HOUSEBLEND

HOUSEBLEND / RESEARCH & PRACTICAL GUIDANCE

NetSuite Units of Measure: Case to Each Conversion Guide

> 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 · 2026-11-26 · netsuite units of measure · netsuite uom · case to each conversion · units type · unit of measure conversion rate · purchase unit vs stock unit · base unit · item fulfillment · celigo · inventory

Original article: https://www.houseblend.io/articles/netsuite-units-of-measure-case-to-each-conversion



In this article

1. Executive Summary
2. Introduction and Background
3. How NetSuite Models Units: Taxonomy and Defaults
4. Choosing the Base Unit and Defaults
5. The Conversion Contract and Proof Sheet
6. Transactions and Interface Boundaries
7. Data Analysis and Evidence
8. Testing and Approving a Catalog Change or Migration
9. Implications and Future Directions
10. Frequently Asked Questions (FAQs)
11. Conclusion

Executive Summary

NetSuite's **Multiple Units of Measure** feature lets one item be bought, stocked and sold in different units, but it does not decide which unit each external system sends. Oracle documents that purchase orders default to purchase units, while an item's units type cannot be changed once assigned (docs.oracle.com). The practical risk is therefore not a platform gap: it is an unstated agreement about which unit, quantity and price basis crosses each boundary between supplier, warehouse and storefront.

This article proposes a **conversion contract and proof sheet** for one physical item. Packaging standards show why the sheet is needed: GS1 assigns each packaging level of a product its own GTIN (ref.gs1.org), and a change in the number of items per case requires a new GTIN (www.gs1.org). A connector vendor's release note shows the same concern in practice: Celigo's Shopify integration app reads quantities that reflect what was actually transacted when NetSuite's UOM feature is enabled (docs.celigo.com). Every row of the sheet names the quantity unit and the price basis, because numbers without units are insufficient.

The worked examples are fictional. Three cases of 12 each equal 36 each, a $120 case price divided by 12 is a $10 each price, and the line value stays at $360 on either basis. A second example ships a 60-each order in two parts and compares the sides only after normalizing to one unit. A third reconciles a physical count of 7 cases and 5 each to 89 each.

Fixed packaging conversions suit the contract. Catch-weight or random-weight goods do not, because the measure varies by package (ref.gs1.org), and they need a separate fit-gap assessment. Rounding, replay and count tie-out finish the sheet: decimal arithmetic stores decimal fractions exactly while binary floating point does not (docs.python.org), retries need a caller-supplied request identifier (aws.amazon.com), and book quantities must reconcile to a physical count at reasonable intervals (www.irs.gov).

NetSuite platform statements are limited to Oracle's help pages; statements about connector behavior come from Celigo's release notes and documentation. Points Oracle does not document (stored rate precision, saved-search quantity fields, the effect of a default change on open transactions) are listed as tests to run in your own NetSuite account.

Introduction and Background

This explainer is for operations leaders, inventory controllers, warehouse leads and the NetSuite or Celigo integration owners who must decide, for one SKU, what each system means by "one." A supplier invoices by the case, a storefront sells singles, a third-party logistics provider (3PL) ships eaches and an electronic data interchange (EDI) partner sends pallets. NetSuite can represent all of these once an administrator enables the Multiple Units of Measure feature. The decision supported here is narrow: fix the base unit, the purchase, stock and sales defaults, the owner of each packaging conversion and the interface contract before quantities are imported or an established catalog is changed.

It is an item-unit and interface acceptance guide. It is not an item-type decision tree, an inventory reconciliation playbook, a costing guide or a general warehouse management system (WMS) setup article. Costing, EDI and 3PL topics are covered in separate guides and appear here only where they touch a unit boundary. This article is an independent explainer built from vendor and standards documentation.

Two limits on evidence apply. First, every statement about NetSuite platform behavior is limited to what Oracle's help pages say, and statements about connector behavior are limited to what Celigo's release notes and documentation say. Second, the numeric examples are illustrative calculations with a fictional item, not customer results or benchmarks. Where Oracle's pages appear to differ from each other, this article says so rather than choosing a reading: one page says units cannot be edited on inventory transactions (docs.oracle.com), while the assemblies page lists fulfillment as editable in units that match the sales order (docs.oracle.com).

A **quantity basis** is the unit in which a quantity is expressed, and a **price basis** is the unit to which a price applies. The proof sheet requires every row to name both.

How NetSuite Models Units: Taxonomy and Defaults

Units type, base unit and conversion rate

A units type is built by assigning a base unit and defining each other unit in terms of that base. Each type has exactly one base unit, and the conversion rate for a non-base unit is the quantity of base units that equal one unit of that line (docs.oracle.com). For case-to-each work, if the each is the base unit, a 12-pack case carries a conversion rate of 12, and the base unit itself is locked at one (docs.oracle.com).

Setup limits are documented: a units type cannot hold more than 650 units, and a transaction line cannot exceed 9,999,999,999 base units (docs.oracle.com). The second limit matters because a quantity entered in a large unit is checked after conversion. Oracle's help pages do not state how matrix parent and child records take the Units Type during item imports, so test it in a sandbox before a bulk catalog load.

Purchase unit, stock unit and sales unit

Choosing a units type on an item defaults the purchase, stock and sale units to the base unit, and the type limits which units can be selected (docs.oracle.com). Each default has a job: the stock unit is the default for the counts shown on the item, the purchase unit governs how Last Purchase Price is displayed, and the item-

record defaults show on purchase and sales transactions. Oracle's illustration of why they differ is a cable bought in pallets and stocked in spools.

Table 1 consolidates the defaults. The right-hand column states the question to answer in the contract; it is not an Oracle statement.

Unit role	Documented behavior	Decision to record in the contract
Base unit	One per units type, conversion rate locked at one; assembly builds, unbuilds and work orders are recorded in base units	The smallest unit you will ever count, pick or serialize
Purchase unit	Purchase orders default to it; it drives Last Purchase Price display	The unit the supplier prices and ships in, and who owns that factor
Stock unit	Inventory adjustments default to it	The unit the warehouse counts in
Sales unit	Invoices default to it	The unit the channel sells in, and its price basis
Pick decomposition units	Separate WMS setup with up to five units, including the base and at least one larger unit (docs.oracle.com)	Which larger units a picker may be asked to pull, kept apart from transaction conversion

Table 1 shows that "unit" covers five different decisions, each needing a named owner. Oracle also records that changing an item's purchase unit converts the standard cost to the new unit on save (docs.oracle.com), so a purchase-unit edit is a cost event as well as a display event.

What is locked and what is editable

Oracle states that after a units type is assigned to any item it can only be extended by adding units (docs.oracle.com). Another page says that after a base unit is saved, the selection can be changed (docs.oracle.com). Read together, these statements do not say what happens to historical or open transactions when a base unit or default is altered, so this article treats base-unit and units-type changes on in-use items as prohibited until a sandbox test proves a specific, reversible procedure. The effect of deactivating a unit on historical lines is not described in Oracle's documentation.

Choosing the Base Unit and Defaults

Enumerate every representation of one item

Start with a single SKU and list every way any system names it. For each representation, record the code, the quantity it contains in the smallest tracked unit, who prints it on a document and which system treats it as authoritative. Packaging standards are a useful cross-check. GS1 defines the base unit as the consumer trade item level or unit of use in a hierarchy of trade item groupings (ref.gs1.org), and links multiple GTINs to represent a product's packaging levels (ref.gs1.org). GS1 US lists giving the retail unit and the case the same GTIN as a mistake, since each requires a unique one (documents.gs1us.org).

Decide the base unit

Pick the smallest unit that must be counted, picked individually or serialized. For serialized items the base unit must be the smallest unit of measure (docs.oracle.com), and a transaction in a non-base unit needs one serial number for each base unit (docs.oracle.com). When stock, purchase and sale units are selected on transactions, serial numbered items cannot use fractional quantities, whereas lot numbered items can (docs.oracle.com).

Use these questions to settle the base unit:

- **Is the item serialized?** If yes, the base must be the smallest unit, and non-base transactions need one serial per base unit.
- **Can eaches be sold, returned or counted loose?** If yes, make the each the base so broken-case events stay whole numbers.
- **Is the item lot-tracked and sold by measure?** Fractional quantities are allowed for lot items, so state the permitted decimals.
- **Does the item vary by weight or count per package?** If yes, stop and open a fit-gap assessment instead of assigning a fixed factor.

These constraints argue for choosing the each as the base wherever eaches can be sold, returned or counted loose. A base of "case" forces fractional cases for every broken-case event, and the contract must then say how many decimals are legal.

Set defaults and assign conversion ownership

Make each default match the unit the originating party uses on its documents. The supplier's invoice unit is the purchase unit, the warehouse's count unit is the stock unit, and the channel's price-list unit is the sales unit. Quantity pricing schedules can also be keyed to a units type, and the reorder multiple works in base units, so a planner who thinks in cases must translate it.

Every conversion factor needs one owner and one effective date. The factor for a case is a fact about the supplier's packaging. GS1's example is a 12-case pallet becoming a 16-case pallet (www.gs1.org), and the contained item keeps the same GTIN whether it ships in a case of 12 or 24 (www.gs1.org). In NetSuite terms a new pack size should be a new, dated case definition that does not silently rewrite old documents. The proof sheet records who approves such a change.

The Conversion Contract and Proof Sheet

Normalize quantity and price at every boundary

The contract has one invariant: **quantity in transaction units multiplied by the conversion rate equals quantity in base units, and quantity multiplied by price equals line value in whichever basis it is expressed.** The check is meaningful only after both sides are normalized to one unit.

Worked example 1 (Hypothetical Example). A supplier sells a fictional item, tea-12, in cases of 12 each. A purchase order for 3 cases at $120 per case is expressed two ways:

- **Case basis:** 3 cases at $120 per case, line value $360.
- **Each basis:** 3 x 12 = 36 each at $120 / 12 = $10 per each, line value $360.
- **Check:** 36 x $10 = 3 x $120 = $360, so conversion changed neither stock value nor payable value.

Table 2 is the proof sheet. It is an illustrative, fictional matrix: unit codes and internal IDs are placeholders that each team replaces with the codes present in its own systems and units type.

Item / boundary	Source system and UOM code	NetSuite units type and unit (ID placeholder)	Conversion factor	Quantity basis	Price basis	Target transaction	Owner / effective date	Rounding and precision
tea-12, supplier PO	EDI 850, placeholder "CA"	Units type "Tea-Pack", unit "Case" (ID 102)	12 base units per case	3 cases	$120 per case	Purchase order, purchase unit	Purchasing lead / 2026-12-01	Integer cases, price 2 decimals as quoted
tea-12, supplier ASN	EDI 856, placeholder "CA"	Same type, unit "Case"	12	3 cases shipped	None carried	Reference for receiving	Integration owner / 2026-12-01	Integer cases
tea-12, item receipt	3PL receipt message, placeholder "EA"	Unit "Each" (ID 101), the base unit	1	36 each	$10 per each (derived)	Item receipt linked to the PO	Warehouse lead / 2026-12-01	Integer eaches, derived price held to 6 decimals
tea-12, stock	NetSuite on-hand	Stock unit "Each"	1	36 each	$10 per each	Inventory records	Inventory controller / 2026-12-01	Integer eaches
tea-12, sales order	Storefront line, "EA"	Sales unit "Each"	1	60 each ordered	$15 per each	Sales order	Order management / 2026-12-01	Integer eaches, price 2 decimals
tea-12, 3PL ship confirm	EDI 945 or flat file, "EA"	Unit to be confirmed in sandbox	1	28 each shipped	None carried	Item fulfillment	Integration owner / 2026-12-01	Integer eaches

Table 2 is deliberately repetitive: a "3" with no unit can be read as 3 cases or 3 eaches, and both pass a field-type check. The matrix also separates quantity basis from price basis, since a message may carry a case quantity with an each price. Each row therefore needs these fields before a message is accepted:

- **Unit code and its lookup:** the source code and the NetSuite unit it maps to, with unknown codes rejected.
- **Conversion factor and owner:** one number, one named approver and one effective date.
- **Quantity basis:** the unit in which the quantity field is expressed.
- **Price basis:** the unit to which the price applies, and which side is authoritative.
- **Rounding rule:** decimals retained and the system that rounds.
- **Replay key:** the identifier that makes a repeated message harmless.

Second example: partial shipment and remaining quantities

Worked example 2 (Hypothetical Example). A customer orders 5 cases of tea-12 (60 each) at $180 per case ($15 per each), a total of $900. The warehouse first ships 2 cases and 4 each, which is 28 each.

- **Shipped:** 28 each x $15 = $420, equal to 2 cases at $180 plus 4 each at $15 ($360 + $60).
- **Remaining:** 60 - 28 = 32 each, which is 2 cases and 8 each, valued at 32 x $15 = $480.
- **Equality check, after normalization:** 28 + 32 = 60 each, and $420 + $480 = $900.

Compare the sides in eaches, never "2 cases and 4 each" against "5 cases." Oracle documents that with advanced receiving the item receipt shows a remaining quantity for items not yet received (docs.oracle.com). Whether the remaining quantity is displayed in order units, stock units or base units for a given item is not stated in Oracle's documentation, so it is a sandbox observation to record on the sheet. Oracle also provides preferences that allow a receipt or fulfillment to exceed the remaining quantity; the contract should state whether they are on, because overage interacts with unit rounding.

The invalid-unit reject test

Every interface needs a negative test. Send a line whose unit code is absent from the units type (for example "PK6" for an item whose type has only each and case) and confirm that the message lands in an error queue instead of defaulting to the base unit. A silent default turns a 3-case order into 3 eaches. **The test** confirms that the connector's lookup does not mask the rule that the units type limits which units an item can use.

Transactions and Interface Boundaries

PO to item receipt, SO to item fulfillment

For lot and serial items, Oracle states that the receipt unit always comes from the unit on the originating purchase order (docs.oracle.com) and that an item fulfillment uses the unit on the originating sales order (docs.oracle.com). For other item types Oracle's documentation does not state the rule in the same terms, so the acceptance test is concrete: create a PO in cases, receive part, and record the unit and quantity shown on the receipt, the item's stock quantity and the PO's remaining quantity. Repeat for a sales order in eaches and a fulfillment.

Returns have their own price basis. For an item receipt created from a return, the default return cost rate is always based on the stock unit (docs.oracle.com), while the override rate is based on the units on the originating transaction. A return authorized in cases but costed in eaches is the pattern the proof sheet's price-basis column exists to catch.

Serial and lot detail, and warehouse pick decomposition

Lot quantities are entered in a quantity equal to the quantity of units on the transaction, while adjustment worksheets show only base units for these items. A warehouse that receives a lot in cases and adjusts it in eaches will see both bases in one lot history, and the contract must say which one the lot ledger uses.

Pick decomposition is a different mechanism from transaction conversion and should be tested separately. WMS lets teams set up different units for pick task decomposition, with item-level units considered before the all-items setup. Decomposition enables picking in larger units than the order's, but quantities that cannot be converted into a larger unit are not decomposed. The accounting boundary is that fulfillments still follow the order's unit (docs.oracle.com) and inventory tracking keeps using base or stock units (docs.oracle.com).

External messages: 3PL, EDI and storefront boundaries

Each external message carries its own unit vocabulary. X12, the North American EDI standards body, describes the 850 as the set for placing purchase orders (x12.org) and the 856 as a ship notice that includes the configuration of goods within the transportation equipment (x12.org). On the warehouse side, the 940 advises a warehouse to make, confirm, modify or cancel a shipment (x12.org), the 945 is used to reconcile order quantities with shipment quantities (x12.org) and the 944 reports receipt of a transfer shipment (x12.org). This article does not reproduce the X12 unit-of-measure code list, so partners should exchange explicit code tables. The United Nations' Recommendation 20 draws on the ANSI ASC X12 unit code among its sources (digitallibrary.un.org) and uses alphanumeric common codes of up to three characters (digitallibrary.un.org), so a code from one system needs an explicit lookup before another accepts it.

GS1 data gives a second hazard. Total Count of All Products is the sum of all pieces in the package regardless of GTIN (ref.gs1.org), while Count of Each Specific Product gives the quantity of each unique GTIN (ref.gs1.org). The orderable unit and shipping unit need not be the same level (ref.gs1.org). The same distinction appears in nested packaging: Amazon's example is a pallet containing 20 case packs, each containing 15 single units (sell.amazon.com), and its inbound API guidance expects sellers to know carton-level information (developer-docs.amazon.com). A feed that maps "count" to "quantity" without choosing among these will multiply or divide incorrectly. The indicator digit in a grouping GTIN-14 carries no meaning (www.gs1.org), so pack size should never be inferred from the GTIN.

Marketplace and warehouse platforms differ again. Amazon describes a case pack as a package that contains multiple units of the same product (sell.amazon.com) and a single unit as the smallest sellable package (sell.amazon.com). Shopify computes bundle availability by dividing a component's available inventory by the quantity required for the bundle (help.shopify.com), rounding down (help.shopify.com), and its order lines list component SKUs rather than the bundle SKU (help.shopify.com). ShipHero similarly derives kit availability from the available quantity of all component SKUs (software-help.shiphero.com). A 3PL platform such as Extensiv requires every SKU to carry a primary and a packaging unit (help.extensiv.com) and stores the case factor as the quantity of primary units in a secondary unit (help.extensiv.com). These are three vocabularies for one fact; the contract names the authoritative one.

Where the connector sits

An integration platform maps and transforms; it does not decide meaning. Celigo's documentation shows NetSuite imports searching for existing records using lookup criteria (docs.celigo.com) that are combined with a logical AND by default (docs.celigo.com), and matching an existing line by a mapped line ID (docs.celigo.com) or by item unless a key field is chosen (docs.celigo.com). Fulfillment rows are likewise matched by item unless a key is set (docs.celigo.com). These rules matter for units because a line keyed by item alone cannot distinguish a case line from an each line of the same item.

Celigo's 2026.5.1 release notes for its Shopify and NetSuite integration app state that fulfillment remaining-quantity checks use UOM-adjusted values (docs.celigo.com) and that refund per-unit rate calculations use UOM-adjusted quantities (docs.celigo.com). The notes also instruct customers with UOM enabled to update the quantity formula in two NetSuite saved searches before the next sync (docs.celigo.com). A connector's UOM behavior is versioned and configurable, so the contract records the connector version and the saved searches it depends on. The exact saved-search field identifiers for base versus transaction-unit quantity should be confirmed in the target account, since Oracle does not publish a full list and the Celigo release note addresses only the two saved searches it names. Where NetSuite is the master system for product data (docs.celigo.com), the unit contract belongs on NetSuite's side of the interface.

Variable weight and catch weight

Everything above assumes a fixed packaging conversion: 12 each in a case, every time. Variable-measure goods break that assumption. An item must be treated as variable measure if its measure is variable at any point in the supply chain (www.gs1.org), and GS1's own example is a supplier selling and invoicing chickens in cases of 15 by weight (www.gs1.org). GS1 US describes such products as priced, ordered or invoiced in quantities that can vary (www.help.gs1us.org), and trade measures are the net measures used for invoicing (ref.gs1.org). Where count varies but weight is fixed, or the reverse, GDSN recommends populating net content for both measures (ref.gs1.org). In GS1's scheme the measure becomes an attribute of the GTIN, so no new GTIN is required for each weight (www.gs1.org).

Regulatory definitions point the same way. The USDA's meat-labeling rules define a random weight package as one of a lot of the same product with varying weights (www.ecfr.gov). NIST Handbook 130 defines a random package as one with no fixed pattern of net contents (www.nist.gov), and a random weight package label must declare net weight, unit price and total price (www.nist.gov). Items sold in discrete, predefined bands are treated as fixed measure (www.gs1.org).

For NetSuite the safe statement is a boundary, not a recommendation: a **fixed** conversion rate cannot represent a case whose each-count or weight differs from carton to carton. Whether a business can run a catch-weight process with standard unit conversions, a customization or a separate application is a **fit-gap assessment** that must cover:

- **Valuation:** how cost is assigned when weight varies per carton.
- **Receiving and fulfillment:** how actual weight or count is captured per package.
- **Invoicing:** whether the customer is billed by actual weight or by nominal unit.
- **The 3PL message:** whether it carries both count and weight.

The assessment's output is a decision, not a default. Do not push such an item through the fixed-case contract and assume the weights average out.

Data Analysis and Evidence

The evidence for this topic is mostly documented limits and arithmetic, not market statistics. No public benchmark of unit-conversion defect rates was found, and none is invented here. *Table 3* collects the numeric limits that the contract depends on.

Numeric fact	Value	Why it matters to the contract
Units in one units type	Maximum 650 (Oracle Units Type setup page)	Caps packaging levels and variants per type
Quantity on a transaction line	Maximum 9,999,999,999 base units (Oracle Units Type setup page)	A big transaction unit multiplies toward the ceiling
Exact integers in a JavaScript Number	Only within the safe-integer range (developer.mozilla.org)	Large base-unit quantities in scripts need care
Connector flow history	30 days by default, up to 180 by plan (docs.celigo.com)	A replay window is limited by what the platform keeps
JavaScript Number digits	About 15 to 17 significant decimal digits (developer.mozilla.org)	Derived each prices belong in decimals or strings
Python decimal precision	28 places by default (docs.python.org)	A reference for exact decimal arithmetic

The table's value is its ceilings: none is a defect, but each is a place where a naive import can cross a boundary without a visible warning in the source system. A replay plan that assumes 180 days of connector history may run on a 30-day plan.

Rounding and precision

Oracle's documentation does not state how many decimal places a transaction line retains for quantity or rate when units differ, so this must be tested. A fictional $100 case of 12 gives a derived each price of $8.333333. Rounded to two decimals the each price is $8.33, and 12 x $8.33 = $99.96, a 4 cent difference per case that becomes $4.00 across 100 cases. Carry the authoritative price on the basis in which it was quoted, derive the other basis only for display, or hold at least six decimals and test what the account stores.

General computing references explain why. Python's documentation states that decimal numbers can be represented exactly (docs.python.org), that 0.1 is not exactly representable in binary floating point (docs.python.org) and that quantizing suits monetary applications that round to a fixed number of places (docs.python.org). JavaScript numbers are double-precision 64-bit binary values (developer.mozilla.org). Rounding mode also matters: Python distinguishes ties going to the nearest even integer (docs.python.org) from ties going away from zero (docs.python.org), so two systems can disagree on a half-cent. The contract names the rounding mode, the number of decimals and the system that rounds.

Idempotent replay

Interfaces fail and retry. HTTP semantics define an idempotent request as one whose intended effect is the same whether sent once or several times (www.rfc-editor.org), and say a client should not automatically retry a non-idempotent request (www.rfc-editor.org). A receipt posted twice is not idempotent unless the message carries a key. AWS recommends a unique caller-provided client request identifier in the API contract (aws.amazon.com) and a validation error when the same identifier arrives with different parameters (aws.amazon.com). Stripe's documentation compares incoming parameters to the original request (docs.stripe.com), returns the saved response on a retry with the same key (docs.stripe.com) and notes that a

key reused after pruning is treated as a new request (docs.stripe.com). An IETF Internet-Draft suggests an HTTP 422 response for a reused key with a different payload (datatracker.ietf.org); as a draft it is work in progress, not a standard.

NetSuite offers its own mechanisms. For upsert, a record is identified by its external ID and record type (docs.oracle.com), and each record instance holds only one external ID value, so one integration should own that field. Oracle's reliability guidance also tells integrators to make update logic idempotent. The replay test for the contract: post the 36-each receipt message twice with the same key and confirm one receipt, then post it with the same key and a changed unit and confirm a reject.

Physical-count tie-out

The last row of the proof sheet is a count. The U.S. Government Accountability Office (GAO) defines cycle counting as a method in which a portion of inventory is counted on a schedule (www.gao.gov) and cautions about a unit-of-measure trap: a box of many items with a record quantity of one can bias a counter to record 1 (www.gao.gov). It also defines a blind count as counting without access to the on-hand balance (www.gao.gov), and notes that accurate counts verify that on-hand balances agree with financial and logistical records (www.gao.gov).

Worked example 3 (Hypothetical Example). A count finds 7 full cases and 5 loose each of tea-12: 7 x 12 + 5 = 89 each. NetSuite shows 89 each, so variance is zero and the value at an illustrative $10 per each is $890. If the counter keys "7" with the unit left as each, the sheet shows 7 against 89, a false variance of 82 each unrelated to stock. IRS Publication 538 asks for consistent inventory practices from year to year (www.irs.gov), and under IFRS inventory losses are expensed in the period they occur (ifrs.org) and inventories are measured at the lower of cost and net realisable value (ifrs.org), so a unit error in a count can become a valuation error. This is general context, not accounting advice; consult a qualified accountant.

Testing and Approving a Catalog Change or Migration

The test sequence

Run these in a sandbox that mirrors production units types, recording the observed unit, quantity and price on each result.

- **PO to receipt:** create a PO in cases, receive part, then the rest; record the unit on the receipt, remaining quantity and stock quantity.
- **SO to fulfillment:** create an SO in eaches, fulfill part; record the unit and remaining quantity.
- **Returns:** return one case of a three-case shipment; record the rate basis.
- **Packaging change:** add a 24-each case as a new dated unit and confirm old PO lines keep the old factor.
- **Serial and lot:** receive a lot in cases and adjust in eaches; for a serial item confirm one serial per base unit.
- **Pick decomposition:** run a wave with and without decomposition units and compare fulfillment units.
- **Invalid unit:** send an unknown code and confirm rejection.
- **Idempotent replay:** repeat a message with the same key.

- **Count tie-out:** count 7 cases and 5 each and tie to NetSuite.
- **Report check:** confirm what a saved search shows for quantity, since Oracle states that reports show units based on the units used in transactions.

Report quantity versus transaction quantity

Oracle does not publish a field-by-field list of which saved-search columns hold base-unit versus transaction-unit quantity. Celigo's release note distinguishes the base-unit quantity field from a transacted-unit field, quantityUOM, that its flows now read (docs.celigo.com). Before building a dashboard or reconciliation, create a saved search with every candidate quantity column, starting with quantity and quantityuom, on a known case-denominated line and write down which column equals 3 and which equals 36. Verify the answer in your own sandbox, not from community posts.

Approval criteria and rollback

An approval should be a short signed sheet with these elements:

- **Base unit and units type** named, with a note that an in-use item's units type cannot be changed.
- **Reconciliation:** quantities and values in eaches tie before and after, per item, at zero tolerance.
- **Owner and effective date** for each conversion factor.
- **Rollback criterion:** a defined trigger (for example, any unit-mismatch reject rate above an agreed threshold on day one) and a defined action (pause the flow, keep the replay key log).
- **Open-transaction treatment:** a written statement, confirmed in the sandbox, of what happens to open POs and SOs.
- **Variable-weight exclusion:** catch-weight items go to the fit-gap assessment.

Teams without an internal integration owner may bring in an implementation partner for sandbox testing and connector configuration, and the contract is useful with or without one.

Implications and Future Directions

Three directions follow from the evidence. First, unit conversion is becoming a connector release-note topic. A documented UOM change in a mainstream NetSuite and Shopify integration app suggests integration owners should subscribe to release notes and re-run the proof sheet after each upgrade, not only at go-live. Second, packaging standards are converging on explicit hierarchy data: GDSN makes a trade item unit descriptor mandatory (ref.gs1.org), and a despatch unit must be identified with a GTIN (ref.gs1.org). GS1 Australia's data dictionary shows the pattern with a worked each, inner pack and case example (www.gs1au.org), and the UN's Recommendation 20 notes a single worldwide list of unit codes (digitallibrary.un.org) that sits alongside a separate code list for package types (digitallibrary.un.org). Supplier feeds that carry the hierarchy help only if the receiving system maps it to the units type deliberately.

Third, replay safety is moving toward a standard: the IETF idempotency key proposal is still an Internet-Draft (datatracker.ietf.org), while Shopify's inventory adjustment mutation already requires an idempotency key in recent API versions (shopify.dev). Warehouse platforms keep their own limits: Extensiv does not support

decimals for primary, secondary and packaging units (help.extensiv.com), which constrains fractional-case designs on that side of an interface.

Open questions remain that need a sandbox: the precision of stored rates, the exact saved-search fields for base and transaction quantities, how a base-unit or default change affects historical and open transactions, and current user-interface and REST support for editing units.

Frequently Asked Questions (FAQs)

What is a NetSuite units type and how is one set up?

A units type is a family of units with one base unit and a conversion rate for each other unit. After creating it, assign it to an item and the purchase, stock and sale units default to the base unit.

What is the unit of measure conversion rate for case to each?

If the each is the base unit, the case's rate is the number of eaches in one case (for example 12), because the rate is the quantity of base units equal to one unit.

Purchase unit versus stock unit, and base unit versus sales unit?

The purchase unit is the default on purchase orders and the stock unit is the default for inventory adjustments. The base unit has a conversion rate of one, and the sales unit is the default on invoices; it may be larger than or equal to the base.

Which unit does an item fulfillment or item receipt use?

For lot and serial items, Oracle documents that receipts follow the originating PO's unit and fulfillments follow the originating SO's unit. For other items, confirm in a sandbox.

How can rounding errors be avoided?

Treat the quoted basis as authoritative, hold derived prices to several decimals, name the rounding mode and test what the account stores.

Can the units type of an existing item be changed?

Oracle states that an item's units type cannot be changed once assigned. Plan any correction as a controlled, sandbox-tested change.

Conclusion

Case-to-each conversion in NetSuite is a contract problem before it is a configuration problem. The platform documents a clear model: one base unit per units type, a conversion rate for each other unit, defaults by transaction type and restrictions on what can change after assignment. The proof sheet turns that model into an acceptance test by naming the unit, quantity basis, price basis, owner and rounding rule at every boundary from supplier PO to 3PL confirmation.

The decisions to take first are the base unit, because an in-use item's units type is fixed; the owner of each packaging factor, because packaging standards treat a pack-size change as a new identity; and the price basis, because the same line value must survive the move from cases to eaches. The tests to run next are the ones that need a sandbox account: stored precision, reporting fields, partial and reverse flows, and the effect of a default change on open transactions. Fixed packaging fits the contract; variable-weight goods need a separate fit-gap assessment. With those pieces in place a team can approve a catalog change with a reconciliation, a named owner and a rollback trigger before peak-season volume arrives.

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/chapter_N2211898.html
2. https://ref.gs1.org/guidelines/gdm-implementation/1.3.0
3. https://www.gs1.org/1/gtinrules/en/rule/270/packcase-quantity
4. https://docs.celigo.com/hc/en-us/articles/50306423069723-Shopify-NetSuite-integration-app-release-notes-2026-5-1
5. https://ref.gs1.org/standards/genspecs/gscn/2025/GSCN-25-351-Glossary.pdf
6. https://docs.python.org/3/library/decimal.html
7. https://aws.amazon.com/builders-library/making-retries-safe-with-idempotent-APIs
8. https://www.irs.gov/publications/p538
9. https://docs.oracle.com/en/cloud/saas/netsuite/ns-online-help/section_N2212390.html
10. https://docs.oracle.com/en/cloud/saas/netsuite/ns-online-help/section_N2320172.html
11. https://docs.oracle.com/en/cloud/saas/netsuite/ns-online-help/section_N2212143.html
12. https://docs.oracle.com/en/cloud/saas/netsuite/ns-online-help/section_0207111545.html
13. https://documents.gs1us.org/adobe/assets/deliver/urn:aaid:aem:3b583151-bc32-4223-a190-5bb1ae565b0e/Guideline-An-Introduction-to-the-Global-Trade-Item-Number-GTIN.pdf
14. https://docs.oracle.com/en/cloud/saas/netsuite/ns-online-help/section_N2214525.html
15. https://www.gs1.org/docs/barcodes/GS1_General_Specifications.pdf
16. https://docs.oracle.com/en/cloud/saas/netsuite/ns-online-help/section_N2412119.html
17. https://docs.oracle.com/en/cloud/saas/netsuite/ns-online-help/section_N2199328.html
18. https://docs.oracle.com/en/cloud/saas/netsuite/ns-online-help/section_0207120728.html
19. https://x12.org/products/transaction-sets
20. https://x12.org/index.php/node/4398
21. https://x12.org/node/4444
22. https://x12.org/node/4447
23. https://x12.org/node/4446
24. https://digitallibrary.un.org/record/431844/files/%5BE_ECE_%5DTRADE_CEFACT_2001_20-EN.pdf
25. https://sell.amazon.com/blog/case-packs-and-pallets
26. https://developer-docs.amazon.com/sp-api/docs/fulfillment-inbound-api-v2024-03-20-use-case-guide

27. https://help.shopify.com/en/manual/products/bundles/eligibility-and-considerations
28. https://help.shopify.com/en/manual/products/bundles/shopify-bundles
29. https://software-help.shiphero.com/hc/en-us/articles/4419344900749-How-Kit-Availability-is-Calculated
30. https://help.extensiv.com/en_US/understanding-units-of-measure
31. https://help.extensiv.com/defining-units-of-measure
32. https://docs.celigo.com/hc/en-us/articles/360015780951
33. https://docs.celigo.com/hc/en-us/articles/360052710532-Import-NetSuite-line-items
34. https://docs.celigo.com/hc/en-us/articles/49643858688667-Sync-products-and-inventory-from-NetSuite-to-Shopify
35. https://www.help.gs1us.org/is-this-item-variable-measure
36. https://ref.gs1.org/guidelines/tiig
37. https://www.gs1.org/1/gtinrules/en/rule/138/trade-item-sold-in-variable-measure-varies-in-total-weight
38. https://www.ecfr.gov/current/title-9/chapter-III/subchapter-A/part-317/subpart-A/section-317.2
39. https://www.nist.gov/system/files/documents/2025/12/23/2026_HB130-IV_A_UPLR-20251215.pdf
40. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Number
41. https://docs.celigo.com/hc/en-us/articles/360048814732-Intro-to-Error-Management-2-0
42. https://www.rfc-editor.org/rfc/rfc9110.html
43. https://docs.stripe.com/api/idempotent_requests
44. https://datatracker.ietf.org/doc/draft-ietf-httpapi-idempotency-key-header
45. https://docs.oracle.com/en/cloud/saas/netsuite/ns-online-help/article_0807093604.html
46. https://www.gao.gov/assets/660/651157.txt
47. https://ifrs.org/issued-standards/list-of-standards/ias-2-inventories
48. https://www.gs1au.org/services/data-and-content/national-product-catalogue/npc-data-dictionary/data-attribute/trade-item-unit-descriptor-code
49. https://digitallibrary.un.org/record/635432/files/%5BE_%5DECE_TRADE_C_CEFACT_2008_6-EN.pdf
50. https://shopify.dev/docs/api/admin-graphql/latest/mutations/inventoryAdjustQuantities

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.