COENNECT



COENNECT GUIDE

Managing Salesforce in the Agentic Era

A practical guide to reducing repetitive work, finding blind spots, and making better decisions

CONTENTS

What is inside

01	Why routine operations consume so much attention	p. 04
02	What changes when agents take on the preparation	p. 06
03	Keep the work connected	p. 07
04	Give the work the context it needs	p. 09
05	Apply the approach to familiar Salesforce questions	p. 11
06	Give agents useful scope and clear limits	p. 13
07	Know whether operations are improving	p. 14
08	Keep this reference: what a useful AI investigation should tell you	p. 15
09	Where the answer lives: a Salesforce evidence map	p. 16
10	References and source notes	p. 31
11	About Coennect	p. 33

INTRODUCTION

Your Salesforce team is expected to keep the platform reliable, secure, and useful while continuing to deliver improvements. That means understanding how the org works, recognizing when something changes, and knowing where attention is needed before every issue becomes another interruption.

WHAT YOU WILL TAKE FROM THIS GUIDE

- Recognize operational work that AI can help carry, including recurring investigations and neglected reviews.
- Understand how configuration, changes, permissions, activity, and business context fit together.
- Know what useful agent output looks like and how to judge whether your team is gaining capacity and better coverage.

Much of the effort sits between receiving a question and being ready to answer it. Someone has to locate the relevant configuration, compare changes, trace permissions, review activity, and ask the business what should have happened. The same preparation returns with the next incident, access review, or release.

AI agents create an opportunity to make that preparation more consistent and to extend attention to areas that rarely receive it. With appropriate access and maintained context, an agent can investigate a signal, connect the evidence, and bring the team a clearer decision to review.

The aim is to give admins more time for decisions and improvements, architects a clearer view of dependencies and impact, and platform leaders a better understanding of where operational attention is needed.

01

Why routine operations consume so much attention

An enterprise org contains the accumulated decisions of many teams and releases. Some decisions are documented. Others live in configuration, old tickets, deployment notes, or the memory of the person who built a process. As the org changes, those explanations become harder to keep aligned.

This creates two different demands on a Salesforce team.

Work that keeps repeating

An admin investigates another Flow failure, explains why a user still has a permission, or checks what a release could affect. The question may be new, but much of the preparation is familiar: finding the right sources, establishing what changed, and separating relevant evidence from background activity. Each interruption can pull the team away from planned improvements.

Work that receives too little attention

Some issues never arrive as a clear request. Repeated errors may be handled one ticket at a time. An integration may become less reliable without stopping completely. Permissions may outlast their original purpose. A slower page or an awkward process may cause users to work around Salesforce instead of reporting a problem.

> These are operational blind spots: areas where a useful question is not being asked consistently, or where the evidence is too scattered to answer it efficiently. Closing them requires ongoing attention as well as good incident response.

A familiar symptom reveals the hidden work

> **EXAMPLE**
> A sales rep reports that an Opportunity will not save as Closed Won.

The admin must establish who is affected, what changed, which configuration governs the save, and whether the problem is isolated or recurring. The error message helps, but the explanation may also depend on permissions, the form being used, validation rules, automation, and recent changes.

While the admin reconstructs that picture, the rep waits, another team may be pulled into the investigation, and planned work is interrupted. Even if the eventual correction is small, reaching a trustworthy explanation can require substantial coordination.

The broader problems are recognizable

- **Disconnected evidence:** configuration, changes, errors, and user reports must be assembled before someone can reason about the issue.
- **Unclear scope:** a single ticket does not show whether other users or related processes are affected.
- **Lost learning:** the next investigator may repeat the same discovery if the explanation and its supporting evidence are not retained.

A better operating approach addresses those gaps throughout the life of the org. It helps the team arrive at a decision with more context already available.

02

What changes when agents take on the preparation

The opportunity begins with work that repeats across different questions: collecting approved evidence, comparing versions, grouping related events, tracing contributing access, and maintaining explanations of how a process works.

These activities can be time-consuming even when the final decision still needs an experienced person.

An AI operations agent is software given an objective, access to selected evidence, and a defined set of actions. Within that scope, it can work through multiple investigation steps and return its findings. For example, it could review recurring errors, compare them with relevant configuration changes, and identify the cases that deserve an admin's attention.

This requires more than asking an AI model a question. The operating process needs a trigger or review cadence, access to current sources, a place to retain useful findings, and a person responsible for the result. A conversation can assist an individual investigation; a recurring responsibility also needs those arrangements to continue between conversations.

Conventional automation still has a useful role. A known threshold or a fixed policy check can run as a rule. AI becomes useful when the next step depends on interpreting several sources, comparing plausible explanations, or explaining why a pattern matters.

> **WHO DOES WHAT**
>
> The platform owner or an authorized admin defines the responsibility and assigns the agent specific work. The business owner contributes the intended outcome and legitimate exceptions. The agent brings back evidence, an explanation, and a proposed next step. The team can then spend more attention on the decisions that require its judgment.

For an existing investigation, the benefit may be less manual preparation. For an area nobody reviews regularly, the benefit may be newly established coverage. Both matter, but they should be evaluated differently.

03

Keep the work connected

A useful operating model links understanding, monitoring, decisions, action, and learning. If those activities remain disconnected, a team can gain more alerts without gaining a clearer view of what to do.

Use five stages to keep that continuity: Understand, Observe, Prioritize, Deliver, and Optimize. Each stage should leave something useful for the next one.



Understand

Build an explanation of the process or capability under review. Connect the relevant configuration, dependencies, permissions, known integrations, and business purpose. Include who owns it and which assumptions still need confirmation.

Useful output: A current explanation of how the process works, with links to the evidence and visible gaps.

Observe

Watch the available activity and changes for signs that the process is behaving differently. Useful signals include recurring errors, unexpected access changes, unusual API activity, deteriorating performance, and changes in usage.

Useful output: A meaningful change or recurring pattern, with its scope, timing, and evidence coverage.

Prioritize

Decide why the finding deserves attention. Consider the business process affected, the people or integrations involved, recurrence, available workarounds, and confidence in the explanation. A high event count alone does not establish urgency.

Useful output: A reason to act, investigate further, or continue observing, with a named owner.

Deliver

Produce the work needed for that decision. It may be a supported explanation, an access review, a proposed correction, a change-impact assessment, or updated documentation. Where execution is authorized, the work can include the agreed action.

The output should be specific enough that the next person can use it without repeating the investigation. Explain what is proposed, why it fits the evidence, who or what it could affect, and what remains unresolved.

Useful output: A reviewable result or completed authorized action, with evidence and a clear next step.

Optimize

Confirm the intended outcome, learn from the results, and make improvements where needed. Check whether the business task works, whether the recurring problem has changed, and whether the team's explanation was accurate.

Carry that learning back into the operating process. Update documentation, refine a detection that produced noise, retain a newly understood dependency, or revise the agent's investigation instructions. The next run should benefit from what the team has learned.

Useful output: An evidenced outcome and retained learning that improves future operations.

> **THE AGENTIC CENTER OF EXCELLENCE**
>
> These stages support an agentic Center of Excellence: an operating model in which people and agents share defined work around the platform. It does not require a separate department. Its value comes from connecting the work so that an observation can lead to a decision, an action, and a better understanding of the org.

04

Give the work the context it needs

An agent cannot reliably explain your org from general Salesforce knowledge alone. It needs evidence about your configuration and activity, plus the business meaning that makes those facts relevant.

Start with the question being investigated. For an access concern, grant paths and business need matter most. For a recurring failure, configuration, execution evidence, and changes become central. Extend the context as the question requires it.

CONTEXT	EVIDENCE TO CONNECT	WHAT IT HELPS THE TEAM UNDERSTAND
Configuration and dependencies	Fields, validation rules, Flows, Apex, pages, and available metadata relationships.[1]	What governs the behavior and which related components may be affected.
Changes	Setup Audit Trail, release records, and retained configuration versions.[2]	What changed around the issue and which changes need examination.
Permissions	Profiles, permission sets, permission set groups, assignments, and relevant access settings.[3]	How a capability is granted and who else a proposed change might affect.
Activity and execution	Available Event Monitoring Logs, API activity, and relevant error evidence.[4] [5] [6]	What was recorded, where failures or unusual patterns concentrate, and which paths need investigation.
Usage and performance	Page and interaction activity, performance measures, and the relevant user population.[7]	Whether an experience changed and where users may be encountering friction.
Business purpose and ownership	Process documentation, requirements, prior decisions, known exceptions, and owner input.	Why the process exists, which outcomes matter, and whether observed behavior is expected.
Evidence quality	Source timestamps, collection status, supported events, and known gaps.[4] [8]	How current and complete the evidence is, and what cannot yet be concluded.

> **CONNECT THE SOURCES**
>
> The value comes from the connections. A change record helps identify something worth examining. Configuration shows how that change could affect behavior. Activity shows what was recorded afterward. A process owner explains whether the resulting behavior meets the business need. None of those sources supplies the entire explanation on its own.

Keep facts, interpretations, and business decisions distinct. A permission assignment is a fact. Whether it explains an unexpected capability requires tracing the relevant grants. Whether the capability should remain is a business decision. Similarly, a dependency relationship identifies a connection; it does not guarantee a complete impact assessment.

Salesforce APIs provide ways to retrieve available evidence. Event Monitoring Logs and Setup Audit Trail are sources with different purposes. Event logs describe supported recorded activity; Setup Audit Trail records supported setup changes. The retrieval mechanism does not create information that the source did not capture.[4] [2] [8]

> **FRESHNESS MATTERS**
>
> Event log files can arrive after the activity they describe, and collection can have gaps. Available event types and retention depend on the org's subscription and settings. Before interpreting a decline or an absence, establish that the relevant evidence was actually available for the period.[4] [8]

Retain the explanation with its sources, owner, and relevant change history. When a dependency, permission, or business requirement changes, update the affected context. This reduces reliance on individual memory and gives the next admin a more useful starting point.

See the **Salesforce Evidence Map** at the end of this guide for source locations, example queries, retention windows, and the limits of the available evidence.

05

Apply the approach to familiar Salesforce questions

The same approach can support several parts of the admin's workload. Each situation below begins with a recognizable question and ends with information the team can use.

Why does the same automation keep generating tickets?

A Flow failure is investigated, the immediate user is helped, and a similar ticket arrives later. The team needs to know whether the failures share a cause, whether they are increasing, and whether the same part of the business is repeatedly affected.

An agent can organize the available error evidence by the relevant automation, version, timing, and error details, then compare the pattern with configuration, dependencies, change history, and prior incidents. Screen flow error events can identify a Flow and version; other automation paths need the evidence appropriate to those executions.[5]

Useful output identifies the recurring pattern, the affected scope supported by the evidence, and the leading explanation or missing fact. The admin receives an investigation to assess, while the platform owner can see a reliability problem that separate tickets might conceal.

Is an API spike a problem or normal business activity?

An increase in API activity can reflect legitimate demand, a new integration, or repeated attempts around a failure. Volume is a reason to investigate, but it does not by itself establish the cause or business impact.

The agent can compare available API activity across time, users or integration identities, and relevant event details. Where the source includes errors or duration, those fields help separate increased demand from deteriorating behavior. Related changes and authorized external integration evidence can add context.[4] [8]

The useful result explains where the pattern concentrates, what changed, which owner should review it, and what remains unknown outside Salesforce. This helps the admin direct the investigation instead of treating every spike as an incident or passing an unexplained chart to the integration team.

Why does someone still have a sensitive capability?

A user changes responsibilities, but a permission review shows access that may no longer fit the role. The admin must explain how that access is granted before proposing a change.

The agent can trace the relevant profile, permission sets, groups, and assignments, then compare them with recorded changes and documented exceptions. Removing one assignment may leave another grant in place, and changing a shared permission definition can affect other users. Those distinctions belong in the explanation.[3]

The business owner confirms whether the capability is still needed. The admin receives the contributing grants, the scope of any proposed adjustment, and the unresolved exceptions. Repeating this review when relevant access changes occur can reveal issues between scheduled reviews, provided those changes are captured and available.

Are users adopting the process, or working around it?

Activity in a service workspace falls. A training request may seem reasonable, but the same signal could reflect changed navigation, a smaller active population, slower pages, or a process users find difficult to complete.

The agent can compare usage with page and interaction activity, performance signals, errors, and recent navigation changes. Salesforce provides page-performance measures such as Experienced Page Time, and supported Event Monitoring events can add evidence about the experience.[7]

The process owner adds the missing business perspective: Are cases being completed on time? Is rework increasing? Are users moving work outside the intended process? A useful investigation distinguishes a measurement change from an actual obstacle, helping the team choose between improving the experience, correcting the measure, or providing targeted support.

06

Give agents useful scope and clear limits

Once the team understands the work and its evidence, delegation becomes a practical choice. An admin or platform lead assigns a responsibility within their authority, such as reviewing recurring automation failures or preparing an explanation of sensitive access changes.

Begin with a result the team would find useful. The instruction should describe the process or population to review, the evidence the agent can use, the output expected, and when a person needs to decide what happens next.

- **Investigate and explain:** Collect authorized evidence, connect relevant facts, identify gaps, and return findings. This can reduce preparation even while every change remains with the team.
- **Prepare a decision:** Propose a correction or access adjustment, explain likely impact, and identify the checks needed to assess it. The reviewer should be able to understand the recommendation from the evidence.
- **Carry out agreed work:** Where appropriate, allow a specific approved action or a narrowly defined recurring action. Its scope, execution access, stop conditions, and recovery responsibility must be clear.

Authority should match the action. Reading configuration has different consequences from changing a shared permission set or production automation. Testing should address the uncertainty in the proposed change and the users or processes it could affect. These boundaries make the result easier to trust and the review easier to perform.

The human owner remains accountable. Business owners decide what the process should achieve and which exceptions are legitimate. Admins and architects use the returned work to make and implement the appropriate technical decisions. Roles can follow the team's existing operating model.

Useful progress does not depend on granting broad autonomy. An agent that consistently brings the right evidence and a clear explanation can already change how much effort the team spends getting ready to act.

07

Know whether operations are improving

Evaluate the change through the team's work and the business outcome. A growing count of alerts, summaries, or completed agent runs says little on its own.

Use questions that make the improvement visible:

- **Are investigations easier to complete?** Compare the effort needed to gather evidence, establish scope, and reach a supported decision for similar cases.
- **Is the returned work trustworthy?** Check whether explanations link to current sources, acknowledge gaps, and survive review. Include corrections and rejected recommendations in the assessment.
- **Are recurring problems receiving more consistent attention?** Look for patterns that now receive an owner and follow-up instead of remaining scattered across tickets or logs.
- **Are blind spots becoming visible?** Identify access changes, errors, performance trends, or neglected processes now being reviewed. This is additional coverage even when no earlier time-saving baseline exists.
- **Does the business task work better?** Check whether users can complete the intended work, integrations deliver the expected result, or access fits the person's responsibilities. Activity counts are supporting evidence.
- **Is the team gaining usable capacity?** Count preparation, review, correction, and follow-up. Notice whether the capacity becomes time for improvements, better documentation, or deeper investigation.
- **Does each cycle improve the next one?** Look for context that stays current, fewer repeated discovery steps, better detections, and fewer unsupported assumptions.

Compare similar work over a suitable period and account for changes in business volume or user population. An investigation that finishes faster but requires extensive rechecking may not save effort. A newly monitored area may create valuable coverage before it reduces workload.

> **THE STRONGEST EVIDENCE**
>
> It is a combination: the team spends less effort reconstructing the situation, important questions receive more consistent attention, and the intended business work is completed correctly.

08

Keep this reference

What a useful AI investigation should tell you. Use this as a reading guide when an agent brings your team a finding. A useful result should make the next decision easier to understand.

WHAT YOU SHOULD RECEIVE	WHAT IT SHOULD TELL YOU
A clear problem	What changed or failed, which process is involved, and who or what is affected. The scope should match the evidence.
Evidence you can inspect	The relevant configuration, events, changes, and source dates, with enough detail to check the explanation.
An explanation of the connections	How the evidence relates to the symptom and business process, which explanation it supports, and which alternatives still matter.
Visible uncertainty	What is missing, stale, or untested, and how that limits the conclusion. A gap should lead to a useful next question.
A practical next step	What should happen next, why, who owns the decision, and whether the agent is recommending or carrying out an action.
A way to judge the outcome	What evidence would show that the intended work succeeded, and what should be retained for the next investigation.

"Errors increased after a release" is a starting observation. A useful investigation explains which errors, which part of the org, what changed, what supports the explanation, and what decision is now possible.

A practical place to begin is a question your team already struggles to answer consistently. Use the evidence available for that question and judge whether the returned work improves the decision. That creates a concrete basis for expanding the approach.

Over time, the opportunity is an org the team understands more clearly, operational issues that receive more consistent attention, and expertise available for the decisions and improvements that matter most.

Use the Salesforce Evidence Map that follows to locate the information behind a question, understand its limits, and decide what needs to be preserved for the next investigation.

09

Where the Answer Lives: A Salesforce Evidence Map

How to use this map

Use this map to locate the evidence behind 21 common Salesforce questions. Start with the quick reference, then use the matching detail entry for the Setup path, an example query to adapt, and the limits of what the evidence can establish.

Availability and retention depend on your edition, licenses, permissions, and settings. "None" under extra license means no paid add-on beyond a supported edition.

Quick reference

QUESTION YOU GET ASKED	WHERE THE ANSWER LIVES	HOW LONG IT'S KEPT	EXTRA LICENSE
Who has a specific permission?	Profiles, permission sets, and permission set groups	Current state only	None
Can this user see or edit this record?	UserRecordAccess, record sharing	Current state only	None
Who logged in, when, and from where?	Login History	6 months	None
Who hasn't logged in recently?	User LastLoginDate	Most recent login only	None
Which connected apps have access?	OauthToken, Connected Apps OAuth Usage	Current state only	None
Who changed this setting?	Setup Audit Trail	180 days	None
Who changed this field value?	Field History Tracking	Up to 18 months in UI; 24 via supported API	None; add-on for archives
What depends on this component?	Where is this used?, dependency API (beta)	Current state only	None
What was deployed, and when?	Deployment Status, DeployRequest	About 30 days	None
How does production differ from the full sandbox?	Metadata retrieved from each org	The moment you retrieve it	None
Why is this integration failing?	Integration logs, Bulk Data Load Jobs, debug logs, API event types	Bulk results 7 days; debug 1 or 7 days; event logs vary	Event Monitoring

Quick reference, continued

QUESTION YOU GET ASKED	WHERE THE ANSWER LIVES	HOW LONG IT'S KEPT	EXTRA LICENSE
What is using our daily API limit?	System Overview, API Usage Last 7 Days, API Total Usage events	24 hours, 7 days, 1 day free	None
Why did this flow fail?	Flow error emails, failed flow interviews	Your mailbox; the failed list is short-term	None
What happened inside this transaction?	Debug logs	System logs 24 hours; monitoring logs 7 days	None
Did this batch or scheduled job succeed?	Apex Jobs, AsyncApexJob	Short-term	None
Is anyone using this report?	Report LastRunDate, Report events	Latest run; event retention varies	Event Monitoring
Which pages and apps are used?	Lightning Usage app	Daily rollups; confirm available history	None
Are licenses going unused?	Company Information, User LastLoginDate	Current state only	None
Who deleted this record?	Recycle Bin, Setup Audit Trail for mass deletes	Normally 15 days in Recycle Bin	None
Who exported this report or data?	Report Export and Report events, Setup Audit Trail	Event retention varies; audit trail 180 days	Event Monitoring
Who viewed this sensitive record?	Real-Time Event Monitoring	Up to 6 months when stored	Event Monitoring

How long the evidence lasts

Source notes: M1, M2, M3, M4, M7, M8, M9, M10. Windows depend on the conditions described in the entries.

Permissions, sharing, dependencies, and configuration primarily describe current state. Setup Audit Trail records supported changes, but it does not preserve a complete earlier org. Retain snapshots when historical comparison matters. [M1]

ACCESS AND SECURITY

Salesforce can show who has access today. Establishing who had access on a past date often requires retained snapshots and available change records.

Who has a specific permission, such as Export Reports or Modify All Data?

WHERE IT LIVES: Profiles, permission sets, and permission set groups. PermissionSetAssignment provides a starting point for inspecting grants. For one user, use the User Access Summary to inspect the effective access.

HOW LONG IT'S KEPT: Current state only. Assignment changes appear in Setup Audit Trail for 180 days.

LICENSE: None.

HOW TO GET IT: For one user, open the user record and click View Summary. For everyone with a permission, run:

```
SELECT Assignee.Name, Assignee.IsActive, PermissionSet.Label,
       PermissionSet.IsOwnedByProfile, PermissionSet.PermissionSetGroupId
FROM PermissionSetAssignment
WHERE PermissionSet.PermissionsExportReport = true
```

Swap in a supported permission field, such as PermissionsModifyAllData or PermissionsViewAllData.

WATCH OUT: Profile grants appear as permission sets with IsOwnedByProfile = true, so one user can show up on several rows. A permission set group can mute a permission, so confirm the final result in the user's access summary before acting. [M14]

Can this user see or edit this record, and why?

WHERE IT LIVES: The UserRecordAccess object, the record's sharing details, and the Object Access Summary in Object Manager.

HOW LONG IT'S KEPT: Current state only.

LICENSE: None.

HOW TO GET IT:

```
SELECT RecordId, HasReadAccess, HasEditAccess, MaxAccessLevel
FROM UserRecordAccess
WHERE UserId = '005XXXXXXXXXXXXXX'
  AND RecordId = '001XXXXXXXXXXXXXX'
```

WATCH OUT: UserRecordAccess tells you whether a user has access, not which sharing rule, role, or team grants it. It also ignores restriction rules, so a user it reports as having access can still be blocked. [M15]

Who logged in, when, from where, and how?

WHERE IT LIVES: Login History in Setup, and the LoginHistory object.

HOW LONG IT'S KEPT: Six months. The Setup page shows up to 20,000 recent attempts; download the CSV for the full six months.

LICENSE: None.

HOW TO GET IT: From Setup, enter Login History in Quick Find. Or query:

```
SELECT UserId, LoginTime, SourceIp, LoginType, Status, Application
FROM LoginHistory
WHERE LoginTime = LAST_N_DAYS:30
ORDER BY LoginTime DESC
```

WATCH OUT: Attempts made with a username that doesn't exist never appear, because there is no user to attach them to. Login Type separates browser, API, and OAuth logins, which matters when an integration user looks dormant. [M2]

Which users have access but haven't logged in recently?

WHERE IT LIVES: The LastLoginDate field on the User object.

HOW LONG IT'S KEPT: Current state only. The field holds the most recent login, not a history.

LICENSE: None.

HOW TO GET IT:

```
SELECT Name, Username, Profile.Name, LastLoginDate
FROM User
WHERE IsActive = true
  AND (LastLoginDate < LAST_N_DAYS:90 OR LastLoginDate = null)
```

WATCH OUT: Inactivity is not proof that access is unneeded. Seasonal users, backup approvers, and integration users can all look dormant and still be required. [M22]

Which connected apps have access, and are they still used?

WHERE IT LIVES: Connected Apps OAuth Usage in Setup, and the OauthToken object.

HOW LONG IT'S KEPT: Current state only. Each token shows its last use and total use count, not a log of each use.

LICENSE: None. Seeing every user's tokens requires the Customize Application permission.

HOW TO GET IT:

```
SELECT AppName, User.Name, LastUsedDate, UseCount
FROM OauthToken
ORDER BY LastUsedDate
```

WATCH OUT: Salesforce recommends splitting this query by user when an org has more than 2,500 tokens, or results can be incomplete. A token shows that access was granted and used, not what the app did. [M16]

CHANGES AND CONFIGURATION

Change history in Salesforce records who and when far more reliably than what and why.

Who changed this setting, and when?

WHERE IT LIVES: Setup Audit Trail, and the SetupAuditTrail object.

HOW LONG IT'S KEPT: 180 days, then deleted. The Setup page shows only the 20 most recent changes; download the CSV for the rest.

LICENSE: None. Viewing it requires View Setup and Configuration.

HOW TO GET IT: From Setup, enter View Setup Audit Trail in Quick Find. Or query:

```
SELECT CreatedDate, CreatedBy.Name, Action, Section, Display, DelegateUser
FROM SetupAuditTrail
WHERE CreatedDate = LAST_N_DAYS:30
ORDER BY CreatedDate DESC
```

WATCH OUT: It records supported changes, the actor, and a short description. It does not preserve a full before and after configuration or the business reason for a change. DelegateUser can identify a delegated actor. [M1]

Who changed this field on a record, and what was the old value?

WHERE IT LIVES: Field History Tracking, stored in each object's history table, such as AccountHistory or Customer_Profile__History.

HOW LONG IT'S KEPT: Up to 18 months in the UI and up to 24 months through supported API retrieval. Field Audit Trail can retain archived history until deleted and increases the field limit.

LICENSE: None for standard tracking, which covers up to 20 fields per object.

HOW TO GET IT:

```
SELECT Field, OldValue, NewValue, CreatedBy.Name, CreatedDate
FROM AccountHistory
WHERE AccountId = '001XXXXXXXXXXXXXX'
ORDER BY CreatedDate DESC
```

For a custom object, query its __History table and filter on ParentId. To retrieve history older than 18 months, use SOAP queryAll() or Data Loader's Export All with Bulk API disabled.

WATCH OUT: History starts only when tracking is turned on for that field. Nothing is recorded retroactively, and some field types, such as formula and roll-up summary fields, can't be tracked. [M7]

What depends on this field, flow, or class before I change it?

WHERE IT LIVES: The Where is this used? button on a custom field, and the MetadataComponentDependency object (beta) in the Tooling API and Bulk API 2.0.

HOW LONG IT'S KEPT: Current state only.

LICENSE: None. The dependency object is a beta feature.

HOW TO GET IT: Open the field in Object Manager and click Where is this used? For a list you can work with, query the Tooling API:

```
SELECT MetadataComponentName, MetadataComponentType
FROM MetadataComponentDependency
WHERE RefMetadataComponentId = '00NXXXXXXXXXXXXXX'
```

WATCH OUT: Tooling API queries return at most 2,000 rows and exclude reports; Bulk API 2.0 includes reports and allows up to 100,000. Neither catches every reference, such as field names built dynamically in Apex. Treat the result as a starting list, not proof that nothing else depends on the component. [M11]

What was deployed, and when?

WHERE IT LIVES: Deployment Status in Setup, and the DeployRequest object in the Tooling API.

HOW LONG IT'S KEPT: Deployment Status shows deployments in progress or completed within the last 30 days.

LICENSE: None.

HOW TO GET IT: From Setup, enter Deployment Status in Quick Find. Or query the Tooling API:

```
SELECT Id, Status, CheckOnly, StartDate, CompletedDate, CreatedBy.Name
FROM DeployRequest
ORDER BY CompletedDate DESC
```

WATCH OUT: Deployment records include attempts and validations. Check Status and CheckOnly before concluding that a change reached the org. Direct changes in Setup require separate review in Setup Audit Trail. Business intent and prior component versions belong in release notes, tickets, or source control. [M12]

What's different between production and the full sandbox?

WHERE IT LIVES: Nowhere in one place. You compare metadata retrieved from each org, and check the sandbox's last refresh in production's sandbox list or the SandboxProcess object.

HOW LONG IT'S KEPT: Each comparison reflects the two orgs at the moment you retrieve them.

LICENSE: None.

HOW TO GET IT: From a Salesforce DX project, retrieve the same manifest from both orgs and compare. These shell examples use prod and fullsb as existing org aliases:

```
sf project retrieve start --manifest package.xml \
  --target-org prod --output-dir compare/prod
sf project retrieve start --manifest package.xml \
  --target-org fullsb --output-dir compare/fullsb
diff -r compare/prod compare/fullsb
```

To inspect sandbox copy or refresh processes, query the Tooling API in production. Check the status as well as the completion time:

```
SELECT SandboxName, Status, EndTime
FROM SandboxProcess
ORDER BY EndTime DESC
```

WATCH OUT: A recent refresh is not proof of alignment. Changes made in either org since then create drift, and data, users, and some settings aren't captured by a metadata comparison. [M13]

INTEGRATIONS AND API

Start with the error response the integration retained, if it was captured. Salesforce does not keep a complete journal of every request and response.

Why is this integration failing?

WHERE IT LIVES: Integration logs may contain the error Salesforce returned, such as FIELD_CUSTOM_VALIDATION_EXCEPTION. On the Salesforce side, check Bulk Data Load Jobs, captured debug logs for the execution user, and relevant Event Monitoring API event types.

HOW LONG IT'S KEPT: Bulk ingest results are available for seven days after completion unless deleted earlier. System debug logs last 24 hours; monitoring logs last seven days. Paid EventLogFile retention depends on configuration and can extend to one year.

LICENSE: None for bulk jobs and debug logs. Detailed API and Bulk API event types generally require Shield or the Event Monitoring add-on in production editions.

HOW TO GET IT: From Setup, enter Bulk Data Load Jobs in Quick Find. To capture the next run in detail, set a trace flag on the integration user under Debug Logs before it runs. With Event Monitoring, find the relevant log files:

```
SELECT EventType, LogDate, LogFileLength
FROM EventLogFile
WHERE EventType IN ('API', 'RestApi', 'BulkApi')
  AND LogDate = LAST_N_DAYS:7
```

WATCH OUT: Save result files promptly; job cleanup depends on API and job or batch state. Debug logs help only if logging captured the transaction. Daily event files are normally generated the following day and can arrive later; hourly files are also delayed. [M3, M8, M9]

Which integration is using up our daily API limit?

WHERE IT LIVES: System Overview in Setup for the org total, the API Usage Last 7 Days administrative report for a per-user breakdown, and the API Total Usage event type for detail by connected app.

HOW LONG IT'S KEPT: System Overview shows the last 24 hours; the report covers seven days. API Total Usage files have one-day retention in Enterprise, Unlimited, and Performance editions without an add-on. Paid retention depends on configuration.

LICENSE: None for System Overview and the report. API Total Usage is one of the few event types available free with 1-day retention.

HOW TO GET IT: From Setup, enter System Overview in Quick Find. For the per-user view, open Reports, then Administrative Reports, then API Usage Last 7 Days (a Salesforce Classic report). To get warned before you hit the limit, set up API Usage Notifications in Setup.

WATCH OUT: The daily allowance is shared across the org. The seven-day report groups calls by user, so shared integration users can obscure which application generated the calls. Use connected-app or client attribution where available. [M4, M20]

AUTOMATION AND ERRORS

Detailed execution evidence can expire quickly or depend on logging being enabled. Decide what you need to capture before the next failure.

Why did this flow fail?

WHERE IT LIVES: Flow error emails, the Paused and Failed Flow Interviews list in Setup, and debug logs.

HOW LONG IT'S KEPT: Error emails last as long as your mailbox keeps them. Treat the failed interview list as short-term working data, not a record.

LICENSE: None.

HOW TO GET IT: Choose who receives error emails in Setup under Process Automation Settings, using Send Process or Flow Error Email To. To review failures, enter Paused and Failed Flow Interviews in Quick Find and switch the list view to all failed interviews.

WATCH OUT: Not every failure appears as a failed interview; it depends on the flow type and how faults are handled. A fault path that catches an error quietly sends no email. Error emails can include record data and values users entered, so send them to a controlled mailbox. [M18]

What happened inside this transaction?

WHERE IT LIVES: Debug logs, which can show Apex, Flow, validation, and callout activity when logging captures the execution.

HOW LONG IT'S KEPT: System debug logs, 24 hours. Monitoring debug logs, 7 days. Each log is capped at 20 MB, with older lines trimmed beyond that.

LICENSE: None.

HOW TO GET IT: In Setup, open Debug Logs and add a user-based trace flag for the execution identity before reproducing the problem. Class or trigger flags adjust logging levels; they do not start logging on their own.

WATCH OUT: A missing log cannot reconstruct an earlier transaction. Tests and API debugging options can also generate logs. More than 1,000 MB of logs in 15 minutes disables trace flags; tracing busy execution can affect requests. Keep the capture period focused. [M8]

Did this scheduled or batch job run, and did it succeed?

WHERE IT LIVES: Apex Jobs and Scheduled Jobs in Setup, and the AsyncApexJob object.

HOW LONG IT'S KEPT: Short-term. Job records aren't designed as a long-term history, so export anything you need to keep.

LICENSE: None.

HOW TO GET IT:

```
SELECT ApexClass.Name, JobType, Status, NumberOfErrors,
       ExtendedStatus, CompletedDate
FROM AsyncApexJob
WHERE CreatedDate = LAST_N_DAYS:7
ORDER BY CreatedDate DESC
```

WATCH OUT: A batch job can finish with a Completed status and still have failed batches, so check NumberOfErrors. A completed job also doesn't prove the business outcome happened, such as records actually being updated as intended. [M21]

USAGE AND ADOPTION

A latest-use value can help identify what deserves review. Understanding who used a report and how often may require detailed activity evidence such as Event Monitoring.

Is anyone using this report?

WHERE IT LIVES: The LastRunDate field on the Report object. With Event Monitoring, the Report event type shows who ran each report.

HOW LONG IT'S KEPT: LastRunDate holds only the most recent run. Report event log retention depends on your subscription and configured settings.

LICENSE: None for LastRunDate. Detailed Report events generally require Event Monitoring in production editions.

HOW TO GET IT:

```
SELECT Name, FolderName, LastRunDate
FROM Report
WHERE LastRunDate < LAST_N_DAYS:180 OR LastRunDate = null
ORDER BY LastRunDate
```

WATCH OUT: A report run once a quarter or once a year can look abandoned. Confirm the purpose with its owner or audience before retiring it. [M17]

Which Lightning pages and apps are people using?

WHERE IT LIVES: The Lightning Usage app in the App Launcher, backed by objects such as LightningUsageByFlexiPageMetrics and LightningUsageByAppTypeMetrics.

HOW LONG IT'S KEPT: Daily aggregates. Confirm the available history in your org and export trends you need to retain.

LICENSE: None.

HOW TO GET IT: Open the App Launcher and search for Lightning Usage. To find the most viewed custom pages over the past week:

```
SELECT FlexiPageNameOrId, SUM(TotalCount) Total
FROM LightningUsageByFlexiPageMetrics
WHERE MetricsDate = LAST_N_DAYS:7
GROUP BY FlexiPageNameOrId
ORDER BY SUM(TotalCount) DESC
LIMIT 10
```

WATCH OUT: These are daily rollups, and Salesforce notes they may not match raw Event Monitoring logs. Page views show attention, not whether people completed their work. [M19]

Are we paying for licenses nobody uses?

WHERE IT LIVES: Company Information in Setup for licenses purchased and assigned, and the User object for last login by license.

HOW LONG IT'S KEPT: Current state only.

LICENSE: None.

HOW TO GET IT:

```
SELECT Name, Profile.UserLicense.Name, LastLoginDate
FROM User
WHERE IsActive = true
  AND (LastLoginDate < LAST_N_DAYS:90 OR LastLoginDate = null)
```

WATCH OUT: Permission set licenses and feature licenses are counted separately from user licenses, so check those lists in Company Information too. A recent login shows access, not productive use. [M22]

RECORDS AND DATA

Deletion, export, and record-access questions require different sources. Confirm that the relevant activity was recorded before treating an empty result as an answer.

Who deleted this record?

WHERE IT LIVES: The org Recycle Bin, including its Deleted By column. Admins can inspect deleted records across users. Setup Audit Trail can also record mass-delete operations.

HOW LONG IT'S KEPT: Records normally remain in the Recycle Bin for 15 days unless removed earlier. API visibility and the timing of final purge can differ.

LICENSE: None.

HOW TO GET IT: Open the Recycle Bin from the App Launcher and switch to the org view. For API retrieval, use queryAll, or Export All in Data Loader:

```
SELECT Id, Name, LastModifiedById,
       LastModifiedBy.Name, LastModifiedDate
FROM Account
WHERE IsDeleted = true
```

WATCH OUT: Hard deletes bypass the Recycle Bin. Records absent from the bin may remain queryable briefly before final purge, but that is not a recovery plan. Keep backups for records the business needs to recover. [M10]

Who exported this report or data?

WHERE IT LIVES: Event Monitoring's Report Export and Report event types. Data Export service requests are recorded in Setup Audit Trail. Exports through Data Loader appear in the API and Bulk API event types.

HOW LONG IT'S KEPT: Paid event log retention depends on configuration and can extend to one year. Setup Audit Trail retains supported changes for 180 days.

LICENSE: Detailed report-export event types generally require Shield or the Event Monitoring add-on in production editions. Setup Audit Trail alone is not a record of every export.

HOW TO GET IT:

```
SELECT EventType, LogDate, LogFileLength
FROM EventLogFile
WHERE EventType IN ('ReportExport', 'Report')
  AND LogDate = LAST_N_DAYS:30
```

WATCH OUT: Check ReportExport and Report events. Formatted Report and XLSX Detail exports appear under Report rather than ReportExport. An API activity record also does not establish every field or record returned. [M5]

Who viewed this sensitive record?

WHERE IT LIVES: Real-Time Event Monitoring's LightningUriEvent stores activity for records created, accessed, updated, or deleted in Lightning Experience, including record IDs. Other access channels require other event types.

HOW LONG IT'S KEPT: LightningUriEvent storage supports up to six months when enabled; Developer Edition retains one day. Stored-event retention is separate from streaming replay.

LICENSE: Shield or the Event Monitoring add-on, plus the View Real-Time Event Monitoring Data permission.

HOW TO GET IT: Enable storage for the event type before it is needed. Query a bounded EventDate window, then filter the returned RecordId values in your analysis tool. LightningUriEvent does not support a RecordId filter in SOQL. For streaming, use LightningUriEventStream.

WATCH OUT: Browser upload timing and missed events can leave gaps. LightningUriEvent covers Lightning Experience, so review other sources for reports, list views, and APIs. Absence of an event does not prove that a record was never accessed. [M6]

Questions that need evidence captured in advance

Some questions cannot be answered reliably from the evidence still available when an incident occurs. Retained snapshots, suitable logs, and documented intent help close different parts of that gap.

QUESTION	WHAT MAY BE MISSING	WHAT TO CAPTURE GOING FORWARD
Who had this permission on a past date?	Current assignments are not a complete historical snapshot; available change records have limits.	Dated snapshots of assignments and permission definitions
What did this setting look like before it changed?	Setup Audit Trail records a short description of most changes, not the full prior configuration.	Scheduled metadata snapshots, or source control for every change
What depended on this component before we changed it?	Dependency data reflects only the current org.	A dependency check saved with each change record
Why was this changed?	A system change log does not explain the business decision.	A linked ticket or release note for every production change
What did the integration send, and what came back?	No complete retained request and response journal exists by default.	Integration logs and request IDs with agreed retention
Who viewed or exported this data?	Required activity may not have been captured or may already have expired.	Appropriate event collection, enabled storage, and retained copies
What happened in last night's failed transaction?	Detailed execution evidence may not have been captured or may have expired.	Monitored error notifications and suitable application or integration error logs

Keeping evidence longer

Scheduled exports preserve evidence that would otherwise expire. They cannot recover activity that was never captured or supply business intent that was never documented. Set the collection schedule shorter than the retention window and agree where copies will live.

EVIDENCE	NATIVE WINDOW	HOW TO KEEP IT LONGER
Debug logs	System logs 24 hours; monitoring logs 7 days	Don't rely on them for history; log errors from your own code and integrations instead
Bulk ingest results	7 days after completion unless deleted earlier	Have the integration store job IDs and result files
Recycle Bin	Normally 15 days; can be removed earlier	Regular backups of the objects that matter
Event Monitoring log files	Varies by subscription and settings	Confirm paid retention up to one year; collect copies on a schedule shorter than the active window
Setup Audit Trail	180 days	Query SetupAuditTrail or download the CSV monthly
Login History	6 months	Export LoginHistory monthly
Field history	Up to 18 months in UI; 24 via supported API	Field Audit Trail archives until deleted, or periodic exports
Permissions, dependencies, configuration	Primarily current state	Dated snapshots plus relevant change records

BEFORE THE NEXT INCIDENT

Decide which fields to track and which logs to keep before an incident. Longer retention preserves collected evidence; it does not recreate missing history.

SOURCES

References and source notes

Salesforce documentation checked October 23, 2026. References 1 to 8 support the main guide. References M1 to M22 support the evidence map. Availability and retention can vary by edition, license, settings, and API version.

[1] MetadataComponentDependency. Metadata relationships and the limits of dependency inspection.

[2] Monitor Setup Changes with Setup Audit Trail. Recorded setup changes, including who made a change and when.

[3] Revoke Permissions and Access. Contributing grants and the effects of changing shared permissions.

[4] Using Event Monitoring. API access to event logs, delivery timing, and potential gaps or duplicates.

[5] FlowExecutionErrorEvent. Error evidence for screen flow executions. It is not a record of every Flow failure.

[6] Apex Unexpected Exception Event Type. Evidence about unhandled exceptions during Apex execution.

[7] Measure Performance for Your Salesforce Org. Page-performance measures and relevant usage and Event Monitoring signals.

[8] Salesforce Event Monitoring Frequently Asked Questions. Event availability, collection, retention, and limits on the detail recorded.

Evidence map sources

[M1] Setup Audit Trail; SetupAuditTrail object.

[M2] LoginHistory object.

[M3] Using Event Monitoring; Event Monitoring FAQ.

[M4] Get Started with Event Monitoring.

[M5] Report Export Event Type; Report Event Type.

[M6] LightningUriEvent; Real-Time Event Monitoring Data Storage.

[M7] Field Audit Trail; Export Field History Data up to 24 Months.

[M8] Debug Logs; Debug Log Order of Precedence.

[M9] Bulk API and Bulk API 2.0 Limits and Allocations.

[M10] Find Out Who Deleted a Record; Deleting Data.

[M11] MetadataComponentDependency; Dependency Queries with Bulk API 2.0.

Evidence map sources, continued

[M12] DeployRequest; Metadata API deploy().

[M13] SandboxProcess.

[M14] Audit Privileged Users with SOQL; PermissionSet; User Access Summary.

[M15] UserRecordAccess; Restriction Rules Considerations.

[M16] OauthToken.

[M17] Report object.

[M18] FlowInterview.

[M19] Lightning Usage by FlexiPage; Lightning Usage by Page.

[M20] API Request Limits and Allocations; Monitor API Usage.

[M21] AsyncApexJob.

[M22] User object.

11

About Coennect

Coennect helps enterprise Salesforce teams connect configuration, dependencies, permissions, change history, activity, and usage into a shared operational view. Its focus is helping teams understand how their org works, investigate issues, and identify where attention is needed. This guide reflects a practical view of future operations: maintain useful context, assign clear work, and judge results by their effect on the team and the business. The approach can be applied with the tools and processes a team already uses.

COENNECT



Managing Salesforce in the Agentic Era

A practical guide to reducing repetitive work, finding blind spots, and making better decisions

coennect.com