHAPTER 09

# Connecting Event Monitoring with Metadata

## Behavior explains what happened. Metadata explains what it means.

A raw event can identify a user, request, page, report, API resource, or duration. It rarely contains enough information to determine business impact, ownership, data sensitivity, dependency, or root cause. Metadata and operational registries provide that context.

The highest-value Event Monitoring use cases therefore depend on a point-in-time relationship between behavior and the structure of the Salesforce environment.

### THE SIX CONTEXT LAYERS

#### 1. Identity and Permission Context

Enrich the actor with user status, type, license, role, profile, permission sets, permission-set groups, critical permissions, delegated administration, manager, business unit, and approved purpose.

This turns “user exported a report” into “a dormant contractor with temporary elevated access exported a regulated report outside the approved support window.”

Use point-in-time access where possible. Current permissions can differ materially from the permissions that existed when the event occurred.

## 2. Asset and Configuration Context

Map reports, dashboards, objects, fields, applications, pages, flows, Apex classes, named credentials, connected clients, packages, agents, actions, and other assets to their metadata definitions.

Useful attributes include:

- Owner and business purpose.
- Environment and lifecycle status.
- API version and last modified date.
- Description and documentation quality.
- Sensitivity classification.
- Managed-package namespace.
- Deployment or release reference.
- Criticality and support tier.

## 3. Dependency Context

A user-facing action often crosses multiple components. A page can invoke a Flow, which calls Apex, which uses a named credential, which calls an external service. A report can expose fields governed by several objects and sharing rules. An agent can select an action that invokes Flow, Apex, an API, or a downstream system.

A dependency graph helps answer:

- What is upstream and downstream of the event?
- Which component owner should respond?
- What else could be affected by the same failure or change?
- What is the blast radius of a permission, credential, or agent-action issue?

## 4. Business Context

Map the activity to the business capability, process, product, region, customer journey, control objective, service level, and business owner.

A technical event becomes operationally useful when it can be expressed as “case-routing latency is affecting premium-support customers” rather than “Apex entry point exceeded a duration threshold.”

## 5. Change Context

Correlate events with deployments, metadata changes, permission updates, package upgrades, Salesforce releases, data migrations, configuration changes, and approved maintenance windows.

Change correlation is especially valuable for performance and integration investigations. A new pattern immediately after a release is not proof of causation, but it is a strong hypothesis that should be tested.

## 6. Data-Sensitivity and Policy Context

Map reports, objects, fields, files, APIs, knowledge sources, retrievers, and agent actions to data classification, legal purpose, retention, geographic restrictions, and control requirements.

This allows a detection to distinguish high-volume use of public data from limited use of highly regulated data.

### FIVE HIGH-VALUE CORRELATION PATTERNS

**Sensitive report export**

**Event evidence:** Report or ReportExport, user, rows, format, time, session.

**Metadata context:** Report owner, report folder, underlying objects and fields, field sensitivity, eligible population, user permissions, approval or case context.

**Result:** A defensible data-exposure finding rather than a generic export alert.

**Slow Apex transaction**

**Event evidence:** ApexExecution, entry point, CPU, run time, SOQL, DML, callout, quiddity, request ID.

**Metadata context:** Class version, trigger or Flow dependencies, recent deployment, package namespace, business process, owner, related test failures.

**Result:** A prioritized root-cause hypothesis and accountable owner.

**Failing integration**

**Event evidence:** API or callout event, client, identity, resource, named credential, duration, status, rows, request ID.

**Metadata context:** Integration registry, endpoint, credential owner, OAuth scope, middleware flow, service level, business transaction, downstream status.

**Result:** Clear separation of authentication, Salesforce, middleware, downstream, and data-quality failure modes.

### Low feature adoption

**Event evidence:** App, page, interaction, report, search, mobile, and session behavior.

**Metadata context:** Eligible users, license, permissions, app navigation, feature assignment, training cohort, release date, performance, business outcome.

**Result:** A targeted intervention instead of a broad claim that users are resistant.

### Agent action risk

**Event evidence:** API client category, planner or bot identifiers, Apex, Flow, named credential, callout, request, and session-related activity.

**Metadata context:** Agent version, subagent, instructions, action registry, agent user, permissions, execution context, retriever, external credential, approval requirement, and business purpose.

**Result:** An explanation of the action blast radius rather than a count of agent requests.

## THE POINT-IN-TIME REQUIREMENT

Historical event analysis must use historical context. Names, ownership, permissions, classifications, dependencies, and configuration can change after the event.

At minimum, retain recurring snapshots or change history for:

- Users, roles, profiles, permission sets, and critical access.
- Connected and external clients, OAuth scopes, and integration identities.
- Reports, folders, objects, fields, classifications, and owners.
- Apex, Flow, pages, applications, packages, and API versions.
- Named and external credentials, endpoints, and ownership.
- Agent, subagent, action, prompt, retriever, model, and channel configuration.
- Production and sandbox metadata differences.

An as-of join should select the context effective at or immediately before the event, not the latest available record.

## THE NORMALIZED EVENT RECORD

A practical enterprise model contains:

### Source evidence

Org, event type, file ID, interval, sequence, API version, raw row, checksum, event timestamp, ingestion timestamp, and schema version.

### Normalized identity

User, user type, session, login key, client, device, network, bot, planner, and integration identity.

### Normalized action

Action family, resource, entity, operation, success or error state where supported, rows, bytes, duration, and request identifiers.

### Context

Business capability, owner, environment, permissions, sensitivity, asset, dependency, change window, criticality, and policy.

### Finding attributes

Detection rule or model, baseline, severity, confidence, evidence coverage, explanation, recommendation, owner, status, and outcome.

Keep the raw payload. Normalization should add consistency without deleting source-specific meaning.

## COMMON ENRICHMENT FAILURES

- Joining historical events to current permissions.
- Using mutable names instead of stable IDs.
- Losing context for deleted or renamed metadata.
- Assigning an owner from an outdated spreadsheet.
- Treating missing metadata as proof that an event is invalid.
- Over-enriching personal data without a defined purpose.
- Hiding uncertainty behind one composite risk score.
- Replacing raw fields with derived values that cannot be reproduced.

## THE EXPLAINABLE FINDING

A high-quality finding should be readable in this order:

Observation

What changed or occurred.

**Context**

Who, what, where, and which business service or sensitive data was involved.

Evidence

Source events, metadata, baseline, and correlated change.

Risk or impact

Why the condition matters and the degree of certainty.

Recommended action

What to validate, contain, remediate, and measure.

**Limitations**

Missing evidence, telemetry gaps, or assumptions that could change the conclusion.

### ENTERPRISE EXAMPLE

A spike in Apex execution time appears across several entry points. Metadata correlation shows that all of the affected paths invoke the same shared utility class, which was changed in the previous deployment. Dependency analysis identifies five business processes and two Agentforce actions that rely on the class. The event pattern is broad because the shared dependency is broad.

The finding can now prioritize the shared component, identify the deployment, quantify the affected processes, route the issue to the correct owner, and validate recovery across all dependent paths.

**KEY TAKEAWAYS**

- Event data becomes operational intelligence only after context is added.
- Enrich identity, assets, dependencies, business purpose, change, and data sensitivity.
- Preserve point-in-time history and use as-of joins for historical analysis.
- Keep raw evidence alongside normalized and derived fields.
- Use stable identifiers and explicit ownership registries.
- Explain findings through observation, context, evidence, impact, action, and limitations.

CHAPTER 10

# AI-Powered Detection

## AI should improve the decision, not hide the evidence

Event Monitoring creates a large and varied evidence base. AI can help teams discover patterns, connect related signals, summarize investigations, and prioritize what deserves attention. It cannot compensate for missing logs, unreliable schemas, weak ownership, or undefined response processes.

The correct progression is:

Trusted evidence → contextual detection → explainable analysis → bounded action → measured outcome.

AI belongs inside that progression, not above it.

### THE DETECTION LADDER

#### 1. Deterministic Rules

Use explicit logic when the condition is known and the consequence is clear. Examples include an unauthorized permission grant, an unapproved client, a missing critical integration run, or an action that violates a defined data-use policy.

Deterministic rules are easy to explain and test. They should remain the control of record for conditions that must be enforced consistently.

## 2. Contextual Rules

Add role, data sensitivity, client ownership, business calendar, deployment window, environment, and service criticality. Context allows the same event to be treated differently when the business purpose or exposure is different.

**Example:** a large export by an approved data-engineering integration is not equivalent to the same export by a dormant contractor from a new session.

## 3. Behavioral Baselines

Compare an actor, integration, page, report, action, or business service with its own history and a relevant peer group. Baselines are useful when normal behavior varies materially across populations.

A baseline should account for:

- Day and time.
- Seasonal and financial-calendar patterns.
- Release and maintenance windows.
- User role, client, process, and geography.
- Business transaction volume.
- Population changes and new-feature adoption.

A deviation is a reason to investigate, not proof of harm.

## 4. Multivariate Detection

Combine several weak signals into one stronger finding. Examples include:

> - New client + privileged identity + sensitive API access.
> - Permission elevation + unusual report export + off-hours session.
> - Rising retries + falling throughput + downstream latency.
> - Agent instruction change + quality decline + action failures.

Multivariate detection reduces the noise created by independent threshold alerts and produces a more coherent investigation.

## 5. Predictive and Risk Forecasting

Historical patterns can help identify deteriorating conditions before a formal failure occurs, such as increasing API retries, growing p95 Apex duration, declining adoption after releases, or a rising concentration of privilege.

Forecasts must disclose their horizon, uncertainty, training period, and factors. They should support planning and prioritization, not be presented as certainty.

## 6. AI-Assisted Investigation

Generative AI is most useful after a finding has been grounded in evidence. It can:

- Build a timeline across events, requests, sessions, and changes.
- Summarize affected users, assets, business services, and data.
- Cluster recurring exceptions or incident patterns.
- Compare the finding with historical incidents and approved baselines.
- Generate plausible root-cause hypotheses ranked by supporting evidence.
- Identify missing evidence and recommend the next query or validation step.
- Translate technical findings into business language.
- Draft a response checklist from an approved playbook.

A hypothesis must remain labeled as a hypothesis until the evidence confirms it.

## 7. Bounded Automation

Automation is appropriate when the condition is high-confidence, the response is approved, the action is reversible, and delay creates material risk.

Examples include routing a case, collecting additional evidence, notifying an owner, pausing a noncritical retry loop, or requiring an approval. Disabling a critical integration, revoking broad access, or blocking a customer process usually requires stronger policy controls and human authorization.

### THE FINDING CONTRACT

Every AI-assisted finding should contain a minimum contract.

Observation

What occurred or changed, expressed without speculation.

### Evidence

Source events, raw-data location, time range, schema, request and session identifiers, metadata context, and baseline.

Impact

The users, data, service, or outcome potentially affected.

### Confidence

How strongly the evidence supports the interpretation. Confidence is not severity.

### Severity

The consequence if the finding is accurate. A high-impact condition can have low confidence and still require rapid validation.

### Explanation

The factors that contributed to the detection, including counterevidence and alternative explanations.

### Recommendation

The approved next validation, containment, remediation, and measurement steps.

### Limitations

Missing events, incomplete context, model uncertainty, telemetry caveats, or assumptions.

### Provenance

The rule or model version, prompt or instruction version where applicable, execution time, and human review status.

## THE TRUST REQUIREMENTS

### Evidence grounding

Every factual claim should trace to governed evidence. A summary should not introduce entities, events, or causes that are absent from the source data.

### Abstention

When the evidence cannot support a conclusion, the system should say what is unknown and identify what would resolve it.

### Reproducibility

An analyst should be able to regenerate the finding from the retained evidence, detection version, and enrichment snapshot.

### Least data

Provide the model only the evidence required for the task. Remove secrets, credentials, unnecessary personal information, and unrelated customer data.

### Separation of analysis and action

The model can recommend an approved response. Policy and workflow should decide whether the response is executed.

### Human accountability

A named person or team owns material decisions. “The AI flagged it” is not an accountable operating model.

## THE DEPLOYMENT METHOD

1. Begin with a well-understood use case and trusted evidence.
2. Establish a deterministic or human-reviewed baseline.
3. Test on historical incidents, normal periods, seasonal peaks, and known false positives.
4. Run in shadow mode without changing production behavior.
5. Compare AI findings with expert review.
6. Calibrate confidence and severity separately.
7. Release to a limited population with explicit escalation rules.
8. Monitor drift after Salesforce releases, business changes, model changes, and new data sources.
9. Retest when detection logic, prompts, models, schemas, permissions, or metadata context changes.

### ENTERPRISE EXAMPLE

A model detects an unusual increase in failed outbound callouts. A simple anomaly alert would assign high severity because the volume is unprecedented. The AI-assisted investigation correlates the failures with one named credential, a specific Apex entry point, a recent deployment, and an increase in downstream response time. It also finds that successful business transactions are declining while retries are rising.

The evidence supports an integration degradation hypothesis, not a security incident. The system recommends validating the downstream service, pausing uncontrolled retries, reviewing the deployment, and reconciling unprocessed transactions. A human owner confirms the cause and initiates the approved recovery playbook.

The value of AI was not that it found an unusual count. It connected the event pattern to the likely service, change, and business impact while preserving the evidence and uncertainty.

### WHAT AI SHOULD NOT DO

- Infer malicious intent from one event.
- Declare a breach or root cause without supporting evidence.
- Replace deterministic policy for a clearly defined control.
- Generate production commands that bypass approval and rollback.
- Expose sensitive evidence to an unapproved model or user.
- Hide the factors behind one composite risk score.
- Continue producing an answer when the required evidence is missing.

**KEY TAKEAWAYS**

- Build AI on trustworthy collection, context, and ownership.
- Use deterministic rules for known controls and AI for complex pattern discovery and investigation.
- Separate confidence from severity.
- Require evidence, provenance, limitations, and abstention in every material finding.
- Evaluate unsupported claims, investigation speed, recommendation quality, and outcomes.
- Automate only bounded, approved, reversible actions.

CHAPTER 11

# Measuring Success and Realizing ROI

## Event Monitoring creates value only when it changes an outcome

The purpose of Event Monitoring is not to collect more activity data. It is to help the organization make better decisions about security, reliability, integrations, adoption, governance, and the value of Salesforce.

That is the central lesson of this guide. Logs are the evidence. Context gives the evidence meaning. A defined response turns that meaning into action. Measurement proves whether the action made Salesforce better.

### THE EVENT MONITORING VALUE CHAIN

| 01 | 02 | 03 | 04 | 05 | 06 |
|---|---|---|---|---|---|
| **Collect** | **Contextualize** | **Detect** | **Act** | **Improve** | **Prove** |
| Complete, timely, trusted evidence with proven coverage. | Identity, permission, asset, dependency, and business context. | Material, explainable findings with owners and severity. | Contain, remediate, and close through a defined playbook. | Fix the underlying cause so the condition does not return. | Measure the outcome and improve the next decision. |

FIGURE 2 — THE EVENT MONITORING VALUE CHAIN. VALUE IS CREATED ONLY WHEN EVERY LINK HOLDS.

A successful program follows a simple sequence:

### Collect

Capture the right evidence with dependable coverage, retention, schema validation, and monitoring of the collection process itself.

### Contextualize

Connect events to users, permissions, clients, metadata, integrations, data sensitivity, business services, ownership, and recent change.

### Detect

Identify conditions that are material, explainable, and actionable. Use rules for known controls, baselines for changing behavior, and AI only where it improves analysis without hiding the evidence.

### Act

Route the finding to a qualified owner with the information and authority required to validate, contain, remediate, and recover.

### Improve

Use the outcome of each investigation to strengthen the control, architecture, process, training, or monitoring logic.

### Prove

Measure whether risk declined, service improved, adoption increased, work was reduced, or the Salesforce investment produced greater value.

If any link is missing, the program loses value. Collection without context creates noise. Detection without ownership creates a backlog. Response without measurement creates activity without proof.

## WHAT SUCCESS LOOKS LIKE

A mature Event Monitoring program can answer five questions consistently:

1. Can we trust the evidence?

The organization knows which critical services and risks are observable, whether the data is complete and current, and where limitations remain. Missing files, schema changes, delayed telemetry, and incomplete context are visible rather than hidden.

1. Are the findings worth acting on?

Findings explain what happened, why it matters, who or what is affected, how confident the interpretation is, and what should happen next. The team is not overwhelmed by disconnected threshold alerts.

1. Can the organization respond effectively?

Every material finding has an accountable owner, an expected response time, a tested playbook, and clear closure evidence. Security, platform, integration, identity, and business teams know their role.

1. Do the same problems keep returning?

The organization distinguishes temporary recovery from root-cause remediation. Repeated failures, recurring access issues, persistent adoption friction, and unresolved technical debt decline over time.

1. Are business outcomes improving?

Event Monitoring is connected to measurable outcomes such as reduced exposure, faster investigation, more reliable integrations, better user experience, stronger adoption, less manual work, and more defensible Salesforce spending decisions.

## A SMALL SCORECARD IS ENOUGH

The closing scorecard should be simple enough to review and strong enough to guide investment. Eight measures cover most enterprise programs:

### Evidence coverage

The percentage of critical Salesforce services and risks supported by trusted, current telemetry and context.

### Actionable finding rate

The percentage of findings that include sufficient evidence, a qualified owner, and a clear next action.

### Time to validate

How quickly the team determines whether a material condition is real, benign, or still unresolved.

### Time to contain or restore

How quickly exposure is reduced or the affected business service returns to an acceptable operating state.

### Recurrence rate

How often a previously closed condition returns. This is one of the clearest signals that remediation addressed only the symptom.

### Critical-service reliability

Whether the Salesforce processes and integrations that matter most are completing successfully, within the expected performance and freshness objectives.

### Adoption of priority capabilities

Whether the intended users are activating, repeatedly using, and successfully completing the workflows that justify the investment.

### Realized value compared with program cost

Whether measured improvements in labor, reliability, adoption, compliance, and operating efficiency exceed the cost required to produce them.

These measures should be segmented by use case and business service. One enterprise-wide average can hide a high-performing security program and a weak integration-monitoring practice.

## MEASURING ROI WITHOUT OVERSTATING IT

Event Monitoring often sits inside a significant Salesforce investment. The most credible ROI story is not that every alert prevented an incident. It is that the organization used evidence to improve outcomes that can be observed and valued.

The strongest benefit categories are:

### Faster investigation and response

Measure the reduction in time spent locating logs, identifying owners, correlating activity, validating impact, and assembling evidence.

### Reduced service disruption and rework

Measure processing delays, failed transactions, repeated incidents, reconciliation work, support effort, and user impact before and after remediation.

### Improved adoption and utilization

Measure activation, repeat use, workflow completion, performance, and whether premium licenses or capabilities are assigned to populations that receive value from them.

### Stronger security and governance

Measure exposure duration, unauthorized privilege, unapproved clients, sensitive-data use outside policy, overdue remediation, and audit evidence effort.

### Better investment decisions

Measure spending avoided or redirected because the organization can distinguish what is heavily used, underused, redundant, risky, or operationally critical.

The full program cost should include the relevant Salesforce entitlement, collection and analytics tooling, storage, engineering, administration, security operations, and ongoing review. Ignoring the operating cost makes the value claim less credible.

A simple ROI calculation is:

> **(Validated annual benefit − annual program cost) ÷ annual program cost**

The discipline is in the word validated. Separate benefits into three categories:

### Realized value

A measured improvement supported by operational or financial evidence.

### Evidenced estimate

A calculation based on measured volume and an approved unit value, with the assumptions disclosed.

### Opportunity

A potential benefit that has been identified but will not exist until the organization completes the remediation or adoption work.

Do not combine all three into one headline number. Clear labeling builds more trust than an inflated estimate.

Be equally careful with avoided-loss claims. “Incidents prevented” is difficult to prove because the organization cannot observe the outcome that did not occur. More defensible statements include:

- A high-risk action was blocked by an approved control.
- Exposure was contained before additional activity occurred.
- A known failure pattern was detected earlier than the previous baseline.
- A recurring incident stopped after root-cause remediation.
- Time at risk or time out of service declined.

## THE PRACTICAL PATH FORWARD

The guide has covered a wide range of Event Monitoring capabilities, but an enterprise does not need to operationalize every event type at once.

Begin with five to ten business capabilities whose failure, abuse, or underuse would matter materially. For each one:

- Define the business outcome and accountable owner.
- Identify the event and contextual evidence required.
- Establish the current baseline.
- Decide whether the condition requires immediate control, rapid investigation, or trend analysis.
- Create one clear finding and response path.
- Measure whether the response improved the original outcome.

This approach produces depth before breadth. Five well-owned monitoring use cases create more value than fifty dashboards that no one is expected to act on.

As the program matures, expand in the same order described throughout this guide:

Reliable collection

1. Then standardized investigation.
2. Then actionable detection and response.
3. Then contextual baselines and cross-org intelligence.

Then evidence-grounded AI assistance where it measurably improves speed, accuracy, or coverage.

## THE ROLE OF COENNECT

Salesforce provides the Event Monitoring data and native controls. Coennect helps enterprises turn that evidence into continuous operational intelligence by combining event activity with metadata, permissions, integrations, ownership, usage, change history, and business context across production orgs and sandboxes.

The value is not another place to store logs. It is a clearer path from activity to understanding, from understanding to action, and from action to measurable improvement.

## THE FINAL TAKEAWAY

Event Monitoring becomes a high-value Salesforce investment when the organization stops treating it as a set of files and begins treating it as an operating discipline.

The complete model is simple:

1. Know what matters.
2. Collect the right evidence.
3. Add the context required to interpret it.
4. Create findings people can act on.
5. Fix the underlying problem.
6. Measure the outcome.
7. Use the result to make the next decision better.

That is how Event Monitoring moves beyond compliance and troubleshooting. It becomes a practical system for running Salesforce with greater security, reliability, clarity, and confidence.

ABOUT COENNECT

# From activity to understanding. From understanding to action.

Salesforce provides the Event Monitoring data and the native controls. Coennect helps enterprises turn that evidence into continuous operational intelligence — reliable collection, contextual enrichment, explainable findings, and measured outcomes across every org.