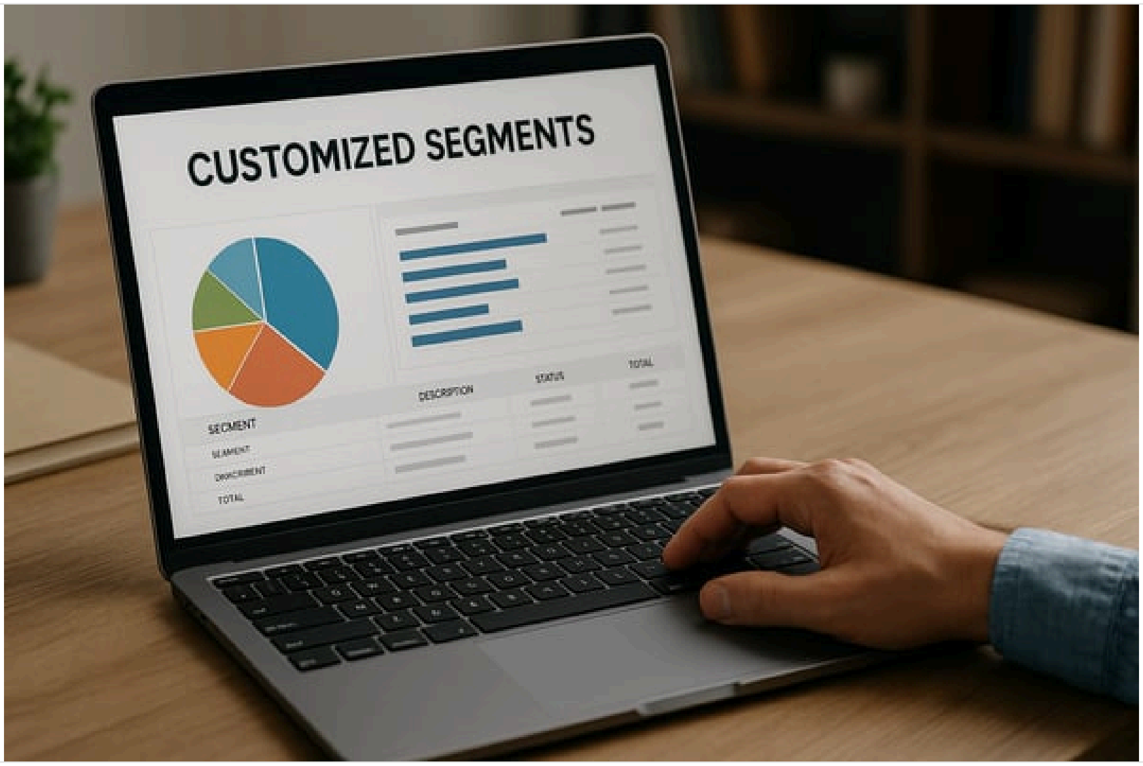
Configuring NetSuite Custom Segments for Financial Reporting

Published August 26, 2025 70 min read



Using Customized Segments in NetSuite: A Comprehensive Guide

Introduction

Modern businesses often need to categorize financial and operational data in ways that go beyond NetSuite's standard classifications. NetSuite administrators, financial controllers, and ERP consultants frequently leverage **custom segments** to create additional dimensions for reporting and analysis. Custom segments are user-defined classification fields (similar to Department, Class, or Location) that allow unlimited, flexible tagging of records for various purposes. This guide provides a professional-level deep dive into using custom segments in NetSuite – from understanding what they are and how they differ from standard segments, to configuring and deploying them, to real-world use cases across industries. We will also cover advanced topics such as SuiteGL (General Ledger) impact, SuiteScript considerations, reporting/analytics, integration, security/permissions, and ongoing maintenance. By the end, you should have a thorough understanding of how to harness custom segments to meet unique business requirements while avoiding common pitfalls.

Understanding Standard vs. Custom Segments in NetSuite

Standard Segments (Department, Class, Location)

NetSuite provides three main native segments for classifying transactions and records: **Department**, **Class**, and **Location** (a fourth segment, **Subsidiary**, exists in OneWorld accounts for entity-level classification). These standard segments are built-in fields used to organize financial data:

- **Department** is typically used to represent cost centers or internal teams (organizational units that incur costs but may not directly generate revenue).
- **Class** often denotes profit centers, lines of business, product lines, or revenue streams. For example, classes might distinguish different product categories or business units for income tracking.
- **Location** usually represents physical or virtual locations such as offices, warehouses, regions, or stores. It can be used for inventory locations or geographic divisions (not to be confused with subsidiaries).

Standard segments are available on most transactions and records by default, and they can be hierarchical (e.g. Departments and Classes support parent-child rollups) and importable via CSV. Tagging transactions with these segments allows financial reports to be filtered or broken down by department, class, or location, instead of needing separate general ledger accounts for each category. For instance, rather than maintaining separate salary expense accounts for each department, a single account can be used with department segments to produce departmental expense reports. Standard segments also support certain native features: you can restrict user access to records by department or class (for data security), and NetSuite ensures that transactions balance by subsidiary for financial consolidation.

However, the limitation is that you only get these three classification dimensions out-of-the-box. They may not cover all the ways a company wants to categorize data. Standard segments also cannot have dependencies between each other (e.g. you cannot natively make the list of Classes filtered by the selected Department), and each standard segment field only allows one value per record (no multi-select). This is where **custom segments** come into play.

Custom Segments – Extending Classification Dimensions

Custom segments are a NetSuite SuiteGL feature that lets you define new classification fields analogous to Class, Department, or Location to capture business-specific data categories. With custom segments, you can create an **unlimited number of additional segments** to tag records in ways that standard segments don't support. Each custom segment has a defined list of possible values (maintained as a custom record type) and can be applied to specific record types or transaction types as needed. In effect, custom segments give you extra "buckets" to classify and report on data according to your organization's unique needs.

The table below summarizes key differences between standard and custom segments:

ASPECT	STANDARD SEGMENTS (DEPT/CLASS/LOC)	CUSTOM SEGMENTS (USER-DEFINED)
Quantity Available	Three fixed fields (plus Subsidiary in OneWorld)	Unlimited – create as many as needed
Example Uses	Dept = cost center; Class = product line; Location = office/region	Any classification (e.g. Region, Project, Fund, Channel, Cost Center, etc.)
Hierarchy Support	Yes – Departments/Classes can have parent-child hierarchy (Locations can be structured as sub-locations)	Yes – segment values can be hierarchical (parent/child values)
Multi-Select Allowed	No – one value per record (one Dept, one Class, etc.)	Optional – can designate segment as List/Record (single select) or Multiple Select (Source: docs.oracle.com)
GL Impact	Always part of GL posting by design (appear on GL Impact and financial reports)	Configurable – "GL Impact" box decides if segment posts to GL (irreversible once enabled)
Reporting	Can filter and subtotal standard financial reports by these segments natively	Custom segments appear as filters/columns on reports; can be added as financial report dimensions if GL Impact
Dependency Filtering	Not directly supported between standard segments (cannot filter one by another)	Supported – can filter segment values based on selections of other segments or standard fields (Dept, Class, Loc, Subsidiary)
Role-Based Permissions	Standard segments visible to any user with record access; support record-level data restriction by Dept/Class via roles	Granular control – can restrict which roles can see or set the segment on records, manage its values, or use it in reports (no native record restriction by segment value)
Maintenance of Values	Values managed in Lists > Employees/Classes/Locations (no external ID on values)	Values are custom records (can have External IDs, hierarchies, etc.). Can be added via UI or CSV, inactivated or even deleted (with special settings if GL-impacting).

As the table suggests, custom segments function very much like the native segments – once configured, they can be selected on records and used in reporting similarly – but they are *highly customizable*. For example, you might create a custom segment called "Region" to classify transactions by geographical region (if Location is being used for something else), or a segment called "Project" to tag transactions to projects for profitability tracking. Each custom segment you create introduces a new field (or set of fields) in the system that behaves like an additional dimension for your data. These fields can be placed on transaction forms (body or line-level) and on other record types (customers, vendors, items, etc.) as you specify, allowing users to tag records with the appropriate segment values.

Under the hood, NetSuite creates a custom record type to hold the values for each segment, and behind that it generates internal fields to link transactions to those values. This architecture gives custom segments some powerful capabilities. Notably, custom segments **natively integrate with the General Ledger when configured to have GL impact**. If you check the "GL Impact" option for a custom segment, any value chosen will flow through to the GL Impact report of a transaction, and standard NetSuite financial reports can use the segment as a reporting dimension. In other words, a GL-impacting custom segment acts as a true accounting segment or "financial dimension" alongside class, department, etc. If the segment is not marked GL-impacting, it can still be used for classifying records and in saved searches or custom reports, but it won't appear on the formal financial statements or GL postings.

When to use custom segments vs. other custom fields? Generally, you should use a custom segment when you need a classification that behaves like a financial segment – particularly if you plan to pivot financial reports or budgets on that field or need it visible on GL postings. If the additional field you need is purely informational or transactional (and not required for GL or enterprise-wide reporting), a simpler custom field (List/Record or multi-select) might suffice. Keep in mind that custom segments carry more complexity: they are global classifications that affect reporting structure. As one NetSuite expert notes, *"use a Custom Segment if you need to separate your financials by this attribute. If you don't... don't."*. In fact, custom segments shine in scenarios where in the past one might have added dozens of accounts or repurposed class/department for something they weren't meant for. By using a custom segment, you can dramatically simplify your Chart of Accounts while achieving the granular reporting you need.

It's also worth noting some advantages of custom segments over regular custom list fields in NetSuite:

- **Unified deployment**: A custom segment can be deployed to many record types in one configuration screen, instead of creating separate custom fields for each record type. You simply check which record types and transactions the segment should appear on, all in the segment setup (Source: netsuite.alphabold.com)(Source: netsuite.alphabold.com). This ensures consistency and saves time.
- **Native reporting integration**: Once a custom segment is created (especially if GL Impact is enabled), it *automatically* becomes available in standard report filters and columns without extra customization (Source: netsuite.alphabold.com). By contrast, with a normal custom field you might

need to manually customize reports or searches to include it. Custom segments are treated as first-class dimensions in NetSuite's reporting tools.

- **GL line tagging**: Custom segments (with GL impact) can be tagged on the line-level of GL postings, just like standard segments (Source: netsuite.alphabold.com). Standard custom fields cannot attach to individual GL impact lines. This means segments can be used in true financial statements and even the new Balance Sheet by Segment capabilities (more on that later), whereas ordinary custom fields are more for supplemental info.

In summary, custom segments extend NetSuite's native classification framework to accommodate virtually any reporting dimension your business requires. Next, we will explore how to enable and configure custom segments in detail.

Configuring and Deploying Custom Segments

Using custom segments effectively starts with proper configuration. Below is an in-depth guide on how to enable the feature, create a custom segment, and set up all the relevant options. We'll also touch on SuiteGL and SuiteScript aspects where applicable.

Enabling the Custom Segments Feature

Before you can create custom segments, the feature must be enabled in your NetSuite account. Navigate to **Setup > Company > Enable Features**, and under the SuiteCloud (or sometimes labeled SuiteGL) subtab, check the box for **Custom Segments**, then Save. (If you don't see this option, you may need the appropriate permissions or NetSuite edition that includes SuiteGL). Once enabled, a new menu item will appear under **Customization > Lists, Records, & Fields** for Custom Segments.

Creating a New Custom Segment

To create a custom segment, you need Administrator access or a role with the Custom Segments permission (Create or Full level). Follow these steps to define a new segment:

1. **Start a New Segment Record:** Go to **Customization > Lists, Records, & Fields > Custom Segments > New**. This opens the custom segment definition form.

2. **Label (Name):** Enter a **Label** for the segment – this is the user-friendly name that will appear on forms (e.g., "Region", "Project Code", "Fund", etc.). The label must be unique across all existing custom segments, custom fields, and standard classification names. NetSuite will prevent duplicates or reserved names (you cannot use "Class", "Department", "Location", "Subsidiary", or "Account" as the label, for example).

3. **ID:** Optionally specify an **ID** (script ID) for the segment. NetSuite will prefix any custom segment ID with "`cseg`" automatically. A common practice is to provide an ID that mirrors the name (for example, if Label is "Channel", you might enter "_channel" so the system generates `cseg_channel`). This ID cannot be changed later, so use a concise, alphanumeric name. As of NetSuite 2019.1, new custom segments use a unified ID model by default (the "Use as Field ID" box is no longer shown for new segments). This means the one ID will represent the segment across all contexts (body, line, etc.), simplifying SuiteScript references. *(Note: If you are editing an older segment from before 2019.1, you may see a "Use as Field ID" option. Changing that setting on an existing segment can break SuiteScripts, imports, or integrations that reference the old field IDs, so be cautious)*.

4. **Type (List/Record vs Multi-Select):** Choose the field **Type** for the segment – either **List/Record** (single-select) or **Multiple Select**(Source: docs.oracle.com). By default, segments are single-select list fields, meaning a user can choose one value. If you set it to Multiple Select, users will be able to select multiple values from the list on a single record (much like a multi-select custom field). *Be aware:* multiple-select segments have some limitations in terms of certain filtering and reporting contexts (Source: docs.oracle.com), so only use multi-select if you truly need one record to carry multiple segment values. In most financial use cases, single-select is the norm.

5. **Inactive (if applicable):** If you are setting up a segment for future use or testing and don't want it active yet, you can mark it **Inactive**. An inactive segment will not appear on forms for users (Source: docs.oracle.com). (Typically, you will leave it active by default.)

6. **Display Type:** Choose the **Display Type** – Normal, Disabled, or Hidden. In most cases, you'll use **Normal**, which means the field is visible and editable on forms. **Disabled** means the field shows up but is read-only (could be used if you plan to set it via scripts or defaults only). **Hidden** means it won't show on the form at all for users. Hidden segments must have a default value or sourcing because users can't set them manually. *Note:* Hidden does **not** equate to secure – the value is still in the page HTML, just not rendered in the UI, and can be seen by viewing source. Hidden segments can only be set via script (user event, scheduled, or Suitelet scripts) since users can't interact with them.

7. **Show In List (Optional):** If this segment is applied to custom record types (or you plan to view segment values as a column in some record lists), you can check **Show In List** to display it as a column in those lists. This is more relevant when segment values might be shown in an overview of another record.

8. **Filtering (Dependencies):** To establish dependent filters (conditional values) for this segment, use the **Filtered by** field. Here you can select one or multiple other classification fields (including standard Class, Department, Location, Subsidiary, or even other custom segments) that will act as parents for filtering. If you select any fields here, you are indicating that the availability of each

segment value may depend on one or more of those other field selections. For example, if you are creating a "Sales Channel" segment that should vary based on a "Region" segment selection, you would set "Filtered by: Region" on the Sales Channel segment. You can choose multiple filters (Ctrl+click) if the values depend on combinations of fields (Source: netsuitedocumentation1.gitlab.io). After saving the segment, you will configure the specific allowed combinations per value (we'll cover that shortly). If you don't need dependent filtering, leave **Filtered by** blank.

9. **GL Impact:** If this segment represents a financial classification that you want to flow to the general ledger, check the **GL Impact** box. **Enabling GL Impact means that when this segment is used on a transaction, the segment's value will appear on the GL Impact page of that transaction and in associated GL reports**. This is crucial if you intend to use the segment in financial reporting (income statements, balance sheets, budget vs actual, etc.). For example, if you create a segment "Product Line" and want an Income Statement by Product Line, you must check GL Impact so that transaction postings carry that tag. **Important:** Once you save the segment with GL Impact on, you cannot later turn it off. NetSuite locks this setting because removing a GL-impacting dimension could invalidate historical financial data. Therefore, decide upfront if the segment truly needs to hit the GL. If it's more of an operational or analytical category not needed in the formal ledger, it may be wise to leave GL Impact unchecked for flexibility. (Also note, if a segment has GL impact, you cannot populate or change it on transactions in closed periods without reopening the period, since it would alter the GL classification.) In short: **only enable GL Impact when necessary for reporting** – it's a one-way decision.

10. **Help/Description (Optional):** You can enter field-level help text in the **Help** field to guide users (this appears as a popup when users click the field label). A **Description** can be added for admin/internal notes about the segment's purpose (visible only on the config page). These are good places to document the intended use of the segment so future administrators or users understand its meaning.

11. **Values:** Next, define the list of **Values** for your segment. On the **Values** subtab, add a line for each possible value (just like adding entries to a custom list). For each value you can specify:

```
*   **Value (Name):** the name of the segment value (e.g., "North America", "Europe" fo
*   **ID:** an optional script ID for the value (often not needed unless you will refere

*   **Parent**: if you want a hierarchical list, you can designate a parent value here 

*   **Active**: you can uncheck this to inactivate specific values if needed (only acti

*   **Filtered By columns**: If you set _Filtered by_ earlier, additional columns appea


You can add as many values as needed. Custom segments support a large number of values
```

12. **Application & Sourcing:** On the **Application & Sourcing** subtab, specify **where this segment will appear** and any dynamic defaulting logic. You will see various categories and record types listed here, usually grouped by: Transactions, Transaction Columns, Entities, Items, CRM, Custom Records, etc. Check the box next to each record or transaction type where you want the segment to be used. For example:

```
*   Under **Transactions**: you might see "Sales Transactions", "Purchase Transactions",

*   Under **Transaction Columns**: you can specify which transactions should have the se

*   Under **Entities**: you can check Customer, Vendor, Employee, etc., if you want the 

*   Under **Items**: you can apply the segment to item records if it's something releva

*   Under **Event, CRM, Other records**: if needed, apply to additional record types or


Essentially, this subtab is one-stop configuration for deploying the segment on various

**Dynamic Default (Source List):** If you apply the segment to multiple related record 
```

13. **Validation & Defaulting:** On this subtab, you can enforce requirements and set a fixed default if desired.

```
*   **Mandatory:** To require that users fill in this segment on certain records, check

*   **Default Value:** In the **Default Selection** field, you can choose a static defa
```

14. **Permissions:** The **Permissions** subtab is crucial for security and governance. Here you control which roles can interact with this segment and to what extent. There are three permission types for segments:

```
*   **Value Management Access Level:** Who can create/edit the segment’s values (the li

*   **Record Access Level:** Who can see and set the segment on records. This controls v

*   **Search/Reporting Access Level:** Who can use the segment in saved searches and rep


Configure these permissions according to your governance policy. If you don’t add a role
```

15. **Save:** Once all settings are configured, click **Save**. NetSuite will create the custom segment and an associated custom record type (with the same name as the segment) to store its values. The segment is now active in the system.

After saving, you may do some *post-creation steps*: for example, if you set up filtering, you’ll want to go back to the **Values** subtab, click **Set Filters** on each value and choose which parent segment values (or departments, etc.) allow that value. If you didn’t add all your values initially, you can add more now either via the segment screen (Add button on Values sublist) or by editing the new custom record type that was created for the segment. You can also adjust the **Display Order** of segments (under Customization > Lists, Records, & Fields > Custom Segments > Set Display Order) to control the order in which multiple segments appear on forms and in GL Impact reports. This is useful if you have many custom segments – you might want the most important one to show first on transaction forms.

Tip: If you have multiple NetSuite environments (e.g., Sandbox and Production) and need to replicate segments, you have options: custom segments are supported in SuiteCloud Development Framework (SDF) for deployment in XML form. There is also a “Copy to Account” utility that lets you copy a segment definition to another account with one click. These can save effort in multi-account scenarios.

At this point, the custom segment is configured and ready to use. Next, we’ll see how to deploy and use it in practice on records and transactions.

Deploying Custom Segments on Records and Transactions

Once a custom segment is created and applied to record types, it will begin appearing on those records' forms. As an administrator, you should review your form layouts to ensure the segment field is placed appropriately for users. NetSuite will automatically add the segment to the "custom" field group or column on standard forms, but you might want to reposition it or include it in specific form layouts.

Using Segments on Transaction Body vs. Lines: If you enabled the segment on transaction body (header), it will show up as a field in the main section of the transaction (typically under Class/Dept/Location or in a custom field area). The user selects one value for the whole transaction. Example: a "Region" segment on an Invoice – the user picks the Region that applies to that invoice's revenue. If instead (or additionally) you enabled it on transaction lines, then each line item (or expense line, etc., depending on sublist) will have a column for that segment. For example, a "Project" segment on an Expense Report lines lets the employee tag each expense line to a different project. Line-level segments are extremely useful for granular tracking of revenues and expenses when multiple segments could apply within one transaction.

Custom Forms: If you use custom forms, you may need to edit those forms to include the segment field. Navigate to Customize Form for the record and ensure the custom segment is added to an appropriate field group or sublist and not hidden. If the segment was marked mandatory in setup, make sure it's visible on all forms so users can fill it in; a hidden mandatory field will block record submission. Also consider the form's default values or source lists: if you want a segment to default from an entity, use the Source List as configured; if you have multiple forms for different contexts, verify each behaves as expected.

Dynamic Default in Action: With Source List configured, you can test that behavior. For instance, if your segment "Division" is sourced from the Customer record, edit a Customer to assign it a Division value. Then create a new transaction for that customer – you should see the Division segment auto-fill with the customer's value. Users can override it on the transaction if needed (unless you set the field to disabled).

Dependent Filtering in Action: If you set up filtering, try selecting the various "filter by" fields on a form and observe the segment's available values. They should narrow down according to your filter rules. For example, if "Profit Center" segment is filtered by Department, when Department = "Men's Department" on a sales order, maybe the Profit Center segment only shows values relevant to Men's (as per your configuration, e.g., excludes "Jewelry" if that's only for Women's Dept). This greatly improves user accuracy by preventing invalid combinations.

Multi-Select Behavior: If the segment is multi-select, users will get a list with checkboxes and can pick multiple. The values will usually display separated by commas on the record. Think carefully about reporting in such cases – a transaction with multiple segment values might be counted in multiple categories in a report, which can complicate sum totals. Most financial reporting works best with single-valued segments (or you'd need to use custom reports that explode those values). Multi-select custom segments are more common for non-GL cases, like tagging transactions with multiple attributes for analysis.

General Ledger Impact: For segments marked as GL Impact, it's important to understand how they manifest in the accounting. When a user saves a transaction with the segment filled, the GL Impact page (accessible from the transaction's menu or via the GL Impact subtab on the record) will list all ledger postings (debits/credits) and include a column for the custom segment's value on each line. If the segment is applied at the body level, typically all GL lines inherit that body value. If it's at line level, each GL line corresponding to a specific item/expense will carry that line's segment value. This means you can filter GL reports (like a Trial Balance or Income Statement) by that segment – effectively getting a P&L for each segment value. For example, if you run an Income Statement and add a column for your custom "Division" segment, NetSuite will automatically segregate amounts by that segment.

One advanced feature to note: starting in NetSuite 2020.1, there is an ability to **balance transactions by custom segment** (for OneWorld accounts). This *Custom Segment Balancing* ensures that if you want a balance sheet by, say, Division, NetSuite will automatically post inter-segment due to/from entries to keep the balance sheet balanced for each Division (Source: netsuite.alphabold.com)(Source: netsuite.alphabold.com). In the past, balance sheets could only balance by subsidiary, but now if you designate a segment as a balancing segment, the system can enforce Assets = Liabilities+Equity within each segment value. For instance, if you have an interdivision transaction that would throw off the balance sheet by division, NetSuite will create adjustment lines to fix that, given the feature is enabled. To use this, you must mark the segment as GL Impact (and likely check a "balancing" option in setup), but it can be transformative for producing division-level balance sheets or fund balance reports for nonprofits (Source: netsuite.alphabold.com)(Source: netsuite.alphabold.com). Keep this in mind if your use case involves balance sheet segmentation (e.g., tracking asset/liability by project or fund – something not previously feasible in NetSuite standard functionality).

SuiteGL Custom GL Plugins: If you are using the SuiteGL Custom GL Lines plug-in for advanced accounting logic, custom segments play nicely with it. The plugin implementation can programmatically set or read custom segment values on the lines it creates. For example, if you have a plug-in that allocates expenses to a certain account, you can also allocate them to a specific custom segment value by setting that value on the new GL lines in script. The plugin can also read what segment value is on the transaction or line to use in its logic. This is helpful for automation like spreading costs by departments or tagging generated journal lines with the same project segment as the source transaction.

SuiteScript Access: In general, SuiteScript (2.x) treats custom segments similarly to other fields. They usually have field IDs like `cseg_xyz` or `custbody_xyz` depending on unified ID usage. You can use `record.setValue({ fieldId: 'cseg_mysegment', value: someId })` or search filters on `cseg_mysegment` just like a normal field. One thing to be aware of: if a custom segment is not exposed on a form that a script is using (especially in client scripts or user events using current record), it might not be available until you provide the field or load the record with an appropriate form. For example, a developer on StackOverflow found that when creating a Sales Order via script, the custom segment field `cseg1` was only accessible after specifying a customForm that includes that field. The solution was to either load the record with a form that has the segment or ensure the field is exposed. In server-side scripts like scheduled scripts or RESTlets, you can generally set the value directly by internal ID since you're not bound by form layouts. Also note, if you changed the "Use as field ID" option on an existing segment, any script or saved search using the old field IDs would need updating. Always test scripts when deploying new segments to ensure the field is captured.

To summarize deployment: once configured, custom segments become part of users' daily data entry and provide new slicing options for your data. Ensure training for users so they know when and how to use the new fields (for example, instruct salespeople to pick the correct "Channel" on sales orders, or require accountants to fill in "Project Code" on journal entries). With the segments in use, the next step is to leverage them in reporting and analytics.

Industry Use Cases and Benefits of Custom Segments

One of the best ways to appreciate custom segments is through real-world use cases. Different industries have leveraged this feature to solve reporting challenges and add granular tracking without cluttering the general ledger. Below are several scenarios illustrating how custom segments can be deployed, along with their benefits:

Project Profitability and Job Costing

For professional services organizations, construction firms, or any project-driven business, tracking revenue and costs by project is critical. NetSuite's standard customer-job hierarchy can track project transactions (using the Customer:Job on transactions), but some companies prefer a dedicated dimension for internal project codes (especially if projects aren't tied to customers or if they want to track internal R&D projects). By creating a **Project** custom segment, you can tag every revenue and expense transaction with a project identifier. For example, a consulting firm might create a segment called "Project Code" with values for each active project. They apply it to sales invoices, vendor bills, time entries, etc. Now, project managers and controllers can run profitability reports by project – income and expenses – without needing separate accounts per project. All GL postings carry the project code as a dimension, so a **Profit & Loss by Project** becomes a simple report filter or column selection. This approach enhances visibility into project margins in real-time. One NetSuite consultant noted that *"Project Profitability"* reporting is a common requirement that segments can address. The benefit is a unified chart of accounts (e.g., one revenue account instead of one per project) while still getting granular P&L reporting. Additionally, companies can budget versus actual by project (using saved searches or analytics) if needed. **Tip:** If using NetSuite's Project Management module, consider if the built-in project (as a customer subtype) meets your needs; but for many, a custom segment is simpler for internal tracking, especially if you don't manage project tasks in the system. Be cautious not to duplicate data: if you already specify a project via another field, you may not need a segment. Use a segment when it fills a gap (e.g., internal projects that aren't set up as NS Project records).

Regional or Geographic Reporting

Global companies often want to analyze financial results by region or market. Sometimes the NetSuite **Location** field is used for this, but Location may already represent physical warehouses or retail stores, which might be too granular or differently purposed. By defining a **Region** custom segment (with values like North America, EMEA, APAC, etc.), you gain a flexible way to tag transactions and even master records with a regional identifier. For instance, you could tag each Customer with a Region, and have that default onto transactions for that customer (via dynamic sourcing) – yielding regional sales reports effortlessly. Alternatively, tag each transaction's body with a region based on where the sale is recognized. Regions can also be hierarchical (e.g., Americas parent with USA, Canada, LATAM children) to allow roll-ups. **Benefit:** Management can see consolidated financials by region across subsidiaries (if regions span multiple legal entities) or within a single subsidiary. It also facilitates regional profitability analysis without the overhead of separate subsidiaries or segmenting your chart of accounts. A **Regional segment** can be used in combination with standard segments; e.g., filter a P&L report by Department = "Consulting" and column = Region to see how each region's consulting department is doing. Regional segments have been used in industries like retail (to compare performance of APAC vs NA markets) and software companies (to break out revenue by sales region for forecasting). Using a segment ensures consistency – you can even filter available region values by subsidiary or other fields if certain regions only apply to certain entities.

Granular Cost Center or Profit Center Tracking

Companies that have complex organizational structures sometimes outgrow the single Department or Class fields. For example, a large enterprise might have **Cost Centers** that roll up into Departments, or multiple dimensions like **Business Unit, Product Line, Brand** in addition to the basic segments. Custom segments allow introduction of a **new dimension for internal accounting** without convoluting the existing ones. Consider a manufacturing company with Departments for functional groups (R&D, Production, Sales) and product lines as Classes. If they also want to track **Brands**, they could create a Brand segment. Each sales transaction could then be tagged with Brand (in addition to Class=product line), allowing a P&L by Brand. Or, a company might treat Department as the high-level division and use a custom "Cost Center" segment for the specific team or project within that division that incurred an expense. Because you can filter segments by each other or by department, you could even enforce that each cost center is tied to a specific division department (the Concentrus example of dependent segmentation). The benefit here is a multi-dimensional reporting matrix: you can analyze expenses by Department *and* Cost Center, or revenues by Product Line *and* Sales Channel, etc. It provides finer granularity for internal analysis and accountability. In practice, one NetSuite user described segments as mainly used to provide additional "buckets" for P&L reporting – with Departments as cost centers, Classes as revenue streams, and any further buckets implemented as custom segments. Common extra buckets include things like **Channel** (e.g., Ecommerce, Retail, Wholesale), **Campaign** (for marketing spend attribution), or **Fund** (in nonprofit, see next use case). By carefully designing these segments, companies can simplify their general ledger (fewer accounts) yet slice data in more ways. This *also* makes NetSuite's financial planning or allocation processes more powerful, since you can allocate or budget by these new segments as well (though native budgeting UI is limited to class/department/location, many companies export to Excel or use SuiteAnalytics for custom segment budgets).

Fund Accounting and Grants (Nonprofit)

Nonprofits and government organizations often need to track financials by **fund, grant, or program**. NetSuite's Class or Department is often repurposed for "Program" or "Fund", but when an organization needs to track multiple dimensions (e.g., Program *and* Funding Source), a custom segment is the answer. For example, a nonprofit could use Department for Program (Education, Research, Outreach, etc.), and introduce a custom segment called **Fund** for the specific grant or fund that money comes from. Every expense can then be tagged with both the Program and the Fund. This is essential for compliance and reporting to donors – they can produce statements of expense by fund across programs. Alphabold (a NetSuite partner) noted this scenario: biomedical companies and NPOs receiving grants needed to classify expenses against specific funds/grants, and segments fulfilled that need (Source: netsuite.alphabold.com)(Source: netsuite.alphabold.com). Another segment could even be used for **Donor Restrictions** (if funds are unrestricted vs restricted). The benefit of custom segments here is clear separation of concerns: your core financials remain intact, but you have additional reporting streams to satisfy external requirements. And with the custom segment balancing feature, nonprofits could even balance the books by fund if needed (ensuring net assets by fund roll up correctly). Using segments for fund accounting greatly enhances transparency and auditability in NetSuite's standard reports, which is a big win for the nonprofit sector.

Multi-Entity and Intercompany Dimensions

In a multi-subsidiary (OneWorld) environment, sometimes there's a need for an overarching classification that spans subsidiaries. For instance, suppose you have multiple legal entities, but you want to report by **Global Business Unit** that can include transactions from several subsidiaries. A custom segment can be created for “Business Unit” with values that apply across entities (and you can filter them by subsidiary so users only see relevant ones). Users in each subsidiary tag transactions with the appropriate Business Unit. Then corporate can run a consolidated P&L by Business Unit (the segment cuts across the legal entity boundaries). Another use case: if you have multiple companies sharing customers or projects, you might use a segment for “Global Project” or “Global Customer Group” to unify reporting. Essentially, custom segments can act as a bridge classification in consolidations. Additionally, if intercompany transactions need tracking by some project or campaign ID, a segment ensures the same code travels with both sides of the entry.

Real-World Example – Dependent Segments (Class by Department)

To illustrate a practical scenario: A company wants to restrict which **Class** values can be used under each **Department** (maybe certain product lines are only relevant to certain departments). Standard NetSuite doesn’t allow making Class depend on Department. The solution: create a custom segment “Sub-Class” or a custom Department that mimics Department and then filter a second segment by it. One approach (per a NetSuite solutions article) is to implement both Department and Class as custom segments (or one as custom) so you can set up the filtering logic. For example, define a custom segment “Dept” and a custom segment “ClassX” and use the Filtered By on ClassX = Dept. Then maintain the filter mappings (e.g. Dept=Sales allows ClassX A, B, C; Dept=Service allows ClassX D, E, etc.). Users selecting a Dept will then only see appropriate ClassX values. This ensures data cleanliness and meaningful combinations. Although this approach replaces native Department/Class for transactions, it showcases how custom segments can create **dependent dropdowns**, a capability many ERP users desire for better UI/UX. The trade-off is losing some native functionality (like employee restrictions by department, if you go fully custom), but some choose consistency and accuracy over those. Alternatively, one could keep native Department and make *Class* a custom segment filtered by Department – that way Department is still native but classes are custom (requires possibly not using native Class at all on transactions). In any case, the benefit is eliminating invalid category combinations and simplifying selection for users, which leads to better reporting (no need to wade through irrelevant classes per department).

These examples barely scratch the surface – other creative uses of custom segments include: **Orders** (tagging transactions that belong to a large customer contract or order grouping), **Compliance Codes** (to mark transactions for regulatory reporting, e.g. taxable/non-taxable beyond native flags), **Investor** (if sharing financials by investor or fund in real estate), **Equipment or Asset** (to tag maintenance costs to specific assets without creating endless expense accounts), and more. The key is that custom segments allow you to tailor NetSuite’s data categorization to exactly match how you want to analyze your business. They *simplify the chart of accounts* by absorbing what would otherwise be handled by duplicate accounts or clunky manual tracking. With segments, you maintain one unified accounting structure but gain endless reporting flexibility.

Reporting and Analytics with Custom Segments

Once you have custom segments populated with data, the real payoff comes from reporting and analysis. NetSuite's reporting tools fully support custom segments, especially when they are GL Impacting. Here's how you can leverage them:

- **Standard Reports (Financial Statements & Transaction Reports):** NetSuite allows customization of any standard report to include custom segment criteria. For example, on an Income Statement or Balance Sheet, you can go to Edit Layout and add your custom segment as a column (horizontal axis) or filter. The system treats a GL-impacting custom segment similarly to Class/Dept/Loc for financial reports. **Financial Report Builder:** If you open the Financial Report Builder, under Columns (for a columnar report) you'll find your custom segment listed as an available column dimension (assuming GL Impact was enabled). You can create, say, a "Income Statement by Project" by adding Project as the column and perhaps filtering to a specific year. Each project will get its own column in the report. You can likewise filter financial reports by segment (e.g., run a P&L *for one specific* Region by setting Region = EMEA in the filters). NetSuite Help confirms that *"standard reports can be customized to use custom segments as filters and columns… In financial reports, custom segments can be added as a horizontal dimension."*. This means you have near-first-class support in the native reporting engine. Some reports (like Sales by Item, Transaction Detail reports, etc.) also allow adding the custom segment as a grouping or column. For instance, a Sales by Customer report could be grouped by your "Channel" segment to see sales by channel. If you have a Saved Report that you want to reuse with different segment filters, you can create saved report versions for each segment value as needed.

- **Saved Searches:** Saved searches are very powerful with custom segments. When creating a transaction saved search (or a search on any record type that you applied the segment to), you will see the custom segment available as a field. For example, in a Transaction search, you might add Criteria: Project Segment = "Project Alpha" to pull all transactions for that project. Or create a summary search grouped by Project to get total revenue or expense per project. If the segment is on an associated record (like on customer), you can also do joins (e.g., an Invoice search can pull the Customer's Region segment via join). One advantage is that custom segment values, being a custom record, can store additional attributes you might use in searches (though typically values are just labels). Saved searches also respect the permissions: only roles with Search access to the segment will be able to use it in criteria or see the values in results. One thing to be mindful of: if your segment is multi-select, searching for a specific value requires the "contains" type criteria since the field may have multiple values. Saved searches can serve as the backbone for custom KPI's, reminders, or portlet metrics based on segments (e.g., create a KPI for Sales by Channel by having a saved search per Channel).

- **SuiteAnalytics Workbook / Analytics:** The newer Analytics Workbook interface treats custom segments as just another field in the data model. If GL Impacting, they appear in transaction records and GL datasets; if not GL, they appear in the record dataset they're applied to. You can drag a custom segment into a pivot table or chart to visualize data by that dimension. For example, using the Dataset builder, you can create a dataset of GL Balances with "Division" segment as a dimension to produce a Balance Sheet by Division. Oracle has also introduced a feature to create **Segment Hierarchy** filters in workbooks if the segment values are hierarchical (for rolling up). The bottom line is that your analytics capabilities expand in tandem with segments – any analysis you could do by class or department, you can now do by your custom segments.

- **Budgeting and Forecasting:** One question that arises is whether you can budget by custom segments. Out-of-the-box, NetSuite's budgeting tool (Setup > Accounting > Budget) allows budgets by Class, Department, Location (and Subsidiary) only. It does not directly support entering budgets by custom segment values via the UI. However, if you need a **Budget vs Actual by custom segment**, there are a couple of approaches:

  - **Workaround with existing fields:** Some companies map their custom segment values to classes or departments in the budgeting tool for entry, then use a report to compare (this is clunky and not dynamic).

  - **SuiteAnalytics / Excel:** You can export actuals by segment (since actuals carry the segment in GL) and then compare to an external budget keyed by the segment. Fusion CPA (an accounting firm) noted that custom segments can be leveraged in Excel via ODBC or Web Query to produce Budget vs Actual reports. Essentially, you maintain budgets in Excel by segment and use a NetSuite ODBC query to fetch actuals by segment.

  - **Planning and Budgeting module (NSPB):** If you have Oracle NetSuite Planning and Budgeting (the Adaptive Insights-based tool), custom segments can typically be modeled as custom dimensions in your planning cubes, allowing full budgeting and forecasting by those segments. That's an advanced solution beyond core NetSuite.

  Even without the native budgeting UI, many have successfully done budget vs actual analysis by exporting data. **Important:** Actuals will only be tracked by the segment if it is GL Impacting (otherwise, actual GL postings don't carry that info, so you'd have to rely on transactional saved searches rather than GL extract). A best practice mentioned is to **ensure custom segments are GL impacting if you plan to do budget vs actuals with them**. This ensures consistency in financial rollups.

- **Drilldown and Allocations:** On financial reports, when you drill into a total, if that report is segmented, you'll see the segment values on the individual lines. For instance, clicking on "Total Expenses" when viewing by Region will show each transaction line with its Region. This helps trace the data. If using NetSuite's Allocations feature, note that currently you can allocate by class/department/location. Allocations by custom segment might require a custom script or manual saved search approach to redistribute amounts by segment. However, with SuiteGL, one could write a Custom GL Lines plug-in to automatically allocate costs based on segment values (like allocate corporate overhead to divisions based on a driver). The plugin could read the segment on the source entry and create new entries accordingly.

In summary, custom segments unlock a new dimension in reporting that was previously unavailable. The ability to **group, filter, and compare** data by these tailor-made categories is immensely valuable. Many companies find after implementing custom segments that they can sunset numerous manual spreadsheets that were used to piece together data by product, project, region, etc. Now it's available at the click of a button inside NetSuite (Source: netfreak.co.uk). One UK NetSuite provider even touted that custom segments let you **"skyrocket your financial reporting"** by organizing and tracking information at the click of a button (Source: netfreak.co.uk). While that may be marketing hyperbole, there's truth that segments greatly enhance real-time reporting capabilities of NetSuite.

Integration Considerations for Custom Segments

Integrating NetSuite with other systems (such as procurement software, CRM, or data warehouses) requires special attention when custom segments are involved. Since custom segments are essentially custom records and fields, external systems need to handle them properly. Here are key points regarding integration:

- **External IDs and Custom Records:** Each custom segment's values reside in a custom record type (named after your segment). This means each value has an internal ID and can have an External ID (if you enable the External ID field on that custom record type, which NetSuite usually does by default). This is advantageous for integration: you can reference a segment value by external ID during CSV import or via SOAP/REST web services, which is often easier than using the internal ID or exact text name. This addresses a pain point that standard custom list fields have (where you can't easily use external IDs for list values in imports). Plan to assign external IDs to segment values if you'll reference them from outside systems – e.g., if your CRM or e-commerce has "Region" codes, set those codes as External IDs on the corresponding NetSuite segment values.

- **SOAP Web Services / SuiteTalk:** In SOAP web services, custom segment fields on records typically appear as **RecordRef** fields (similar to how Class or Department appear). The `scriptId` will be the custom segment's field ID (e.g., `cseg_region`). When creating or updating records via SOAP, you need to supply a RecordRef with the internalId (or externalId) pointing to one of the segment's custom record values. Also, remember that only GL-impacting segments reflect in the accounting context; if an external system is pulling transaction GL impacts, segment values come through if applicable. There is a known limitation in older API versions that the search by custom segment might require joins, but in newer versions, you can search transactions by segment as a field.

- **REST Web Services / RESTlet:** In REST API or RESTlet scripts, you set the segment field by its script ID, providing either an internal id of the value or perhaps a text if allowed. Ensure the integration user has permission to see and set that field (more on permissions below). In REST web services (NetSuite 2020+ REST API), custom segments show up as `custcol_xyz` or `custbody_xyz` fields (depending on context) in the records' JSON. You input the reference to the custom record instance. Consult the SOAP/REST schema to confirm the exact field identifiers.

- **CSV Import:** If you are importing transactions or records via CSV and need to include a custom segment, it will show up as a mappable field by the segment's label. You can provide the **internal ID or exact text** of the segment value for single-select segments. If external IDs are enabled for the values, you might also map External ID. For multi-select segments, you may provide multiple values separated by the appropriate delimiter. Keep in mind, if the segment's value list is very large, consider using internal IDs to avoid text mismatches.

- **Third-Party Integrations and Connectors:** Many integration platforms (like Celigo, Boomi, etc.) have adapters for NetSuite that can work with custom fields. A custom segment on a transaction will generally be exposed as a field in those connectors, which you can map from source data. Ensure the connector is using an API version that recognizes custom segments (most do). In some cases, you might have to manually configure the connector to fetch the custom record data for the segment values if needed for lookup. For example, a procurement system like Precoro can import custom segment values from NetSuite to use them in the procurement interface. In fact, Precoro's documentation provides a recipe for integrating custom segments:

    - First, in NetSuite, update the integration user role to have full access to Custom Segments and Custom Record Types. This typically means adding the "Custom Segments" permission (Full) and "Custom Record Types" permission (Full) to the role.

    - Then, ensure the role also has access to each specific segment's record type: under Permissions > **Custom Record**, add an entry for your custom segment (it will appear by its segment name) with Full access. This explicitly allows reading/writing the segment's values.

    - Additionally, Precoro notes to set the segment's **Default Record Access Level** to Edit for the integration (so that any role not specifically listed can at least edit the field). In practice, you might instead explicitly list your integration role with Edit access on the segment's Permissions subtab (Record Access Level) to be sure.

- After that, Precoro can connect and import the list of segment values (via API) and allow you to map them in their system.

This example underlines a common integration task: **synchronizing segment values** to external systems. If you want an external app to use the same categorization (e.g., an expense tool to tag expenses with NetSuite segments), you may need to periodically update that system with any new segment values from NetSuite. This can be automated by a scheduled RESTlet or using the external system's integration features (like Precoro's Import Custom Segments button). Conversely, if external systems manage the values (less common), you'd import them into NetSuite's custom record.

- **Data Warehouse / BI:** If you use SuiteAnalytics Connect (ODBC) to pull data into a warehouse or BI tool, be aware that custom segments will appear as additional tables or fields in the NetSuite Connect schema. Specifically, there will be a table for the segment's values (likely named something like CustomSegment_[ID]) and transaction or record tables will have an internal ID reference. You will need to join them to get the value names. For example, if you have a segment "ProjectCode", the transaction table might have a column `ProjectCode_ID` which holds the internal id referencing the CustomSegment_ProjectCode table. (NetSuite's schema documentation can confirm the exact structure, but this is the general idea.) Also note from NetSuite documentation: SuiteAnalytics Connect (ODBC) does not yet support the unified custom segment ID fully – meaning you might see separate fields per context (like one for body, one for line) rather than one unified field in ODBC. Just plan your queries accordingly.

- **Permissions Recap for Integration:** The integration user must have permission to see and edit custom segments on records. This means in addition to adding the "Custom Segments" global permission, you should verify the role's Record Access Level for each segment (on the segment definition) is set to Edit for that role (or at least default Edit for all roles). If you forget this, your integration might be able to read the transaction but not set the segment field (getting permission errors). One common troubleshooting step if a RESTlet can't set a segment: check that the role has the segment listed with Edit under Permissions subtab, or simply set the default access to Edit for the configuration period of integration.

- **Use Case: E-commerce Integration:** Imagine you have an e-commerce platform and you want to tag each sales order in NetSuite with a "Sales Channel" (Online vs Retail) custom segment. The integration pulling orders from the e-commerce will need to decide the segment value for each order (likely based on some attribute in the order source). You'd then map that to the internal ID of the "Online" or "Retail" value in NetSuite when creating the Sales Order via API. Storing those mapping (maybe via external IDs that match channel codes from the source) makes it easier. On the return side, if pushing data from NetSuite to a data warehouse, including the segment dimension means analysts can slice the sales data by channel easily in the BI tool – ensure the warehouse pulls the segment value name or external ID via the join.

In summary, integrating custom segments isn't difficult, but it requires proper setup of permissions and sometimes additional mapping logic. Document the internal IDs or external IDs of segment values for your integration developers, and consider building a simple lookup service (maybe a saved search JSON export) that external systems can call to fetch the latest segment values. The good news is, since segments are standard NetSuite records under the hood, all the integration mechanisms (SuiteTalk, REST, etc.) support them. Just treat a custom segment field similar to how you treat the Class field in integrations, with the nuance that the list of values is custom-maintained.

Security, Permissions, and Maintenance of Custom Segments

Security and Permissions Considerations

Custom segments introduce additional data governance considerations. We've already discussed the role-based permissions on segments, but let's put it in context:

- **Field-Level Visibility/Edit Control:** By using the Record Access Level settings, you can ensure that only appropriate users can set or even view a given segment's values on records. For example, you might have an "Internal Profitability Code" segment that only finance managers should assign; you can give line managers *View* access (so they see which code was assigned) but *None* for editing (they can't change it). Or if a segment is irrelevant to certain roles, you can hide it (None) for them to reduce screen clutter and confusion. This is more granularity than standard Class/Dept, which every user with record access will see by default.
- **Value Management Delegation:** Think about who is allowed to add or modify the list of segment values. It's usually best to centralize this with administrators or power users. By giving most roles no access to manage values, you prevent unauthorized expansion of the list. Some organizations implement an approval process for new segment values (e.g., a user requests a new Project code, and an admin creates it). Keeping control prevents duplication or erroneous values that could hurt reporting.
- **Search/Report Access:** If a segment contains sensitive information (for instance, a segment that tags the profitability rating of a customer or an internal risk code), you might restrict who can run reports on it. Typically, though, segments are used for fairly standard classifications that many people need to report on, so you'd grant reporting access widely. One scenario for limiting search access: perhaps a "Compliance" segment used by legal, which general staff shouldn't use in their saved searches as it could confuse or reveal too much.

- **Record Access vs. Record Restriction:** It's important to clarify that setting **Record Access = None** for a segment on a role will hide the segment field (and its value) on records for that role, but it does *not* restrict access to the *record itself*. In other words, unlike the native "Restrict by Department" or "Restrict by Class" features that limit which records a user can see based on record classification, custom segments do not natively provide record-level security. If you need to restrict, say, transactions for a certain segment value from being seen by a role, you would have to implement a custom solution (perhaps separate roles and saved search-driven center tabs, or script logic to throw an error if they try to load a disallowed record). The custom segment permissions only govern the field visibility/editability, not the entire record visibility. Standard segments still have an edge here: e.g., you can restrict customers by subsidiary, department, etc., via role settings. So if data security by classification is a requirement, consider using standard segments for that purpose if possible, or be prepared to script it with custom segments.

- **Auditing and Change Management:** Custom segments themselves (the definitions) can be critical pieces of configuration. It's wise to restrict the ability to edit segment definitions to Admins only, or use NetSuite's Employee Access Control to log changes. If you have SuiteCloud Change Management or SDF deployment, treat segments as configuration to be deployed carefully across environments. Also, if multiple admins exist, coordinate changes to segments (for example, don't have two people editing the values list simultaneously to avoid save conflicts).

- **Hidden Segment Data Exposure:** As noted earlier, a segment marked Hidden is not secure from a tech-savvy user – they could inspect the web page or use the ODBC connection if they have access to find out the value. Similarly, if a role has No Access to a segment (so it's hidden), that role could still potentially see the segment's value via an indirect route (like if a saved search is created by someone else that exposes the value in the results, and that search is shared). However, if the role lacks Search access for the segment, they shouldn't be able to create a search on it or see it in search results. So to truly hide a segment's info from a role, set Record Access = None *and* Search/Reporting = None for that role. Then even if they somehow get to a search result including that field, it might show as blank (or they wouldn't be able to add it at all). Testing role perspectives is important to ensure your permissions are doing what you expect.

- **Segmentation vs. Segregation:** Don't confuse segment security with NetSuite's general segregation of duties. If you need to ensure a user can only enter certain segment values, that's a more granular control than NetSuite natively provides (except by splitting roles, as mentioned). One workaround for enforcement is via **Workflow or Script**: e.g., a Client Script that defaults or locks a segment for certain roles, or a User Event that throws an error if an unauthorized change is attempted. For instance, some companies use scripting to automatically set a "Business Unit" segment based on the user's role or department, thereby preventing them from picking a wrong unit.

In summary, use the custom segment permissions to align with your organization's data governance: who should fill in these fields, and who should see them. The permissions model is quite robust (more so than normal custom fields which don't have their own subtab for permissions). This ensures custom segments can be introduced without losing control.

Maintenance and Ongoing Management

After initial setup, you will need to maintain custom segments over time as your business evolves. Maintenance tasks include updating segment values, managing inactive values, adjusting segment settings, and monitoring performance. Here are guidelines and tips:

- **Adding New Segment Values:** As new categories arise (new projects, regions, cost centers, etc.), you'll add values to the segment. This can be done by editing the custom segment (Customization > Custom Segments > Edit) and adding lines on the Values subtab. Alternatively, you can go to **Lists > Custom > {Your Segment Record}** (the custom record type) and add a new record there – adding a new record in that list is equivalent to adding a value. Ensure you set any parent or filter criteria for the new value if applicable. For consistency, fill in an External ID if you use them for integration. Also consider using a naming convention for values (especially if the segment is used across subsidiaries – e.g., include a code or prefix to avoid name collisions). If you have to add many values at once (say 100 new cost centers), you can import them via CSV into the custom record type (just get the internal ID of the custom record type by looking at the URL when you view the list of values, or use the Import Assistant which will list it). Remember that adding values does not generally affect historical data – it just expands what can be used going forward.

- **Inactivating or Deleting Values:** When a value is no longer needed (e.g., a project completed, a department closed), the best practice is to **inactivate** it rather than delete (Source: docs.oracle.com). To do so, edit the segment and mark that value's Active box off (or edit the value record and check "Inactive"). Inactive segment values won't appear in dropdowns for users, preventing new usage, but **existing records tagged with them will retain them** (so historical reporting remains intact). Deletion of a value is possible (by deleting the custom record entry), but not recommended if it's already used on transactions – those transactions would then show a blank (the link to the deleted value breaks). It's similar to deleting a Class that's on transactions – not a good idea since it orphan references. If you *must* eliminate a value entirely (say it was created by mistake and used only on a few records), you'd want to clean those records (reassign them to a correct value or remove the segment) before deletion. NetSuite does provide a way to mass update or global replace classification fields, but for custom segments you might need to use CSV export/edit/import or a script to change a batch of records from one segment value to another. Once no records reference the value, deletion is safer. But again, usually **Inactive = Yes** is the way to retire a segment value.

- **Deleting a Custom Segment:** What if a custom segment itself is no longer needed? Maybe you set one up experimentally or your business merged it with another dimension. NetSuite allows deletion of custom segments *only under strict conditions*. If the segment was never used on any transaction or record, you can likely delete it straight away. However, if it has GL Impact and has been used in posted transactions, deletion is locked by default. To override this, an administrator must enable **Allow GL Custom Segment Deletion** under **Setup > Accounting > Accounting Preferences > General > General Ledger**. This preference acknowledges that deleting a GL-impacting segment can mess with historical financial reports. After enabling, you could go to the custom segment and choose Delete. NetSuite will attempt to remove the segment and presumably also remove the references from transactions (which might result in those transactions losing that classification). This could have significant reporting implications – e.g., you'll no longer be able to filter those historical transactions by that segment because it's gone. Therefore, Oracle's documentation (and SuiteRep blog) rightly suggests using the **Inactive** flag on the segment if you think you might want it again or you want to preserve history, rather than deletion. Only delete a custom segment if you are absolutely sure it was a mistake or you've migrated to a different solution and you've archived the info elsewhere. Additionally, note that to delete a segment, you'd likely need to delete all its values first (the system might handle that, but typically you can't have orphan custom records floating around). In summary, deletion is possible but approach with caution – and likely only in a sandbox/test environment first to see effects.

- **Changing Segment Configurations:** Many settings of a custom segment can be modified after creation (especially for non-GL-impact segments). For example, you can add or remove record types it's applied to (Application & Sourcing) at any time. You can change the Label if needed (though be careful, as that will change it everywhere and could confuse users – better to do such changes during off-hours and communicate it). You can change default values, add new filtered-by relationships, etc. One thing you cannot change is the GL Impact flag (as noted) and the segment ID. Also, if you initially made it single-select, you cannot simply switch it to multi-select if the unified ID is used (the Type field might be locked after values exist). If you truly need to change from single to multi-select or vice versa, you might have to create a new segment and migrate data (since those are fundamentally different field types in NetSuite's eyes). Before making any big change, consider the impact on SuiteScripts, saved searches, or integrations – if those refer to a segment or value name that you plan to alter, update them accordingly.

- **Performance and Scalability:** Custom segments inherently add joins in reporting (to resolve the value names) and can add additional fields on records, which might have some performance overhead. In most cases, a handful of custom segments won't be noticeable in performance. But if you create a large number of segments or segments with extremely many values, you might see slower form loads or reporting queries. The Slack archive excerpt earlier indicated that there have been instances of up to 15 million custom record entries without major issues, but performance *may* degrade as numbers grow. If you anticipate, for example, a segment that could get thousands of new values every year (like a "Transaction ID" segment which would be ill-advised design), consider whether that should be a segment at all. Typically, segment values are relatively static sets (like a few dozen projects or cost centers at a time). If your segment value list becomes huge (many thousands), you might find the dropdown list is not user-friendly; in such cases, alternative UI like a popup search or using an existing record field might be better. Keep an eye on usage: NetSuite's Record page and search pages may show the count of values – if it becomes unwieldy, maybe it's time to archive some (mark old ones inactive) or split into multiple segments if conceptually different.

- **Documentation and Training:** Maintain documentation for each custom segment: what it represents, what the expected values are (and their definitions), who owns it (business-wise), and how it's used in reports. This is important because years later, new administrators might find an obscure segment and not know its purpose. Also, document any relationships – e.g., "Segment X is filtered by Department; see filter setup" so that if Department values change, you know to update filters. Train users on using the segments: e.g., how to choose the right Project code, or how the Region is auto-assigned from customer but can be changed if necessary. Clear guidelines help maintain data quality. One consultant on a forum emphasized getting the segmentation design right up front because it's painful to change later if done incorrectly. This implies maintaining consistency and not repurposing a segment for something else without thorough analysis. If you do need to repurpose a segment (maybe after deleting values), consider the impact on historical data and consider starting a fresh new segment if the meaning is substantially different.

- **Monitoring Usage:** It's a good idea to periodically review how segments are being used. Run a saved search to count how many transactions have "(Empty)" for a required segment to catch any that slipped through (maybe via custom form that didn't enforce it). Or review if users are overusing a generic "Other" value, indicating you might need more specific values. If you have dependent filters, verify that new base values (like new Department) have been accounted for in segment filters (so new department isn't inadvertently excluded from all segment values). If a segment was supposed to be on all transactions and you find transactions without it, maybe mandatory wasn't fully enforced – you might update forms to fix that.

- **Inter-segment Relationships and Changes:** If you restructure one segment (like merge or rename departments), how does it affect custom segments filtered by department? You'd need to update filter mappings for values that were tied to the old department. Similarly, if you inactivate a parent in a hierarchy, all its children might become orphan unless you reparent them. So treat changes in the "filter by" fields or hierarchical structure carefully.

- **New Features and Upgrades:** Stay informed on NetSuite release notes. Oracle often enhances custom segment capabilities (for example, the balancing by segment feature introduced in 2020.1 (Source: netsuite.alphabold.com), or any new search or reporting functions). Also, SuiteAnalytics may bring more support for segments (like multi-select handling improvements). If a new release offers something like "budget by custom segment" or better filtering, you want to take advantage of it. Keep an eye on user community forums as well, since best practices evolve as the feature matures.

In summary, maintaining custom segments is not very labor-intensive day-to-day, but it requires *discipline* and planning. By governing who can create values, regularly cleaning up unused values, and monitoring how the segments are filled in, you ensure the data remains reliable. Many companies treat the introduction of a new custom segment as a mini-project – considering reporting needs, training, and documentation – because it becomes a fundamental part of the ERP data model. As long as you manage them proactively, custom segments will continue to deliver value throughout their lifecycle.

Conclusion

Custom segments are a powerful feature in NetSuite that enable organizations to tailor the system's data segmentation to their unique business requirements. By understanding the differences between custom segments and standard segments, you can choose the right tool for each classification need. Custom segments allow you to introduce new dimensions – whether for projects, regions, cost centers, funds, or any other category – and to do so in a way that integrates seamlessly with NetSuite's GL, reporting, and scripting frameworks.

In this guide, we covered how to enable and configure custom segments in depth, including best practices like carefully deciding on GL impact, setting up dynamic sourcing, and leveraging filtering for dependent dropdowns. We explored several industry-specific use cases demonstrating the versatility of segments: from tracking project profitability without proliferating accounts, to simplifying multi-regional reporting, to meeting nonprofit fund accounting needs, among others. Real-world tips such as using segments to simplify the Chart of Accounts and avoiding common pitfalls (like improper segment design or unchecked GL impact) were highlighted throughout.

We also delved into reporting and analytics, emphasizing that once segments are in place, NetSuite's native reporting can be extended to use them as if they were built-in dimensions. Whether you're customizing an income statement by a new segment or building a saved search for operational analysis, segments provide the flexibility to "slice and dice" enterprise data in practically unlimited ways. SuiteGL and SuiteScript considerations were addressed – showing that custom segments are not just for static reports but can drive automation (via Custom GL plugins) and should be accounted for in any integrations or scripts that handle record data.

Security and maintenance form the backbone of sustainable segment usage. By using role permissions on segments, administrators can ensure proper control and data integrity. And through regular maintenance – adding new values as needed, inactivating or retiring old ones, and reviewing usage – the segment continues to serve the organization's needs as it grows. It's worth reiterating the advice from experienced NetSuite professionals: invest time in designing your segmentation strategy up front, as redesign later can be complex. Consider not just current reporting needs but also future scalability when deciding on what segments to create and how to structure their values.

In conclusion, custom segments empower NetSuite administrators and ERP consultants to configure the system in a way that **mirrors the business** – rather than forcing the business to fit into predefined categories. They enhance the ability to categorize data and meet unique reporting and analysis needs. When properly implemented, custom segments can dramatically improve financial insight (for example, enabling management to view real-time P&Ls by any dimension of interest), all while keeping the system cleaner and more user-friendly than alternative approaches (like overly complex account coding). For professionals tasked with financial reporting and ERP customization, mastering custom segments is essential. With the guidance and best practices outlined in this report, you can confidently deploy custom segments to unlock new analytical capabilities in NetSuite, driving better decision-making and organizational performance.

Sources: The information in this report was compiled from official NetSuite documentation, expert blogs, and community forums. Key references include Oracle's Help Center on custom segments, SuiteGL documentation, implementation insights from NetSuite solution providers (Source: netsuite.alphabold.com), and practical experiences shared by the NetSuite user community. These sources (cited throughout) provide further reading for those looking to deepen their understanding of custom segment functionality and best practices.

Tags: netsuite, custom segments, erp configuration, financial reporting, suitegl, netsuite administrator, standard segments, suitescript

About Houseblend

HouseBlend.io is a specialist NetSuite™ consultancy built for organizations that want ERP and integration projects to accelerate growth—not slow it down. Founded in Montréal in 2019, the firm has become a trusted partner for venture-backed scale-ups and global mid-market enterprises that rely on mission-critical data flows across commerce, finance and operations. HouseBlend's mandate is simple: blend proven business process design with deep technical execution so that clients unlock the full potential of NetSuite while maintaining the agility that first made them successful.

Much of that momentum comes from founder and Managing Partner **Nicolas Bean**, a former Olympic-level athlete and 15-year NetSuite veteran. Bean holds a bachelor’s degree in Industrial Engineering from École Polytechnique de Montréal and is triple-certified as a NetSuite ERP Consultant, Administrator and SuiteAnalytics User. His résumé includes four end-to-end corporate turnarounds—two of them M&A exits—giving him a rare ability to translate boardroom strategy into line-of-business realities. Clients frequently cite his direct, “coach-style” leadership for keeping programs on time, on budget and firmly aligned to ROI.

End-to-end NetSuite delivery. HouseBlend’s core practice covers the full ERP life-cycle: readiness assessments, Solution Design Documents, agile implementation sprints, remediation of legacy customisations, data migration, user training and post-go-live hyper-care. Integration work is conducted by in-house developers certified on SuiteScript, SuiteTalk and RESTlets, ensuring that Shopify, Amazon, Salesforce, HubSpot and more than 100 other SaaS endpoints exchange data with NetSuite in real time. The goal is a single source of truth that collapses manual reconciliation and unlocks enterprise-wide analytics.

Managed Application Services (MAS). Once live, clients can outsource day-to-day NetSuite and Celigo® administration to HouseBlend’s MAS pod. The service delivers proactive monitoring, release-cycle regression testing, dashboard and report tuning, and 24 × 5 functional support—at a predictable monthly rate. By combining fractional architects with on-demand developers, MAS gives CFOs a scalable alternative to hiring an internal team, while guaranteeing that new NetSuite features (e.g., OAuth 2.0, AI-driven insights) are adopted securely and on schedule.

Vertical focus on digital-first brands. Although HouseBlend is platform-agnostic, the firm has carved out a reputation among e-commerce operators who run omnichannel storefronts on Shopify, BigCommerce or Amazon FBA. For these clients, the team frequently layers Celigo’s iPaaS connectors onto NetSuite to automate fulfilment, 3PL inventory sync and revenue recognition—removing the swivel-chair work that throttles scale. An in-house R&D group also publishes “blend recipes” via the company blog, sharing optimisation playbooks and KPIs that cut time-to-value for repeatable use-cases.

Methodology and culture. Projects follow a “many touch-points, zero surprises” cadence: weekly executive stand-ups, sprint demos every ten business days, and a living RAID log that keeps risk, assumptions, issues and dependencies transparent to all stakeholders. Internally, consultants pursue ongoing certification tracks and pair with senior architects in a deliberate mentorship model that sustains institutional knowledge. The result is a delivery organisation that can flex from tactical quick-wins to multi-year transformation roadmaps without compromising quality.

Why it matters. In a market where ERP initiatives have historically been synonymous with cost overruns, HouseBlend is reframing NetSuite as a growth asset. Whether preparing a VC-backed retailer for its next funding round or rationalising processes after acquisition, the firm delivers the technical depth, operational discipline and business empathy required to make complex integrations invisible—and powerful—for the people who depend on them every day.
