HOUSEBLEND

HOUSEBLEND / RESEARCH & PRACTICAL GUIDANCE

NetSuite Deleted Records Incremental Sync: Controls

> Make NetSuite work better for your finance and operations teams with Houseblend. We help design, implement, integrate and improve ERP systems, with practical support for the people who use them every day.

Published by Houseblend · 2026-09-28 · netsuite deleted records incremental sync · netsuite deleted records · netsuite tombstones · suiteql · system notes v2 · getdeleted · change data capture · data reconciliation · suiteanalytics connect

Original article: https://www.houseblend.io/articles/netsuite-deleted-record-sync-reconciliation

In this article

1. Executive Summary
2. Introduction and Background
3. Key Changes
4. Implementation Considerations and Process Changes
5. Tombstone Application and Reconciliation
6. Recovery, Replay, and Audit Evidence
7. Data Analysis and Evidence
8. Implications and Future Directions
9. Frequently Asked Questions (FAQs)
10. Conclusion

Executive Summary

An incremental export that selects records by last-modified time can reproduce inserts and updates while leaving a deleted source row active downstream. The remedy is a separate **delete-propagation control**: establish which NetSuite record types expose a deletion signal through the chosen channel, land that signal as a durable tombstone, apply it in key order, and reconcile the resulting active set against a current source snapshot. Oracle documents a **Deleted Record search** for supported old-data-source records and a **SuiteQL path** for supported new-analytics-source records; the latter requires the NetSuite Analytics Warehouse feature. (docs.oracle.com) A record's absence from a modified-row extract is therefore insufficient evidence of deletion. (learn.microsoft.com)

As of September 2026, the channel choice has a release dimension. Oracle says the old **NetSuite.com Connect** data source is unavailable after the **2026.1** account upgrade, leaving NetSuite2.com for Connect. (docs.oracle.com) SOAP **getDeleted** can return deletion references for supported types, but Oracle calls **2025.2** its last planned SOAP endpoint and directs new integrations from 2026.1 to REST with OAuth 2.0. (docs.oracle.com) Neither statement implies that a generic REST record query supplies a complete delete stream. System Notes v2 can retain deleted-record detail, but its published support list is narrow and its events can appear after a short background delay. (docs.oracle.com)

Connector behavior must be checked separately from NetSuite capability. Fivetran documents soft deletes for single-key NetSuite2.com tables, hard deletes for composite-key tables, and a standard-table detection method that does not rely on DeletedRecords. (fivetran.com) Stitch exposes a Deleted table populated through getDeleted, while Celigo added a Deleted Records export option in a 2026 release. (www.stitchdata.com) (docs.celigo.com) These are different downstream contracts. The integration owner should test a real delete under the production role and record type, inspect the actual destination row, and preserve the evidence. (www.stitchdata.com)

The core control is measurable. Let **active orphan count** be downstream active IDs absent from both a complete current source set and the eligible tombstone set. Divide by downstream active IDs to obtain **orphan rate**, with an explicit zero-denominator rule. (Hypothetical Example) Five orphans among 10,000 active IDs yield 0.05%; that is an illustration, not a benchmark. Keep separate measures for extraction lag, unapplied

tombstones, unresolved keys, and snapshot completeness. Reconcile only after the source snapshot and delete feed have reached the same cutoff. Warehouse systems document why this matters: at-least-once delivery requires idempotent application, and change feeds may have finite retention. (docs.cloud.google.com) (docs.snowflake.com) This report supplies a coverage matrix, tombstone contract, replay procedure, and anti-join check for that control. (docs.aws.amazon.com)

Introduction and Background

(Hypothetical Example) Consider a warehouse customer dimension loaded nightly from NetSuite. Monday's run copies customer 4812. On Tuesday, the source record is deleted. A Wednesday job asks for rows modified since its last watermark; the deleted row cannot appear in the returned active set. The warehouse still presents customer 4812 as active. Any dashboard, operational cache, or customer relationship management (CRM) integration using that dimension now has an **orphan**. The example is hypothetical, but the failure mode follows from the semantics of a watermark-only copy: Microsoft describes watermark copying as comparison against an incremental column, while its change data capture (CDC) mode explicitly captures deletes. (learn.microsoft.com) Stitch likewise notes that a vanished source row can remain in a key-based incremental destination. (www.stitchdata.com) (www.stitchdata.com)

Deletion is also distinct from **inactivation**. Oracle says an inactive record remains in NetSuite for future reference and can appear in searches when the Inactive filter includes it. (docs.oracle.com) A downstream row can therefore be active in storage but inactive for business use, or physically deleted in NetSuite and still active in the warehouse. The data owner must specify separate states for those cases. A single `is_deleted` flag does not explain whether the source record was deleted, inactivated, excluded by permissions, or temporarily absent because an extraction failed.

This report addresses analytics engineers, integration developers, controllers, and audit teams. It assumes an existing source-to-target pipeline and concentrates on **delete propagation**, not general SuiteQL or connector setup. Houseblend identifies integration design and validation as part of its NetSuite integration service; that service perspective is relevant because the control spans source permissions, transport, destination application, and finance review. (www.houseblend.io) The design recommendations below are analytical guidance, not a claim that a particular connector covers every record type. (docs.aws.amazon.com)

The first decision is the population: record type, account, feature set, role, channel, and destination. The second is the signal: deleted-record search, System Notes v2, SOAP getDeleted, SuiteQL through NetSuite2.com, connector-specific handling, or an intentional source-side inactive state. The third is proof: a reproducible test delete plus a full-set comparison. Oracle warns that available query types can depend on account, role, and enabled features. Consequently, a capability table is a test plan, not a universal support warranty.

Key Changes

A delete is an event, while inactivity is a source state

Hard deletion removes the active source row. Its event may survive as limited metadata in NetSuite's deleted-record facilities, but Oracle frames retention as applying to **some records**. (docs.oracle.com) **Inactivation** keeps a source row and should normally move the destination into an inactive business state through the ordinary incremental update path. That makes inactivation a practical source-side choice when the business needs continued reference to the same record. That does not make inactive and deleted interchangeable: a restored inactive record and a recreated record with a new internal ID have different histories. (debezium.io)

The policy should name the target action for each state. A warehouse may retain a deleted dimension row with `is_deleted=true` for historical joins, while an operational cache may remove it from active reads. A CRM may deactivate its corresponding object rather than erase it. Those are destination policies, not NetSuite facts. Store the raw tombstone even if the serving table physically removes the row. A durable event record makes replay, evidence review, and corrections possible after destination code changes. Debezium's documented tombstone pattern illustrates the minimal event idea: a deleted key paired with a null value. (debezium.io) (docs.databricks.com)

Different NetSuite surfaces expose different evidence

Oracle separates the **old data source**, used for saved searches and reports, from the **new analytics data source**, used for SuiteAnalytics Workbook. Its deleted-record page routes the first to Deleted Record search and the second to SuiteQL through SuiteScript or SuiteAnalytics Connect, subject to the NetSuite Analytics Warehouse prerequisite. (docs.oracle.com) An old-source saved search can be rerun and can expose fields such as deletion date, name, type, user, and context, but only for supported record types and with the appropriate permission. (docs.oracle.com) SuiteQL examples for the new source use `deletedRecordInConnect`; Oracle cautions that record-type strings may have inconsistent casing.

System Notes v2 is a richer audit-history source for **supported types**, not a blanket CDC substitute. Oracle says v2 retains information for deleted records, unlike original System Notes, and its current published support page lists six record types or configuration pages. (docs.oracle.com) (docs.oracle.com) A team seeking transaction or customer details should check that precise list in its account instead of assuming the general v2 description applies. A role can also change what history is visible: administrators can search all v2 notes, whereas other users see only their own notes for records they may view. (docs.oracle.com)

SOAP getDeleted returns references for supported types, filtered by record type, script ID, and deletion date. It has paging through the `pageSize` preference and `pageIndex`, and Oracle says deletion data remains available indefinitely. (docs.oracle.com) Those properties make it useful for an existing SOAP integration's replay, but an implementation must still test type support, role permissions, and endpoint plans. Oracle identifies the DeletedRecordType enumeration as the support reference; it does not say every NetSuite record supports this operation. (help.boomi.com)

Table 1 summarizes the channel decision. Each row is a candidate that needs a record-level test in the production account configuration.

Signal or channel	What it can prove	Coverage and prerequisite	Target action
Deleted Record saved search	A supported old-source record was deleted, with basic metadata.	Confirm record support and Deleted Record Search permission.	Preserve the raw result; emit a keyed tombstone only after verifying a stable internal ID for that record type against the destination key. If no key is available, use a tested deletion channel that supplies one.
SuiteQL or NetSuite2.com Connect	A supported new-source record appears in `deletedRecordInConnect`.	Confirm NetSuite Analytics Warehouse feature, role, and type.	Normalize type and timestamp, then emit tombstone.
REST SuiteQL (`deletedrecord`)	A deleted record can be returned by a SuiteQL query through REST web services. (docs.oracle.com)	Use `POST /services/rest/query/v1/suiteql`; test record-type coverage and role access in the account.	Land the returned deletion data and verify its destination key before applying a tombstone.
System Notes v2	Detailed delete action and retained history for supported types.	Check the published support list and role visibility.	Enrich evidence; account for background delay.
SOAP getDeleted	Deleted-record reference with type, key, and date.	Check DeletedRecordType, SOAP permissions, and endpoint lifecycle.	Page results into a durable event log.
Source-side inactive state	Record remains available with an inactive state.	The type must expose an inactive field or equivalent business status.	Apply an update, preserving a distinct state.
Connector-managed delete	Vendor-specific destination behavior, not a universal NetSuite event. (fivetran.com) (www.stitchdata.com)	Inspect the exact connector, source, table key, and version.	Translate vendor flag or hard delete into local contract.
Houseblend integration service	Implements an agreed source channel and destination control; the service is not itself a deletion signal. (www.houseblend.io)	Define account, role, record scope, and test evidence in the engagement.	Build and validate the chosen feed and reconciliation.

The table includes the implementation-service route because Houseblend provides NetSuite integration work, while its actual delete signal must still be selected from the documented channels. The table's most important boundary is **coverage by record type and channel**. A connector may synchronize a deleted row correctly without providing a reusable source tombstone, and a system note may preserve audit detail without offering an efficient complete feed. The old NetSuite.com Connect path should not be selected for a current design after Oracle's 2026.1 transition. (docs.oracle.com)

Connector semantics change the destination contract

Fivetran's current NetSuite SuiteAnalytics documentation describes chunk comparison for detecting standard-table deletes. It says single-primary-key rows are soft deleted in the destination and composite-key rows are hard deleted; its documentation also says it does not rely on DeletedRecords for standard tables. (fivetran.com) A Fivetran soft delete is represented by `_fivetran_deleted=true`, according to its general sync-mode documentation. (fivetran.com) Therefore an analyst must exclude that flag from active views, while a composite-key table requires a separate retained history if deletion evidence is needed. Fivetran also says a replacement connection does not inherit deleted rows from the prior connection, so migration of connector configuration is a reconciliation event. (fivetran.com) (fivetran.com)

Stitch documents a **Deleted** table sourced through getDeleted and warns that unsupported record types are absent from it. (www.stitchdata.com) Boomi documents a **Get Deleted** operation for a selected type and period. (help.boomi.com) Celigo's June 2026 release notes say Deleted Records can be selected as an export type using a NetSuite saved search. (docs.celigo.com) Airbyte's own NetSuite-to-Snowflake guidance says its cursor-based incremental path does not capture source deletions. (airbyte.com) These descriptions should be treated as product-specific documentation current at the cited date, then verified against the deployed connector version and output schema. (docs.celigo.com)

Implementation Considerations and Process Changes

Build a coverage register before writing extraction code

Create one register row per `(account, record type, source, role, channel, connector version)`. Include the business owner, downstream tables, delete policy, expected key, and a test-result field. Oracle's support lists and the SuiteScript Records Browser designation help select candidates, but the account-specific result remains decisive. (docs.oracle.com) (docs.oracle.com) A role that can read live records may still lack deleted-record access; Oracle requires Deleted Records plus Web Services permissions for getDeleted and describes separate Deleted Record Search access for saved searches. (docs.aws.amazon.com)

Use a controlled test to distinguish coverage from permission problems:

- **Seed:** create a record that is safe to remove in the test account and capture its internal ID, type, external ID if present, and source timestamp.
- **Update:** confirm the ordinary incremental path sees a change before testing deletion.
- **Delete:** remove the record or use an allowed test type, then record the action time and actor.
- **Observe:** query each proposed deletion surface with the intended integration role, not only an administrator role.
- **Inspect:** compare raw event fields and destination state, including connector flags and hard-removal behavior.
- **Repeat:** run the same query after a delay, because System Notes v2 updates run in the background. (docs.oracle.com)
- **Record:** preserve account type, features, role, channel, connector version, date, and the exact query or search definition. (learn.microsoft.com)

The test should distinguish **unsupported**, **not authorized**, **not yet visible**, and **not emitted by connector**. Those outcomes lead to different remedies. A complete source snapshot can be an independent safety net when incremental deletion coverage is uncertain; a snapshot mismatch should create a reviewable candidate, not immediately erase a destination row. Differences in role visibility, subsidiary scope, filters, and timing can also produce apparent absence.

The coverage register should also capture these decision fields:

- **Eligible source population:** record types, subsidiaries, and business filters included in the control.
- **Signal authority:** the documented source surface and the account test that confirms it.
- **Extraction identity:** the role, authentication method, and channel used in production.
- **Destination contract:** hard removal, soft-delete flag, inactive state, or retained history.
- **Fallback control:** full-ID snapshot cadence and owner when no reliable event feed exists.

Give tombstones a durable, idempotent contract

A **tombstone** is a record that says a source key became deleted at a specified event time. The feed needs a stable compound identity because internal IDs can be meaningful only with their account and record type. Keep both raw and normalized type values when SuiteQL casing can vary. Record source event time separately from ingestion time; Google's Datastream event model distinguishes source-change time from read time and uses Coordinated Universal Time (UTC) for both. (docs.cloud.google.com) Oracle's new-source deletion display uses Pacific Time for Date Deleted and offers a user-time-zone variant, while the old-source search displays the user's time zone. (docs.oracle.com) (docs.oracle.com) Normalize timestamps into UTC, preserve the original field and zone, and test daylight-saving boundaries.

Table 2 is a proposed durable tombstone schema. It is a design contract, not an Oracle table definition.

Field	Purpose	Validation rule
`account_id`, `record_type`, `internal_id`	Compound source identity.	Non-null; type normalized with raw type retained.
`source_deleted_at_utc`	Event time used for cutoff and ordering.	Keep original timestamp and zone alongside it.
`observed_at_utc`, `batch_id`	Arrival and replay lineage.	Never overwrite the first observation.
`channel`, `source_reference`	Search, SuiteQL, SOAP, or connector origin.	Link to raw payload or immutable extract.
`event_key`, `event_rank`	Deduplication and conflict ordering.	Stable across retries; deterministic within a key.
`apply_state`, `applied_at_utc`	Destination processing evidence.	Distinguish pending, applied, and exception.
`raw_payload_hash`, `run_id`	Audit comparison and run trace.	Preserve the raw payload separately.

The schema separates **what NetSuite reported** from **what the destination did**. That separation matters when a retry arrives twice or when a late update and delete race. Google's Datastream documentation describes at-least-once event delivery, so deduplication by a stable event key is a general CDC requirement,

even when the NetSuite transport differs. (docs.cloud.google.com) Snowflake's MERGE guidance likewise recommends reducing a source batch to one row per target key to prevent ambiguous matches. (docs.snowflake.com) (docs.cloud.google.com)

Watermark with overlap, then order by source event time

Persist the last **fully committed** extraction cutoff, not merely the latest timestamp seen. Query an overlap interval on every run, deduplicate by event identity, and advance the cutoff only after raw landing and destination application succeed. A fixed overlap is a local service decision. It should exceed measured source visibility delay, scheduler jitter, and clock conversion uncertainty, then be reviewed from observed lag. There is no published universal NetSuite overlap interval in the cited documentation. System Notes v2's background update warning reinforces the need to measure delay in the actual account.

For a page-based feed, walk every page before committing the window. SOAP getDeleted respects `pageSize` and uses `pageIndex`; total pages are part of its response. Bound the query by a closed interval, then persist the interval, page count, raw payload counts, and a checksum. Requery the boundary in the next run and deduplicate. Avoid moving the watermark on an empty result until the query, authorization, and completeness checks have succeeded. A permission change can otherwise look like a valid empty day.

Tombstone Application and Reconciliation

Apply delete state without losing history

The downstream application should be **idempotent**: replaying the same tombstone yields the same serving state. First stage raw events. Next choose the latest valid event per `(account, type, internal ID)` under an explicit ordering rule. Finally update the serving table in one transaction or a recoverable batch. Snowflake documents MERGE for applying a change log and warns that a target row should match one source row in a deterministic merge. (docs.snowflake.com) For a warehouse dimension, set `is_deleted=true` and `deleted_at_utc`; for an operational cache, remove the active key while retaining its event in the audit store. Microsoft Fabric distinguishes a Type 1 destination that removes source-deleted rows from a historical Type 2 pattern that marks rows inactive. (learn.microsoft.com) (docs.cloud.google.com)

Use **key order** rather than trusting arrival order. A delete for key K may be observed after a late update whose source event time is earlier. Do not let the older update resurrect K. Conversely, if a new record legitimately reuses a business identifier, its NetSuite internal ID and creation lineage should determine whether it is a new entity. Google BigQuery's CDC documentation notes that a custom ordering key may be necessary when ingestion order is insufficient for updates to one primary key. (docs.cloud.google.com) That is a downstream analogy, not a claim about NetSuite event ordering. (docs.cloud.google.com)

Parent-child relationships need an explicit policy. If a parent is deleted, child rows can remain valid historical facts while the parent is no longer an active dimension member. Apply a parent tombstone to the parent serving view, then check whether children should be deactivated, detached, or preserved for financial history. Do not cascade deletion merely because a relational foreign key exists. For an active-only application, process child removals before the parent when referential constraints require it; for a historical warehouse, keep the

key bridge and mark the parent state. PostgreSQL logical replication documentation illustrates the general need for a usable replica identity so a subscriber can identify the row affected by DELETE. (www.postgresql.org) (www.postgresql.org)

Prove there are no active orphans

An event feed proves that some deletes were observed. It does not prove that all active downstream IDs still exist in NetSuite. Periodically obtain a **complete current-source ID snapshot** for each covered population, using the same business filters and permissions as the feed. Compare it with the active downstream set and the tombstone set at a shared cutoff. AWS Database Migration Service (DMS) validation is an example of row-level source-to-target comparison and records validation exceptions; the principle applies even if NetSuite extraction uses different tooling. (docs.aws.amazon.com) (docs.getdbt.com)

The following SQL is **pseudocode**. Names and syntax need adaptation to the warehouse. The source snapshot must be complete and the tombstone feed caught up to `:cutoff_utc` before the result is interpreted.

```
WITH downstream_active AS (
  SELECT account_id, record_type, internal_id
  FROM target_entity
  WHERE is_deleted = FALSE AND is_inactive = FALSE
), source_current AS (
  SELECT account_id, record_type, internal_id
  FROM source_id_snapshot
  WHERE snapshot_cutoff_utc = :cutoff_utc
), known_deletes AS (
  SELECT account_id, record_type, internal_id
  FROM raw_tombstone
  WHERE source_deleted_at_utc <= :cutoff_utc
)
SELECT d.account_id, d.record_type, d.internal_id
FROM downstream_active d
LEFT JOIN source_current s
  ON d.account_id = s.account_id
 AND d.record_type = s.record_type
 AND d.internal_id = s.internal_id
LEFT JOIN known_deletes t
  ON d.account_id = t.account_id
 AND d.record_type = t.record_type
 AND d.internal_id = t.internal_id
WHERE s.internal_id IS NULL AND t.internal_id IS NULL;
```

This query returns **unexplained active absences**. A separate query should return active rows that *do* have tombstones, because those are unapplied known deletes. Another should return target rows absent from the source but marked inactive. Review false positives from scope changes, permissions, late ingestion, and snapshot defects before changing production state. dbt supports tests that pass when a query returns zero failing rows, making the anti-join a repeatable assertion. (docs.getdbt.com) Freshness thresholds should be monitored separately so an old snapshot cannot make an orphan metric appear clean. (docs.getdbt.com) (docs.aws.amazon.com)

Classify each candidate before applying an automated correction:

- **Known delete:** a qualifying tombstone exists but the destination still treats the key as active.
- **Unexplained absence:** no source row and no tombstone are visible at the shared cutoff.
- **Scope mismatch:** the source snapshot and destination view use different filters or entities.
- **Incomplete run:** a page, partition, or connector batch is missing from the source snapshot.
- **Timing gap:** the event or snapshot has not passed the agreed completeness watermark.

Recovery, Replay, and Audit Evidence

Treat the raw tombstone log as the **system of evidence** for delete propagation. Keep extracted payloads, source query parameters, page boundaries, the role used, batch identifiers, and the destination application result. Store run-level counts for observed, deduplicated, applied, ignored-as-older, and failed events. AWS DMS records pending, suspended, and failed validation counts and can keep primary-key and failure details in a target control table; a NetSuite pipeline can implement analogous evidence without claiming to use DMS. (docs.aws.amazon.com) Azure Data Factory's CDC monitoring also separates changes read from changes written. (learn.microsoft.com) (learn.microsoft.com)

Replay should be deterministic:

- **Freeze the cutoff:** identify the last complete source window and target application checkpoint.
- **Preserve raw input:** copy or version the original payload and query metadata before any correction.
- **Classify exceptions:** separate unsupported type, permission failure, transport gap, duplicate, and target error.
- **Re-extract:** rerun an overlapped deletion interval or a full source-ID snapshot under the recorded role.
- **Deduplicate:** keep one canonical event per stable event key and resolve same-key ordering explicitly.
- **Apply:** rerun the idempotent target operation and log previous and resulting state.
- **Reconcile:** repeat active-set anti-join and known-delete application checks at a common cutoff.
- **Sign off:** attach counts, exception IDs, timestamps, and owner approval to the run record.

Retention is a cross-system constraint. Oracle says getDeleted deletion data remains available indefinitely, but that statement is about the NetSuite operation, not a warehouse stream or connector cache. (docs.oracle.com) Snowflake warns that a stream becomes stale when its offset falls outside source-table retention; Databricks says its change data feed records are transient and should be copied for permanent history. (docs.snowflake.com) (docs.databricks.com) Databricks Auto CDC temporarily retains tombstones and documents a default two-day interval, with guidance to exceed expected event delay. (docs.databricks.com) BigQuery documents a two-day window for out-of-order CDC delete operations. (docs.cloud.google.com) These are **destination product limits**, not NetSuite deletion-retention limits. The control owner should calculate the shortest usable replay horizon across source access, landing storage, stream retention, and operational response time. (docs.databricks.com)

Audit evidence should answer five questions without reconstructing the pipeline from logs: Which source key was deleted? When did the source report it? Which channel and role produced the evidence? When did the target cease to present it as active? Was the active-set comparison complete at the same cutoff? Houseblend

describes ongoing NetSuite administration and reporting support, but responsibility for these answers must be assigned explicitly in the operating model. (www.houseblend.io) A test record, saved query, role grant, and signed reconciliation report are more reproducible than an undocumented dashboard total. (docs.aws.amazon.com)

Data Analysis and Evidence

Public documentation gives **capability and product-limit evidence**, not a universal NetSuite deletion success rate. Oracle's current System Notes v2 support page names **six** record types or configuration pages, while its deleted-record documentation publishes separate supported-type lists for old and new data sources. (docs.oracle.com) Channel lifecycle is part of the measurement context because source availability and endpoint support change by release. These numbers do not predict a particular account's coverage; a coverage register and test run must supply that denominator. (docs.snowflake.com)

For each population and cutoff, compute the following measures from raw events and a complete source snapshot. *Table 3* gives formulas and interpretation; all sample values below are hypothetical.

Measure	Formula or evidence	Interpretation
Coverage rate	Tested supported record-type/channel pairs divided by pairs in scope.	Report numerator, denominator, role, and test date.
Tombstone application rate	Applied unique tombstones divided by eligible unique tombstones.	Exclude duplicate deliveries but disclose them.
Active orphan rate	Active downstream IDs absent from current source and tombstones divided by active downstream IDs.	Zero denominator means not applicable; scope and cutoff must match.
Known-delete backlog	Active downstream IDs with a qualifying tombstone.	Direct evidence of unapplied deletion state.
Propagation lag	Target applied time minus source deletion time, summarized by percentile.	Keep extraction lag and target lag separate.
Snapshot completeness	Expected partitions and pages received divided by those planned.	Withhold orphan conclusions until complete.

The measures separate **coverage**, **delivery**, and **correctness**. (docs.getdbt.com) (Hypothetical Example) In a run with 10,000 active downstream IDs and 5 unexplained active absences, the active orphan rate is `5 / 10,000 = 0.05%`. If 3 additional active IDs have valid tombstones, they belong in the known-delete backlog rather than the unexplained-orphan numerator. No public benchmark in the researched primary documentation establishes an acceptable NetSuite orphan rate or deletion lag. The data owner should set a service objective from business use, measured baseline, and exception tolerance, then publish both numerator and denominator.

Lag has more than one clock. AWS DMS defines source and target CDC latency around source commit and target application; Google's Datastream metadata distinguishes source-change time from read time. (docs.aws.amazon.com) (docs.cloud.google.com) A NetSuite pipeline should record source deletion time, first

observation, raw landing, and target application separately. Comparing a Pacific Time display field with a UTC warehouse timestamp without the original zone can create an artificial lag. Oracle explicitly documents the Pacific Time display behavior for its new-source deletion date. (docs.oracle.com)

Retention numbers from other platforms are useful as design warnings, not portable defaults. Snowflake may extend an unconsumed stream's retention to a default maximum of **14 days** in the documented circumstance, while BigQuery and Databricks each document **two-day** delete or tombstone windows for particular CDC features. (docs.snowflake.com) (docs.cloud.google.com) (docs.databricks.com) Confluent documents a **24-hour** default for compacted-topic delete markers. (docs.confluent.io) A multi-stage pipeline must identify the shortest relevant retention period in its own deployed configuration. None of those figures measures NetSuite's deleted-record coverage. (docs.databricks.com)

Implications and Future Directions

The practical architecture is a **dual control**. An incremental event path gives timely delete propagation, while a periodic complete-source snapshot checks whether any event was missed. The event path can be a native NetSuite deleted-record surface or a connector's own change contract. The snapshot path should use an independently verified role and a recorded cutoff. Microsoft's CDC versus watermark-copy distinction explains why a timestamp filter alone cannot stand in for a delete feed. (learn.microsoft.com) AWS DMS validation illustrates a comparison that records mismatches rather than assuming replication success. (docs.aws.amazon.com) (docs.getdbt.com)

Several implementation decisions deserve explicit ownership:

- **Finance owner:** decides whether deleted and inactive source records should remain visible in historical reports.
- **NetSuite administrator:** confirms source features, permissions, supported types, and release changes.
- **Integration owner:** owns raw landing, pagination, watermarks, retries, and connector-version checks.
- **Data platform owner:** applies tombstones and maintains active views, retention, and replay capacity.
- **Control owner:** reviews orphan counts, known-delete backlog, lag, and unresolved exceptions.

Release changes require a retest. A source path retirement or endpoint migration can change the available deletion signal. Connector releases can also change the available export shape, as Celigo's 2026 Deleted Records option shows. (docs.celigo.com) The correct response is to version the coverage register, retain a reproducible test, and compare the post-change destination contract with the previous one. Fivetran's documented single-key versus composite-key behavior illustrates why a row-level output check matters. (fivetran.com) (docs.celigo.com)

Where coverage is genuinely unavailable, a **complete ID snapshot** can detect disappearance, but it cannot by itself prove deletion cause. The system should quarantine an absent key for review until permissions, filters, source scope, and extraction completeness have been checked. Soft deletion or inactivation at source can be a deliberate alternative when the business process allows it, because the ordinary update feed can propagate an explicit state. Oracle confirms that inactive records remain available and searchable under suitable filters. This policy needs a business owner, since an inactive record and a hard-deleted record may have different reporting meanings.

Frequently Asked Questions (FAQs)

How can a team sync deleted records from NetSuite?

Choose a documented deletion surface for each supported record type, extract events under the intended role, land raw tombstones, apply them idempotently, and reconcile against a current source-ID snapshot. Oracle documents Deleted Record search for old-source types and SuiteQL for supported new-source types, while SOAP getDeleted is a legacy reference path with an endpoint lifecycle. Test the exact account and channel before treating any option as complete.

Can a Deleted Record saved search act as the incremental feed?

It can be saved and rerun for supported old-source records, with the Deleted Record Search permission. Before applying a saved-search result as a keyed tombstone, verify that a stable internal ID for each in-scope type maps to the destination key. Preserve any result without a verified key as raw evidence and use a tested deletion channel that supplies the key. The pipeline still needs a deletion-date window, overlap, deduplication, and periodic full-set reconciliation. A saved search result is a source signal, not proof that every downstream row was updated.

Can SuiteQL find all deleted NetSuite records?

Oracle documents a SuiteQL path for supported new-analytics-source records and a `deletedRecordInConnect` example, with the NetSuite Analytics Warehouse feature requirement. (docs.oracle.com) Oracle also documents querying `deletedrecord` through REST SuiteQL at `POST /services/rest/query/v1/suiteql`; confirm record-type coverage and permissions for the intended role. (docs.oracle.com) Neither route establishes universal all-type coverage. The old NetSuite.com Connect source is unavailable after the 2026.1 transition.

Does System Notes v2 solve delete CDC by itself?

System Notes v2 retains deleted-record detail for supported types, but its published support list is limited and history can arrive after a background delay. (docs.oracle.com) It is valuable audit evidence where supported; confirm that the role can view the needed events and that extraction is operationally suitable.

What proves a warehouse has no active orphans?

A complete current-source ID set and an active downstream ID set at the same cutoff support an anti-join. Known tombstones explain some absences but also identify unapplied deletes. Log the snapshot's scope, completeness, and role, then publish unexplained orphan count and rate. A zero count is meaningful only when the source snapshot and deletion feed are complete. (docs.aws.amazon.com) (docs.getdbt.com) (docs.getdbt.com)

Conclusion

NetSuite deleted-record sync is a **separate CDC control**. A last-modified extract can maintain active rows while silently retaining a row that no longer exists at source. The control begins by distinguishing hard deletion from inactivation, then selecting a deletion signal for each record type and channel. Oracle's Deleted Record search, supported new-source SuiteQL path, System Notes v2, and SOAP getDeleted expose different evidence and have different prerequisites. Connector outputs add another layer: a soft-delete flag, a hard-removed destination row, or a separate Deleted table are not the same contract. (fivetran.com) (www.stitchdata.com) (learn.microsoft.com)

The reliable downstream pattern is straightforward: preserve raw tombstones, normalize keys and time zones, deduplicate overlapping windows, apply events in deterministic order, and retain enough lineage to replay a run. A complete source-ID snapshot then tests the active downstream set. Report unexplained orphans separately from known deletes awaiting application, with matched cutoffs and explicit denominators. That combination can show both that events moved and that no active row was left unexplained.

The remaining uncertainty is account-specific coverage. Record type support, features, permissions, connector version, and release date can all change the result. A tested coverage register and repeatable reconciliation provide stronger evidence than a general claim that a platform or connector "supports deletes." The register should be retested when either the source configuration or destination contract changes.

Source links

Links cited in this article, in order of first appearance. Full addresses are provided for readers using extracted text.

1. https://docs.oracle.com/en/cloud/saas/netsuite/ns-online-help/section_4031815869.html
2. https://learn.microsoft.com/en-us/fabric/data-factory/cdc-copy-job
3. https://docs.oracle.com/en/cloud/saas/netsuite/ns-online-help/article_0128043134.html
4. https://docs.oracle.com/en/cloud/saas/netsuite/ns-online-help/article_2104046421.html
5. https://docs.oracle.com/en/cloud/saas/netsuite/ns-online-help/section_158108495822.html
6. https://fivetran.com/docs/connectors/applications/netsuite-suiteanalytics
7. https://www.stitchdata.com/docs/integrations/saas/netsuite-suitetalk
8. https://docs.celigo.com/hc/en-us/articles/51135588132891-Celigo-NetSuite-SuiteBundle-v1-38-9-0-SS-and-SuiteApp-v1-23-0-SDF-release-notes
9. https://docs.cloud.google.com/datastream/docs/events-and-streams?hl=en
10. https://docs.snowflake.com/en/user-guide/streams-intro
11. https://docs.aws.amazon.com/dms/latest/userguide/CHAP_Validating.html
12. https://www.stitchdata.com/docs/replication/deleted-record-handling
13. https://docs.oracle.com/en/cloud/saas/netsuite/ns-online-help/section_N490731.html
14. https://www.houseblend.io/services/netsuite-integrations
15. https://debezium.io/documentation/reference/3.1/transformations/event-flattening.html
16. https://docs.databricks.com/gcp/en/ldp/developer/ldp-python-ref-apply-changes

17. https://docs.oracle.com/en/cloud/saas/netsuite/ns-online-help/section_163584624721.html
18. https://docs.oracle.com/en/cloud/saas/netsuite/ns-online-help/section_163584617278.html
19. https://docs.oracle.com/en/cloud/saas/netsuite/ns-online-help/subsect_159499523675.html
20. https://docs.oracle.com/en/cloud/saas/netsuite/ns-online-help/subsect_158108595525.html
21. https://docs.oracle.com/en/cloud/saas/netsuite/ns-online-help/section_N3497592.html
22. https://help.boomi.com/docs/Atomsphere/Integration/Connectors/r-atm-NetSuite_operation_8c774e1e-5b05-4a8b-ab5a-2cfcb6ed918a
23. https://docs.oracle.com/en/cloud/saas/netsuite/ns-online-help/article_0701101315.html
24. https://houseblend.io/articles/netsuite-two-way-integration-guide
25. https://fivetran.com/docs/core-concepts/syncoverview/sync-modes/soft-delete
26. https://fivetran.com/docs/connectors/troubleshooting/preserve-deleted-rows-replacing-connection
27. https://airbyte.com/data-engineering-resources/netsuite-to-snowflake
28. https://houseblend.io/articles/netsuite-external-id-best-practices
29. https://learn.microsoft.com/en-us/azure/data-factory/how-to-change-data-capture-resource
30. https://docs.snowflake.com/en/sql-reference/sql/merge
31. https://docs.cloud.google.com/bigquery/docs/change-data-capture
32. https://www.postgresql.org/docs/18/logical-replication-publication.html
33. https://docs.getdbt.com/docs/build/data-tests?name=Fusion&version=2.0
34. https://docs.getdbt.com/reference/resource-configs/freshness
35. https://docs.databricks.com/gcp/en/tables/features/change-data-feed
36. https://www.houseblend.io/services/netsuite-managed-support
37. https://docs.aws.amazon.com/dms/latest/userguide/CHAP_Troubleshooting_Latency.html
38. https://docs.confluent.io/kafka/design/log_compaction.html

THE PUBLISHER

About Houseblend

Houseblend is a NetSuite consulting firm serving finance and operations teams. We help organizations implement ERP systems, connect business applications, improve existing configurations and maintain the systems that support everyday work. Our audience includes finance leaders, controllers, operations managers, NetSuite administrators and implementation teams.

Implementation and architecture

Houseblend provides NetSuite implementation, architecture and data migration services. We help teams evaluate how business processes, reporting requirements and existing data should fit together in an ERP environment. Training supports the people responsible for adopting and operating the resulting system.

Integrations, customization and AI

Our services include NetSuite integrations and customization, as well as AI integrations and AI transformation work. These engagements connect ERP data and workflows with the broader application landscape. The right design depends on the organization's systems, controls and operating needs.

Improve and support an existing system

Houseblend offers NetSuite health checks, optimization, managed support and project rescue services. We also provide expertise for analytics and specialist workflows, including NetSuite Analytics Warehouse, warehouse management and field service management. Published educational material helps teams investigate options and prepare informed questions for their implementation or support work.

Work with Houseblend

Explore NetSuite implementation, integrations, managed support and AI integrations. Contact Houseblend to discuss your current system and priorities.

Article examples explain concepts rather than promising a particular license, product capability, delivery schedule or outcome. Engagement scope is confirmed with the Houseblend team.

Website: https://www.houseblend.io

This article is educational. Verify consequential medical, legal, financial, regulatory and technical decisions with appropriate professionals and the cited primary sources. Company services are described separately from the article's research findings.